

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ ” _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
комп'ютерних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Методика та програмне забезпечення формування груп учасників
А/В тестів з мінімальним взаємним впливом

Виконав студент IV курсу, групи IT-91
(шифр групи)

Хільчук Олени Валеріївни
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник доцент, к.т.н., доц., Крамар Ю. М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Консультант з графічної документації ст.викл., Головченко М. М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Ткач М. М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
комп'ютерних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Хільчук Олені Валеріївні
(прізвище, ім'я, по батькові)

1. Тема проєкту Методика та програмне забезпечення формування груп
учасників А/В тестів з мінімальним взаємним впливом

керівник проєкту Крамар Юлія Михайлівна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31» травня 2023 р. №2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни,
опис предметної області, аналіз вимог, огляд ринку програмних продуктів,
постановка задачі.

2) Моделювання та конструювання програмного забезпечення: опис розробленої
методики, розробка архітектурного рішення, опис структури бази даних, опис
засобів розробки.

3) Аналіз якості та тестування програмного забезпечення: аналіз якості
розробленого програмного забезпечення, опис процесів тестування та тестових
прикладів.

4) Впровадження та супровід програмного забезпечення: опис процесів
розгортання та підтримки програмного забезпечення.

5. Перелік графічного матеріалу

- 1) Схема структурна варіантів використань _____
- 2) Схема структурна компонентів програмного забезпечення _____
- 3) Схема бази даних _____
- 4) Схема структурна діяльності _____
- 5) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв'язання задачі	17.03.2023	
3	Постановка та формалізація задачі	26.03.2023	
4	Розробка інформаційного забезпечення	03.04.2023	
5	Алгоритмізація задачі	10.04.2023	
6	Обґрунтування вибору використаних технічних засобів	15.03.2023	
7	Розробка програмного забезпечення	22.04.2023	
8	Налагодження програми	13.05.2023	
9	Виконання графічних документів	25.05.2023	
10	Оформлення пояснювальної записки	27.05.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент

(підпис)

Олена ХІЛЬЧУК

(ініціали, прізвище)

Керівник

(підпис)

Юлія КРАМАР

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 54 таблиці, 21 рисунок та 19 джерел – загалом 73 сторінки.

Дипломний проєкт присвячений формуванню методики для розподілу учасників А/В тестів на групи, так щоб взаємний вплив учасників різних груп був мінімальним, а також створенню відповідного програмного забезпечення.

Мета проєкту полягає у підвищенні валідності А/В тестів у системах, де присутній взаємний вплив користувачів, шляхом формування тестових груп таким чином, щоб учасники мали мінімальну взаємодію.

Об'єкт дослідження: програмне забезпечення для проведення А/В тестів у мультикористувацьких сервісах.

Предмет дослідження: методика формування груп учасників А/В тестів.

У розділі аналізу вимог до програмного забезпечення було досліджено предметну область та існуючі альтернативи створюваного сервісу, визначено сценарії використання компонентів системи та сформульовано технічні вимоги до програмного забезпечення.

Розділ моделювання та розробка програмного забезпечення присвячений аналізу обраних засобів розробки, проєктуванню та створенню сховища даних і формуванню архітектурного рішення для розроблюваного сервісу.

У розділі про якість та тестування програмного забезпечення описано процеси тестування, а також наведено основні контрольні приклади.

КЛЮЧОВІ СЛОВА: МЕТОДИКА, ВЕБ-ЗАСТОСУНОК, А/В ТЕСТ, СПЛІТУВАННЯ, ASP.NET CORE, REACT, MYSQL, RABBITMQ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, МІКРОСЕРВІС.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 53 tables, 23 figures and 14 sources – in total 71 pages.

The purpose of the diploma project is to formulate a methodology for dividing A/B test participants into groups so that the mutual influence of participants in different groups is minimal, as well as to develop the corresponding software.

The goal of the project is to increase the validity of A/B tests in systems where there is a mutual influence of users by forming test groups in such a way that participants have minimal interaction.

Research object: software for performing A/B tests in multi-user services.

Subject of research: the method of forming groups of A/B test participants.

In the section of software requirements analysis, the subject area and existing alternatives to the service being developed were investigated, scenarios for using system components were identified, and technical requirements for the software were formulated.

The section on software modeling and development is dedicated to analyzing the selected development tools, designing and creating a data warehouse, and forming an architectural solution for the service being developed.

The section on software quality and testing describes the testing processes and provides basic test cases.

KEYWORDS: METHODOLOGY, WEB APP, A/B TEST, SPLITTING, ASP.NET CORE, REACT, MYSQL, RABBITMQ, CLIENT-SERVER ARCHITECTURE, MICROSERVICE.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

Методика та програмне забезпечення формування груп учасників А/В тестів

з мінімальним взаємним впливом

Технічне завдання

КПІ.IT-9127.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юлія КРАМАР

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олена ХІЛЬЧУК

Київ – 2023

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	6
4.1	Вимоги до функціональних характеристик	6
4.1.1	Функції користувачького інтерфейсу	6
4.1.2	Для користувача:	8
4.1.3	Для адміністратора системи (якщо він передбачений):	14
4.1.4	Додаткові вимоги:	15
4.2	Вимоги до надійності	15
4.3	Умови експлуатації	15
4.3.1	Вид обслуговування	15
4.3.2	Обслуговуючий персонал.....	15
4.4	Вимоги до складу і параметрів технічних засобів	16
4.5	Вимоги до інформаційної та програмної сумісності	16
4.5.1	Вимоги до вхідних даних	16
4.5.2	Вимоги до вихідних даних	17
4.5.3	Вимоги до мови розробки	17
4.5.4	Вимоги до середовища розробки.....	17
4.5.5	Вимоги до представленню вихідних кодів	17
4.6	Вимоги до маркування та пакування.....	17
4.7	Вимоги до транспортування та зберігання	18
4.8	Спеціальні вимоги	18
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	19
5.1	Попередній склад програмної документації.....	19
5.2	Спеціальні вимоги до програмної документації	19
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	20
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	21

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Методика та програмне забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення Web-додатки, клієнти для доступу до Internet-ресурсів [045440], котра використовується для розподілу користувачів системи на групи для проведення А/В тесту та призначена для менеджерів та аналітиків, які працюють над розробкою та підтримкою продуктів, що розраховні на велику кількість користувачів, та включають взаємодію та, відповідно, взаємний вплив користувачів на досвід використання продукту одне одним, а також для багатокористувацьких систем, де присутній ризик зменшення прибутку внаслідок безпосередньої модифікації певного функціоналу без попередньої перевірки.

					КПІ.IT-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки Методики та програмного забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом є завдання на дипломне проектування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для формування груп учасників А/В тесту, враховуючи взаємний вплив між учасниками та мінімізуючи цей вплив між учасниками з різних груп.

Метою розробки є підвищення валідності результатів А/В тестів, в результаті створення сервісу, який формує групи із урахуванням взаємного впливу учасників з різних груп на користувацький досвід та зводить його до мінімуму.

					КПІ.IT-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

- можливість реєстрації;
- можливість входу в акаунт;
- отримання переліку А/В тестів (рисунок 4.2);
- отримання інформації про окремий А/В тест та відображення його графу (рисунок 4.1);
- керування станом А/В тесту;
- створення А/В тесту (рисунок 4.3);
- отримання розподілу учасників на групи в А/В тесті;
- отримання переліку груп конкретного учасника (рисунок 4.4).

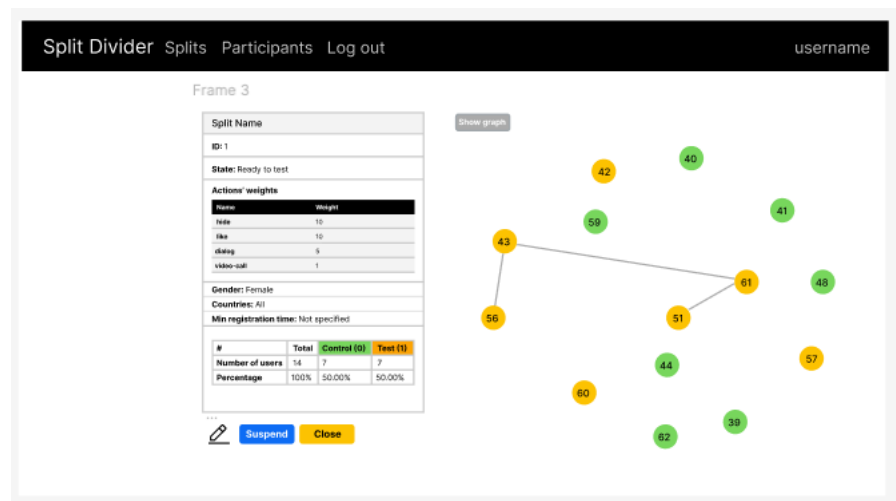


Рисунок 4.1 – Прототип сторінки для відображення повної інформації про А/В тест

Split Divider
Splits
Participants
Log out
username

State

Frame 1

Id	Name	State

Activate
Suspend
Close

Рисунок 4.2 – Прототип сторінки для відображення списку А/В тестів

Create new split

Name

Interactions' weights

Name	Weight
dialog	1
like	1
hide	1
video-call	1

Countries

Ukraine
Poland
USA
Great Britain
Germany

Gender

Male
Female
All

Minimal Registration Time

Рисунок 4.3 – Прототип сторінки для створення А/В тесту

Split Divider
Splits
Participants
Log out
username

Search

Split id	Group

Рисунок 4.4 – Прототип сторінки для відображення інформації про групи учасника

На рисунку 4.2 відображено прототип сторінки із переліком А/В тестів. При натисканні на рядок певного А/В тесту користувач потрапляє на сторінку з повною інформацією про цей тест (прототип даної сторінки наведено на рисунку 4.1).

Також на сторінці із переліком знаходиться кнопка для створення нового А/В тесту. Після натискання на неї, відображається модальне вікно (рисунок 4.3) для заповнення інформації про новий А/В тест.

На рисунку 4.4 наведено прототип сторінки з інформацією про групи учасника. На дану сторінку можна потрапити, натиснувши на кнопку «Participants» у хедері сайту.

4.1.2 Для користувача:

4.1.2.1 Авторизація в системі:

4.1.2.1.1 Реєстрація в системі:

4.1.2.1.1.1 Запит повинен містити електронну пошту користувача:

4.1.2.1.1.1.1 Якщо користувач із такою електронною поштою уже зареєстрований, то виникає помилка.

4.1.2.1.1.2 Запит повинен містити пароль користувача, який складається із принаймні 6 символів та обов'язково містить принаймні 1 цифру та 1 літеру:

4.1.2.1.1.2.1 Якщо пароль не відповідає вимогам, то виникає помилка.

4.1.2.1.1.3 Запит повинен містити підтвердження паролю користувача, яке співпадає із введеним паролем:

4.1.2.1.1.3.1 Якщо значення паролю та

підтвердження паролю не співпадають –
виникає помилка.

4.1.2.1.2 Вхід в систему:

4.1.2.1.2.1 Запит повинен містити електронну пошту
zareestrovanoogo korystuvacha:

4.1.2.1.2.1.1 Якщо користувача із такою
електронною поштою не знайдено, то
виникає помилка.

4.1.2.1.2.2 Запит повинен містити пароль користувача,
який складається із принаймні 6 символів та
обов'язково містить принаймні 1 цифру та 1
літеру:

4.1.2.1.2.2.1 Якщо пароль не відповідає
вимогам, то виникає помилка.

4.1.2.1.2.2.2 Якщо введений пароль не
відповідає збереженому паролю – виникає
помилка.

4.1.2.2 Отримання інформації про створені А/В тести:

4.1.2.2.1 Отримання повного переліку А/В тестів:

4.1.2.2.1.1 Запит може містити статус (у вигляді
числового номеру), із яким потрібно отримати
А/В тести:

4.1.2.2.1.1.1 Якщо фільтр статусу не надіслано –
повертається масив об'єктів із всіма
створеними А/В тестами, де об'єкти
містять назву А/В тесту у вигляді рядка,
ідентифікатор у вигляді числа та статус у
вигляді числа.

4.1.2.2.1.1.2 Якщо фільтр статусу надіслано –

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

повертаються тільки ті тести, які мають статус, що надіслано в запиті, у вигляді, описаному вище.

4.1.2.2.2 Отримання детальної інформації конкретного A/B тесту:

4.1.2.2.2.1 Запит має містити ідентифікатор A/B тесту у вигляді числа:

4.1.2.2.2.1.1 Дані повертаються у вигляді об'єкту, який містить інформацію про назву A/B тесту у вигляді рядка, його статус у вигляді числа, розподіл ваги між типами взаємодії у вигляді словника, країни сегментації у вигляді масиву, стать учасників у вигляді рядка та мінімальну дату реєстрації учасників у вигляді рядка.

4.1.2.2.2.1.2 Якщо A/B тесту із таким ідентифікатором не існує, повертається повідомлення про відсутність даних.

4.1.2.2.3 Отримання розподілу учасників на групи в конкретному A/B тесті:

4.1.2.2.3.1 Запит має містити ідентифікатор A/B тесту у вигляді числа:

4.1.2.2.3.1.1 Дані повертаються у вигляді масиву пар «ключ-значення», де ключем є ідентифікатор учасника, а значення – номер групи учасника в даному A/B тесті.

4.1.2.2.3.1.2 Якщо A/B тесту із таким ідентифікатором не існує, повертається повідомлення про відсутність даних.

4.1.2.2.4 Отримання графу, за яким відбувся розподіл в

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		10

конкретному А/В тесті:

4.1.2.2.4.1 Запит має містити ідентифікатор А/В тесту у вигляді числа:

4.1.2.2.4.1.1 У відповідь надсилається файл із розширенням .graphml, який містить побудований граф взаємодії для даного А/В тесту.

4.1.2.2.4.1.2 Якщо А/В тесту із таким ідентифікатором не існує, повертається повідомлення про відсутність даних.

4.1.2.3 Керування А/В тестами:

4.1.2.3.1 Створення А/В тесту:

4.1.2.3.1.1 Запит повинен містити назву А/В тесту у вигляді рядка із мінімальною довжиною 5 символів та максимальною – 30:

4.1.2.3.1.1.1 Якщо значення не відповідає вимозі, то виникає помилка.

4.1.2.3.1.2 Запит повинен містити словник із налаштуваннями ваги взаємодії для типів, де ключ – це рядок назви виду взаємодії, а значення – його базова вага:

4.1.2.3.1.2.1 Якщо значення назви взаємодії не коректне (тобто не підтримується застосунком) – виникає помилка.

4.1.2.3.1.2.2 Якщо значення ваги для виду взаємодії менше 1 або більше 30 – виникає помилка.

4.1.2.3.1.2.3 Якщо не надіслано базове значення ваги для якогось із видів взаємодії – виникає помилка.

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		11

4.1.2.3.1.3 Запит може містити стаття учасників A/B

тесту у вигляді рядка:

4.1.2.3.1.3.1 Якщо рядок має значення відмінне від одного з двох підтримуваних («male» або «female») – виникає помилка.

4.1.2.3.1.4 Запит може містити масив країн реєстрації учасників у вигляді ідентифікаторів цих країн:

4.1.2.3.1.4.1 Якщо країни з таким ідентифікатором не існує – виникає помилка.

4.1.2.3.1.5 Запит може містити мінімальний час реєстрації учасників у вигляді рядка формату «MM/dd/yyyy HH:mm»:

4.1.2.3.1.5.1 Якщо дані надіслано в некоректному форматі, то виникає помилка.

4.1.2.3.2 Редагування A/B тесту:

4.1.2.3.2.1 Запит може містити нову назву A/B тесту у вигляді рядка із мінімальною довжиною 5 символів та максимальною – 30:

4.1.2.3.2.1.1 Якщо значення не відповідає вимозі, то виникає помилка.

4.1.2.3.2.2 Запит може містити словник із новими налаштуваннями ваги взаємодії для типів, де ключ – це рядок назви виду взаємодії, а значення – його базова вага:

4.1.2.3.2.2.1 Якщо значення назви взаємодії не коректне (тобто не підтримується застосунком) – виникає помилка.

4.1.2.3.2.2.2 Якщо значення ваги для виду

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		12

взаємодії менше 1 або більше 30 – виникає помилка.

4.1.2.3.2.3 Якщо не надіслано базове значення ваги для якогось із видів взаємодії – виникає помилка.

4.1.2.3.2.3 Запит може містити статі учасників A/B тесту у вигляді рядка:

4.1.2.3.2.3.1 Якщо рядок має значення відмінне від одного з двох підтримуваних («male» або «female») – виникає помилка.

4.1.2.3.2.4 Запит може містити масив країн реєстрації учасників у вигляді ідентифікаторів цих країн:

4.1.2.3.2.4.1 Якщо країни з таким ідентифікатором не існують – виникає помилка.

4.1.2.3.2.5 Запит може містити мінімальний час реєстрації учасників у вигляді рядка формату «MM/dd/yyyy HH:mm»:

4.1.2.3.2.5.1 Якщо дані надіслано в некоректному форматі, то виникає помилка.

4.1.2.3.3 Видалення A/B тесту:

4.1.2.3.3.1 Запит повинен містити ідентифікатор A/B тесту, який потрібно видалити.

4.1.2.3.3 Зміна стану A/B тесту:

4.1.2.3.3.1 Запит повинен містити ідентифікатор A/B тесту, для якого потрібно змінити стан:

4.1.2.3.3.1.1 Якщо A/B тесту з таким ідентифікатором не знайдено – виникає помилка.

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		13

4.1.2.3.3.2 Якщо неможливо здійснити перехід із поточного стану А/В тесту в цільовий стан – виникає помилка.

4.1.2.4 Отримання інформації про групи учасників А/В тестів:

4.1.2.4.1 Отримання інформації про А/В тести, в яких бере участь конкретний учасник::

4.1.2.4.1.1 Запит повинен містити ідентифікатор учасника, для якого потрібно отримати інформацію про групи, у вигляді числа:

4.1.2.4.1.1.1 Дані повертаються у вигляді масиву пар «ключ-значення», де ключем є ідентифікатор А/В тесту, а значення – номер групи учасника в даному А/В тесті.

4.1.2.4.1.1.1 Якщо такого учасника не знайдено – повертається порожній масив

4.1.3 Для адміністратора системи (якщо він передбачений):

4.1.3.1 Зміна прав користувача:

4.1.3.1.1 Видалення прав користувачу:

4.1.3.1.1.1 Запит повинен містити електронну пошту зареєстрованого користувача, для якого потрібно змінити права:

4.1.3.1.1.1.1 Якщо користувача із такою електронною поштою не знайдено – виникає помилка.

4.1.3.1.1.2 Запит повинен містити ідентифікатор ролі, яку потрібно видалити, у вигляді рядка:

4.1.3.1.1.2.1 Якщо такої ролі не знайдено – виникає помилка.

4.1.3.1.2 Додавання прав користувачу:

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		14

4.1.3.1.2.1 Запит повинен містити електронну пошту зареєстрованого користувача, для якого потрібно змінити права:

4.1.3.1.2.1.1 Якщо користувача із такою електронною поштою не знайдено – виникає помилка.

4.1.3.1.2.2 Запит повинен містити ідентифікатор ролі, яку потрібно додати, у вигляді рядка:

4.1.3.1.2.2.1 Якщо такої ролі не знайдено – виникає помилка.

4.1.4 Додаткові вимоги:

– інформація щодо учасників передається повідомленнями з іншого сервісу через RabbitMQ.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		15

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на Macintosh-сумісних персональних комп'ютерах.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;
- жорсткі накопичувачі з швидким виконанням операцій читання і

запису;

- встановлено .NET 7.0.0;
- встановлено СУБД MySQL 8.0.0;
- встановлено Node 18.16.0.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i9;
- об'єм ОЗП: 32 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
- жорсткі накопичувачі з швидким виконанням операцій читання і

запису;

- встановлено .NET 7.0.23;
- встановлено СУБД MySQL 8.0.32;
- встановлено Node 18.16.0.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix та підтримувати браузері Google Chrome та Safari.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі:

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		16

- налаштування A/B-тесту, а саме, назва у вигляді рядка та набір значень ваги взаємодії (цілого числа) для кожного типу взаємодії (у вигляді рядка) у форматі словника;

- критерії потрапляння у тест, якщо такі потрібні, а саме стать учасників у вигляді рядка, мінімальна дата реєстрації у форматі MM/dd/yyyy HH:mm, та перелік ідентифікаторів країн користувачів у вигляді масиву.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:

- значення групи кожного учасника тесту, яку він отримав в результаті розбиття графу;

- файл із побудованим графом взаємодії користувачів, які відповідають критеріям A/B тесту та беруть участь у ньому, у форматі .graphml, та браузерна сторінка, де відображається даний граф.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C#, її фреймворк ASP.NET Core. Розробку клієнту виконати на мові програмування JavaScript та за допомогою бібліотеки React.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі .NET за допомогою середовища JetBrains Rider, та за допомогою середовища JetBrains WebStorm.

4.5.5 Вимоги до представлення вихідних кодів

Вихідний код програми має бути представлений у вигляді набору файлів у форматі .cs та набору файлів у форматі .js та .css.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		17

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

					КПІ.ІТ-9127.045440.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		18

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна діяльності;
- схема структурна компонентів програмного забезпечення;
- схема бази даних;
- креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03	
2.	Розробка технічного завдання	26.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	03.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	15.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	22.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	13.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	20.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	25.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	27.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

					КПІ.ІТ-9127.045440.01.91	Арк.
						21
Змін.	Арк.	№ докум.	Підп.	Дата.		

Пояснювальна записка до дипломного проєкту

на тему: Методика та програмне забезпечення формування груп
учасників А/В тестів з мінімальним взаємним впливом

КПІ.ІТ-9127.045440.02.81

Київ – 2023

ЗМІСТ

ВСТУП 5

1	АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	7
1.1	Загальні положення	7
1.2	Змістовний опис і аналіз предметної області	8
1.3	Аналіз існуючих технологій та успішних ІТ-проектів	9
1.3.1	Аналіз відомих алгоритмічних та технічних рішень	9
1.3.2	Аналіз допоміжних програмних засобів та засобів розробки	11
1.3.3	Аналіз відомих програмних продуктів	12
1.4	Аналіз вимог до програмного забезпечення	19
1.4.1	Розроблення функціональних вимог	27
1.4.2	Розроблення нефункціональних вимог	30
1.5	Постановка задачі	31
	Висновки до розділу	32
2	МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	33
2.1	Моделювання та аналіз програмного забезпечення	33
2.2	Архітектура програмного забезпечення	34
2.3	Конструювання програмного забезпечення	36
2.4	Аналіз безпеки даних	47
	Висновки до розділу	48
3	АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 50	
3.1	Аналіз якості ПЗ	50
3.2	Опис процесів тестування	52
3.3	Опис контрольного прикладу	63
	Висновки до розділу	67
4	ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.	68
4.1	Розгортання програмного забезпечення	68

4.2	Підтримка програмного забезпечення	69
	Висновки до розділу	69
	ВИСНОВКИ	70
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	72
	ДОДАТОК А ЗВІТ ПОДІБНОСТІ	74
	ДОДАТОК Б ОПИС АРІ СЕРВЕРНОЇ ЧАСТИНИ	75
	ДОДАТОК В ОПИС КЛЮЧОВИХ КОМПОНЕНТІВ СЕРВЕРНОЇ ЧАСТИНИ	

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс
SDK	– Software development kit
IT	– Інформаційні технології
ER	– Entity-Relation diagram
OC	– Операційна система.
БД	– База даних.
СУБД	– Система управління базами даних.
Спліт	– Альтернативна назва A/B тесту
JSON	– JavaScript Object Notation, текстовий формат обміну даними
UX	– User Experience – користувацький досвід
ORM	– Object Relation Mapping, спосіб інтуїтивної взаємодії із реляційними базами даних використовуючи об'єктно-орієнтовані парадигми
HTTPS	– Hypertext Transfer Protocol Secure - захищена версія протоколу HTTP, що використовується для передачі даних між сервером і клієнтом.
XML	– Extensible Markup Language - мова розмітки для зберігання та обміну структурованою інформацією між різними системами.

ВСТУП

У наш час із невпинною діджиталізацією суспільства зростає потреба у додатках, які надають можливість взаємодії з іншими людьми через мережу Internet. Такі сервіси охоплюють різноманітні сфери нашого життя: від комунікації та знайомств, до освіти та ігор. Надзвичайно популярними подібні додатки стали внаслідок пандемії COVID-19 та, зокрема в Україні, після початку повномасштабного вторгнення. Підтримка контакту в реальності стала важчою, відповідно зріс попит на застосунки, які надають таку можливість віртуально.

Природним чином, кількість користувачів таких платформ з часом тільки зростає, тому додавання нового функціоналу чи модифікація існуючого може призвести до збитків для бізнесу.

A/B тестування (також відоме як спліт-тестування) – це метод порівняння двох версій функціоналу, щоб визначити, який з них є оптимальним [1]. Для проведення тесту користувачі діляться на 2 групи: контрольну, для якої зміни не впроваджуються, та тестову, для якої функціонал змінюється. Після проведення тесту відбувається аналіз даних, на підставі якого приймається рішення, чи варто поширювати зміни на всіх користувачів.

У системах, які включають взаємодію користувачів, якщо функціонал безпосередньо стосується цієї взаємодії, то результати тестування не можна вважати цілком валідними, адже можуть виникати ситуації, коли користувачі різних груп впливають на користувацький досвід та поведінку одне одного. Відповідно виникає потреба в такому розподілі учасників тесту на групи, щоб взаємний вплив між групами був мінімальним.

Нині серед продуктів, які зіштовхуються із такою проблемою, присутня тенденція ізолювати групи так, аби зводити саму взаємодію між групами до мінімуму. Але подібне рішення у певних системах може бути занадто ресурсозатратним. Наприклад, якщо програмний продукт має мікросервісну

					КПІ.IT-9127.045440.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

архітектуру, то всі компоненти екосистеми повинні знати, який користувач у якій групі, або постійно робити запити за цими даними до відповідного мікросервісу. Це, у свою чергу, може призвести до зростання часу відповіді, навантаження мережі та погіршення досвіду користувачів продукту. Тож якщо система з тих чи інших причин не може надати можливість ізолювати групи, виникає потреба у пошуку іншого рішення та розгляді проблеми під іншим кутом.

Альтернативним рішенням є враховувати взаємодію вже на етапі формування груп, що й було досліджено у даній роботі.

Метою створення цієї роботи є підвищення валідності А/В тестів у системах із взаємним впливом користувачів шляхом формування груп із врахуванням взаємного впливу між групами та зведенням його до мінімуму.

Для досягнення поставленої мети вирішуються такі задачі: детальний аналіз існуючих рішень та предметної області, розробка методики формування груп, формулювання функціональних та нефункціональних вимог, розробка високорівневої архітектури продукту, вибір засобів для розробки, дизайн інтерфейсу користувача, розробка серверної та клієнтської складових.

Результатом є сформульована методика для формування тестованих груп відповідно до умов поставленої задачі, та веб-застосунок, який надає можливість створювати А/В тести, запускати їх, керувати їхнім станом, а також отримувати інформацію про те, до яких груп було розподілено учасників.

					КПІ.IT-9127.045440.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

У продуктах, де прибуток напряму залежить від того, наскільки задоволені користувачі користуванням ним, додавання чи модифікація функціоналу може привести до значних збитків. Тому важливо попередньо провести тестування на певному сегменті користувачів, щоб мати дані для прийняття фінального рішення. Нині, з цією метою, все поширенішим стає А/В тестування.

А/В-тестування, також відоме як спліт-тестування, - це стандартний спосіб оцінити залученість або задоволеність користувачів новою послугою, функцією або продуктом [2]. Для проведення такого тесту як правило потрібно дві групи користувачів – контрольна та тестова. Цільові метрики тестової групи порівнюються із відповідними показниками контрольної групи, і на підставі даного порівняння приймається рішення – поширювати функціонал на всіх користувачів чи ні.

Найбільш поширеним способом поділу користувачів на групи є за допомогою рандомізації. Однак механізми рандомізації, що використовуються в А/В тестах, можуть бути досить складними і можуть бути причиною невалідності результатів [3], особливо якщо тестування проводиться у багатокористувацьких системах, які включають взаємодію користувачів між собою.

Дана робота напрямлена на пошук альтернативного способу формування груп спліт тесту, таким чином, щоб взаємний вплив користувачів між групами був мінімальним.

Окрім цього, програмне забезпечення має в цілому надавати підтримку та засоби для проведення А/В тестів. Тобто, виникає потреба у тому, щоб надати інструменти для створення та керування А/В тестами, їхніми життєвими циклами, тощо.

1.2 Змістовний опис і аналіз предметної області

А/В-тестування - це науковий метод, що ґрунтується на даних і застосовується в масштабі для визначення успішності різних варіацій товару чи послуги [4]. Випадковим чином розподіляючи користувачів між контрольною та основною групами, А/В-тестування дозволяє порівняти дві або більше версій продукту, веб-сайту чи маркетингової кампанії, щоб визначити, яка з них працює краще.

У ІТ-індустрії А/В-тестування стало важливим інструментом для оптимізації цифрових продуктів та послуг. Компанії з ІТ сектору все частіше застосовують цей науковий метод для тестування різних варіантів своїх пропозицій та отримання цінної інформації про поведінку та вподобання користувачів.

Проведення спліт тесту включає в себе багато етапів:

- визначення мети експерименту та цільових метрик;
- розробка гіпотези, яка буде перевірятися;
- імплементація гіпотези – тобто написання відповідного функціоналу для неї;
- формування груп учасників тесту;
- збір та аналіз даних;
- прийняття рішення.

При проведенні А/В тестів важливим є те, яким саме чином поділяються користувачі на групи та наскільки даний спосіб підходить для продукту. Якщо при класичному варіанті А/В тесту рандомізований розподіл є цілком задовільним варіантом, то зі зростаючою популярністю багатокористувацьких платформ, які надають можливість взаємодії, приходить і потреба в іншому способі формування груп учасників.

Існують різні альтернативні варіанти формування груп, окрім звичайного рандомізованого. Наприклад, сегментування користувачів залежно від статі, країни чи дати реєстрації. Це може бути корисно для

					КПІ.ІТ-9127.045440.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		8

проведення спліт тестів, що перевіряють функціонал, який може по-різному впливати на різних користувачів залежно від перерахованих вище характеристик. Однак, дане рішення не підходить для вирішення поставленої проблеми.

Оскільки метою є мінімізація взаємного впливу між учасниками різних груп, то альтернативним рішенням є ізоляція користувачів у межах груп, зводячи цим контакт з іншою групою до мінімуму. Але даний спосіб має ряд недоліків. Він може бути надто ресурсоємним у деяких системах, наприклад, у платформах із мікросервісною архітектурою, що може спричинити повільний час відповіді, збільшення навантаження на мережу та погіршення UX користувачів.

Відповідно, виникає потребу у пошуку іншого методу формування груп учасників спліт тесту та розробці відповідного програмного забезпечення.

1.3 Аналіз існуючих технологій та успішних ІТ-проектів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації програмного забезпечення для формування груп учасників А/В тестів з мінімальним взаємним впливом. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

При дослідженні існуючих рішень було виділено наступні способи формування груп учасників А/В тестів.

Рандомізація – учасників неупереджено розділяють у групи випадковим чином. Це найбільш поширений, класичний підхід до проведення спліт тестів. Алгоритми рандомізації варіюються та впливають на складність і швидкість даного способу. Однак, в нього є вирішальний недолік – взаємні впливи користувачів ніяк не враховуються та не можуть бути відслідкованими.

Сегментація – учасників розподіляють на групи залежно від певних характеристик (стать, країна, час реєстрації, тощо). Проте, знову таки, взаємодія користувачів при формуванні груп не враховується.

Стратифікована вибірка – спосіб, який передбачає поділ користувачів на невеликі підгрупи, де користувачі пов'язані між собою, та випадковий розподіл користувачів у цих підгрупах на тестовані групи. Даний підхід найбільше відповідає висунутим вимогам, але у високонавантажених багатокористувацьких системах реалізація є досить складною, адже коректно поділити користувачів на підгрупи досить складно, як і визначити точні критерії розподілу. Відповідно, потрібно зберігати дуже велику кількість інформації про діяльність користувача та його характеристики, обробити інформацію про всіх користувачів та розподілити на підгрупи. Не всі продукти можуть забезпечити необхідні технічні засоби.

Таким чином приходимо до висновку, що необхідно розробити альтернативний спосіб формування груп учасників спліт тесту.

Оскільки розроблюваний сервіс є інструментом для проведення тестів у великих продуктових системах, доцільно розглянути варіанти його реалізації.

Модульний – написання окремого модуля, який міститиме всю логіку та алгоритми. Але визначальний мінус: завантаження даних, які потім будуть оброблюватися, займатиме надто багато ресурсів, тому підключення модуля може виявитися досить складним. Окрім того, демонстрація результатів дослідження теж може бути проблематичною.

Написання мікросервіса із додатковим моделюванням екосистеми – рішення, яке одразу продемонструє, яким чином дане програмне забезпечення може поводитися та взаємодіяти з іншими сервісами у системі. Оскільки при мікросервісній архітектурі кожен сервіс є незалежним, містить власну логіку та не сильно зв'язаний з іншими, то це рішення є найбільш влучним для демонстрації роботи «в природному середовищі».

Таким чином було прийнято рішення розробки мікросервісу та додавання окремого компоненту, який буде генерувати дані, для моделювання обміну повідомленнями та взаємодії між мікросервісами цільової системи.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

При дослідженні програмних засобів було розглянуто наступні фреймворки, за допомогою яких здійснюється розробка веб-застосунків на мові програмування C#:

- ASP.NET MVC – фреймворк для створення веб-додатків за шаблоном Model-View-Controller [15]. ASP.NET MVC існує вже кілька років і має розвинену екосистему сторонніх інструментів та бібліотек;
- ASP.NET Core Web API – фреймворк для створення HTTP-сервісів з використанням архітектурного стилю RESTful [16]. Він побудований на основі .NET Core. ASP.NET Core Web API оптимізований для продуктивності та масштабованості і часто використовується для створення мікросервісів та API;
- NancyFX – фреймворк, який підкреслює простоту і модульність, популярний для створення RESTful API та мікросервісів [17].

Таким чином, оскільки розроблюване програмне забезпечення буде функціонувати як мікросервіс, то доцільно розглядати ASP.NET Core Web API та NancyFX фреймворки. Було обрано перший із перерахованих, оскільки він є офіційно підтримуваним компанією Microsoft, надає багато функціоналу «із коробки» та має широку популярність.

Також важливим рішенням перед початком роботи є вибір середовища для розробки. Було розглянуто декілька варіантів:

- Visual Studio – IDE від Microsoft для розробки C#. Підтримує широкий спектр платформ, включаючи Windows, macOS та Linux, а також добре інтегрується з іншими технологіями Microsoft, такими як Azure та SQL Server [18];
- Visual Studio Code – кросплатформенний кодовий редактор. Він добре

налаштовується і включає багато розширень для розробки на C#, включаючи підтримку IntelliSense, налагодження та навігацію коду [19];

- JetBrains Rider – кросплатформенна IDE, розроблена спеціально для розробки .NET. Включає в себе багато функцій для розробки C#. Також включає розширені інструменти рефакторингу та інтеграцію з іншими інструментами JetBrains, такими як ReSharper [6].

Оскільки JetBrains Rider має велику кількість вбудованого функціоналу та різноманітних плагінів, було обрано це середовище розробки.

1.3.3 Аналіз відомих програмних продуктів

При дослідженні готових програмних продуктів, які реалізують проведення А/В тестів, було сформульовано список критеріїв, за якими проводився аналіз:

- сервіс повинен враховувати взаємодії користувачів при формуванні груп учасників тесту;
- сервіс повинен надавати інформацію про взаємодію між користувачами (тобто дані, на основі яких відбувався поділ) у тому чи іншому вигляді, та візуалізувати ці дані у доступному вигляді;
- А/В тести повинні бути гнучкими та мати ряд налаштувань для сегментації;
- враховується те, що для різних тестів різні види взаємодії можуть мати різну вагу;
- А/В тест має певний життєвий цикл, надається можливість запускати тест, перезапускати та закінчувати його.

Було розглянуто декілька рішень, які в тій чи іншій мірі відповідають висунутим критеріям та можуть вважатися аналогами пропонованого рішення. Варто зазначити, що дані рішення є незалежними зовнішніми інструментами, з якими програмні продукти можуть робити інтеграцію. Але оскільки вони є досить популярними, та враховуючи те, що продукти із користувацькою

взаємодією як правило не надають вихідний код, то розглянемо ці знайдені рішення:

- **Google Optimize**

Безкоштовний інструмент А/В-тестування, який інтегрується з Google Analytics. Він дозволяє користувачам створювати і запускати А/В тести на своїх веб-сайтах. Google Optimize також включає функції персоналізації, які дозволяють показувати різний контент різним групам користувачів (рисунок 1.1).

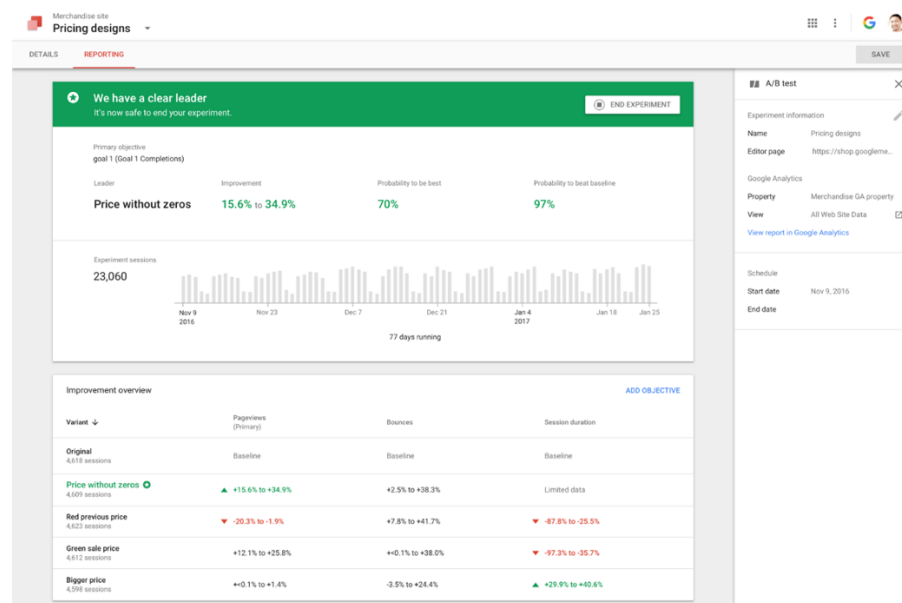


Рисунок 1.1 – Візуалізація результатів А/В тесту в Google Optimize

Google Optimize використовує статистичний алгоритм, щоб гарантувати, що розподіл на тестові групи є випадковим і рівномірно розподіленим між різними користувачами.

