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 інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-застосунок для підтримки діяльності організаторів командних аматорських ігор

Виконав студент IV курсу, групи ІТ-94
(шифр групи)

Подмоков Андрій Вячеславович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., ст. викл., Стельмах О. П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Рецензент доцент кафедри ІСТ, к.т.н., доц., Олійник В. Г. _____
(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 інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Подмокову Андрію Вячеславовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Веб-застосунок для підтримки діяльності організаторів командних аматорських ігор

керівник проєкту Стельмах Олександр Петрович, к.т.н., ст. викл.
(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: загальні положення, змістовний опис і аналіз предметної області, аналіз існуючих технологій та успішних ІТ-проєктів, аналіз вимог до програмного забезпечення, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки даних.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.
4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, підтримка програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) 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	20.03.2023	
3	Постановка та формалізація задачі	25.03.2023	
4	Розробка інформаційного забезпечення	15.04.2023	
5	Алгоритмізація задачі	25.04.2023	
6	Обґрунтування вибору використаних технічних засобів	28.04.2023	
7	Розробка програмного забезпечення	17.05.2023	
8	Налагодження програми	27.05.2023	
9	Виконання графічних документів	30.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	
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 _____ Андрій ПОДМОКОВ
(підпис) (ініціали, прізвище)

Керівник _____ Олександр СТЕЛЬМАХ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 45 таблиць, 40 рисунків та 10 джерел – загалом 74 сторінки.

Дипломний проєкт присвячений написанню веб-застосунку для підтримки діяльності організаторів командних аматорських ігор.

Мета підтримка діяльності організаторів командних аматорських ігор.

Об'єкт дослідження: командні аматорські ігри

Предмет дослідження: веб-застосунок для підтримки діяльності організаторів командних аматорських ігор

У розділі «аналіз вимог до програмного забезпечення» визначено загальні положення, представлений опис та аналіз предметної області, аналіз успішних IT-проектів, аналіз вимог до програмного забезпечення та постановка задачі.

Розділ «моделювання та конструювання програмного забезпечення» описує моделюванню та аналізу програмного забезпечення, архітектурі програмного забезпечення, конструюванню програмного забезпечення та аналізу безпеки даних.

Наступний розділ «аналіз якості та тестування програмного забезпечення» присвячений аналізу якості програмного забезпечення, опису процесів тестування та опису контрольного прикладу.

Останній розділ «впровадження та супровід програмного забезпечення» розкриває такі питання, як розгортання програмного забезпечення, підтримка програмного забезпечення.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, MERN, MONGODB.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 45 tables, 40 figures and 10 sources – in total 74 pages.

The purpose of the diploma project is to support the activities of the organizers of amateur team games.

Object of research: amateur team games.

Subject of research: a web application to support the activities of the organizers of amateur team games.

In the "analysis of software requirements" section, general provisions are defined, a description and analysis of the subject area, an analysis of successful IT projects, an analysis of software requirements and a problem statement are presented.

The software modeling and engineering section describes software modeling and analysis, software architecture, software engineering, and security data analysis.

The next section "software quality analysis and testing" is devoted to software quality analysis, a description of the testing processes and a description of the control example.

The last chapter "software implementation and maintenance" covers such issues as software deployment, software support.

KEYWORDS: WEB APPLICATION, MERN, MONGODB.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9419.045440.01.91	Технічне завдання	16	
3	А4	КПІ.ІТ-9419.045440.02.81	Пояснювальна записка	74	
4	А4	КПІ.ІТ-9419.045440.03.12	Текст програми	16	
5	А4	КПІ.ІТ-9419.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9419.045440.05.34	Керівництво користувача	19	
7	А3	КПІ.ІТ-9419.045440.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІТ-9419.045440.07.99	Схема бази даних	1	
9	А3	КПІ.ІТ-9419.045440.08.99	Креслення вигляду екранних форм	1	

					КПІ.ІТ-9419.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Подмоков А.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Стельмах О.П.					
Н.контр.		Вітковська І.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-94	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ОРГАНІЗАТОРІВ КОМАНДНИХ АМАТОРСЬКИХ ІГОР

Технічне завдання

КПІ.ІТ-9419.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
Олександр СТЕЛЬМАХ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Андрій ПОДМОКОВ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосунок для підтримки діяльності організаторів командних аматорських ігор.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення для веб-застосунку для підтримки діяльності організаторів командних аматорських ігор, котра використовується для організації командних аматорських ігор та призначена для організаторів командних аматорських ігор та людей, що бажають прийняти участь у командних аматорських іграх.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку для підтримки діяльності організаторів командних аматорських ігор є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для полегшення організації командних аматорських ігор.

Метою розробки є підтримка діяльності організаторів командних аматорських ігор.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

У користувача є можливість зареєструватись та авторизуватись. При натискані на кнопку “Login” на сторінці реєстрації, користувача перенаправляє на сторінку авторизації, аналогічно на сторінці авторизації при натискані на кнопку “Реєстрація”, користувача перенаправляє на сторінку реєстрації. Після натискання кнопки “Зареєструватися”, користувача перенаправляє на сторінку авторизації. При натискані на кнопку “Login” на сторінці авторизації, користувача перенаправляє на головну сторінку. На рисунку 4.1 зображено прототип сторінки реєстрації та авторизації.

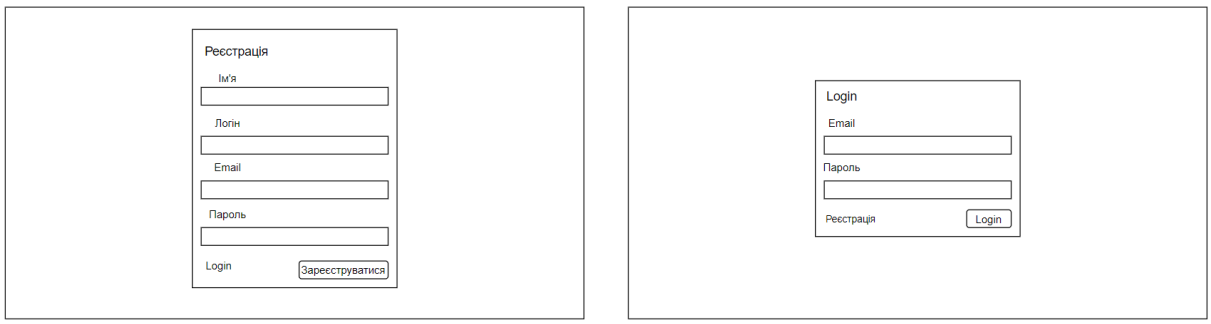


Рисунок 4.1 – Прототип сторінки реєстрації та авторизації

Під час взаємодія користувача з деякими елементами інтерфейсу, відображається повідомлення з статусом виконання операції. Окрім цього, при відправці, отриманні даних відображається анімація завантаження. На

рисунку 4.2 зображено прототип відображення завантаження та повідомлення.

Рисунок 4.2 – Прототип завантаження та повідомлення

На головній сторінці є поле пошуку, список створених ігор, зверху кнопка для звертання меню, зліва меню для навігації по сайту, знизу навігація по списку створених ігор, з можливістю обрати скільки відображати ігор на сторінці. При натисканні на матч користувача перенаправляє на сторінку матчу. На рисунку 4.3 зображено прототип головної сторінки та головної сторінки зі згорнутим меню.

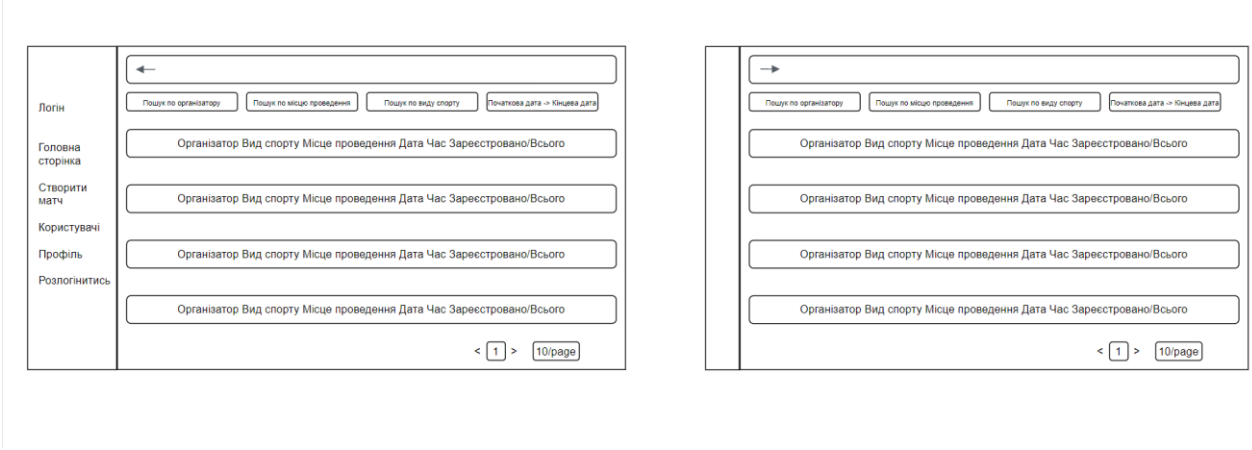


Рисунок 4.3 – Прототип головної сторінки та головної сторінки зі згорнутим меню

На сторінці матчу знаходиться вся інформація про матч, а також кнопка “Зареєструватися” або “Скасувати реєстрацію” на матч. При натисканні на кнопку “Зареєструватися”, користувач реєструється на матч і

перенаправляється на сторінку профілю, а при натисканні на кнопку “Скасувати реєстрацію”, користувач скасовує реєстрацію і перенаправляється на головну сторінку. На рисунку 4.4 зображено прототип сторінки матчу.

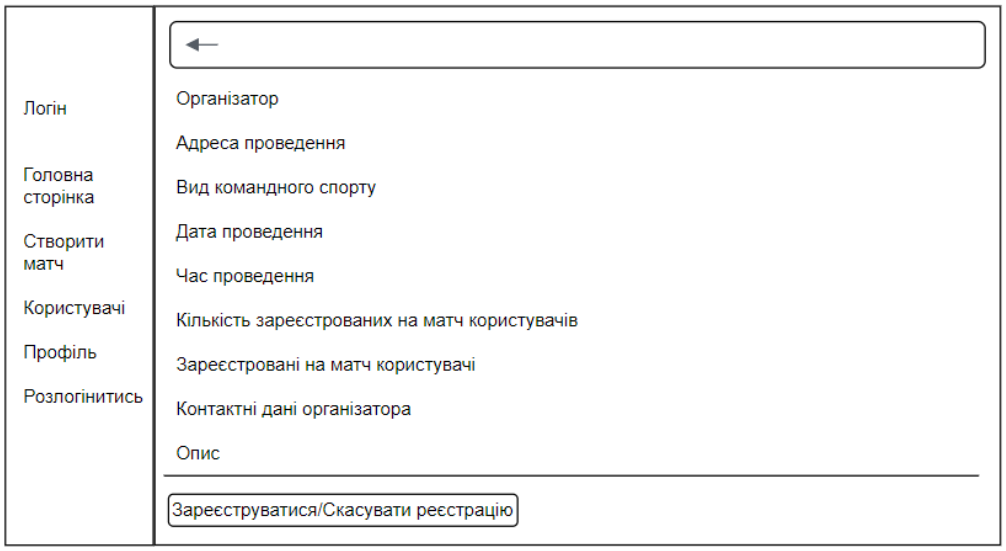


Рисунок 4.4 – Прототип сторінки матчу

На сторінці “Профіль” є два відділи – “Matches” та “Rating”, відділ на якому знаходиться користувач підсвічується синім. На відділі “Matches” відображаються матчі, на які користувач зареєстрований, а на відділі “Rating” відображаються відгуки про користувача. На рисунку 4.5 зображено прототип сторінки профілю.

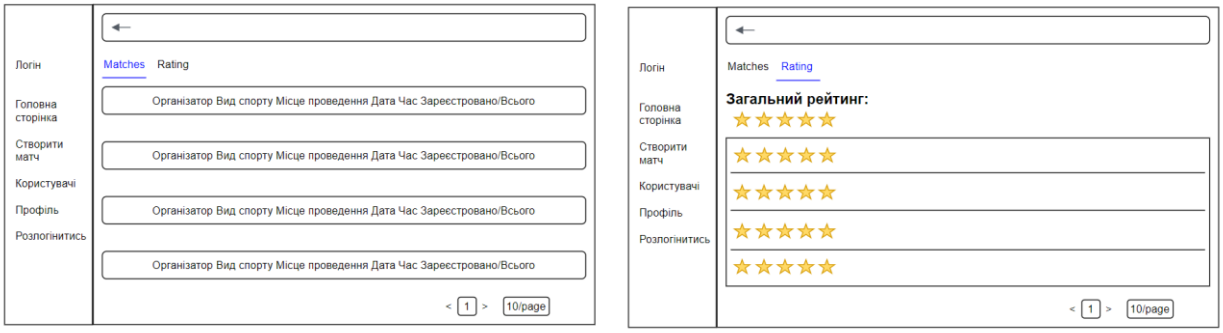


Рисунок 4.5 – Прототип сторінки “Профіль”

На основній частині сторінки “Створити матч” знаходиться таблиця зі створеними матчами та кнопками для їх редагування/видалення. Також там

знаходиться кнопка “Створити матч”, при натисканні на неї, відкривається модальне вікно для створення матчу, таке ж саме вікно відкривається й при натисканні на кнопку редагування матчу, але вже з заповненими даними про матч. На рисунку 4.6 зображено прототип сторінки створення матчу та модального вікна для створення/редагування матчу.

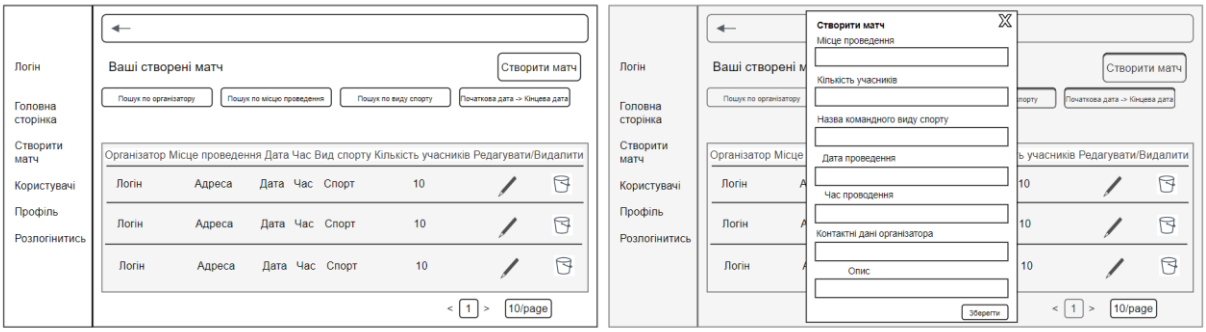


Рисунок 4.6 – Прототип сторінки “Створити матч” та модального вікна для створення/редагування матчу

Сторінка “Користувачі” розділяється для адміністратора та звичайного користувача. Для адміністратора користувачі відображаються у вигляді таблиці та з можливістю редагування їх статус, при натисканні на логін користувача, йде перенаправлення на сторінку цього користувача. Для звичайного користувача користувачі відображаються у вигляді списку, при натисканні на користувача, йде перенаправлення на сторінку цього користувача, На рисунку 4.7 зображено прототип сторінки “Користувачі” для адміністратора та звичайного користувача.

Рисунок 4.7 – Прототип сторінки “Користувачі” для адміністратора та звичайного користувача

Сторінка користувача розділяється для адміністратора та звичайного користувача. У звичайного користувача відображаються відгуки про користувача і кнопка “Додати відгук”, а у адміністратора ще є кнопки для видалення відгуку. При натисканні на кнопку “Додати відгук”, відкривається модальне вікно для додавання відгуку. На рисунку 4.8 зображено прототип сторінки користувача для адміністратора та звичайного користувача і модального вікна для додавання відгуку.

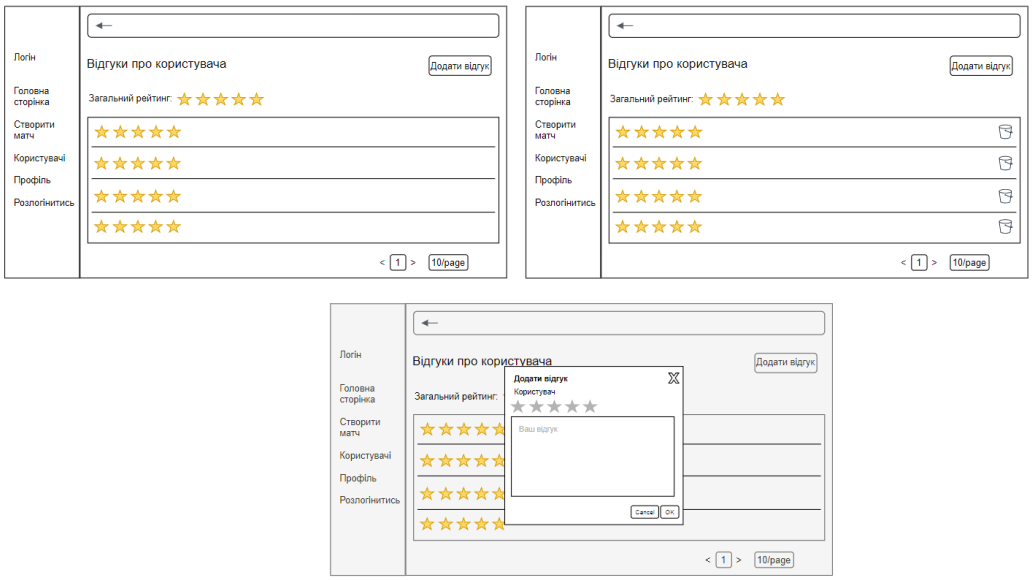


Рисунок 4.8 – Прототип сторінки користувача для адміністратора та звичайного користувача і модального вікна для додавання відгуку

4.1.2 Для користувача:

– реєстрація;
– авторизація;
– пошук бажаної гри;
– пошук користувачів;
– створення гри;
– керування своїми створеними іграми;
– залишення відгуків про користувачів;
– реєстрація/скасування реєстрації на гру.

4.1.3 Для адміністратора системи:

– керування створеними іграми;
– керування правами користувачів;
– керування відгуками користувачів.
– залишення відгуків про користувачів;
– створення гри;
– реєстрація/Скасування реєстрації на гру;
– пошук бажаної гри;
– пошук користувачів.

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.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– операційна система: Windows 7 або вище, MacOS 10.12 Sierra або вище, Linux (будь-який сучасний дистрибутив, що підтримує браузери останньої версії);
– процесор: 1 GHz або швидше;

– об‘єм ОЗП: щонайменше 1 GB;

– жорсткий диск: щонайменше 1 GB вільного місця для зберігання кешу браузера та інших тимчасових файлів;

– відеокарта: будь-яка сумісна з операційною системою;

– дисплей: рекомендована роздільна здатність 1024x768 або вище;

– інтернет з'єднання: широкосмугове.

Рекомендована конфігурація технічних засобів:

– операційна система: Windows 10;

– процесор: Intel Core i5;

– об‘єм ОЗП: 16 ГБ;

– жорсткий диск: щонайменше 2 GB вільного місця для зберігання кешу браузера та інших тимчасових файлів;

– відеокарта: NVIDIA GeForce GTX 1050 Ti 4GB;

– дисплей: роздільна здатність 1920x1080;

– інтернет з'єднання: широкосмугове, зі стабільною швидкістю для комфортної взаємодії з веб-застосунком.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати на наступних веб-браузерах: Google Chrome, Mozilla Firefox, Safari або Microsoft Edge останньої версії;

4.5.1 Вимоги до вхідних даних

Вимоги до вхідних даних не висуваються.

4.5.2 Вимоги до вихідних даних

Вимоги до вихідних даних не висуваються.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування JavaScript.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища розробки Visual Studio Code.

4.5.5 Вимоги до представленню вихідних кодів

Вимоги до представленню вихідних кодів не висуваються.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

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 бази даних;

– креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Спеціальні вимоги до програмної документації не висуваються.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосунок для підтримки діяльності організаторів командних аматорських ігор

КПІ.ІТ-9419.045440.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
IT	–	Інформаційні технології
БД	–	База даних.
JWT	–	JSON Web Token, відкритий стандарт для створення токенів доступу.
JSON	–	JavaScript Object Notation, текстовий формат обміну даними.
JS	–	JavaScript, мова програмування.
DOM	–	Document Object Model.
NPM	–	Node Package Manager.
ПЗ	–	Програмне забезпечення.

ВСТУП

У наше сьогодення спорт є одним з найбільш популярних способів проведення свого дозвілля, а також необхідністю для підтримки здоров’я. Командні види спорту завжди були популярними, вони викликають захват та великий інтерес серед різних прошарків людей навколо всього світу. Окрім цього, спорт є неймовірно необхідною частиною у сучасному суспільстві, тому що більшість існуючих професій та повсякденних занять пов'язані з сидінням перед комп'ютером або використанням сучасних технологій. Це призводить до зменшення фізичної активності і проведення більшої частини часу в статичному положенні, що негативно впливає на стан здоров’я людини. Враховуючи це важливо допомагати людям починати займатись спортом та додавати більше фізичної активності.

Хоч і різні види командного спорту є дуже популярними, проте люди не поспішають приймати в них участь. Більшість людей просто спостерігає за змаганням, сидячи на дивані або дивлячись їх через свої портативні пристрої. Це проявляється у тому що люди стикаються з недостатнім часом для занять спортом через заповнений розклад, роботу, сімейні зобов'язання та інші соціальні активності. Внаслідок цього, спорт відступає на задній план. Крім того, постає проблема у відсутності спортивних майданчиків, тренажерних залів або інших спортивних об'єктів у близькості. Навіть якщо спортивні майданчики є у пішій доступності, виникає проблема у пошуку людей для спільного зайняття спортом та команди для гри у футбол, баскетбол, волейбол і т.д.

Пошук команди для гри також є значним бар’єром у початку займанням спортом. Люди стикаються з тим, що інформація про доступні команди та спортивні клуби може бути обмеженою або недоступною. Це ускладнює пошук команди, оскільки люди не знають, де шукати інформацію або де знаходяться команди, які відповідають їхнім інтересам. Окрім цього, знайти команду, яка відповідає вашому графіку також може бути викликом.

У всіх різний робочий графік, сімейні зобов'язання та інші зайнятості, що можуть впливати на їхню можливість проведення вільного часу.

Тож актуальність даного веб-застосунку проявляється як ніколи, адже його мета – зробити так, щоб допомогти людям з вирішенням цих проблем і спрощенням можливості організовувати різні спортивні аматорські ігри з командних видів спорту.

Також актуальність проявляється у необхідності розв'язання проблеми організації аматорських ігор з командних видів спорту в онлайн середовищі. Адже існуючих аналогів майже немає, а зростання інформаційних технологій надає нові можливості для спілкування та залучення спортивних ентузіастів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Командні аматорські ігри стають все більш популярними в сучасному світі. Та стан речей у командних видів спорту може бути різноманітним, залежно від конкретного виду спорту та контексту. Проте всіх їх об’єднує те, що вони потребують ефективного організаційного інструменту. який може полегшити планування ігор, визначення місця, часу проведення, пошук команди.

Сучасні технології веб-розробки надають широкий спектр можливостей для вирішення цих потреб. Тож цілком доцільно розробити веб-застосунок для організації аматорських командних ігор.

Подібний веб-застосунок повинен охоплювати різні аспекти організації гри, включаючи планування, вибір місця, часу проведення, пошук команди. Крім того, застосунок може давати можливість дізнатись більше про людей, що збираються прийняти участь у грі.

Хоч й існують деякі розробки у цій сфери, вони повність не задовольняють конкретні потреби користувачів [1].

1.2 Змістовний опис і аналіз предметної області

На сьогоднішній день існує мала кількість веб-застосунків, які спрямовані на організацію аматорських ігор з командних видів спорту. Поточний стан речей в цій області відзначається певними недоліками. Наприклад, існуючі додатки не надають достатньої функціональності і зручності, а також не задовольняють різним потребам людей. Для того щоб імплементувати дану предметну область у ІТ необхідно розуміти, що саме необхідно людям на даний момент, а це – можливість зручно і швидко знайти місце і людей, з якими можна влаштувати спортивний івент, обрати зручний для них час, і отримати мінімальне уявлення про людей, з якими вони будуть

приймати участь. Проте великі компанії не зацікавлені у вирішені даних проблем, тому що вони більше не про прибуток, а про користь людям.

Тож, проаналізувавши таке явище як командні види спорту, можна перенести їх у сферу ІТ, додавши зручний і простий інтерфейс для користувачів, з можливістю самостійно створювати спортивні івенти за зручною для них формулою.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку для підтримки діяльності організаторів командних аматорських ігор. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Обрана предметна область не потребує реалізації алгоритмів для розв’язання задач і підзадач. Тож далі буде розглянуто відомі технічні рішення, а саме відомі архітектури ПЗ.

Монолітна архітектура – це традиційний архітектурний стиль, у якому всі компоненти програми тісно пов’язані та працюють в одному процесі. Недоліком цієї архітектури є те, що в міру збільшення розміру програми вона стає складнішою та важчою для підтримки. Будь-які зміни в окремому компоненті вимагають перебудови та розгортання всієї програми, що призводить до меншої гнучкості.

Архітектура мікросервісів – це архітектура, у якій додаток розділено на слабо пов’язані служби, які можна розгортати незалежно, кожна з яких працює у своєму процесі. Хоч ця архітектура і збільшує складність розгортання, вона пропонує більшу гнучкість і масштабованість, особливо для великих програм із декількома незалежними модулями.

Клієнт-серверна архітектура є широко використовуваним підходом у розробці програмного забезпечення, де функції та обов'язки системи розділяються між двома основними компонентами: клієнтом і сервером.

Клієнт - надає засоби для введення даних, відображення результатів та взаємодії з сервером. Клієнт може бути веб-браузером, мобільним додатком, або будь-яким іншим інтерфейсом, який використовується користувачем.

Сервер - це центральний компонент системи, який надає послуги та обробляє запити від клієнтів. Він відповідає за обробку бізнес-логіки, зберігання даних, доступ до ресурсів та надання відповідей клієнтам. Сервер може бути фізичним апаратним пристроєм або програмою, яка працює на хостинговому середовищі.

Основним принципом клієнт-серверної архітектури є розділення обов'язків між клієнтом і сервером, що дозволяє покращити масштабованість, безпеку та обробку даних. Клієнт і сервер взаємодіють між собою за допомогою комунікаційного каналу, такого як мережа, де клієнт надсилає запити до сервера, а сервер обробляє їх і надсилає відповіді назад. На рисунку 1.1 зображена клієнт-серверна архітектура.

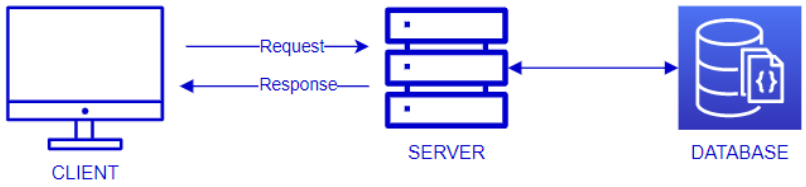


Рисунок 1.1 - Візуалізація клієнт-серверної архітектури

Клієнт-серверна архітектура має кілька переваг, зокрема:

– розділення обов'язків: Клієнт та сервер виконують різні функції, що дозволяє легко змінювати або розширювати кожну компоненту незалежно від іншої;

– масштабованість: Система може бути масштабована шляхом додавання додаткових серверів або розподілу завдань між ними;

– ефективність: Розподіл завдань між клієнтом та сервером дозволяє розподіляти навантаження та оптимізувати роботу системи.

Тож, виходячи з цього усього, було вирішено використовувати клієнт-серверну архітектуру для реалізації ПЗ.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Однією з головних необхідністой у виборі мови програмування є її розповсюдженість, для пошуку можливих рішень і простота її використання. Окрім цього, для створення веб-застосунку потрібна така мова, щоб можна було реалізувати і клієнтську, і серверну частину.

JavaScript – є однією з найпопулярніших мов для веб-розробки. В основному він використовується на стороні клієнта, щоб зробити веб-сторінки інтерактивними, але з появою Node.js його можна використовувати і на стороні сервера. Основні переваги JavaScript включають його широке поширення, велику спільноту, величезну екосистему бібліотек і фреймворків (наприклад, React.js, Angular.js, Vue.js для інтерфейсної розробки та Express.js для серверної), а також керовану подіями неблокуючу модель вводу-виводу, що робить її ефективною для додатків, що працюють в реальному часі.

Python – відомий своєю простотою та зручністю читання, що робить його чудовою мовою для початківців. Він має потужну підтримку інтеграції з іншими мовами та інструментами, а також поставляється з великими стандартними бібліотеками. Django та Flask — два популярних фреймворки для веб-розробки на Python. Однак Python може бути не таким ефективним, як JavaScript або Java, коли йде справа про застосунки, що працюють в реальному часі.

Java — це універсальна об’єктно-орієнтована мова програмування, яка використовується у веб-розробці, а також у багатьох інших програмах. Він відомий своєю надійністю та масштабованістю. Spring — популярний

фреймворк для створення веб-застосунків на Java. Однак Java є більш складною у вивчені порівняно з Python або JavaScript.

PHP — це серверна мова програмування, яка широко використовується для веб-розробки. З ним легко почати, особливо для невеликих проектів, оскільки його можна вставляти безпосередньо в HTML. Однак PHP не настільки підходить для великих, складних програм, і він не такий сучасний і швидкий, як JavaScript або Python.

Ruby разом зі своїм популярним фреймворком Ruby on Rails відомий своєю простотою та швидкістю розробки. Він дотримується принципів «Конвенція над конфігурацією» та DRY (Don’t Repeat Yourself), що робить його хорошим вибором для швидкої розробки. Однак Ruby може бути не таким ефективним у обробці сайтів із високим трафіком.

Враховуючи контекст розробки веб-застосунку для підтримки діяльності організаторів командних аматорських ігор, JavaScript (з його різними зовнішніми та внутрішніми фреймворками) є чудовим вибором. Він забезпечує єдину мову як для клієнта, так і для сервера, має велику спільноту з відкритим вихідним кодом.

Після того як було обрано мову програмування, слід обрати стек розробки.

Стек MEAN (MongoDB, Express.js, Angular.js, Node.js): Подібно до MERN, MEAN також використовує JavaScript як для інтерфейсу, так і для серверу, забезпечуючи плавний процес розробки. Основна відмінність полягає в інтерфейсній структурі Angular.js, яка є комплексною, багатофункціональною структурою. Однак його часто вважають більш складним для вивчення, ніж React.js, який використовується в MERN. Також варто зазначити, що React.js, як правило, має кращу продуктивність, ніж Angular.js, завдяки своїй віртуальній реалізації DOM.

Стек LAMP (Linux, Apache, MySQL, PHP): цей стек перевірений часом, однак його мова, PHP, втратила популярність порівняно з JavaScript через меншу гнучкість, повільнішу швидкість виконання та менш активну

спільноту. Крім того, MySQL не масштабується горизонтально так добре, як MongoDB.

Виходячи з цього, було прийнято рішення використовувати MERN стек архітектуру, яка складається з наступних компонентів:

– mongoDB - документоорієнтована система управління базами даних;

– express - фреймворк для створення веб-застосунків;

– react - бібліотека JavaScript для створення користувацьких інтерфейсів;

– node.js - кросплатформенне середовище розробки JavaScript;

– додаткові утиліти та бібліотеки.

На рисунку 1.2 зображено MERN стек архітектуру.

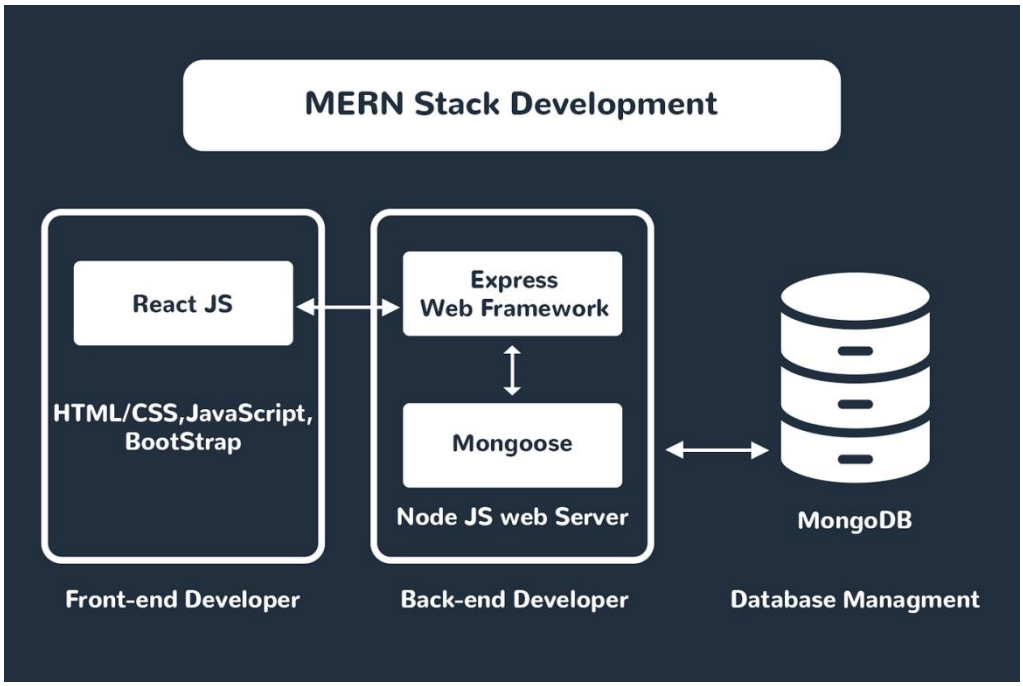


Рисунок 1.2 - Візуалізація MERN стек архітектури
(https://www.slideshare.net/Surya937648/introduction-to-mern-stack)

Node.js – програмна платформа, що перетворює JavaScript у мову загального призначення.