Серед переваг можна виділити:

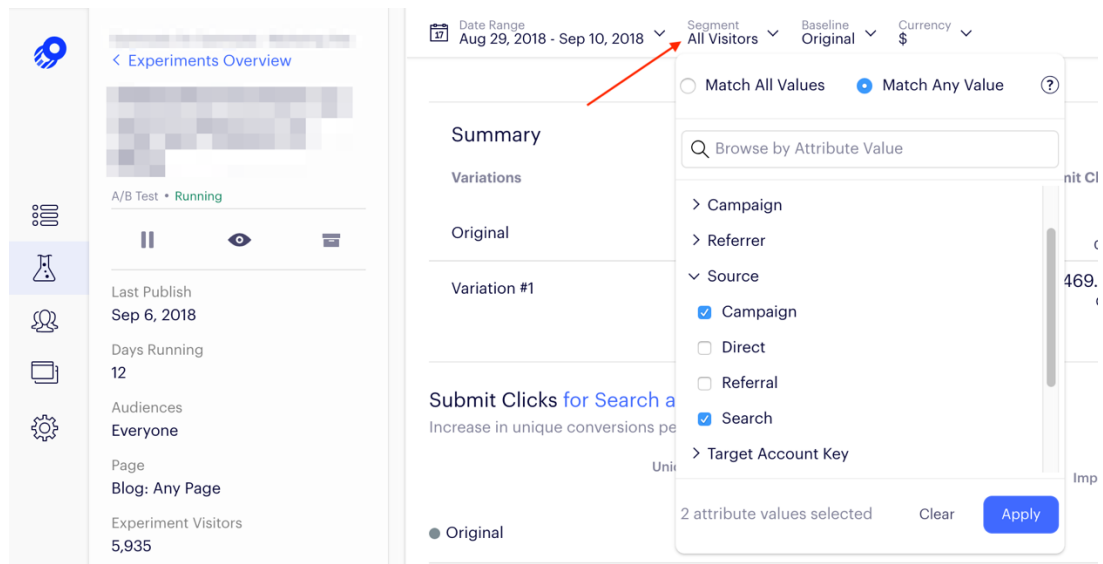
- засоби візуалізації для аналізу даних;
- надаються засоби для керування життєвим циклом А/В тесту;
- наявність невеликої кількості засобів сегментації.

Серед недоліків можна виділити:

- відсутність засобів для врахування взаємодії користувачів;
- відсутність засобів впливу на спосіб формування груп учасників спліт тесту.

– Optimizely

Популярний інструмент для А/В-тестування, який пропонує широкий спектр функцій, включаючи А/В-тестування, персоналізацію та аналітику. Optimizely пропонує інтеграцію з іншими інструментами, такими як Google Analytics та Adobe Analytics (рисуюнок 1.2).



Рисуюнок 1.2 – налаштування сегментації в Optimizely

Optimizely дозволяє користувачам визначати критерії виключення. Наприклад, користувачі можуть виключити користувачів, які вже здійснили конверсію або відвідували сайт протягом останніх 24 годин. Даний продукт використовує техніку бакетування, яка заснована на рандомізації, для розподілу користувачів між групами А/В тесту.

Серед переваг можна виділити:

- візуалізація даних для проведення аналізу А/В тесту;
- є засоби для керування життєвим циклом А/В тесту;
- надається можливість сегментувати користувачів.

Серед недоліків можна виділити:

- відсутність засобів для врахування взаємодії користувачів;
- дуже обмежені дані щодо користувачів та відсутність засобів інтеграції чи додавання нових даних;

- високі тарифи за повний функціонал.

- **VWO (Visual Website Optimizer)**

Ще один популярний інструмент для А/В-тестування, який пропонує широкий спектр функцій, включаючи А/В-тестування, багатовимірне тестування та персоналізацію. VWO, аналогічно до попередніх рішень, пропонує інтеграцію з іншими інструментами, такими як Google Analytics та Adobe Analytics (рисунок 1.3).

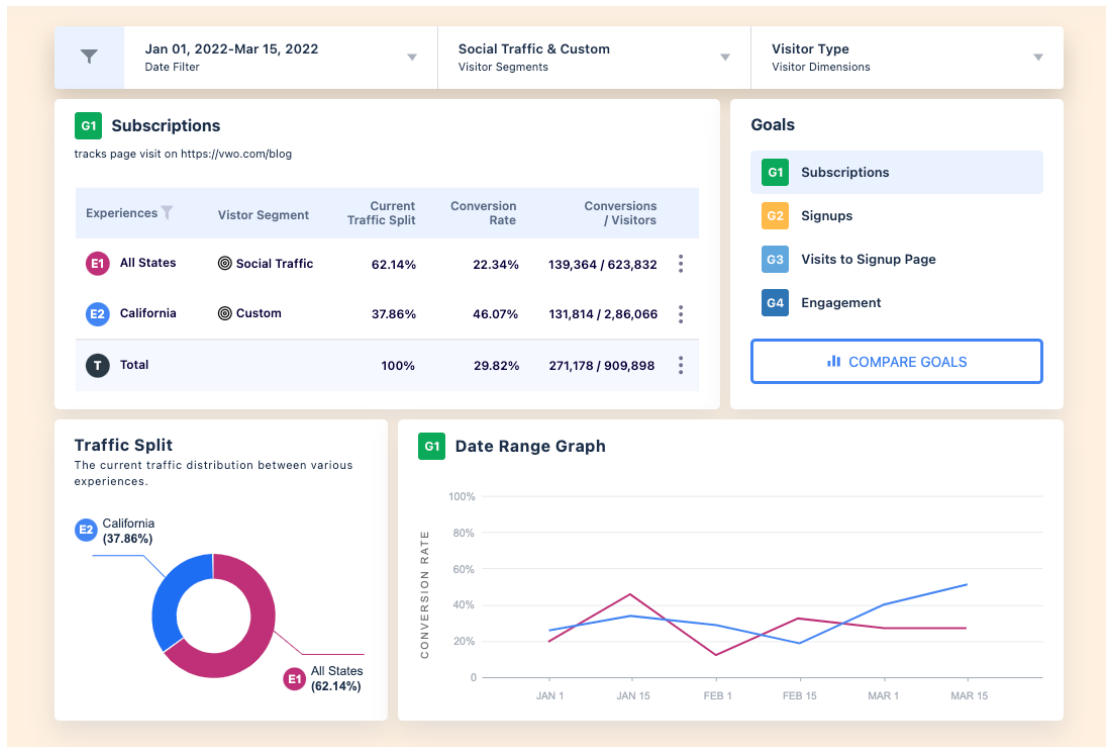


Рисунок 1.3 – Візуалізація даних А/В тесту в VWO

VWO використовує техніку розподілу трафіку, яка використовує рандомізацію, щоб забезпечити розподіл користувачів на тестовані групи.

Серед переваг можна виділити:

- візуалізація даних для проведення аналізу А/В тесту;
- є засоби для керування життєвим циклом А/В тесту;
- широкий спектр засобів сегментації;
- доступні засоби інтеграції.

Серед недоліків можна виділити:

- відсутність засобів для врахування взаємодії користувачів;
- відсутність налаштувань розподілу користувачів;
- високі тарифи підписки.

Із наданих результатів аналізу приходимо до висновку, що різні програмні продукти надають різноманітний функціонал для проведення А/В тестів, але жоден із них не враховує взаємодію між користувачами при формуванні груп або не надає інтеграційні можливості для цього. Також варто зазначити, що процес отримання доступу до програм також є досить ускладненим. Коротке порівняння згаданих програмних продуктів із пропонованим рішенням можна переглянути у таблиці 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проект (SplitDivider)	Google Optimize	Optimizely	VWO	Пояснення
Отримання переліку всіх А/В тестів та інформації про конкретний тест	Так	Так	Так	Так	Всі аналоги надають можливість переглядати створені А/В тести та інформацію про них
Отримання інформації про тестові групи конкретного учасника	Так	Ні	Ні	Ні	Жоден із розглянутих аналогів не надає інформацію про те, в які тестові групи потрапили конкретні учасники

Продовження таблиці 1.1

Функціонал	Дипломний проект (SplitDivider)	Google Optimize	Optimizely	VWO	Пояснення
Керування життєвим циклом A/B тесту	Так	Так	Так	Так	Всі програмні рішення надають можливість керувати життєвим циклом A/B тесту
Налаштування сегментації A/B тесту	Так	Так	Так	Так	Всі програмні засоби в тій чи іншій мірі надають можливість сегментувати учасників тесту
Формування груп учасників із врахуванням взаємодії між ними	Так	Ні	Ні	Ні	Аналоги використовують різні методи, засновані на рандомізації
Налаштування ваги видів взаємодії при	Так	Ні	Ні	Ні	Розглянуті аналоги не враховують

Продовження таблиці 1.1

Функціонал	Дипломний проект (SplitDivider)	Google Optimize	Optimizely	VWO	Пояснення
формуванні груп					взаємодію, тому, відповідно, не надають такого функціоналу
Візуалізація даних, на основі яких відбувався розподіл	Так	Ні	Так	Ні	Тільки один аналог надає можливість переглядати інформацію щодо учасників тесту, на основі якої відбувається розподіл
Візуалізація статистичних результатів А/В тесту	Ні	Так	Так	Так	Для дипломного проекту цей функціонал не є цільовим

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є формування груп учасників А/В тесту із врахуванням взаємного впливу користувачів та налаштування формування цих груп, більше функцій можна побачити на діаграмі варіантів використання, яку наведено на кресленні "Схема структурна варіантів використання".

В таблицях 1.2 - 1.11 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 – Варіант використання UC-01

Use case name	Вхід в систему
Use case ID	UC-01
Goals	Авторизація користувача в системі
Actors	Неавторизований користувач (гість)
Trigger	Користувач бажає авторизуватися
Pre-conditions	-
Flow of Events	Користувач заходить на сайт. Він автоматично потрапляє на сторінку з авторизацією, де йому потрібно ввести дані входу (пошту та пароль). Користувач вводить дані та натискає кнопку входу. Система шукає профіль користувача. При успішній авторизації користувач потрапляє на головну сторінку.
Extension	Якщо авторизація не відбулася, відображається повідомлення про помилку. Якщо якісь дані введено неправильно (не пройдено валідацію) – з'являється відповідне повідомлення, кнопка підтвердження стає неактивною.
Post-Condition	Користувача авторизовано в системі.

Таблиця 1.3 – Варіант використання UC-02

Use case name	Створення нового акаунту
Use case ID	UC-02
Goals	Створення нового користувача в системі
Actors	Неавторизований користувач (гість)
Trigger	Користувач бажає створити новий акаунт
Pre-conditions	-
Flow of Events	Користувач заходить на сайт. Він автоматично потрапляє на сторінку з авторизацією, де є кнопка реєстрації. Користувач натискає кнопку, після чого потрапляє на сторінку для створення нового акаунту. В поля форми реєстрації користувач вводить дані: – пошту; – пароль; – підтвердження паролю. Після введення даних користувач натискає кнопку підтвердження. Система реєструє нового користувача. Після цього автоматично відбувається вхід. Користувач переходить на головну сторінку.
Extension	Якщо реєстрація не відбулася, відображається повідомлення про помилку. Якщо якісь дані введено неправильно (не пройдено валідацію) – з'являється відповідне повідомлення, кнопка підтвердження стає неактивною.
Post-Condition	Створення нового акаунту, користувача автоматично авторизовано в системі.

Таблиця 1.4 – Варіант використання UC-03

Use case name	Перегляд створених А/В тестів
Use case ID	UC-03
Goals	Отримання переліку всіх створених А/В тестів
Actors	Авторизований користувач

Продовження таблиці 1.4

Trigger	Користувач бажає переглянути список А/В тестів
Pre-conditions	Користувача авторизовано в системі
Flow of Events	Користувач заходить на сторінку, де відображаються всі створені А/В тести. На сторінці розміщено коротку інформацію (назву, ідентифікатор) кожного тесту. При натисканні на назву певного тесту, користувач переходить на сторінку, де відображається повна інформація про даний тест.
Extension	При натисканні на назву конкретного тесту, користувач переходить на сторінку із повною інформацією про тест. Повна інформацію включає — назву; — ідентифікатор; — налаштування сегментації та ваги взаємодій; — актуальний статус. інформація, щодо розподілу, якщо тест запущено або закрито
Post-Condition	-

Таблиця 1.5 – Варіант використання UC-04

Use case name	Редагування створеного А/В тесту
Use case ID	UC-04
Goals	Редагування інформації щодо конкретного А/В тесту
Actors	Авторизований користувач
Trigger	Користувач бажає змінити інформацію щодо А/В тесту
Pre-conditions	Користувача авторизовано в системі. А/В тест знаходиться в станах «створено» або «призупинено».
Flow of Events	Користувач заходить на сторінку, де відображається повна інформація щодо тесту, та натискає на кнопку редагування. Він переходить на сторінку редагування, де є можливість змінити дані спліта (назва), налаштування сегментації спліта, налаштування ваги видів взаємодій. Користувач змінює необхідні дані та натискає кнопку

Продовження таблиці 1.5

	підтвердження. Система оновлює дані А/В тесту.
Extension	Якщо змінити дані неможливо, з'являється повідомлення про помилку. Якщо нові дані введено некоректно (не пройшли валідацію), кнопка підтвердження стає неактивною, відображається повідомлення про помилку.
Post-Condition	Інформацію щодо А/В тесту змінено

Таблиця 1.6 – Варіант використання UC-05

Use case name	Сворення нового А/В тесту
Use case ID	UC-05
Goals	Створення нового А/В тесту
Actors	Авторизований користувач
Trigger	Користувач бажає створити новий А/В тест
Pre-conditions	Користувача авторизовано в системі.
Flow of Events	Користувач заходить на сторінку, де відображається список всіх А/В тестів та натискає на кнопку для створення нового тесту. Користувач переходить на сторінку для створення. Там відображаються поля для інформації про тест: – назва; – налаштування сегментації (стать, країна та мінімальна дата реєстрації учасників); – налаштування ваги видів взаємодій. Користувач вводить інформацію та натискає кнопку підтвердження. Система оброблює запит та створює новий А/В тест. Користувач автоматично переходить на сторінку, де відображається повні інформація про щойно створений А/В тест.
Extension	Якщо створення не відбулося, відображається повідомлення про помилку.

Продовження таблиці 1.6

	Якщо якісь дані введено неправильно (не пройдено валідацію) – з’являється відповідне повідомлення, кнопка підтвердження стає неактивною.
Extension	Якщо створення не відбулося, відображається повідомлення про помилку. Якщо якісь дані введено неправильно (не пройдено валідацію) – з’являється відповідне повідомлення, кнопка підтвердження стає неактивною.
Post-Condition	Створення нового А/В тесту. Тест знаходиться у стані «створено».

Таблиця 1.7 – Варіант використання UC-06

Use case name	Активація А/В тесту
Use case ID	UC-06
Goals	Користувач хоче активувати А/В тест
Actors	Авторизований користувач
Trigger	Користувач бажає запустити А/В тест
Pre-conditions	Користувача авторизовано в системі. А/В тест створено. Його поточний стан має значення «створено» або «призупинено»
Flow of Events	Користувач заходить на сторінку, де відображається перелік всіх тестів. Він натискає на кнопку активації, яка знаходиться поруч із тестом, який бажає активувати. Система отримує запит, відбирає користувачів, які відповідають налаштуванням сегментації, формує граф взаємодії та зберігає цей граф у файлі з форматом .graphml. Після цього система формує групи учасників внаслідок розрізу графу, зберігає дані щодо групи кожного з учасників та переводить А/В тест у стан «готовий до тестування». Користувач потрапляє на сторінку з повною інформацією про А/В тест.
Extension	Якщо активація не відбулася, відображається повідомлення

Продовження таблиці 1.7

	про помилку.
Post-Condition	Тест знаходить в стані «готовий до тестування». Учасників розподілено між групами. Створено граф взаємодії між учасниками тесту, граф збережено у файлі з форматом .graphml. На сторінці з повною інформацією відображається граф взаємодії між учасниками.

Таблиця 1.8 – Варіант використання UC-07

Use case name	Перезапуск (призупинення) А/В тесту
Use case ID	UC-07
Goals	Користувач хоче призупинити та перезапустити А/В тест
Actors	Авторизований користувач
Trigger	Користувач бажає розпочати А/В тест наново
Pre-conditions	Користувача авторизовано в системі. А/В тест створено. Його поточний стан має значення «активний» або «готовий до тестування»
Flow of Events	Користувач заходить на сторінку, де відображається перелік всіх тестів. Він натискає на кнопку призупинення, яка знаходиться поруч із тестом, який бажає перезапустити. Система оброблює запит та видаляє дані про групи учасників тесту, який потрібно призупинити для перезапуску. А/В тест переходить в статус призупинено, після якого його можна повторно активувати. Користувач потрапляє на сторінку з повною інформацією про А/В тест.
Extension	Якщо призупинення не відбулося, відображається повідомлення про помилку.
Post-Condition	Тест знаходить в стані «призупинено». Дані про групи учасників видалено, щоб можна було провести розподіл заново.

Таблиця 1.9 – Варіант використання UC-08

Use case name	Зупинення A/B тесту
Use case ID	UC-08
Goals	Користувач хоче зупинити A/B тест
Actors	Авторизований користувач
Trigger	Користувач бажає закінчити A/B тестування
Pre-conditions	Користувача авторизовано в системі. A/B тест створено. Його поточний стан має значення «створено», «призупинено» або «готовий до тестування». Тест, який ще знаходить на етапі формування груп (стан між «активовано» та «готовий до тестування»), не можна зупиняти.
Flow of Events	Користувач заходить на сторінку, де відображається перелік всіх тестів. Він натискає на кнопку зупинення, яка знаходиться поруч із тестом, який бажає закінчити. Система оброблює запит та змінює статус A/B тесту на «закінчено». Подальші зміни тесту стають неможливими. Користувач потрапляє на сторінку з повною інформацією про A/B тест.
Extension	Якщо зупинення не відбулося, відображається повідомлення про помилку.
Post-Condition	Тест знаходить в стані «зупинено». Надалі інформацію про нього неможливо модифікувати, а сам тест неможливо перевести в інший стан.

Таблиця 1.10 – Варіант використання UC-09

Use case name	Перегляд тестових груп конкретного учасника
Use case ID	UC-09
Goals	Отримання інформації, до яких груп було розподілено учасника
Actors	Авторизований користувач
Trigger	Користувач бажає переглянути, до яких груп потрапив учасник у A/B тестах
Pre-	Користувача авторизовано в системі

Продовження таблиці 1.10

conditions	
Flow of Events	Користувач заходить на сторінку, де відображається інформація про учасників. У пошукове поле вводиться ідентифікатор учасника. Система виконує пошук та передає інформацію про те, до яких груп було розподілено учасника. Користувач переходить на сторінку, де відображається інформація про те, в яких тестах бере (та брав) участь учасник та до якої групи він належить (чи належав раніше).
Extension	Якщо при пошуку введені дані не проходять валідацію, повідомлення про це відображається.
Post-Condition	-

Таблиця 1.11 – Варіант використання UC-10

Use case name	Редагування прав користувача
Use case ID	UC-10
Goals	Змінити права доступу конкретного користувача
Actors	Авторизований користувач із правами адміністратора
Trigger	Користувач бажає додати або видалити права доступу іншого користувача системи
Pre-conditions	Користувача авторизовано в системі та він має права адміністратора
Flow of Events	Користувач заходить на сторінку керування правами. Серед переліку користувачів він обирає того, чий права хоче відредагувати. Натискає на значок редагування, після чого з'являється модальне вікно, у якому можна додати або видалити права обраного користувача. Користувач змінює права та натискає кнопку підтвердження. Система оновлює дані щодо прав обраного користувача. Модальне вікно зникає, сторінка відображає оновлені дані.
Extension	На модальному вікні можна як додавати права із випадючого списку, так і видаляти права певного користувача.

Продовження таблиці 1.11

Extension	На модальному вікні можна як додавати права із випадального списку, так і видаляти права певного користувача.
Post-Condition	Права користувача оновлено.

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.4 наведено загальну модель вимог у деревовидному вигляді, а функціональні вимоги описано в таблицях 1.18 – 1.27. Матрицю трасування вимог можна побачити на рисунку 1.5.

Таблиця 1.16 – Функціональна вимога FR-1

Назва	Вхід в акаунт користувача
Опис	Система повинна надавати можливість увійти в акаунт користувачеві шляхом введення пошти та паролю.

Таблиця 1.17 – Функціональна вимога FR-2

Назва	Реєстрація користувача
Опис	Система повинна надавати можливість реєстрації користувачеві шляхом введення пошти, паролю та підтвердження паролю.

Таблиця 1.18 – Функціональна вимога FR-3

Назва	Перегляд створених А/В тестів
Опис	Система повинна надавати можливість авторизованому користувачеві переглядати перелік створених А/В тестів.

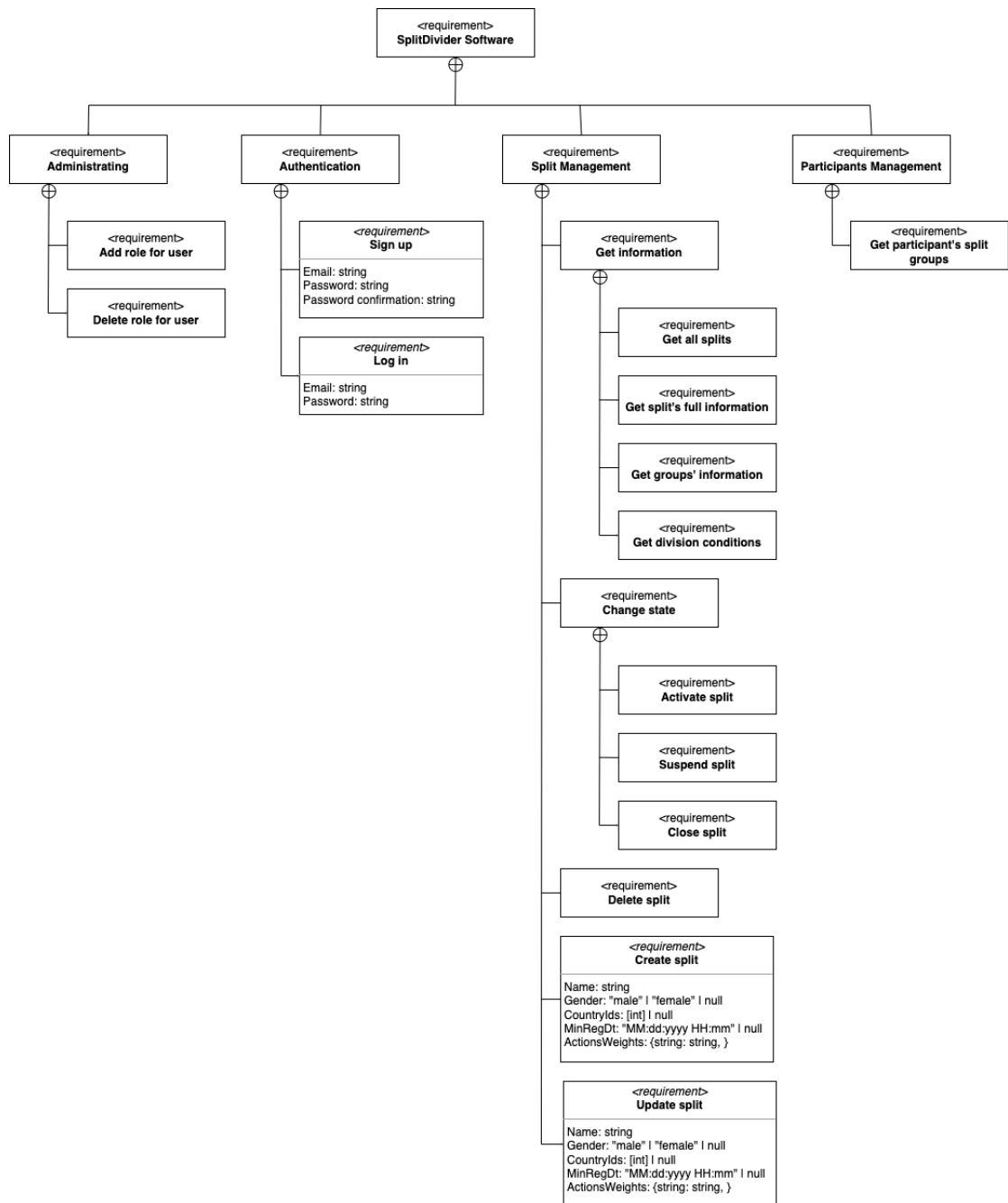


Рисунок 1.4 – Модель вимог

Таблиця 1.19 – Функціональна вимога FR-4

Назва	Перегляд детальної інформації щодо А/В тесту
Опис	Система повинна надавати можливість авторизованому користувачеві переглядати детальну інформацію щодо А/В тесту, яка включає налаштування сегментації та ваги видів взаємодій, а також можливість переглядати дані, на підставі яких відбувалося формування груп у даному тесті.

Таблиця 1.20 – Функціональна вимога FR-5

Назва	Створення А/В тесту
Опис	Система повинна надавати можливість авторизованому користувачеві створювати нові А/В тести шляхом введення інформації про нього.

Таблиця 1.21 – Функціональна вимога FR-6

Назва	Редагування А/В тесту
Опис	Система повинна надавати можливість авторизованому користувачеві змінювати інформацію та налаштування А/В тесту.

Таблиця 1.22 – Функціональна вимога FR-7

Назва	Активація А/В тесту
Опис	Система повинна надавати можливість авторизованому користувачеві активувати А/В тест для формування груп.

Таблиця 1.23 – Функціональна вимога FR-8

Назва	Призупинення А/В тесту
Опис	Система повинна надавати можливість авторизованому користувачеві призупиняти А/В тест для очищення даних щодо розподілу учасників та подальшого перезапуску тесту.

Таблиця 1.24 – Функціональна вимога FR-9

Назва	Зупинення А/В тесту
Опис	Система повинна надавати можливість авторизованому користувачеві зупиняти А/В тест для переведення його в статус «закрито».

Таблиця 1.25 – Функціональна вимога FR-10

Назва	Перегляд інформації щодо тестових груп конкретного учасника
Опис	Система повинна надавати можливість авторизованому користувачеві переглядати інформацію про те, в які групи потрапив конкретний учасник в А/В тестах.

Таблиця 1.26 – Функціональна вимога FR-11

Назва	Надання прав користувачу
Опис	Система повинна надавати можливість адміністратору системи додавати права користувачу.

Таблиця 1.27 – Функціональна вимога FR-12

Назва	Видалення прав користувачу
Опис	Система повинна надавати можливість адміністратору системи видаляти права користувачу.

1.4.2 Розроблення нефункціональних вимог

В результаті дослідження існуючих програмних рішень та аналізу предметної області було виділено низку нефункціональних вимог:

- безпека даних – застосунок не повинен зберігати паролі користувачів у відкритому вигляді, паролі повинні зберігатися у захешованому вигляді; комунікація між клієнтом та сервером повинна відбуватися по протоколу HTTPS;
- зручність та зрозумілість користувацького інтерфейсу – оскільки розроблювана система може бути частиною програмного продукту, який розробляється та підтримується багатонаціональною командою, то найбільш доцільною мовою для користувацького інтерфейсу є англійська; інтерфейс повинен бути простим, лаконічним та зручним.

	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07	FR-08	FR-09	FR-10	FR-11	FR-12
UC-01												
UC-02												
UC-03												
UC-04												
UC-05												
UC-06												
UC-07												
UC-08												
UC-09												
UC-10												

Рисунок 1.5 – Матриця трасування вимог

1.5 Постановка задачі

Метою дипломної роботи є підвищення валідності результатів А/В тестів у багатокористувацьких системах, де присутній взаємний вплив користувачів на їх досвід використання продукту, шляхом формування груп А/В тесту таким чином, щоб учасники з різних груп мали мінімальний вплив одне на одного. Для досягнення поставленої методи необхідно сформулювати відповідну методику та розробити програмне забезпечення, яке буде її використовувати.

Програмне забезпечення повинне надавати необхідний інструментарій для проведення А/В тестів. Виділимо основні задачі ПЗ:

- авторизація користувачів (включаючи реєстрацію та вхід в систему);
- керування правами користувачів системи;
- отримання інформації щодо тестів, у яких бере участь певний учасник;
- керування А/В тестами (створення, редагування, видалення);
- активація А/В тесту із подальшим формуванням груп;
- отримання повної інформації щодо А/В тесту, включно із розбиттям.

Висновки до розділу

У даному розділі було описано результати аналізу предметної області та відомих аналогів пропонованого рішення. Аналіз показав, що на даний момент проблема взаємного впливу учасників різних груп А/В тестів не є вирішеною, або ж ці рішення не представляються на загал. Тому виникає потреба у тому, аби розробити метод для формування груп, який враховував би взаємний впливи та мінімізував його. Формування такого методу та розробка програмного забезпечення, яке використовуватиме його, і є метою даної роботи.

При аналізі предметної області було виокремлено ряд особливостей у роботі із А/В тестами. Наприклад, важливою складовою є можливість налаштовувати сегментацію – тобто визначення характеристик користувачів, які можуть бути учасниками тесту. Також при розробці потрібно враховувати те, що різні види взаємодії можуть мати різний ступінь важливості для того чи іншого тесту.

Було виділено ряд варіантів використання, які враховують потреби користувачів, що можуть виникнути при процесі організації проведення А/В тестів.

Для ПЗ було висунуто ряд функціональних та нефункціональних вимог, які наведені вище. Загалом сервіс повинен бути гнучким, надавати різні інструменти для керування та налаштування А/В тестів, бути зручним у використанні та надавати необхідну інформацію про учасників, тести та підстави формування груп.

Отже, даний сервіс може стати корисним інструментом для проведення А/В тестів у системах, де традиційні способи формування груп учасників не підходять, адже взаємний вплив учасників на користувацький досвід та поведінку на продукті приводять до того, що результати не є цілком валідними. Дипломна робота націлена на те, аби розв’язати дану проблему.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 2.1).

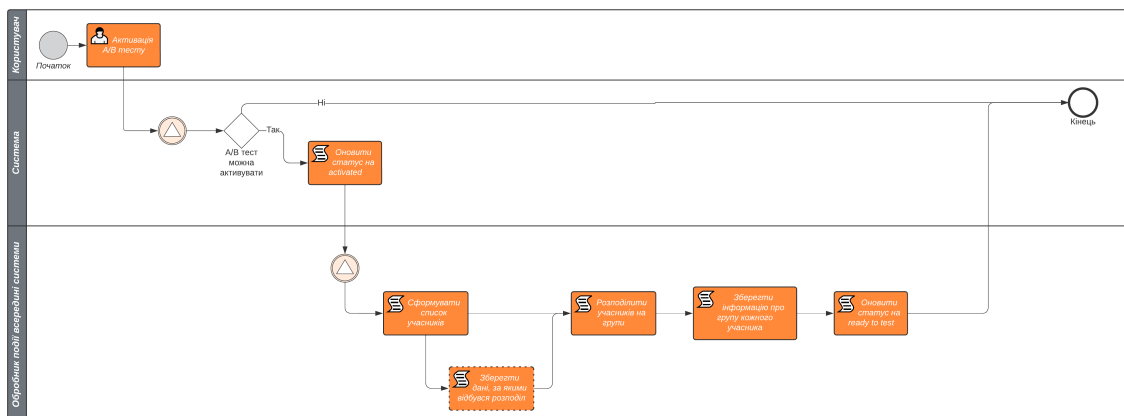


Рисунок 2.1 – Схема бізнес-процесу

Опишемо послідовність активації А/В тесту:

- користувач ініціює активація А/В тесту;
- система приймає запит, перевіряє, чи можна активувати тест;
- якщо тест не знаходиться у якомусь зі станів – «created», «suspended» то його не можна активувати, процес закінчується, з'являється відповідне повідомлення;
- якщо тест можна активувати, система оновлює його стан на «activated» та передає на подальше опрацювання для обробника подій;
- обробник подій формує список учасників тесту відповідно до заданих при створенні А/В тесту критеріїв;
- відбувається побудова графу взаємодії, відповідно до алгоритму, описаного нижче, який серіалізується та зберігається;
- виконується мінімальний розріз графу та формуються дві групи для проведення тесту;
- дані щодо груп учасників зберігаються;

- тест переводиться в стан «ready to test», що сигналізує про закінчення процесу формування груп;
- процес закінчено.

2.2 Архітектура програмного забезпечення

Оскільки розроблюване програмне забезпечення є інструментом для проведення А/В тестів у програмних продуктах із взаємодією користувачів, то необхідно визначити, яким чином вийде найкраще демонструвати його роботу та поведінку.

Метою розробки є саме сервіс для проведення А/В тестів, то немає потреби розробляти цілий програмний продукт із взаємодією. Із цього приходимо до висновку, що потрібно розробити мікросервіс, який може повноцінно функціонувати в екосистемі мікросервісів згадуваного вище продукту. Тобто, мікросервіс має надавати API, яким, за потреби, будуть користуватися інші мікросервіси, а також клієнтська частина проєкту, для демонстрації роботи та зручності використання. Окрім цього, мікросервіс повинен вміти оброблювати повідомлення, які приходитимуть з інших мікросервісів, коли відбувається та чи інша подія.

Переходячи до самого програмного забезпечення, розглянемо архітектурне рішення серверної частини.

Для реалізації було обрано «багатошарову» архітектуру (також «Onion» архітектура). У багаторівневій архітектурі основна увага приділяється різним рівням абстракції в системі. Для схематичного представлення часто використовують стос коробок або ряд концентричних кіл (рисунок 2.2). Відповідно до архітектурного принципу, зв'язки між різними шарами повинні відбуватися лише в інтерфейсах між двома сусідніми шарами. [7]

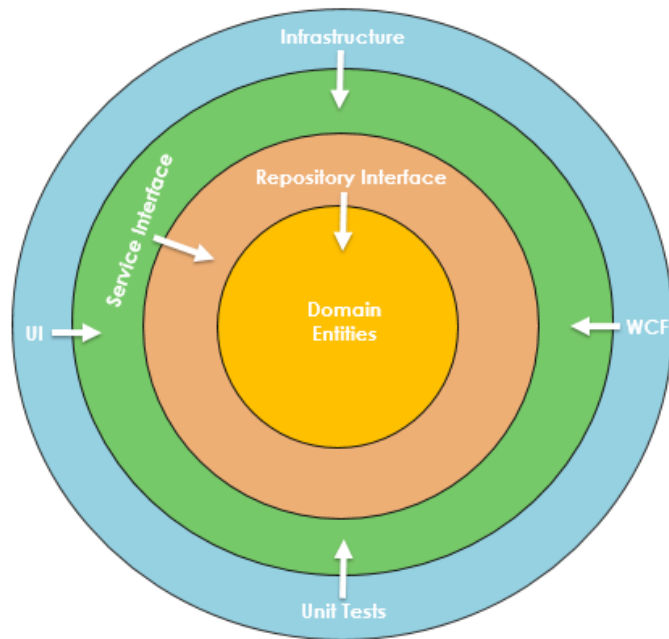


Рисунок 2.2 – Схематичне представлення багатошарової архітектури

Виділимо основні рівні розроблюваного застосунку, та компоненти, які там знаходяться:

- **Domain Entities** – доменні моделі: моделі, які презентують дані предметної області, об'єкти бази даних та конфігурації до них;
- **Repository Interface** – рівень доступу до даних, на ньому, як правило, розміщуються репозиторії та об'єкти доступу до бази даних. Для роботи з БД було обрано Entity Framework, то на цьому рівні розміщується контекст для роботи з базою даних;
- **Service Interface** – сервіси, у яких розміщено бізнес логіку та основні операції над даними, а також допоміжні сервіси, необхідні в роботі, реалізації алгоритмів тощо;
- **User Interface** – контролери, фільтри, валідація – зовнішній рівень, який надає API;
- **Infrastructure** – компоненти, за допомогою яких відбувається отримання повідомлень із інших мікросервісів.

Варто також зауважити, що як патерн для реалізації шару сервісів було обрано CQRS (Command Query Responsibility Segregation) Pattern - архітектурний патерн, який розділяє читання даних (запити) від модифікації

даних (команди) у додатку. Основна ідея CQRS полягає в тому, щоб мати окремі моделі та механізми для обробки операцій читання та запису [13].

Діаграма модулів серверної частини застосунку наведено на рисунку 2.3, а на рисунку 2.4 зображено високорівневу архітектуру застосунку.

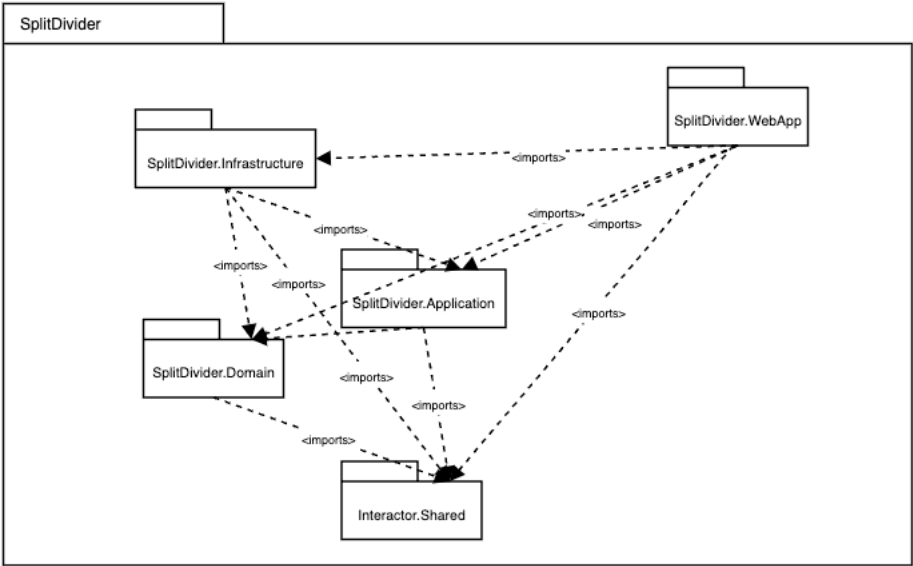


Рисунок 2.3 – Діаграма пакетів серверної частини

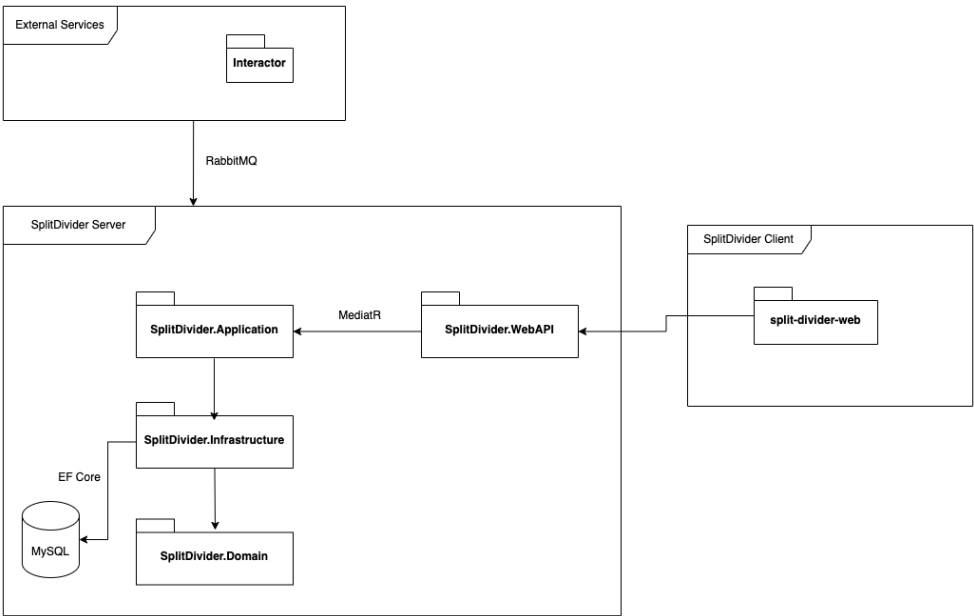


Рисунок 2.4 – Високорівнева архітектура застосунку

2.3 Конструювання програмного забезпечення

У предметній області можна виділити наступні сутності:

- User – учасник A/B тесту;
- Relation – взаємодія між користувачами;
- Split – A/B тест;
- UserSplit – зв'язок між користувачем та A/B тестом.

ER діаграму сутностей наведено на рисунку 2.5.

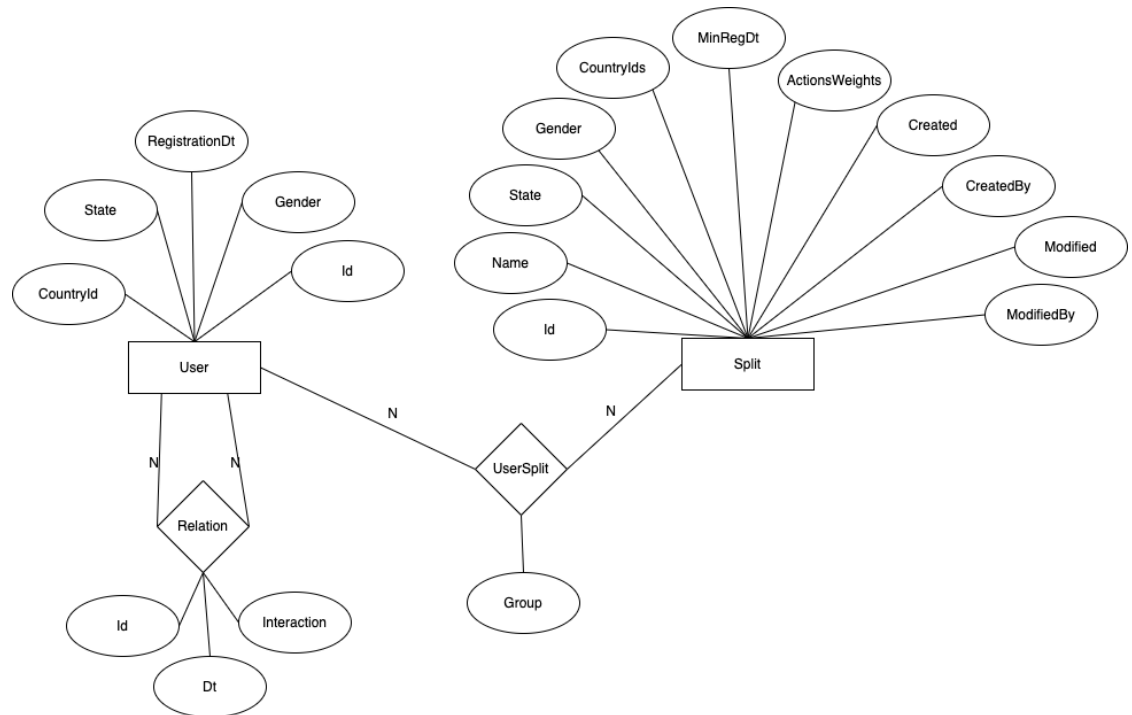


Рисунок 2.5 – ER діаграма сутностей.

Однією із проблем, на вирішення якої націлена дана робота, є розробка методу для формування груп учасників A/B тестів із мінімальним взаємним впливом. Опишемо далі сформульований метод, який буде реалізовано в даній роботі.

Одним із поширених способів математичного представлення зв'язків є граф. Граф $G(V, E, w)$ – математична структура, яка складається із множини вершин V , множини ребер E – множини неупорядкованих пар виду (i, j) або (j, i) , таких що i, j належать множині V [9] та функції w – функції обчислення ваги ребер.

Теорія графів все частіше використовується в психології для моделювання взаємодії в соціальних групах [10]. У даній роботі було вирішено також використати даний підхід.

Для моделювання соціальних зв'язків було вирішено використати наступні правила:

- оскільки зв'язки є двосторонніми, то граф доцільно робити ненапрямленим;
- вершина графа репрезентує учасника А/В тесту; для кожного учасника додається вершина із відповідним ідентифікатором учасника;
- ребро графа – зв'язок між учасниками; між будь-якими двома учасниками може існувати тільки одне ребро із деякою вагою;
- вага ребра означає ступінь взаємодії між двома учасниками: дане значення формується внаслідок аналізу всіх взаємодій, які відбувалися між відповідними учасниками.

Варто також зазначити те, як саме відбувається формування ваги ребра взаємодії.

Відповідно до вимог, які були поставлені раніше, для різних А/В тестів різні види взаємодій можуть мати різні значення. Тому одним із налаштувань тестів є ваги взаємодій, які використовуються при обрахунку акумульованої ваги.

Також потрібно зауважити те, що дата взаємодії теж повинна бути врахованою. Мається на увазі те, що чим давніше відбулася взаємодія, тим меншу вона повинна мати вагу. Тому значення ваги, яке має одна певна взаємодія, потрібно помножити на множник, який є результатом функції від різниці часу між моментом обрахунку та моменту, в який відбулася взаємодія. Верхній поріг функції становить 1. Функція має вигляд, наведений на рисунку 2.6.

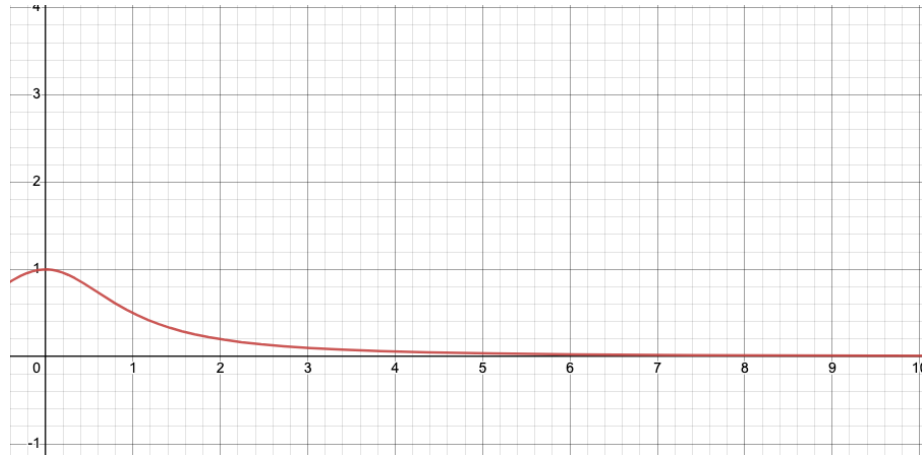


Рисунок 2.6 – Функція залежності множника ваги від різниці часу

Оскільки різниця становить $x \in [0, \infty]$, то розглядається тільки ця область значень, що й задовольняє умови.

Даний графік відповідає функції $f(x) = \frac{1}{1+x^2}$, де x – згадувана вище різниця часу. За цією функцією обраховується значення множника часу для формування ваги.

Таким чином для обчислення дістаються дані всіх взаємодій, які відбувалися між учасниками, із налаштувань тесту береться вага даної взаємодії залежно від її виду, обраховується різниця часу в днях між поточним моментом та моментом взаємодії, обраховується множник. Потім вага множиться на множник, отримані значення для кожної одиниці взаємодії сумуються. В результаті отримуємо вагу ребра між двома учасниками. Інтуїтивно обрахунок ваги ребра – ваги взаємодії – між двома учасниками (x, y), $x, y \in V$, $x \neq y$, можна зобразити наступною формулою:

$$w = \sum_i \left(\frac{1}{1 + \Delta t(i)^2} * k(i) \right),$$

де $\Delta t(i)$ – функція різниці часу (в днях) між моментом формування графу та моментом, у який відбулася взаємодія,

$k(i)$ – початкова вага взаємодії для типу взаємодії, до якого належить взаємодія i (задається при створенні графу).

Оскільки є потреба у відображенні даних, на основі яких відбувся розподіл, то в даній ситуації в якості таких даних є побудований граф взаємодій.

Граф серіалізується у формат graphml та зберігається у файлі із відповідним розширенням.

GraphML (Graph Markup Language) - це формат, який використовується для представлення графів. Він розроблений на основі XML та дозволяє описувати структуру графа, включаючи вузли (вершини) і ребра, а також додаткові атрибути і метадані. GraphML забезпечує стандартизований спосіб представлення графів, що полегшує обмін та спільне використання даних графів між різними програмами та платформами [11].

Наступним кроком для формування груп є мінімальний розріз графу. Для реалізації було обрано алгоритм Стоера-Вагнера.

Алгоритм Стоера-Вагнера використовується для пошуку мінімального розрізу в неорієнтованому, зваженому графі. Мінімальний розріз - це розбиття графа на дві непересічні множини вершин з мінімальною сумарною вагою ребер між цими двома множинами.

Даний алгоритм використовує концепцію "розширення мінімального розрізу" для ітеративного скорочення кількості вершин до тих пір, поки не залишиться тільки одна вершина. Процес включає повторюваний пошук вершини з максимальним значенням ваги ребра, яка є найбільш перспективною вершиною для додавання до зростаючої множини. Скорочуючи вершини, алгоритм ефективно об'єднує вершини і ребра, зменшуючи розмір графа до тих пір, поки не залишиться лише дві множини [12].

Діаграма класів, які створені для реалізації даного алгоритму, наведено на рисунку 2.7.

					КПІ.IT-9127.045440.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		40

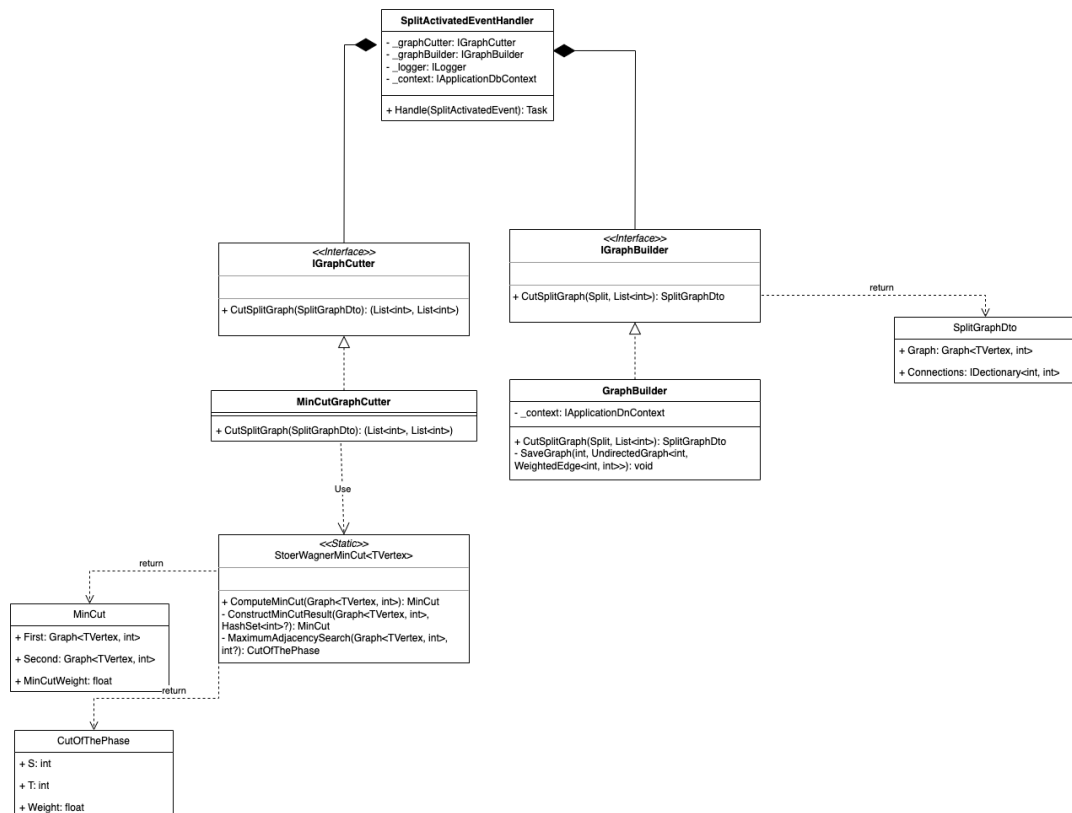


Рисунок 2.7 – Діаграма класів для реалізації побудови та розрізу графу взаємодії

При виборі СУБД було прийнято рішення використовувати MySQL.

MySQL – одна із найпопулярніших систем управління базами даних із відкритим вихідним кодом, яка розроблюється, поширюється та підтримується корпорацією Oracle [5]. Вона підходить для застосунків, де схема є не надто складною, нема інтенсивних математичних обчислень при виконанні запитів. Окрім того, MySQL має велику ком'юніті.

База даних серверу зберігає дані про А/В тести, учасників, взаємодії між учасниками та до яких груп було розподілено учасників, а також дані таблиць для аутентифікації, які використовує ASP.NET Core Identity – обрана бібліотека для проведення реєстрації та керування обліковими записами користувачів.

Опис таблиць БД наведено у таблицях 2.1 – 2.5. У застосунку створено також ряд таблиць для використання ASP.NET Core Identity, але вони не є ключовими, тому будуть наведені тільки на схемі бази даних.

Модель бази даних наведено на кресленні «Схема бази даних».

Таблиця 2.1 – Опис таблиці Splits

Таблиця	Назва поля	Тип даних	Опис
Splits	Id	int	ідентифікаційний номер А/В тесту
	Name	varchar	назва А/В тесту
	State	int	поточний стан А/В тесту (н-д, «створений», «активний», тощо)
	ActionsWeights	json	мапа відповідностей видів взаємодій та їхньої ваги для даного А/В тесту, у серіалізованому вигляді
	MinRegistration Dt	timestamp	мінімальний час реєстрації учасників, які можуть брати участь в А/В тесті
	CountryIds	json	масив внутрішніх ідентифікаторів країн, звідки можуть бути учасники А/В тесту, у серіалізованому вигляді
	Gender	varchar	стать учасників, які можуть брати участь в А/В тест

Продовження таблиці 2.1

Таблиця	Назва поля	Тип даних	Опис
	Created	timestamp	часова відмітка створення А/В тесту
	CreatedBy	varchar	ім'я користувача, який створив А/В тест
	Modified	timestamp	часова відмітка останньої модифікації А/В тесту
	ModifiedBy	varchar	ім'я користувача, який зробив останню модифікацію А/В тест

Таблиця 2.2 – Опис таблиці Users

Таблиця	Назва поля	Тип даних	Опис
Users	Id	int	ідентифікаційний номер учасника
	State	int	поточний стан користувача на платформі (н-д, «активний», «забанений», тощо)
	Gender	varchar	стать учасника
	RegistrationDt	timestamp	часова відмітка реєстрації користувача

Продовження таблиці 2.1

Таблиця	Назва поля	Тип даних	Опис
	CountryId	int	Внутрішній ідентифікатор країни учасника

Таблиця 2.3 – Опис таблиці Relations

Таблиця	Назва поля	Тип даних	Опис
Relations	Id	int	ідентифікаційний номер зв'язку між учасниками
	Interaction	varchar	назва виду взаємодії
	Dt	timestamp	часова відмітка взаємодії користувачів
	UserId	int	ідентифікатор першого учасника
	ContactId	int	ідентифікатор другого учасника

Таблиця 2.4 – Опис таблиці UserSplits

Таблиця	Назва поля	Тип даних	Опис
UserSplits	UserId	int	ідентифікаційний номер учасника
	SplitId	int	ідентифікаційний номер A/B тесту

Продовження таблиці 2.4

Таблиця	Назва поля	Тип даних	Опис
	Group	int	значення групи, до якої розподілено учасника

Таблиця 2.5 – Опис таблиці Users

Таблиця	Назва поля	Тип даних	Опис
Users	Id	int	ідентифікатор учасника
	State	int	стан користувача на платформі
	Gender	varchar	стать учасника
	CountryId	Int	ідентифікатор країни учасника
	RegistrationDt	timestamp	часова відмітка реєстрації користувача

При розробці було використано ряд утиліт, фреймворків та інших засобів розробки ПЗ. Їхній перелік та опис наведено у таблиці 2.6.

Таблиця 2.6 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Rider	Середовище розробки програмного забезпечення серверної частини дипломної роботи. Це кросплатформенна IDE на базі платформи InetllіJ та

Продовження таблиці 2.6

№ п/п	Назва утиліти	Опис застосування
		ReSharper[6]. Також надає можливості підключення до серверу бази даних та роботи з нею.
2	Postman	Програмне забезпечення для виконання REST запитів. Було використане для тестування API інтерфейсів, та клієнтських запитів до серверу.
3	WebStorm	Середовище розробки для клієнтської частини дипломного проєкту.
4	Gephi	Програмне забезпечення для візуалізації та дослідження графів. Має відкритий вихідний код. Було використано для перевірки формування файлів у форматі .graphml, у яких містяться серіалізовані графи.
5	ASP.NET Core	Кросплатформенний фреймворк для створення сучасних веб-додатків за допомогою мови програмування C#. Використано для побудови веб-застосунку, який надає API.
6	Entity Framework Core	Кросплатформенний фреймворк об'єктно-реляційного відображення (ORM) для .NET, який спрощує доступ до баз даних та маніпуляції з ними, надаючи високорівневу абстракцію над базою даних. Використано для реалізації доступу до бази даних.
7	RabbitMQ	Програмне забезпечення для обміну повідомленнями, яке полегшує зв'язок і передачу даних між розподіленими системами або додатками за допомогою протоколу обміну повідомленнями.

Продовження таблиці 2.6

		Використано для реалізації обміну повідомленнями між мікросервісами.
8	MediatR	Бібліотека для реалізації шаблону Mediator у .NET-додатках, що забезпечує вільний зв'язок та спрощену комунікацію між компонентами за допомогою обміну повідомленнями між запитами та відповідями. Використано для реалізації шаблону CQRS.
9	QuickGraph	Бібліотека з відкритим вихідним кодом для .NET, яка надає ефективні структури даних та алгоритми для роботи з графами.
10	React	Бібліотека для побудови SPA. Було використано при розробці клієнтської частини.
11	Bootstrap	Фреймворк для фронтенду, який надає набір готових стилів. Було використано при розробці клієнтської частини.
12	Cytoscape	Модуль, який надає засоби для роботи з файлами у форматі .graphml на веб-сторінках. Використовувалося при розробці клієнтської частини.

Детальний опис компонентів серверної частини винесено в Додаток В, а опис API застосунку розміщено в Додатку Б.

2.4 Аналіз безпеки даних

У сучасному світі дані стали надзвичайно цінним ресурсом для організацій. Розвиток технологій зробив збір даних ефективним та результативним, як ніколи раніше. Однак для повного використання потенціалу даних вкрай важливо забезпечити їхню безпеку та конфіденційність [8].

При розробці програмного забезпечення захист даних було реалізовано наступним чином:

- доступ до особистих даних учасників можливий тільки авторизованим користувачам;
- обмін повідомленнями між сервером та клієнтом здійснюється через захищені протоколи обміну – у даному випадку HTTPS, який має додаткове шифрування;
- паролі користувачів зберігаються тільки у вигляді хешу механізмами бібліотеки ASP.NET Core Identity, яку надає та поширює корпорація Microsoft.

Окрім цього варто зауважити, що подібні інструменти в компаніях, як правило, мають додатковий шар захисту, а саме доступ до них можливий тільки із внутрішньої мережі компанії, а віддалений доступ надається виключно через VPN. Тому користувачі поза компанією не мають доступу до даних.

Висновки до розділу

Даний розділ було присвячено аналізу предметної області, проєктуванню архітектури програмного забезпечення та відповідній реалізації. Було обрано підходи до побудови ПЗ, описано архітектурне рішення та засоби, за допомогою яких було здійснено реалізацію. Було прийнято рішення розробляти ASP.NET Core застосунок, який надаватиме API. Взаємодія з базою даних відбувається засобами Entity Framework Core. Також застосунок побудовано відповідно до архітектурного шаблону багатоварової архітектури та із застосуванням принципів CQRS. Для останнього використовується бібліотека MediatR. Взаємодія з іншими сервісами відбувається через RabbitMQ. Клієнтську частину розроблено за використання бібліотеки React.

Було розроблено метод для формування груп учасників A/B тесту із мінімальним взаємним впливом та описано його реалізацію. Сам метод полягає у формуванні графу взаємодії між користувачами, де вага ребра

становить обчислене значення взаємодії залежно від налаштувань A/B тесту та різниці часу між тепершнім моментом та моментом взаємодії. Потім відбувається мінімальний розріз даного графу за допомогою алгоритму Стоера-Вагнера. На виході маємо два графи, які представляють собою дві групи учасників.

Окрім цього, було спроектовано схему бази даних застосунку та описано відповідні сутності. В якості СУБД обрано MySQL.

Наостанок було проаналізовано засоби, якими досягається безпека даних у застосунку.

					КПІ.IT-9127.045440.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		49

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для аналізу якості ПЗ було використано сервіс embold.io. Даний сервіс є статичним аналізатором коду на основі штучного інтелекту. Embold є досить популярним засобом та оцінює код, знаходячи помилки, вразливості та проблемні місця в архітектурному рішенні.

На рисунку 3.1 зображено результати аналізу ПЗ за допомогою згаданого вище сервісу:

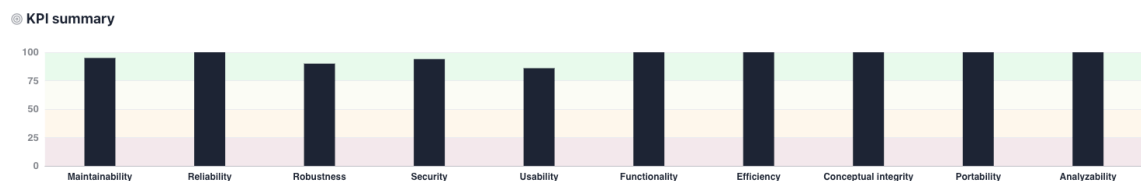


Рисунок 3.1 – Результати аналізу ПЗ статичним аналізатором коду
Детальний аналіз ключових показників наведено у таблиці 3.1.

Таблиця 3.1 – Аналіз ключових результатів статичного аналізатору.

Характеристика	Оцінка, %	Коментар
Підтримуваність	95	Характеризує наскільки легко програмне забезпечення можна модифікувати та розширювати.
Надійність	100	Характеризує наскільки програмне забезпечення за тих чи інших умов виконує поставлені задачі.
Безпека	92	Характеризує ступінь захисту даних в системі. Даний показник має нижче значення через те, що певні секретні дані зберігаються напряду в коді або у файлах налаштувань. При розгортанні застосунку

Продовження таблиці 3.1

Характеристика	Оцінка, %	Коментар
		подібні дані варто винести в спеціалізоване сховище (наприклад, у Vault).
Зручність використання	86	Характеризує наскільки програмне забезпечення є зручним для використання певними людьми та для розширення. Через використання фреймворків, які суворо прописують правила написання коду, цей показник є нижчим.
Функціональність	100	Характеризує наскільки програмне забезпечення відповідає поставленим йому функціям та виконує необхідну роботу.
Ефективність	100	Характеризує ефективність використання ресурсів системою.

Як видно на графіку, найнижчий показник має метрика “Usability”, яка відповідає за підтримку коду. Це пов’язано із тим, що використані фреймворки строго диктують інтерфейси, які повинні бути реалізовані, але сервіс пропонує здійснити реалізацію через статичні методи. Дане рішення видається надлишковим, а показник у 86% достатнім. Решта показників мають оптимальні значення, в середньому вищі 90%.

Аналіз безпеки даних було здійснено в розділі 2.4. Окрім того, програмне забезпечення показало себе як таке, що має захист від некоректних дій користувача, надає контроль введення інформації, має зручний та зрозумілий користувацький інтерфейс та забезпечує цілісність даних у базі даних.

3.2 Опис процесів тестування

Для перевірки роботи програмного забезпечення було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.2 – 3.20.

Таблиця 3.2 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 8 до 32 символів, який містить хоча б одну маленьку та хоча б одну велику англійські літери, одне число і один спеціальний символ, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку.

Таблиця 3.3 – Тест 1.2

Тест	Вхід користувача в систему
Модуль	Вхід користувача в систему
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці входу

Продовження таблиці 3.3

Вхідні данні	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, за якою зареєстровано акаунт користувача та правильний пароль, який відповідає паролю, за яким відбулася реєстрація. Після цього натискається кнопка підтвердження входу.
Очікуваний результат	Вхід в систему проходить успішно, користувача авторизовано, він перенаправляється на головну сторінку.
Фактичний результат	Вхід в систему проходить успішно, користувача авторизовано, він перенаправляється на головну сторінку.

Таблиця 3.4 – Тест 1.3

Тест	Вихід користувача із системи
Модуль	Вихід користувача із системи
Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на будь-якій зі сторінок веб-сайту
Вхідні данні	-
Опис проведення тесту	Натискається кнопка Logout, яка знаходиться в хедері сторінки.
Очікуваний результат	Користувач більше не вважається авторизованим. Він переходить на головну сторінку.
Фактичний результат	Користувач більше не вважається авторизованим. Він переходить на головну сторінку.

Таблиця 3.5 – Тест 1.4

Тест	Перегляд тестових груп існуючого учасника
Модуль	Перегляд інформації про учасника
Номер тесту	1.4
Початковий стан системи	Користувач знаходиться на сторінці з пошуком учасників.
Вхідні данні	Ідентифікатор існуючого учасника.
Опис проведення тесту	Користувач вводить у поле пошуку ідентифікатор учасника, після цього натискає кнопку пошуку.
Очікуваний результат	На сторінці відображається інформація про те, у яких А/В тестах бере участь учасник та до яких груп він належить.
Фактичний результат	На сторінці відображається інформація про те, у яких А/В тестах бере участь учасник та до яких груп він належить.

Таблиця 3.6 – Тест 1.5

Тест	Перегляд тестових груп неіснуючого учасника
Модуль	Перегляд інформації про учасника
Номер тесту	1.5
Початковий стан системи	Користувач знаходиться на сторінці з пошуком учасників.
Вхідні данні	Ідентифікатор неіснуючого учасника.
Опис проведення тесту	Користувач вводить у поле пошуку ідентифікатор учасника, після цього натискає кнопку пошуку.
Очікуваний результат	На сторінці відображається повідомлення про те, що дані відсутні.
Фактичний результат	На сторінці відображається повідомлення про те, що дані відсутні.

Таблиця 3.7 – Тест 1.6

Тест	Перегляд списку А/В тестів
Модуль	Перегляд списку А/В тестів
Номер тесту	1.6
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці.
Вхідні данні	-
Опис проведення тесту	Користувач натискає на кнопку переходу на сторінку із списком тестів.
Очікуваний результат	Користувач перенаправляється на сторінку зі списком А/В тестів. На сторінці відображається список А/В тестів із загальною інформацією про тести.
Фактичний результат	Користувач перенаправляється на сторінку зі списком А/В тестів. На сторінці відображається список А/В тестів.

Таблиця 3.8 – Тест 1.7

Тест	Перегляд списку А/В тестів
Модуль	Перегляд списку А/В тестів
Номер тесту	1.6
Початковий стан системи	Неавторизований користувач знаходиться на головній сторінці.
Вхідні данні	-
Опис проведення тесту	Користувач натискає на кнопку переходу на сторінку із списком тестів.
Очікуваний результат	Користувач отримує повідомлення про необхідність входу.

Продовження таблиці 3.8

Фактичний результат	Користувача перенаправлено на сторінку з повідомленням про те, що необхідно увійти в систему.
---------------------	---

Таблиця 3.9 – Тест 1.8

Тест	Перегляд списку А/В тестів з певним статусом
Модуль	Перегляд списку А/В тестів
Номер тесту	1.8
Початковий стан системи	Користувач знаходиться на сторінці зі списком А/В тестів.
Вхідні данні	Статус А/В тестів.
Опис проведення тесту	Користувач натискає на випадючий список та обирає статус.
Очікуваний результат	На сторінці оновлюється інформація та відображаються тільки А/В тести із обраним статусом.
Фактичний результат	На сторінці оновлюється інформація та відображаються тільки А/В тести із обраним статусом.

Таблиця 3.10 – Тест 1.9

Тест	Перегляд списку А/В тестів з статусом, з яким нема А/В тестів.
Модуль	Перегляд списку А/В тестів
Номер тесту	1.9
Початковий стан системи	Користувач знаходиться на сторінці зі списком А/В тестів.
Вхідні данні	Статус А/В тестів за яким нема А/В тестів.

Продовження таблиці 3.10

Опис проведення тесту	Користувач натискає на випадаючий список та обирає статус.
Очікуваний результат	Користувач отримує повідомлення про те, що А/В тести з таким статусом відсутні.
Фактичний результат	На сторінці зникає попередній список та з'являється інформація про те, що А/В тести з таким статусом відсутні.

Таблиця 3.11 – Тест 1.10

Тест	Створення А/В тесту
Модуль	Створення А/В тесту
Номер тесту	1.10
Початковий стан системи	Користувач знаходиться на сторінці зі списком А/В тестів.
Вхідні данні	Назва А/В тесту, значення ваги видів взаємодії, параметри сегментації: країни, стать, мінімальна дата реєстрації.
Опис проведення тесту	Користувач натискає на кнопку створення нового А/В тесту. На екрані з'являється модальне вікно із відповідними полями. Користувач вводить коректу назву довжиною від 5 до 30 символів, заповнює значення в таблиці для ваги видів взаємодії, обирає країни реєстрації у чекбоксі, обирає стать учасників та мінімальну дату реєстрації у календарі. Натискає кнопку підтвердження.
Очікуваний результат	Користувач потрапляє на сторінку зі списком А/В тестів. У списку відображається щойно створений А/В тест.
Фактичний результат	Користувач потрапляє на сторінку зі списком А/В тестів. У списку відображається щойно створений А/В тест.

Таблиця 3.12 – Тест 1.11

Тест	Редагування А/В тесту
Модуль	Редагування А/В тесту
Номер тесту	1.11
Початковий стан системи	Користувач знаходиться на сторінці зі списком А/В тестів.
Вхідні данні	Назва А/В тесту, значення ваги видів взаємодії, параметри сегментації: країни, стать, мінімальна дата реєстрації. А/В тест знаходиться у статусі, з яким можна редагувати дані (створений або призупинений).
Опис проведення тесту	Користувач натискає на кнопку редагування біля обраного А/В тесту. На екрані з'являється модальне вікно із відповідними полями. Користувач редагує інформацію: вводить нову назву довжиною від 5 до 30 символів, оновлює значення в таблиці для ваги видів взаємодії, редагує країни реєстрації, стать учасників та мінімальну дату реєстрації. Натискає кнопку підтвердження.
Очікуваний результат	Користувач потрапляє на сторінку зі списком А/В тестів. У списку відображається оновлений А/В тест.
Фактичний результат	Користувач потрапляє на сторінку зі списком А/В тестів. У списку відображається оновлений А/В тест.

Таблиця 3.13 – Тест 1.12

Тест	Перегляд повної інформації А/В тесту без розподілу
Модуль	Перегляд повної інформації А/В тесту
Номер тесту	1.12
Початковий стан системи	Користувач знаходиться на сторінці зі списком А/В тестів.

Продовження таблиці 3.13

Вхідні данні	А/В тест, який знаходиться у статусі без розподілу (створений або призупинений).
Опис проведення тесту	Користувач натискає на А/В тест у загальному списку та потрапляє на сторінку із повною інформацією. Після цього натискає на кнопку для відображення графу.
Очікуваний результат	На сторінці відображається повна інформація про А/В тест (назва, параметри сегментації та ваги видів взаємодії). Також відображається повідомлення про те, що тест ще не має розподілу.
Фактичний результат	На сторінці відображається повна інформація про А/В тест (назва, параметри сегментації та ваги видів взаємодії). Також відображається повідомлення про те, що тест ще не має розподілу.

Таблиця 3.14 – Тест 1.13

Тест	Перегляд повної інформації А/В тесту з розподілом
Модуль	Перегляд повної інформації А/В тесту
Номер тесту	1.13
Початковий стан системи	Користувач знаходиться на сторінці зі списком А/В тестів.
Вхідні данні	А/В тест, який знаходиться у статусі з розподілом (готовий до тестування або закритий).
Опис проведення тесту	Користувач натискає на А/В тест у загальному списку та потрапляє на сторінку із повною інформацією. Після цього натискає на кнопку для відображення графу.
Очікуваний результат	На сторінці відображається повна інформація про А/В тест (назва, параметри сегментації та ваги видів взаємодії), а також інформація про розподіл. Також

Продовження таблиці 3.14

	відображається граф взаємодії, за яким відбувався розподіл.
Фактичний результат	На сторінці відображається повна інформація про A/B тест (назва, параметри сегментації та ваги видів взаємодії), а також інформація про розподіл. Також відображається граф взаємодії, за яким відбувався розподіл.

Таблиця 3.15 – Тест 1.14

Тест	Активація A/B тесту
Модуль	Керування життєвим циклом A/B тесту
Номер тесту	1.14
Початковий стан системи	Користувач знаходиться на сторінці з повною інформацією про A/B тест.
Вхідні данні	A/B тест, який знаходиться у статусі «створений» або «призупинений»
Опис проведення тесту	Користувач натискає на кнопку активації.
Очікуваний результат	Сторінку перезавантажено, інформацію про статус оновлено. Після того, як відбувся розподіл, на сторінці можна переглянути інформацію про групи та граф взаємодії.
Фактичний результат	Сторінку перезавантажено, інформацію про статус оновлено. Після того, як відбувся розподіл, на сторінці можна переглянути інформацію про групи та граф взаємодії.

Таблиця 3.16 – Тест 1.15

Тест	Призупинення А/В тесту
Модуль	Керування життєвим циклом А/В тесту
Номер тесту	1.15
Початковий стан системи	Користувач знаходиться на сторінці з повною інфомрацією про А/В тест.
Вхідні данні	А/В тест, який знаходиться у статусі «готовий до тестування».
Опис проведення тесту	Користувач натискає на кнопку призупинення.
Очікуваний результат	Сторінку перезавантажено, інформацію про статус оновлено, інформація про розподіл та граф взаємодії відсутня.
Фактичний результат	Сторінку перезавантажено, інформацію про статус оновлено. Інформацію про розподіл не відображено, а після натискання на кнопку відображення графа з'являється повідомлення про те, що тест не має груп.

Таблиця 3.17 – Тест 1.16

Тест	Перегляд ролей користувача
Модуль	Перегляд ролей користувача
Номер тесту	1.16
Початковий стан системи	Користувач із правами адміністратора знаходиться на сторінці пошуку користувачів.
Вхідні данні	Пошта існуючого користувача.
Опис проведення тесту	Адміністратор вводить в пошуковий рядок електронну пошту та натискає кнопку пошуку.

Продовження таблиці 3.17

Очікуваний результат	На сторінці відображається інформація про те, які ролі має користувач із такою електронною поштою.
Фактичний результат	На сторінці відображається список ролей, які має користувач із такою електронною поштою.

Таблиця 3.18 – Тест 1.17

Тест	Перегляд ролей неіснуючого користувача
Модуль	Перегляд ролей користувача
Номер тесту	1.17
Початковий стан системи	Користувач із правами адміністратора знаходиться на сторінці пошуку користувачів.
Вхідні данні	Пошта неіснуючого користувача.
Опис проведення тесту	Адміністратор вводить в пошуковий рядок електронну пошту та натискає кнопку пошуку.
Очікуваний результат	На сторінці відображається повідомлення про те, що такого користувача не зареєстровано в системі.
Фактичний результат	На сторінці відображається повідомлення про те, що такого користувача не зареєстровано в системі.

Таблиця 3.19 – Тест 1.18

Тест	Видалення ролі користувачу
Модуль	Зміна ролей користувача
Номер тесту	1.18
Початковий стан системи	Користувач із правами адміністратора знаходиться на сторінці, де відображаються ролі користувача системи.
Вхідні данні	Роль, яку потрібно видалити.

Продовження таблиці 3.19

Опис проведення тесту	Адміністратор натискає на кнопку видалення біля ролі, яку потрібно видалити.
Очікуваний результат	На сторінці відображається оновлена інформація про ролі користувача (без видаленої ролі)
Фактичний результат	На сторінці відображається оновлена інформація про ролі користувача (без видаленої ролі)

Таблиця 3.20 – Тест 1.19

Тест	Додавання ролі користувачу
Модуль	Зміна ролей користувача
Номер тесту	1.19
Початковий стан системи	Користувач із правами адміністратора знаходиться на сторінці, де відображаються ролі користувача системи.
Вхідні данні	Роль, яку потрібно додати.
Опис проведення тесту	Адміністратор натискає на кнопку додавання ролі та натискає на роль у випадаючому списку.
Очікуваний результат	На сторінці відображається оновлена інформація про ролі користувача, серед яких є додана роль.
Фактичний результат	На сторінці відображається оновлена інформація про ролі користувача, серед яких є додана роль.

3.3 Опис контрольного прикладу

Далі детально опишемо контрольний приклад створення А/В тесту.

Авторизований користувач знаходиться на головній сторінці та натискає на кнопку переходу на сторінку зі списком А/В тестів (рис. 3.2).