Node.js надає зручність в розробці серверних додатків, оскільки дозволяє розробникам використовувати JavaScript як мову програмування із великою екосистемою модулів та бібліотек. Це дозволяє розробникам

будувати швидкі та масштабовані серверні додатки, обробляти запити від клієнтів, здійснювати операції з базами даних, взаємодіяти з файловою системою та багато іншого.

Однією з основних переваг Node.js є його асинхронна та подієва модель програмування, що дозволяє ефективно обробляти багато одночасних запитів без блокування виконання. Це досягається за допомогою використання неблокуючих операцій вводу/виводу та зворотного виклику (callback) для обробки результатів операцій.

Node.js також має потужну систему пакетного менеджера, відому як npm (Node Package Manager), яка дозволяє розробникам легко установлювати, оновлювати та керувати залежностями своїх проектів.

Завдяки своїй швидкості, простоті використання та широкому спектру можливостей, Node.js став популярним серед розробників для створення серверних додатків, мережевих додатків, API, чат-ботів, веб-серверів та інших застосувань [5].

React - це відкрита JavaScript-бібліотека, яка використовується для розробки інтерфейсів користувача. Вона була створена компанією Facebook і в даний час активно підтримується та використовується широким загалом розробників.

Однією з ключових особливостей React є його компонентний підхід до розробки. Він дозволяє розбити користувацький інтерфейс на незалежні компоненти, які можна повторно використовувати та комбінувати для побудови складних інтерфейсів. Компоненти React можуть мати внутрішній стан (state), що дозволяє їм реагувати на зміни даних та оновлювати відповідний візуальний вихід.

React також пропонує використання віртуального DOM (Document Object Model). Замість безпосереднього маніпулювання реальним DOM, React працює з віртуальним представленням сторінки, яке потім ефективно оновлює реальний DOM тільки в тих місцях, де відбулися зміни. Це

призводить до покращеної продуктивності та швидкодії веб-додатків, особливо в разі змінюваного інтерфейсу [3].

MongoDB – документоорієнтована, нереляційна база даних, яка зберігає дані у вигляді документів у форматі JSON.

Однією з ключових особливостей MongoDB є гнучкість схеми даних. У відмінності від традиційних реляційних баз даних, де потрібно передбачити та визначити структуру таблиць і поля заздалегідь, MongoDB дозволяє динамічно зберігати документи різних структур у колекціях. Це дозволяє зручно зберігати та маніпулювати даними, які можуть мати змінну структуру.

MongoDB також має горизонтально масштабовану архітектуру, що дозволяє легко розширювати обсяги даних та обробку завантаження. Вона підтримує реплікацію (розмноження даних на кількох серверах) та шардування (розподіл даних між різними серверами). Це дозволяє підтримувати високу доступність, забезпечувати відмовостійкість та підтримувати високу продуктивність навіть при великому обсязі даних [2].

ExpressJS - мінімалістичний та гнучкий фреймворк для створення веб-додатків Node.js, який забезпечує надійний набір функцій.

Однією з основних особливостей Express.js є його простота використання та мінімалістичний підхід. Він надає лише базові функції для обробки запитів та маршрутизації, дозволяючи розробникам вільно вибирати та використовувати додаткові пакети для розширення функціоналу за потребою. Це робить Express.js дуже гнучким та розширюваним фреймворком.

Express.js також підтримує обробку HTTP-запитів з різними методами, такими як GET, POST, PUT, DELETE та інші. Він дозволяє визначати маршрути (routes), які відповідають на певні URL-адреси та виконують необхідні дії. Це дозволяє створювати потужні API для взаємодії з клієнтськими додатками або іншими серверами.

За допомогою Express.js можна також використовувати середні програми (middleware), які обробляють запити до сервера перед їх передачею до відповідного маршруту. Це дає можливість реалізувати різні функції, такі як аутентифікація, обробка помилок, журналювання, тощо [3].

Дві найпопулярніші IDE для розробки на JavaScript – це Visual Studio Code і Atom.

Visual Studio Code (VS Code) — це кросплатформна IDE із відкритим кодом, розроблена Microsoft. Він підтримує широкий спектр мов, включаючи JavaScript, і має велику кількість розширень для подальшого вдосконалення його функціональності. Він чудово підтримує розробку Node.js із такими функціями, як IntelliSense, автоматичний рефакторинг і налагодження.

Atom – розроблений GitHub, є ще однією кросплатформною IDE з відкритим кодом. Він підтримує кілька мов програмування, включаючи JavaScript. Atom можна налаштувати та має великий реєстр пакетів. Однак він може працювати не так добре, як VS Code для великих проектів через проблеми з продуктивністю.

Враховуючи ці параметри, VS Code є оптимальним вибором завдяки своїм потужним функціям, розширюваності та чудовій продуктивності.

Існує декілька менеджерів пакетів JavaScript, двома найпопулярнішими з них є npm і yarn.

NPM (Node Package Manager) є менеджером пакетів за замовчуванням для Node.js. Він надає широкий реєстр бібліотек і пакетів, якими можна легко керувати та встановлювати їх у проект.

Yarn — це менеджер пакетів, розроблений Facebook як більш ефективна та надійна альтернатива npm. Він пропонує швидше встановлення пакетів і краще керування залежностями. Однак, починаючи з версії 5 npm, багато переваг Yarn були включені в npm, що робить різницю між ними несуттєвою.

У випадку даного ПЗ npm обрано через його широке застосування, сумісність з Node.js і покращені функції в останніх версіях.

Окрім цього, будуть використовують додаткові утиліти та бібліотеки, такі як:

а) bcrypt — адаптивна криптографічна хеш-функція формування ключа, що використовується для захищеного зберігання паролів [7];

б) mongoose - ODM-бібліотека, що створює з'єданння між MongoDB і JavaScrypt NodeJs [9];

в) бібліотека Axios необхідна для створення асинхронних запитів [6];

г) nodemon – утиліта, що відстежує будь-які зміни в коді та автоматично перезапускає з ними сервер [10];

д) бібліотека Redux управляє станом застосунку [8].

1.3.3 Аналіз відомих програмних продуктів

На даний момент в Україні існує дуже мала кількість веб-застосунків, які б задовільняли потребам користувачів у данній предметній області.

Одним з таких є Прищепкін Event-Team, що займається організацію різних івентів, у тому числі й спортивних. Їх сайт виконаний у сучасному, красивому дизайні з плавними анімаціями. На сайті можна обрати послуги, які необхідні для проведення ваших спортивних змагань, такі як: замовити спортивну форму, інвентар, ведучого, суддю, нагороди.

Головною проблемою цього сервісу є те, що він платний і спрямований на організацію великих заходів. Тобто вони лише займаються тим, що підготовлюють місце, та весь необхідний інвентар для проведення змагань. Бажаючому провести івент, вже необхідно мати людей з якими він хоче його провести. А нині важливою постає проблема в тому, щоб знайти собі підходящих людей, з якими ти можеш провести командну аматорську гру і у зручний для всіх час. Тобто щоб не було проблеми зібрати всіх знайомих разом, а просто обрати івент, який тобі підходить.

На рисунку 1.3 та 1.4 зображено сайт Прищепкін Event-Team. Посилання: https://prischepkin.com/ua/services/sportyvni-zahody-ta-chempionaty

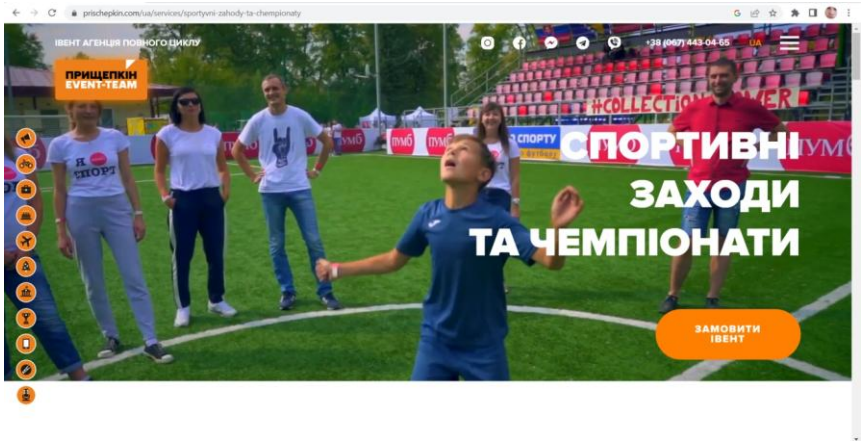


Рисунок 1.3 – Головна сторінка prischepkin.com для організації івентів

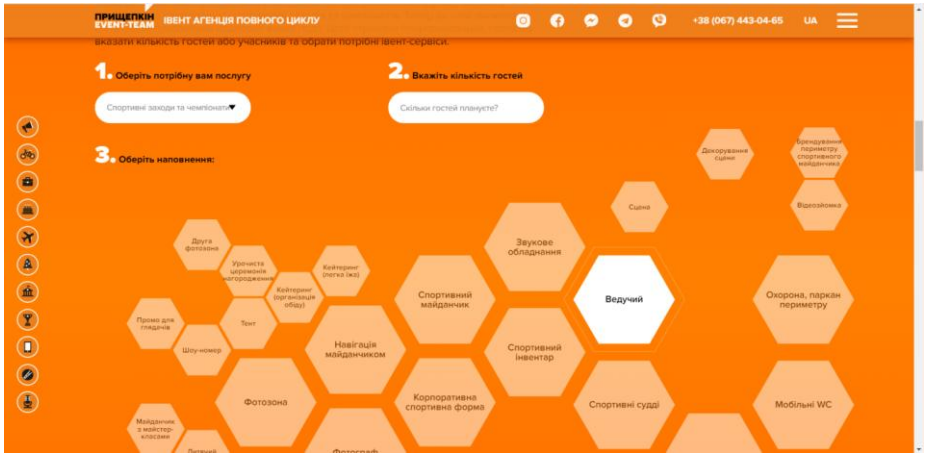


Рисунок 1.4 – Функціонал на сайті prischepkin.com

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Comand Match Finder	Прищепкін Event-Team	Пояснення
Сучасний дизайн	+	+	В обох сервісів сучасний дизайн
Зручний інтрефейс	+	+	В обох сервісів зручний інтерфейс
Мінімалізм	+	-	У Прищепкін Event-Team занадто

Продовження таблиці 1.1

			перенавантажений дизайн
Авторизація	+	-	У Прищепкін Event-Team відсутня авторизація
Зв’язок з організатором	+	+	В обох сервісах можна зв’язатись з організатором
Створення власного івенту	+	+	В обох сервісах можна організувати івент
Оцінка користувачів	+	-	У Прищепкін Event-Team не можна оцінювати користувачів
Пошук спортивних івентів за необхіними критеріями	+	-	У Прищепкін Event-Team не можна знайти вже існуючий івент за необхіними критеріями

Тож виходячи з цього, можна зробити висновок, що обрана проблема є актуальною і потребує вирішення.

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є створення форми для реєстрації на матчі з командних видів спорту, більше функцій можна побачити на рисунку 1.5.

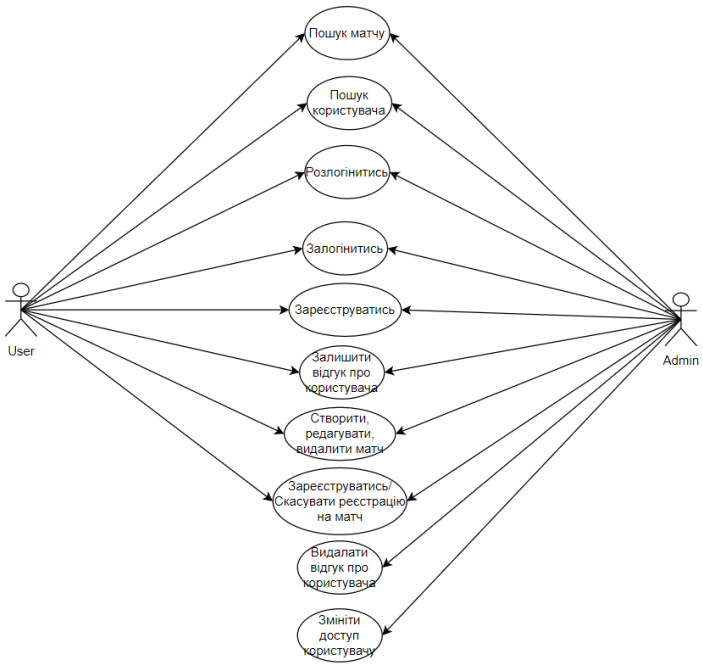


Рисунок 1.5 – Діаграма варіантів використання

В таблицях 1.2 - 1.13 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача
Actors	Незареєстрований користувач
Trigger	Бажання користувача зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім’я, логін, пошта користувача, пароль в системі. Після заповнення даних,

Продовження таблиці 1.2

	користувач натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних, поля в які введені некоректні дані видають помилку, і необхіно ввести дані у правильному форматі.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу

Таблиця 1.3 - Варіант використання UC-2

Use case name	Авторизація користувача
Use case ID	UC-02
Goals	Авторизація існуючого користувача
Actors	Зареєстрований користувач
Trigger	Бажання користувача авторизуватись
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку авторизації. В поля для авторизації вводяться відповідні дані: пошта користувача, пароль в системі. Після заповнення даних користувача натискає кнопку авторизація. Після цього з’являється повідомлення про успішну авторизацію, і користувач перенаправляється на головну сторінку.
Extension	В випадку введення некоректних даних, поля в які введені некоректні дані видають помилку, і необхіно ввести дані у правильному форматі.
Post-Condition	Авторизація користувача, перехід на головку входу

Таблиця 1.4 - Варіант використання UC-3

Use case name	Вихід з системи
Use case ID	UC-03
Goals	Вийти з системи
Actors	Зареєстрований користувач
Trigger	Бажання вийти з системи
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на головну сторінку, натискає на кнопку виходу і після цього перенаправляється на сторінку входу
Extension	-
Post-Condition	Вихід з системи, перехід на сторінку входу

Таблиця 1.5 - Варіант використання UC-4

Use case name	Створення нового матчу
Use case ID	UC-04
Goals	Створити новий матч
Actors	Зареєстрований користувач
Trigger	Бажання створити матч
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на сторінку створення матчу, натискає на кнопку створення нового матчу. В поля для створення матчу вводяться відповідні дані: вид командного спорту, місце проведення, дата проведення, час проведення, кількість людей для реєстрації на матч, контактні дані, опис. Підтверджує створення, і перенаправляється на сторінку з матчами.
Extension	В випадку введення не коректних даних, відповідні поля

Продовження таблиці 1.6

	видають помилку.
Post-Condition	Створення матчу, перехід на сторінку з матчами

Таблиця 1.6 - Варіант використання UC-5

Use case name	Редагування матчу
Use case ID	UC-05
Goals	Редагувати матч
Actors	Людина, що створила матч, або адміністратор
Trigger	Бажання змінити інформацію про матч
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на сторінку зі створеними матчами, натискає на кнопку редагування матчу. В потрібні поля вводить нові дані про матч, підтверджує редагування, і перенаправляється на сторінку з матчами.
Extension	В випадку введення не коректних даних, відповідні поля видають помилку.
Post-Condition	Редагування матчу, перехід на сторінку з матчами

Таблиця 1.7 - Варіант використання UC-6

Use case name	Видалення матчу
Use case ID	UC-06
Goals	Видалити матч
Actors	Людина, що створила матч, або адміністратор
Trigger	Бажання видалити матч
Pre-conditions	Увійти в систему

Продовження таблиці 1.7

Flow of Events	Користувач заходить у систему, переходить на сторінку зі створеними матчами, натискає на кнопку видалити матч.
Extension	-
Post-Condition	Видалення матчу.

Таблиця 1.8 - Варіант використання UC-7

Use case name	Реєстрація на матч
Use case ID	UC-07
Goals	Зареєструватися на матч
Actors	Зареєстрований користувач
Trigger	Бажання зареєструватися на матч
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на сторінку зі створеними матчами, переходить на сторінку матчу, на який він бажає зареєструватися, натискає на кнопку зареєструватися на матч, і перенаправляється на сторінку з матчами, на які він зареєстрований.
Extension	-
Post-Condition	Реєстрація на матч, перехід на сторінку з матчами, на які він зареєстрований.

Таблиця 1.9 - Варіант використання UC-8

Use case name	Скасування реєстрації на матч
Use case ID	UC-08
Goals	Скасувати реєстрацію на матч
Actors	Зареєстрований користувач
Trigger	Бажання скасувати реєстрацію на матч

Продовження таблиці 1.9

Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на сторінку з матчами, на які він зареєстрований, переходить на сторінку матчу, на якому він хоче скасувати реєстрацію, натискає на кнопку скасувати реєстрацію на матч, і перенаправляється на сторінку з матчами.
Extension	-
Post-Condition	Скасування реєстрації на матч, перехід на сторінку з матчами

Таблиця 1.10 - Варіант використання UC-9

Use case name	Оцінка користувача
Use case ID	UC-09
Goals	Оцінити користувача
Actors	Зареєстрований користувач
Trigger	Бажання оцінити користувача
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на сторінку з усіма користувачами, оцінює бажаного користувача.
Extension	-
Post-Condition	Оцінка користувача

Таблиця 1.11 - Варіант використання UC-10

Use case name	Редагування статусу користувача
Use case ID	UC-10
Goals	Додавання нового адміністратора, або блокування користувача

Продовження таблиці 1.11

Actors	Адміністратор
Trigger	Потреба в додаванні нового адміністратора або блокування користувача
Pre-conditions	Увійти в систему як адміністратор
Flow of Events	Адміністратор заходить у систему, переходить на сторінку з усіма користувачами, та або надає права адміністратора, або блокує користувача.
Extension	-
Post-Condition	Редагування статусу користувача

Таблиця 1.12 - Варіант використання UC-11

Use case name	Пошук матчу
Use case ID	UC-11
Goals	Знайти матч за бажаними критеріями
Actors	Зареєстрований користувач
Trigger	Бажання знайти потрібний матч
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на головну сторінку, у полі пошуку вказує потрібні йому критерії і отримує результат пошуку.
Extension	-
Post-Condition	Отримання результату пошуку по сайту

Таблиця 1.13 - Варіант використання UC-12

Use case name	Пошук користувача
Use case ID	UC-12

Продовження таблиці 1.13

Goals	Знайти необхідного користувача
Actors	Зареєстрований користувач
Trigger	Бажання знайти необхідного користувача
Pre-conditions	Увійти в систему
Flow of Events	Користувач заходить у систему, переходить на сторінку з користувачами та шукає користувача за логіном.
Extension	-
Post-Condition	Отримання списку користувачів за введеним логіном

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.6 наведено загальну модель вимог, а в таблицях 1.14 – 1.28 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.5.

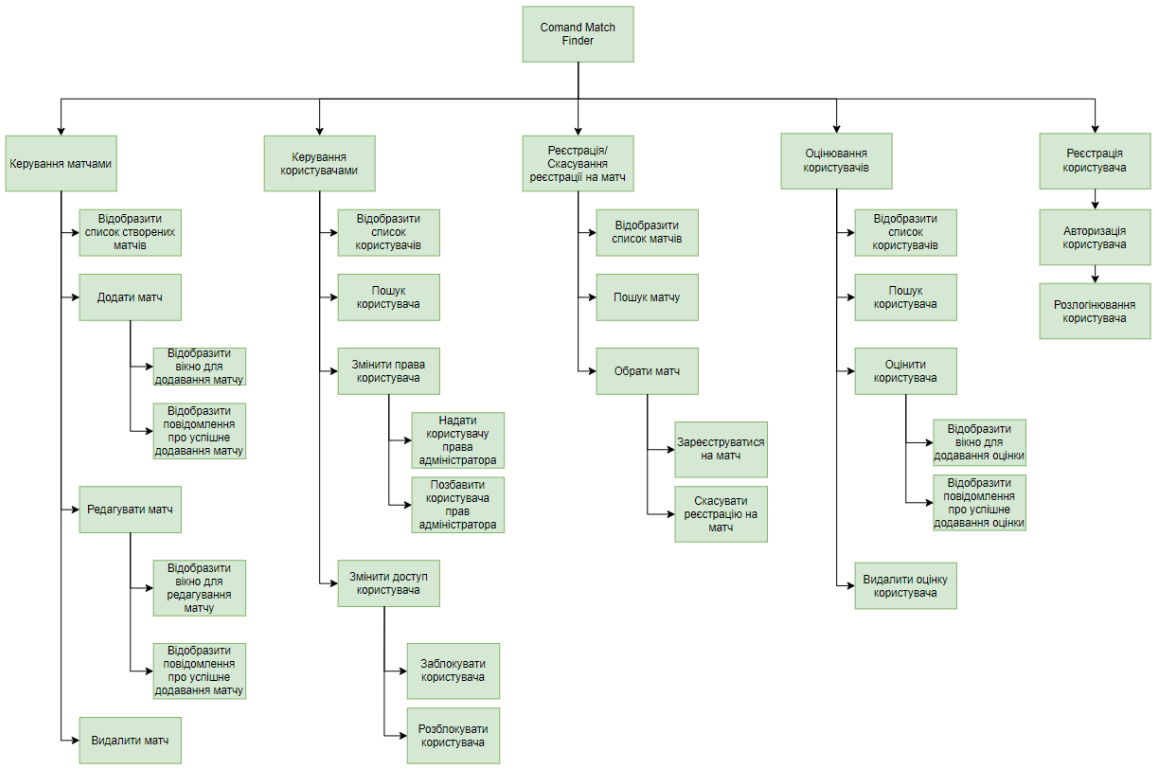


Рисунок 1.6 – Модель вимог у загальному вигляді

Таблиця 1.14 – Функціональна вимога FR-1

Назва	Реєстрація користувача
Опис	Система повинна надавати можливість реєстрації для користувачів шляхом введення імені, логіну, пошти, паролю.

Таблиця 1.15 – Функціональна вимога FR-2

Назва	Авторизація користувача
Опис	Система повинна надавати можливість авторизації для користувачів шляхом введення пошти, паролю

Таблиця 1.16 – Функціональна вимога FR-3

Назва	Вихід з профілю
Опис	Система повинна надавати користувачам можливість вийти зі свого профілю.

Таблиця 1.17 – Функціональна вимога FR-4

Назва	Пошук по сайту
Опис	Система повинна надавати можливість пошуку матчів, користувачів по сайту.

Таблиця 1.18 – Функціональна вимога FR-5

Назва	Створення нового матчу
Опис	Система повинна надавати користувачам можливість створення нових матчів.

Таблиця 1.19 – Функціональна вимога FR-6

Назва	Редагування матчу
Опис	Система повинна надавати користувачам можливість редагувати свої створені матчі. Для адміністратора редагування усіх створених

Продовження таблиці 1.19

	матчів.

Таблиця 1.20 – Функціональна вимога FR-7

Назва	Видалення матчу
Опис	Система повинна надавати користувачам можливість видаляти свої створені матчі. Для адміністратора видалення усіх створених матчів.

Таблиця 1.21– Функціональна вимога FR-8

Назва	Реєстрація, скасування реєстрації на матч
Опис	Система повинна надавати користувачам можливість зареєструватися на матч або скасувати реєстрацію на нього.

Таблиця 1.22 – Функціональна вимога FR-9

Назва	Перегляд матчів
Опис	Система повинна надавати користувачам можливість передивлятись існуючі матчі.

Таблиця 1.23 – Функціональна вимога FR-10

Назва	Перегляд користувачів
Опис	Система повинна надавати користувачам можливість передивлятись список користувачів.

Таблиця 1.24 – Функціональна вимога FR-11

Назва	Перегляд створених матчів
Опис	Система повинна надавати користувачам можливість передивлятися свої створенні матчі. Для адміністратора перегляд усіх створених матчів

Таблиця 1.25 – Функціональна вимога FR-12

Назва	Перегляд матчів на які зареєстрований
Опис	Система повинна надавати користувачам можливість передивлятися матчі на які вони зареєстровані.

Таблиця 1.26 – Функціональна вимога FR-13

Назва	Оцінка користувачів
Опис	Система повинна надавати користувачам можливість оцінювати інших користувачів.

Таблиця 1.27 – Функціональна вимога FR-14

Назва	Редагування статусу користувача
Опис	Система повинна давати адміністратору можливість надавати/позбавляти прав адміністратора користувачів або блокувати/розблоковувати їх.

Таблиця 1.28 – Функціональна вимога FR-15

Назва	Видалення оцінки користувача
Опис	Система повинна надавати адміністратору можливість видаляти оцінки користувачів.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15
UC-1	+														
UC-2		+													
UC-3			+												
UC-4					+										
UC-5						+									
UC-6							+								
UC-7								+							
UC-8								+							
UC-9													+		+
UC-10														+	
UC-11									+		+	+			
UC-12										+					

Рисунок 1.7 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Нефункціональні вимоги визначають властивості системи, що непов'язані з її прямою функціональністю, але впливають на її якість, ефективність, надійність та інші аспекти. Вони описують характеристики системи, які не стосуються конкретних функцій, але важливі для забезпечення задоволення потреб користувачів та виконання бізнес-вимог.

Нефункціональні вимоги доповнюють функціональні вимоги та допомагають забезпечити створення системи, яка відповідає потребам користувачів і вимогам бізнесу з точки зору якості та продуктивності.

Нефункіональні вимоги, які планується імплементувати:

- мінімалістичний і зручний інтерфейс для користувача;
- конфідеційність і захист даних користувачів;
- швидкодія виконання запитів;
- повідомлення про успішні, чи не успішні дії.

1.5 Постановка задачі

Метою є підтримка діяльності організаторів командних аматорських ігор.

Головна ціль – це створення веб-застосунку з простим і зручним інтерфейсом для користувачів.

Задачі, що необхідно вирішити для цього:

- розробити зручний веб-інтерфейс для користувачів;
- створити базу даних, що зберігатиме всю необхідну інформацію для організації командних аматорських ігор, та інформацію про користувачів;
- розробити API для взаємодії клієнта з сервером;
- розробити функціонал створення спортивного івенту;
- розробити функціонал оцінювання користувачів;

Висновки до розділу

В цьому розділі були викладені загальні положення обраної предметної області. Було пояснено актуальність проблематики та важливість даної області, що підтверджує, що вона потребує актуальних реалізацій в ІТ сфері. Також було викладено конкретне завдання, яке планується реалізувати в рамках предметної області. Окрім цього, було обрано мову програмування, необхідні фреймворки, бібліотеки й утиліти. Розглянуто існуючі реалізації та порівняння їх з власним ПЗ. Визначили функціональні і нефункціональні вимоги.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Так як користувач бачить лише клієнтську частину веб-застосунку і взаємодіє тільки з нею, то буде використовуватись поняття веб-застосунку для відображення взаємодії користувача з програмним забезпеченням.

Для зображення бізнес процесів програмного забезпечення використовуватиметься BPMN модель (рисунок 2.1).

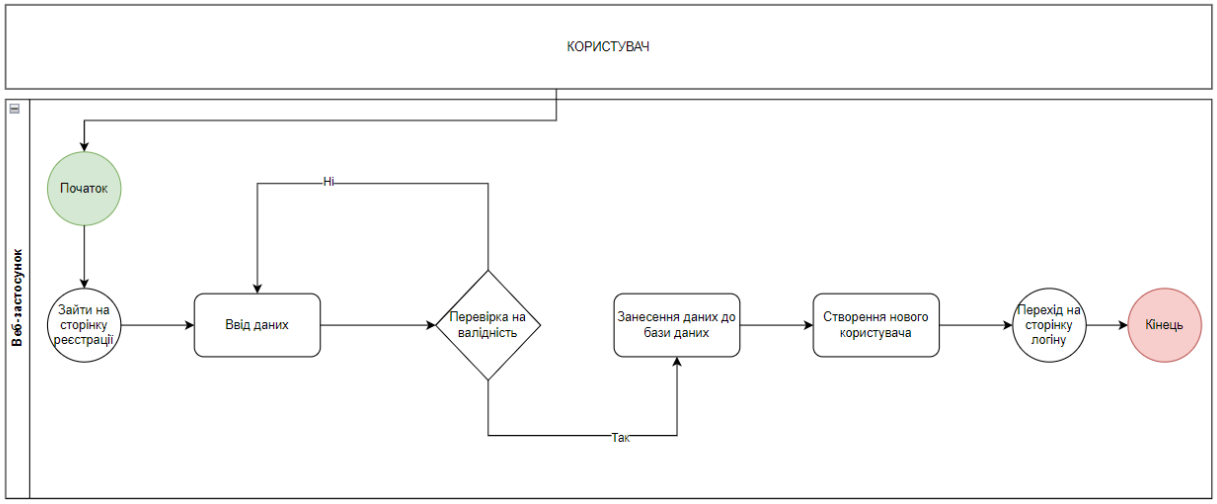


Рисунок 2.1 – Схема бізнес процесу реєстрації користувача

Опис послідовності реєстрації користувача:

– користувач переходить на сторінку реєстрації;
– користувач заповнює необхідні поля на сторінці реєстрації.

Якщо введені поля не проходять валідацію, створити користувача неможливо;

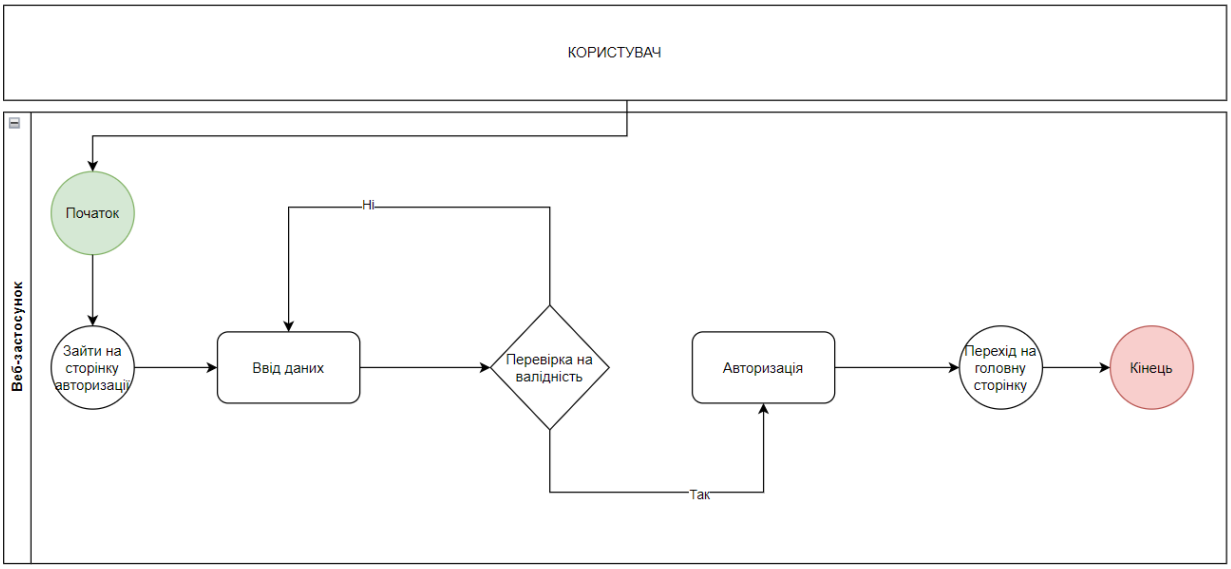


Рисунок 2.2 – Схема бізнес процесу авторизації користувача

Опис послідовності авторизації користувача:

– користувач переходить на сторінку авторизації;

– користувач заповнює необхідні поля на сторінці авторизації;

– якщо введені поля проходять валідацію, користувача авторизується і перенаправляється на головну сторінку.

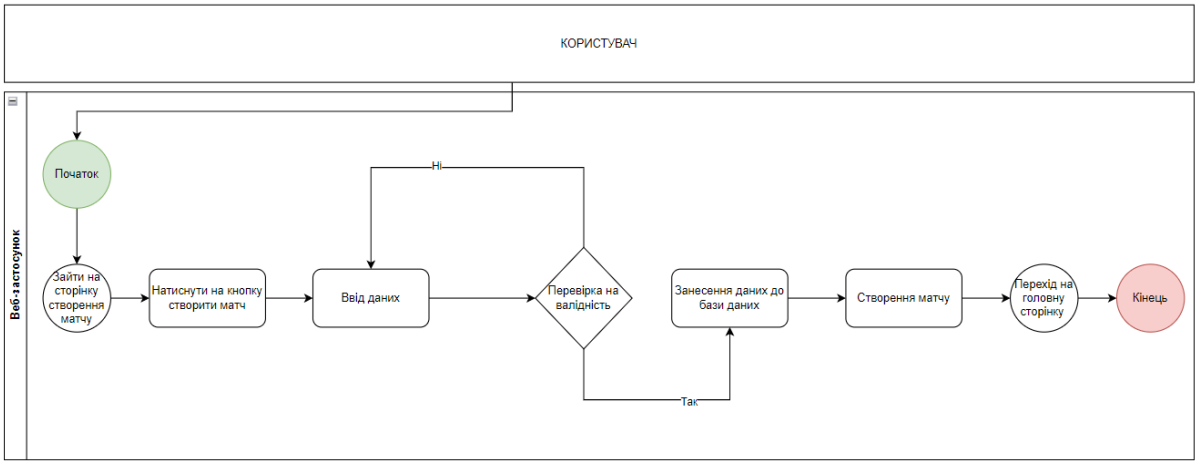


Рисунок 2.3 – Схема бізнес процесу створення матчу

Опис послідовності створення матчу з командного виду спорту авторизованим користувачем:

– користувач переходить на сторінку створення матчу;

– користувач натискає на кнопку створити матч;

– користувач заповнює необхідні поля у панелі створення матчу;

– якщо введені поля проходять валідацію, матч створюється і публікується на головній сторінці.