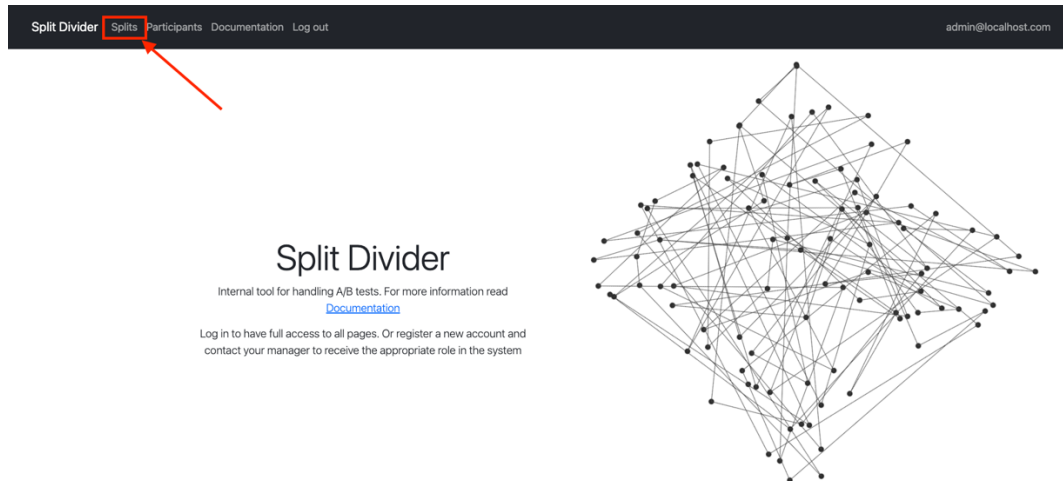


Рисунок 3.2 – Головна сторінка, червоним виділено кнопку переходу на сторінку зі списком А/В тестів

Якщо в авторизованого користувача відсутні права для керування А/В тестами, його перенаправлено на сторінку з повідомленням (рис. 3.3).

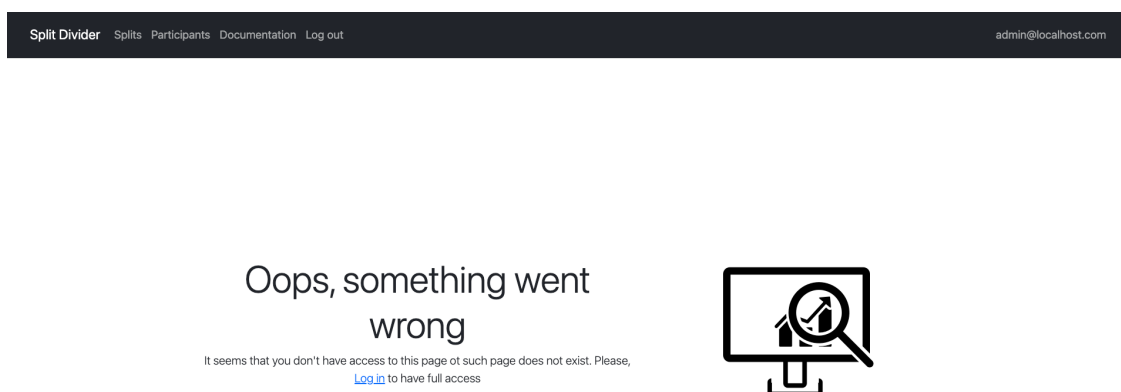


Рисунок 3.3 – Сторінка з повідомленням про помилку

Якщо користувач має згадані вище права, його перенаправлено на сторінку зі списком тестів. Користувач натискає на кнопку додавання нового А/В тесту (рис. 3.4).

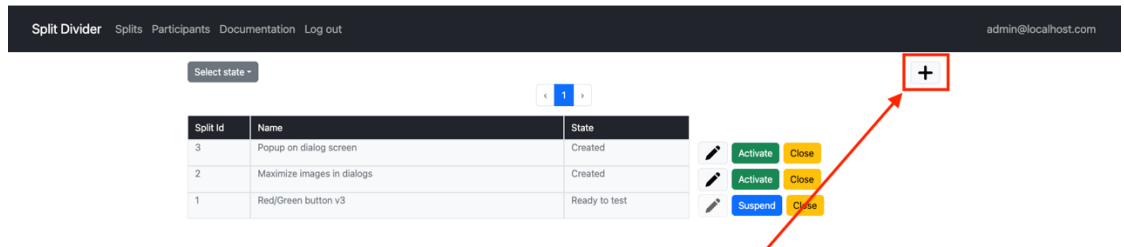


Рисунок 3.4 – Сторінка зі списком А/В тестів, червоним виділено кнопку створення нового А/В тесту

Після цього на екрані з’являється модальне вікно з полями для створення А/В тесту.

Якщо користувач вводить некоректне ім’я – з’являється відповідне повідомлення (рис. 3.5).

Create new split

Name Length should be between 5 and 30 characters

Interactions' weights

Name	Weight
dialog	1
like	1
hide	1
video-call	1

Countries

Gender

Minimal Registration Time

Рисунок 3.5 – Модальне вікно, на якому відображається помилка про некоректне ім’я А/В тесту

При спробі ввести значення ваги взаємодії для певного типу, яке менше 1 або більше 30 – значення встановлюється у відповідний ліміт. Користувач заповнює поля та натискає кнопку підтвердження (рис. 3.6).

Create new split

Name
Test A/B test

Interactions' weights

Name	Weight
dialog	30
like	30
hide	1
video-call	1

Countries
Ukraine Poland USA Great Britain Germany

Gender
Male Female All

Minimal Registration Time
11.05.2023, 15:44

Submit Cancel

Рисунок 3.6 – Заповнене модальне вікно, червоним виділено кнопку підтвердження

При успішному створенні А/В тесту модальне вікно закривається, користувач потрапляє на сторінку зі списком, щойно створений тест відображається у загальному списку зі статусом «Created» (рис. 3.7).

Split Divider Splits Participants Documentation Log out admin@localhost.com

Select state +

< 1 >

Split id	Name	State		
4	Test A/B test	Created	✎	Activate Close
3	Popup on dialog screen	Created	✎	Activate Close
2	Maximize images in dialogs	Created	✎	Activate Close
1	Red/Green button v3	Ready to test	✎	Suspend Close

Рисунок 3.7 – Сторінка зі списком А/В тестів, червоним виділено щойно створений тест

Якщо виникла помилка при створенні, на екрані з'являється повідомлення про це (рис. 3.8).

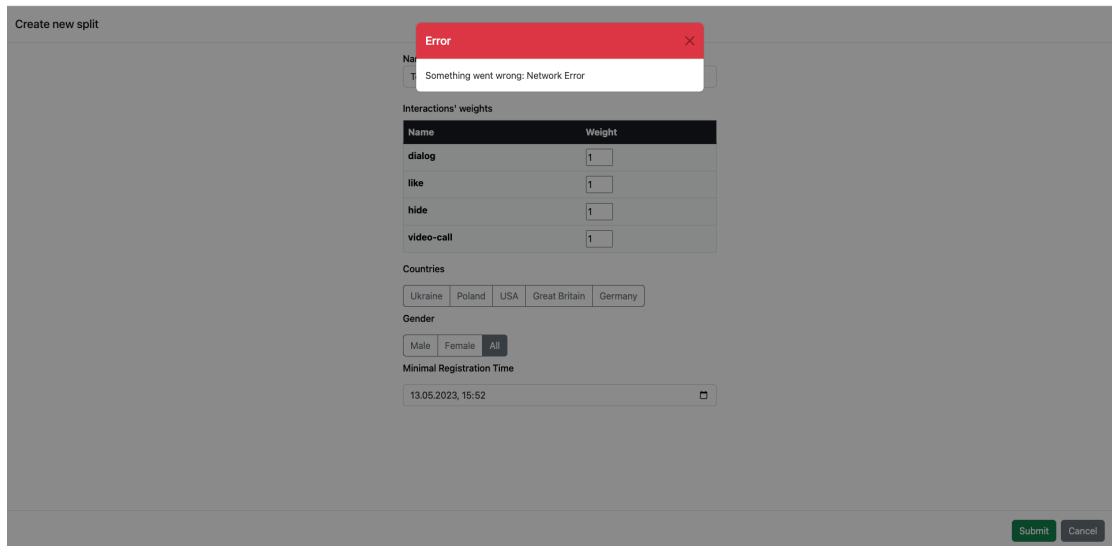


Рисунок 3.8 – Повідомлення про помилку при створенні А/В тесту

Висновки до розділу

В даному розділі було описано результати аналізу якості розробленого ПЗ. Аналіз написаного коду було зроблено за допомогою сервісу embold.io. Сервіс оцінив програмне забезпечення за основними метриками, такими як використання, розширюваність, підтримання, надійність, безпека, функціональність, ефективність тощо. Розроблене програмне забезпечення отримало високі показники із мінімальною оцінкою 86%.

Також було описано процес мануального тестування сервісу. Детальні щодо кожного з тестів наведено в пункті 3.2. Загалом, як показало тестування, сервіс поводить себе так, як це очікується користувачем, є зручним, простим у використанні та надає весь необхідний функціонал.

Насамкінець було описано контрольний приклад тестування. В якості контрольного прикладу було обрано створення нового А/В тесту з метою охопити якомога більше кроків та нюансів, які можуть виникнути. Детальний опис та зображення сторінок клієнтською частини наведено у відповідному пункті.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Враховуючи архітектурне рішення, клієнтську та серверну частини було вирішено розгортати на різних серверах. Діаграму розгортання наведено на рисунку 4.1.

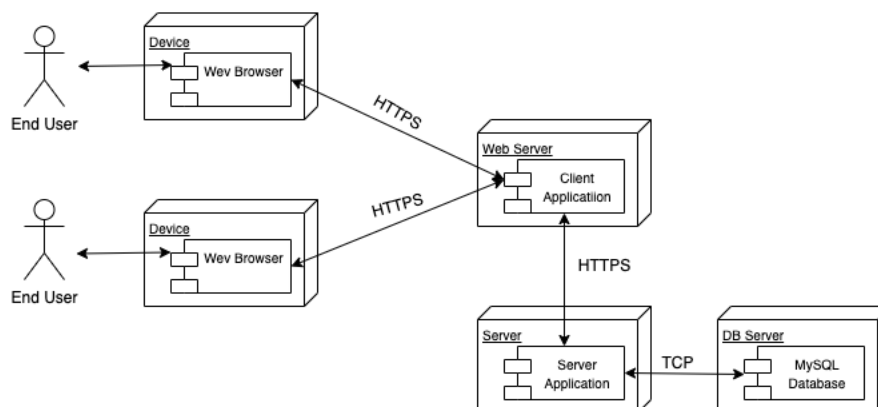


Рисунок 4.1 – Діаграма розгортання застосунку

Розгортання було здійснено за допомогою сервісу Heroku. Оскільки в Heroku нема підтримки застосунків, написаних на платформі .NET, потрібно додатково загорнути застосунки в Docker контейнери. Для реалізації CI/CD було також використано GitHub Actions.

CI/CD (Continuous Integration/Continuous Delivery)- набір практик, які автоматизують процес створення, тестування та розгортання змін у програмному забезпеченні. Безперервна інтеграція передбачає часте об'єднання змін коду в спільний репозиторій, в той час як безперервна доставка фокусується на автоматизації процесу розгортання[14].

Розгортання автоматично запускається, коли код гілки master змінюється. Тоді створюється новий образ Docker, використовуючи Dockerfile. Після цього образ розгортається у Heroku.

Інформацію про роботу клієнтської частини наведено на рисунку 4.1.

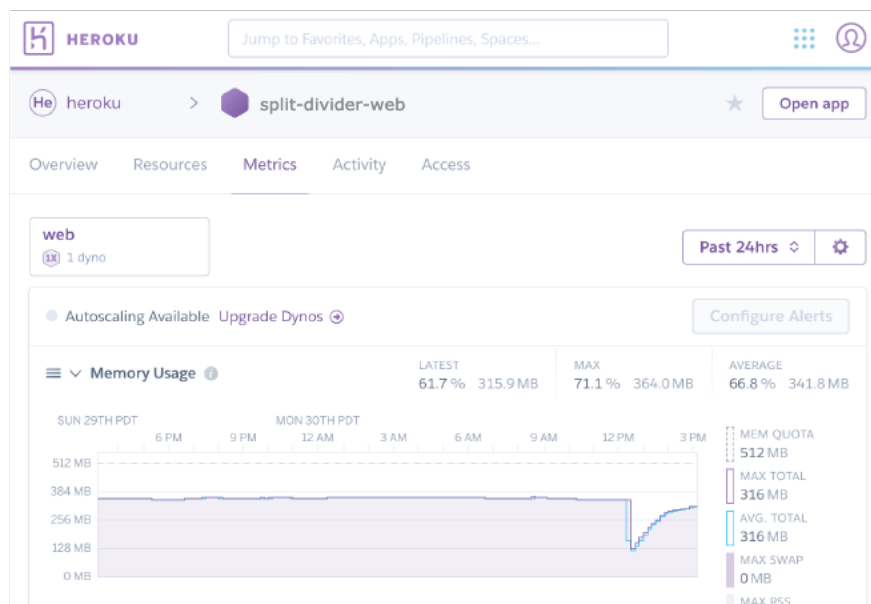


Рисунок 4.1 – Інформація про роботу клієнту

4.2 Підтримка програмного забезпечення

З часом виникне необхідність внести зміни в застосунок, тому повинна бути можливість оновити версію. Для постійної доставки нової версії було використано GitHub Actions.

Новий реліз починається, коли в репозиторії оновлюється гілка master. Таким чином досягається continuous delivery, який включає постійно доставку нової версії застосунку.

Висновки до розділу

У розділі було розглянуто загальний процес розгортання застосунку та доставки нових версії. Для розгортання було використано сервіс Heroku, який було інтегровано з GitHub Actions.

ВИСНОВКИ

Головною метою даної роботи є підвищення валідності результатів А/В тестів у багатокористувацьких системах із взаємним впливом. Мається на увазі, що при проведенні А/В тестів у продуктах, де є взаємний вплив, учасники з різних тестованих груп можуть впливати на досвіди використання одне одного. Це у свою чергу впливає на показники, які аналізуються після проведення тесту.

Поставленої мети було досягнуто шляхом розроблення нової методики для формування груп учасників. В основі методики лежать тенденції соціологічних наук до представлення груп людей та зв'язків між ними у вигляді графа.

Сама методика полягає у формуванні графу взаємодії між учасниками А/В тесту, де вершини – це учасники, ребра – зв'язки між ними, а вага ребра – обчислене значення ваги взаємодії між двома учасниками. Потім для сформованого графу знаходять мінімальний розріз, таким чином формуючи два підграфи. Множини вершин підграфів і є групами А/В тесту.

Одержані результати на даному етапі розвитку ідеї відповідають висунутим вимогам, є актуальними та важливими в галузі проведення А/В тестів та аналізу даних. Однак, робота має великий спектр точок зростання, тому доцільними були б подальші дослідження.

В результаті, окрім розробленої методики, після детального аналізу предметної області, було створено веб-застосунок, який використовує згадану вище методику, та спроектовано базу даних. В якості СУБД було обрано MySQL. Подібні застосунок, як правило, є внутрішніми корпоративними інструментами для проведення А/В тестів у кінцевому продукті компанії.

Після реалізації застосунків було мануально протестовано для перевірки, чи виконує ПЗ всі поставлені вимоги.

					КПІ.IT-9127.045440.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		70

Розроблене ПЗ орієнтоване на продуктові компанії, які доставляють продукти, що включають соціальну взаємодію, та хочуть підвищити якість проведення А/В тестів.

Підсумовуючи, варто відзначити, що розроблені методика та продукт пропонують новий спосіб формування груп учасників А/В тестів. Нині все більше компаній приходять до практики проведення А/В тестів, але часто зустрічаються із вище згаданою проблемою. Дана робота дає можливість зменшити наслідки проблеми, та досягнути як кращого досвіду використання для користувачів, так і швидшого розвитку кінцевого продукту.

					КПІ.ІТ-9127.045440.02.81	Арк.
						71
Змін.	Арк.	№ докум.	Підп.	Дата.		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Holmström Olsson H., Bosch J., Fabijan A. Experimentation that Matters: A Multi-case Study on the Challenges with A/B Testing, 2017. URL: https://link.springer.com/chapter/10.1007/978-3-319-69191-6_12 (дата звернення 06.05.2023).
- 2) Xu Y., Nanyu Ch., Fernandez A., Sinno O., Bhasin A. From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks. 2015. URL: <https://dl.acm.org/doi/abs/10.1145/2783258.2788602> (дата звернення 07.05.2023).
- 3) Deng. A, Lu J., Litz J. Trustworthy Analysis of Online A/B Tests: Pitfalls, challenges and solutions. 2017. URL: <https://dl.acm.org/doi/abs/10.1145/3018661.3018677> (дата звернення 07.05.2023).
- 4) King R., Churchill E. F., Tan C. Designing with Data: Improving the User Experience with A/B Testing. Sebastopol, California, 2017. 29 p.
- 5) Офіційна документація MySQL. URL: <https://dev.mysql.com/doc/refman/8.0/en/preface.html> (дата звернення 10.05.2023).
- 6) Офіційний вебсайт JetBrains Rider. URL: <https://www.jetbrains.com/rider/> (дата звернення 14.05.2023).
- 7) Land R. A Brief Survey of Software Architecture. Department of Computer Engineering, Mälardalen University, Västerås, Sweden, 2002. 5 p.
- 8) Bertino E. Data Security and Privacy: Concepts, Approaches, and Research Directions. Atlanta, 2016. 11 p.
- 9) Chen W. K. Applied Graph Theory. 13th edition. London, 1976. 32 p.
- 10) Harary F., Norman R. Z. Graph Theory as a Mathematical Model in Social Science. Research Center for Group Dynamics, University of Michigan, 1971. 21 p.
- 11) Brandes U., Eiglsperger M., Lerner J., Pich Ch. Handbook of Graph

drawing and visualization. London, 2010. 512 p.

12) Stoer M., Wagner F. A Simple Min-Cut Algorithm. Journal of the ACM, 1997. 586 p.

13) Kabbedijk J., Jansen S., Brinkkemper S. A case study of the variability consequences of the CQRS pattern in online business software. EuroPLoP '12: Proceedings of the 17th European Conference on Pattern Languages of Programs, 2012. 8 p.

14) Singh Ch., Seth Gaba N., Kaur M., Kaur B. Comparison of Different CI/CD Tools Integrated with Cloud Platform. Noida, 2019. 4 p.

15) Офіційний вебсайт ASP.NET MVC. URL:
<https://dotnet.microsoft.com/en-us/apps/aspnet/mvc> (дата звернення: 14.05.2023)

16) Офіційний вебсайт ASP.NET Core. URL:
<https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0> (дата звернення: 14.05.2023)

17) Офіційний вебсайт NancyFX. URL: <https://nancyfx.org/> (дата звернення: 14.05.2023)

18) Офіційний вебсайт Microsoft Visual Studio. URL:
<https://visualstudio.microsoft.com/> (дата звернення: 14.05.2023)

19) Офіційний вебсайт Microsoft Visual Studio Code. URL:
<https://code.visualstudio.com/> (дата звернення: 14.05.2023)

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



Ім'я користувача:
Лісовиченко Олег Іванович

Дата перевірки:
01.06.2023 08:37:38 EEST

Дата звіту:
01.06.2023 08:50:56 EEST

ID перевірки:
1015357496

Тип перевірки:
Doc vs Internet + Library

ID користувача:
76913

Назва документа: IT-91_Хільчук_ПЗ
Кількість сторінок: 70 Кількість слів: 10907 Кількість символів: 84397 Розмір файлу: 2.60 MB ID файлу: 1015024956

6.12%
Схожість

Найбільша схожість: 2.55% з джерелом з Бібліотеки (ID файлу: 1014981451)

3.11% Джерела з Інтернету	101	Сторінка 72
6.07% Джерела з Бібліотеки	199	Сторінка 74

0% Цитат

- Вилучення цитат вимкнене
- Вилучення списку бібліографічних посилань вимкнене

0.14%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 0%)

0% Вилучення з Інтернету	27	Сторінка 75
0.14% Вилученого тексту з Бібліотеки	33	Сторінка 75

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	71
------------------	----

ДОДАТОК Б ОПИС API СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина застосунку надає REST API з усіма необхідними для повноцінного функціонування програмного продукту точками входу. OpenAPI документація наведена на рисунках Б.1 – Б.2, а опис точок входу наведено в таблиці Б.1.

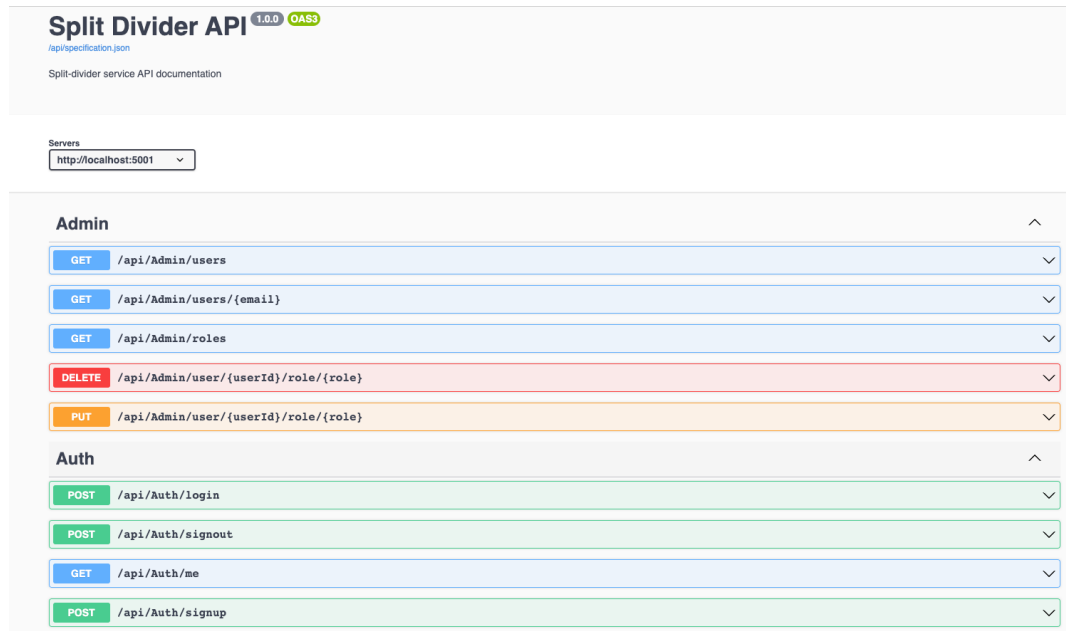


Рисунок Б.1 – Частина 1. OpenAPI документація серверної частини

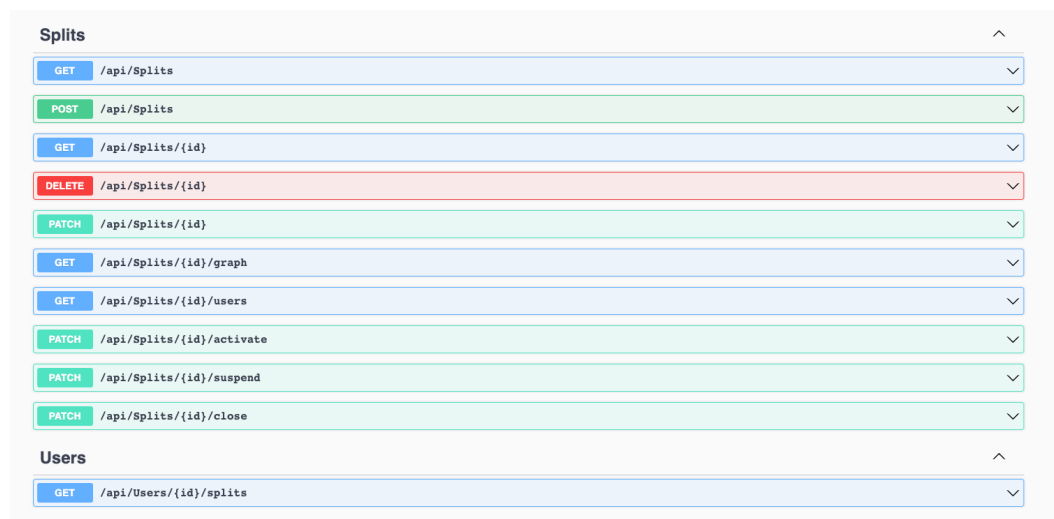


Рисунок Б.2 – Частина 2. OpenAPI документація серверної частини

Таблиця Б.1 – Опис точок входу серверної частини.

Шлях	HTTP метод	Опис
api/admin/users	GET	Отримання користувачів програмного продукту.
api/admin/users {email}	GET	Отримання користувача програмного продукту із вказаною електронною поштою.
api/admin/roles	GET	Отримання списку ролей.
api/admin/user/{userId}/role/{roleId}	PUT	Додавання для користувача з ідентифікатором userId ролі з ідентифікатором roleId.
api/admin/user/{userId}/role/{roleId}	DELETE	Видалення користувачу з ідентифікатором userId ролі з ідентифікатором roleId.
api/auth/signup	POST	Реєстрація користувача в системі. В тілі передаються електронна пошта, пароль та підтвердження паролю.
api/auth/login	POST	Вхід користувача в систему. В тілі запиту повинні приходити електронна пошта та пароль.
api/auth/logout	POST	Вихід з системи.
api/auth/me	GET	Отримання поточного користувача.

Продовження таблиці Б.1

Шлях	HTTP метод	Опис
api/admin/user/{userId}/role/{roleId}	PUT	Додавання для користувача з ідентифікатором userId ролі з ідентифікатором roleId.
api/admin/user/{userId}/role/{roleId}	DELETE	Видалення користувачу з ідентифікатором userId ролі з ідентифікатором roleId.
api/auth/signup	POST	Реєстрація користувача в системі. Точка приймає в тілі параметри електронна пошти, паролю та підтвердження паролю.
api/auth/login	POST	Вхід користувача в систему. В тілі запиту повинні приходити електронна пошта та пароль.
api/auth/logout	POST	Вихід з системи.
api/splits	GET	Отримання переліку всіх тестів. У рядку запита також можна передавати необов'язковий параметр фільтрації за станом. Повертає пагінований результат.

Продовження таблиці Б.1

Шлях	HTTP метод	Опис
api/splits	POST	Створення нового A/B тесту.
api/splits/{id}	GET	Отримання детальної інформації щодо A/B тесту з ідентифікатором id.
api/splits/{id}	DELETE	Видалення A/B тесту з ідентифікатором id.
api/splits/{id}	PATCH	Редагування даних A/B тесту з ідентифікатором id.
api/splits/{id}/users	GET	Отримання переліку учасників та до яких груп вони належать у A/B тесті з ідентифікатором id
api/splits/{id}/activate	PATCH	Активація A/B тесту з ідентифікатором id
api/splits/{id}/suspend	PATCH	Призупинення A/B тесту з ідентифікатором id
api/splits/{id}/close	PATCH	Закінчення A/B тесту з ідентифікатором id
api/users/{id}/splits	GET	Отримання даних щодо тестіх груп у тестах, у яких брав чи бере участь учасник з ідентифікатором id.

ДОДАТОК В ОПИС КЛЮЧОВИХ КОМПОНЕНТІВ СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина складається з ряду компонентів. На рисунку В.1 наведено загальну структуру проєкту.

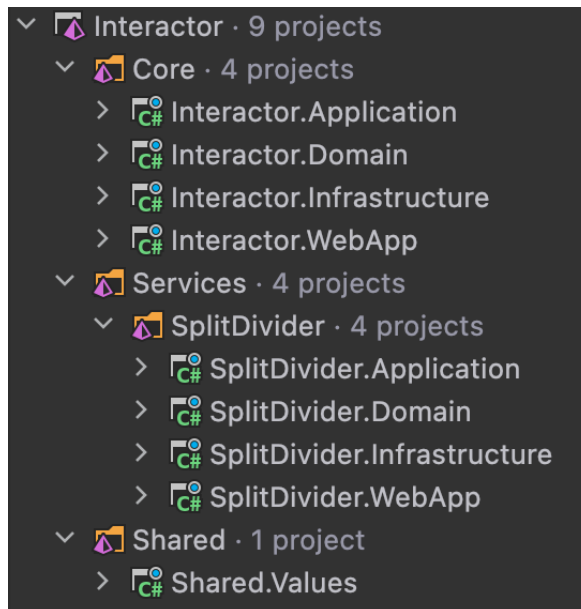


Рисунок В.1 – Загальна структура проєкту серверної частини

Папка Core містить у собі проєкти, які допомагають моделювати поведінку екосистеми мікросервісів. Тобто це окремий застосунок, який надає можливість генерувати дані (користувачів, взаємодії), та надсилати повідомлення про це до розроблюваного сервісу, цим самим моделюючи взаємодію між мікросервісами.

Папка SplitDivider містить проєкти серверної частини розроблюваного застосунку. Основна кодова база знаходиться тут.

Папка Shared містить у собі бібліотеку, у якій знаходяться спільні між мікросервісами константи та сутності, таким чином будучи єдиним джерелом правди щодо спільних даних.

Опис ключових компонентів наведено у таблиці В.1.

Таблиця В.1 – Опис ключових компонентів серверної частини

Компонент	Опис
AdminController	Контролер, який опрацьовує запити, пов'язані з адмініструванням програмного продукту. Тобто керування користувачами та ролями.
AuthController	Контролер, який опрацьовує запити авторизації користувача (реєстрація, вхід).
UsersController	Контролер, який опрацьовує запити щодо інформації про учасника А/В тесту.
SplitsController	Контролер, що відповідає за обробку всіх запитів, пов'язаних з А/В тестами.
Receiver	Компонент, який відповідає за отримання повідомлень через RabbitMQ та передачу їх на обробку в зареєстровані хендлери.
UserRegisteredEventHandler	Обробник повідомлення про реєстрацію учасника.
UserChangedEventHandler	Обробник повідомлення про зміну даних учасника.
UsersInteractedEventHandler	Обробник повідомлення про взаємодію між учасниками.
CreateRelationsCommand	Команда для створення екземпляру взаємодії між учасниками.
CreateRelationsCommandHandler	Обробник відповідної команди

Продовження таблиці В.1

Компонент	Опис
SplitActivatedEventHandler	Обробник внутрішньої події активації А/В тесту.
GraphBuilder	Компонент, який відповідає за побудову графа взаємодії відповідно до алгоритму, описаного в пункті 2.3 пояснювальної записки.
MinCutGraphCutter	Компонент, який формує групи учасників за результатами розрізу графа.
StoerWagnerMinCut	Компонент, який виконує мінімальний розріз графу відповідно до алгоритму Стоера-Вагнера.
IdentityService	Компонент, який дає можливість фасадно взаємодіяти з сервісами, які надає ASP.NET Core Identity
ApplicationDbContext	Контекст Entity Framework Core, який надає можливість взаємодіяти з БД.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

Методика та програмне забезпечення формування груп учасників А/В тестів

з мінімальним взаємним впливом

Текст програми

КПІ.IT-9127.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юлія КРАМАР

Нормоконтроль:

_____ Максим

ГОЛОВЧЕНКО

Виконавець:

_____ Олена ХІЛЬЧУК

Файл BaseEntity.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace SplitDivider.Domain.Common;

public abstract class BaseEntity
{
    public int Id { get; set; }

    private readonly List<BaseEvent> _domainEvents = new();

    [NotMapped]
    public IReadOnlyCollection<BaseEvent> DomainEvents =>
        _domainEvents.AsReadOnly();

    public void AddDomainEvent(BaseEvent domainEvent)
    {
        _domainEvents.Add(domainEvent);
    }

    public void RemoveDomainEvent(BaseEvent domainEvent)
    {
        _domainEvents.Remove(domainEvent);
    }

    public void ClearDomainEvents()
    {
        _domainEvents.Clear();
    }
}
```

Файл BaseEvent.cs

```
using MediatR;

namespace SplitDivider.Domain.Common;

public abstract class BaseEvent : INotification
{
}
```

Файл Split.cs

```
namespace SplitDivider.Domain.Entities;

public class Split : BaseAuditableEntity
{
    public static readonly SplitState[] EditableStates = {
        SplitState.Created, SplitState.Suspended };

    public static readonly SplitState[] ActivatableStates = {
        SplitState.Created, SplitState.Suspended };

    public static readonly SplitState[] SuspendableStates = {
        SplitState.Activated, SplitState.ReadyToTest };

    public static readonly SplitState[] ClosableStates = {
        SplitState.Created, SplitState.ReadyToTest, SplitState.Suspended };
}
```

					КПІ.ІТ-9127.045440.03.12	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

private Gender? _gender;

public string Name { get; set; } = "None";

public SplitState State { get; set; }

public Dictionary<InteractionType, int> ActionsWeights { get; set; } =
new();

public List<int>? CountryIds { get; set; }

public string? Gender {
    get => _gender?.Name;
    set => _gender = value == null ? null :
Shared.Values.ValueObjects.Gender.From(value);
}

public DateTime? MinRegistrationDt { get; set; }
}

```

Файл Relation.cs

```

namespace SplitDivider.Domain.Entities;

public class Relation : BaseEntity
{
    public int UserId { get; set; }

    public int ContactId { get; set; }

    private InteractionType? _interaction;

    public DateTime Dt { get; set; }

    public string Interaction {
        get => _interaction?.Name ?? "undefined";
        set => _interaction = InteractionType.From(value);
    }
}

```

Файл User.cs

```

namespace SplitDivider.Domain.Entities;

public class User : BaseEntity
{
    private Gender? _gender;

    public int CountryId { get; set; }

    public DateTime RegistrationDt { get; set; }

    public UserState State { get; set; }

    public string Gender {
        get => _gender?.Name ?? "undefined";
        set => _gender = Shared.Values.ValueObjects.Gender.From(value);
    }
}

```

					КПІ.ІТ-9127.045440.03.12	Арк.
						3
Змін.	Арк.	№ докум.	Підп.	Дата.		

Файл UserSplit.cs

```
namespace SplitDivider.Domain.Entities;

public class UserSplit
{
    public static readonly int GroupControl = 0;

    public static readonly int GroupTest = 1;

    public int UserId { get; set; }

    public int SplitId { get; set; }

    public int Group { get; set; }
}
```

Файл SplitState.cs

```
namespace SplitDivider.Domain.Enums;

public enum SplitState
{
    Created = 0,
    Activated,
    ReadyToTest,
    Suspended,
    Closed
}
```

Файл SplitActivatedEvent.cs

```
namespace SplitDivider.Domain.Enums;

public enum SplitState
{
    Created = 0,
    Activated,
    ReadyToTest,
    Suspended,
    Closed
}
```

Файл SplitConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.ChangeTracking;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using Newtonsoft.Json;
using Shared.Values.ValueObjects;

namespace SplitDivider.Infrastructure.Persistence.Configurations;

public class SplitConfiguration : IEntityTypeConfiguration<Split>
{
    public void Configure(EntityTypeBuilder<Split> builder)
    {
        builder.Property(s => s.Name)
            .HasMaxLength(30)
    }
}
```

					КПІ.ІТ-9127.045440.03.12	Арк.
						4
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        .IsRequired();

builder.Property(s => s.State)
    .IsRequired();

builder.Property(s => s.LastModified)
    .HasColumnType("timestamp");

builder.Property(s => s.Created)
    .HasColumnType("timestamp")
    .IsRequired();

builder.Property(s => s.MinRegistrationDt)
    .HasColumnType("timestamp");

var countryIdsValueComparer = new ValueComparer<List<int>>>(
    (c1, c2) => c1.SequenceEqual(c2),
    c => c.Aggregate(0, (a, v) => HashCode.Combine(a,
v.GetHashCode())),
    c => c.ToList());

builder.Property(s => s.CountryIds)
    .HasConversion(new ValueConverter<List<int>?, string>(
        v => JsonConvert.SerializeObject(v),
        v => JsonConvert.DeserializeObject<List<int>>(v) ?? new()))
    .HasColumnType("json")
    .IsRequired();

builder.Property(s => s.CountryIds)
    .Metadata
    .SetValueComparer(countryIdsValueComparer);

var actionsWeightsValueComparer = new
ValueComparer<Dictionary<InteractionType, int>>>(
    (c1, c2) => c1.SequenceEqual(c2),
    c => c.Aggregate(0, (a, v) => HashCode.Combine(a,
v.GetHashCode())),
    c => c.ToDictionary(cp => cp.Key, cp => cp.Value));

builder.Property(s => s.ActionsWeights)
    .HasConversion(new ValueConverter<Dictionary<InteractionType,
int>, string>(
        v => JsonConvert.SerializeObject(
            v.ToDictionary(p => p.Key.Name, p => p.Value)),
        v => (JsonConvert.DeserializeObject<Dictionary<string,
int>>(v) ?? new())
            .ToDictionary(p => InteractionType.From(p.Key), p =>
p.Value)))
    .HasColumnType("json")
    .IsRequired();

builder.Property(s => s.ActionsWeights)
    .Metadata
    .SetValueComparer(actionsWeightsValueComparer);
}
}

```

Файл RelationConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace SplitDivider.Infrastructure.Persistence.Configurations;

```

					КП.ІТ-9127.045440.03.12	Арк.
						5
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

public class RelationConfiguration : IEntityTypeConfiguration<Relation>
{
    public void Configure(EntityTypeBuilder<Relation> builder)
    {
        builder.Property(r => r.Dt)
            .HasColumnType("timestamp")
            .IsRequired();

        builder.Property(r => r.Interaction)
            .IsRequired();

        builder.HasOne<User>()
            .WithMany()
            .HasForeignKey(r => r.UserId)
            .IsRequired()
            .OnDelete(DeleteBehavior.Cascade);

        builder.HasOne<User>()
            .WithMany()
            .HasForeignKey(r => r.ContactId)
            .IsRequired()
            .OnDelete(DeleteBehavior.Cascade);
    }
}

```

Файл UserConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace SplitDivider.Infrastructure.Persistence.Configurations;

public class UserConfiguration : IEntityTypeConfiguration<User>
{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.Metadata.SetTableName("Users");

        builder.Property(u => u.Id)
            .ValueGeneratedNever();

        builder.Property(u => u.RegistrationDt)
            .HasColumnType("timestamp")
            .IsRequired();

        builder.Property(u => u.Gender)
            .IsRequired();

        builder.Property(u => u.CountryId)
            .IsRequired();

        builder.Property(u => u.State)
            .IsRequired();
    }
}

```

Файл UserSplitConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

```

					КПІ.IT-9127.045440.03.12	Арк.
						6
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

namespace SplitDivider.Infrastructure.Persistence.Configurations;

public class UserSplitConfiguration : IEntityTypeConfiguration<UserSplit>
{
    public void Configure(EntityTypeBuilder<UserSplit> builder)
    {
        builder.HasKey(us => new { us.UserId, us.SplitId });

        builder.Property(us => us.Group)
            .IsRequired();

        builder.HasOne<User>()
            .WithMany()
            .HasForeignKey(r => r.UserId)
            .IsRequired()
            .OnDelete(DeleteBehavior.Cascade);

        builder.HasOne<Split>()
            .WithMany()
            .HasForeignKey(r => r.SplitId)
            .IsRequired()
            .OnDelete(DeleteBehavior.Cascade);
    }
}

```

Файл ApplicationDbContext.cs

```

using System.Reflection;
using Duende.IdentityServer.EntityFramework.Options;
using MediatR;
using Microsoft.AspNetCore.ApiAuthorization.IdentityServer;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Options;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Infrastructure.Identity;
using SplitDivider.Infrastructure.Persistence.Interceptors;

namespace SplitDivider.Infrastructure.Persistence;

public class ApplicationDbContext :
    ApiAuthorizationDbContext<ApplicationUser>, IApplicationDbContext
{
    private readonly IMediator _mediator;

    private readonly AuditableEntitySaveChangesInterceptor
        _auditableEntitySaveChangesInterceptor;

    public ApplicationDbContext(
        DbContextOptions<ApplicationDbContext> options,
        IOptions<OperationalStoreOptions> operationalStoreOptions,
        IMediator mediator,
        AuditableEntitySaveChangesInterceptor
            auditableEntitySaveChangesInterceptor)
        : base(options, operationalStoreOptions)
    {
        _mediator = mediator;
        _auditableEntitySaveChangesInterceptor =
            auditableEntitySaveChangesInterceptor;
    }

    public DbSet<User> AppUsers => Set<User>();
}

```

					КПІ.IT-9127.045440.03.12	Арк.
						7
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

public DbSet<Relation> Relations => Set<Relation>();

public DbSet<Split> Splits => Set<Split>();

public DbSet<UserSplit> UserSplits => Set<UserSplit>();

protected override void OnModelCreating(ModelBuilder builder)
{
    builder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());

    base.OnModelCreating(builder);
}

protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
{
    optionsBuilder.AddInterceptors(_auditableEntitySaveChangesInterceptor);
}

public override async Task<int> SaveChangesAsync(CancellationToken
cancellation_token = default)
{
    await _mediator.DispatchDomainEvents(this);

    return await base.SaveChangesAsync(cancellation_token);
}
}

```

Файл CreateUserCommand.cs

```

using System.Globalization;
using MediatR;
using Shared.Values.Enums;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.Application.Users.Commands.CreateUser;

public record CreateUserCommand : IRequest
{
    public int Id { get; init; }

    public int CountryId { get; init; }

    public string Gender { get; init; }

    public string RegDt { get; init; }
}

public class CreateUserCommandHandler : IRequestHandler<CreateUserCommand>
{
    private readonly IApplicationDbContext _context;

    public CreateUserCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(CreateUserCommand request, CancellationToken
cancellation_token)
    {
        var entity = new User

```

					КПІ.IT-9127.045440.03.12	Арк.
						8
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        {
            Id = request.Id,
            State = UserState.Registered,
            CountryId = request.CountryId,
            Gender = request.Gender,
            RegistrationDt = DateTime.Parse(request.RegDt,
CultureInfo.InvariantCulture)
        };

        _context.AppUsers.Add(entity);

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

Файл CreateUserCommandValidator.cs

```

using System.Globalization;
using FluentValidation;
using Shared.Values.ValueObjects;

namespace SplitDivider.Application.Users.Commands.CreateUser;

public class CreateUserCommandValidator :
AbstractValidator<CreateUserCommand>
{
    public CreateUserCommandValidator()
    {
        RuleFor(u => u.Gender)
            .Must(Gender.IsSupported);

        RuleFor(u => u.RegDt)
            .Must(dt => DateTime.TryParse(dt, CultureInfo.InvariantCulture,
out _));

        RuleFor(u => u.CountryId)
            .Must(cid => cid >= 0);
    }
}

```

Файл UpdateUserCommand.cs

```

using MediatR;
using Shared.Values.Enums;
using SplitDivider.Application.Common.Exceptions;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.Application.Users.Commands.UpdateUser;

public record UpdateUserCommand : IRequest
{
    public int Id { get; init; }

    public UserState? State { get; init; }

    public string? Gender { get; init; }

    public int? CountryId { get; init; }
}

public class UpdateUserCommandHandler : IRequestHandler<UpdateUserCommand>

```

					КПІ.IT-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

```

{
    private readonly IApplicationDbContext _context;

    public UpdateUserCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(UpdateUserCommand request, CancellationToken
cancellationToken)
    {
        var entity = await _context.AppUsers
            .FindAsync(request.Id, cancellationToken);

        if (entity == null)
        {
            throw new NotFoundException(nameof(User), request.Id);
        }

        if (request.State.HasValue)
        {
            entity.State = request.State.Value;
        }

        if (request.Gender != null)
        {
            entity.Gender = request.Gender;
        }

        if (request.CountryId.HasValue)
        {
            entity.CountryId = request.CountryId.Value;
        }

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

Файл UpdateUserCommandValidator.cs

```

using FluentValidation;
using Shared.Values.ValueObjects;

namespace SplitDivider.Application.Users.Commands.UpdateUser;

public class UpdateUserCommandValidator :
    AbstractValidator<UpdateUserCommand>
{
    public UpdateUserCommandValidator()
    {
        RuleFor(u => u.Gender)
            .Must(Gender.IsSupported!)
            .When(u => u.Gender != null);

        RuleFor(u => u.CountryId)
            .Must(cid => cid >= 0)
            .When(u => u.CountryId.HasValue);

        RuleFor(u => u.State)
            .IsInEnum()
            .When(u => u.State.HasValue);
    }
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.IT-9127.045440.03.12

Арк.