2.2 Архітектура програмного забезпечення

Задля реалізація ПЗ було прийнято рішення побудувати систему за клієнт-серверною архітектурою.

Розглянемо файлову структуру проекту, щоб більш детально зрозуміти його архітектуру. Уся клієнтська частина розташована в папці client, а серверна частина розташована в папці server. Структуру файлів зображено на рисунку 2.4.

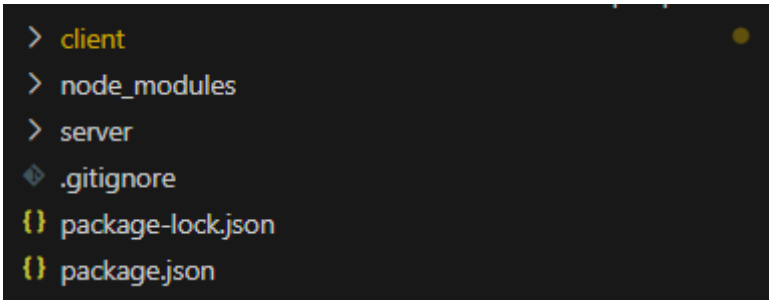


Рисунок 2.4 – Файлова структура програмного забезпечення

Клієнтська частина веб-застосунку розроблена за допомогою React.js. React.js - це JavaScript-бібліотека для побудови інтерфейсів користувача. Вона дозволяє розробляти складні інтерфейси з простих компонентів, що повторно використовуються. Усі залежності клієнтської частини розташовані у підпапці node_modules, що додані за допомогою Node.js. А весь код клієнтської частини розташований у папці src. Структуру клієнтської частини зображено на рисунку 2.5.

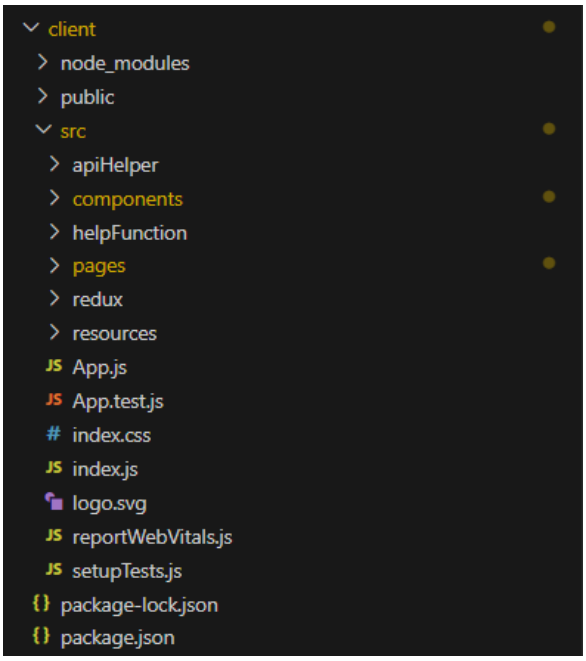


Рисунок 2.5 – Структура папки client

Клієнтська частина складається з кількох важливих папок:

– pages – ця папка містить файли для кожної сторінки веб-застосунку. Такі як: головна сторінка, сторінки адміністратора та сторінки звичайного користувача, окремі сторінки матчів та сторінки користувачів;

– resources – ця папка містить стильові файли, які визначають зовнішній вигляд сторінок;

– components – ця папка містить повторно використовувані компоненти, які використовуються на різних сторінках веб-застосунку. Такі як: форми для створення/редагування матчу, форми для залишення рейтингу, коментаря про користувача, дані зі сторінки профілю, а саме матчі, на які зареєстрований користувач, а також рейтинг користувача;

– helpFunction – ця папка містить допоміжні функції, які використовуються у веб-застосунку. А саме форматування дати та часу;

– apiHelper – ця папка містить функції, які виконують деякі запити до сервера API. Вона включає функції для надсилання даних на сервер, отримання даних із сервера, оновлення даних на сервері, а також видалення даних із сервера.

Серверна частина веб-програми розроблена на Node.js з використанням Express.js. Node.js – це середовище виконання JavaScript на серверній стороні. Express.js – це мінімалістичний веб-фреймворк для Node.js, який надає набір функцій для веб-застосунків. Структуру серверної частини зображено на рисунку 2.6.

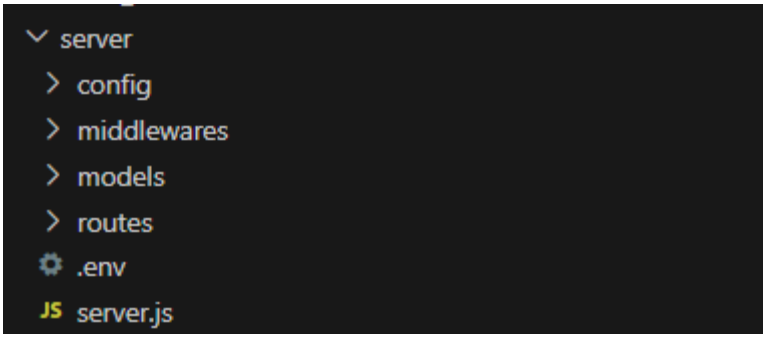


Рисунок 2.6 – Структура серверної частини

Серверна частина складається з наступних папок:

– config – тут розташовані налаштування серверної частини. У даному випадку тут знаходиться код для підключення до бази даних MongoDB за допомогою бібліотеки Mongoose;

– middlewares – ця папка є стандартною для MERN архітектури. Тут знаходяться функції, що обробляють запити перед отриманням і передачею до маршрутів. У випадку нашого ПЗ, тут знаходиться файл з кодом, що перевіряє автентифікацію користувача, за допомогою JWT токену, який ми зберігаємо у локальному сховищі браузеру;

– models – ця папка призначена для опису моделі даних, що описують структуру даних та взаємодію з базою даних;

– routes – тут розмішені файли, що відповідають за обробку запитів і маршрутизацію;

– файл server.js – призначений для створення та налаштування серверу. Тут відбувається конфігурування сервера, обробка маршрутів та його ініціалізація. Окрім цього тут сервер запускається на певному порті.

Розглянемо ще декілька файлів.

Файл package.json містить конфігураційну інформацію Node.js. Тут знаходяться всі необхідні залежності і скрипти для керування проектом. В розділі "dependencies" вказуються зовнішні пакети та бібліотеки, які використовує проект. При виконанні команди npm install залежності, перераховані у файлі package.json, автоматично завантажуються та встановлюються з реєстру пакетів npm. Крім того, у файлі package.json можна визначити власні скрипти, які дозволяють запускати різні дії та команди для розробки, тестування, збірки проекту тощо.

Файл package.json можна редагувати вручну або використовувати команду npm init для створення нового файлу package.json та заповнення його початковими даними. Файл package.json є важливим елементом проекту Node.js і використовується для управління залежностями, конфігурації та скриптами, що спрощує розробку та керування проектом.

Файл .env призначений для зберігання конфіденційних змінних, таких як підключення до бази даних.

Веб-застосунок використовує MongoDB як базу даних. MongoDB - це документоорієнтована база даних NoSQL, яка забезпечує високу продуктивність, високу доступність та легку масштабованість.

Модель бази даних складається з наступних сутностей:

– users – містить інформацію про зареєстрованих користувачів;

– mathes – містить інформацію про матчі;

– ratings – містить інформацію про рейтинг користувачів;

– registered – містить інформацію про те на які матчі які користувачі зареєстровані.

2.3 Конструювання програмного забезпечення

В якості бази даних використовується СУБД MongoDB. База даних призначена для зберігання даних про зареєстрованих користувачів, зберігання інформації про створенні матчі, рейтинги користувачів і матчі на

які зареєстровані користувачі. Опис таблиць бази даних наведено у таблицях 2.1 - 2.4. Логічна модель бази даних наведена на рисунку 2.7.

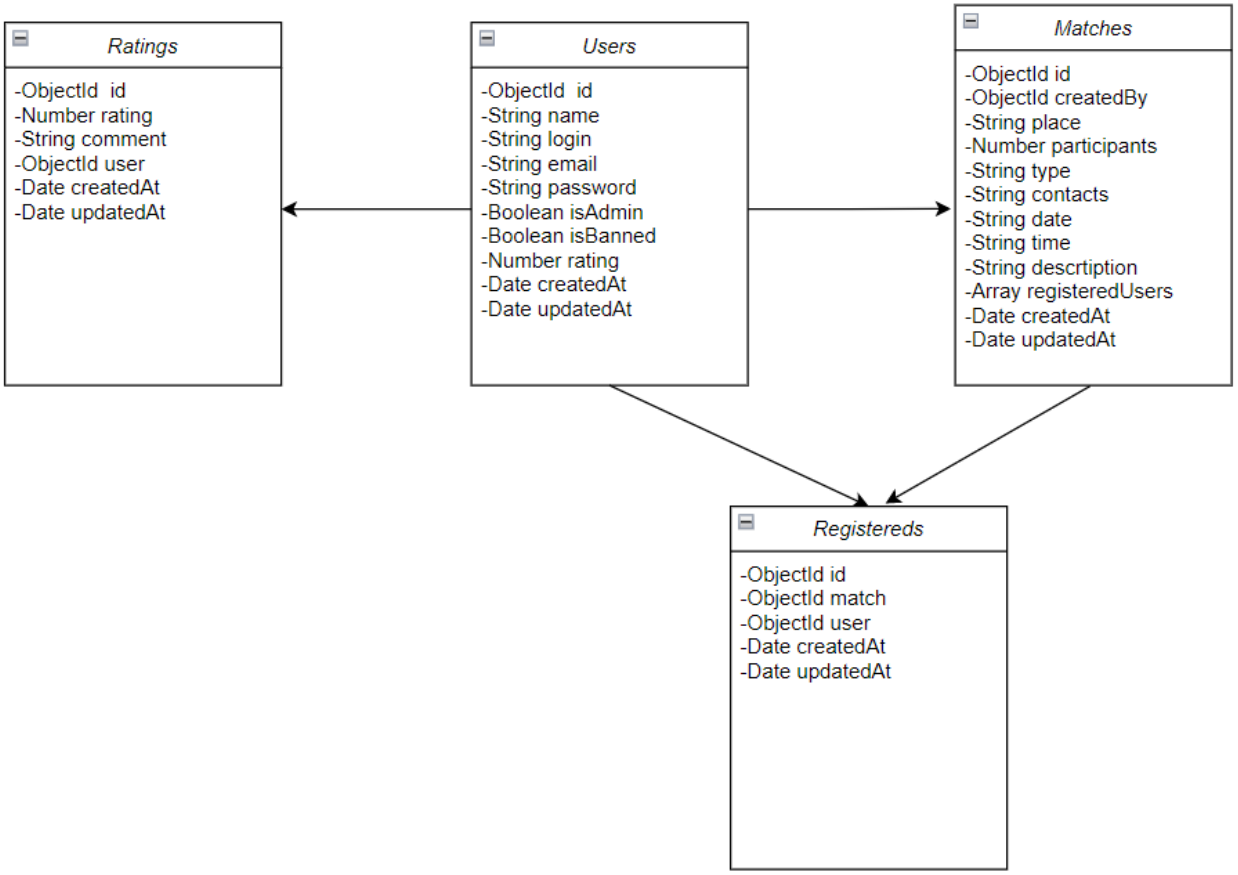


Рисунок 2.7 – Логічна модель бази даних

Таблиця 2.1 – Опис таблиці users

Таблиця	Назва поля	Тип даних	Опис
users	id	objectId	ідентифікаційний номер користувача
	name	string	ім’я користувача
	login	string	логін користувача
	email	string	електронна пошта користувача

Продовження таблиці 2.1

	password	string	пароль користувача
	isAdmin	boolean	перевірка чи є користувач адміністратором
	isBanned	boolean	перевірка чи користувача заблоковано
	rating	number	рейтинг користувача
	createdAt	date	дата створення таблички
	updatedAt	date	дата оновлення таблички

Таблиця 2.2 – Опис таблиці matches

Таблиця	Назва поля	Тип даних	Опис
matches	id	objectId	ідентифікаційний номер матчу
	createdBy	objectId	посилання на запис у табличці users, де зберігаються дані користувача, що створив матч
	registeredUsers	array	посилання на записи у табличці users, де зберігаються дані користувачів, що

Продовження таблиці 2.2

			зареєструвались на матч
	place	string	місце проведення матчу
	participants	number	кількість гравців у матчі
	type	string	назва командного виду спорту
	contacts	string	контакти користувача, що організовую матч
	date	string	дата проведення матчу
	time	string	час проведення матчу
	description	string	опис матчу
	createdAt	date	дата створення таблички
	updatedAt	date	дата оновлення таблички

Таблиця 2.3 – Опис таблиці ratings

Таблиця	Назва поля	Тип даних	Опис
ratings	id	objectId	ідентифікаційний номер рейтнигу
	rating	number	рейтинг користувача
	comment	string	коментарій до рейтингу
	user	objectId	посилання на запис у табличці users, де

Продовження таблиці 2.3

	user	objectId	зберігаються дані користувача, якому поставили рейтинг
	createdAt	date	дата створення таблички
	updatedAt	date	дата оновлення таблички

Таблиця 2.4 – Опис таблиці registereds

Таблиця	Назва поля	Тип даних	Опис
registereds	id	objectId	ідентифікаційний номер реєстрації на матч
	rating	number	рейтинг користувача
	match	objectId	посилання на запис у табличці matches, де зберігаються дані про матч, на який зареєструвався користувач
	user	objectId	посилання на запис у табличці users, де зберігаються дані користувача, який зареєструвався на матч
	createdAt	date	дата створення таблички

Продовження таблиці 1.4

	updatedAt	date	дата оновлення таблички

Увесь програмний код був написаний у інтегрованому середовищі розробки(IDE) Visual Studio code. Це дуже зручне середовище для написання JavaScript проектів. Інтерфейс Visual Studio code зображено на рисунку 2.8.

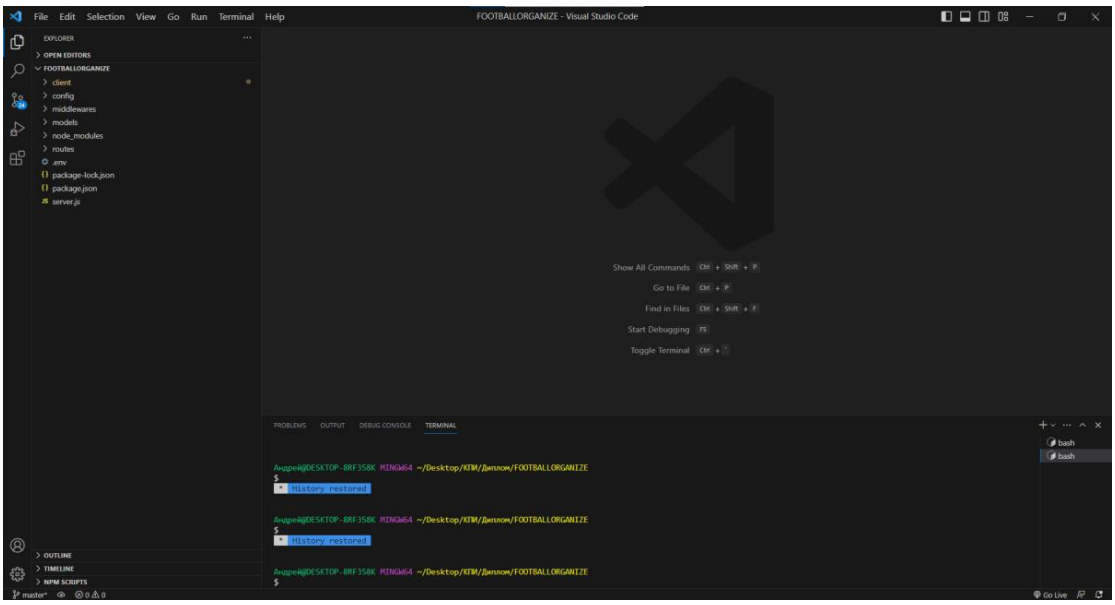


Рисунок 2.8 – Середовище для написання програмного коду Visual Studio code

Опис утиліт, бібліотек та пакетів, що використовуються у розробці програмного забезпечення наведено в таблиці 2.5.

Таблиця 2.5 – Опис утиліт

№ п/п	Назва утиліти, бібліотеки, пакета	Опис застосування

Продовження таблиці 2.5

1	Visual Studio code	Головне середовище розробки програмного забезпечення для клієнтської та серверної частини дипломної роботи.
2	MongoDB Compass	Програмне забезпечення для бази даних MongoDb, яке надає зручний інтерфейс для доступу до бази даних.
3	Express	Мінімалістичний та гнучкий фреймворк для створення веб-додатків Node.js, який забезпечує надійний набір функцій.
4	antd	Бібліотека для розробки інтерфейсу користувача на основі React.
5	axios	HTTP-клієнт на основі промісів. Необхідний для створення асинхронних запитів.
6	React	Бібліотека JavaScript для створення користувацьких інтерфейсів.
7	@reduxjs/toolkit	Призначений для створення Redux-логіки. Управляє станом застосунку
8	BcryptJS	Адаптивна криптографічна хеш-функція формування ключа, що використовується для захищеного зберігання паролів.

Продовження таблиці 2.5

9	Nodemon	Утиліта, що відстежує будь-які зміни в коді та автоматично перезапускає з ними сервер.
10	Mongoose	ODM-бібліотека, що створює з'єданння між MongoDB і JavaScrypt NodeJs.
11	jsonwebtoken	Бібліотека для створення та верифікації JWT в Node.js.
12	dotenv	Бібліотека для Node.js, яка дозволяє завантажувати змінні середовища з файлу .env в проекті.

Далі будуть наведені системні вимоги для використання веб-застосунку для підтримки діяльності організаторів аматорських командних ігор.

Мінімальна конфігурація технічних засобів:

Операційна система: Windows 7 або вище, MacOS 10.12 Sierra або вище, Linux (будь-який сучасний дистрибутив, що підтримує браузери останньої версії);

Процесор: 1 GHz або швидше;

Оперативна пам'ять (RAM): щонайменше 1 GB;

Жорсткий диск: щонайменше 1 GB вільного місця для зберігання кешу браузера та інших тимчасових файлів;

Відеокарта: будь-яка сумісна з операційною системою;

Дисплей: рекомендована роздільна здатність 1024x768 або вище;

Інтернет-з'єднання: широкосмугове;

Рекомендована конфігурація технічних засобів:

Операційна система: Windows 10;

Процесор: Intel Core i5;

Оперативна пам'ять (RAM): 16 ГБ;

Жорсткий диск: щонайменше 2 GB вільного місця для зберігання кешу браузера та інших тимчасових файлів;

Відеокарта: NVIDIA GeForce GTX 1050 Ti 4GB;

Дисплей: роздільна здатність 1920x1080;

Інтернет-з'єднання: широкосмугове, зі стабільною швидкістю для комфортної взаємодії з веб-застосунком;

Програмні вимоги:

Веб-браузер: Google Chrome, Mozilla Firefox, Safari або Microsoft Edge останньої версії;

2.4 Аналіз безпеки даних

Всі веб-застосунки, що зберігаються дані користувача стикаються з проблемою безпеченого збереження цих даних. Адже якщо важливі дані зберігати просто в текстовому вигляді, то зловмисники можуть легко отримати до доступ до них. Це більше всього стосується паролів, бо люди часто використовують один й той же пароль на декількох сервісах. Тож для того щоб запобігти втраті паролю користувачів, дані про пароль необхідно шифрувати.

Для цього використовується бібліотека Bcrypt, що хеширує пароль додаючи до нього “сіль”.

Хешування:

hashPassword = bcrypt.hash(password, 10);

Порівняння:

bcrypt.compare(password,hashPassword)

Окрім цього за допомогою JWT ідентифікуємо користувачів, щоб розуміти чи потрібні дані належать даному користувачу.

JWT це стандарт для створення токенів доступу, що використовує формат JSON.

Перевірка автентифікації за допомогою JWT (JSON Web Token) токену - це процес перевірки правильності та достовірності токену, щоб підтвердити ідентичність користувача. JWT є форматом токена, який використовується для безпечної передачі інформації між сторонами в зашифрованому вигляді.

Перевірка автентифікації за допомогою JWT токену дозволяє підтвердити ідентичність користувача без необхідності зберігання стану сесії на сервері.

Висновки до розділу

В цьому розділі були описані основні бізнес-процеси і промодельовані за допомогою засобів BPMN. Було розібрано архітектуру ПЗ, а саме що таке клієнт-серверна архітектура та подано її детальний опис. Також була зображена логічна модель бази даних, та описано кожну її табличку. Було описано бібліотеки, пакети та утиліти, що використовуються у процесі створення ПЗ. Окрім цього, був описаний спосіб безпечного зберігання даних користувачів, а саме збереження і хешування паролів користувачів за допомогою бібліотеки Bcrypt, а також спосіб перевірки автентифікації користувачів за допомогою JWT токену.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення може бути проведено за різними метриками. Розглянемо деякі з них:

Надійність – ця метрика вимірює стійкість програмного забезпечення до відмов. Вона може включати оцінку частоти відмов, часу відновлення після відмови, стійкості до помилок та здатності програми правильно обробляти непередбачувані ситуації. Також вона може містити інформацію про можливу кількість багів в програмі.

Продуктивність – ця метрика вимірює ефективність програмного забезпечення. Вона може включати оцінку швидкості виконання операцій, завантаження системи та використання ресурсів.

Підтримка та зручність використання – ця метрика оцінює, наскільки легко програмне забезпечення може бути встановлено, розміщено та використано користувачем. Вона може включати оцінку наявності документації, інтерфейсу користувача.

Безпека – ця метрика оцінює заходи, вжиті для захисту програмного забезпечення від несанкціонованого доступу. Вона може включати оцінку наявності механізмів автентифікації та авторизації, шифрування даних і захисту від вразливостей.

Аналіз якості програмного забезпечення було проведено за допомогою сервісу SonarCloud. Це сервіс для проведення статичного аналізу, шляхом підключення до репозиторію GitHub. На рисунку 3.1 зображено аналіз якості ПЗ за статичним аналізом.

Рисунок 3.1 – Аналіз якості ПЗ

3.2 Опис процесів тестування

Для тестування ПЗ було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.12.

Таблиця 3.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Ім’я, логін, електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректне ім’я, яке складається лише з українських або англійських літер, логін, який до цього не був зареєстрований у системи та який складається з англійських літер і спеціальних символів довжиною не більше 10 знаків, електрона пошта, яка до цього не була зареєстрована в системі, пароль від 5 до 15 символів. Після цього натискається кнопка підтвердження реєстрації.

Продовження таблиці 3.1

Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.

Таблиця 3.2 – Тест 1.2

Тест	Реєстрація користувача з некоректними вхідними даними
Модуль	Реєстрація користувача
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Ім’я, логін, електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: некоректне ім’я, яке складається з цифр або спеціальних символів, логін, який до цього був зареєстрований у системи та який складається з українських літер і спеціальних символів довжиною більше 10 знаків, електрона пошта, яка до цього була зареєстрована в системі, пароль до 5 або більше 15 символів. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	У реєстрації відмовлено, поля, у які введені некоректні дані, підсвічуються, кнопка реєстрації неактивна.
Фактичний результат	У реєстрації відмовлено, поля, у які введені некоректні дані, підсвічуються, кнопка реєстрації неактивна.

Таблиця 3.3 – Тест 1.3

Тест	Авторизація користувача
Модуль	Авторизація користувача

Продовження таблиці 3.3

Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні данні	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: електрона пошта, яка до цього була зареєстрована в системі, пароль від профілю довжиною від 5 до 15 символів. Після цього натискається кнопка підтвердження авторизації.
Очікуваний результат	Авторизація проходить успішно, користувач перенаправляється на головну сторінку.
Фактичний результат	Авторизація проходить успішно, користувач перенаправляється на головну сторінку.

Таблиця 3.4 – Тест 1.4

Тест	Авторизація користувача з некоректними вхідними даними
Модуль	Авторизація користувача
Номер тесту	1.4
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні данні	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: електрона пошта, яка до цього не була зареєстрована в системі, пароль не від профілю та довжиною до 5 або більше 15 символів. Після цього натискається кнопка підтвердження авторизації.
Очікуваний результат	У авторизації відмовлено, поля, у які введені некоректні дані, підсвічуються, кнопка авторизації неактивна.

Продовження таблиці 3.4

Фактичний результат	У авторизації відмовлено, поля, у які введені некоректні дані, підсвічуються, кнопка авторизації неактивна.

Таблиця 3.5 – Тест 1.5

Тест	Створення матчу
Модуль	Створення матчу
Номер тесту	1.5
Початковий стан системи	Користувач знаходиться на сторінці створення матчу
Вхідні данні	Місце проведення, вид спорту, кількість учасників, дата проведення, час проведення, контактні дані, опис матчу
Опис проведення тесту	У відповідні поля вводяться: місце проведення довжиною до 45 символів, вид спорту, який містить лише літери українського або англійського алфавіту довжиною до 10 символів, кількість учасників довжиною від 5 до 30, дата проведення не раніше ніж сьогоднішня, час проведення не менш ніж 3 години від часу сьогоднішнього дня, контактні дані довжиною до 30 символів, опис матчу довжиною до 300 символів. Після цього натискається кнопка підтвердження створення матчу.
Очікуваний результат	Матч створено успішно, матч додається у базу даних.
Фактичний результат	Матч створено успішно, матч додається у базу даних.

Таблиця 3.6 – Тест 1.6

Тест	Створення матчу з некоректними вхідними даними

Продовження таблиці 3.6

Модуль	Створення матчу
Номер тесту	1.6
Початковий стан системи	Користувач знаходиться на сторінці створення матчу
Вхідні данні	Місце проведення, вид спорту, кількість учасників, дата проведення, час проведення, контактні дані, опис матчу
Опис проведення тесту	У відповідні поля вводяться: місце проведення довжиною більше 45 символів, вид спорту, який містить цифри або спеціальні символи довжиною більше 10 символів, кількість учасників довжиною менше 5 або більше 30, дата проведення з минулого, час проведення менш ніж 3 години від часу сьогоднішнього дня, контактні дані довжиною більше 30 символів, опис матчу довжиною більше 300 символів. Після цього натискається кнопка підтвердження створення матчу.
Очікуваний результат	Матч не створено, поля, у які введені некоректні дані, підсвічуються, кнопка створення матчу неактивна.
Фактичний результат	Матч не створено, поля, у які введені некоректні дані, підсвічуються, кнопка створення матчу неактивна.

Таблиця 3.7 – Тест 1.7

Тест	Редагування матчу
Модуль	Редагування матчу
Номер тесту	1.7
Початковий стан системи	Користувач знаходиться на сторінці редагування матчу
Вхідні данні	Місце проведення, вид спорту, кількість учасників, дата проведення, час проведення, контактні дані, опис матчу

Продовження таблиці 3.7

Опис проведення тесту	У відповідні поля вводяться: місце проведення довжиною до 45 символів, вид спорту, який містить лише літери українського або англійського алфавіту довжиною до 10 символів, кількість учасників довжиною від 5 до 30, дата проведення не раніше ніж сьогоднішня, час проведення не менш ніж 3 години від часу сьогоднішнього дня, контактні дані довжиною до 30 символів, опис матчу довжиною до 300 символів. Після цього натискається кнопка підтвердження редагування матчу.
Очікуваний результат	Матч редаговано успішно, дані про матч оновлюються у базі даних.
Фактичний результат	Матч редаговано успішно, дані про матч оновлюються у базі даних.

Таблиця 3.8 – Тест 1.8

Тест	Редагування матчу з некоректними вхідними даними
Модуль	Редагування матчу
Номер тесту	1.8
Початковий стан системи	Користувач знаходиться на сторінці редагування матчу
Вхідні данні	Місце проведення, вид спорту, кількість учасників, дата проведення, час проведення, контактні дані, опис матчу
Опис проведення тесту	У відповідні поля вводяться: місце проведення довжиною більше 45 символів, вид спорту, який містить цифри або спеціальні символи довжиною більше 10 символів, кількість учасників довжиною менше 5 або більше 30, дата проведення з минулого, час проведення

Продовження таблиці 3.8

	менш ніж 3 години від часу сьогоднішнього дня, контактні дані довжиною більше 30 символів, опис матчу довжиною більше 300 символів. Після цього натискається кнопка підтвердження редагування матчу.
Очікуваний результат	Матч не редаговано, поля, у які введені некоректні дані, підсвічуються, кнопка редагування матчу неактивна.
Фактичний результат	Матч не редаговано, поля, у які введені некоректні дані, підсвічуються, кнопка редагування матчу неактивна.

Таблиця 3.9 – Тест 1.9

Тест	Реєстрація на матч
Модуль	Реєстрація на матч
Номер тесту	1.9
Початковий стан системи	Користувач знаходиться на сторінці матчу
Вхідні данні	Відсутні
Опис проведення тесту	На сторінці матчу натискається кнопка підтвердження реєстрації на матч.
Очікуваний результат	Користувача зареєстровано на матч, дані оновлюються у базі даних.
Фактичний результат	Користувача зареєстровано на матч, дані оновлюються у базі даних.

Таблиця 3.10 – Тест 1.10

Тест	Скасування реєстрації на матч
Модуль	Скасування реєстрації на матч
Номер тесту	1.10

Продовження таблиці 3.10

Початковий стан системи	Користувач знаходиться на сторінці матчу
Вхідні данні	Відсутні
Опис проведення тесту	На сторінці матч натискається кнопка підтвердження скасування реєстрації на матч.
Очікуваний результат	Скасовано реєстрацію користувача на матч, дані оновлюються у базі даних.
Фактичний результат	Скасовано реєстрацію користувача на матч, дані оновлюються у базі даних.

Таблиця 3.11 – Тест 1.11

Тест	Оцінювання користувача
Модуль	Оцінювання користувача
Номер тесту	1.11
Початковий стан системи	Користувач знаходиться на сторінці іншого користувача
Вхідні данні	Рейтинг, коментар
Опис проведення тесту	У відповідні поля вводяться: рейтинг від 1 до 5, коментар довжиною не більше 200 символів. Після цього натискається кнопка підтвердження додавання відгуку.
Очікуваний результат	Відгук про користувача додано, дані оновлюються у базі даних.
Фактичний результат	Відгук про користувача додано, дані оновлюються у базі даних.

Таблиця 3.12 – Тест 1.12

Тест	Оцінювання користувача з некоректними вхідними

Продовження таблиці 3.12

	даними
Модуль	Оцінювання користувача
Номер тесту	1.12
Початковий стан системи	Користувач знаходиться на сторінці іншого користувача
Вхідні данні	Рейтинг, коментар
Опис проведення тесту	У відповідні поля вводяться: рейтинг залишається пустим, коментар довжиною більше 200 символів. Після цього натискається кнопка підтвердження додавання відгуку.
Очікуваний результат	Відгук про користувача не додано, відображається повідомлення про помилку, кнопка підтвердження додавання відгуку неактивна.
Фактичний результат	Відгук про користувача не додано, відображається повідомлення про помилку, кнопка підтвердження додавання відгуку неактивна.

3.3 Опис контрольного прикладу

У якості контрольного прикладу буде наведено процеси від реєстрації користувача до створення матчу, та перегляду створеного матчу.

Перше, що користувач повинен зробити – зареєструватися. На рисунку 3.2 наведено реєстрацію користувача.

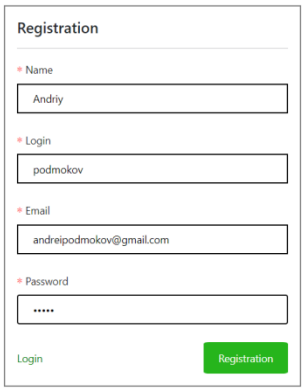


Рисунок 3.2 – Реєстрація нового користувача

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішну реєстрацію, і користувача буде перенаправлено на сторінку авторизації. На рисунку 3.3 зображено успішну реєстрацію користувача.

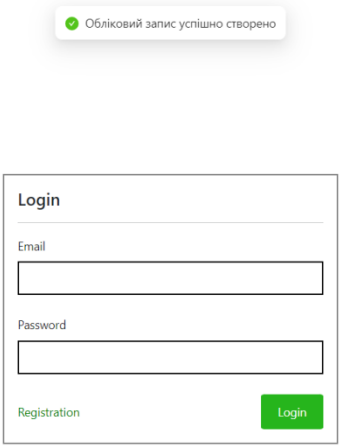


Рисунок 3.3 – Успішна реєстрація нового користувача

Далі необхідно авторизуватись, для цього у відповідні поля вводяться дані, які були використані при реєстрації. На рисунку 3.4 зображено авторизацію користувача.

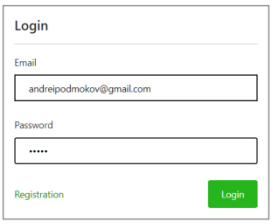


Рисунок 3.4 – Авторизація користувача

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішну авторизацію, і користувача буде перенаправлено на головну сторінку. На рисунку 3.5 зображено успішну авторизацію користувача.

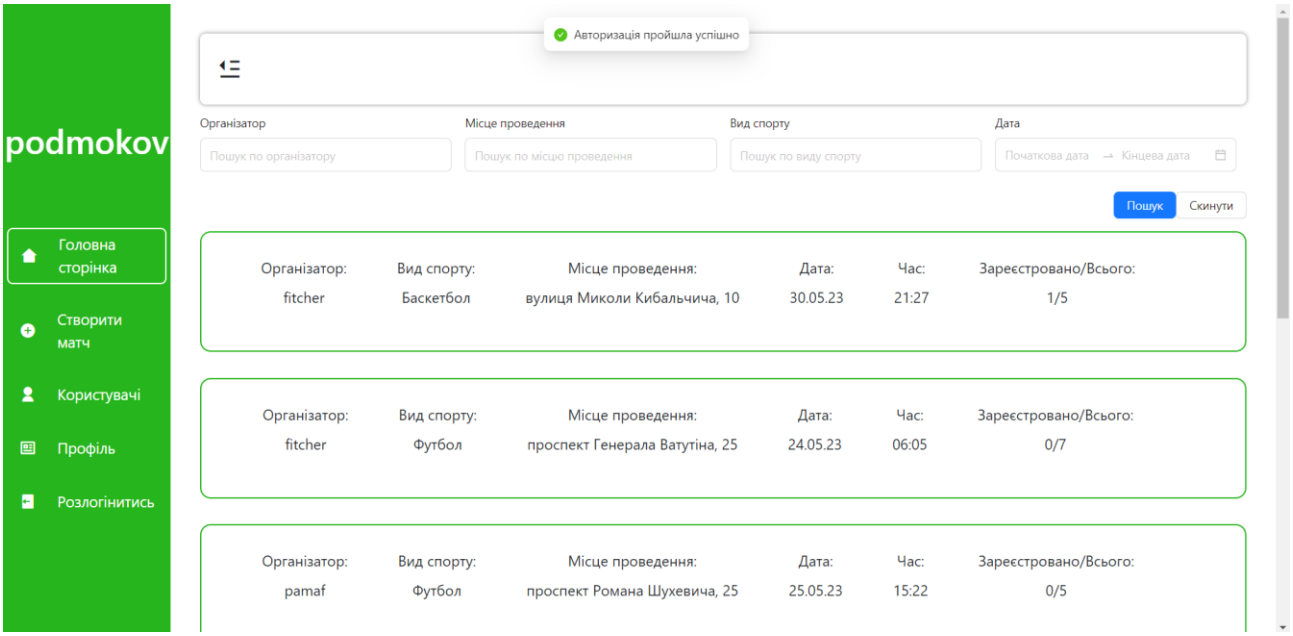


Рисунок 3.5 – Успішна авторизація користувача

Тепер необхідно перейти на сторінку “Створити матч”, на цій сторінці знаходяться створені користувачем матчі, можливість їх редагування та видалення, а також кнопка для створення матчу. На рисунку 3.6 зображена сторінка “Створити матч”.

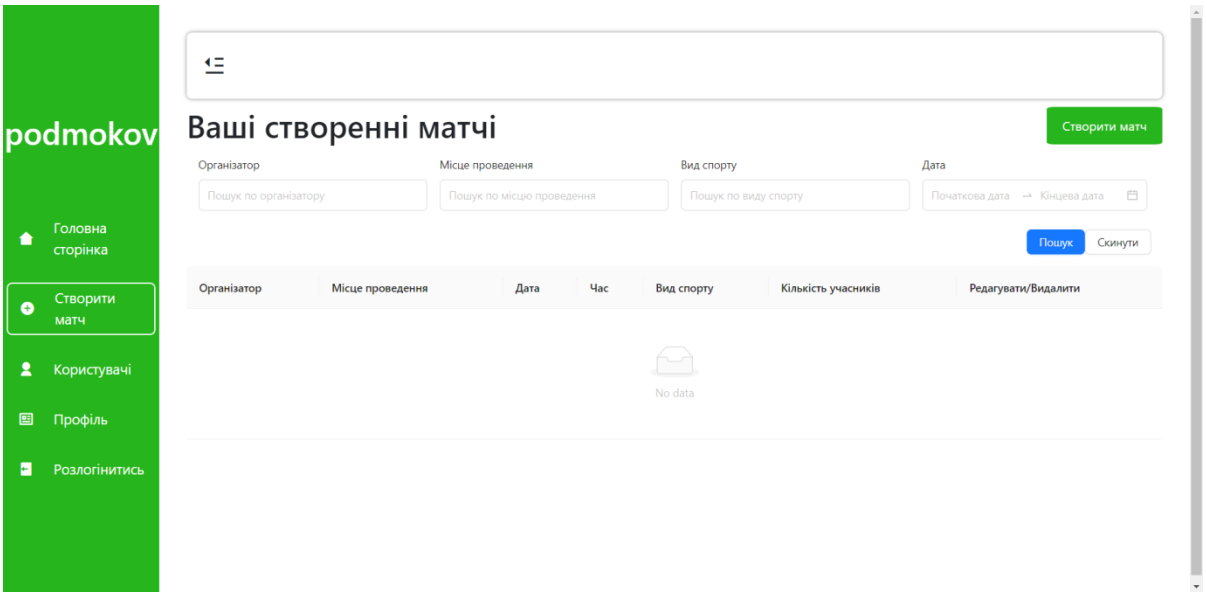


Рисунок 3.6 – Сторінка “Створити матч”

Для створення матчу необхідно натиснути на кнопку “Створити матч”. Після натискання на неї з'явиться модальне вікно, в яке необхідно ввести дані про матч. На рисунку 3.7 зображено процес створення матчу.

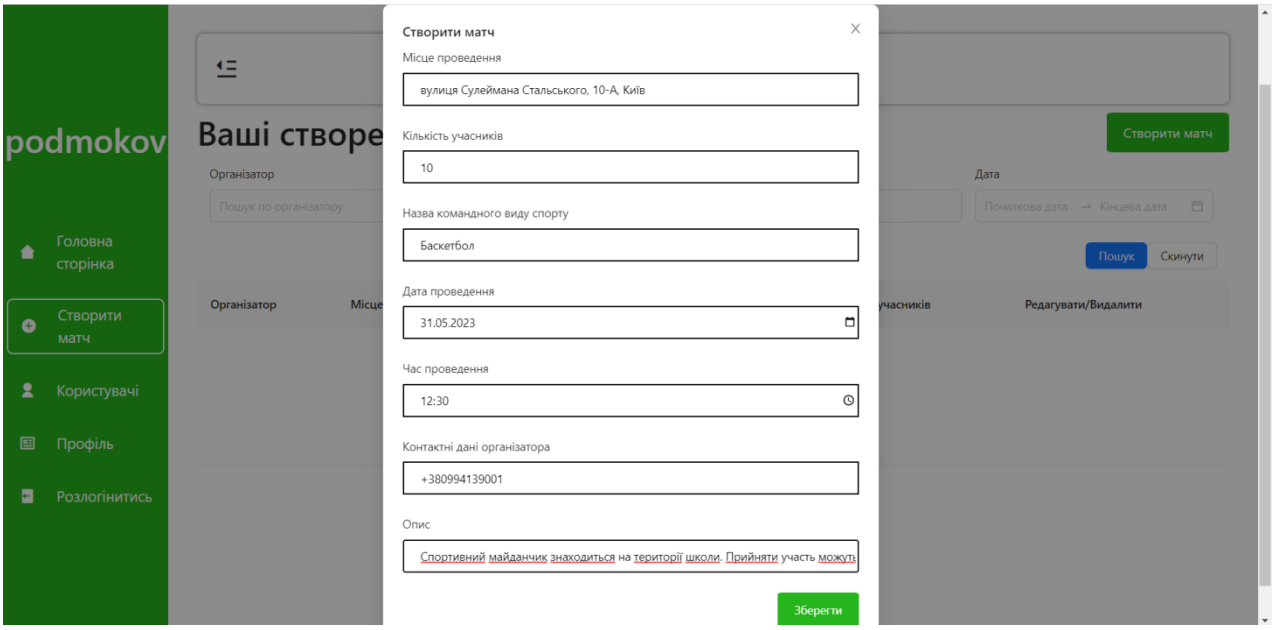


Рисунок 3.7 – Створення матчу

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішне створення матчу, і на сторінці з’явиться щойно створений матч. На рисунку 3.8 зображено успішне створення матчу.

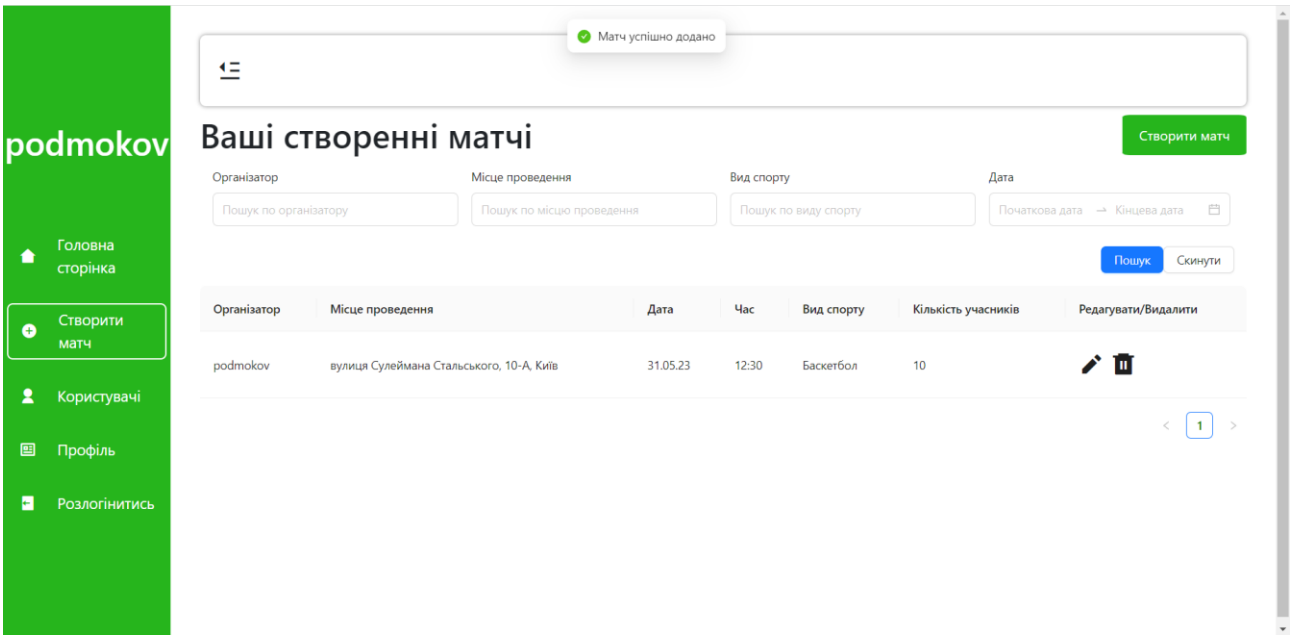


Рисунок 3.8 – Успішне створення матчу

Після того як матч успішно створено, необхідно перейти на головну сторінку і знайти щойно створений матч у списку усіх матчів. Це можна зробити використовуючи пошук по сайту, або самостійно знайти матч на сторінці. На рисунку 3.9 зображено пошук матчу за організатором.

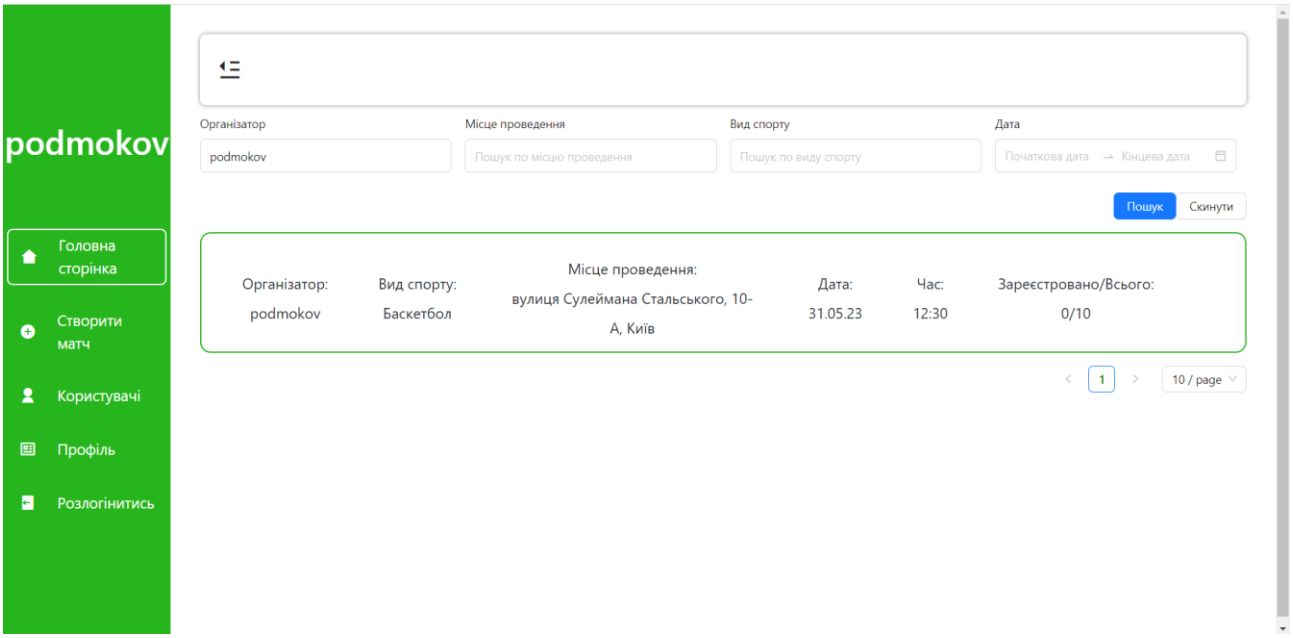


Рисунок 3.9 – Пошук матчу за організатором

Після того як матч було знайдено, можна перейти на сторінку матчу, та передивитись повну інформацію про нього. Тепер всі користувачі, що

побачать створений матч, зможуть зареєструватись на нього, щоб прийняти у ньому участь. На рисунку 3.10 зображено сторінку матчу.

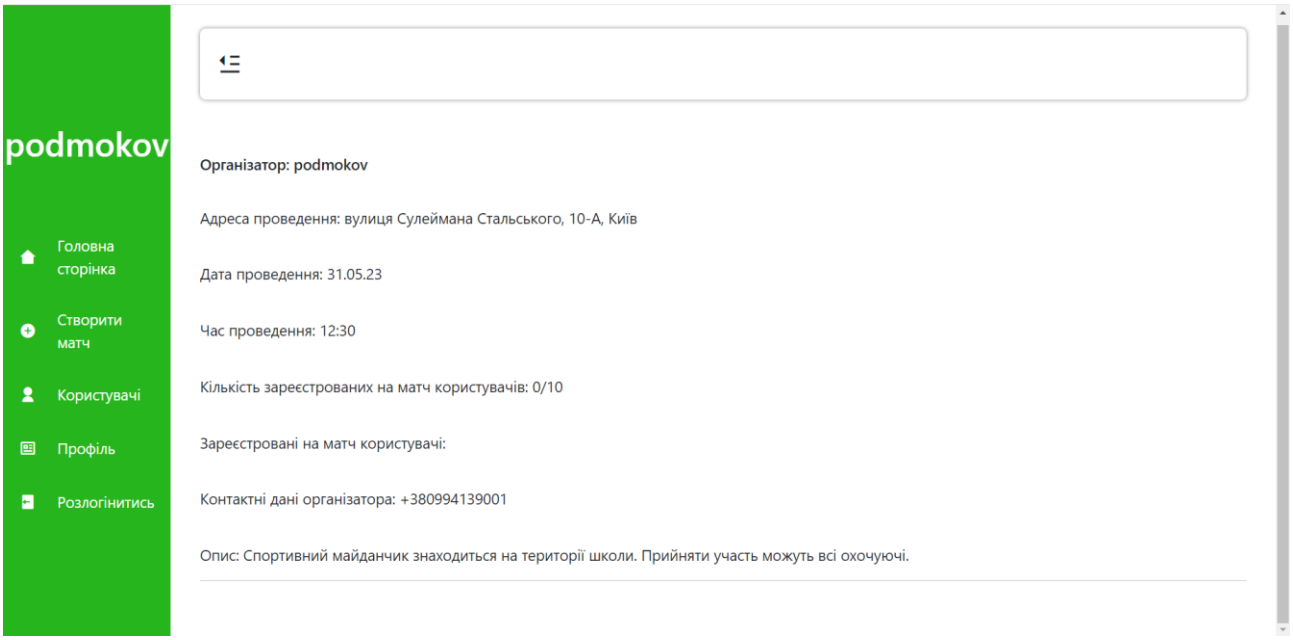


Рисунок 3.10 – Сторінка матчу

Висновки до розділу

В першому розділі був проведений аналіз якості програмного забезпечення за допомогою сервісу SonarCloud.

В другому розділі було проведено мануальне тестування ПЗ, яке включало в себе такі тести як: проведення тесту реєстрації користувача з використанням коректних і некоректних вхідних даних, проведення тесту авторизації користувача з використанням коректних і некоректних вхідних даних, проведення тесту створення матчу з використанням коректних і некоректних вхідних даних і т.д.

Крім того, у якості контрольного прикладу було наведено процеси від реєстрації користувача до створення матчу, та перегляду створеного матчу.

В цілому, проведений аналіз якості ПЗ та опис процесів тестування дозволили підтвердити високу функціональність, ефективність та легкість використання програмного забезпечення. Результати тестування свідчать про його успішну роботу, відповідність вимогам та потребам користувачів.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Серверну частину програмного забезпечення було вирішено розгорнути на платформі Render, а клієнтську частину на платформі Netlify.

Для розгортання ПЗ на цих платформах перш за все необхідно опублікувати код застосунку на репозиторії GitHub, задля подальшого підключення до нього. На рисунку 4.1 зображено опублікований код застосунку на репозиторії GitHub.

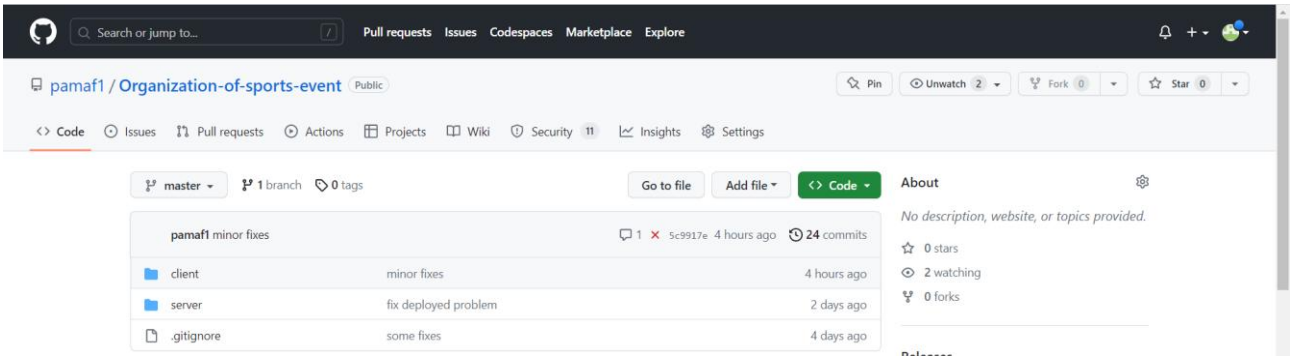


Рисунок 4.1 – Код застосунку на репозиторії GitHub

Після того як код опубліковано, можна почати почергове розгортання серверної і клієнтської частини.

Першою необхідно розгорнути серверну частину, щоб після розгортання клієнтської частини, вони одразу могли бути пов’язаними. Адже клієнтська частина посилає запити на серверну і отримує від неї відповіді.

Задля розгортання серверної частини на платформі Render необхідно зареєструватися за допомогою GitHub, щоб одразу отримати доступ до всіх своїх репозиторіїв. Після цього перейти на сторінку Dashboard, натиснути на кнопку “New” і в розгорнутому списку обрати пункт “Web Service”. На рисунку 4.2 зображено сторінка Dashboard платформи Render.

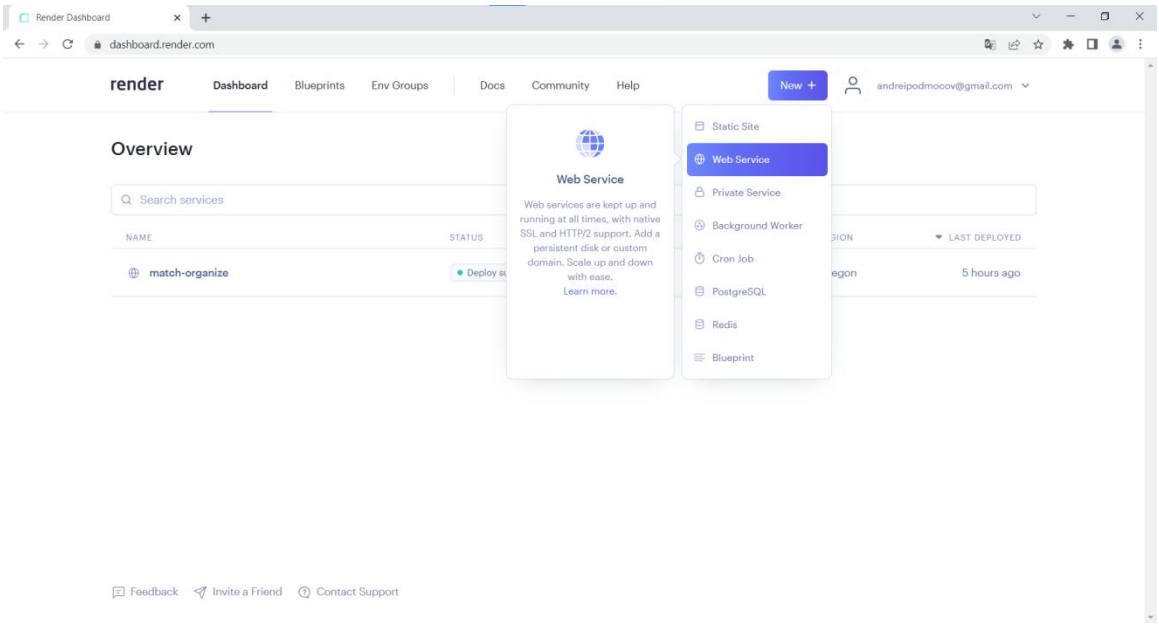


Рисунок 4.2 – Сторінка Dashboard платформи Render

На наступній сторінці необхідно підключити потрібний репозиторій. На рисунку 4.3 зображено підключення необхідного репозиторія.

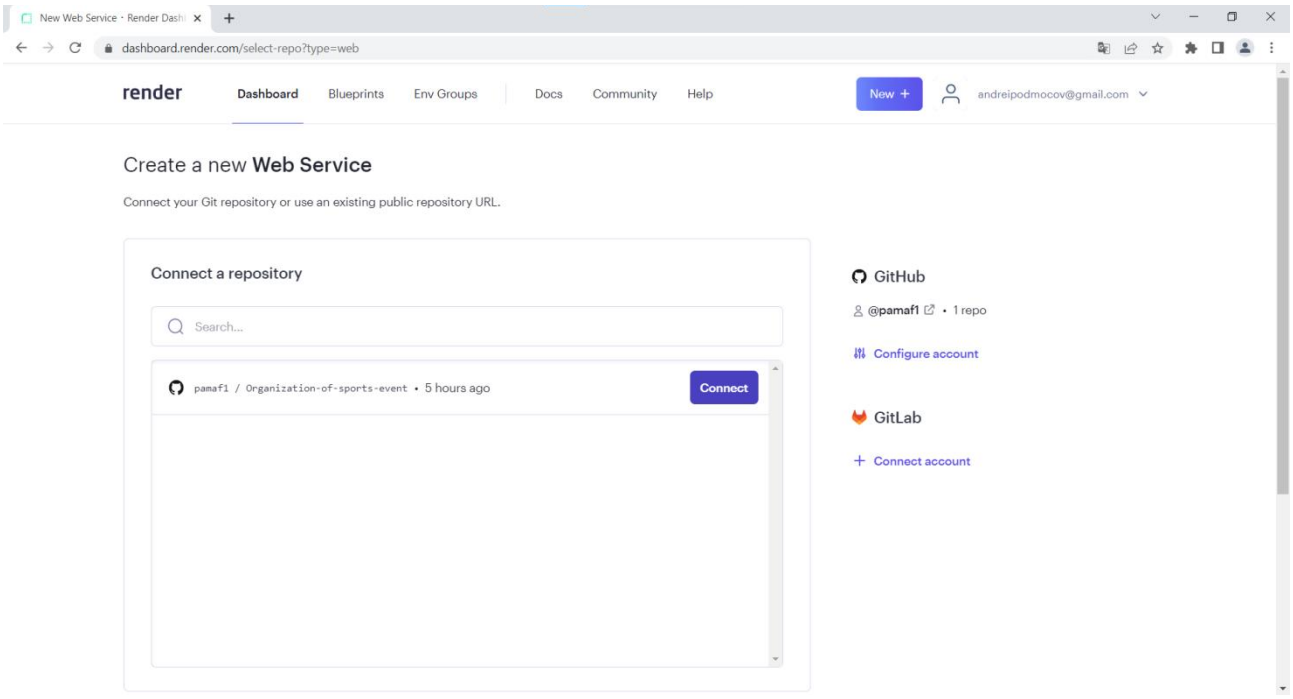


Рисунок 4.3 – Підключення необхідного репозиторія

Після того як репозиторій буде підключено, необхідно заповнити інформацію для розгортання, таку як: назва сервісу, регіон розгортання, гілка

репозиторію, шлях до серверної частини на репозиторії, програмна платформа. На рисунку 4.4 зображено інформацію для розгортання.

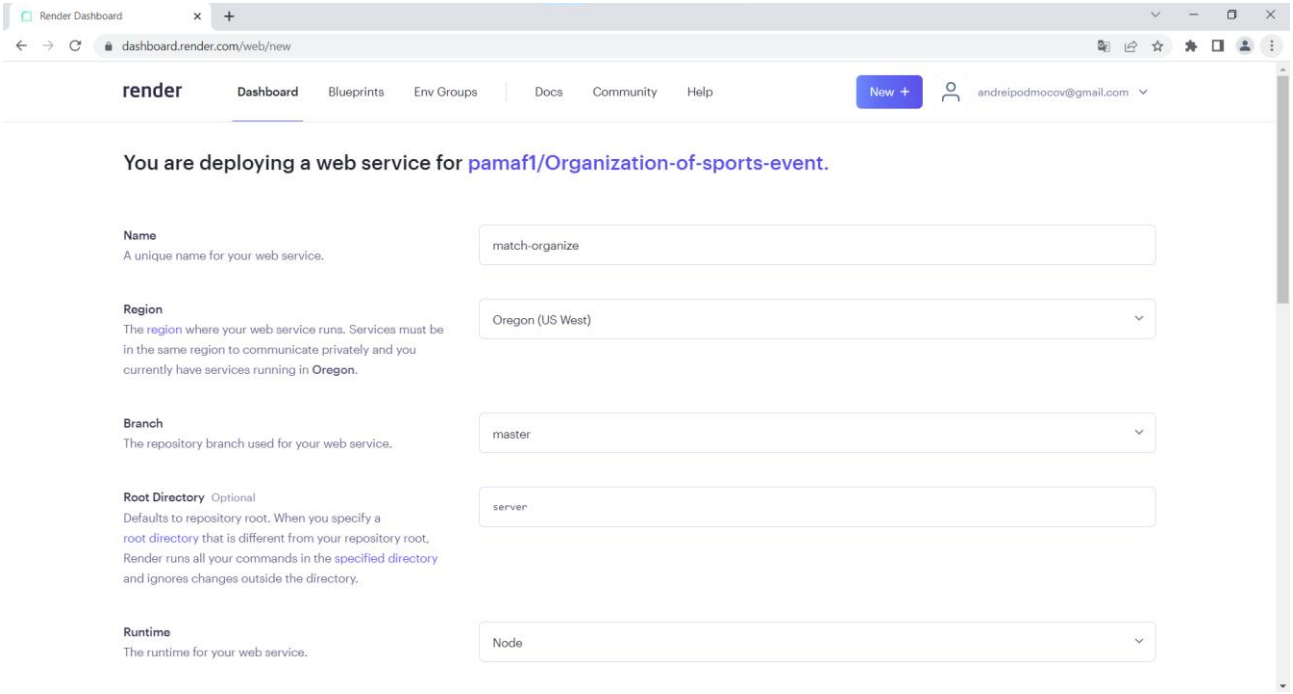


Рисунок 4.4 – Інформація для розгортання

Окрім цього необхідно вказати за допомогою яких команд запускається ПЗ. У випадку розробленого програмного забезпечення це “npm run build”, що запускає “npm install” для завантаження всіх бібліотек і залежностей, а також “npm run start”, що запускає “nodemon server.js” для безпосереднього запуску серверу. На рисунку 4.5 зображені команди для запуску ПЗ.

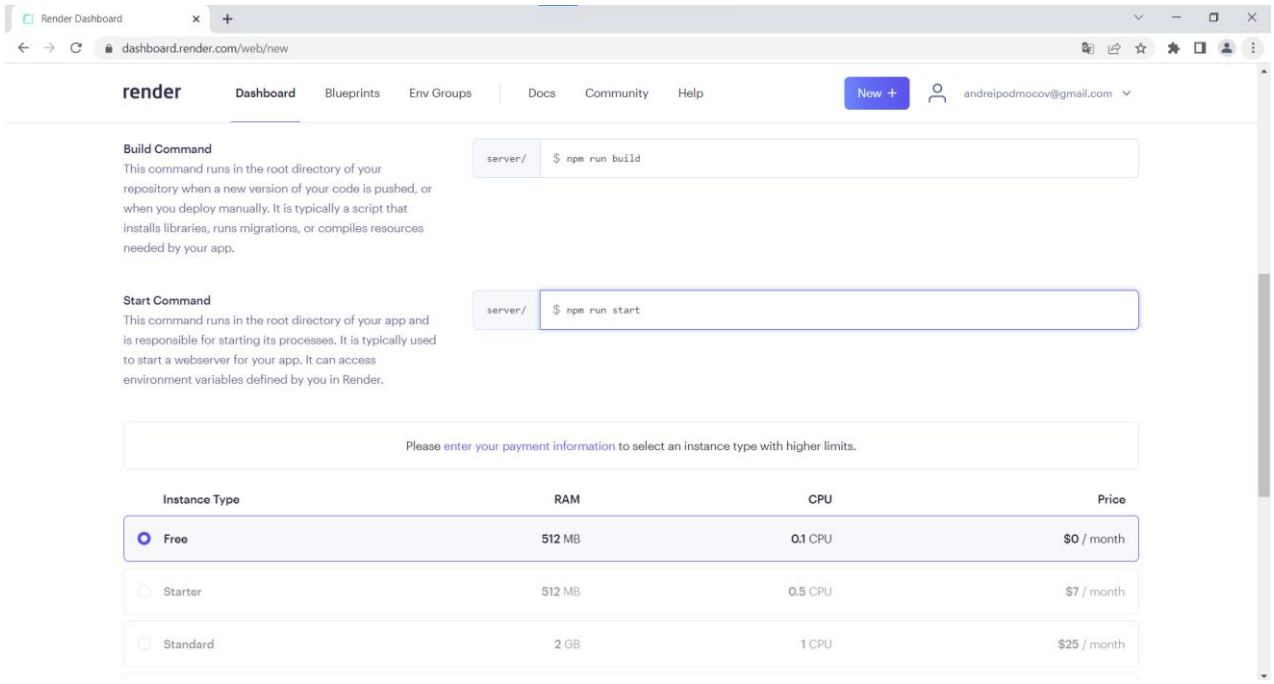


Рисунок 4.5 – Команди для запуску ПЗ

Також у “Advanced” налаштуваннях необхідно вказати зміні з файлу .env, а саме дані для підключення до БЗ і секретний ключ JWT токену, і додати сам файл .env. На рисунку 4.6 зображено додавання змінних середовища.

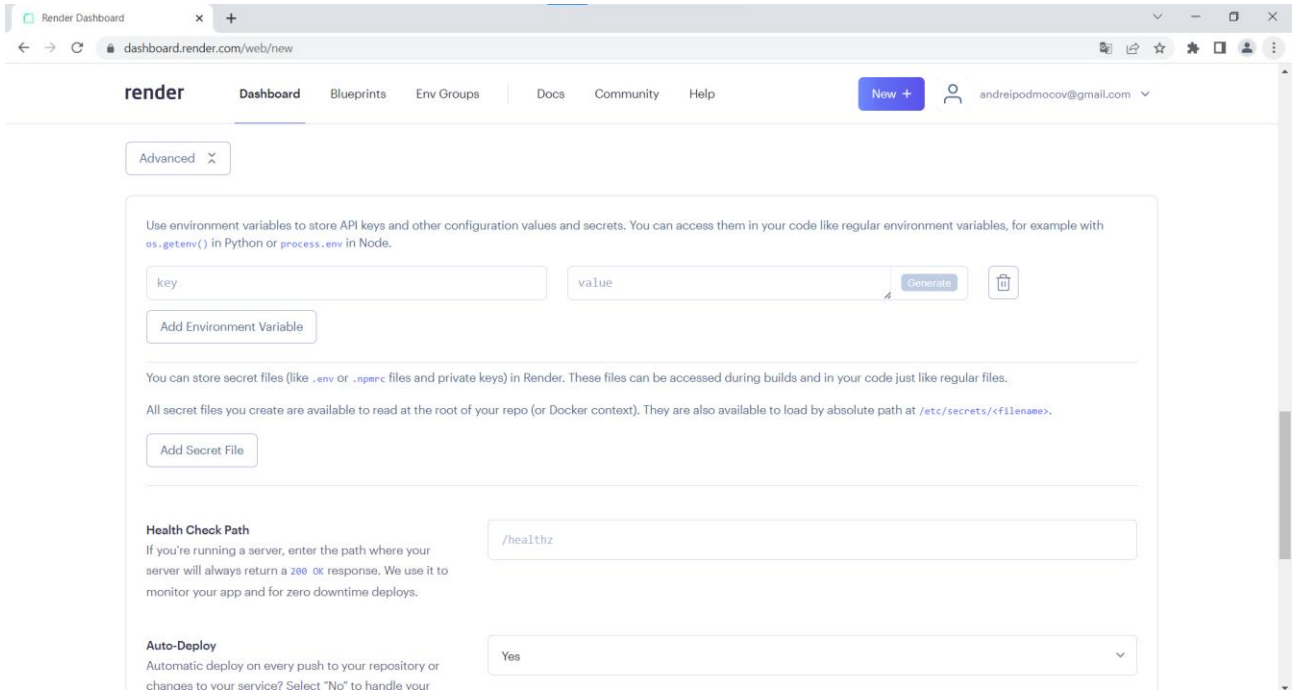


Рисунок 4.6 – Додавання змінних середовища

Після заповнення всіх необхідних даних потрібно натиснути на кнопку “Create Web Service”. Далі розпочинається розгортання серверної частини. На рисунку 4.7 зображено розгортання серверної частини.