10

Файл GetSplitUsersQuery.cs

```
using AutoMapper;
using AutoMapper.QueryableExtensions;
using MediatR;
using Microsoft.EntityFrameworkCore;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.Application.Users.Queries.GetSplitUsers;

public record GetSplitUsersQuery(int Id) :
    IRequest<IReadOnlyCollection<SplitUserDto>>;

public class GetSplitUsersQueryHandler : IRequestHandler<GetSplitUsersQuery,
    IReadOnlyCollection<SplitUserDto>>
{
    private readonly IApplicationDbContext _context;
    private readonly IMapper _mapper;

    public GetSplitUsersQueryHandler(IApplicationDbContext context, IMapper mapper)
    {
        _context = context;
        _mapper = mapper;
    }

    public async Task<IReadOnlyCollection<SplitUserDto>>
    Handle(GetSplitUsersQuery request, CancellationToken cancellationToken)
    {
        return await _context.UserSplits
            .Where(us => us.SplitId == request.Id)
            .OrderByDescending(us => us.SplitId)
            .ProjectTo<SplitUserDto>(_mapper.ConfigurationProvider)
            .ToListAsync(cancellationToken);
    }
}
```

Файл SplitUserDto.cs

```
using SplitDivider.Application.Common.Mappings;

namespace SplitDivider.Application.Users.Queries.GetSplitUsers;

public class SplitUserDto : IMapFrom<UserSplit>
{
    public int UserId { get; set; }

    public int Group { get; set; }
}
```

Файл GetUserSplitQuery.cs

```
using AutoMapper;
using AutoMapper.QueryableExtensions;
using MediatR;
using Microsoft.EntityFrameworkCore;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.Application.Users.Queries.GetUserSplits;
```

					КПІ.ІТ-9127.045440.03.12	Арк.
						11
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

public record GetUserSplitsQuery(int Id) :
    IRequest<IReadOnlyCollection<UserSplitDto>>;

public class GetUserSplitsQueryHandler : IRequestHandler<GetUserSplitsQuery,
    IReadOnlyCollection<UserSplitDto>>
{
    private readonly IApplicationDbContext _context;
    private readonly IMapper _mapper;

    public GetUserSplitsQueryHandler(IApplicationDbContext context, IMapper
mapper)
    {
        _context = context;
        _mapper = mapper;
    }

    public async Task<IReadOnlyCollection<UserSplitDto>>
Handle(GetUserSplitsQuery request, CancellationToken cancellationToken)
    {
        return await _context.UserSplits
            .Where(us => us.UserId == request.Id)
            .OrderByDescending(us => us.SplitId)
            .ProjectTo<UserSplitDto>(_mapper.ConfigurationProvider)
            .ToListAsync(cancellationToken);
    }
}

```

Файл UserSplitDto.cs

```

using SplitDivider.Application.Common.Mappings;

namespace SplitDivider.Application.Users.Queries.GetUserSplits;

public class UserSplitDto : IMapFrom<UserSplit>
{
    public int SplitId { get; set; }

    public int Group { get; set; }
}

```

Файл GetSplitQuery.cs

```

using AutoMapper;
using MediatR;
using SplitDivider.Application.Common.Exceptions;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.Application.Splits.Queries.GetSplit;

public record GetSplitQuery(int Id) : IRequest<SplitDto>;

public class GetSplitQueryHandler : IRequestHandler<GetSplitQuery, SplitDto>
{
    private readonly IApplicationDbContext _context;
    private readonly IMapper _mapper;

    public GetSplitQueryHandler(IApplicationDbContext context, IMapper
mapper)
    {
        _context = context;
    }
}

```

					КПІ.ІТ-9127.045440.03.12	Арк.
						12
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        _mapper = mapper;
    }

    public async Task<SplitDto> Handle(GetSplitQuery request,
        CancellationToken cancellationToken)
    {
        var entity = await _context.Splits
            .FindAsync(request.Id, cancellationToken);

        if (entity == null)
        {
            throw new NotFoundException(nameof(Split), request.Id);
        }

        return _mapper.Map<SplitDto>(entity);
    }
}

```

Файл SplitDto.cs

```

using System.Globalization;
using AutoMapper;
using SplitDivider.Application.Common.Mappings;

namespace SplitDivider.Application.Splits.Queries.GetSplit;

public class SplitDto : IMapFrom<Split>
{
    public int Id { get; init; }

    public string Name { get; set; }

    public int State { get; set; }

    public string CreatedAt { get; set; }

    public IReadOnlyDictionary<string, int> ActionsWeights { get; set; }

    public string? Gender { get; set; }

    public IReadOnlyList<int>? CountryIds { get; set; }

    public string? MinRegDt { get; set; }

    public void Mapping(Profile profile)
    {
        profile.CreateMap<Split, SplitDto>()
            .ForMember(nameof(CreatedAt),
                opts => opts.MapFrom(src =>
                    src.Created.ToString(CultureInfo.InvariantCulture)))
            .ForMember(nameof(ActionsWeights), opts => opts.MapFrom(src =>
                    src.ActionsWeights
                        .ToDictionary(p => p.Key.Name, p => p.Value)
                        .AsReadOnly()))
            .ForMember(nameof(Gender), opts => opts.MapFrom(src => src.Gender
                != null ? src.Gender.ToString() : null))
            .ForMember(nameof(MinRegDt), opts => opts.MapFrom(src =>
                    src.MinRegistrationDt.HasValue
                        ?
                        src.MinRegistrationDt.Value.ToString(CultureInfo.InvariantCulture)
                        : null))
            .ForMember(nameof(CountryIds), opts => opts.MapFrom(src =>
                    src.CountryIds != null ? src.CountryIds.AsReadOnly() : null));
    }
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІТ-9127.045440.03.12

Арк.

13

```
}  
}
```

Файл GetSplitsWithPaginationQuery.cs

```
using AutoMapper;  
using AutoMapper.QueryableExtensions;  
using MediatR;  
using SplitDivider.Application.Common.Interfaces;  
using SplitDivider.Application.Common.Mappings;  
using SplitDivider.Application.Common.Models;  
  
namespace SplitDivider.Application.Splits.Queries.GetSplitsWithPagination;  
  
public record GetSplitsWithPaginationQuery :  
    IRequest<PaginatedList<SplitBriefDto>>  
{  
    public int? State { get; init; }  
    public int PageNumber { get; init; } = 1;  
    public int PageSize { get; init; } = 20;  
}  
  
public class GetSplitsWithPaginationQueryHandler :  
    IRequestHandler<GetSplitsWithPaginationQuery, PaginatedList<SplitBriefDto>>  
{  
    private readonly IApplicationDbContext _context;  
    private readonly IMapper _mapper;  
  
    public GetSplitsWithPaginationQueryHandler(IApplicationDbContext context,  
        IMapper mapper)  
    {  
        _context = context;  
        _mapper = mapper;  
    }  
  
    public async Task<PaginatedList<SplitBriefDto>>  
    Handle(GetSplitsWithPaginationQuery request, CancellationToken  
        cancellation_token)  
    {  
        var res = _context.Splits.AsQueryable();  
  
        if (request.State.HasValue)  
        {  
            res = res.Where(s => (int) s.State == request.State.Value);  
        }  
  
        return await res.OrderByDescending(s => s.Id)  
            .ProjectTo<SplitBriefDto>(_mapper.ConfigurationProvider)  
            .PaginatedListAsync(request.PageNumber, request.PageSize);  
    }  
}
```

Файл GetSplitsWithPaginationQueryValidator.cs

```
using FluentValidation;  
using SplitDivider.Domain.Enums;  
  
namespace SplitDivider.Application.Splits.Queries.GetSplitsWithPagination;  
  
public class GetSplitsWithPaginationQueryValidator :  
    AbstractValidator<GetSplitsWithPaginationQuery>  
{  
    public GetSplitsWithPaginationQueryValidator()  
    {  
    }  
}
```

					КПІ.IT-9127.045440.03.12	Арк.
						14
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

{
    When(x => x.State.HasValue, () =>
    {
        RuleFor(x => (SplitState)x.State!.Value)
            .IsInEnum()
            .WithMessage("Split state must be a valid value");
    });

    RuleFor(x => x.PageNumber)
        .GreaterThanOrEqualTo(1).WithMessage("PageNumber at least greater
than or equal to 1.");

    RuleFor(x => x.PageSize)
        .GreaterThanOrEqualTo(1).WithMessage("PageSize at least greater
than or equal to 1.");
    }
}

```

Файл SplitBriefDto.cs

```

using AutoMapper;
using SplitDivider.Application.Common.Mappings;

namespace SplitDivider.Application.Splits.Queries.GetSplitsWithPagination;

public class SplitBriefDto : IMapFrom<Split>
{
    public int Id { get; init; }

    public string Name { get; set; }

    public int State { get; set; }

    public void Mapping(Profile profile)
    {
        profile.CreateMap<Split, SplitBriefDto>()
            .ForMember(nameof(Id), opts => opts.MapFrom(src => src.Id))
            .ForMember(nameof(Name), opts => opts.MapFrom(src => src.Name))
            .ForMember(nameof(State), opts => opts.MapFrom(src =>
src.State));
    }
}

```

Файл SplitActivatedEventHandler.cs

```

using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Application.Splits.Graph.Interfaces;
using SplitDivider.Domain.Enums;
using SplitDivider.Domain.Events;

namespace SplitDivider.Application.Splits.EventHandlers;

public class SplitActivatedEventHandler :
INotificationHandler<SplitActivatedEvent>
{
    private readonly ILogger<SplitActivatedEventHandler> _logger;

    private readonly IApplicationDbContext _context;

```

					КПІ.IT-9127.045440.03.12	Арк.
						15
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

private readonly IGraphBuilder _graphBuilder;

private readonly IGraphCutter _graphCutter;

public SplitActivatedEventHandler(
    ILogger<SplitActivatedEventHandler> logger,
    IApplicationDbContext context,
    IGraphBuilder graphBuilder,
    IGraphCutter graphCutter
)
{
    _logger = logger;
    _context = context;
    _graphBuilder = graphBuilder;
    _graphCutter = graphCutter;
}

public async Task Handle(SplitActivatedEvent notification,
    CancellationToken cancellationToken)
{
    _logger.LogInformation("Split activated: Id={Id}",
notification.Split.Id);

    var split = notification.Split;
    var usersInSplitQuery = _context.AppUsers.AsQueryable();

    if (split.Gender != null)
    {
        usersInSplitQuery = usersInSplitQuery.Where(u => u.Gender ==
split.Gender);
    }

    if (split.MinRegistrationDt != null)
    {
        usersInSplitQuery = usersInSplitQuery.Where(u => u.RegistrationDt
>= split.MinRegistrationDt);
    }

    if (split.CountryIds is { Count: > 0 })
    {
        usersInSplitQuery = usersInSplitQuery.Where(u =>
split.CountryIds.Contains(u.CountryId));
    }

    var usersInSplit = await usersInSplitQuery
        .Select(u => u.Id)
        .ToListAsync(cancellationToken);

    var graphDto = _graphBuilder.BuildGraph(split, usersInSplit);

    var groups = _graphCutter.CutSplitGraph(graphDto);

    foreach (var id in groups.first)
    {
        var userGroup = new UserSplit
        {
            UserId = id,
            SplitId = split.Id,
            Group = UserSplit.GroupControl
        };

        _context.UserSplits.Add(userGroup);
    }
}

```

					КПІ.IT-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		16

```

        foreach (var id in groups.second)
        {
            var userGroup = new UserSplit
            {
                UserId = id,
                SplitId = split.Id,
                Group = UserSplit.GroupTest
            };

            _context.UserSplits.Add(userGroup);
        }

        var splitEntity = await _context.Splits.FindAsync(split.Id);
        splitEntity!.State = SplitState.ReadyToTest;

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

Файл CreateSplitCommand.cs

```

using System.Globalization;
using MediatR;
using Shared.Values.ValueObjects;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Domain.Enums;

namespace SplitDivider.Application.Splits.Commands.CreateSplit;

public record CreateSplitCommand : IRequest<int>
{
    public string Name { get; init; }

    public IDictionary<string, int> ActionsWeights { get; set; }

    public IList<int>? CountryIds { get; set; }

    public string? Gender { get; init; }

    public string? MinRegDt { get; init; }
}

public class CreateSplitCommandHandler : IRequestHandler<CreateSplitCommand, int>
{
    private readonly IApplicationDbContext _context;

    public CreateSplitCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task<int> Handle(CreateSplitCommand request,
        CancellationToken cancellationToken)
    {
        var entity = new Split
        {
            Name = request.Name,
            State = SplitState.Created,
            ActionsWeights = request.ActionsWeights.ToDictionary(p =>
                InteractionType.From(p.Key), p => p.Value),
            CountryIds = request.CountryIds?.ToList(),

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.IT-9127.045440.03.12

Арк.

17

```

        Gender = request.Gender,
        MinRegistrationDt = request.MinRegDt != null ?
DateTime.Parse(request.MinRegDt, CultureInfo.InvariantCulture) : null
    };

    _context.Splits.Add(entity);

    await _context.SaveChangesAsync(cancellationToken);

    return entity.Id;
}
}

```

Файл CreateSplitCommandValidator.cs

```

using System.Globalization;
using FluentValidation;
using Shared.Values.ValueObjects;

namespace SplitDivider.Application.Splits.Commands.CreateSplit;

public class CreateSplitCommandValidator :
AbstractValidator<CreateSplitCommand>
{
    public CreateSplitCommandValidator()
    {
        RuleFor(s => s.Name)
            .MaximumLength(30)
            .MinimumLength(5)
            .NotEmpty()
            .WithMessage("Split name must have length between 5 and 30
characters");

        RuleFor(s => s.ActionsWeights)
            .NotEmpty()
            .WithMessage("Actions weights must not be empty");

        RuleForEach(s => s.ActionsWeights)
            .Must(p => InteractionType.IsSupported(p.Key) && p.Value >= 0);

        RuleFor(s => s.Gender)
            .Must(Gender.IsSupported!)
            .When(s => s.Gender != null);

        RuleFor(s => s.MinRegDt)
            .Must(dt => DateTime.TryParse(dt, CultureInfo.InvariantCulture,
out _))
            .When(s => s.MinRegDt != null);

        RuleFor(s => s.CountryIds)
            .Must(cIds => cIds!.All(id => id >= 0))
            .When(s => s.CountryIds != null);
    }
}

```

Файл UpdateSplitCommand.cs

```

using System.Globalization;
using MediatR;
using Shared.Values.ValueObjects;
using SplitDivider.Application.Common.Exceptions;

```

					КПІ.ІТ-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		18

```

using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Domain.Enums;
using InvalidOperationException =
SplitDivider.Application.Common.Exceptions.InvalidOperationException;

namespace SplitDivider.Application.Splits.Commands.UpdateSplit;

public record UpdateSplitCommand : IRequest
{
    public int? Id { get; set; }

    public string? Name { get; init; }

    public IDictionary<string, int>? ActionsWeights { get; set; }

    public IList<int>? CountryIds { get; set; }

    public string? Gender { get; init; }

    public string? MinRegDt { get; init; }
}

public class UpdateSplitCommandHandler : IRequestHandler<UpdateSplitCommand>
{
    private readonly IApplicationDbContext _context;

    public UpdateSplitCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(UpdateSplitCommand request, CancellationToken
cancellation_token)
    {
        var entity = await _context.Splits
            .FindAsync(request.Id, cancellation_token);

        if (entity == null)
        {
            throw new NotFoundException(nameof(Split), request.Id);
        }

        if (!Split.EditableStates.Contains(entity.State))
        {
            throw new InvalidOperationException("Invalid split status for
update: split can only be updated before activation");
        }

        if (request.Name != null)
        {
            entity.Name = request.Name;
        }

        if (request.ActionsWeights != null)
        {
            entity.ActionsWeights = request.ActionsWeights.ToDictionary(p =>
InteractionType.From(p.Key), p => p.Value);
        }

        if (request.Gender != null)
        {
            entity.Gender = request.Gender;
        }
    }
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

        if (request.CountryIds != null)
        {
            entity.CountryIds = request.CountryIds.ToList();
        }

        if (request.MinRegDt != null)
        {
            entity.MinRegistrationDt = DateTime.Parse(request.MinRegDt,
CultureInfo.InvariantCulture);
        }

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

Файл UpdateSplitCommandValidator.cs

```

using System.Globalization;
using FluentValidation;
using Shared.Values.ValueObjects;

namespace SplitDivider.Application.Splits.Commands.UpdateSplit;

public class UpdateSplitCommandValidator :
AbstractValidator<UpdateSplitCommand>
{
    public UpdateSplitCommandValidator()
    {
        RuleFor(s => s.Name)
            .MaximumLength(30)
            .MinimumLength(5)
            .NotEmpty()
            .WithMessage("Split name must have length between 5 and 30
characters")
            .When(s => s.Name != null);

        RuleFor(s => s.ActionsWeights)
            .NotEmpty()
            .WithMessage("Actions weights must not be empty")
            .Custom((list, context) =>
            {
                if (list!.Any(p => !InteractionType.IsSupported(p.Key) ||
p.Value < 0))
                {
                    context.AddFailure("Invalid action weight rule");
                }
            })
            .When(s => s.ActionsWeights != null);

        RuleFor(s => s.Gender)
            .Must(Gender.IsSupported!)
            .When(s => s.Gender != null);

        RuleFor(s => s.MinRegDt)
            .Must(dt => DateTime.TryParse(dt, CultureInfo.InvariantCulture,
out _))
            .When(s => s.MinRegDt != null);

        RuleFor(s => s.CountryIds)
            .Must(cIds => cIds!.All(id => id >= 0))
            .When(s => s.CountryIds != null);
    }
}

```

					КПІ.IT-9127.045440.03.12	Арк.
						20
Змін.	Арк.	№ докум.	Підп.	Дата.		

Файл DeleteSplitCommand.cs

```
using MediatR;
using SplitDivider.Application.Common.Exceptions;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.Application.Splits.Commands.DeleteSplit;

public record DeleteSplitCommand(int Id) : IRequest;

public class DeleteSplitCommandHandler : IRequestHandler<DeleteSplitCommand>
{
    private readonly IApplicationDbContext _context;

    public DeleteSplitCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(DeleteSplitCommand request, CancellationToken
cancellationTokn)
    {
        var entity = await _context.Splits
            .FindAsync(request.Id, cancellationTokn);

        if (entity == null)
        {
            throw new NotFoundException(nameof(Split), request.Id);
        }

        _context.Splits.Remove(entity);

        await _context.SaveChangesAsync(cancellationTokn);
    }
}
```

Файл CloseSplitCommand.cs

```
using MediatR;
using SplitDivider.Application.Common.Exceptions;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Domain.Enums;
using InvalidOperationException =
SplitDivider.Application.Common.Exceptions.InvalidOperationException;

namespace
SplitDivider.Application.Splits.Commands.LifecycleCommands.CloseSplit;

public record CloseSplitCommand(int Id) : IRequest;

public class CloseSplitCommandHandler : IRequestHandler<CloseSplitCommand>
{
    private readonly IApplicationDbContext _context;

    public CloseSplitCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }
}
```

					КПІ.IT-9127.045440.03.12	Арк.
						21
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

    public async Task Handle(CloseSplitCommand request, CancellationToken
cancellationToken)
    {
        var entity = await _context.Splits
            .FindAsync(request.Id, cancellationToken);

        if (entity == null)
        {
            throw new NotFoundException(nameof(Split), request.Id);
        }

        if (!Split.ClosableStates.Contains(entity.State))
        {
            throw new InvalidOperationException("Invalid split state for
closing");
        }

        entity.State = SplitState.Closed;

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

Файл SuspendSplitCommand.cs

```

using MediatR;
using SplitDivider.Application.Common.Exceptions;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Domain.Enums;
using Z.EntityFramework.Plus;
using InvalidOperationException =
SplitDivider.Application.Common.Exceptions.InvalidOperationException;

namespace
SplitDivider.Application.Splits.Commands.LifecycleCommands.SuspendSplit;

public record SuspendSplitCommand(int Id) : IRequest;

public class SuspendSplitCommandHandler :
    IRequestHandler<SuspendSplitCommand>
{
    private readonly IApplicationDbContext _context;

    public SuspendSplitCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(SuspendSplitCommand request, CancellationToken
cancellationToken)
    {
        var entity = await _context.Splits
            .FindAsync(request.Id, cancellationToken);

        if (entity == null)
        {
            throw new NotFoundException(nameof(Split), request.Id);
        }

        if (!Split.SuspendableStates.Contains(entity.State))
        {
            throw new InvalidOperationException("Split can only be suspended
after activation");
        }
    }
}

```

					КПІ.IT-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		22

```

    }

    entity.State = SplitState.Suspended;

    await _context.UserSplits
        .Where(ug => ug.SplitId == entity.Id)
        .DeleteAsync(cancellationToken);

    await _context.SaveChangesAsync(cancellationToken);
}
}

```

Файл ActivateSplitCommand.cs

```

using MediatR;
using SplitDivider.Application.Common.Exceptions;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Domain.Enums;
using SplitDivider.Domain.Events;
using InvalidOperationException =
SplitDivider.Application.Common.Exceptions.InvalidOperationException;

namespace
SplitDivider.Application.Splits.Commands.LifecycleCommands.ActivateSplit;

public record ActivateSplitCommand(int Id) : IRequest;

public class ActivateSplitCommandHandler :
IRequestHandler<ActivateSplitCommand>
{
    private readonly IApplicationDbContext _context;

    public ActivateSplitCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(ActivateSplitCommand request, CancellationToken
cancellationToken)
    {
        var entity = await _context.Splits
            .FindAsync(request.Id, cancellationToken);

        if (entity == null)
        {
            throw new NotFoundException(nameof(Split), request.Id);
        }

        if (!Split.ActivableStates.Contains(entity.State))
        {
            throw new InvalidOperationException("Split can only be activated
in Created or Suspended status");
        }

        entity.State = SplitState.Activated;

        entity.AddDomainEvent(new SplitActivatedEvent(entity));

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

					КПІ.IT-9127.045440.03.12	Арк.
						23
Змін.	Арк.	№ докум.	Підп.	Дата.		

Файл MinCutGraphCutter.cs

```
using SplitDivider.Application.Splits.Graph.Algorithms;
using SplitDivider.Application.Splits.Graph.Common;
using SplitDivider.Application.Splits.Graph.Interfaces;

namespace SplitDivider.Application.Splits.Graph;

public class MinCutGraphCutter : IGraphCutter
{
    public (List<int> first, List<int> second) CutSplitGraph(SplitGraphDto graphDto)
    {
        var minCut = new StoerWagnerMinCut<int>();

        var cut = minCut.ComputeMinCut(graphDto.Graph);

        var verticesFirstGroup = new List<int>(cut.First.GetVerticesIds());

        var verticesSecondGroup = new List<int>(cut.Second.GetVerticesIds());

        return (verticesFirstGroup, verticesSecondGroup);
    }
}
```

Файл GraphBuilder.cs

```
using System.Xml;
using QuikGraph;
using QuikGraph.Serialization;
using Shared.Values.ValueObjects;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Application.Splits.Graph.Algorithms;
using SplitDivider.Application.Splits.Graph.Common;
using SplitDivider.Application.Splits.Graph.Interfaces;
using SplitDivider.Application.Splits.Graph.Serialization.Common;

namespace SplitDivider.Application.Splits.Graph;

public class GraphBuilder : IGraphBuilder
{
    private readonly IApplicationDbContext _context;

    public GraphBuilder(IApplicationDbContext context)
    {
        _context = context;
    }

    public SplitGraphDto BuildGraph(Split split, List<int> userIds)
    {
        var qGraph = new UndirectedGraph<int, WeightedEdge<int, int>>();

        var graph = new Algorithms.Common.Graph<int, int>();
        var connections = new Dictionary<int, int>();

        var now = DateTime.Now;

        foreach (var uId in userIds)
        {
            graph.AddVertex(new Vertex<int>
            {
                Id = uId,
            })
        }
    }
}
```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

        Value = uId
    });
}

foreach (var userId in userIds)
{
    foreach (var contactId in userIds)
    {
        if (userId == contactId) continue;

        var relations = _context.Relations
            .Where(r => r.UserId == userId && r.ContactId ==
contactId)
            .ToList();

        var intCost = 1;

        if (relations.Count > 0)
        {
            var cost = 0.0;

            foreach (var relation in relations)
            {
                var interaction =
InteractionType.From(relation.Interaction);

                if (!split.ActionsWeights.Keys.Contains(interaction))
continue;

                var generalWeight =
split.ActionsWeights[interaction];

                var timeDiff = now.Subtract(relation.Dt).Hours / 24;

                var k = 1 / (1 + timeDiff * timeDiff);

                var weight = generalWeight * k;

                cost += weight;
            }

            intCost = (int) (cost * 10);
        }

        var edge = new WeightedEdge<int, int>(userId, contactId,
intCost);

        qGraph.AddVerticesAndEdge(edge);

        graph.AddEdge(userId, new Algorithms.Edge<int>
        {
            DestinationVertexId = contactId,
            Value = intCost
        });

        if (connections.Keys.Contains(userId)) connections[userId] +=
1;
        else connections.Add(userId, 1);

        if (connections.Keys.Contains(contactId))
connections[contactId] += 1;
        else connections.Add(contactId, 1);
    }
}

```

```

        SaveGraph(split.Id, qGraph);

        return new SplitGraphDto
        {
            Graph = graph,
            Connections = connections,
        };
    }

    private void SaveGraph(int splitId, UndirectedGraph<int,
WeightedEdge<int, int>> graph)
    {
        var filename = $"split{splitId}.graphml";
        var fi = new FileInfo(@"Visualization/" + filename);

        using (var fs = fi.Open(FileMode.OpenOrCreate, FileAccess.Write,
FileShare.Read))
        {
            using (var xmlWriter = XmlWriter.Create(fs))
            {
                graph.SerializeToGraphML<int, WeightedEdge<int, int>,
UndirectedGraph<int, WeightedEdge<int, int>>>(xmlWriter);
            }
        }
    }
}

```

Файл SplitGraphDto.cs

```

using SplitDivider.Application.Splits.Graph.Algorithms.Common;

namespace SplitDivider.Application.Splits.Graph.Common;

public class SplitGraphDto
{
    public Dictionary<int, int> Connections { get; set; }

    public Graph<int, int> Graph { get; set; }
}

```

Файл State.cs

```

namespace SplitDivider.Application.Splits.Graph.Algorithms.Structs;

public class State
{
    public State()
    {
        partition = new HashSet<int>();
        bestPartition = null;
        bestCut = null;
    }

    public HashSet<int> partition;
    public HashSet<int>? bestPartition;
    public Cut? bestCut;
}

```

Файл CreateRelationCommand.cs

```

using MediatR;
using SplitDivider.Application.Common.Interfaces;

```

					КПІ.IT-9127.045440.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		26

```

namespace SplitDivider.Application.Relations.Commands.CreateRelation;

public record CreateRelationCommand : IRequest
{
    public int UserId { get; init; }

    public int ContactId { get; init; }

    public string InteractionType { get; init; }
}

public class CreateRelationCommandHanler :
    IRequestHandler<CreateRelationCommand>
{
    private readonly IApplicationDbContext _context;

    public CreateRelationCommandHanler(IApplicationDbContext context)
    {
        _context = context;
    }

    public async Task Handle(CreateRelationCommand request, CancellationToken
cancellationTokn)
    {
        var entity = new Relation
        {
            UserId = request.UserId,
            ContactId = request.ContactId,
            Interaction = request.InteractionType,
            Dt = DateTime.Now
        };

        _context.Relations.Add(entity);

        await _context.SaveChangesAsync(cancellationToken);
    }
}

```

Файл CreateRelationCommandValidator.cs

```

using FluentValidation;
using Shared.Values.ValueObjects;

namespace SplitDivider.Application.Relations.Commands.CreateRelation;

public class CreateRelationCommandValidator :
    AbstractValidator<CreateRelationCommand>
{
    public CreateRelationCommandValidator()
    {
        RuleFor(u => u.InteractionType)
            .Must(InteractionType.IsSupported);
    }
}

```

Файл IEventBusReceiver.cs

```

namespace SplitDivider.Application.Common.Interfaces;

public interface IEventBusReceiver : IDisposable

```

					КПІ.ІТ-9127.045440.03.12	Арк.
						27
Змін.	Арк.	№ докум.	Підп.	Дата.		

```
{
    public void StartReceiving();
}
```

Файл IEventBusEventHandler.cs

```
using Shared.Values.EventBus.Common;

namespace SplitDivider.Application.Common.Interfaces;

public interface IEventBusEventHandler
{
    public Task Handle(BaseEvent eEvent);
}
```

Файл IApplicationDbContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace SplitDivider.Application.Common.Interfaces;

public interface IApplicationDbContext
{
    DbSet<User> AppUsers { get; }

    DbSet<Relation> Relations { get; }

    DbSet<Split> Splits { get; }

    DbSet<UserSplit> UserSplits { get; }

    Task<int> SaveChangesAsync(CancellationToken cancellationToken);
}
```

Файл ConfigureServices.cs

```
using System.Reflection;
using AutoMapper.Internal;
using FluentValidation;
using MediatR;
using Microsoft.Extensions.DependencyInjection;
using SplitDivider.Application.Common.Behaviours;
using SplitDivider.Application.Splits.Graph;
using SplitDivider.Application.Splits.Graph.Interfaces;
using SplitDivider.Application.Users.Commands.CreateUser;

namespace SplitDivider.Application;

public static class ConfigureServices
{
    public static IServiceCollection AddApplicationServices(this
    IServiceCollection services)
    {
        services.AddScoped<IGraphBuilder, GraphBuilder>();
        services.AddScoped<IGraphCutter, MinCutGraphCutter>();

        services.AddAutoMapper(cfg => cfg.Internal().MethodMappingEnabled =
        false, Assembly.GetExecutingAssembly());
        services.AddValidatorsFromAssembly(Assembly.GetExecutingAssembly(),
        ServiceLifetime.Singleton);
    }
}
```

					КП.ІТ-9127.045440.03.12	Арк.
						28
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        services.AddMediatR(cfg => {

cfg.RegisterServicesFromAssembly(typeof(CreateUserCommandHandler).Assembly);
        cfg.AddBehavior(typeof(IPipelineBehavior<,>),
typeof(UnhandledExceptionBehaviour<,>));
        cfg.AddBehavior(typeof(IPipelineBehavior<,>),
typeof(ValidationBehaviour<,>));
        cfg.AddBehavior(typeof(IPipelineBehavior<,>),
typeof(PerformanceBehaviour<,>));
        });

        return services;
    }
}

```

Файл LoginModel.cs

```

using System.ComponentModel.DataAnnotations;

namespace SplitDivider.WebApp.Authorization.Models;

public class LoginModel
{
    [Required]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    public bool RememberMe { get; set; }
}

```

Файл RegistrationModel.cs

```

using System.ComponentModel.DataAnnotations;

namespace SplitDivider.WebApp.Authorization.Models;

public class RegistrationModel
{
    [Required]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [Required]
    [Compare("Password", ErrorMessage = "passwords mismatch")]
    [DataType(DataType.Password)]
    public string PasswordConfirmation { get; set; }
}

```

Файл Receiver.cs

```

using System.Text;
using RabbitMQ.Client;
using RabbitMQ.Client.Events;
using Shared.Values.EventBus.Common;

```

					КПІ.IT-9127.045440.03.12	Арк.
						29
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.WebApp.EventBus;

public class Receiver : IEventBusReceiver, IDisposable
{
    private IAsyncConnectionFactory _connectionFactory;

    private IConnection _connection;

    private IModel _channel;

    private ISet<BaseEvent e, IEventBusEventHandler handler>
    _eventSubscriptions = new HashSet<BaseEvent e, IEventBusEventHandler
    handler>();

    public Receiver(IAsyncConnectionFactory connFactory)
    {
        _connectionFactory = connFactory;
    }

    public void SubscribeToEvent(BaseEvent e, IEventBusEventHandler handler)
    {
        _eventSubscriptions.Add((e, handler));
    }

    public void StartReceiving()
    {
        _connection = _connectionFactory.CreateConnection();
        _channel = _connection.CreateModel();

        _channel.ExchangeDeclare(
            exchange: "interactor_direct_msg",
            type: "direct",
            durable: true);

        foreach (var subscription in _eventSubscriptions)
        {
            var eventType = subscription.e.GetEventType();

            _channel.QueueDeclare(
                queue: eventType,
                durable: false,
                exclusive: false,
                autoDelete: false,
                arguments: null);

            _channel.QueueBind(
                queue: eventType,
                exchange: "interactor_direct_msg",
                routingKey: eventType);

            var consumer = new AsyncEventingBasicConsumer(_channel);
            consumer.Received += async (model, ea) =>
            {
                var body = ea.Body.ToArray();
                var message = Encoding.UTF8.GetString(body);

                var eventData = subscription.e.FromEventData(message);

                await subscription.handler.Handle(eventData);
            };

            _channel.BasicConsume(

```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

        queue: eventType,
        autoAck: true,
        consumer: consumer);
    }
}

public void Dispose()
{
    _connection.Dispose();
    _channel.Dispose();
}
}

```

Файл UserChangedEventHandler.cs

```

using MediatR;
using Shared.Values.Enums;
using Shared.Values.EventBus.Common;
using Shared.Values.EventBus.UserEvents;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Application.Users.Commands.UpdateUser;

namespace SplitDivider.WebApp.EventBus.Handlers.UserEvents;

public class UserChangedEventHandler : IEventBusEventHandler
{
    private IServiceProvider _services;

    public UserChangedEventHandler(IServiceProvider services)
    {
        _services = services;
    }

    public async Task Handle(BaseEvent eEvent)
    {
        var eventData = (UserChangedEvent)eEvent;

        using var scope = _services.CreateScope();
        var mediator = scope.ServiceProvider.GetService<ISender>();

        if (mediator == null)
        {
            throw new Exception("mediator is not found");
        }

        await mediator.Send(new UpdateUserCommand
        {
            Id = eventData.Id,
            CountryId = eventData.CountryId,
            Gender = eventData.Gender,
            State = eventData.State.HasValue ? (UserState)eventData.State :
null
        });
    }
}