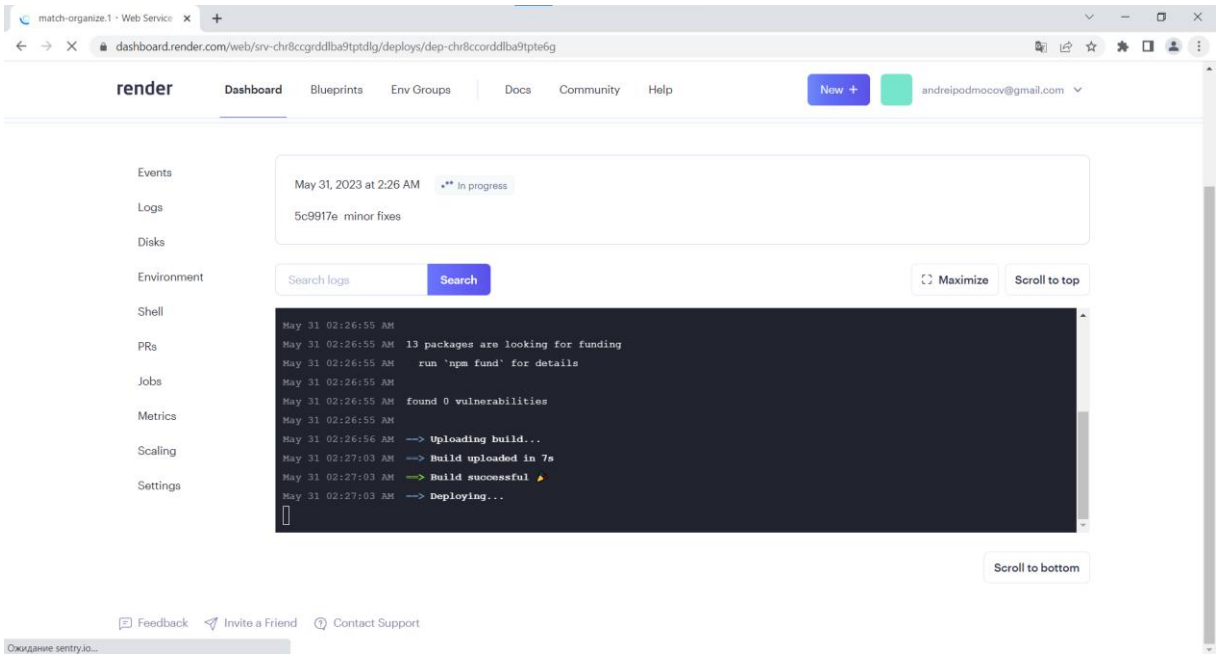


Рисунок 4.7 – Розгортання серверної частини

Після того як розгортання успішно завершено, необхідно скопіювати посилання на розгорнуту серверну частину і замінити ним локальне посилання в коді клієнтської частини для зв’язку із сервером. На рисунку 4.8 зображено успішне розгортання серверної частини.

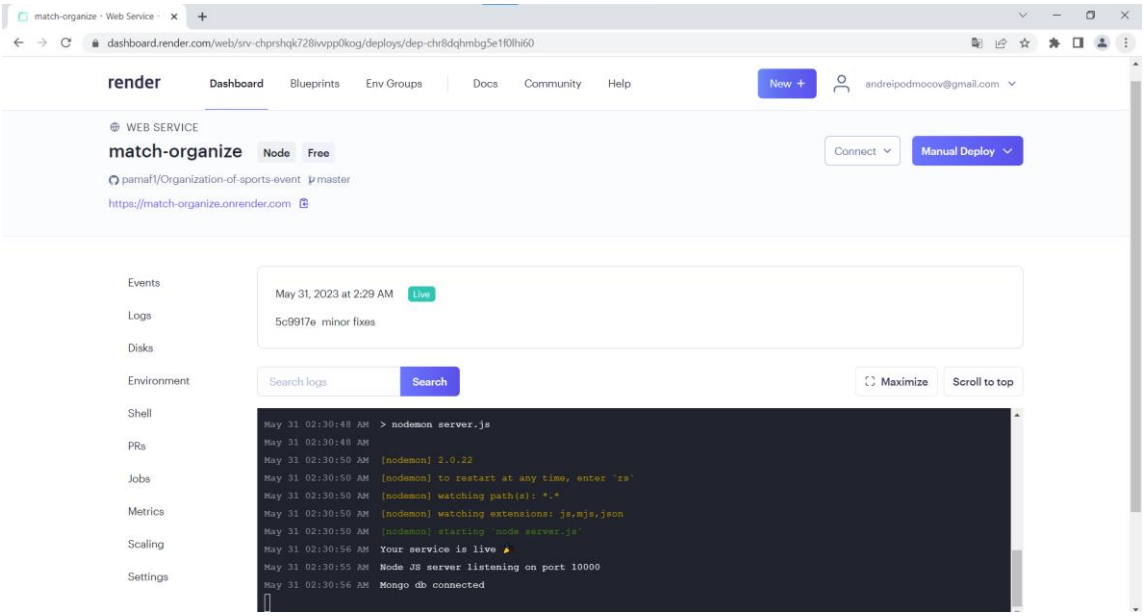


Рисунок 4.8 – Успішне розгортання серверної частини

Тепер можна перейти до розгортання клієнтської частини ПЗ. Для цього необхідно так само зареєструватися на платформі Netlify за допомогою GitHub. Після цього перейти на сторінку Sites, натиснути на кнопку “Add new site” і в розгорнутому списку обрати пункт “Import an existing project”. На рисунку 4.9 зображена сторінка Sites платформи Netlify.

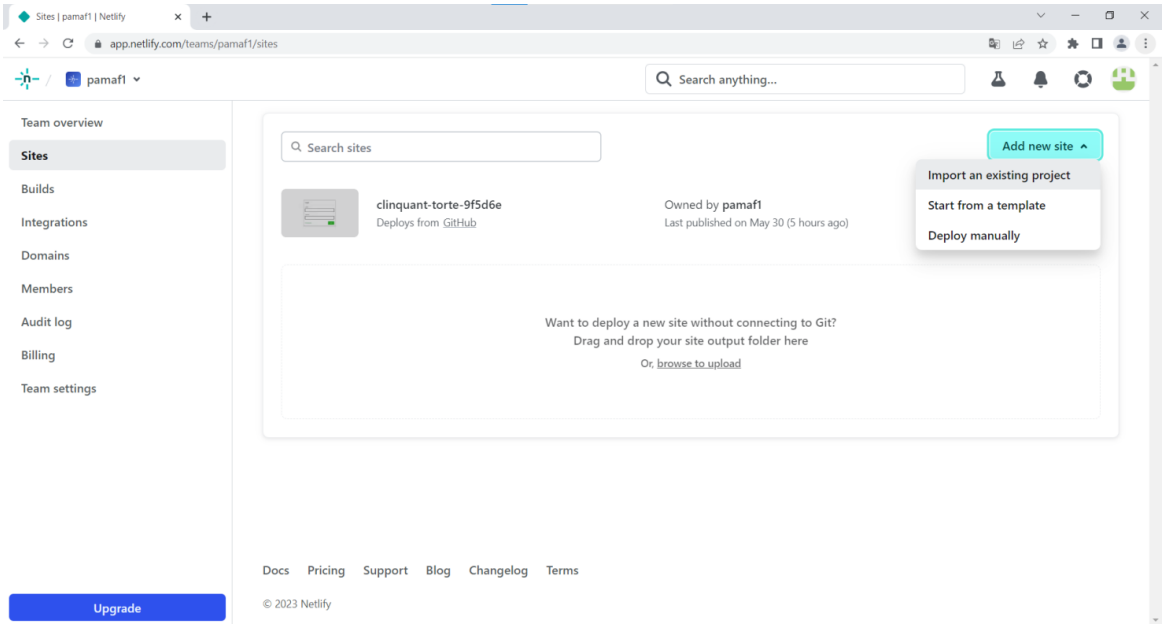


Рисунок 4.9 – Сторінка Sites платформи Netlify

На наступній сторінці необхідно підключити потрібний репозиторій. На рисунку 4.10 зображено підключення необхідного репозиторія.

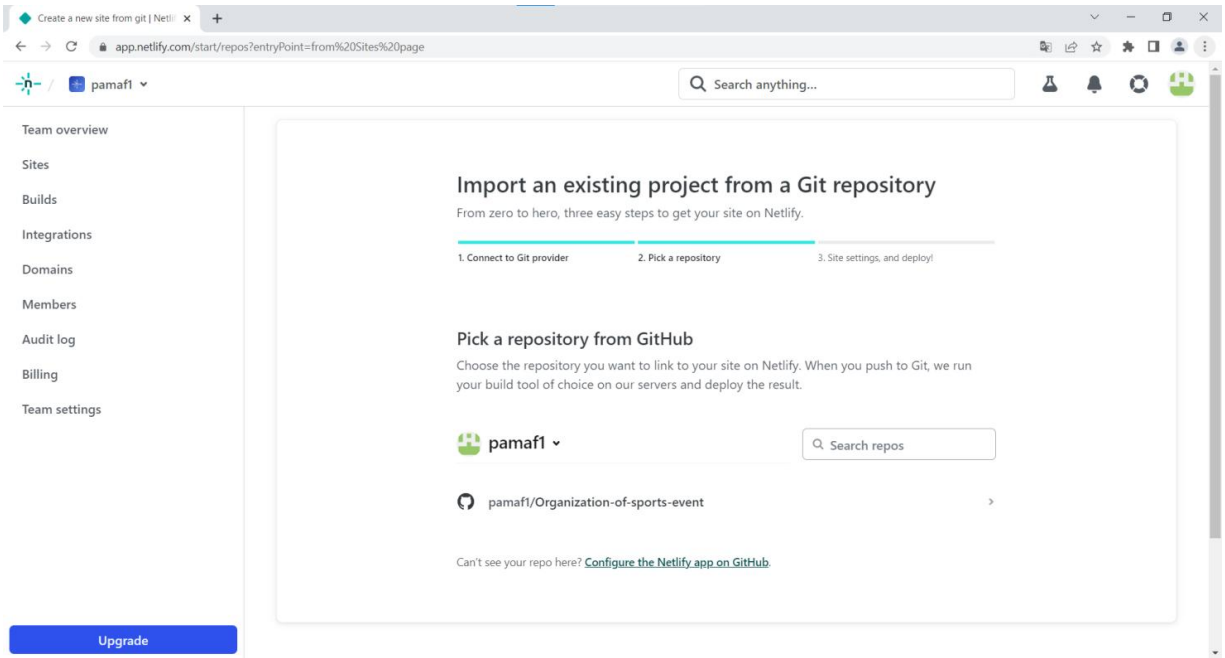


Рисунок 4.10 – Підключення необхідного репозиторія

Після того як репозиторій буде підключено, необхідно заповнити інформацію для розгортання, таку як: власник сайту, гілка репозиторію, шлях до клієнтської частини на репозиторії, папку для публікації, команди запуску. У випадку розробленого програмного забезпечення це “npm run build”, що запускає “react-scripts build” для об’єднання всіх модулів, зменшенню та оптимізації для розгортання. Netlify сам встановлює всі необхідні бібліотеки і залежності, тому тут непотрібно власноруч вказувати “npm install”. Після заповнення всіх необхідних даних потрібно натиснути на кнопку “Deploy site”. Далі розпочинається розгортання клієнтської частини. На рисунку 4.11 зображена інформація для розгортання.

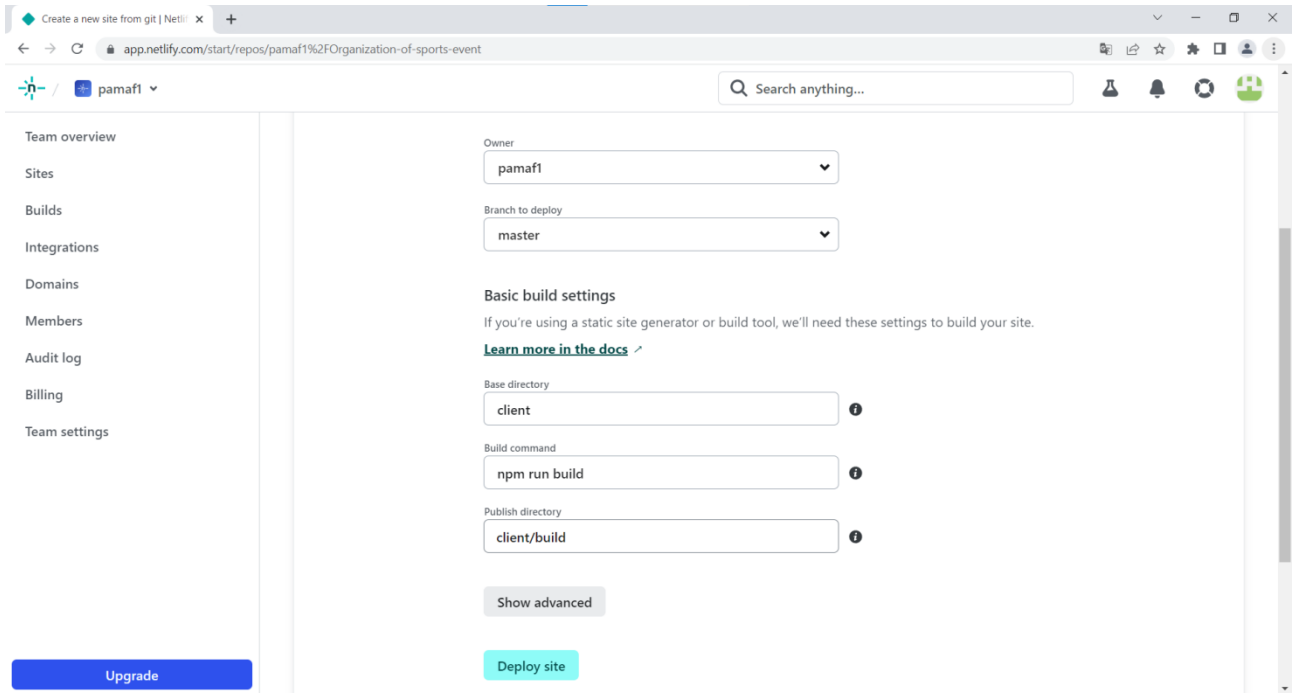


Рисунок 4.11 – Інформація для розгортання

Коли розгортання клієнтської частини буде завершено, можна буде перейти на створений сайт через посилання “Open production deploy”. На рисунку 4.12 зображено успішне розгортання клієнтської частини.

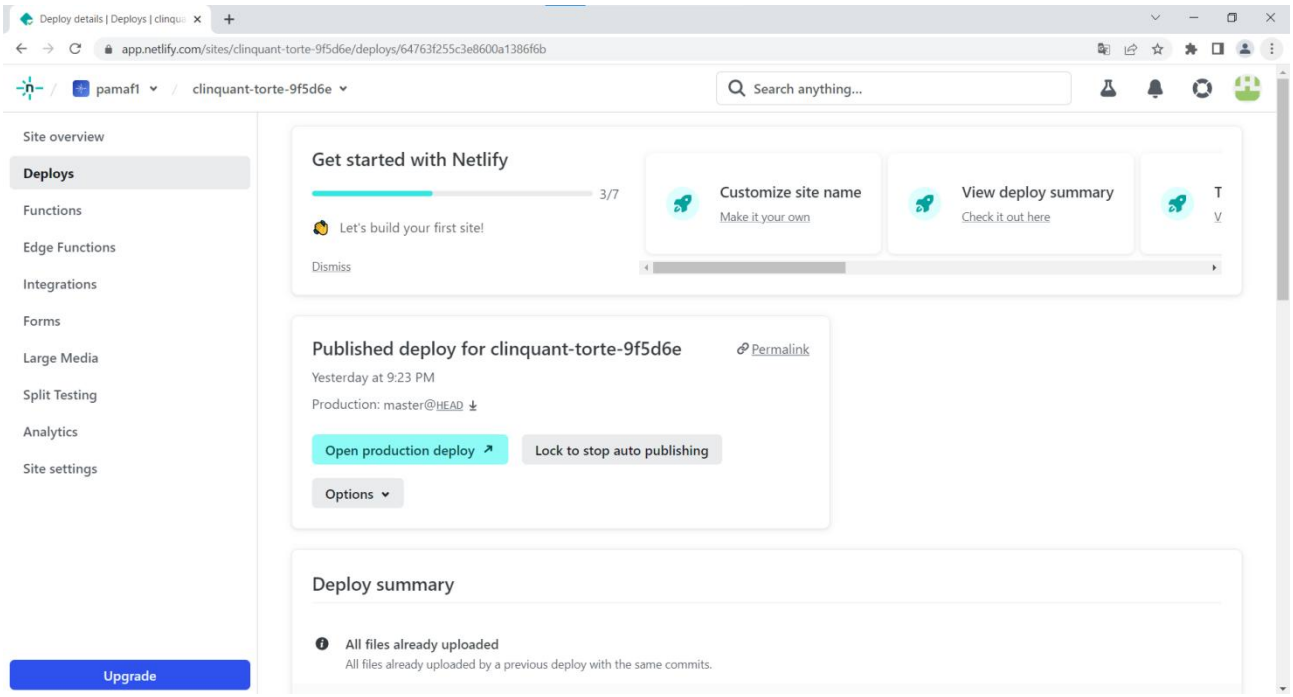


Рисунок 4.12 – Успішне розгортання клієнтської частини

Тепер перевіримо чи працює опублікований сайт, перейшовши за посиланням. На рисунку 4.13 зображена сторінка авторизації розробленого веб-застосунку.

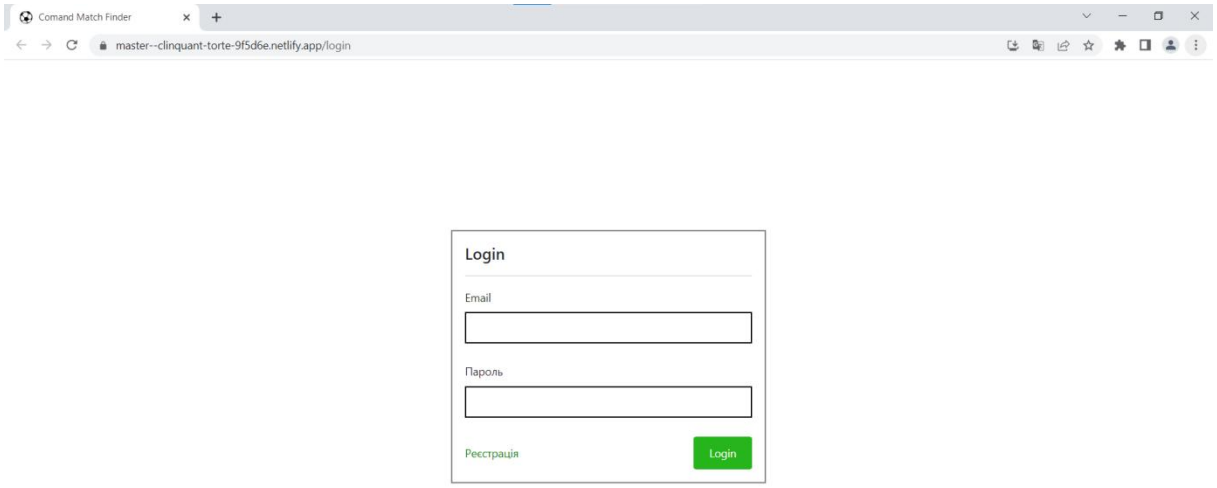


Рисунок 4.13 – Сторінка авторизації розробленого веб-застосунку

Перейшовши за посиланням, бачимо, що сайт працює. Для повної перевірки авторизуємось, щоб побачити чи працює серверна частина. На рисунку 4.14 зображена головна сторінка розробленого веб-застосунку.

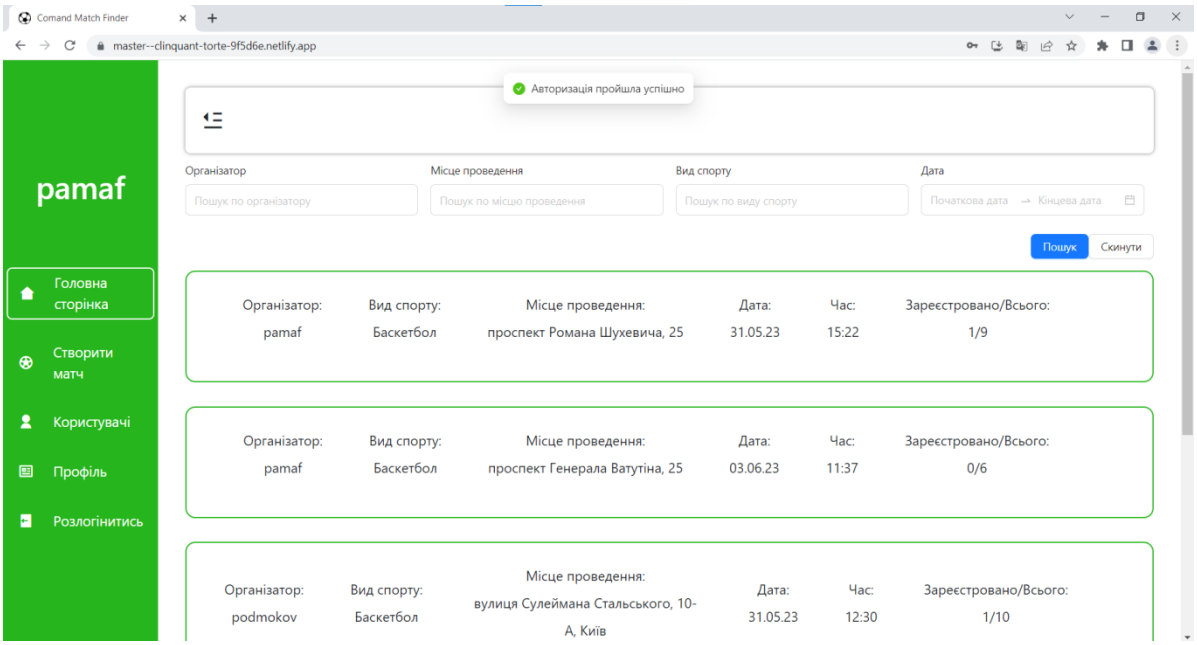


Рисунок 4.14 – Головна сторінка розробленого веб-застосунку

Авторизація пройшла успішно, і нас було перенаправлено на головну сторінку, на якій відображаються усі дані. Це означає, що серверна і клієнтська частина були успішно розгорнуті.

4.2 Підтримка програмного забезпечення

Так як обране ПЗ це веб-застосунок, користувачі одразу отримують останню його версію. Для оновлення веб-застосунку будуть публікуватись його нові версії на репозиторії GitHub, після чого обрані платформи будуть автоматично перерозгортати веб-застосунок за останнім коммітом на GitHub.

Якщо ж користувачі бажають отримувати останню версію початкового коду, для власного використання, їм необхідно перейти на GitHub репозиторій веб-застосунку, натиснути на кнопку “<Code>” і в розгорнутому списку копіювати посилання на репозиторій для його клонування. На рисунку 4.15 зображено GitHub репозиторій веб-застосунку.

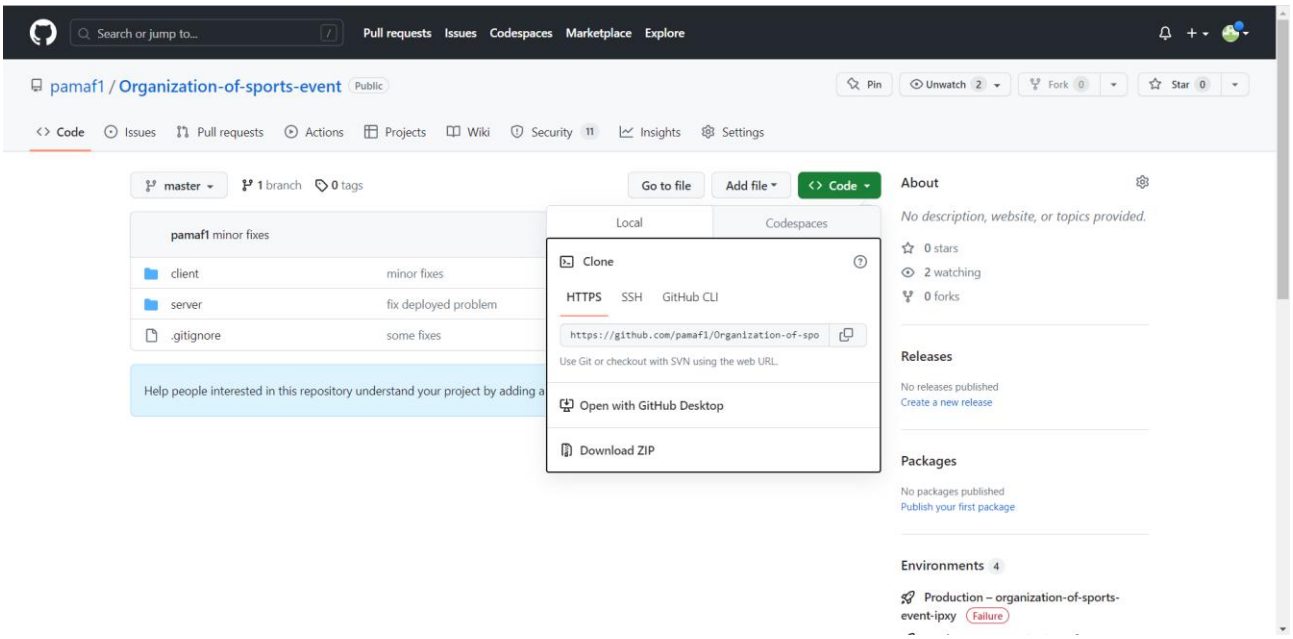


Рисунок 4.15 – GitHub репозиторій веб-застосунку

Для клонування репозиторію необхідно обрати директорію, в яку користувач бажає зберегти репозиторій і в bash терміналі ввести команду “git clone https://”. Після того як клонування пройшло успішно, необхідно завантажити всі необхідні залежності за допомогою наступних команд: спочатку завантажуються залежності для серверної частини “cd server” і “npm install”, потім для клієнтської частини “cd ../”, “cd client” і “npm install”. Окрім цього, для запуску клієнтської частини потрібно замінити посилання для зв'язку з сервером з розгорнутого на локальні посилання, які вказані у файлі server.js серверної частини. Також у директорії серверу необхідно створити файл .env, в якому вказати посилання на підключення до MongoDB і секретний ключ JWT токену. Тепер можна запускати сервер і клієнт. Сервер запускається у директорії серверу за допомогою команди “nodemon server”, а клієнт у директорії клієнту за допомогою команди “npm start”.

Висновки до розділу

У цьому розділі було описано процес розгортання програмного забезпечення на платформах Render та Netlify. Серверна частина була успішно розгорнута на Render, а клієнтська частина – на Netlify. Після

розгортання обидві частини програмного забезпечення були працездатні, що підтверджує успішну реалізацію процесу розгортання. Крім того, було описано як буде проводитись підтримка програмного забезпечення. Оновлення веб-застосунку автоматично розгортаються за останнім комітом на репозиторії GitHub, забезпечуючи користувачам останню версію веб-застосунку. Також було описано як користувачі можуть клонувати репозиторій веб-застосунку та використовувати останню версію початкового коду для власних потреб.

ВИСНОВКИ

В результаті виконання дипломного проєкту було реалізовано веб-застосунок для підтримки діяльності організаторів командних аматорських ігор. Результати дипломного проєкту можуть бути використані в різних галузях і сферах. Зокрема, розроблений веб-застосунок може бути використаний для організації матчів та спортивних івентів. Окрім цього було реалізовано весь функціонал, який планувалося реалізувати.

В якості середовища розробки обрано Visual Studio Code. В свою чергу в якості технології було обрано: JavaScript і MERN-stack

У якості БД використано MongoDB.

Вперше реалізовано можливість створення власного спортивного івенту за зручною формою.

Результати дипломного проєктування відображають стан вирішення всіх поставлених задач, включаючи розробку серверної та клієнтської частини програмного забезпечення, розробку зручного веб-інтерфейсу для користувачів, розробку API для взаємодії клієнта з сервером, розробку функціоналу створення спортивного івенту та розробку функціоналу оцінювання користувачів.

Дослідження в рамках цієї тематики є актуальними, тому продовження досліджень у цьому напрямку може включати розширення функціональності веб-застосунку, покращення його ефективності та безпеки, а також адаптацію для використання в інших спортивних галузях або сферах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Amal Alqahtani, Maram Alqahtani, Bothaynah Alshehri, Manar Abumelha. "Developing and Implementing a Website for Sports Clubs". International Journal of Scientific Research in Science and Technology. 2020. Volume 7. Issue 2. P. 135-146.

2) Документація MongoDB [Електронний ресурс] — Режим доступу: https://docs.mongodb.com/ .

3) Документація Express [Електронний ресурс] — Режим доступу: https://expressjs.com/ .

4) Документація React [Електронний ресурс] — Режим доступу: https://reactjs.org/tutorial/tutorial.html/ .

5) Документація Node.js [Електронний ресурс] — Режим доступу: https://nodejs.org/en/docs/ .

6) Документація Axios [Електронний ресурс] — Режим доступу: https://www.npmjs.com/package/axios/ .

7) Документація Bcrypt [Електронний ресурс] — Режим доступу: https://www.npmjs.com/package/bcrypt/ .

8) Документація Redux [Електронний ресурс] — Режим доступу: https://redux.js.org/introduction/getting-started/ .

9) Документація Mongoose [Електронний ресурс] — Режим доступу: https://mongoosejs.com/ .

10) Документація Nodemon [Електронний ресурс] — Режим доступу: https://nodemon.io/ .

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015404099

Дата перевірки:
03.06.2023 08:46:40 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2023 08:55:49 EEST

ID користувача:
76913

Назва документа: **IT-94_Подмоков_ПЗ**

Кількість сторінок: **68** Кількість слів: **9931** Кількість символів: **78225** Розмір файлу: **3.53 MB** ID файлу: **1015067796**

13.3%
Схожість

Найбільша схожість: **4.76%** з джерелом з Бібліотеки (ID файлу: **1014981451**)

4.85% Джерела з Інтернету	142	Сторінка 70
13% Джерела з Бібліотеки	253	Сторінка 72

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ОРГАНІЗАТОРІВ КОМАНДНИХ АМАТОРСЬКИХ ІГОР

Текст програми

КПІ.ІТ-9419.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
Олександр СТЕЛЬМАХ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Андрій ПОДМОКОВ

Київ – 2023

Файл userMatches.js

```
function UserMatches() {
  const navigation = useNavigate();
  const dispatch = useDispatch();
  const [showMatchForm, setShowMatchForm] = useState(false);
  const { user } = useSelector((state) => state.users);
  const [matches, setMatches] = useState([]);
  const [selectedMatch, setSelectedMatch] = useState(null);
  const [filteredMatches, setFilteredMatches] = useState([]);
  const [searchForm] = Form.useForm();
  const getMatches = async () => {
    try {
      dispatch(ShowLoading());
      const matchResponse = await GetMatchByUserId({user: user._id});
      setMatches(matchResponse.data);
      setFilteredMatches(matchResponse.data);
      dispatch(HideLoading());
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);      }   }
  const deleteMatch = async (id) => {
    try {
      dispatch(ShowLoading());
      const matchResponse = await DeleteMatch({_id: id._id});
      message.success(matchResponse.message);
      getMatches();
      dispatch(HideLoading());
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);      }   }
  const columns = [      {
      title: 'Організатор',
      dataIndex: 'createdBy',
      render: (text, record) =>
      <div className='login-rule' onClick={() => navigation('/profile')}>
        {record.createdBy.login}
      </div>      },
        {
      title: 'Місце проведення',
      dataIndex: 'place'    },
    {
```

```
      title: 'Дата',
      dataIndex: 'date',
      render: (text) => makeDate(text)       },
    {
      title: 'Час',
      dataIndex: 'time'       },
    {
      title: 'Вид спорту',
      dataIndex: 'type'       },
    {
      title: 'Кількість учасників',
      dataIndex: 'partisipants'       },
    {
      title: 'Редагувати/Видалити',
      dataIndex: 'action',
      render: (action, record) => (
        <div className='d-flex gap-2'>
          <i class="ri-pencil-fill" onClick={() => {
              setSelectedMatch(record);
              setShowMatchForm(true);
            }}>
          </i>
          <i class="ri-delete-bin-6-fill" onClick={() => {
              deleteMatch(record);
            }}>       </i>              </div>          )      }     ];
  useEffect(() => {
    getMatches();
  }, []);
  const handleSearch = (values) => {
    const { organizer, place, dateRange, sportType } = values;
    let filteredData = matches;
  if (organizer) {
     filteredData = filteredData.filter((match) =>
      match.createdBy.login.toLowerCase().includes(organizer.toLowerCase())          );       }
    if (place) {
     filteredData = filteredData.filter((match) =>
      match.place.toLowerCase().includes(place.toLowerCase())          );       }
    if (dateRange && dateRange.length === 2) {
      const [start, end] = dateRange;
      const startDate = moment(start.format('YYYY-MM-DD'));
      const endDate = moment(end.format('YYYY-MM-DD')).endOf('day');
```

```
            filteredData = filteredData.filter((match) => {
              const matchDate = moment(match.date, 'YYYY-MM-DD');
                  return matchDate.isSameOrAfter(startDate, 'day') && matchDate.isSameOrBefore(endDate,
'day');            });       }
        if (sportType) {
          filteredData = filteredData.filter((match) =>
            match.type.toLowerCase().includes(sportType.toLowerCase())       );       }
            setFilteredMatches(filteredData);    };
        const handleReset = () => {
        searchForm.resetFields();
        setFilteredMatches(matches);    };
    return (       <div>
          <div className="d-flex justify-content-between">
            <h1>Ваші створенні матчі</h1>
            <button className='secondary-btn'
             onClick={() => setShowMatchForm(true)}>
              Створити матч
            </button>
          </div>
                <Form form={searchForm} layout="vertical" onFinish={handleSearch} style={{padding:
'15px'}}>
            <Row gutter={[16, 16]}>
              <Col span={6}>
              <Form.Item name="organizer" label="Організатор">
                <Input placeholder="Пошук по організатору" />
              </Form.Item>
              </Col>
              <Col span={6}>
              <Form.Item name="place" label="Місце проведення">
                <Input placeholder="Пошук по місцю проведення" />
              </Form.Item>
              </Col>
              <Col span={6}>
              <Form.Item name="sportType" label="Вид спорту">
                <Input placeholder="Пошук по виду спорту" />
              </Form.Item>
              </Col>
              <Col span={6}>
              <Form.Item name="dateRange" label="Дата" style={{height: "40px"}}>
                        <RangePicker placeholder={['Початкова дата', 'Кінцева дата']} style={{height:
"40px"}}/>
              </Form.Item>
```

```
            </Col>
          </Row>
          <Row justify="end">
            <Col>
            <Button type="primary" htmlType="submit">
              Пошук
            </Button>
            <Button onClick={handleReset}>Скинути</Button>
            </Col>
          </Row>
          </Form>
        <Table dataSource={filteredMatches} columns={columns}/>
        {showMatchForm &&
        <MatchForm
          showMatchForm={showMatchForm}
          setShowMatchForm={setShowMatchForm}
          type= {selectedMatch ? 'edit' : 'add'}
          selectedMatch = {selectedMatch}
          setSelectedMatch = {setSelectedMatch}
          getData = {getMatches}
        />}
      </div>   )}
export default UserMatches
```

Файл matchRoute.js

```
const router = require("express").Router();
const Match = require("../models/matchModel");
const RegisterMatch = require("../models/registerMatchModel");
const authMiddleware = require("../middlewares/authMiddleware");
router.post("/", authMiddleware, async (req, res) => {
  try {
    req.body.createdBy = req.body.userId;
    await Match.create(req.body);
    return res.status(200).send({ message: "Матч успішно додано", success: true });    }
  catch (error) {
    res.status(500).send({ message: error.message, success: false });    }});
router.get("/get-match-by-userId", authMiddleware, async (req, res) => {
  try {
      const matches = await Match.find({createdBy:
req.body.userId}).populate('createdBy').populate('registeredUsers')
    res.status(200).send({ data: matches, success: true });     }
```

```
    catch (error) {
      res.status(500).send({ message: error.message, success: false });      }});
router.get("/get-all-match", authMiddleware, async (req, res) => {
  try {
    const matches = await Match.find().populate('createdBy').populate('registeredUsers');
    res.status(200).send({ data: matches, success: true });    }
   catch (error) {
    res.status(500).send({ success: false, message: error.message });    }});
router.get("/:id", authMiddleware, async (req, res) => {
  try {
    const matches = await Match.findById(req.params.id).populate('createdBy').populate('registeredUsers');
    res.status(200).send({ data: matches, success: true });    }
   catch (error) {
    res.status(500).send({ message: error.message, success: false });    }});
router.put("/update-match", authMiddleware, async (req, res) => {
  try {
   await Match.findByIdAndUpdate(req.body._id, req.body);
   return res.status(200).send({
    success: true,
    message: "Матч успішно оновлено",    });
  } catch (error) {
   res.status(500).send({ success: false, message: error.message });  }});
router.delete("/delete-match", authMiddleware, async (req, res) => {
  try {
   await RegisterMatch.findOneAndDelete({match: req.body._id});
   await Match.findByIdAndDelete(req.body._id);
   return res.status(200).send({
    success: true,
    message: "Матч успішно видалено",    });
  } catch (error) {
   res.status(500).send({ success: false, message: error.message });  }});
module.exports = router;
```

Файл matchForm.js

```
function MatchForm({showMatchForm, setShowMatchForm, type = 'add', selectedMatch, setSelectedMatch,getData}) {
   const dispatch = useDispatch();
   const [selectedDate, setSelectedDate] = useState('');
   const [selectedTime, setSelectedTime] = useState('');
   const [isButtonDisabled, setButtonDisabled] = useState(false);
   const handleDateChange = (e) => {
```

```
    const dateValue = e.target.value;
    setSelectedDate(dateValue);
    setSelectedTime('');
  };

  const handleTimeChange = (e) => {
    const timeValue = e.target.value;
    const timeDifference = timeToMinutes(timeValue) - timeToMinutes(getUkraineTime());
    if (selectedDate === getUkraineDate() && timeDifference < 3 * 60) {
      message.error('Ви не можете вибрати такий час');
      setSelectedTime(timeUpThreeHour());
    } else {        setSelectedTime(timeValue);      }   };

  const addMatch = async (values) => {
    try {
      if (isButtonDisabled) {          return;          }
      dispatch(ShowLoading())
      let response = null;
      if(type === 'add'){
        setButtonDisabled(true);
         response = await axiosInstance.post('https://match-organize.onrender.com/api/matches/', values)
    }
      else{
        setButtonDisabled(true);
          response = await axiosInstance.put('https://match-organize.onrender.com/api/matches/update-
match', {          ...values,
          _id: selectedMatch._id          })        }
      if(response.data.success)
      {
        message.success(response.data.message);        }
      else {
        message.error(response.data.message);        }
      getData()
      setShowMatchForm(false);
      setSelectedMatch(null);
      dispatch(HideLoading())      } catch (error) {
      message.error(error.message);
      dispatch(HideLoading())      }   }
    return (        <Modal width={600} title={type === 'add' ? 'Створити матч' : 'Змінити матч'}
open={showMatchForm} onCancel={() => {
      setSelectedMatch(null)
      setShowMatchForm(false)
```

```
        }} footer={false}>
          <Form layout='vertical' onFinish={addMatch} initialValues={selectedMatch}>
            <Row>
              <Col lg={24} xs={24}>
                <Form.Item label = 'Місце проведення' name = 'place'>
                  <input type='text' required maxLength='45'/>
                </Form.Item>
              </Col>
              <Col lg={24} xs={24}>
                <Form.Item label = 'Кількість учасників' name = 'partisipants'>
                  <input type='number' required min="5" max="30"/>
                </Form.Item>
              </Col>
              <Col lg={24} xs={24}>
                <Form.Item label = 'Назва командного виду спорту' name = 'type' rules={[
                 { pattern: /^[A-Za-zA-Яа-яЁёЇїІіЄєҐґ]+$/, message: "Вид спорту повинен містити тільки літери" },{ max: 10 }]}>
                  <input type='text' required maxLength='10'/>
                </Form.Item>
              </Col>
              <Col lg={24} xs={24}>
                <Form.Item label = 'Дата проведення' name = 'date'>
                  <input type='date' required min={getUkraineDate()} onChange={handleDateChange}/>
                </Form.Item>
              </Col>
              <Col lg={24} xs={24}>
                <Form.Item label = 'Час проведення' name = 'time'>
                    <input type='time' required min={selectedDate === getUkraineDate() ? timeUpThreeHour() : ''} onChange={handleTimeChange}/>
                </Form.Item>
              </Col>
              <Col lg={24} xs={24}>
                <Form.Item label = 'Контактні дані організатора' name = 'contacts'>
                  <input type='text' required maxLength='30'/>
                </Form.Item>
              </Col>
              <Col lg={24} xs={24} >
              <Form.Item label='Опис' name='description'>
                <input type='text' required maxLength='300'/>
              </Form.Item>
              </Col>
            </Row>
```

```
                    <div className='d-flex justify-content-end'>
<button className='secondary-btn' type="submit" disabled={isButtonDisabled}>Зберегти</button>
                    </div>
                  </Form>
                </Modal>    )}
            export default MatchForm
```

Файл matchInfo.js

```
            function MatchInfo() {
              const navigation = useNavigate();
              const dispatch = useDispatch();
              const [match, setMatch] = useState();
              const { id } = useParams();
              const { user } = useSelector((state) => state.users);
              const [isButtonDisabled, setButtonDisabled] = useState(false);
              const getMatchData = async () => {
                try {
                  dispatch(ShowLoading(true));
                  const response = await GetMatchById(id);
                  setMatch(response.data);
                  dispatch(HideLoading(false));       } catch (error) {          dispatch(HideLoading(false));
                  message.error(error.message);       }      };
              const registrateUser = async () => {
                try {           if (isButtonDisabled) {               return;           }
                  const currentDateTime = new Date();
                  const matchEndDateTime = new Date(match?.date + 'T' + match?.time + 'Z')
                  matchEndDateTime.setHours(matchEndDateTime.getHours() - 3);
                  if (currentDateTime > matchEndDateTime) {
                    message.error("Реєстрація на цей матч вже завершена");
                    return;           }
                  dispatch(ShowLoading());
                  setButtonDisabled(true);
                  const response = await axiosInstance.post("https://match-organize.onrender.com/api/registration/", {
                    match: match?._id,           });
                  dispatch(HideLoading());
                  if (response.data.success) {               message.success(response.data.message);
                    navigation("/profile");
                  } else {               message.error(response.data.message);           }
                } catch (error) {
                  dispatch(HideLoading());
                  message.error(error.message);       }    };
```

```
  const deleteUser = async () => {
    try {
      if (isButtonDisabled) {          return;          }
      dispatch(ShowLoading());
      setButtonDisabled(true);
      const response = await DeleteRegistration({match: match?._id});
      message.success(response.message);
      navigation("/")
      dispatch(HideLoading());
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);       }    };
  useEffect(() => {
    getMatchData();
  }, []);
       const isRegistered = match?.registeredUsers.find(userObj => userObj._id.toString() === user?._id.toString()) !== undefined;

  return (       <div>
    {match && (
     <Row className="mt-3" gutter={[30, 30]}>
      <Col lg={24} xs={24} sm={24}>
        <div className="flex flex-col gap-2 ">
          <h1 h1 className="text-small">
          Організатор: {match?.createdBy.login}
          </h1>
          <p className="text-small">
            Адреса проведення: {match?.place}
          </p>
          <p className="text-small">
            Вид командного спорту: {match?.type}
          </p>
          <p className="text-small">
            Дата проведення: {makeDate(match?.date)}
          </p>
          <p className="text-small">
            Час проведення: {match?.time}
          </p>
          <p className="text-small">
                Кількість зареєстрованих на матч користувачів: {match?.registeredUsers.length ? match?.registeredUsers.length : 0}/{match?.partisipants}
          </p>
```

```
                <p className="text-small">
                    {'Зареєстровані на матч користувачі: ' + match?.registeredUsers.map((user) => {return ("
"+user?.login)})}
                </p>
                <p className="text-small">
                  Контактні дані організатора: {match?.contacts}
                </p>
                <p className="text-small">
                  Опис: {match?.description}
                </p>
                </div>
              <hr />              { match?.createdBy._id === user._id ? " :
                isRegistered ? (
                <div>
                                                <button  className="third-btn"  onClick={()  =>  deleteUser()}
disabled={isButtonDisabled}>Скасувати реєстрацію</button>
                </div>
                          )  :  ( <div><button  className='secondary-btn'  onClick={()  =>  registrateUser()}
disabled={isButtonDisabled}>Зареєструватися</button>
                  </div>                )              }
          </Col>
        </Row>       )}
      </div>    )}
export default MatchInfo;
```

Файл matchRegistrationRoute.js

```
router.post("/", authMiddleware, async (req, res) => {
   try {
      req.body.user = req.body.userId;
      const checkMatch = await Match.findById(req.body.match);
      if (checkMatch.registeredUsers.includes(req.body.userId)) {
        return res.send({
          message: "Ви вже зареєстровані на цей матч",
          success: false,
          data: null,         });       }
else if(checkMatch.registeredUsers.length == checkMatch.partisipants) {
          return res.send({
            message: "Закінчились місця для реєстрації",
            success: false,
            data: null,
          });       }
      const newRegisterMatch = RegisterMatch(req.body);
```

```
        await newRegisterMatch.save();
        await Match.findOneAndUpdate({_id: req.body.match}, {
           $push: {registeredUsers: req.body.userId}        });
        res.status(200).send({ message: "Ви успішно зареєструвалися на цей матч", success: true });    }
    catch (error) {        res.status(500).send({ message: error.message, success: false });    }});
router.get("/get-registration-by-userId", authMiddleware, async (req, res) => {
  try {
     const matches = await RegisterMatch.find({user: req.body.userId}).populate({
       path: "match",
       populate: {
         path: "createdBy"        },      });
     res.status(200).send({ data: matches.map((match) => match.match), success: true });    }
   catch (error) {      res.status(500).send({ message: error.message, success: false });    }});
router.delete("/delete-registration", authMiddleware, async (req, res) => {
  try {
    await RegisterMatch.findOneAndDelete({ user: req.body.userId, match: req.body.match });
    await Match.findOneAndUpdate({_id: req.body.match},
     { $pull: { registeredUsers: req.body.userId } }    );
    res.status(200).send({ message: "Ви успішно скасували реєстрацію на цей матч", success: true });
  } catch (error) {    res.status(500).send({ success: false, message: error.message });  }});
router.get("/get-all-registration", authMiddleware, async (req, res) => {
  try {      const matches = await Match.find().populate('createdBy')
     res.status(200).send({ data: matches, success: true });    }
   catch (error) {      res.status(500).send({ success: false, message: error.message });    }});
module.exports = router;
```

Файл reviewRoute.js

```
const Review = require("../models/ratingModel");
const router = require("express").Router();
const User = require("../models/usersModel");
const authMiddleware = require("../middlewares/authMiddleware");
const mongoose = require("mongoose");
router.post("/", authMiddleware, async (req, res) => {
  try {    const newReview = new Review(req.body);
    await newReview.save();
    const userId = new mongoose.Types.ObjectId(req.body.user);
    const averageRating = await Review.aggregate([
      {       $match: {
         user: userId,       },     },
      {       $group: {         _id: "$user",         averageRating: { $avg: "$rating" }       },     },    ]);
    const averageRatingValue = averageRating[0]?.averageRating || 0;
```

```
    await User.findOneAndUpdate(userId, {
      rating: averageRatingValue
    });
    res.status(200).json({ message: "Відгук додано успішно", success: true });
  } catch (error) {    res.status(500).json({ message: error.message, success: false });  }});
router.get("/", async (req, res) => {
  try {
    const reviews = await Review.find(req.query || {}).sort({ createdAt: -1 }).populate("user");
    res.status(200).json({ data: reviews, success: true });
  } catch (error) {    res.status(500).json({ message: error.message, success: false });  }});
router.delete("/:id", authMiddleware, async (req, res) => {
  try {
    await Review.findByIdAndDelete(req.body._id);
    const userId = new mongoose.Types.ObjectId(req.body.user);
    const averageRating = await Review.aggregate([      {        $match: {          user: userId,        },      },
      {        $group: {          _id: "$user",          averageRating: { $avg: "$rating" }        },      },    ]);
    const averageRatingValue = averageRating[0]?.averageRating || 0;
    await User.findOneAndUpdate(userId, {      rating: averageRatingValue    });
    res.status(200).json({ message: "Відгук успішно видалено", success: true });  }
catch (error) {    res.status(500).json({ message: error.message, success: false });  }});
module.exports = router;
```

Файл reviewForm.js

```
function ReviewForm({user, reloadData, showReviewForm, setShowReviewForm, selectedReview}) {
    const dispatch = useDispatch();
    const [rating, setRating] = useState();
    const [comment, setComment] = useState("");
    const [isButtonDisabled, setButtonDisabled] = useState(false);
    const addReview = async () => {
      try {          if (isButtonDisabled) {            return;          }
        dispatch(ShowLoading(true));
        setButtonDisabled(true);
        const response = await AddNewReview({user: user._id, rating, comment});
        message.success(response.message);
        reloadData();
        setShowReviewForm(false);
        document.location.reload()
        dispatch(HideLoading(false));
      } catch (error) {          dispatch(HideLoading(false));          message.error('Спочатку поставте
рейтинг!');      }    };
    return (
```

```
          <Modal
            open={showReviewForm}
            onCancel={() => setShowReviewForm(false)}
            centered
            title='Додати відгук'
            onOk={addReview} disabled={isButtonDisabled}>
            <div className="flex flex-col gap-2 w-full">
              <div className="flex w-full">
                <span className="font-semibold">Користувач : </span>
                <span className="ml-2 font-semibold">{user?.login}</span>
              </div>
              <Rate
                value={rating}
                onChange={(value) => setRating(value)}
                allowHalf
                style={{ color: "orange" }}
              />
              <textarea
                value={comment}
                onChange={(e) => setComment(e.target.value.substring(0, 200))}
                placeholder="Ваш відгук"
                cols="30"
                rows="10"
              ></textarea>
            </div>
          </Modal>      );}
export default ReviewForm;
```