```

Файл UserDeletedEventHandler.cs

```

using MediatR;
using Shared.Values.EventBus.Common;
using Shared.Values.EventBus.UserEvents;

```

					КПІ.IT-9127.045440.03.12	Арк.
						31
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Application.Users.Commands.DeleteUser;

namespace SplitDivider.WebApp.EventBus.Handlers.UserEvents;

public class UserDeletedEventHandler : IEventBusEventHandler
{
    private IServiceProvider _services;

    public UserDeletedEventHandler(IServiceProvider services)
    {
        _services = services;
    }

    public async Task Handle(BaseEvent eEvent)
    {
        var eventData = (UserDeletedEvent)eEvent;

        using var scope = _services.CreateScope();
        var mediator = scope.ServiceProvider.GetService<ISender>();

        if (mediator == null)
        {
            throw new Exception("mediator is not found");
        }

        await mediator.Send(new DeleteUserCommand(eventData.Id));
    }
}

```

Файл UserRegisteredEventHandler.cs

```

using MediatR;
using Shared.Values.EventBus.Common;
using Shared.Values.EventBus.UserEvents;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Application.Users.Commands.CreateUser;

namespace SplitDivider.WebApp.EventBus.Handlers.UserEvents;

public class UserRegisteredEventHandler : IEventBusEventHandler
{
    private IServiceProvider _services;

    public UserRegisteredEventHandler(IServiceProvider services)
    {
        _services = services;
    }

    public async Task Handle(BaseEvent eEvent)
    {
        var eventData = (UserRegisteredEvent)eEvent;

        using var scope = _services.CreateScope();
        var mediator = scope.ServiceProvider.GetService<ISender>();

        if (mediator == null)
        {
            throw new Exception("mediator is not found");
        }

        await mediator.Send(new CreateUserCommand
        {

```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

        Id = eventData.Id,
        CountryId = eventData.CountryId,
        Gender = eventData.Gender,
        RegDt = eventData.RegDt
    });
}
}

```

Файл UsersInteractedEventHandler.cs

```

using MediatR;
using Shared.Values.EventBus.Common;
using Shared.Values.EventBus.InteractionEvents;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Application.Relations.Commands.CreateRelation;

namespace SplitDivider.WebApp.EventBus.Handlers.InteractionEvents;

public class UsersInteractedEventHandler : IEventBusEventHandler
{
    private IServiceProvider _services;

    public UsersInteractedEventHandler(IServiceProvider services)
    {
        _services = services;
    }

    public async Task Handle(BaseEvent eEvent)
    {
        var eventData = (UsersInteractedEvent)eEvent;

        using var scope = _services.CreateScope();
        var mediator = scope.ServiceProvider.GetService<ISender>();

        if (mediator == null)
        {
            throw new Exception("mediator is not found");
        }

        await mediator.Send(new CreateRelationCommand
        {
            UserId = eventData.UserId,
            ContactId = eventData.ContactId,
            InteractionType = eventData.InteractionType
        });
    }
}

```

Файл CurrentUserService.cs

```

using System.Security.Claims;
using SplitDivider.Application.Common.Interfaces;

namespace SplitDivider.WebApp.Services;

public class CurrentUserService : ICurrentUserService
{
    private readonly IHttpContextAccessor _httpContextAccessor;

    public CurrentUserService(IHttpContextAccessor httpContextAccessor)
    {
    }
}

```

					КПІ.ІТ-9127.045440.03.12	Арк.
						33
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        _httpContextAccessor = httpContextAccessor;
    }

    public string? UserId =>
        _httpContextAccessor.HttpContext?.User?.FindFirstValue(ClaimTypes.NameIdentifier);
}

```

Файл AdminController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.Infrastructure.Identity;

namespace SplitDivider.WebApp.Controllers;

[Authorize(Roles = "Administrator")]
public class AdminController : ApiControllerBase
{
    private readonly UserManager<ApplicationUser> _userManager;

    private readonly IIdentityService _identityService;

    public AdminController(
        IIdentityService identityService,
        UserManager<ApplicationUser> userManager)
    {
        _identityService = identityService;
        _userManager = userManager;
    }

    [HttpGet("users")]
    public async Task<List<ApplicationUser>> GetUsers()
    {
        return await _userManager.Users.ToListAsync();
    }

    [HttpDelete("user/{userId}/role/{roleId}")]
    public async Task<bool> DeleteRole(string userId, string roleId)
    {
        return await _identityService.DeleteRoleForUser(userId, roleId);
    }

    [HttpPut("user/{userId}/role/{roleId}")]
    public async Task<bool> AddRole(string userId, string roleId)
    {
        return await _identityService.AddRoleForUser(userId, roleId);
    }
}

```

Файл ApiBaseController.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using SplitDivider.WebApp.Filters;

namespace SplitDivider.WebApp.Controllers;

```

					КПІ.IT-9127.045440.03.12	Арк.
						34
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

[ApiController]
[ApiExceptionHandler]
[Route("api/[controller]")]
public abstract class ApiControllerBase : ControllerBase
{
    private ISender? _mediator;

    protected ISender Mediator => _mediator ??=
HttpContext.RequestServices.GetRequiredService<ISender>();
}

```

Файл AuthController.cs

```

using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using SplitDivider.Infrastructure.Identity;
using SplitDivider.WebApp.Authorization.Models;

namespace SplitDivider.WebApp.Controllers;

public class AuthController : ApiControllerBase
{
    private readonly UserManager<ApplicationUser> _userManager;

    private readonly SignInManager<ApplicationUser> _signInManager;

    public AuthController(
        UserManager<ApplicationUser> userManager,
        SignInManager<ApplicationUser> signInManager)
    {
        _userManager = userManager;
        _signInManager = signInManager;
    }

    [HttpPost("signup")]
    public async Task<Response?> Register(RegistrationModel model)
    {
        if (ModelState.IsValid)
        {
            var user = new ApplicationUser { Email = model.Email, UserName =
model.Email };

            var result = await _userManager.CreateAsync(user,
model.Password);

            if (result.Succeeded)
            {
                await _signInManager.SignInAsync(user, true);

                return new Response
                {
                    Result = user
                };
            }

            return new Response
            {
                Errors = result.Errors,
            };
        }

        return null;
    }
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.IT-9127.045440.03.12

Арк.

35

```

[HttpPost("login")]
public async Task<bool> Login(LoginModel model)
{
    if (ModelState.IsValid)
    {
        var result = await _signInManager.PasswordSignInAsync(
            model.Email,
            model.Password,
            model.RememberMe,
            false);

        return result.Succeeded;
    }

    return false;
}

[HttpPost("signout")]
public async Task Logout()
{
    await _signInManager.SignOutAsync();
}
}

```

Файл SplitsController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SplitDivider.Application.Common.Models;
using SplitDivider.Application.Splits.Commands.CreateSplit;
using SplitDivider.Application.Splits.Commands.DeleteSplit;
using
SplitDivider.Application.Splits.Commands.LifecycleCommands.ActivateSplit;
using SplitDivider.Application.Splits.Commands.LifecycleCommands.CloseSplit;
using
SplitDivider.Application.Splits.Commands.LifecycleCommands.SuspendSplit;
using SplitDivider.Application.Splits.Commands.UpdateSplit;
using SplitDivider.Application.Splits.Queries.GetSplit;
using SplitDivider.Application.Splits.Queries.GetSplitsWithPagination;
using SplitDivider.Application.Users.Queries.GetSplitUsers;

namespace SplitDivider.WebApp.Controllers;

[Authorize(Roles = "Experimenter")]
public class SplitsController : ApiControllerBase
{
    [HttpGet]
    public async Task<ActionResult<PaginatedList<SplitBriefDto>>>
GetSplitsWithPagination([FromQuery] GetSplitsWithPaginationQuery query)
    {
        return await Mediator.Send(query);
    }

    [HttpGet("{id}")]
    [ProducesResponseType(StatusCodes.Status204NoContent)]
    public async Task<ActionResult<SplitDto>> Get(int id)
    {
        return await Mediator.Send(new GetSplitQuery(id));
    }

    [HttpGet("{id}/users")]
    public async Task<IReadOnlyCollection<SplitUserDto>> GetSplitUsers(int

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

id)
{
    return await Mediator.Send(new GetSplitUsersQuery(id));
}

[HttpPost]
public async Task<ActionResult<int>> Create(CreateSplitCommand command)
{
    return await Mediator.Send(command);
}

[HttpDelete("{id}")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesDefaultResponseType]
public async Task<IActionResult> Delete(int id)
{
    await Mediator.Send(new DeleteSplitCommand(id));

    return NoContent();
}

[HttpPatch("{id}")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesDefaultResponseType]
public async Task<IActionResult> Update(int id, UpdateSplitCommand
command)
{
    command.Id = id;

    await Mediator.Send(command);

    return NoContent();
}

[HttpPatch("{id}/activate")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesDefaultResponseType]
public async Task<IActionResult> Activate(int id)
{
    await Mediator.Send(new ActivateSplitCommand(id));

    return NoContent();
}

[HttpPatch("{id}/suspend")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesDefaultResponseType]
public async Task<IActionResult> Suspend(int id)
{
    await Mediator.Send(new SuspendSplitCommand(id));

    return NoContent();
}

[HttpPatch("{id}/close")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesDefaultResponseType]
public async Task<IActionResult> Close(int id)
{
    await Mediator.Send(new CloseSplitCommand(id));

    return NoContent();
}

```

```
}
}
```

Файл UsersController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SplitDivider.Application.Users.Queries.GetUserSplits;

namespace SplitDivider.WebApp.Controllers;

[Authorize]
public class UsersController : ApiControllerBase
{
    [HttpGet("{id}/splits")]
    public async Task<IReadOnlyCollection<UserSplitDto>> GetUserSplits(int
id)
    {
        return await Mediator.Send(new GetUserSplitsQuery(id));
    }
}
```

Файл ConfigureServices.cs

```
using Microsoft.AspNetCore.Mvc;
using RabbitMQ.Client;
using Shared.Values.EventBus.InteractionEvents;
using Shared.Values.EventBus.UserEvents;
using SplitDivider.Application.Common.Interfaces;
using SplitDivider.WebApp.EventBus;
using SplitDivider.WebApp.EventBus.Handlers.InteractionEvents;
using SplitDivider.WebApp.EventBus.Handlers.UserEvents;
using SplitDivider.WebApp.Services;
using ZymLabs.NSwag.FluentValidation;

namespace SplitDivider.WebApp;

public static class ConfigureServices
{
    public static IServiceCollection AddWebAppServices(this
IServiceCollection services)
    {
        // services.AddHttpsRedirection(opts =>
        // {
        //     opts.HttpsPort = 5001;
        // });

        services.AddDatabaseDeveloperPageExceptionFilter();

        services.AddScoped<ICurrentUserService, CurrentUserService>();

        services.AddHttpContextAccessor();

        services.AddHealthChecks();
        //.AddDbContextCheck<ApplicationDbContext>();

        services.AddControllersWithViews();

        services.AddScoped<FluentValidationSchemaProcessor>(provider =>
        {
            var validationRules =
```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

provider.GetService<IEnumerable<FluentValidationRule>>();
    var loggerFactory = provider.GetService<ILoggerFactory>();

    return new FluentValidationSchemaProcessor(provider,
validationRules, loggerFactory);
});

// Customise default API behaviour
services.Configure<ApiBehaviorOptions>(options =>
options.SuppressModelStateInvalidFilter = true);

services.AddOpenApiDocument((configure, serviceProvider) =>
{
    var fluentValidationSchemaProcessor =
serviceProvider.CreateScope().ServiceProvider.GetRequiredService<FluentValida
tionSchemaProcessor>();

    // Add the fluent validations schema processor
configure.SchemaProcessors.Add(fluentValidationSchemaProcessor);

configure.Title = "Split Divider API";
configure.Description = "Split-divider service API
documentation";
});

return services;
}

public static Receiver
ConfigureEventSubscriptionReceiver(IServiceProvider serviceProvider)
{
    var factory = new ConnectionFactory
    {
        HostName = "localhost",
        DispatchConsumersAsync = true
    };

    var receiver = new Receiver(factory);

    receiver.SubscribeToEvent(new UserRegisteredEvent(), new
UserRegisteredEventHandler(serviceProvider));
    receiver.SubscribeToEvent(new UserChangedEvent(), new
UserChangedEventHandler(serviceProvider));
    receiver.SubscribeToEvent(new UserDeletedEvent(), new
UserDeletedEventHandler(serviceProvider));
    receiver.SubscribeToEvent(new UsersInteractedEvent(), new
UsersInteractedEventHandler(serviceProvider));

    return receiver;
}
}

```

Файл Program.cs

```

using SplitDivider.Application;
using SplitDivider.Infrastructure;
using SplitDivider.Infrastructure.Persistence;
using SplitDivider.WebApp;
using ConfigureServices = SplitDivider.WebApp.ConfigureServices;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

```

					КПІ.IT-9127.045440.03.12	Арк.
						39
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

builder.Services.AddApplicationServices();
builder.Services.AddInfrastructureServices(builder.Configuration);
builder.Services.AddWebAppServices();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
    app.UseMigrationsEndPoint();

    // Initialise and seed database
    using (var scope = app.Services.CreateScope())
    {
        var initialiser =
scope.ServiceProvider.GetRequiredService<ApplicationDbContextInitializer>();
        await initialiser.InitialiseAsync();
        await initialiser.SeedAsync();
    }
}
else
{
    // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHealthChecks("/health");
//app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseOpenApi(configure =>
{
    configure.Path = "/api/specification.json";
});
app.UseSwaggerUi3(settings =>
{
    settings.Path = "/api";
    settings.DocumentPath = "/api/specification.json";
});

app.UseRouting();

//app.UseIdentityServer();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller}/{action=Index}/{id?}");

app.MapRazorPages();

app.MapFallbackToFile("index.html");

var receiver =
ConfigureServices.ConfigureEventSubscriptionReceiver(app.Services);
receiver.StartReceiving();

app.Lifetime.ApplicationStopped.Register(receiver.Dispose);

app.Run();

```

					КПІ.IT-9127.045440.03.12	Арк.
						40
Змін.	Арк.	№ докум.	Підп.	Дата.		

Файл appsettings.json

```
{
  "UseInMemoryDatabase": false,
  "ConnectionStrings": {
    "DefaultConnection": "server=localhost;database=split-divider-
db;user=root;password=root"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.AspNetCore": "Information"
    }
  },
  "IdentityServer": {
    "Clients": {
      "Interactor.SplitDivider.WebApp": {
        "Profile": "IdentityServerSPA"
      }
    }
  },
  "AllowedHosts": "*"
}
```

Файл User.cs

```
using Shared.Values.Enums;
using Shared.Values.ValueObjects;

namespace Interactor.Domain.Entities;

public class User
{
    private Gender? _gender;

    public int Id { get; set; }

    public int CountryId { get; set; }

    public DateTime RegistrationDt { get; set; }

    public UserState State { get; set; }

    public string Gender {
        get => _gender?.Name ?? "undefined";
        set => _gender = Shared.Values.ValueObjects.Gender.From(value);
    }
}
```

Файл UserConfiguration.cs

```
using Interactor.Domain.Entities;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Interactor.Infrastructure.Persistence.Configurations;

public class UserConfiguration : IEntityTypeConfiguration<User>
```

					КПІ.IT-9127.045440.03.12	Арк.
						41
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.Property(u => u.RegistrationDt)
            .ValueGeneratedOnAdd()
            .IsRequired();

        builder.Property(u => u.Gender)
            .IsRequired();

        builder.Property(u => u.CountryId)
            .IsRequired();

        builder.Property(u => u.State)
            .IsRequired();
    }
}

```

Файл IEventBusPublisher.cs

```

using Shared.Values.EventBus.Common;

namespace Interactor.Application.Common.Interfaces;

public interface IEventBusPublisher
{
    public void Send(BaseEvent eEvent);
}

```

Файл InteractionService.cs

```

using Interactor.Application.Common.Interfaces;
using Microsoft.EntityFrameworkCore;
using Shared.Values.EventBus.InteractionEvents;
using Shared.Values.ValueObjects;

namespace Interactor.Application.Common.Services;

public class InteractionService : IInteractionService
{
    private IApplicationDbContext _context;

    private IEventBusPublisher _publisher;

    public InteractionService(
        IApplicationDbContext context,
        IEventBusPublisher publisher)
    {
        _context = context;
        _publisher = publisher;
    }

    public async Task HandleRandomUsersInteracted()
    {
        var minId = await _context.Users.MinAsync(u => u.Id);
        var maxId = await _context.Users.MaxAsync(u => u.Id);

        var random = new Random();

        var userId = random.Next(minId, maxId + 1);
    }
}

```

					КПІ.IT-9127.045440.03.12	Арк.
						42
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        var contactId = random.Next(minId, maxId + 1);

        var interactionId = random.Next(0,
InteractionType.SupportedActions.Count());

        var eEvent = new UsersInteractedEvent
        {
            UserId = userId,
            ContactId = contactId,
            Dt = DateTime.Now.ToString(),
            Interaction =
InteractionType.SupportedActions.ElementAt(interactionId).Name
        };

        _publisher.Send(eEvent);
    }
}

```

Файл UserService.cs

```

using System.Globalization;
using Interactor.Application.Common.Interfaces;
using Interactor.Domain.Entities;
using Microsoft.EntityFrameworkCore;
using Shared.Values.Enums;
using Shared.Values.EventBus.Common;
using Shared.Values.EventBus.UserEvents;
using Shared.Values.ValueObjects;

namespace Interactor.Application.Common.Services;

public class UserService : IUserService
{
    private IApplicationDbContext _context;

    private IEventBusPublisher _publisher;

    public UserService(
        IApplicationDbContext context,
        IEventBusPublisher publisher)
    {
        _context = context;
        _publisher = publisher;
    }

    public async Task RegisterRandomUser()
    {
        var random = new Random();

        var genderId = random.Next(1, 3);
        var gender = genderId == 1 ? Gender.Male : Gender.Female;

        var countryId = random.Next(1, 50);

        var user = new User
        {
            CountryId = countryId,
            Gender = gender,
            RegistrationDt = DateTime.Now,

            State = UserState.Activated
        };
    }
}

```

```

        var entity = await _context.Users.AddAsync(user);

        await _context.SaveChangesAsync(default);

        var eEvent = new UserRegisteredEvent
        {
            Id = entity.Entity.Id,
            CountryId = user.CountryId,
            Gender = user.Gender,
            RegDt =
user.RegistrationDt.ToString(CultureInfo.InvariantCulture)
        };

        _publisher.Send(eEvent);

        Thread.Sleep(500);

        var activationEvent = new UserChangedEvent
        {
            Id = entity.Entity.Id,
            State = (int)UserState.Activated
        };

        _publisher.Send(activationEvent);
    }

    public async Task BanRandomUser()
    {
        var minId = await _context.Users.MinAsync(u => u.Id);
        var maxId = await _context.Users.MaxAsync(u => u.Id);

        var random = new Random();

        var userId = random.Next(minId, maxId + 1);

        var user = await _context.Users.FindAsync(userId);
        if (user == null)
        {
            return;
        }

        user.State = UserState.Banned;

        var eEvent = new UserChangedEvent
        {
            Id = userId,
            State = (int)UserState.Banned
        };

        _publisher.Send(eEvent);

        await _context.SaveChangesAsync(default);
    }
}

```

Файл Publisher.cs

```

using System.Text;
using Interactor.Application.Common.Interfaces;
using RabbitMQ.Client;
using Shared.Values.EventBus.Common;

namespace Interactor.Application.EventBus;

```

					КПІ.IT-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		44

```

public class Publisher : IEventBusPublisher
{
    private IAsyncConnectionFactory _connectionFactory;

    public Publisher(IAsyncConnectionFactory connFactory)
    {
        _connectionFactory = connFactory;
    }

    public void Send(BaseEvent eEvent)
    {
        using var connection = _connectionFactory.CreateConnection();
        using var channel = connection.CreateModel();

        channel.ExchangeDeclare(
            exchange: "interactor_direct_msg",
            type: "direct",
            durable: true);

        var body = Encoding.UTF8.GetBytes(eEvent.ToEventData());

        channel.BasicPublish(
            exchange: "interactor_direct_msg",
            routingKey: eEvent.GetEventType(),
            basicProperties: null,
            body: body);
    }
}

```

Файл UsersController.cs

```

using Interactor.Application.Common.Interfaces;
using Microsoft.AspNetCore.Mvc;

namespace Interactor.WebApp.Controllers;

[ApiController]
[Route("api/[controller]")]
public class UsersController : ControllerBase
{
    private IUserService _userSvc;

    private IInteractionService _interactionSvc;

    public UsersController(
        IUserService userSvc,
        IInteractionService interactionSvc)
    {
        _userSvc = userSvc;
        _interactionSvc = interactionSvc;
    }

    [HttpPost("register")]
    public async Task SendRandomUserRegistered()
    {
        await _userSvc.RegisterRandomUser();
    }

    [HttpPost("ban")]
    public async Task SendRandomUserBanned()
    {
        await _userSvc.BanRandomUser();
    }
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.IT-9127.045440.03.12

Арк.

45

```

    }

    [HttpPost("interacted")]
    public async Task SendRandomUsersInteracted()
    {
        await _interactionSvc.HandleRandomUsersInteracted();
    }
}

```

Файл Program.cs

```

using Interactor.Application;
using Interactor.Infrastructure;
using Interactor.WebApp;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddApplicationServices();
builder.Services.AddInfrastructureServices(builder.Configuration);
builder.Services.AddWebAppServices();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
    app.UseMigrationsEndPoint();
}
else
{
    // The default HSTS value is 30 days. You may want to change this for
    // production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHealthChecks("/health");
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseOpenApi(configure =>
{
    configure.Path = "/api/specification.json";
});
app.UseSwaggerUi3(settings =>
{
    settings.Path = "/api";
    settings.DocumentPath = "/api/specification.json";
});

app.UseRouting();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller}/{action=Index}/{id?}");

app.MapRazorPages();

app.MapFallbackToFile("index.html");

app.Run();

```

					КПІ.IT-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		46

Файл appsettings.json

```
{
  "UseInMemoryDatabase": false,
  "ConnectionStrings": {
    "DefaultConnection": "server=localhost;database=interactor-
db;user=root;password=root"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.AspNetCore": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Файл UserState.cs

```
namespace Shared.Values.Enums;

public enum UserState
{
    Registered = 0,
    Activated,
    Verified,
    Banned,
    Deleted
}
```

Файл Gender.cs

```
namespace Shared.Values.ValueObjects;

public sealed class Gender : ValueObjectWithName
{
    private Gender(string name) : base(name)
    {
    }

    public static Gender From(string name)
    {
        var gender = new Gender(name);

        if (!SupportedGenders.Contains(gender))
        {
            throw new UnsupportedGenderException(name);
        }

        return gender;
    }

    public static bool IsSupported(string name)
    {
        var action = new Gender(name);

        return SupportedGenders.Contains(action);
    }
}
```

					КПІ.IT-9127.045440.03.12	Арк.
						47
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

public static implicit operator string(Gender gender)
{
    return gender.ToString();
}

public static explicit operator Gender(string name)
{
    return From(name);
}

public static Gender Male = new("male");

public static Gender Female = new("female");

private static IEnumerable<Gender> SupportedGenders
{
    get
    {
        yield return Male;
        yield return Female;
    }
}
}

```

Файл InteractionType.cs

```

namespace Shared.Values.ValueObjects;

public sealed class InteractionType : ValueObjectWithName
{
    private InteractionType(string name) : base(name)
    {
    }

    public static InteractionType From(string name)
    {
        var action = new InteractionType(name);

        if (!SupportedActions.Contains(action))
        {
            throw new UnsupportedInteractionTypeException(name);
        }

        return action;
    }

    public static bool IsSupported(string name)
    {
        var action = new InteractionType(name);

        return SupportedActions.Contains(action);
    }

    public static implicit operator string(InteractionType interactionType)
    {
        return interactionType.ToString();
    }

    public static explicit operator InteractionType(string name)
    {
        return From(name);
    }
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІТ-9127.045440.03.12

Арк.

48

```
public static InteractionType Dialog = new("dialog");

public static InteractionType Like = new("like");

public static InteractionType Hide = new("hide");

public static InteractionType VideoCall = new("video-call");

public static IEnumerable<InteractionType> SupportedActions
{
    get
    {
        yield return Dialog;
        yield return Like;
        yield return Hide;
        yield return VideoCall;
    }
}
```

Файл **UnsupportedGenderException.cs**

```
namespace Shared.Values.Exceptions;

public class UnsupportedGenderException : Exception
{
    public UnsupportedGenderException(string name) : base($"Gender \"{name}\" is unsupported.")
    {
    }
}
```

Файл **UnsupportedInteractionTypeException.cs**

```
namespace Shared.Values.Exceptions;

public class UnsupportedInteractionTypeException : Exception
{
    public UnsupportedInteractionTypeException(string name) :
base($"InteractionType \"{name}\" is unsupported.")
    {
    }
}
```

Файл **BaseEvent.cs**

```
using Newtonsoft.Json;

namespace Shared.Values.EventBus.Common;

[JsonObject(MemberSerialization.OptOut)]
public abstract class BaseEvent
{
    public abstract string GetEventType();

    public string ToEventData()
    {
        return JsonConvert.SerializeObject(this);
    }
}
```

```

    public abstract BaseEvent FromEventData(string eventData);
}

```

Файл UsersInteractedEvent.cs

```

using Newtonsoft.Json;
using Shared.Values.EventBus.Common;

namespace Shared.Values.EventBus.InteractionEvents;

[JsonObject(MemberSerialization.OptOut)]
public class UsersInteractedEvent : BaseEvent
{
    public int UserId { get; set; }

    public int ContactId { get; set; }

    public string Interaction { get; set; }

    public string Dt { get; set; }

    public override UsersInteractedEvent FromEventData(string eventData)
    {
        var obj =
            JsonConvert.DeserializeObject<UsersInteractedEvent>(eventData);

        return obj;
    }

    public override string GetEventType()
    {
        return "user.interacted.event";
    }
}

```

Файл UserChangedEvent.cs

```

using Newtonsoft.Json;
using Shared.Values.EventBus.Common;

namespace Shared.Values.EventBus.UserEvents;

[JsonObject(MemberSerialization.OptOut)]
public class UserChangedEvent : BaseEvent
{
    public int Id { get; set; }

    public int? State { get; set; }

    public string? Gender { get; set; }

    public int? CountryId { get; set; }

    public override UserChangedEvent FromEventData(string eventData)
    {
        var obj = JsonConvert.DeserializeObject<UserChangedEvent>(eventData);

        return obj;
    }
}

```

					КПІ.IT-9127.045440.03.12	Арк.
						50
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

public override string GetEventType()
{
    return "user.changed.event";
}
}

```

Файл UserRegisteredEvent.cs

```

using Newtonsoft.Json;
using Shared.Values.EventBus.Common;

namespace Shared.Values.EventBus.UserEvents;

[JsonObject(MemberSerialization.OptOut)]
public class UserRegisteredEvent : BaseEvent
{
    public int Id { get; set; }

    public string Gender { get; set; }

    public int CountryId { get; set; }

    public string RegDt { get; set; }

    public override UserRegisteredEvent FromEventData(string eventData)
    {
        var obj =
        JsonConvert.DeserializeObject<UserRegisteredEvent>(eventData);

        return obj;
    }

    public override string GetEventType()
    {
        return "user.registered.event";
    }
}

```

Файл index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="Split Divider UI"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/favicon.ico" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Split Divider</title>

    <script type="text/javascript" src="scripts.js"></script>
    <script src="https://unpkg.com/cytoscape/dist/cytoscape.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.0.0.min.js"
      integrity="sha256-JmvOoLtYsmqlsWxa7mDSLMwa6dZ9rrIdtrrVYRnDRH0="
      crossorigin="anonymous"></script>
    <script src="https://github.com/iVis-at-Bilkent/cytoscape.js-
      graphml/blob/master/cytoscape-graphml.js"></script>
  </head>

```

					КПІ.IT-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		51

```

<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
</html>

```

Файл index.js

```

import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import App from "./App";

import "bootstrap/dist/css/bootstrap.min.css";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);

```

Файл App.js

```

import React, { useEffect, useState } from "react";
import "./styles/App.css";
import AuthService from "../services/auth.service";
import Router from "../components/Router";
import HeaderNavbar from "../components/HeaderNavbar";

function App() {
  const [user, setUser] = useState(null);

  useEffect(() => {
    AuthService.getCurrentUser()
      .then(data => {
        setUser(data);
      })
      .catch(err => {});
  }, []);

  return (
    <div className="App">
      <HeaderNavbar user={user} />

      <Router user={user} setUser={setUser} />
    </div>
  );
}

export default App;

```

Файл graph.css

```

#cy {
  height: 800px;
  width: 800px;
}

```

					КПІ.ІТ-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		52

Файл animations.css

```
#graphic {
  position: absolute;
  width: 600px;
  height: 600px;
  margin: auto;
  transform: rotate(45deg);
  top: 0; bottom: 0; left: 50%; right: 0;
}
```

Файл App.css

```
.App {
  text-align: center;
}

.App button {
  border: none;
}

.App button:hover, .link:hover {
  filter: brightness(85%)
}

.input-title {
  color: black;
  font-weight: 500;
}

#light-card-with-input {
  width: 20rem;
  padding: 1rem;
  background: #f7f7f7;
  color: #000000;
}

#light-card-with-input button {
  background: #898989;
  width: 60%;
}

#light-card-with-input #last-div {
  padding-top: 0.4rem;
  padding-bottom: 1rem;
}

.narrow-row {
  padding: .5rem;
}

.like-btn-hover {
  cursor: pointer;
  filter: brightness(95%);
}

.btn-in-row {
  margin: 0.2rem;
}
```

Файл split.service.js

					КПІ.IT-9127.045440.03.12	Арк.
						53
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

import { HttpStatusCode } from "axios";
import { requester } from "../requester";

export const PAGE_SIZE = 20;

const getSplits = async (page, state = null) => {
  let params = {
    pageSize: PAGE_SIZE
  };

  if (state) {
    params = {
      ...params,
      state: state
    };
  }

  return requester
    .get(`/splits`, {
      withCredentials: true,
      params: params
    })
    .then(resp => {
      if (resp.status === HttpStatusCode.NotFound) {
        throw new Error("Splits not found");
      }

      return resp.data;
    })
    .catch(err => {
      if (err.message.includes(HttpStatusCode.Unauthorized)) {
        throw new Error("You have to authorize to access this information");
      }

      if (err.message.includes(HttpStatusCode.Forbidden)) {
        throw new Error("You don't have access to this information");
      }

      if (err.message) throw new Error("Something went wrong: " +
err.message);
    });
};

const getSplit = async id => {
  return requester
    .get(`/splits/${id}`, {
      withCredentials: true
    })
    .then(resp => {
      if (resp.status === HttpStatusCode.NotFound) {
        throw new Error("Split not found");
      }

      return resp.data;
    })
    .catch(err => {
      if (err.message.includes(HttpStatusCode.Unauthorized)) {
        throw new Error("You have to authorize to access this information");
      }

      if (err.message.includes(HttpStatusCode.Forbidden)) {
        throw new Error("You don't have access to this information");
      }
    })
};

```

```

        if (err.message) throw new Error("Something went wrong: " +
err.message);
    });
};

const getSplitGraph = async id => {
    return requester
        .get(`/splits/${id}/graph`, {
            withCredentials: true,
            headers: {
                "Content-Type": "application/octet-stream"
            }
        })
        .then(resp => {
            if (resp.status === HttpStatusCode.NotFound) {
                throw new Error("Split not found");
            }

            return resp.data;
        })
        .catch(err => {
            if (err.message.includes(HttpStatusCode.Unauthorized)) {
                throw new Error("You have to authorize to access this information");
            }

            if (err.message.includes(HttpStatusCode.Forbidden)) {
                throw new Error("You don't have access to this information");
            }

            if (err.message) throw new Error("Something went wrong: " +
err.message);
        });
};

const getSplitUsers = async id => {
    return requester
        .get(`/splits/${id}/users`, {
            withCredentials: true
        })
        .then(resp => {
            if (resp.status === HttpStatusCode.NotFound) {
                throw new Error("Information not found");
            }

            return resp.data;
        })
        .catch(err => {
            if (err.message.includes(HttpStatusCode.Unauthorized)) {
                throw new Error("You have to authorize to access this information");
            }

            if (err.message.includes(HttpStatusCode.Forbidden)) {
                throw new Error("You don't have access to this information");
            }

            if (err.message) throw new Error("Something went wrong: " +
err.message);
        });
};

export const patchStatusName = {
    close: "close",
    activate: "activate",
    suspend: "suspend"
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

};

const patchStatus = async (id, statusName) => {
  return requester
    .patch(`/splits/${id}/${statusName}`, { withCredentials: true })
    .then(resp => {
      if (resp.status !== HttpStatusCodes.Ok) {
        throw new Error(`Could not ${statusName} split`);
      }

      return resp.data;
    })
    .catch(err => {
      if (err.message.includes(HttpStatusCodes.Unauthorized)) {
        throw new Error(`You have to authorize to ${statusName} split`);
      }

      if (err.message.includes(HttpStatusCodes.Forbidden)) {
        throw new Error(`You don't have permissions to ${statusName} split`);
      }

      throw new Error("Something went wrong: " + err.message);
    });
};

const postSplit = async (
  name,
  gender,
  minRegDt,
  countryIds,
  actionsWeights
) => {
  let createParams = {
    name: name,
    gender: gender,
    minRegDt: minRegDt,
    countryIds: countryIds,
    actionsWeights: actionsWeights
  };

  return requester
    .post(`/splits`, createParams, {
      withCredentials: true
    })
    .then(resp => resp.data)
    .catch(err => {
      if (err.message.includes(HttpStatusCodes.Unauthorized)) {
        throw new Error("You have to authorize to create split");
      }

      if (err.message.includes(HttpStatusCodes.Forbidden)) {
        throw new Error("You don't have permissions to create split");
      }

      if (err.message) throw new Error("Something went wrong: " +
err.message);
    });
};

const patchSplit = async (
  id,
  name,
  gender,
  minRegDt,

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІТ-9127.045440.03.12

Арк.

56

```

countryIds,
actionsWeights
) => {
  let params = {
    name: name,
    gender: gender,
    minRegDt: minRegDt,
    countryIds: countryIds,
    actionsWeights: actionsWeights
  };

  return requester
    .patch(`/splits/${id}`, params, {
      withCredentials: true
    })
    .then(resp => resp.data)
    .catch(err => {
      if (err.message.includes(HttpStatusCode.Unauthorized)) {
        throw new Error("You have to authorize to create split");
      }

      if (err.message.includes(HttpStatusCode.Forbidden)) {
        throw new Error("You don't have permissions to create split");
      }

      if (err.message) throw new Error("Something went wrong: " +
err.message);
    });
};

const splitService = {
  getSplits,
  patchStatus,
  postSplit,
  getSplit,
  getSplitUsers,
  getSplitGraph,
  patchSplit
};

export default splitService;

```

Файл split.helper.js

```

const SplitStatusCreated = 0;
const SplitStatusActivated = 1;
const SplitStatusReadyToTest = 2;
const SplitStatusSuspended = 3;
const SplitStatusClosed = 4;

export const StateNames = {
  0: "Created",
  1: "Activated",
  2: "Ready to test",
  3: "Suspended",
  4: "Closed"
};

export const InteractionTypes = {
  dialog: "dialog",
  like: "like",
  hide: "hide",
  "video-call": "video-call"
}

```

					КПІ.ІТ-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		57

```

};

const getStateName = state => {
  return StateNames[state];
};

const canBeEdited = state => {
  return state === SplitStatusCreated || state === SplitStatusSuspended;
};

const canBeActivated = state => {
  return state === SplitStatusCreated || state === SplitStatusSuspended;
};

const canBeSuspended = state => {
  return state === SplitStatusActivated || state === SplitStatusReadyToTest;
};

const splitHasGroups = state => {
  return state === SplitStatusReadyToTest || state === SplitStatusClosed;
};

const canBeClosed = state => {
  return (
    state === SplitStatusCreated ||
    state === SplitStatusReadyToTest ||
    state === SplitStatusSuspended
  );
};

const helper = {
  getStateName,
  canBeEdited,
  canBeActivated,
  canBeSuspended,
  canBeClosed,
  splitHasGroups
};

export default helper;

```

Файл Router.js

```

import React from "react";

import { Route, Routes } from "react-router-dom";
import Login from "../auth/Login";
import Register from "../auth/Register";
import Logout from "../auth/Logout";
import PropTypes from "prop-types";
import HomePage from "../pages/HomePage";
import ErrorPage from "../pages/ErrorPage";
import ParticipantsSearch from "../participants/ParticipantSearch";
import SplitList from "../splits/List/SplitList";
import SplitListPage from "../splits/List/SplitListPage";
import Split from "../splits/Split/Split";
import AdminPage from "../admin/AdminPage";

export default function Router(props) {
  const { user, setUser } = props;

  return (
    <div className="container mt-3">
      <Routes>
        <Route exact path="/documentation" element={<>Documentation</>} />

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.IT-9127.045440.03.12

Арк.

58

```

    {!user && (
      <>
        <Route exact path="/login" element={<Login setUser={setUser} />} />
      />
      <Route
        exact
        path="/register"
        element={<Register setUser={setUser} />}
      />
    </>
  )}
  {user && (
    <>
      <Route
        exact
        path="/logout"
        element={<Logout setUser={setUser} />}
      />
      <Route
        exact
        path="/participants"
        element={<ParticipantsSearch />}
      />
      <Route exact path="/splits" element={<SplitListPage />} />
      <Route path="/splits/:id" element={<Split />} />
      <Route exact path="/admin" element={<AdminPage />} />
    </>
  )}
  <Route path="/" element={<HomePage />} />
  <Route path="*" element={<ErrorPage />} />
</Routes>
</div>
);
}

Router.propTypes = {
  user: PropTypes.object,
  setUser: PropTypes.func.isRequired
};

```

Файл HeaderNavbar.js

```

import React, { useState } from "react";
import {
  Collapse,
  Navbar,
  NavbarToggler,
  NavbarBrand,
  Nav,
  NavItem,
  NavLink,
  NavbarText
} from "reactstrap";

export default function HeaderNavbar(props) {
  const { user } = props;

  const [isOpen, setIsOpen] = useState(false);

  const toggle = () => {
    setIsOpen(!isOpen);
  };

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

return (
  <div>
    <Navbar
      color="dark"
      dark
      expand="md"
      style={{ height: "5rem", fontSize: "120%", paddingLeft: "2rem" }}
    >
      <NavbarBrand href="/" style={{ fontSize: "120%" }}>
        Split Divider
      </NavbarBrand>
      <NavbarToggler onClick={toggle} />
      <Collapse isOpen={isOpen} navbar>
        <Nav className="me-auto" navbar>
          <NavItem>
            <NavLink href="/splits">Splits</NavLink>
          </NavItem>
          <NavItem>
            <NavLink href="/participants">Participants</NavLink>
          </NavItem>
          <NavItem>
            <NavLink href="/documentation">Documentation</NavLink>
          </NavItem>
          <NavItem>
            {!!user ? (
              <NavLink href="/logout">Log out</NavLink>
            ) : (
              <NavLink href="/login">Log in</NavLink>
            )}
          </NavItem>
        </Nav>
        {user && (
          <NavbarText style={{ paddingRight: "2rem" }}>
            {user.email}
          </NavbarText>
        )}
      </Collapse>
    </Navbar>
  </div>
);
}

```

Файл Split.js

```

import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import SplitService from "../../services/split.service";
import {
  Card,
  CardHeader,
  Col,
  Container,
  ListGroup,
  ListGroupItem,
  Modal,
  ModalBody,
  ModalHeader,
  Row,
  Table
} from "reactstrap";
import UpdateSplitModal from "../../Update/UpdateSplitModal";
import SplitHelper from "../../services/split.helper";

```

					КПІ.ІТ-9127.045440.03.12	Арк. 60
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

import ButtonsGroup from "../inputs/ButtonsGroup";
import { GENDERS_MAP } from "../../../../../services/gender.helper";
import { COUNTRIES_MAP } from "../../../../../services/geo.helper";
import { prepareDatetimeForDisplay } from
"../../../../../services/datetime.helper";
import Graph from "../Graph";

export default function Split() {
  const { id } = useParams();

  const [split, setSplit] = useState({
    id: 1,
    actionsWeights: {}
  });

  const [groupsState, setGroupsState] = useState([], []);

  const [msgState, setMsgState] = useState({
    isOpen: false,
    msg: ""
  });

  const closeMsg = () => {
    setMsgState(prev => ({
      ...prev,
      isOpen: false
    }));
  };

  useEffect(() => {
    SplitService.getSplit(id)
      .then(data => {
        setSplit(data);

        if (SplitHelper.splitHasGroups(data.state)) {
          SplitService.getSplitUsers(id).then(data => {
            let groups = [], [];

            data.forEach(e => {
              const group = parseInt(e.group);
              groups[group].push(parseInt(e.userId));
            });

            setGroupsState(groups);
          });
        }
      })
      .catch(err => {
        setMsgState({
          isOpen: true,
          msg: err.message
        });
      });
  }, []);

  const [isEditModalOpen, setIsEditModalOpen] = useState(false);

  const onEditSplit = e => {
    setIsEditModalOpen(true);
  };

  const toggleEditModal = () => {
    setIsEditModalOpen(false);
  };
};

```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

const changeStatus = (id, status) => {
  SplitService.patchStatus(id, status)
    .then(() => {
      window.location.reload();
    })
    .catch(err => {
      setMsgState({
        isOpen: true,
        msg: err.message
      });
    });
};

return (
  <Container style={{ paddingTop: "2rem" }}>
    <Row>
      <Col xs={5}>
        <Card
          style={{
            width: "100%"
          }}
          className={"text-start"}
        >
          <CardHeader tag="h5">{split.name}</CardHeader>
          <ListGroup flush>
            <ListGroupItem>
              <b>ID:</b> {split.id}
            </ListGroupItem>
            <ListGroupItem>
              <b>State:</b> {SplitHelper.getStateName(split.state)}
            </ListGroupItem>
            <ListGroupItem>
              <b>Actions' weights: </b>
              <table className={"table table-light border"}>
                <thead className={"table-dark"}>
                  <tr>
                    <th scope={"col"}>Name</th>
                    <th scope={"col"}>Weight</th>
                  </tr>
                </thead>
                <tbody>
                  {Object.keys(split.actionsWeights).map(k => (
                    <tr key={k}>
                      <th scope={"row"} key={k}>
                        {k}
                      </th>
                      <td key={k}>{split.actionsWeights[k]}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </ListGroupItem>
            <ListGroupItem>
              <b>Gender:</b> {GENDERS_MAP[split.gender]}
            </ListGroupItem>
            <ListGroupItem>
              <b>Countries:</b>{" "}
              {split.countryIds && split.countryIds.length > 0 ? (
                <Table className={"border"}>
                  <tbody>
                    {split.countryIds.map(id => (
                      <tr>
                        <td>{COUNTRIES_MAP[id]}</td>

```

```

        </tr>
    }}}
    </tbody>
</Table>
) : (
    "All"
)}
</ListGroupItem>
<ListGroupItem>
    <b>Min registration time:</b>{" "}
    {split.minRegDt
    ? prepareDatetimeForDisplay(split.minRegDt)
    : "Not specified"}
</ListGroupItem>
{SplitHelper.splitHasGroups(split.state) && (
    <ListGroupItem>
        <b>Groups: </b>{" "}
        <Table bordered>
            <thead>
                <tr>
                    <th>#</th>
                    <th>Total</th>
                    <th style={{ backgroundColor: "#78d65e" }}>
                        Control (0)
                    </th>
                    <th style={{ backgroundColor: "orange" }}>Test
(1)</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <th scope="row">Number of users</th>
                    <td>{groupsState[0].length +
groupsState[1].length}</td>
                    <td>{groupsState[0].length}</td>
                    <td>{groupsState[1].length}</td>
                </tr>
                <tr>
                    <th scope="row">Percentage</th>
                    <td>100%</td>
                    <td>
                        { (
                            (groupsState[0].length /
                                (groupsState[0].length +
groupsState[1].length)) *
                                100
                            ) .toFixed(2) }
                        %
                    </td>
                    <td>
                        { (
                            (groupsState[1].length /
                                (groupsState[0].length +
groupsState[1].length)) *
                                100
                            ) .toFixed(2) }
                        %
                    </td>
                </tr>
            </tbody>
        </Table>
    </ListGroupItem>
)}
</ListGroup>

```

```

</Card>
<Row style={{ paddingTop: "1rem" }} className={"text-start"}>
  <Col>
    <ButtonsGroup
      state={split.state}
      id={split.id}
      changeStatus={changeStatus}
      onEditSplit={onEditSplit}
    />
  </Col>
</Row>
</Col>
<Col>
  {SplitHelper.splitHasGroups(split.state) ? (
    <Graph
      id={split.id}
      groups={groupsState}
      onError={msg => {
        setMsgState({
          isOpen: true,
          msg: msg
        });
      }}
    />
  ) : (
    "Groups are not formed yet"
  )}
</Col>

<Modal isOpen={msgState.show} toggle={closeMsg}>
  <ModalHeader toggle={closeMsg}>Error occurred</ModalHeader>
  <ModalBody>{msgState.msg}</ModalBody>
</Modal>

<UpdateSplitModal
  id={split.id}
  isOpen={isEditModalOpen}
  toggle={toggleEditModal}
/>
</Row>
</Container>
);
}

```

Файл Graph.js

```

import React, { useEffect, useRef, useState } from "react";
import cytoscape from "cytoscape";
import jquery from "jquery";
import graphml from "cytoscape-graphml";

import "../../styles/graph.css";
import { Button } from "reactstrap";
import GraphmlHelper from "../../services/graphml.helper";
import SplitService from "../../services/split.service";

graphml(cytoscape, jquery);

export default function Graph(props) {
  const { id, groups, onError } = props;

  const [isDisbaled, setIsDisbaled] = useState(false);
  const [graphmlStr, setGraphmlStr] = useState("");

```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

const containerRef = useRef();
const [cy, setCy] = useState({});

useEffect(() => {
  SplitService.getSplitGraph(id)
    .then(data => {
      setGraphmlStr(GraphmlHelper.prepareGraphML(data));
    })
    .catch(err => {
      onError(err.message);
    });

  const config = {
    container: containerRef.current,
    style: [
      {
        selector: "node",
        style: {
          label: "data(id)",
          width: "8px",
          height: "8px",
          fontSize: "4px",
          textValign: "center"
        }
      },
      {
        selector: "edge",
        style: {
          width: "data(weight)"
        }
      },
      {
        selector: "node.control",
        style: {
          label: "data(id)",
          width: "8px",
          height: "8px",
          fontSize: "4px",
          textValign: "center",
          backgroundColor: "#78d65e"
        }
      },
      {
        selector: "node.test",
        style: {
          label: "data(id)",
          width: "8px",
          height: "8px",
          fontSize: "4px",
          textValign: "center",
          backgroundColor: "orange"
        }
      }
    ]
  };
  setCy(cytoscape(config));

  return () => {
    setCy({});
  };
}, []);

return (

```

```

<div className={"text-start"}>
  <Button
    disabled={isDisbaled}
    onClick={() => {
      setIsDisabled(true);

      cy.graphml({
        layoutBy: "cose",
        node: {
          data: true,
          position: true
        },
        edge: {
          // css: true,
          data: true
        }
      });
      cy.graphml(graphmlStr);

      groups[0].forEach(id => {
        cy.getElementById("" + id).addClass("control");
      });

      groups[1].forEach(id => {
        cy.getElementById("" + id).addClass("test");
      });

      const eles = cy.elements();

      cy.elements().remove();

      cy.add(eles);
    }}
  >
    Show graph
  </Button>
  <div ref={containerRef} style={{ height: "60vh" }} />
</div>
);
}

```

Файл SplitListPage.js

```

import React, { useState } from "react";
import {
  Button,
  Col,
  Dropdown,
  DropdownItem,
  DropdownMenu,
  DropdownToggle,
  Row
} from "reactstrap";
import SplitList from "../SplitList";
import SplitHelper, { StateNames } from "../../services/split.helper";
import Icon, { iconTypes } from "../../utils/Icon";
import CreateSplitModal from "../Create/CreateSplitModal";

export default function SplitListPage() {
  const [filterState, setFilterState] = useState({
    isOpen: false,
    selected: null,
    isInitial: true
  });

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

});

const [isModalOpen, setIsModalOpen] = useState(false);

const toggleModal = () => {
  setIsModalOpen(!isModalOpen);
};

const onClickSetState = (e, state) => {
  e.preventDefault();

  setFilterState(prev => ({
    ...prev,
    selected: state,
    isInitial: false
  }));
};

const toggleFilter = () => {
  setFilterState(prev => ({
    ...prev,
    isOpen: !prev.isOpen
  }));
};

return (
  <>
    <Row>
      <Col className={"text-start"}>
        <Dropdown isOpen={filterState.isOpen} toggle={toggleFilter}>
          <DropdownToggle caret>
            {filterState.isInitial
              ? "Select state"
              : filterState.selected
              ? SplitHelper.getStateName(filterState.selected)
              : "All"}
          </DropdownToggle>
          <DropdownMenu>
            <DropdownItem header>State</DropdownItem>
            <DropdownItem onClick={e => onClickSetState(e, null)}>
              All
            </DropdownItem>
            {Object.keys(StateNames).map(k => (
              <DropdownItem onClick={e => onClickSetState(e, k)} key={k}>
                {StateNames[k]}
              </DropdownItem>
            ))}
            <DropdownItem></DropdownItem>
          </DropdownMenu>
        </Dropdown>
      </Col>
      <Col className={"text-end"}>
        <Button
          className={"btn border btn-light btn-block"}
          onClick={toggleModal}
        >
          <Icon type={iconTypes.plusSign} />
        </Button>
      </Col>
    </Row>
    <SplitList stateFilter={filterState.selected} />

    <CreateSplitModal isOpen={isModalOpen} toggle={toggleModal} />
  </>
);

```

```
);
}
```

Файл SplitList.js

```
import React, { useEffect, useState } from "react";
import SplitService, { PAGE_SIZE } from "../../services/split.service";
import { PaginationControl } from "react-bootstrap-pagination-control";
import {
  Alert,
  Col,
  Container,
  Modal,
  ModalBody,
  ModalHeader,
  Row
} from "reactstrap";
import SplitItem from "../SplitItem";
import SplitHelper from "../../services/split.helper";
import UpdateSplitModal from "../Update/UpdateSplitModal";

function SplitList(props) {
  const { stateFilter } = props;

  const [listState, setListState] = useState({
    splits: [],
    totalCount: 0,
    page: 1,
    isLoading: false,
    isInitial: true,
    message: null,
    showMessage: false
  });

  const [modal, setModal] = useState({
    show: false,
    message: null
  });

  useEffect(() => {
    SplitService.getSplits(listState.page, stateFilter)
      .then(res => {
        setListState(prev => ({
          ...prev,
          isInitial: false,
          message: null,
          isLoading: false,
          showMessage: false,
          splits: res.items,
          totalCount: res.totalCount
        }));
      })
      .catch(err => {
        setListState(prev => ({
          ...prev,
          isInitial: true,
          message: err.message,
          showMessage: true,
          isLoading: false,
          splits: null
        }));
      });
  }, [listState.page, stateFilter]);
```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

const setPage = page => {
  setListState(prev => ({
    ...prev,
    page: page
  }));
};

const closeAlert = () => {
  setListState(prev => ({
    ...prev,
    showMessage: false
  }));
};

const changeStatus = (id, status) => {
  SplitService.patchStatus(id, status)
    .then(() => {
      window.location.reload();
    })
    .catch(err => {
      setModal({
        show: true,
        message: err.message
      });
    });
};

const closeModal = () => {
  setModal({
    show: false,
    message: null
  });
};

const [editModalState, setEditModalState] = useState({
  isOpen: false,
  id: 1
});

const toggleEditModal = () => {
  setEditModalState(prev => ({
    ...prev,
    isOpen: false
  }));
};

const onUpdate = id => {
  setEditModalState({
    id: id,
    isOpen: true
  });
};

return (
  <>
    {listState.totalCount === 0 && (
      <Alert color="primary" isOpen={true} style={{ marginTop: "2rem" }}>
        No results found
      </Alert>
    )}
    {listState.totalCount > 0 && (
      <>
        <PaginationControl

```

```

    page={listState.page}
    between={4}
    total={listState.totalCount}
    limit={PAGE_SIZE}
    changePage={page => {
      setPage(page);
    }}
    ellipsis={1}
  />

  <div>
    {listState.isLoading && (
      <span className="spinner-border spinner-border-sm"></span>
    )}
    {listState.showMessage && (
      <Alert
        color="danger"
        isOpen={listState.showMessage}
        toggle={closeAlert}
      >
        {listState.message}
      </Alert>
    )}
  </div>

  {listState.splits && (
    <Container className={"mt-auto"}>
      <Row>
        <Col
          className="bg-dark border text-light text-start narrow-row"
          xs={1}
        >
          Split Id
        </Col>
        <Col
          className="bg-dark border text-light text-start narrow-row"
          xs={5}
        >
          Name
        </Col>
        <Col
          className="bg-dark border text-light text-start narrow-row"
          xs={2}
        >
          State
        </Col>
        <Col className="text-start narrow-row"></Col>
      </Row>
      {listState.splits.map(s => {
        return (
          <SplitItem
            item={s}
            onUpdate={e => {
              onUpdate(s.id);
            }}
            key={` ${s.id} `}
            changeStatus={changeStatus}
          />
        );
      })}
    </Container>
  )}

  {!listState.splits && !listState.isInitial && (

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІТ-9127.045440.03.12

Арк.

70

```

        <div
          style={{
            paddingTop: "1rem"
          }}
        >
          <Alert color="primary" isOpen={true}>
            No results found
          </Alert>
        </div>
      )}

      <UpdateSplitModal
        id={editModalState.id}
        isOpen={editModalState.isOpen}
        toggle={toggleEditModal}
      />

      <Modal isOpen={modal.show} toggle={closeModal}>
        <ModalHeader toggle={closeModal}>Error occurred</ModalHeader>
        <ModalBody>{modal.message}</ModalBody>
      </Modal>
    </>
  )}
</>
);
}

export default SplitList;

```

Файл SplitItem.js

```

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";
import { Col, Row } from "reactstrap";
import SplitHelper from "../../services/split.helper";
import ButtonsGroup from "../inputs/ButtonsGroup";

function SplitItem(props) {
  const { item, changeStatus, onUpdate } = props;

  const navigate = useNavigate();

  const navigateToInfo = () => {
    navigate(`/splits/${item.id}`);
  };

  const onEditSplit = e => {
    onUpdate(e);
  };

  const originalStyle = "bg-light text-secondary border text-start";
  const [dataClassName, setDataClassName] = useState(originalStyle);

  const addStyle = () => {
    setDataClassName(prev => prev + " like-btn-hover");
  };

  const removeStyle = () => {
    setDataClassName(originalStyle);
  };

  return (

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

<>
<Row>
  <Col
    className={dataClassName}
    xs={1}
    onClick={navigateToInfo}
    onMouseEnter={addStyle}
    onMouseLeave={removeStyle}
  >
    {item.id}
  </Col>
  <Col
    className={dataClassName}
    xs={5}
    onClick={navigateToInfo}
    onMouseEnter={addStyle}
    onMouseLeave={removeStyle}
  >
    {item.name}
  </Col>
  <Col
    className={dataClassName}
    xs={2}
    onClick={navigateToInfo}
    onMouseEnter={addStyle}
    onMouseLeave={removeStyle}
  >
    { SplitHelper.getStateName(item.state) }
  </Col>
  <Col className="text-secondary text-start">
    <ButtonsGroup
      state={item.state}
      id={item.id}
      changeStatus={changeStatus}
      onEditSplit={onEditSplit}
    />
  </Col>
</Row>
</>
);
}

export default SplitItem;

```

Файл NameInput.js

```

import React from "react";
import { Input } from "reactstrap";
import PropTypes from "prop-types";

function NameInput(props) {
  const { state, setState, validations } = props;

  const onChangeText = e => {
    for (let validate of validations) {
      const feedback = validate(e.target.value);

      if (feedback) {
        setState(prev => ({
          ...prev,
          value: e.target.value,
          isValid: false,
          feedback: feedback

```

```

    ));

    return;
  }

  setState(prev => ({
    ...prev,
    value: e.target.value,
    isValid: true,
    feedback: null
  }));
}
};

return (
  <div className="form-group">
    <label htmlFor="text" className="input-title">
      Name
    </label>
    <Input
      type="text"
      className="form-control"
      name="text"
      value={state.value}
      onChange={onChangeText}
      invalid={!state.isValid}
    />
    {!state.isValid && state.feedback}
  </div>
);
}

NameInput.propTypes = {
  state: PropTypes.object.isRequired,
  setState: PropTypes.func.isRequired,
  validations: PropTypes.arrayOf(PropTypes.func)
};

export default NameInput;

```

Файл MinRegDtInput.js

```

import React from "react";
import { Input, Label } from "reactstrap";

import "react-datetime-picker/dist/DateTimePicker.css";

export default function MinRegDtInput(props) {
  const { state, setState } = props;

  const onDtInput = e => {
    setState(e.target.value);
  };

  return (
    <>
      <div>
        <Label
          htmlFor="dt_picker"
          className="input-title"
          style={{ paddingBottom: ".5rem", paddingTop: ".5rem" }}
        >
          Minimal Registration Time
        </Label>
      </div>
    </>
  );
}

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.IT-9127.045440.03.12

Арк.

73

```

        </Label>
      </div>
      <div id={"dt_picker"}>
        <div>
          <Input type={"datetime-local"} value={state} onInput={onDtInput} />
        </div>
      </div>
    </>
  );
}

```

Файл GenderInput.js

```

import React from "react";
import { Button, ButtonGroup, Label } from "reactstrap";
import { GENDERS_MAP } from "../../services/gender.helper";

export default function GenderInput(props) {
  const { state, setState } = props;

  return (
    <>
      <div>
        <Label
          htmlFor="gender_selector"
          className="input-title"
          style={{ paddingBottom: ".5rem", paddingTop: ".5rem" }}
        >
          Gender
        </Label>
      </div>
      <ButtonGroup id={"gender_selector"}>
        {Object.keys(GENDERS_MAP).map(k => (
          <Button
            color="secondary"
            outline
            key={k}
            onClick={() => setState(k)}
            active={state === k}
          >
            {GENDERS_MAP[k]}
          </Button>
        ))}
      </ButtonGroup>
    </>
  );
}

```

Файл CountryIdsInput.js

```

import React from "react";
import { Button, ButtonGroup, Label } from "reactstrap";
import { COUNTRIES_MAP } from "../../services/geo.helper";

export default function CountryIdsInput(props) {
  const { state, setState } = props;

  const onCheckboxBtnClick = selected => {
    const index = state.indexOf(selected);

    if (index < 0) {

```

					КПІ.ІТ-9127.045440.03.12	Арк.
						74
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        state.push(selected);
    } else {
        state.splice(index, 1);
    }

    setState([...state]);
};

return (
    <>
        <div>
            <Label
                htmlFor="country_selector"
                className="input-title"
                style={{ paddingBottom: ".5rem" }}
            >
                Countries
            </Label>
        </div>
        <ButtonGroup id="country_selector">
            {Object.keys(COUNTRIES_MAP).map(k => (
                <Button
                    color="secondary"
                    outline
                    key={k}
                    onClick={() => onCheckboxBtnClick(k)}
                    active={state.includes(k)}
                >
                    {COUNTRIES_MAP[k]}
                </Button>
            ))}
        </ButtonGroup>
    </>
);
}

```

Файл ButtonsGroup.js

```

import React from "react";
import SplitHelper from "../../services/split.helper";
import { Button } from "reactstrap";
import { patchStatusName } from "../../services/split.service";
import Icon, { iconTypes } from "../../utils/Icon";

export default function ButtonsGroup(props) {
    const { state, onEditSplit, changeStatus, id } = props;

    return (
        <>
            <Button
                className={"bg-light border btn-in-row"}
                onClick={onEditSplit}
                disabled={!SplitHelper.canBeEdited(state)}
            >
                <Icon type={iconTypes.editPencil} />
            </Button>
            {SplitHelper.canBeActivated(state) && (
                <Button
                    color="success"
                    className={"btn-in-row"}
                    onClick={() => {
                        changeStatus(id, patchStatusName.activate);
                    }}
                >

```

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.IT-9127.045440.03.12

Арк.

75

```

        window.location.reload();
    }}
    >
    Activate
  </Button>
)}
{ SplitHelper.canBeSuspended(state) && (
  <Button
    color={"primary"}
    className={"btn-in-row"}
    onClick={() => {
      changeStatus(id, patchStatusName.suspend);

      window.location.reload();
    }}
  >
    Suspend
  </Button>
)}
{ SplitHelper.canBeClosed(state) && (
  <Button
    color={"warning"}
    className={"btn-in-row"}
    onClick={() => {
      changeStatus(id, patchStatusName.close);

      window.location.reload();
    }}
  >
    Close
  </Button>
)}
</>
);
}

```

Файл ActionsWeightsInput.js

```

import React from "react";

export default function ActionsWeightsTableInput(props) {
  const { state, onChangeInput } = props;

  return (
    <div style={{ paddingTop: "1.5rem" }}>
      <label
        htmlFor="input_table"
        className="input-title"
        style={{ paddingBottom: ".5rem" }}
      >
        Interactions' weights
      </label>
      <table id={"input_table"} className={"table table-light border"}>
        <thead className={"table-dark"}>
          <tr>
            <th scope={"col"}>Name</th>
            <th scope={"col"}>Weight</th>
          </tr>
        </thead>
        <tbody>
          {state.map(v => (
            <tr key={v.i}>
              <th scope={"row"}>{v.name}</th>

```

```

        <td>
          <input
            name="value"
            value={v.value}
            type="number"
            key={v.id}
            id={v.id}
            min={1}
            max={30}
            onChange={onChangeInput}
            placeholder="Enter interaction's weight"
          />
        </td>
      </tr>
    </tbody>
  </table>
</div>
);
}

```

Файл CreateSplitModal.js

```

import React, { useState } from "react";
import { Button, Modal, ModalBody, ModalFooter, ModalHeader } from
"reactstrap";
import SplitService from "../../services/split.service";
import { validText } from "../../utils/ValidationFeedback";
import NameInput from "../inputs/NameInput";
import ActionsWeightsTableInput from "../inputs/ActionsWeightsInput";
import { InteractionTypes } from "../../services/split.helper";
import CountryIdsInput from "../inputs/CountryIdsInput";
import { ALL_GENDERS } from "../../services/gender.helper";
import GenderInput from "../inputs/GenderInput";
import MinRegDtInput from "../inputs/MinRegDtInput";

export default function CreateSplitModal(props) {
  const [msgState, setMsgState] = useState({
    show: false,
    message: null
  });

  const { isOpen, toggle } = props;

  const [nameState, setNameState] = useState({
    value: "",
    isValid: true,
    feedback: null
  });

  let initialActionsWeightsData = [];
  let i = 0;

  Object.keys(InteractionTypes).forEach(k => {
    initialActionsWeightsData.push({
      id: i++,
      name: InteractionTypes[k],
      value: 1
    });
  });

  const [actionsWeightsState, setActionsWeightsState] = useState(
    initialActionsWeightsData
  );

```

```

);

const onChangeInput = e => {
  e.preventDefault();

  let { value, id } = e.target;

  value = parseInt(value);

  if (value < 1) value = 1;
  if (value > 30) value = 30;

  id = parseInt(id);

  setActionsWeightsState(prev => {
    return prev.map(o => {
      if (o.id === id) {
        return {
          id: o.id,
          name: o.name,
          value: value
        };
      } else {
        return o;
      }
    });
  });
};

const [countryIdsState, setCountryIdsState] = useState([]);

const [genderState, setGenderState] = useState(ALL_GENDERS);

const [minRegDtState, setMinRegDtState] = useState(new Date());

const createSplit = e => {
  e.preventDefault();

  if (!nameState.value || !nameState.isValid) {
    setMsgState({
      show: true,
      message: "Name value is not valid"
    });

    return;
  }

  let actionsWeight = {};
  actionsWeightsState.forEach(v => {
    actionsWeight[v.name] = v.value;
  });

  SplitService.postSplit(
    nameState.value,
    genderState === ALL_GENDERS ? null : genderState,
    minRegDtState,
    countryIdsState.map(id => parseInt(id)),
    actionsWeight
  )
  .then(_ => {
    window.location.reload();
  })
  .catch(err => {
    setMsgState({

```

```

        show: true,
        message: err.message
    });
});
};

const closeMsg = () => {
    setMsgState(_ => ({
        show: false,
        message: ""
    }));
};

return (
    <>
        <Modal isOpen={isOpen} toggle={toggle} fullscreen>
            <ModalHeader>Create new split</ModalHeader>
            <ModalBody style={{ width: "30%" }} className={"mx-auto"}>
                <NameInput
                    state={nameState}
                    setState={setNameState}
                    validations={[validText]}
                />
                <ActionsWeightsTableInput
                    state={actionsWeightsState}
                    setState={setActionsWeightsState}
                    onChangeInput={onChangeInput}
                />
                <CountryIdsInput
                    state={countryIdsState}
                    setState={setCountryIdsState}
                />
                <GenderInput state={genderState} setState={setGenderState} />
                <MinRegDtInput state={minRegDtState} setState={setMinRegDtState} />
            </ModalBody>
            <ModalFooter>
                <Button color="success" onClick={createSplit}>
                    Submit
                </Button>{" "}
                <Button color="secondary" onClick={toggle}>
                    Cancel
                </Button>
            </ModalFooter>
        </Modal>

        <Modal isOpen={msgState.show} toggle={closeMsg}>
            <ModalHeader toggle={closeMsg} className={"text-bg-danger"}>
                Error
            </ModalHeader>
            <ModalBody>{msgState.message}</ModalBody>
        </Modal>
    </>
);
}

```

Файл ParticipantsSearch.js

```

import React from "react";

import ParticipantService from "../../services/participant.service";

import { useState } from "react";
import IdSearchBar from "../../utils/inputs/IdSearchBar";

```

					КПІ.ІТ-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		79

```

import { required, valuePositive } from "../utils/ValidationFeedback";
import { Alert, Col, Container, Row } from "reactstrap";

export default function ParticipantsSearch(props) {
  const [state, setState] = useState({
    isLoading: false,
    isInitial: true,
    results: null,
    message: null,
    showMessage: false
  });

  const doSearch = id => {
    setState(prev => ({
      ...prev,
      isInitial: true,
      isLoading: true
    }));

    ParticipantService.getParticipantsSplitGroups(id)
      .then(res => {
        setState(prev => ({
          isInitial: false,
          message: null,
          showMessage: false,
          isLoading: false,
          results: res
        }));
      })
      .catch(err => {
        setState(prev => ({
          isInitial: true,
          message: err.message,
          showMessage: true,
          isLoading: false,
          results: null
        }));
      });
  };

  const closeAlert = () => {
    setState(prev => ({
      ...prev,
      showMessage: false
    }));
  };

  const onChangeInput = () => {
    setState(prev => ({
      ...prev,
      isInitial: true
    }));
  };

  const mapGroupToName = group => {
    return group === 0 ? "Control" : "Test";
  };

  return (
    <>
      <div>
        <IdSearchBar
          validations={[required, valuePositive]}
          onSubmit={doSearch}

```

```

        onChange={onChangeInput}
      />
    </div>
    <div>
      {state.isLoading && (
        <span className="spinner-border spinner-border-sm"></span>
      )}
      {state.showMessage && (
        <Alert color="danger" isOpen={state.showMessage}
toggle={closeAlert}>
          {state.message}
        </Alert>
      )}
    </div>

    <Container
      style={{
        width: "75%"
      }}
    >
      <Row>
        <Col className="bg-dark border text-light text-start">Split
Id</Col>
        <Col className="bg-dark border text-light text-start">Group</Col>
      </Row>
      {state.results !== null && state.results.length > 0
        ? state.results.map(p => (
          <Row>
            <Col className="bg-light border text-start">{p.splitId}</Col>
            <Col className="bg-light border align-content-end text-
start">
              {mapGroupToName(p.group)} ({p.group})
            </Col>
          </Row>
        ))
        : !state.isInitial && (
          <div
            style={{
              paddingTop: "1rem"
            }}
          >
            <Alert color="primary" isOpen={true}>
              No results found
            </Alert>
          </div>
        )}
    </Container>
  </>
);
}

```

Файл HomePage.js

```

import React from "react";
import { Container } from "reactstrap";

import "../styles/animation.css";
import { Link } from "react-router-dom";

export default class HomePage extends React.Component {
  componentDidMount() {
    window.animate();
  }
}

```

					КПІ.ІТ-9127.045440.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		81

```

render() {
  return (
    <div>
      <div
        style={{
          display: "flex",
          flexDirection: "row",
          flexWrap: "wrap",
          justifyContent: "flex-start"
        }}
      >
        <div
          style={{
            paddingTop: "20rem",
            paddingRight: "40rem"
          }}
        >
          <Container>
            <h1 className="display-3">Split Divider</h1>
            <p className="lead">
              Internal tool for handling A/B tests. For more information
              <Link to="/documentation">Documentation</Link>
            </p>
            <p className="lead">
              Log in to have full access to all pages. Or register a new
              account and contact your manager to receive the appropriate
              role
              in the system
            </p>
          </Container>
        </div>
        <div>
          <canvas id="graphic" width="600" height="600"></canvas>
        </div>
      </div>
    </div>
  );
}

```

Файл ErrorPage.js

```

import React from "react";
import { Container } from "reactstrap";

import "../styles/animation.css";
import { Link } from "react-router-dom";
import { Image } from "react-bootstrap";

export default class ErrorPage extends React.Component {
  render() {
    return (
      <div
        style={{
          paddingTop: "20rem"
        }}
      >
        <div
          style={{
            display: "flex",
            flexDirection: "row",

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

        justifyContent: "flex-start"
      }}
    >
      <div>
        <Container>
          <h1 className="display-3">Oops, something went wrong</h1>
          <p className="lead">
            It seems that you don't have access to this page ot such page
            does not exist. Please, <Link to="/login">Log in</Link> to
have
            full access
          </p>
        </Container>
      </div>
      <div>
        <Image
          height={"50%"}
          src={require("../../images/charts-error.png")}
        ></Image>
      </div>
    </div>
  </div>
);
}
}

```

Файл Register.js

```

import React, { useState } from "react";
import { Form } from "reactstrap";

import AuthService from "../../services/auth.service.js";

import {
  required,
  validEmail,
  validPassword,
  valueEquals
} from "../utils/ValidationFeedback.js";
import PasswordInput from "../utils/inputs/PasswordInput.js";
import EmailInput from "../utils/inputs/EmailInput.js";
import PropTypes from "prop-types";
import { useNavigate } from "react-router-dom";
import TokenService from "../../services/token.service";

function Register(props) {
  const { setUser } = props;

  const navigate = useNavigate();

  const [formState, setFormState] = useState({
    success: false,
    message: ""
  });

  const [passwordState, setPasswordState] = useState({
    password: "",
    isValid: true,
    feedback: null
  });

  const [confirmPasswordState, setConfirmPasswordState] = useState({
    password: "",

```

```

        isValid: true,
        feedback: null
    });

    const [emailState, setEmailState] = useState({
        email: "",
        isValid: true,
        feedback: null
    });

    const handleRegister = e => {
        e.preventDefault();

        setFormState(prev => ({
            ...prev,
            success: false,
            message: ""
        }));

        if (
            !emailState.email ||
            !passwordState.password ||
            !confirmPasswordState.password
        ) {
            setFormState(prev => ({
                ...prev,
                success: false,
                message: "Fields cannot be empty"
            }));
            return;
        }

        if (
            passwordState.isValid &&
            confirmPasswordState.isValid &&
            emailState.isValid
        ) {
            AuthService.register(
                emailState.email,
                passwordState.password,
                confirmPasswordState.password
            ).then(
                data => {
                    setFormState(prev => ({
                        ...prev,
                        message: "Your account has been registered",
                        success: true
                    }));

                    setUser(data["user"]);
                    TokenService.saveToken(data["token"]);

                    navigate("/");
                },
                err => {
                    setFormState(prev => ({
                        ...prev,
                        message: err.message
                    }));
                }
            );
        }
    };
};

```

```

return (
  <div className="col-md-12" style={{ paddingTop: "9rem" }}>
    <div className="card card-container mx-auto" id="light-card-with-
input">
      <Form onSubmit={handleRegister}>
        <EmailInput
          state={emailState}
          setState={setEmailState}
          validations={[required, validEmail]}
        />

        <div
          style={{
            paddingTop: "0.4rem"
          }}
        >
          <PasswordInput
            state={passwordState}
            setState={setPasswordState}
            validations={[required, validPassword]}
          />
        </div>

        <div id="last-div">
          <PasswordInput
            title="Confirm password"
            state={confirmPasswordState}
            setState={setConfirmPasswordState}
            validations={[
              required,
              validPassword,
              valueEquals(passwordState.password)
            ]}
          />
        </div>

        <div className="form-group">
          <button type="submit" className="btn btn-primary btn-block">
            <span>Register</span>
          </button>
        </div>

        {formState.message && (
          <div className="form-group" style={{ paddingTop: "0.5rem" }}>
            <div
              className={
                formState.success
                  ? "alert alert-success"
                  : "alert alert-danger"
              }
              role="alert"
            >
              {formState.message}
            </div>
          </div>
        )}
      </Form>
    </div>
  </div>
);
}

Register.propTypes = {
  setUser: PropTypes.func.isRequired

```

```
};

export default Register;
```

Файл Logout.js

```
import React from "react";
import { Navigate } from "react-router-dom";
import authService from "../../services/auth.service";
import PropTypes from "prop-types";
import TokenService from "../../services/token.service";

function Logout(props) {
  const { setUser } = props;

  setUser(null);
  TokenService.clearToken();
  authService
    .logout()
    .then(_ => {
      window.location.reload();
    })
    .catch(err => {
      console.log(err);
    });

  return <Navigate to="/" />;
}

Logout.propTypes = {
  setUser: PropTypes.func.isRequired
};

export default Logout;
```

Файл Login.js

```
import React, { useState } from "react";
import { Form } from "reactstrap";

import {
  required,
  validEmail,
  validPassword
} from "../utils/ValidationFeedback.js";
import AuthService from "../../services/auth.service.js";

import EmailInput from "../utils/inputs/EmailInput.js";
import PasswordInput from "../utils/inputs/PasswordInput.js";
import { Link, useNavigate } from "react-router-dom";
import PropTypes from "prop-types";
import TokenService from "../../services/token.service";

function Login(props) {
  const { setUser } = props;

  const navigate = useNavigate();

  const [formState, setFormState] = useState({
    loading: false,
    message: ""
  });
```

Змін.	Арк.	№ докум.	Підп.	Дата.