Файл usersInfo.js

```
function UserInfo() {
    const dispatch = useDispatch();
    const [currentPage, setCurrentPage] = useState(1);
    const [pageSize, setPageSize] = useState(10);
    const [showReviewForm, setShowReviewForm] = useState(false);
    const [reviewes, setReviews] = useState([]);
    const [users, setUsers] = useState();
    const { id } = useParams();
    const { user } = useSelector(state => state.users)
    const getReview = async () => {
        try {
          dispatch(ShowLoading(true));
```

```
      const response = await GetUserById(id);
      const reviewsResponse = await GetAllReviews({user: id});
      setReviews(reviewsResponse.data);
      setUsers(response.data);
      dispatch(HideLoading(false));
    } catch (error) {          dispatch(HideLoading(false));
      message.error(error.message);        }    };
  const deleteReview =  async (review) => {
    try {
      dispatch(ShowLoading(true));
      const reviewsDeleteResponse = await DeleteReview({_id: review._id, user: review.user._id});
      message.success(reviewsDeleteResponse.message);
      document.location.reload()
      getReview();
      dispatch(HideLoading(false));
    } catch (error) {            dispatch(HideLoading(false));
      message.error(error.message);        }    };
  useEffect(() => {
    getReview();
  }, []);
  const handlePageChange = (page, pageSize) => {
    setCurrentPage(page);      };
     const handlePageSizeChange = (current, size) => {
    setCurrentPage(1);
    setPageSize(size);    };
  const startIndex = (currentPage - 1) * pageSize;
  const endIndex = startIndex + pageSize;
  const paginatedUsers = reviewes.slice(startIndex, endIndex);
  return (
    users && (
      <div>
        <div className="flex justify-between items-center mt-5">
          <span className="text-big flex-grow-0">Відгуки про користувача: {users.login}

          </span>
          {  id === user._id ? " :
            <button className='fourth-btn' onClick={() => setShowReviewForm(true)}>
            Додати відгук
            </button>                  }
        </div>
                    <div className='rate-text'> Загальний рейтинг:
          <Rate className="value"
```

```
                    disabled
                    defaultValue={users.rating || 0}
                    allowHalf
                    style={{ color: "orange" }}
                  />
                </div>
                <div className="mt-5 flex flex-col gap-2">
                  {paginatedUsers.map((review) => {
                    return (
                    <div
                      key={review?._id}
                      className="flex justify-between border-solid border p-2">
                      <div className="flex flex-col">
                      <Rate
                        disabled
                        defaultValue={review?.rating || 0}
                        allowHalf
                        style={{ color: "orange" }}
                        className="mt-2"                    />
                      <span className="text-sm">
                        {review?.comment}
                      </span>
                      </div>
                                          {user.isAdmin  ?  <i  class="ri-delete-bin-6-fill"  onClick={()  =>
{deleteReview(review)}}></i> : ''}
                                    </div>                );                })}
                </div>
                        {showReviewForm  &&  <ReviewForm  user={users}  reloadData={getReview}
showReviewForm={showReviewForm} setShowReviewForm={setShowReviewForm}/>}
                <Pagination className='pagination'
                current={currentPage}
                pageSize={pageSize}
                total={reviewes.length}
                onChange={handlePageChange}
                onShowSizeChange={handlePageSizeChange}
                showSizeChanger
                pageSizeOptions={['10', '20', '30', '40']}/>
              </div>       )  )}
export default UserInfo;
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ОРГАНІЗАТОРІВ КОМАНДНИХ АМАТОРСЬКИХ ІГОР

Програма та методика тестування

КПІ.ІТ-9419.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
Олександр СТЕЛЬМАХ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Андрій ПОДМОКОВ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є веб-застосунок для підтримки діяльності організаторів командних аматорських ігор.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише людина, що тестує програмне забезпечення.

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.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ОРГАНІЗАТОРІВ КОМАНДНИХ АМАТОРСЬКИХ ІГОР

Керівництво користувача

КПІ.ІТ-9419.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
Олександр СТЕЛЬМАХ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Андрій ПОДМОКОВ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Comand Match Finder» - це веб-застосунок для організації командних аматорських ігор. Він надає можливість користувачам знаходити та реєструватися на існуючі матчі, а також створювати власні ігри. Застосунок також дозволяє користувачам оцінювати та коментувати інших користувачів.

Кінцева збірка програмного забезпечення включає в себе серверну частину, написану на мові програмування Node.js з використанням фреймворка Express.js, а також клієнтську частину, розроблену з використанням бібліотеки React.js. Веб-застосунок використовує базу даних MongoDB для зберігання інформації про користувачів, матчі, реєстрації на матчі, відгуки користувачів.

Інсталяційна версія програмного забезпечення буде доступна через репозиторій GitHub. У репозиторії містяться всі вихідні коди програми. Користувачі можуть скопіювати репозиторій і запустити програму на своєму локальному середовищі для тестування або власного використання.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного веб-застосунку необхідне виконання наступних вимог:

Веб-браузер: Google Chrome, Mozilla Firefox, Safari або Microsoft Edge останньої версії.

Мінімальні системні вимоги:

– операційна система: Windows 7 або вище, MacOS 10.12 Sierra або вище, Linux (будь-який сучасний дистрибутив, що підтримує браузери останньої версії);

– процесор: 1 GHz або швидше;

– об‘єм ОЗП: щонайменше 1 GB;

– жорсткий диск: щонайменше 1 GB вільного місця для зберігання кешу браузера та інших тимчасових файлів;

– відеокарта: будь-яка сумісна з операційною системою;

– дисплей: рекомендована роздільна здатність 1024x768 або вище;

– інтернет з'єднання: широкосмугове.

Рекомендовані системні вимоги:

– операційна система: Windows 10;

– процесор: Intel Core i5;

– об‘єм ОЗП: 16 ГБ;

– жорсткий диск: щонайменше 2 GB вільного місця для зберігання кешу браузера та інших тимчасових файлів;

– відеокарта: NVIDIA GeForce GTX 1050 Ti 4GB;

– дисплей: роздільна здатність 1920x1080;

– інтернет з'єднання: широкосмугове, зі стабільною швидкістю для комфортної взаємодії з веб-застосунком.

2.2 Завантаження застосунку

Щоб отримати доступ до веб-застосунку, необхідно перейти за наступним посиланням: https://master--clinquant-torte-9f5d6e.netlify.app/ .

Для отримання доступу до інсталяційної версії, необхідно перейти на GitHub репозиторій веб-застосунку, натиснути на кнопку “<Code>” і в розгорнутому списку копіювати посилання на репозиторій для його клонування. На рисунку 2.1 зображено GitHub репозиторій веб-застосунку.

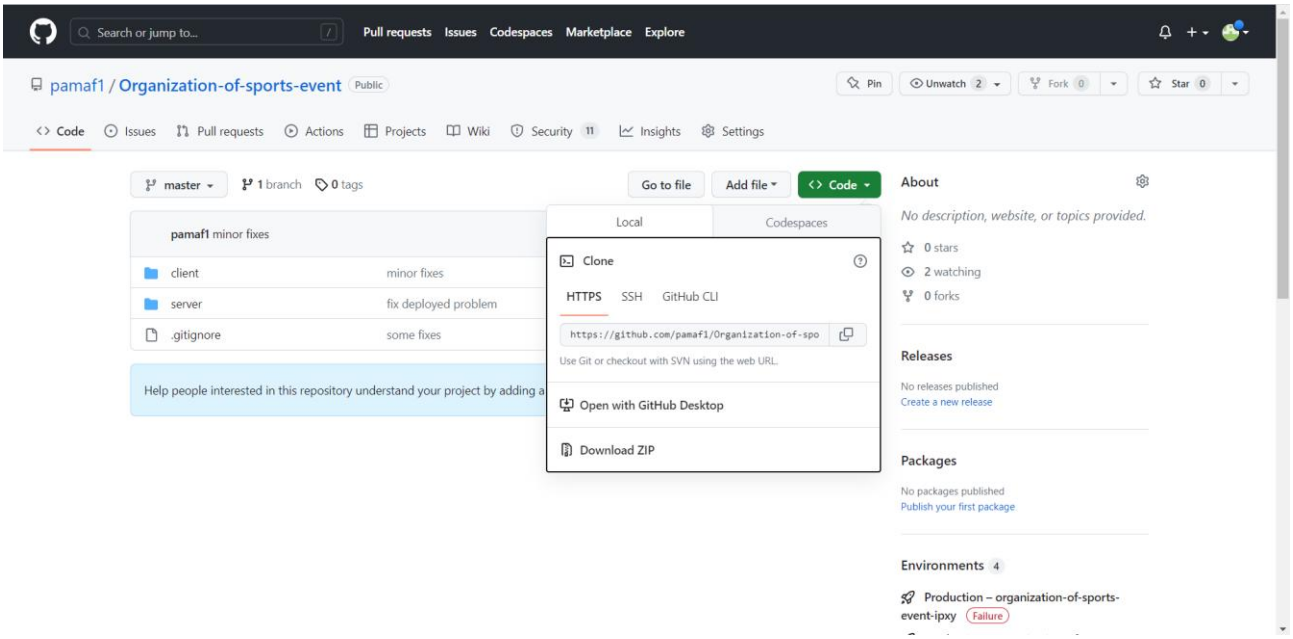


Рисунок 2.1 – GitHub репозиторій веб-застосунку

Для клонування репозиторію необхідно обрати директорію, в яку користувач бажає зберегти репозиторій і в bash терміналі ввести команду “git clone https://”. Після того як клонування пройшло успішно, необхідно завантажити всі необхідні залежності за допомогою наступних команд: спочатку завантажуються залежності для серверної частини “cd server” і “npm install”, потім для клієнтської частини “cd ../”, “cd client” і “npm install”. Окрім цього, для запуску клієнтської частини потрібно замінити посилання для зв'язку з сервером з розгорнутого на локальні посилання, які вказані у файлі server.js серверної частини. Також у директорії серверу необхідно створити файл .env, в якому вказати посилання на підключення до MongoDB

і секретний ключ JWT токену. Тепер можна запускати сервер і клієнт. Сервер запускається у директорії серверу за допомогою команди “nodemon server”, а клієнт у директорії клієнту за допомогою команди “npm start”.

2.3 Перевірка коректної роботи

Для перевірки роботи веб-застосунку необхідно перейти за посиланням: https://master--clinquant-torte-9f5d6e.netlify.app/ . Якщо все успішно працює, за цим посиланням відкриється сторінка авторизації веб-застосунку.

Для перевірки інсталяційної версії веб-застосунку необхідно перейти за посиланням: http://localhost:3000/ , де “3000” це порт, на якому запускається веб-застосунок. При успішній інсталяції, за цим посиланням відкриється сторінка авторизації веб-застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

При відкриті веб-застосунку, відкривається сторінка авторизації. Якщо у користувача вже є аккаунт, він авторизується, якщо аккаунту немає, необхідно пройти реєстрацію. На рисунку 3.1 наведено реєстрацію користувача.

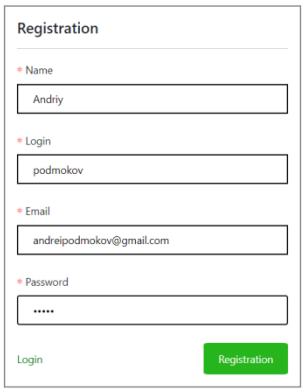


Рисунок 3.1 – Реєстрація нового користувача