```

});

const [emailState, setEmailState] = useState({
  email: "",
  isValid: true,
  feedback: null
});

const [passwordState, setPasswordState] = useState({
  password: "",
  isValid: true,
  feedback: null
});

const handleLogin = e => {
  e.preventDefault();

  setFormState(prev => ({ ...prev, message: "", loading: true }));

  if (!emailState.email || !passwordState.password) {
    setFormState(prev => ({
      ...prev,
      loading: false,
      message: "Fields cannot be empty"
    }));
    return;
  }

  if (passwordState.isValid && emailState.isValid) {
    AuthService.login(emailState.email, passwordState.password).then(
      data => {
        TokenService.saveToken(data["token"]);
        setUser(data["user"]);

        setFormState(prev => ({
          ...prev,
          loading: false
        }));

        navigate("/");
      },
      err => {
        let resMsg =
          (err.response && err.response.data && err.response.data.message)
||
          err.message ||
          err.toString();

        if (resMsg.search("401") !== -1) {
          resMsg = "Incorrect email or password";
        }

        setFormState({
          loading: false,
          message: resMsg
        });
      }
    );
  } else {
    setFormState(prev => ({ ...prev, loading: false }));
  }
};

return (

```

```

<div className="col-md-12" style={{ paddingTop: "9rem" }}>
  <div className="card card-container mx-auto" id="light-card-with-
input">
    <Form onSubmit={handleLogin}>
      <EmailInput
        state={emailState}
        setState={setEmailState}
        validations={[required, validEmail]}
      />

      <div id="last-div">
        <PasswordInput
          state={passwordState}
          setState={setPasswordState}
          validations={[required, validPassword]}
        />
      </div>

      <div className="form-group">
        <button
          type="submit"
          className="btn btn-primary btn-block"
          disabled={formState.loading}
        >
          {formState.loading && (
            <span className="spinner-border spinner-border-sm"></span>
          )}
          <span>Login</span>
        </button>
      </div>
      <div
        style={{
          paddingTop: "0.5rem"
        }}
      >
        <Link to="/register">Don't have an account?</Link>
      </div>

      {formState.message && (
        <div className="form-group" style={{ padding: "0.5rem" }}>
          <div className="alert alert-danger" role="alert">
            {formState.message}
          </div>
        </div>
      )}
    </Form>
  </div>
</div>
);
}

Login.propTypes = {
  setUser: PropTypes.func.isRequired
};

export default Login;

```

Файл UserAndRoles.js

```

import React, { useEffect, useState } from "react";
import {
  Button,
  Col,

```

					КПІ.IT-9127.045440.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		88

```

Container,
Dropdown,
DropdownItem,
DropdownMenu,
DropdownToggle,
Row,
Table
} from "reactstrap";
import Icon, { iconTypes } from "../utils/Icon";
import adminService from "../../services/admin.service";
import AdminService from "../../services/admin.service";

export default function UserAndRoles(props) {
  const { user, userRoles, setUserRoles, onError } = props;

  const [addableRoles, setAddableRoles] = useState([]);
  const [isOpenDropdown, setIsOpenDropdown] = useState(false);

  useEffect(() => {
    AdminService.fetchRoles()
      .then(resp => {
        let addableRolesR = resp.filter(
          r => !userRoles.map(r => r.id).includes(r.id)
        );
        setAddableRoles(addableRolesR);
      })
      .catch(err => {
        onError(err);
      });
  }, [userRoles]);

  const removeRole = role => {
    adminService
      .deleteRoleForUser(user.id, role.name)
      .then(res => {
        let updatedRoles = [];
        userRoles.forEach(role => {
          if (role.id !== role.id) {
            updatedRoles.push(role);
          } else {
            addableRoles.push(role);
          }
        });
        setUserRoles(updatedRoles);
        setAddableRoles(addableRoles);
      })
      .catch(err => {
        onError(err.msg);
      });
  };

  const addRole = role => {
    adminService
      .addRoleForUser(user.id, role.name)
      .then(res => {
        userRoles.push(role);
        setUserRoles(userRoles);

        setAddableRoles(addableRoles.filter(r => r.id !== role.id));
      })
      .catch(err => {
        onError(err.msg);
      });
  };

```

```

};

return (
  <Container className={"text-start"} style={{ paddingLeft: "20%" }}>
    <Row style={{ padding: "1rem" }}>
      <Col tag={"h5"}>{user.email}</Col>
    </Row>
    <Row>
      <Col>
        <Table className={"border"} style={{ width: "65%" }}>
          <tbody>
            {userRoles.map(role => (
              <tr>
                <td>
                  <Row>
                    <Col>{role.name}</Col>
                    <Col className={"text-end"}>
                      <Button
                        className={"btn border btn-light btn-block"}
                        onClick={() => {
                          removeRole(role);
                        }}
                      >
                        <Icon type={iconTypes.closeSign} />
                      </Button>
                    </Col>
                  </Row>
                </td>
              </tr>
            ))}
            {addableRoles.length > 0 && (
              <tr>
                <td>
                  <div style={{ overflow: "hidden" }}>
                    <Dropdown
                      isOpen={isOpenDropdown}
                      toggle={() => {
                        setIsOpenDropdown(!isOpenDropdown);
                      }}
                    >
                      <DropdownToggle caret>Add role</DropdownToggle>
                      <DropdownMenu container="body">
                        {addableRoles.map(r => (
                          <DropdownItem onClick={() => addRole(r)}>
                            {r.name}
                          </DropdownItem>
                        ))}
                      </DropdownMenu>
                    </Dropdown>
                  </div>
                </td>
              </tr>
            ))}
          </tbody>
        </Table>
      </Col>
    </Row>
  </Container>
);
}

```

Файл AdminPage.js

					КПІ.ІТ-9127.045440.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		90

```

import React, { useState } from "react";
import EmailSearchBar from "../utils/inputs/EmailSearchBar";
import { required, validEmail } from "../utils/ValidationFeedback";
import { Alert } from "reactstrap";
import AdminService from "../../services/admin.service";
import UserAndRoles from "../UserAndRoles";
import trim from "validator/es/lib/trim";

export default function AdminPage() {
  const [state, setState] = useState({
    isLoading: false,
    isInitial: true,
    results: {
      user: {},
      roles: []
    },
    message: null,
    showMessage: false
  });

  const doSearch = email => {
    email = trim(email);
    setState(prev => ({
      ...prev,
      isInitial: true,
      isLoading: true
    }));

    AdminService.getUserByEmail(email)
      .then(res => {
        setState(prev => ({
          isInitial: false,
          message: null,
          showMessage: false,
          isLoading: false,
          results: res
        }));
      })
      .catch(err => {
        setState(prev => ({
          isInitial: true,
          message: err.message,
          showMessage: true,
          isLoading: false,
          results: {
            user: {},
            roles: []
          }
        }));
      });
  };

  const closeAlert = () => {
    setState(prev => ({
      ...prev,
      showMessage: false
    }));
  };

  const onChangeInput = () => {
    setState(prev => ({
      ...prev,
      isInitial: true
    }));
  };

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

};

const setUserRoles = roles => {
  setState(prev => ({
    ...prev,
    results: {
      user: prev.results.user,
      roles: roles
    }
  }));
};

const onError = msg => {
  setState(prev => ({
    ...prev,
    showMessage: true,
    msg: msg
  }));
};

return (
  <>
    <div>
      <EmailSearchBar
        validations={[required, validEmail]}
        onSubmit={doSearch}
        onChange={onChangeInput}
      />
    </div>

    <div>
      {state.isLoading && (
        <span className="spinner-border spinner-border-sm"></span>
      )}
      {state.showMessage && (
        <Alert color="danger" isOpen={state.showMessage}
toggle={closeAlert}>
          {state.message}
        </Alert>
      )}
    </div>
    {state.results && state.results.user && state.results.user.email && (
      <UserAndRoles
        user={state.results.user}
        userRoles={state.results.roles}
        setUserRoles={setUserRoles}
        onError={onError}
      />
    )}
  </>
);
}

```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

Методика та програмне забезпечення формування груп учасників А/В тестів
з мінімальним взаємним впливом

Програма та методика тестування

КПІ.IT-9127.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юлія КРАМАР

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олена ХІЛЬЧУК

Київ – 2023

ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ	3
2	МЕТА ТЕСТУВАННЯ.....	4
3	МЕТОДИ ТЕСТУВАННЯ	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

					КПІ.ІТ-9127.045440.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		2

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є програмне забезпечення для формування груп учасників А/В тестів із мінімальним взаємним впливом між групами.

					КПІ.ІТ-9127.045440.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		3

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- знаходження вразливих місць та некоректного виконання програми з метою налагодження;
- перевірка коректності збережених даних;
- перевірка зручності та інформативності графічного інтерфейсу.

					КПІ.ІТ-9127.045440.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		4

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

					КПІ.ІТ-9127.045440.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування правильності виведення візуалізації даних;
- тестування коректності роботи алгоритмів.

					КПІ.ІТ-9127.045440.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		6

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

Методика та програмне забезпечення формування груп учасників А/В тестів
з мінімальним взаємним впливом

Керівництво користувача

КПІ.IT-9127.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юлія КРАМАР

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олена ХІЛЬЧУК

Київ – 2023

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ.....	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи	4
3	ВИКОНАННЯ ПРОГРАМИ.....	5

					КПІ.ІТ-9127.045440.05.34	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Split Divider» - це веб-застосунок, який є допоміжним інструментом для проведення А/В тестів. Застосунок надає можливість створювати А/В тести, задавати їм певну сегментацію та формувати групи учасників цих тестів так, щоб взаємний вплив між учасниками з різних груп був мінімальним.

Застосунок, відповідно до свого призначення, є корпоративним засобом, тому для отримання доступу до нього користувач повинен звернутися до свого безпосереднього керівника для отримання детальних інструкцій.

					КПІ.ІТ-9127.045440.05.34	Арк.
						3
Змін.	Арк.	№ докум.	Підп.	Дата.		

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність пристрою зі встановленим браузером, у якому увімкнено JavaScript;
- наявність доступу до Інтернету;
- для встановлення додатку на пристрої повинно бути не менше 600 МБ вільної пам'яті.

2.2 Завантаження застосунку

«Split Divider» - це веб-застосунок, який не розрахований на завантаження.

2.3 Перевірка коректної роботи

Після відкриття веб-застосунку у браузері користувача повинно автоматично переадресувати на головну сторінку. Для доступу до інформації та повного функціоналу користувачу потрібно буде увійти в систему, якщо акаунт уже існує, або зареєструватися та сконтактуватися із адміністратором сайту або своїм безпосереднім керівником для отримання відповідних прав.

					КПІ.ІТ-9127.045440.05.34	Арк.
						4
Змін.	Арк.	№ докум.	Підп.	Дата.		

3 ВИКОНАННЯ ПРОГРАМИ

У даній частині надається перелік операцій, які можна виконати у застосунку.

3.1 Реєстрація

Користувач має можливість зареєструватися у системі. Для цього потрібно натиснути на кнопку «Log in» у хедері веб-сайту. Після чого користувач потрапляє на сторінку входу. Там необхідно перейти за посиланням для реєстрації.

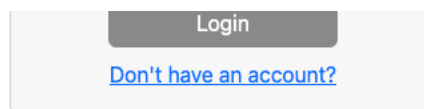


Рисунок 3.1 – Посилання для переходу на сторінку реєстрації

Після натискання користувач потрапляє на сторінку реєстрації. Щоб зареєструватися потрібно ввести коректну електронну пошту, пароль та підтвердження паролю. Після чого натисну кнопку «Register».

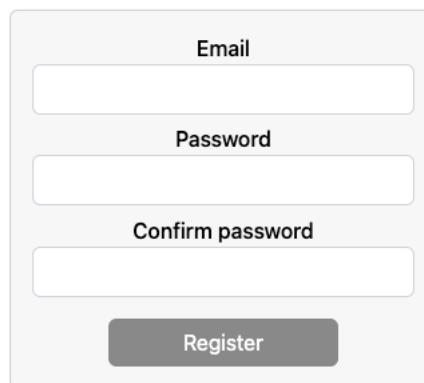
A registration form with a light grey background. It contains three input fields: 'Email', 'Password', and 'Confirm password'. Each field has a label above it. At the bottom of the form is a dark grey button with the text 'Register' in white.

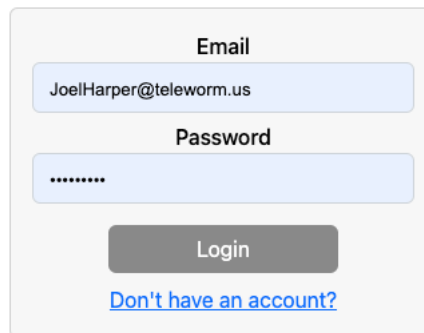
Рисунок 3.2 – Форма реєстрації

3.2 Вхід

Якщо користувач уже зареєструвався у системі, він має можливість увійти у свій акаунт. Для цього потрібно натиснути на кнопку «Log in» у хедері веб-сайту. Після чого користувач потрапляє на сторінку входу.

					КПІ.ІТ-9127.045440.05.34	Арк.
						5
Змін.	Арк.	№ докум.	Підп.	Дата.		

На сторінці потрібно ввести електронну пошту та пароль, з якими було створено обліковий запис. Після чого натиснути кнопку «Log in».



The image shows a login form with two input fields: 'Email' containing 'JoelHarper@teleworm.us' and 'Password' containing seven dots. Below the fields is a 'Login' button and a link that says 'Don't have an account?'.

Рисунок 3.3 – Заповнена форма входу

3.3 Вихід

Авторизований користувач може вийти із системи, натиснувши на кнопку «Log out». Після цього користувача перенаправлено на головний екран.

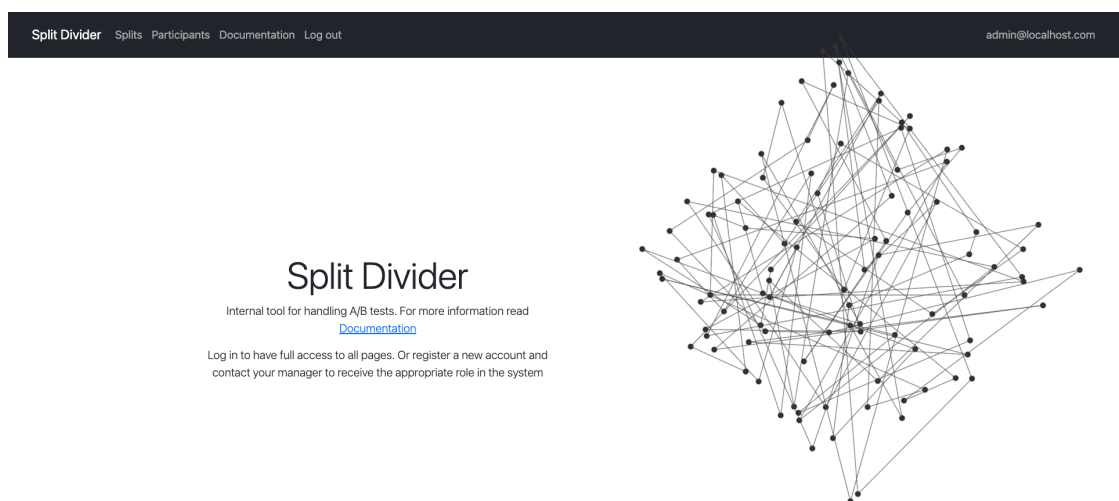


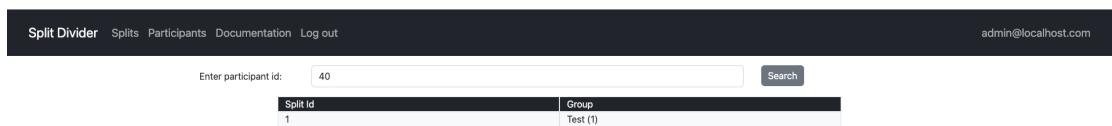
Рисунок 3.4 – Головна сторінка, кнопка «Log out» знаходиться на хедері сторінки.

3.4 Перегляд груп учасника

Користувач має змогу переглянути, у яких тестах бере участь учасник та до яких груп він у них належить. Для цього потрібно натиснути на кнопку

«Participants» у хедері веб-сайту, після чого користувач потрапить на сторінку пошуку.

У рядок пошуку потрібно ввести ідентифікатор учасника, після чого відобразиться інформація.

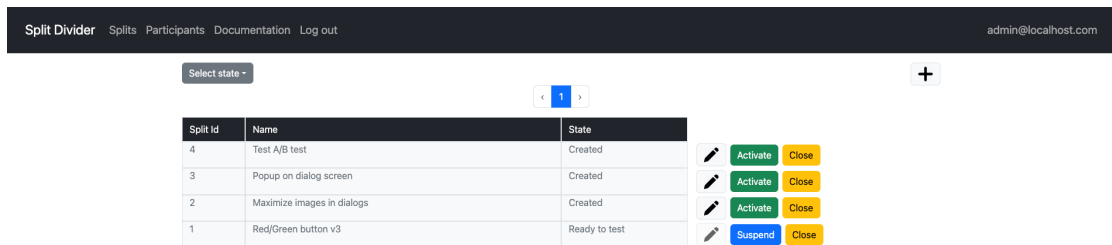


Split Id	Group
1	Test (1)

Рисунок 3.5 – Сторінка з інформацією про групи учасника.

3.5 Перегляд списку A/B тестів

Користувач має змогу переглянути список створених A/B тестів. Для цього потрібно натиснути на кнопку «Splits» у хедері веб-сайту, після чого користувач потрапить на сторінку зі списком тестів.



Split Id	Name	State
4	Test A/B test	Created
3	Popup on dialog screen	Created
2	Maximize images in dialogs	Created
1	Red/Green button v3	Ready to test

Рисунок 3.6 – Сторінка зі списком A/B тестів.

Також є можливість відфільтрувати A/B тести залежно від статусу. Для цього потрібно натиснути на кнопку «Select state», після чого з'явиться випадаючий список, у якому можна обрати необхідний статус.

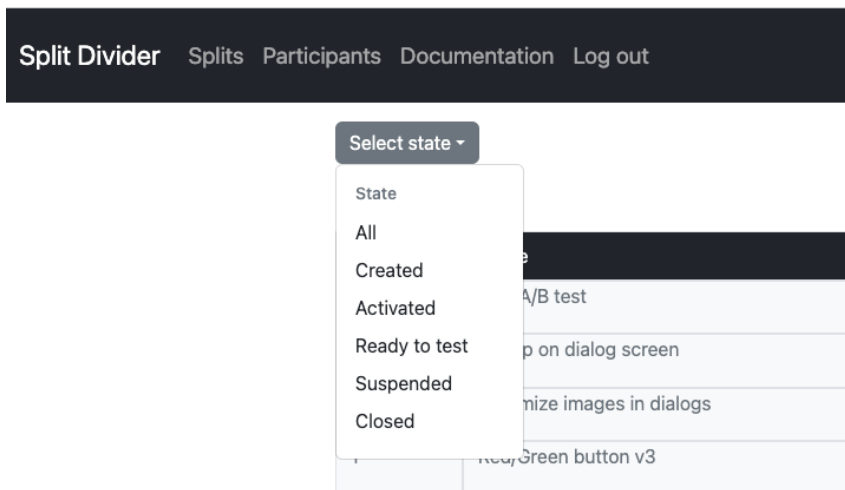


Рисунок 3.7 – Випадаючий список для фільтрації за статусом.

3.6 Перегляд інформації про A/B тест

Користувач має можливість переглянути повну інформацію про A/B тест. Для цього потрібно на сторінці зі списком натиснути на потрібний A/B тест.

Користувача буде перенаправлено на сторінку, де відображається вся інформація про тест.

Якщо тест було активовано, також буде відображатися інформація про розподіл учасників на групи. Окрім цього є можливість переглянути граф, який було побудовано при розподілі. Для цього потрібно натиснути на кнопку «Show graph».

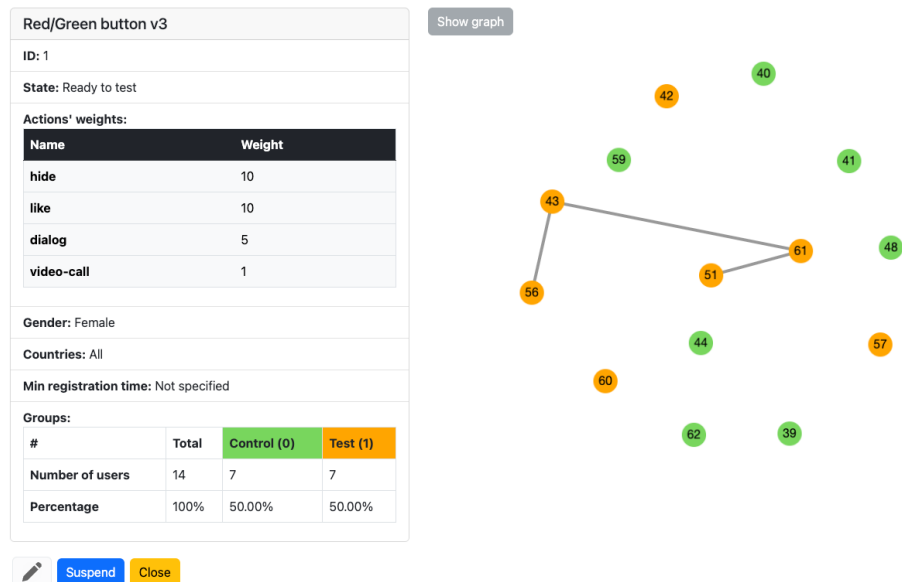


Рисунок 3.8 – Повна інформація про активний А/В тест.

3.7 Створення А/В тесту

Користувач має можливість створити новий А/В тест. Для цього потрібно на сторінці зі списком натиснути на кнопку створення «+». Після цього з'явиться модальне вікно, у якому необхідно заповнити дані А/В тесту.

При створенні обов'язково потрібно заповнити поля «Name» та «Actions' weights». Параметри сегментації є необов'язковими.

Після того, як всі дані введено, потрібно натиснути на кнопку «Submit».

Create new split

Name

Interactions' weights

Name	Weight
dialog	1
like	1
hide	1
video-call	1

Countries

Ukraine Poland USA Great Britain Germany

Gender

Male Female All

Minimal Registration Time

dd.mm.yyyy, --:--

Submit Cancel

Рисунок 3.9 – Модальне вікно для створення нового А/В тесту.

3.8 Редагування А/В тесту

Користувач має можливість редагувати уже створений А/В тест, якщо він знаходиться в статусах «Created» або «Suspended». Для цього потрібно натиснути на кнопку редагування. Вона є на сторінці з повною інформацією про А/В тест та на сторінці зі списком А/В тестів біля кожного тесту, який можна відредагувати.



Рисунок 3.10 – Кнопка редагування на сторінці з повною інформацією.

Після натискання з'являється модальне вікно, у якому можна відредагувати необхідну інформацію. Для підтвердження змін потрібно натиснути кнопку «Submit».

Update split ID=2

Name

Maximize images in dialogs

Interactions' weights

Name	Weight
hide	5
like	5
dialog	20
video-call	10

Countries

Ukraine

Poland

USA

Great Britain

Germany

Gender

Male

Female

All

Minimal Registration Time

03.05.2023, 00:00

Submit

Cancel

Рисунок 3.11 – Модальне вікно для редагування А/В тесту.

3.9 Запуск А/В тесту

Створений чи призупинений А/В тест можна активувати. Для цього потрібно натиснути на кнопку «Activate», яка є на сторінці з повною інформацією про А/В тест та на сторінці зі списком А/В тестів біля кожного тесту, який можна відредагувати.



Рисунок 3.12 – Кнопка активації на сторінці з повною інформацією.

Після натискання тест перейде у статус «Activated», а після успішного формування груп учасників, тест отримає статус «Ready to test», що означає, що можна проводити збір даних для майбутнього аналізу.

3.10 Призупинення А/В тесту

Активований А/В тест можна призупинити. Для цього потрібно натиснути на кнопку «Suspend», яка є на сторінці з повною інформацією про А/В тест та на сторінці зі списком А/В тестів біля кожного тесту, який можна призупинити.

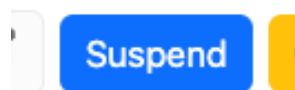


Рисунок 3.13 – Кнопка призупинення на сторінці з повною інформацією.

Після натискання тест перейде у статус «Suspended», а дані про розподіл та граф взаємодії буде видалено. Після того, як тест було призупинено, його можна буде активувати наново, щоб здійснити новий розподіл.

3.11 Закриття А/В тесту

Користувач має можливість закрити А/В тест. Для цього потрібно натиснути на кнопку «Close», яка є на сторінці з повною інформацією про А/В тест та на сторінці зі списком А/В тестів біля кожного тесту, який можна зупинити.

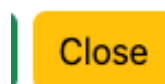
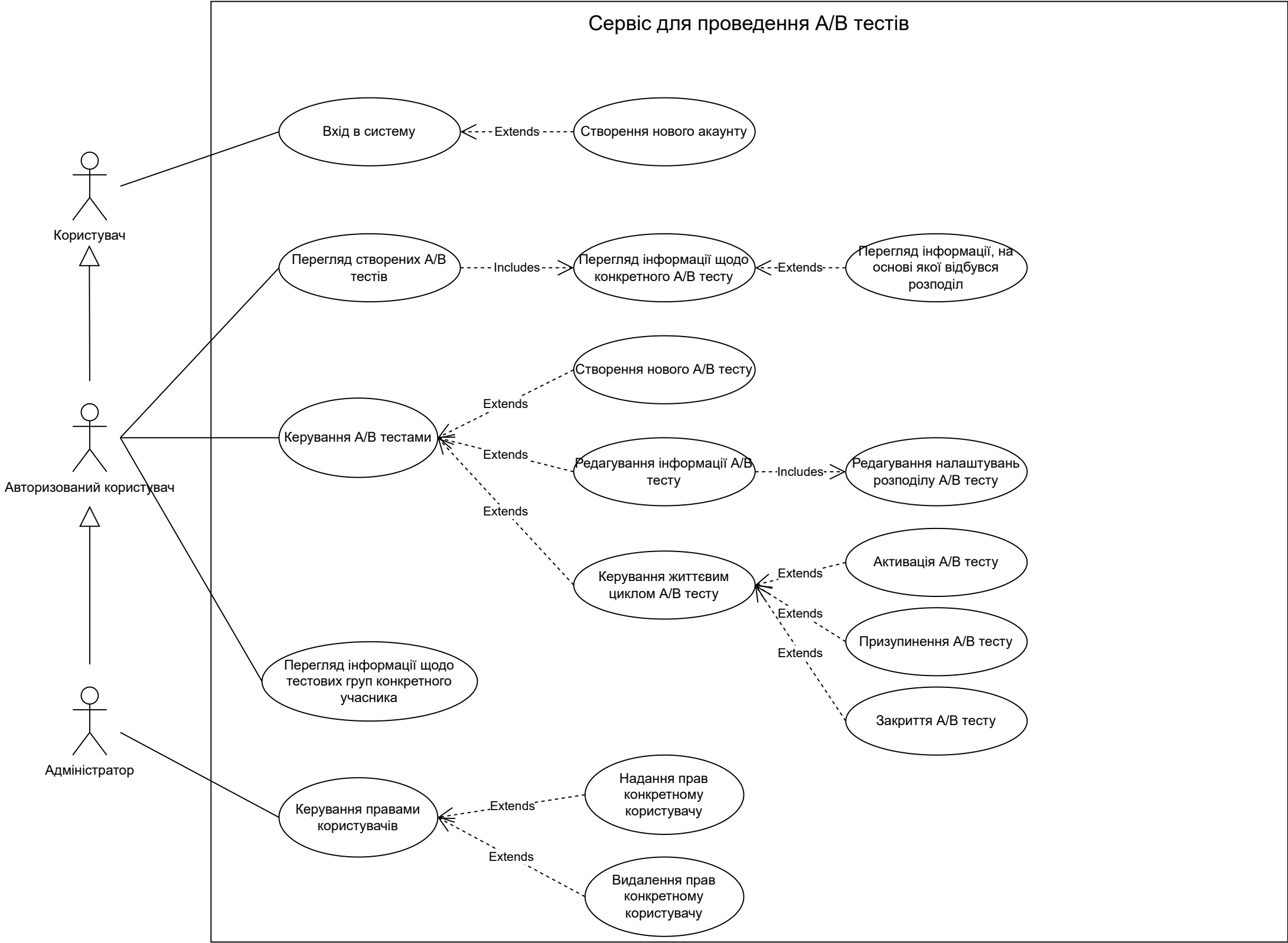


Рисунок 3.14 – Кнопка закриття на сторінці з повною інформацією.

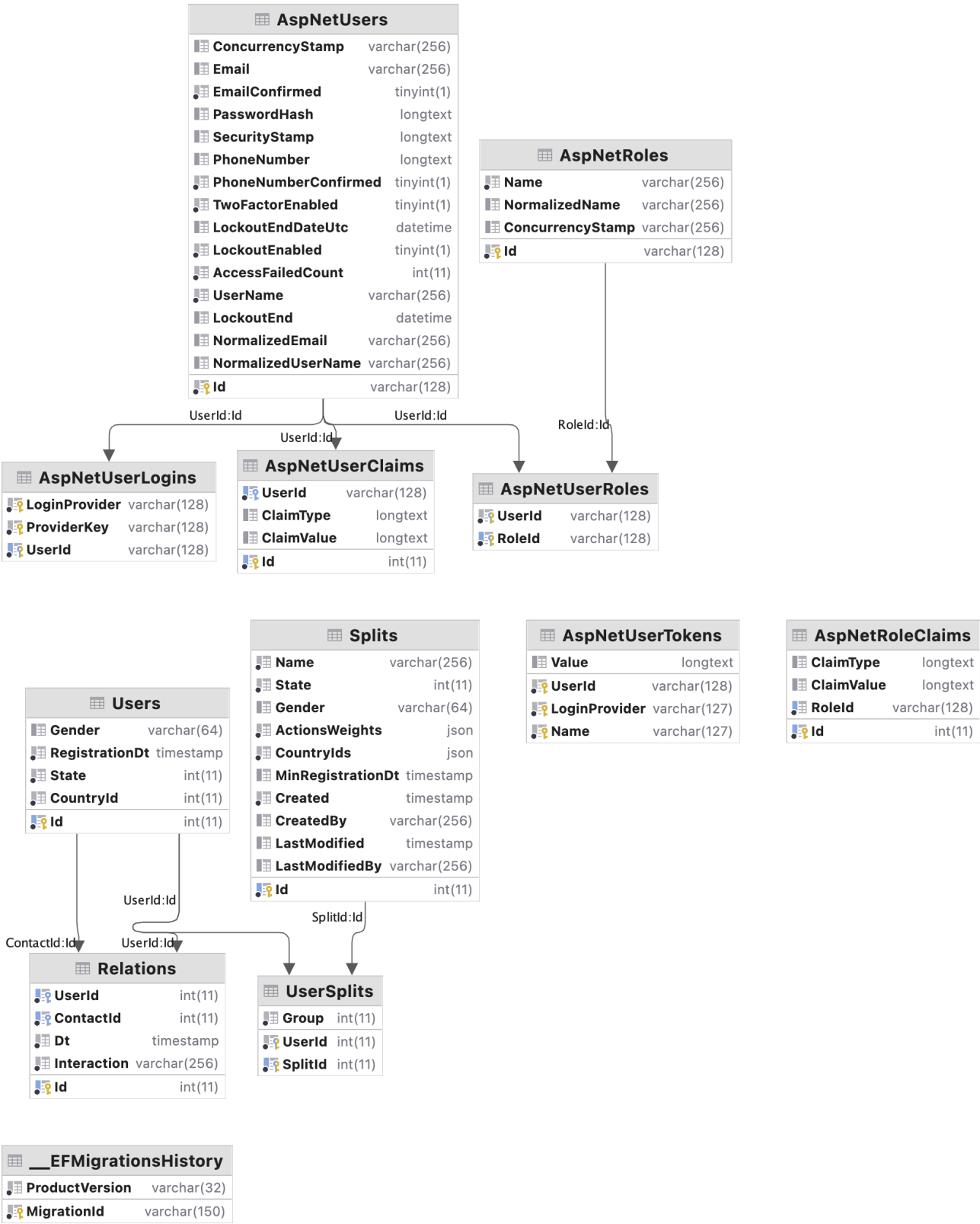
					КПІ.IT-9127.045440.05.34	Арк.
						11
Змін.	Арк.	№ докум.	Підп.	Дата.		

Після натискання тест перейде у статус «Closed». Цей статус є кінцевою точкою у життєвому циклі A/B тесту. Після переходу тест не можна буде змінювати.

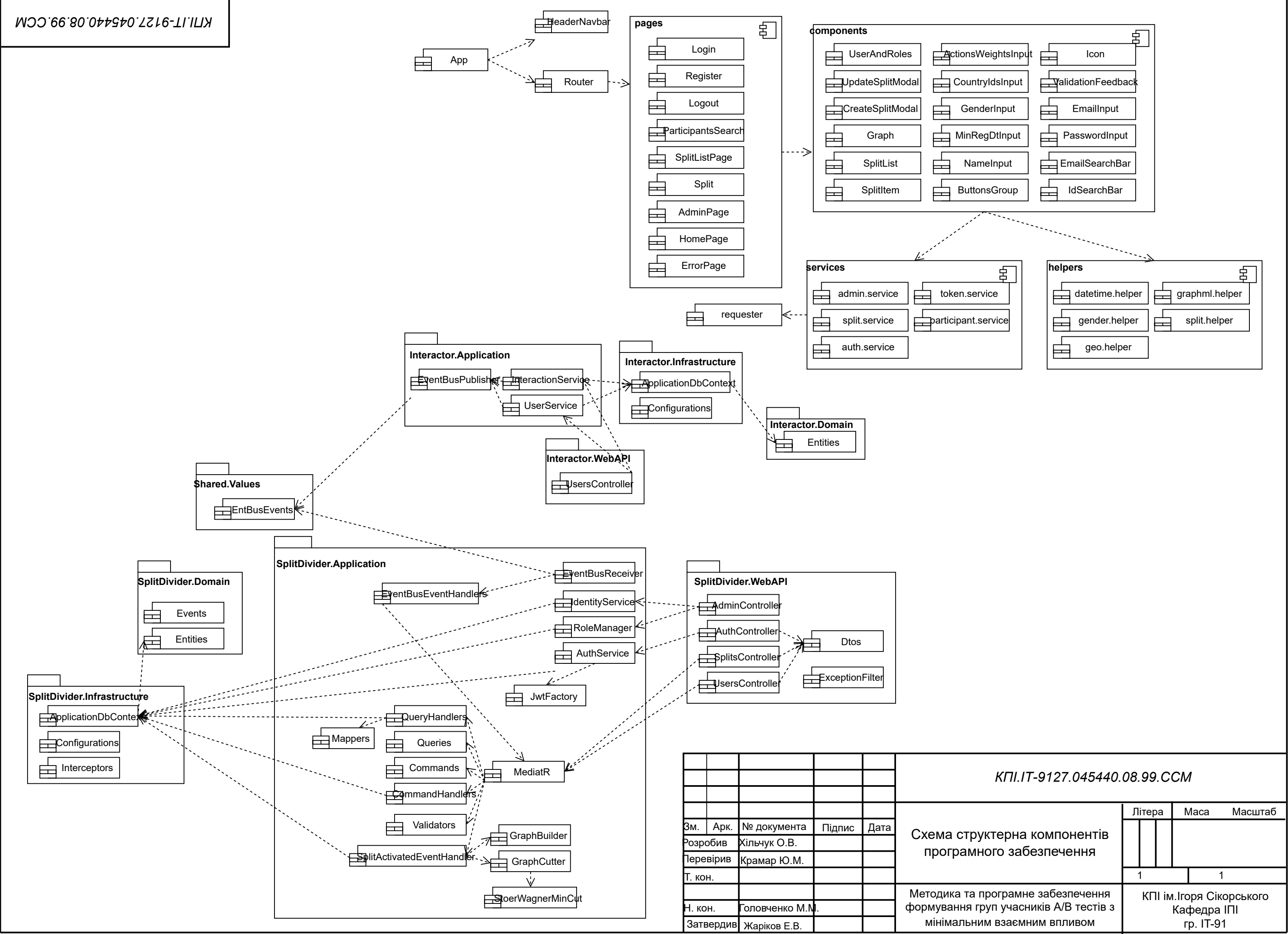
					КПІ.ІТ-9127.045440.05.34	Арк.
						12
Змін.	Арк.	№ докум.	Підп.	Дата.		



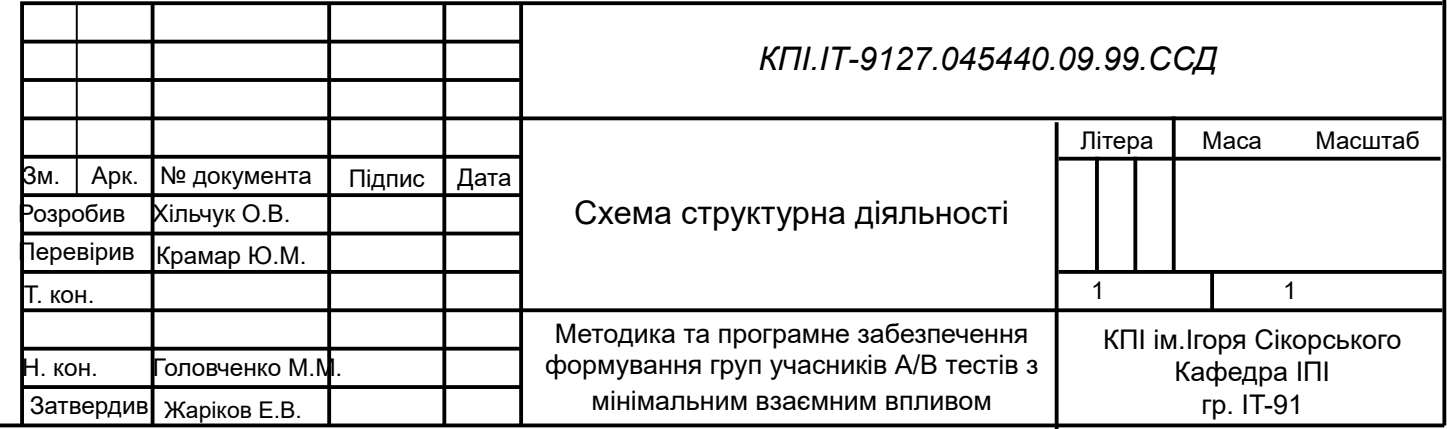
					КПІ.ІТ-9127.045440.06.99.ССВ										
					Схема структурна варіантів використань				Літера		Маса		Масштаб		
Зм.	Арк.	№ документа	Підпис	Дата											
Розробив		Хільчук О.В.													
Перевірів		Крамар Ю.М.													
Т. кон.															
									1		1				
Н. кон.		Головченко М.М.			Методика та програмне забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91						
Затвердив		Жаріков Е.В.													



					КПІ.ІТ-9127.045440.07.99.СБД				
					Схема бази даних				
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив	Хільчук О.В.								
Перевірив	Крамар Ю.М.								
Т. кон.									
Н. кон.	Головченко М.М.				Методика та програмне забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом		КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91		
Затвердив		Жаріков Е.В.							



					КПІ.ІТ-9127.045440.08.99.ССМ						
					Схема структурна компонентів програмного забезпечення	Літера			Маса	Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата							
Розробив	Хільчук О.В.										
Перевірів	Крамар Ю.М.										
Т. кон.											
					Методика та програмне забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом	1			1		
Н. кон.	Головченко М.М.					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91					
Затвердив	Жаріков Е.В.										



					КПІ.ІТ-9127.045440.09.99.ССД					
					Схема структурна діяльності	Літера			Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата						
Розробив		Хільчук О.В.								
Перевірів		Крамар Ю.М.								
Т. кон.										
					Методика та програмне забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом	1			1	
Н. кон.		Головченко М.М.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91				
Затвердив		Жаріков Е.В.								

Create new split

Name

Interactions` weights

Name	Weight
dialog	1
like	1
hide	1
video-call	1

Countries

Ukraine

Poland

USA

Great Britain

Germany

Gender

Male

Female

All

Minimal Registration Time

dd.mm.yyyy. --:--

privacy

Split Divider
Splits
Participants
Log out

username

Frame 3

Split Name

ID: 1

State: Ready to test

Actions' weights

Name	Weight
like	10
like	10
chatting	5
video-call	1

Gender: Female

Countries: All

Min registration time: Not specified

#	Total	Control (0)	Test (1)
Number of users	14	7	7
Percentage	100%	50.00%	50.00%

...

Suspend

Close

Show graph

Split Divider

Splits

Participants

Log out

username

NAME

Frame 1

State

id	Name	State

Activate

Suspend

Close

Split Divider

Splits

Participants

Log out

username

participant id

Search

Split id	Group

[Split Divider](#) [Splits](#) [Participants](#) [Log in](#)

					КПІ.ІТ-9127.045440.10.99.КЕ				
					Креслення вигляду екранних форм	Літера		Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив	Хільчук О.В.								
Перевірів	Крамар Ю.М.								
Т. кон.						1	1		
					Методика та програмне забезпечення формування груп учасників А/В тестів з мінімальним взаємним впливом	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91			
Н. кон.	Головченко М.М.								
Затвердив	Жаріков Е.В.								