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішну реєстрацію, і користувача буде перенаправлено на сторінку авторизації.

Далі необхідно авторизуватись, для цього у відповідні поля вводяться дані, які були використані при реєстрації. На рисунку 3.2 зображено авторизацію користувача.

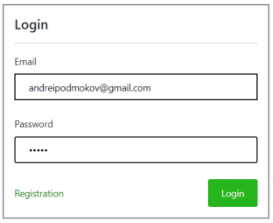


Рисунок 3.2 – Авторизація користувача

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішну авторизацію, і користувача буде перенаправлено на головну сторінку. На рисунку 3.3 зображено успішну авторизацію користувача.

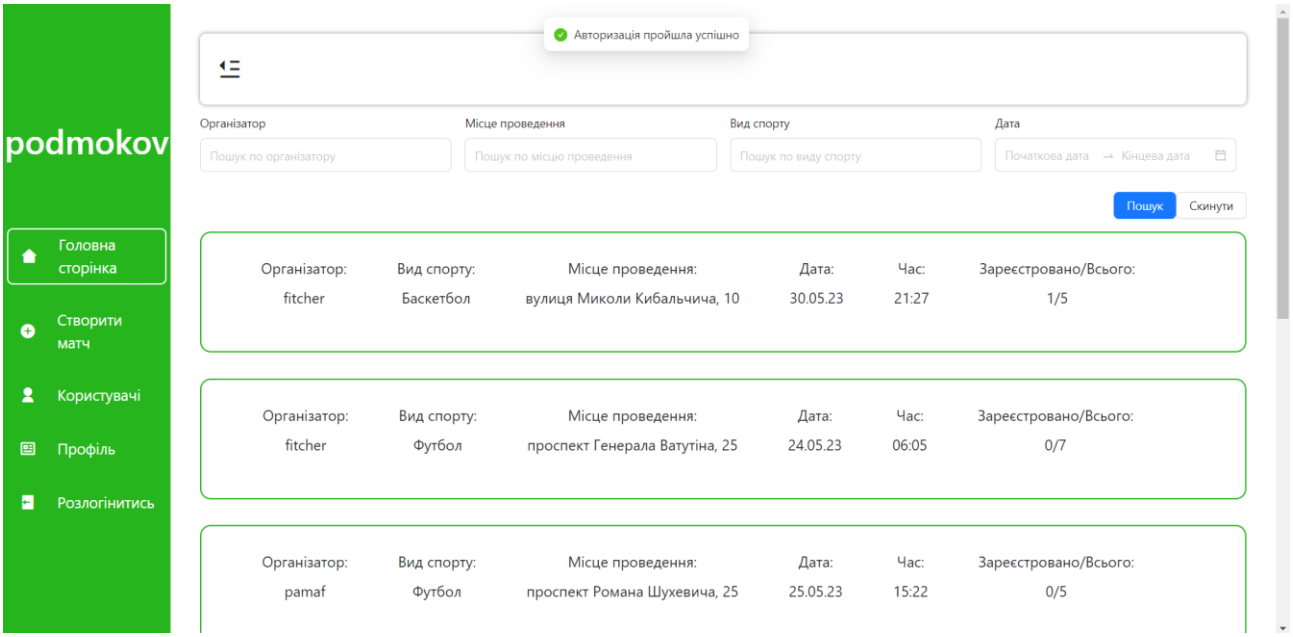


Рисунок 3.3 – Успішна авторизація користувача

Після авторизації, користувачі розділяються на дві ролі: адміністратор і звичайний користувач. Спочатку розглянемо їх спільні функції.

Для початку створимо новий матч. Для цього необхідно перейти на сторінку “Створити матч”, на цій сторінці знаходяться створені користувачем матчі, а у випадку адміністратора матчі створені всіма користувачами, також

можливість їх редагування та видалення й кнопка для створення матчу. На рисунку 3.4 зображена сторінка “Створити матч”.

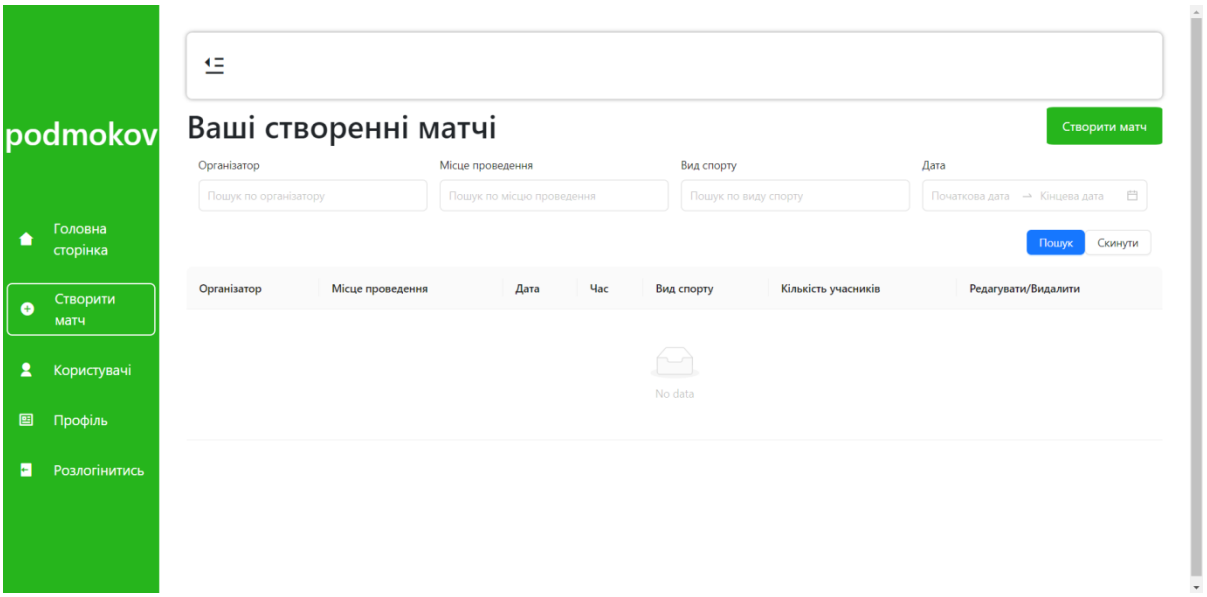


Рисунок 3.4 – Сторінка “Створити матч”

Для створення матчу необхідно натиснути на кнопку “Створити матч”. Після натискання на неї з'явиться модальне вікно, в яке необхідно ввести дані про матч. На рисунку 3.5 зображено процес створення матчу.

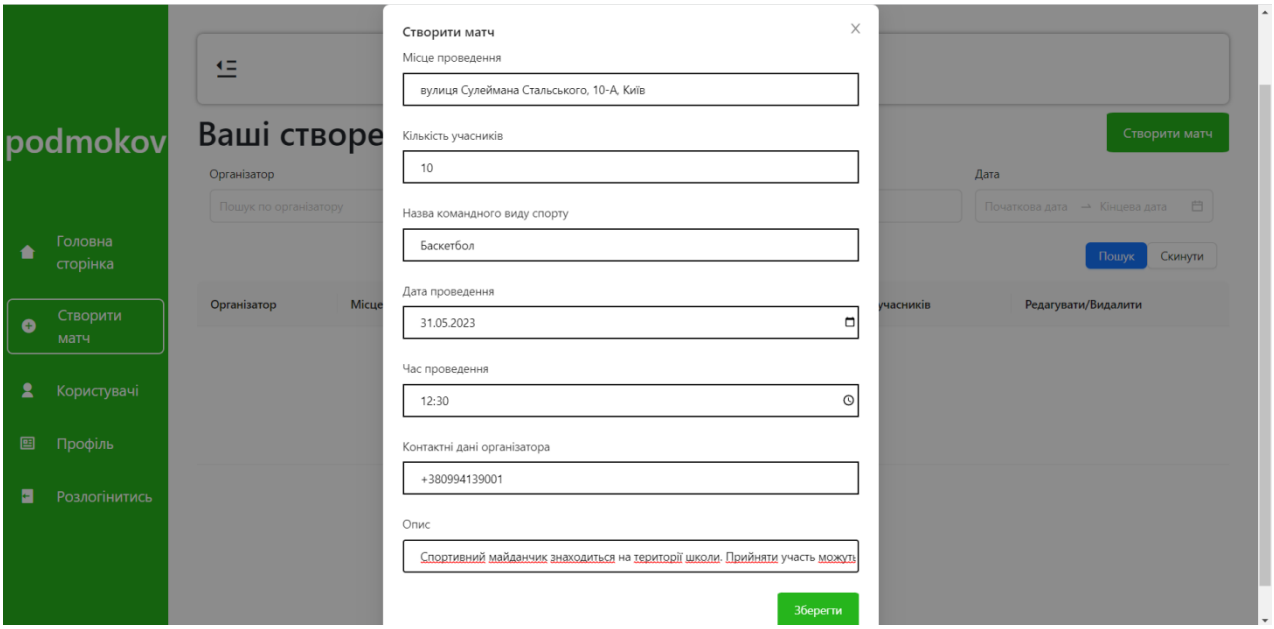


Рисунок 3.5 – Створення матчу

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішне створення матчу, і на сторінці з’явиться щойно створений матч. На рисунку 3.6 зображено успішне створення матчу.

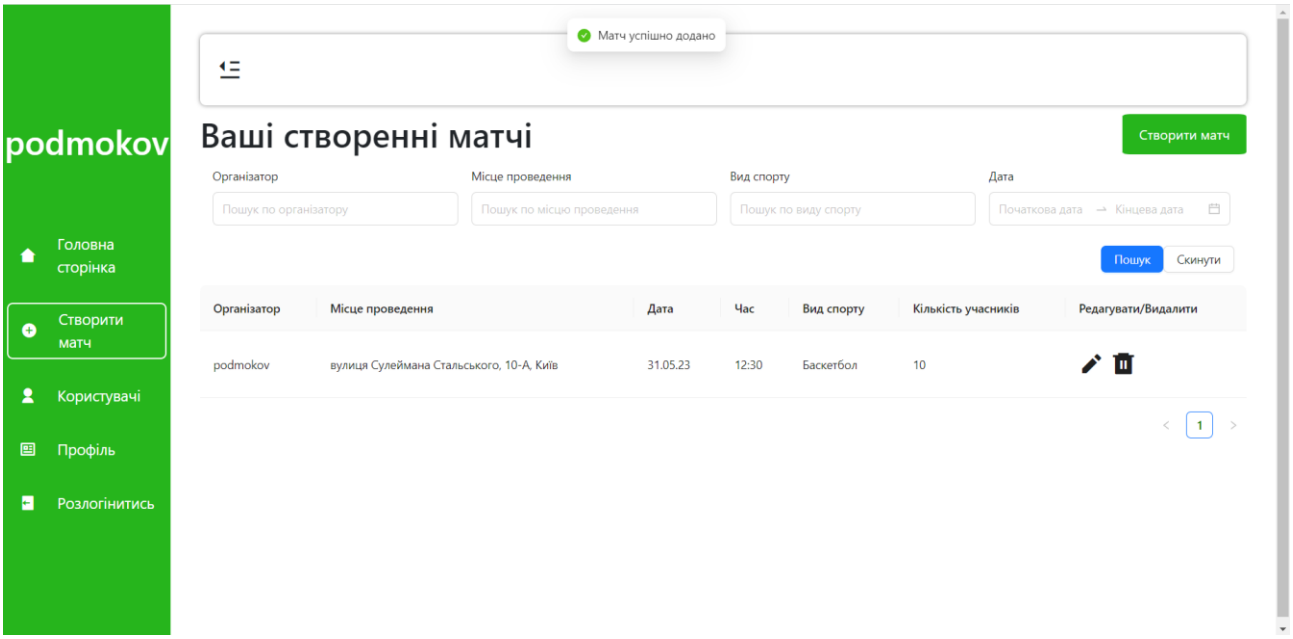


Рисунок 3.6 – Успішне створення матчу

Для редагування матчу необхідно натиснути на кнопку редагування. Після натискання на неї з'явиться модальне вікно, в якому необхідно ввести нові дані про матч. На рисунку 3.7 зображено процес редагування матчу.

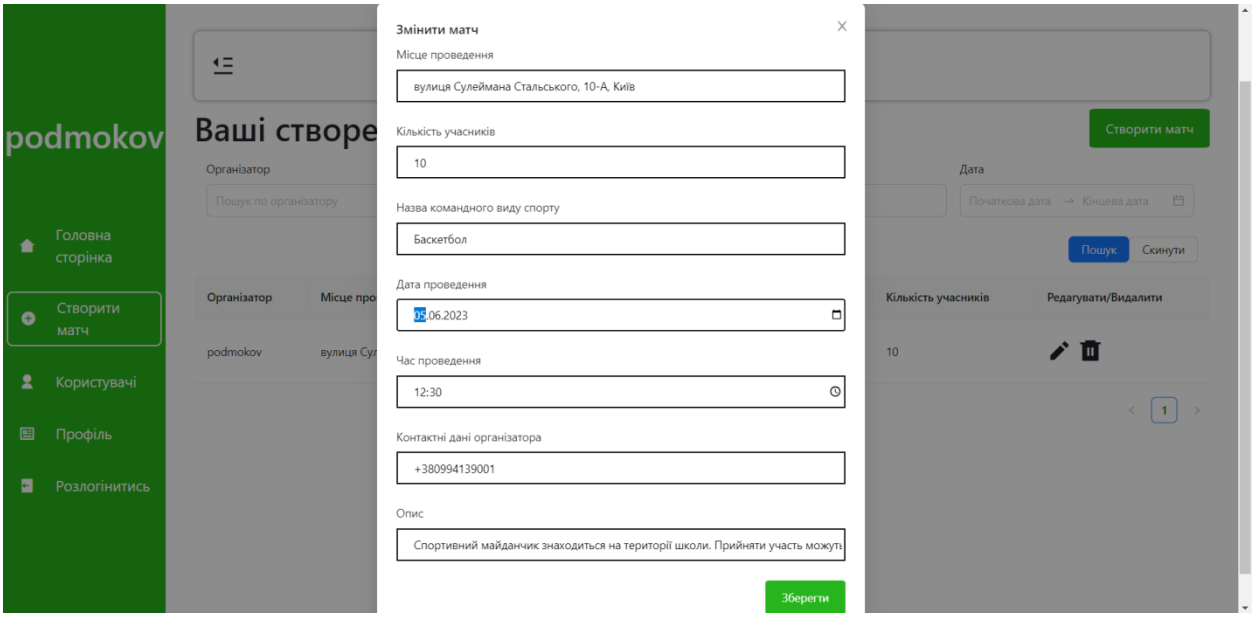


Рисунок 3.7 – Редагування матчу

Якщо всі дані було введено коректно, то з’явиться повідомлення про успішне редагування матчу, і на сторінці оновляться дані про матч.

Для видалення матчу необхідно натиснути на кнопку видалення. На рисунку 3.8 зображено успішне видалення матчу.

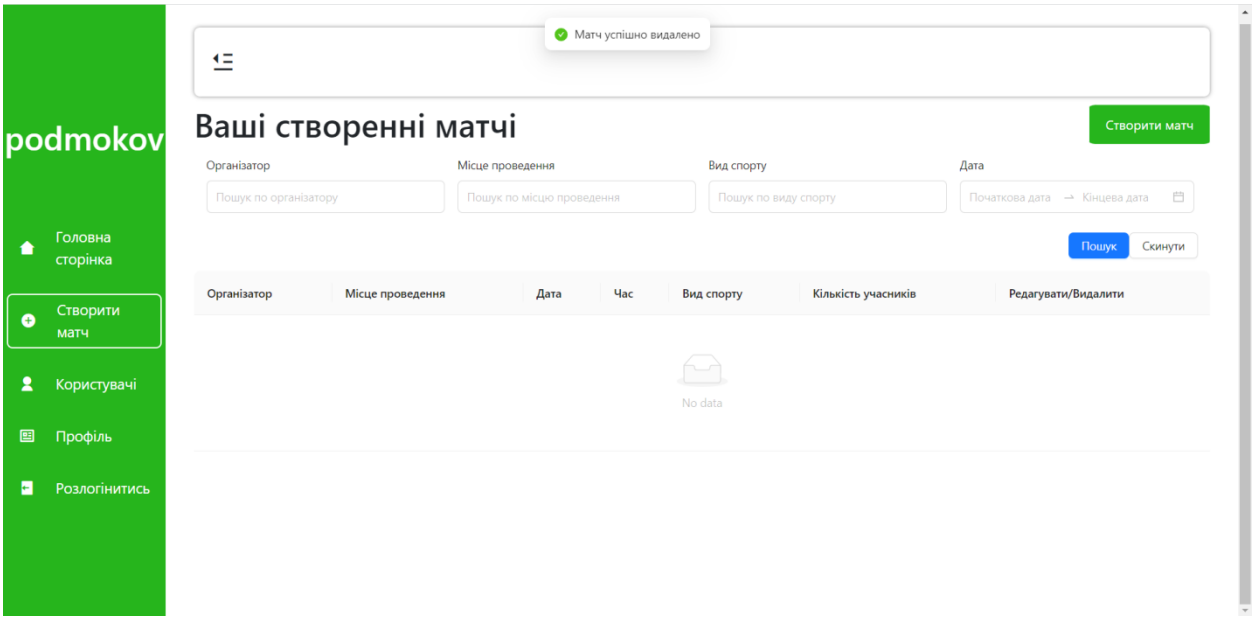


Рисунок 3.8 – Успішне видалення матчу

Для того щоб переглянути створений матч, необхідно перейти на головну сторінку і знайти щойно створений матч у списку усіх матчів. Це можна зробити використовуючи пошук по сайту, або самостійно знайти матч на сторінці. На рисунку 3.9 зображено пошук матчу за організатором.

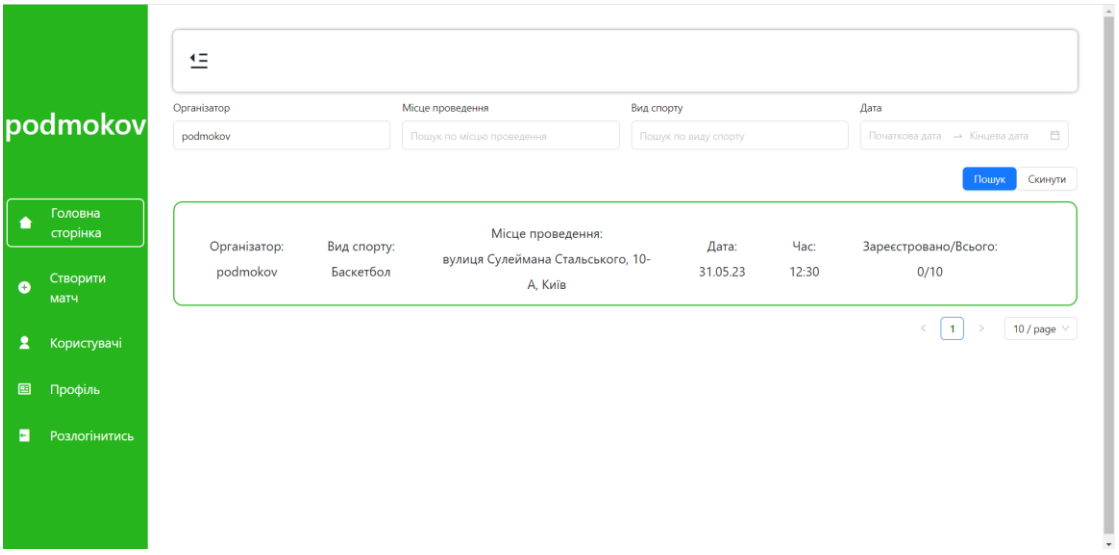


Рисунок 3.9 – Пошук матчу за організатором

Після того як матч було знайдено, можна перейти на сторінку матчу, та передивитись повну інформацію про нього. Тепер всі користувачі, що побачать створений матч, зможуть зареєструватись на нього, щоб прийняти у ньому участь. На рисунку 3.10 зображено сторінку матчу.

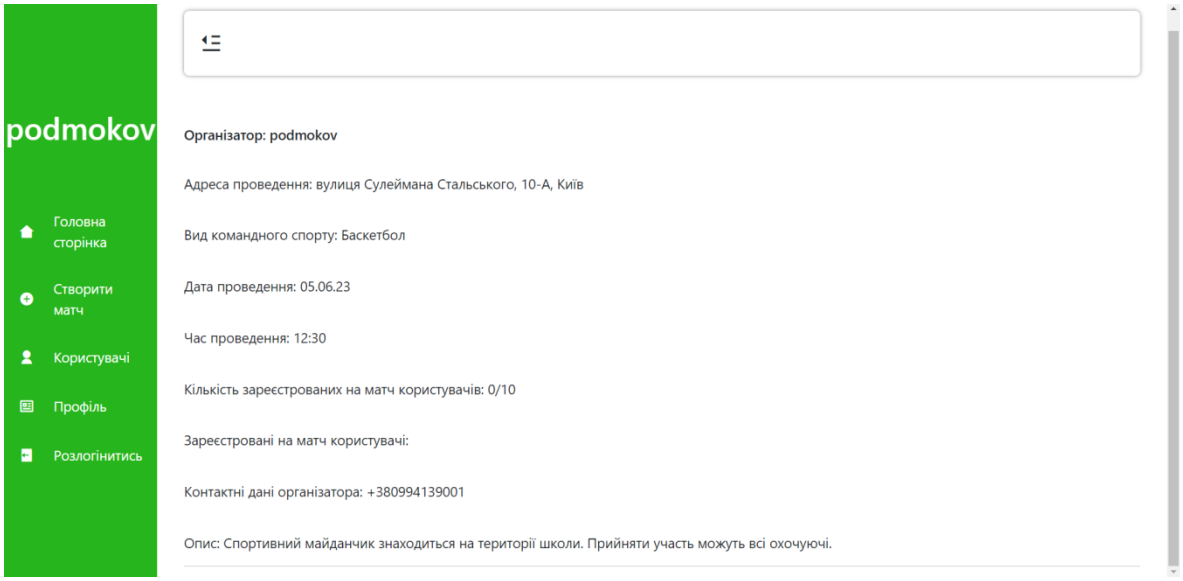


Рисунок 3.10 – Сторінка матчу

Для реєстрації на матч необхідно обрати матч, на який є бажання зареєструватись, та перейти на його сторінку. На сторінці матчу потрібно натиснути на кнопку “Зареєструватися”. На рисунку 3.11 зображено реєстрацію на матч.

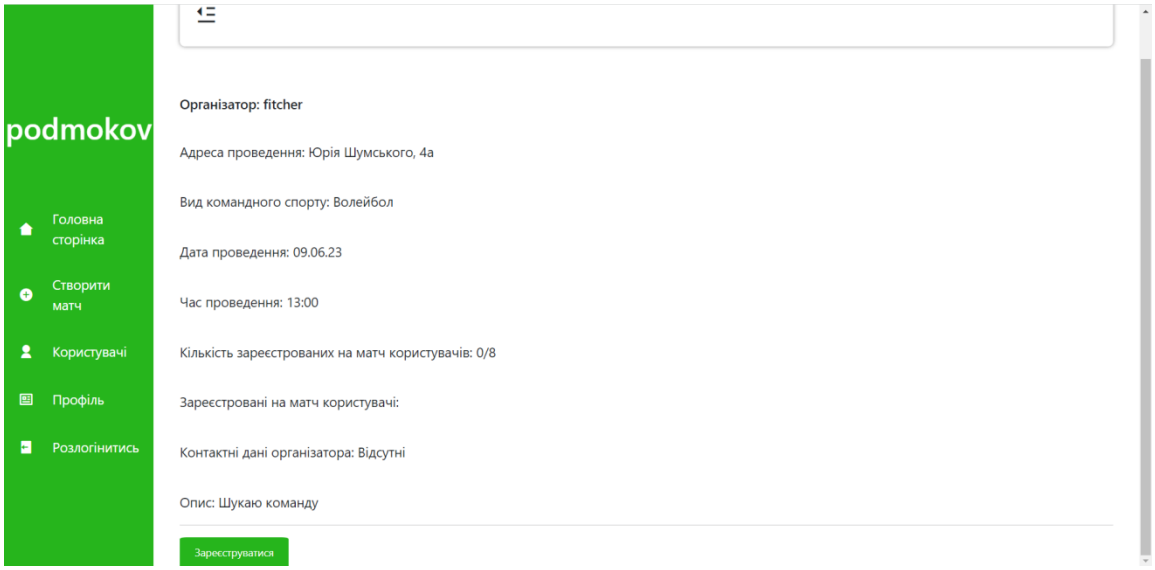


Рисунок 3.11 – Реєстрація на матч

Після натискання на кнопку “Зареєструватися” користувача буде зареєстровано на матч, перенаправлено на сторінку “Профіль” та він отримає повідомлення про успішну реєстрацію. На рисунку 3.12 зображено успішну реєстрацію на матч.

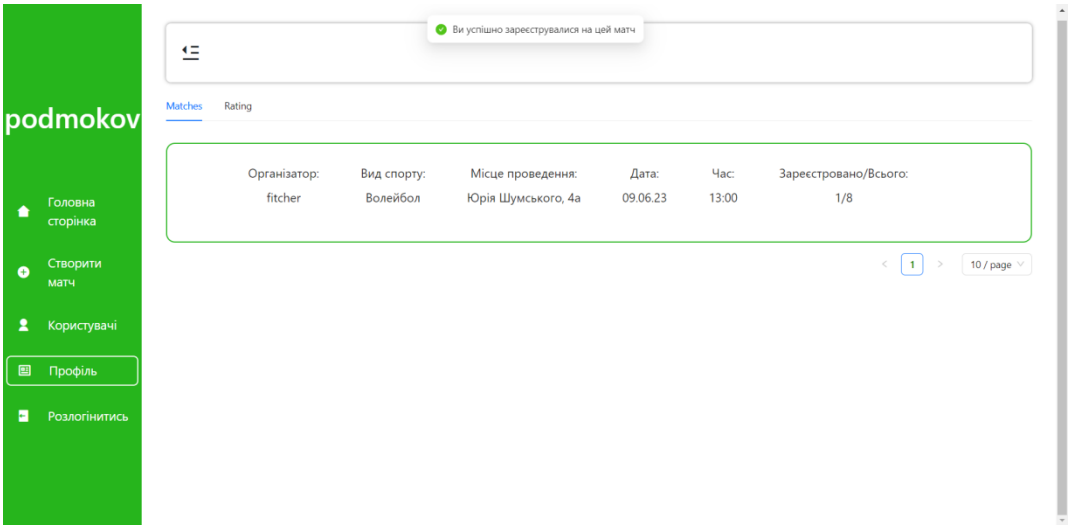


Рисунок 3.12 – Успішна реєстрація на матч

Для скасування реєстрації на матч необхідно на сторінці профілю, або головній сторінці знайти матч на який треба скасувати реєстрацію, перейти на його сторінку та натиснути на кнопку “Скасувати реєстрацію”. На рисунку 3.13 зображено скасування реєстрації на матч.

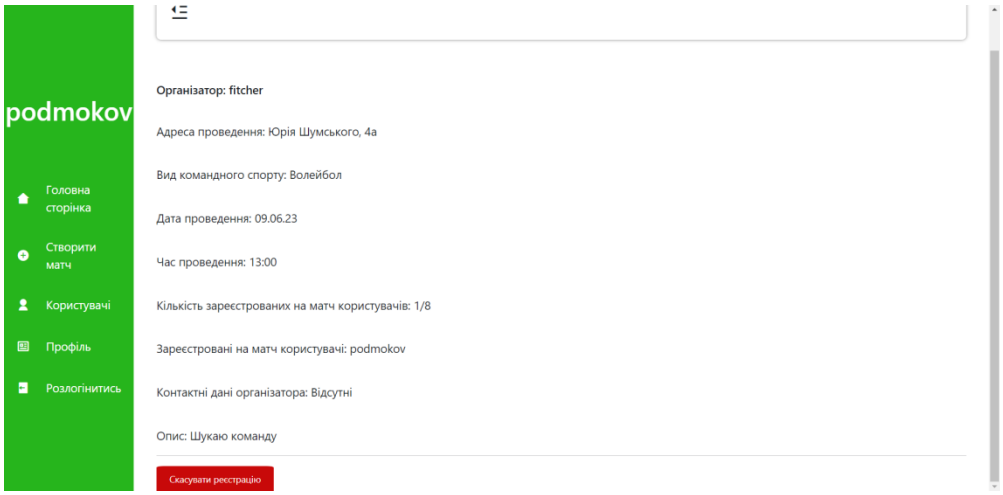


Рисунок 3.13 – Скасування реєстрації на матч

Після натискання на кнопку “Скасувати реєстрацію”, користувач відмінить реєстрацію на матч та буде перенаправлений на головну сторінку. На рисунку 3.14 зображено успішне скасування реєстрації на матч.

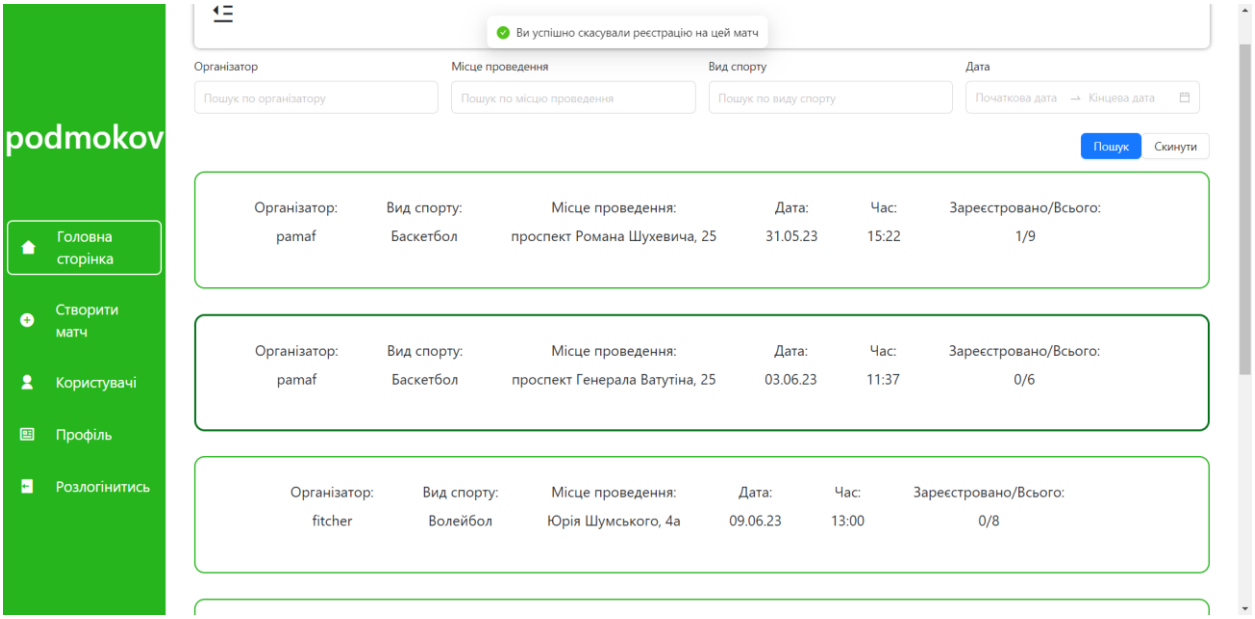


Рисунок 3.14 – Успішне скасування реєстрації на матч

Для того щоб переглянути свій рейтинг, необхідно перейти на сторінку профілю й відкрити вкладку “Rating”. На рисунку 3.15 зображено сторінку з рейтингом користувача.

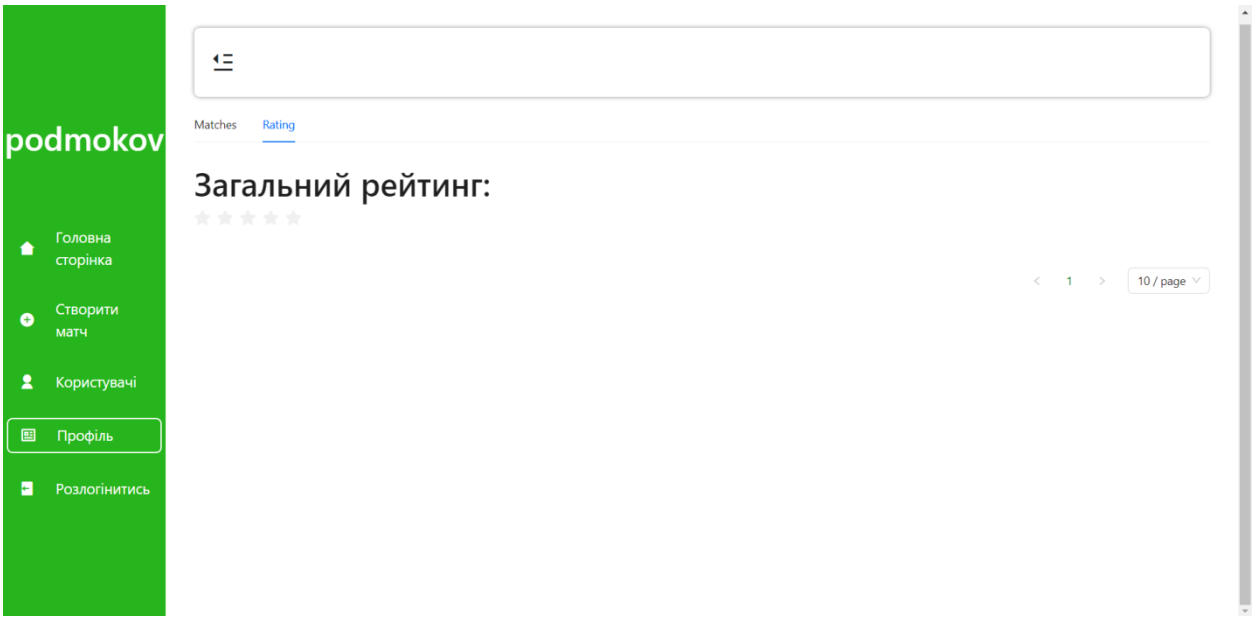


Рисунок 3.15 – Сторінка рейтингу користувача.

Для того щоб переглянути список користувачів, необхідно перейти на сторінку “Користувачі”. На рисунку 3.16 зображено сторінку “Користувачі” для звичайного користувача.

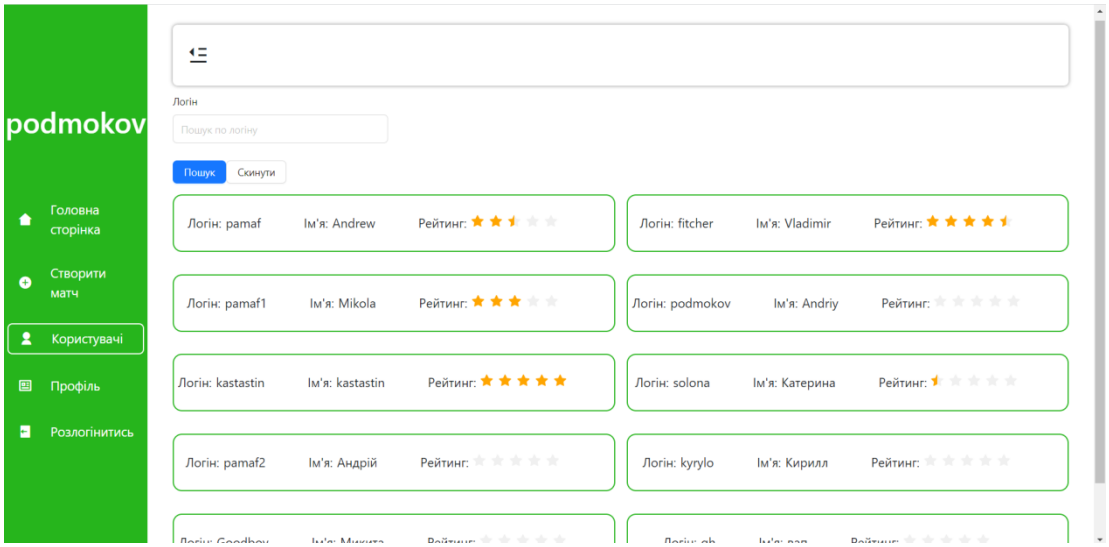


Рисунок 3.16 – Сторінка “Користувачі” для звичайного користувача

На цій сторінці звичайний користувач може знайти іншого користувача за логіком, або просто знайти бажаного користувача у списку. Натиснувши на потрібного користувача, відбудеться пере направлення на сторінку цього користувача. На рисунку 3.17 зображено сторінку користувача для звичайного користувача.

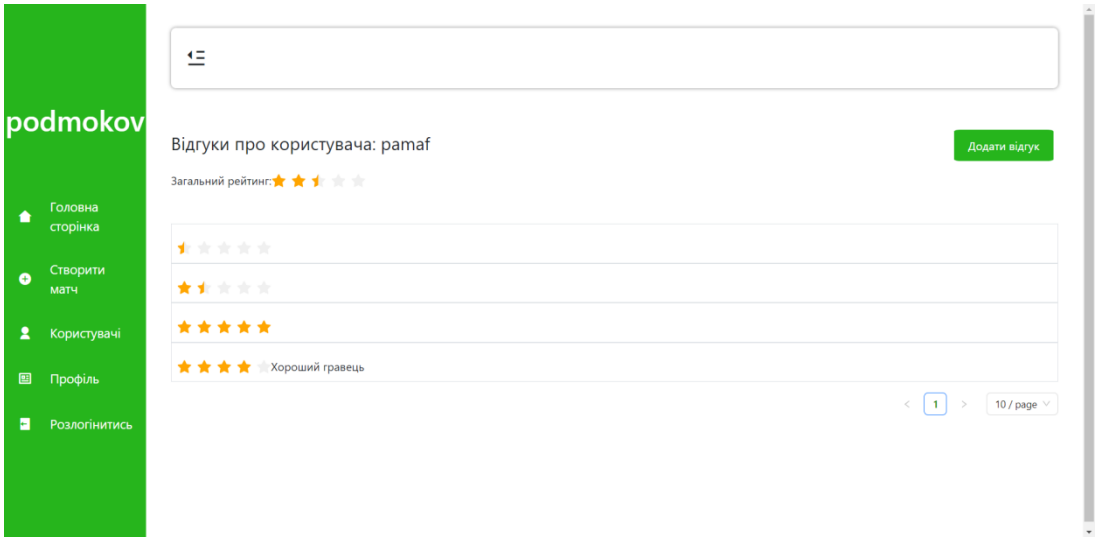


Рисунок 3.17 – Сторінка користувача для звичайного користувача

На цій сторінці звичайний користувач бачить відгуки й рейтинг користувача, а також може додати новий відгук.

Для того щоб додати новий відгук необхідно натиснути на кнопку “Додати відгук”. Після натискання на неї з'явиться модальне вікно, в якому необхідно залишити відгук. На рисунку 3.18 зображено процес додавання відгуку.

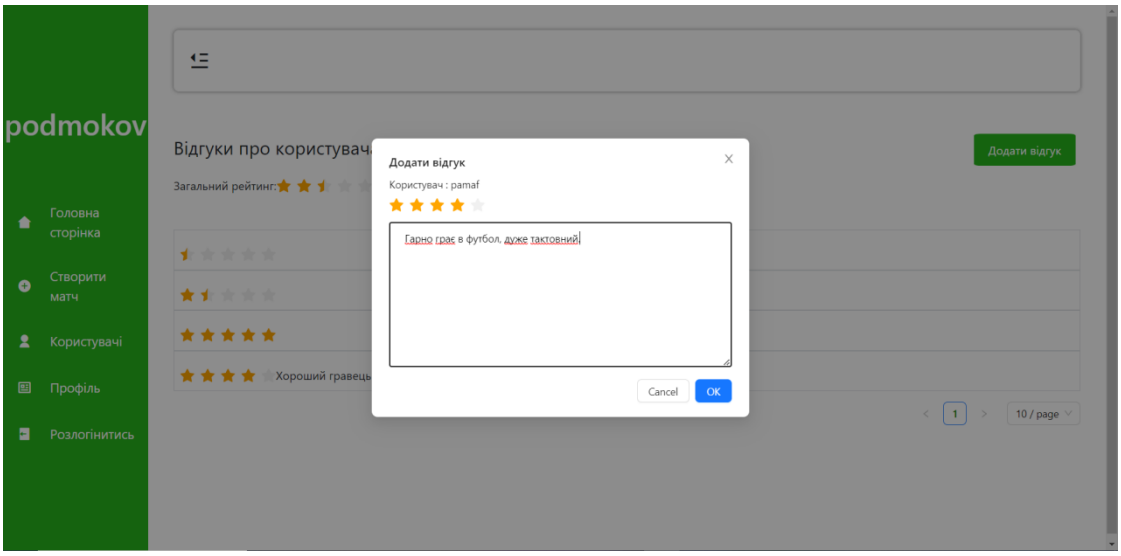


Рисунок 3.18 – Додавання відгуку

Після введення всієї необхідної інформацію, відгук буде додано, відобразиться повідомлення про успішне додавання відгуку і сторінку буде оновлено. На рисунку 3.19 зображено успішне додавання відгуку.

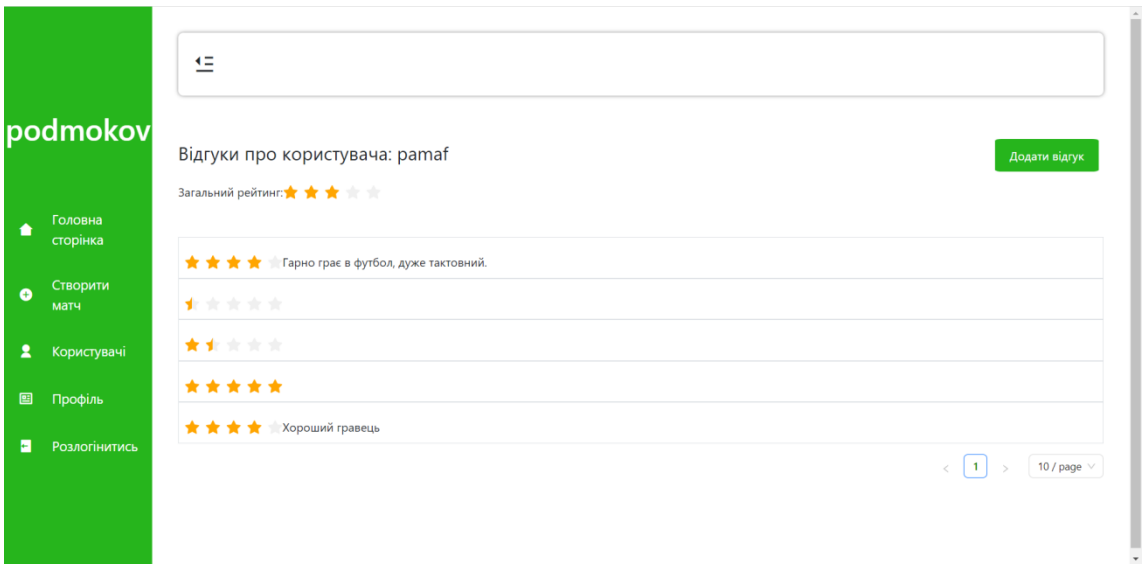


Рисунок 3.19 – Успішне додавання відгуку

У випадку адміністратора, сторінка “Користувачі” виглядає інакше. На рисунку 3.20 зображено сторінку “Користувачі” для адміністратора.

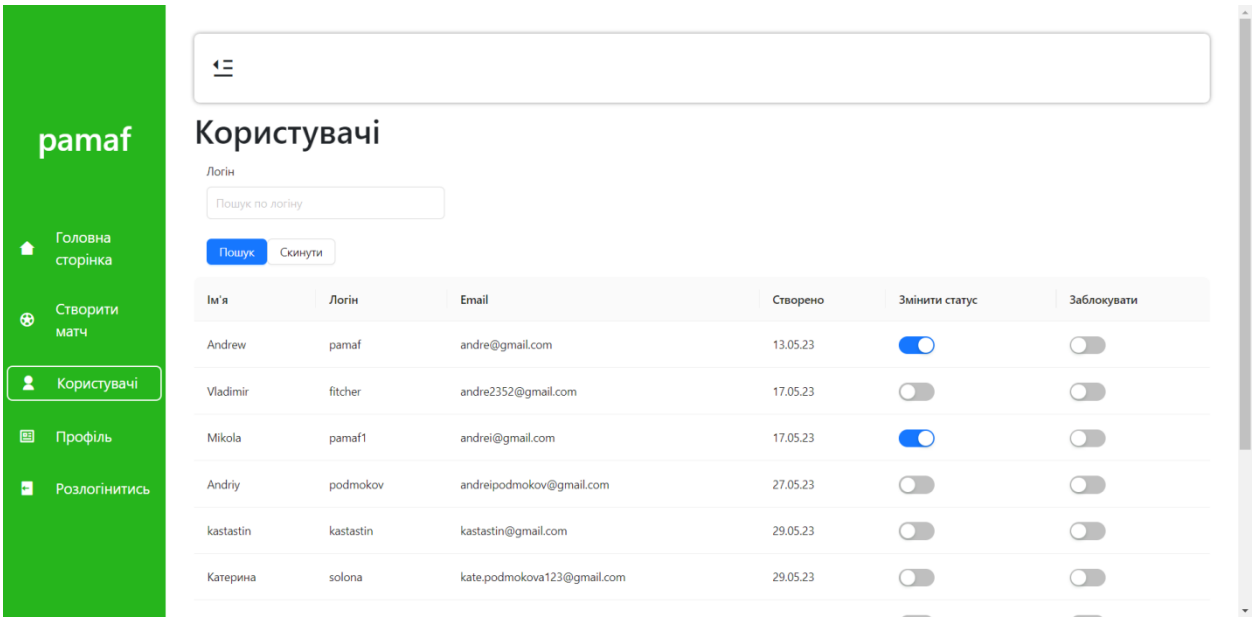


Рисунок 3.20 – Сторінка “Користувачі” для адміністратора

Адміністратор може надавати права адміністратора іншим користувачам, а також блокувати їх. Для цього йому потрібно натиснути на відповідні кнопки. На рисунку 3.21 зображено надавання прав адміністратора.

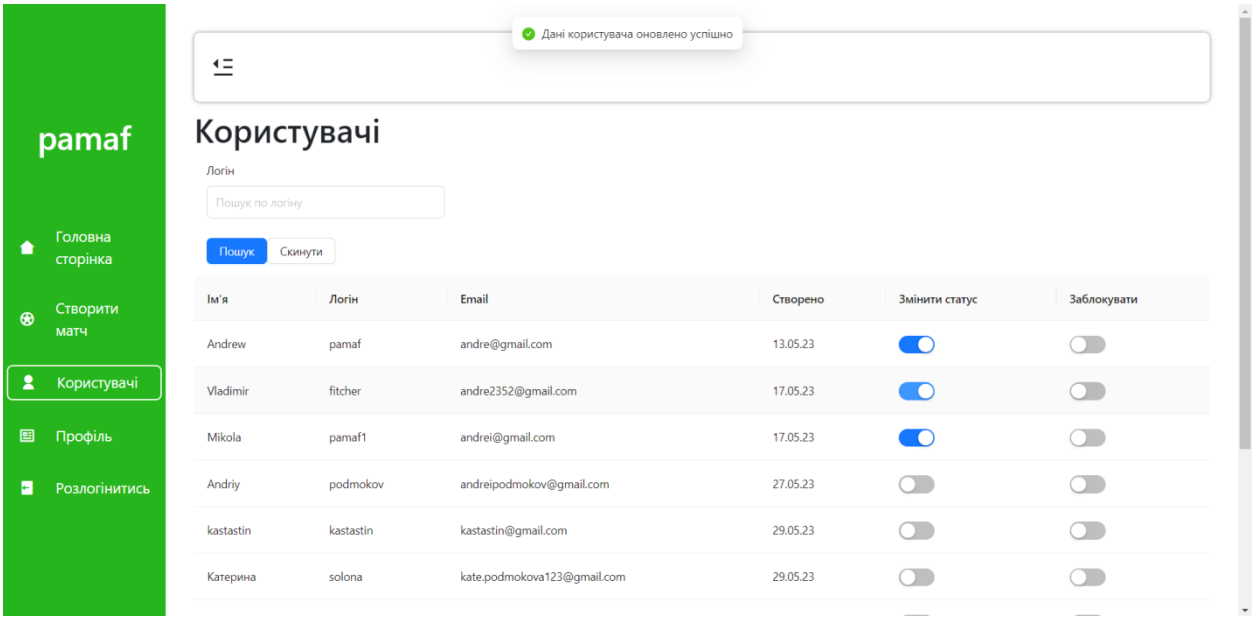


Рисунок 3.21 – Надавання прав адміністратора

Для того, щоб перейти на сторінку користувача, адміністратору необхідно натиснути на логін бажаного користувача. Сторінка користувача для адміністратора виглядає майже так само, як і для звичайного користувача, окрім того, що адміністратор може видаляти відгуки. На рисунку 3.22 зображено сторінку користувача для адміністратора.

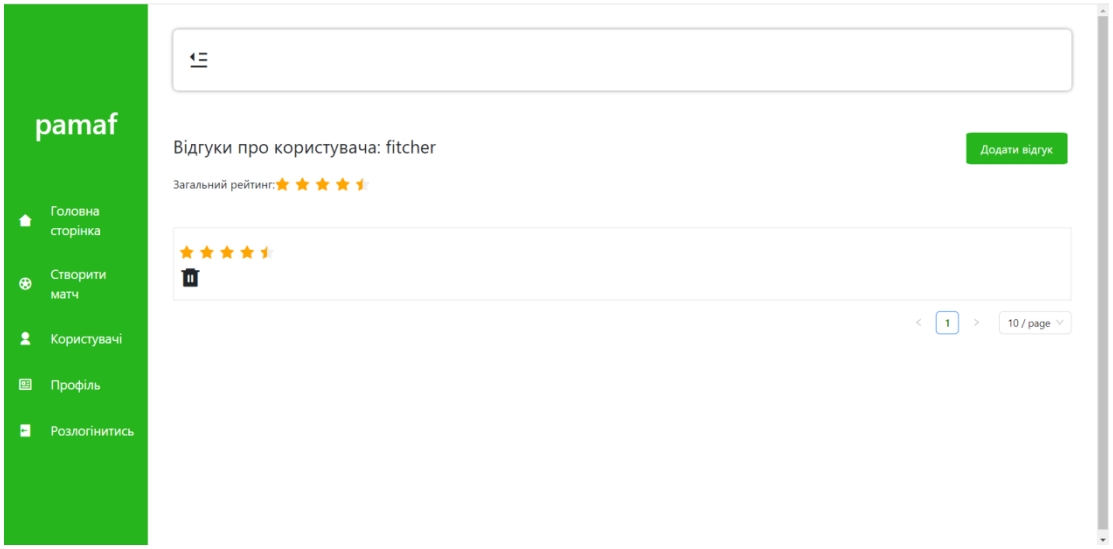


Рисунок 3.22 – Сторінка користувача для адміністратора

Для того, щоб видалити відгук, необхідно натиснути на кнопку видалення. Після чого з’явиться повідомлення про успішне видалення відгуку й сторінку буде оновлено. На рисунку 3.23 зображено успішне видалення відгуку.

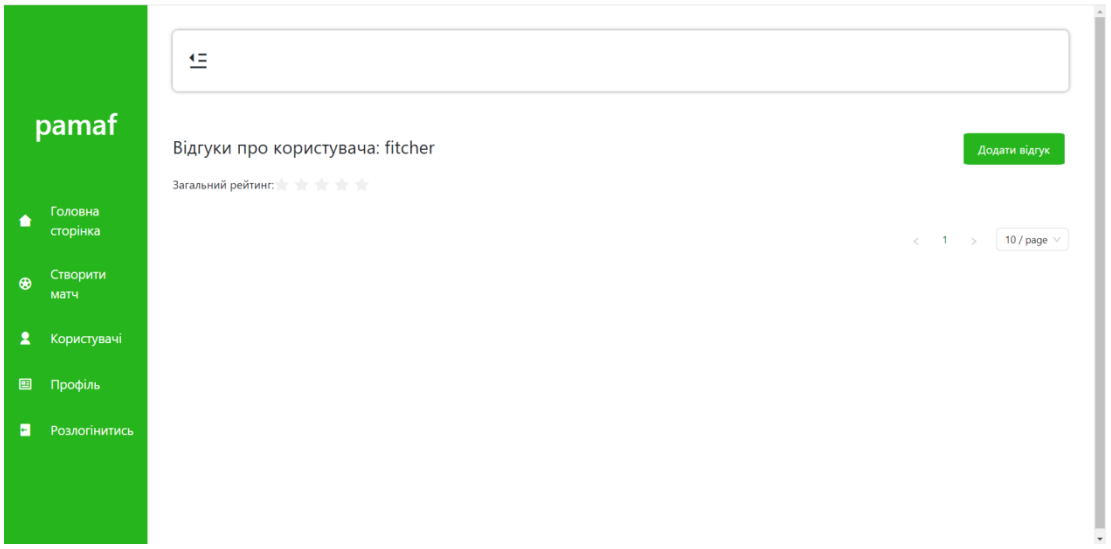


Рисунок 3.23 – Успішне видалення відгуку

Для того щоб розлогінитись, потрібно натиснути на “Розлогінитись” у лівому меню, після чого користувача буде перенаправлено на сторінку авторизації. Для того щоб зменшити ліве меню, необхідно натиснути на іконку закриття меню у верхній частині екрану. На рисунку 3.24 зображено головну сторінку з закритим меню.

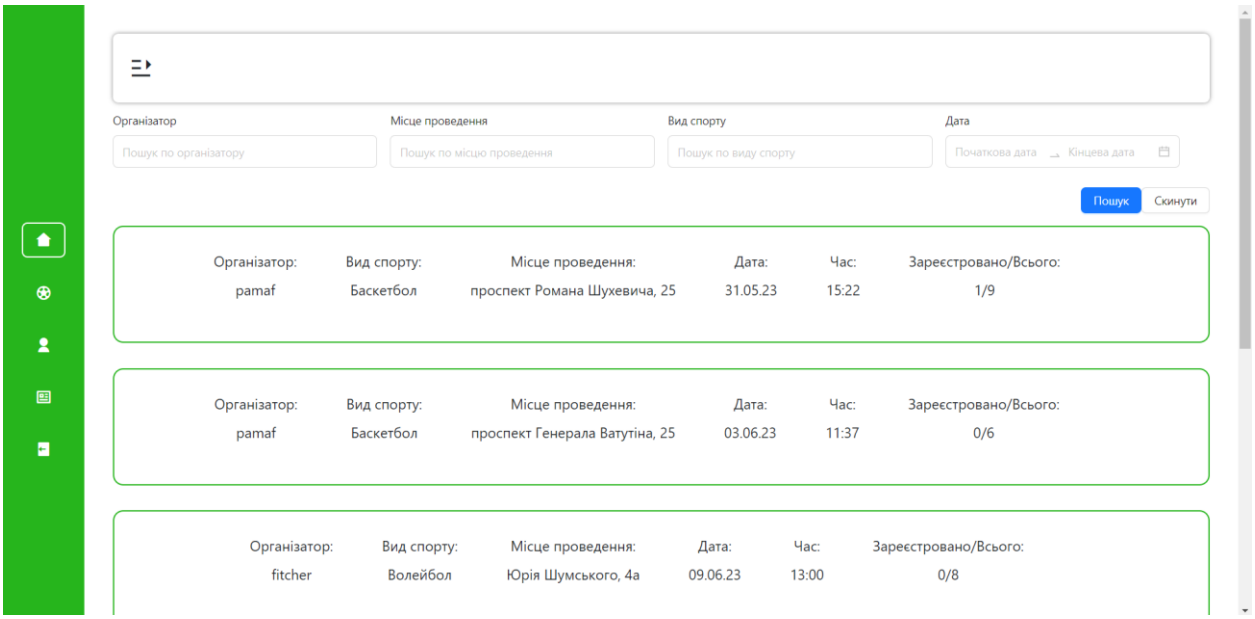


Рисунок 3.24 – Головна сторінка з закритим меню

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ОРГАНІЗАТОРІВ КОМАНДНИХ АМАТОРСЬКИХ ІГОР

Графічний матеріал

КПІ.ІТ-9419.045440.06.99

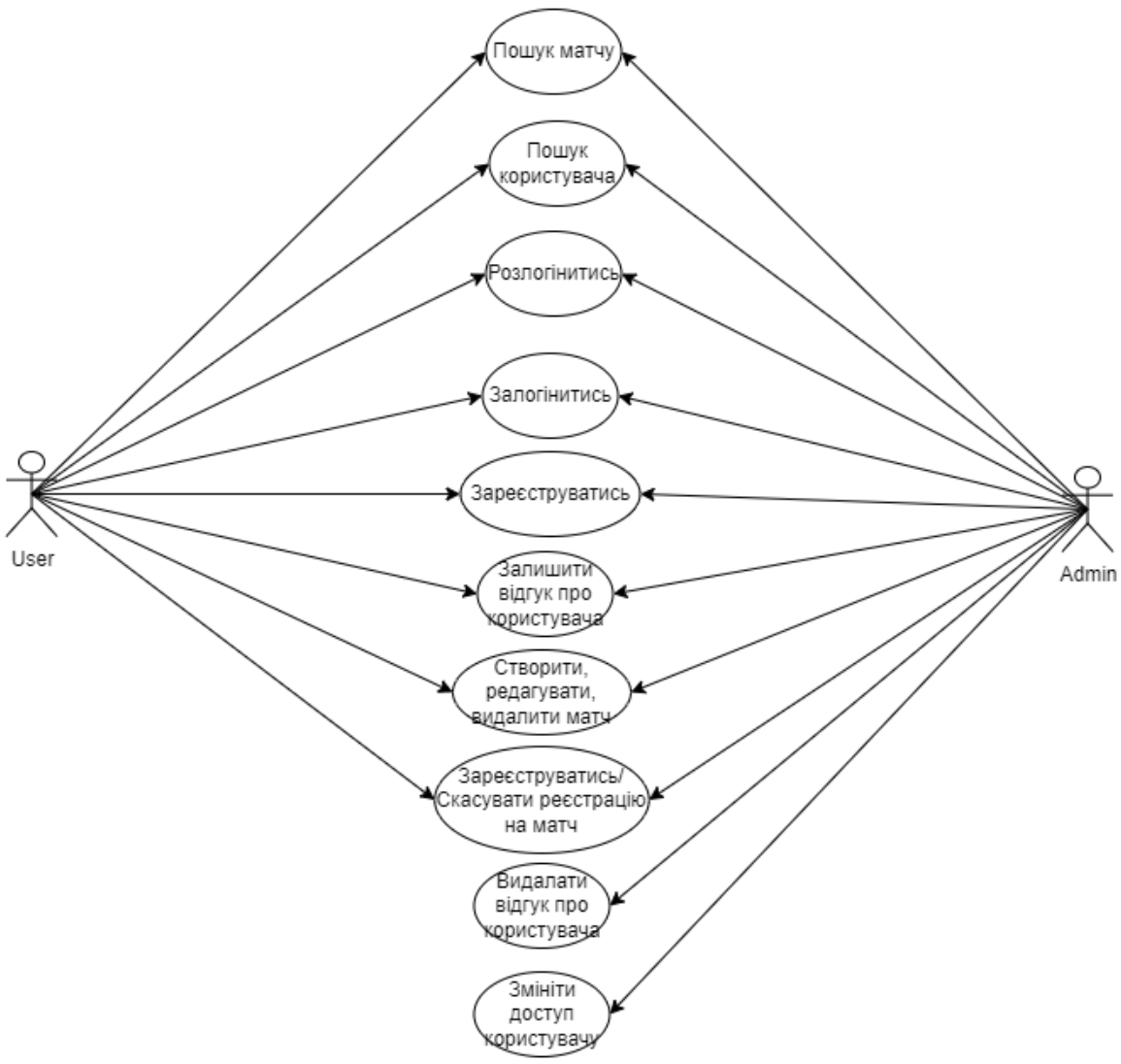
“ПОГОДЖЕНО”
Керівник проєкту:
Олександр СТЕЛЬМАХ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Андрій ПОДМОКОВ

Київ – 2023

КПІ.ІТ-9419.045440.06.99.ССВ



					КПІ.ІТ-9419.045440.06.99.ССВ			
						Лит.	Арк.	Аркушів
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використання			1
Розроб.		Подмоков А.В.				Аркуш		Аркушів
Перев.		Стельмах О.П.			Веб-застосунок для підтримки діяльності організаторів командних аматорських ігор	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-94		
Т. Кон.								
Н. Кон.		Вітковська І.І.						
Затв.		Жаріков Е.В.						

Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.		Подмоков А.В.		
Перев.		Сменьмах О.Л.		
Т. Кон.				
Н. Кон.		Вімковська І.І.		
Затв.		Жарікова Е.В.		

КПІ.ІТ-9419.045440.07.99.СБД

Літ.	Арк.	Аркушів
		1

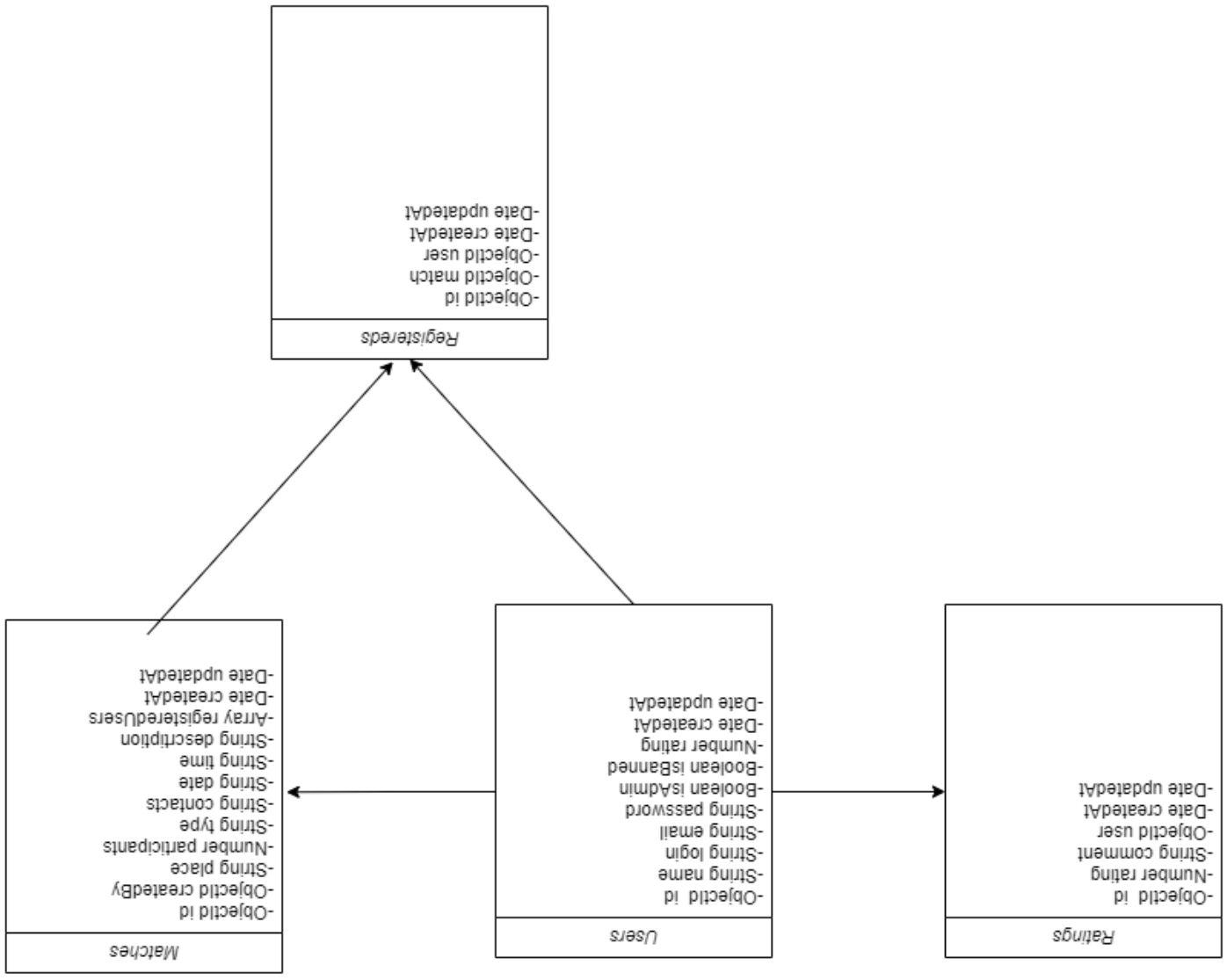
Схема бази даних

Веб-застосунок для підтримки діяльності організаційних комітетів студентських змагань

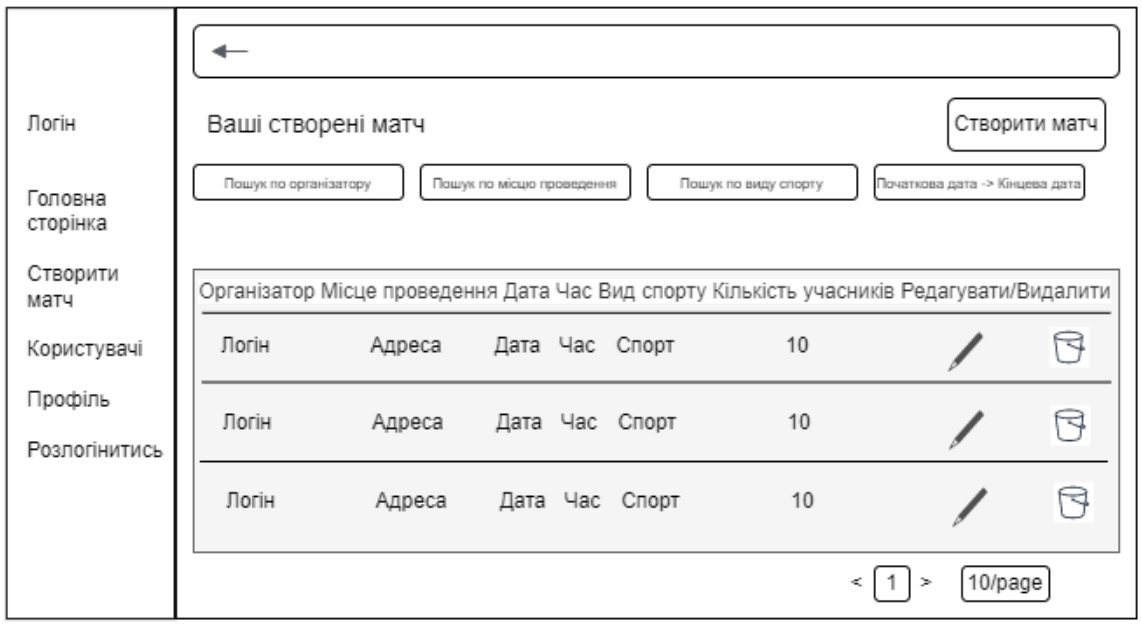
КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-94

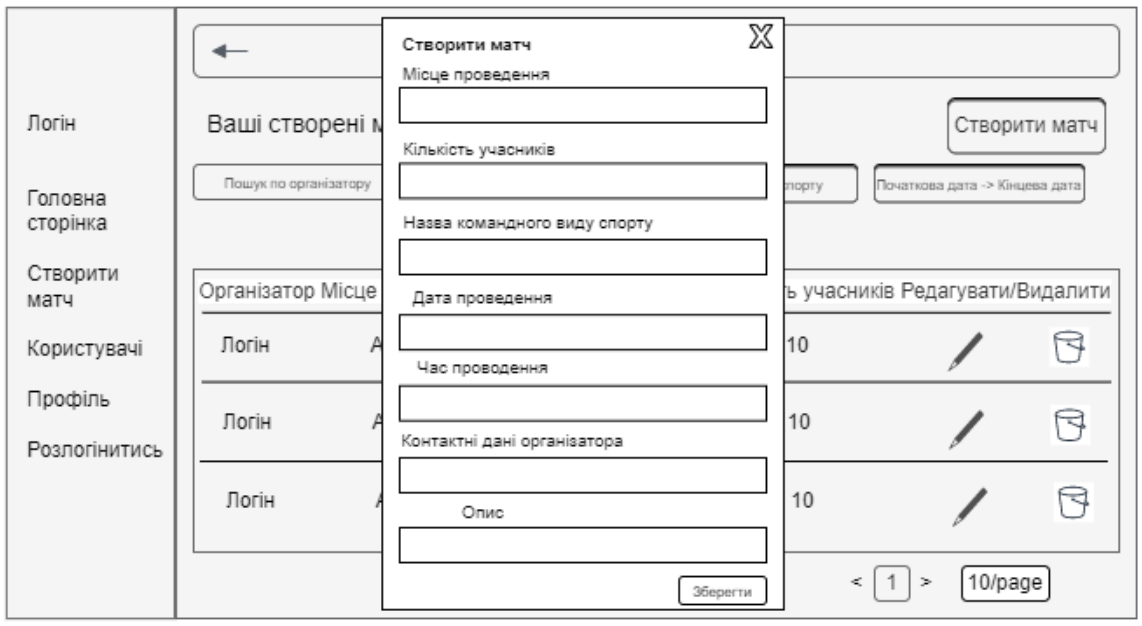
Аркуш

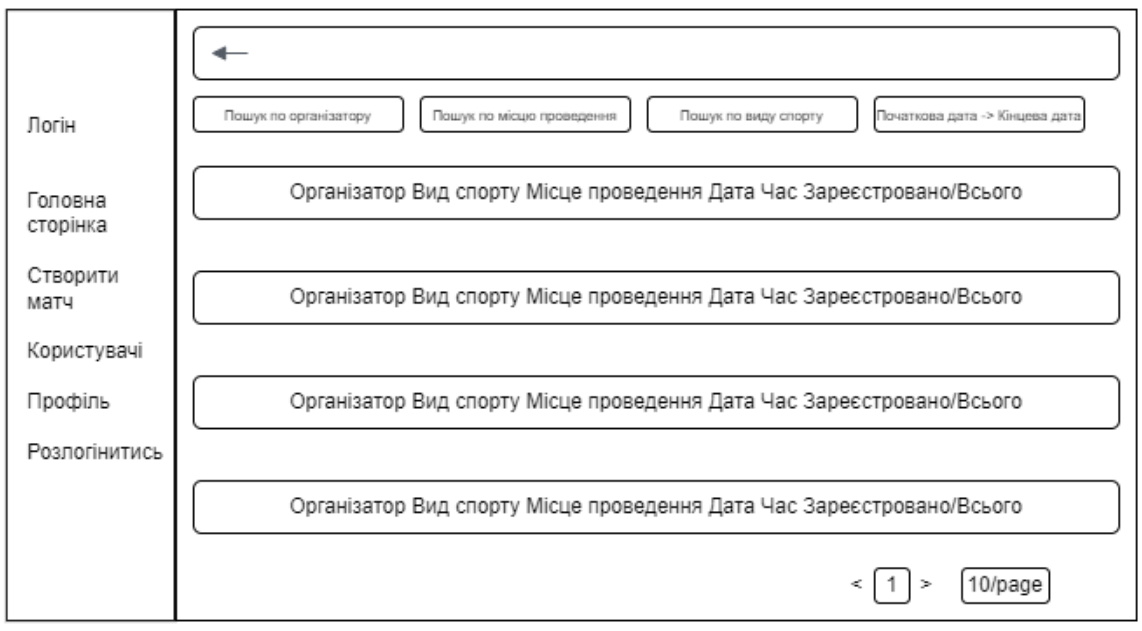
Аркушів



КПІ.ІТ-9419.045440.07.99.СБД







Зм.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9419.045440.08.99.КЕ		
Розроб.		Подмоков А.В.			Креслення вигляду екранних форм	Лит.	Арк. / Аркушів: 1 / —
Перев.		Стельмах О.П.				Аркуш	Аркушів
Т. Кон.					Веб-застосунок для підтримки діяльності організаторів командних аматорських ігор	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-94	
Н. Кон.		Вітковська І.І.					
Затв.		Жаріков Е.В.					