НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

кафедра обчислювальної техніки
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Стіренко С.Г.
(підпис)
“___”_____2021 р.

Дипломний проект

освітньо-кваліфікаційного рівня “бакалавр”
(назва ОКР)

з напряму підготовки (спеціальності) 123 «Комп’ютерна інжнерія»

на тему: Система створення й проведення тестів з перевірки знань

Виконав: студент 4 курсу, групи ІО-71
Авдієнко Ілля Олексійович _____
(прізвище, ім’я, по батькові) (підпис)

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

Консультант основна частина к.т.н., доц. Роковий О.П. _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___”_____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Авдієнка Ілля Олексійовича

1. Тема проєкту *Система створення й проведення тестів з перевірки знань*
керівник проєкту Роковий Олександр Петрович, к.т.н., доцент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від_____2021 року №___
2. Термін здачі студентом закінченого проєкту_____
3. Вихідні дані по проєкту *технічна документація, теоретичні та статистичні дані*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *Опис та аналіз предметної області, аналіз інструментів для розробки програмного забезпечення, розробка програмного забезпечення*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) *структурна схема системи, узагальнена схема роботи системи, схема взаємодії програми*
6. Консультанти проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання_____

Календарний план

№ з/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	15.12.2020-14.01.2021	
2.	*Вивчення та аналіз завдання*	15.01.2021-14.03.2021	
3.	*Розробка загальної структури системи та її архітектури*	15.03.2021-24.03.2021	
4.	*Розробка структури компонентів системи*	25.03.2021-04.04.2021	
5.	*Програмна реалізація системи*	05.04.2021-20.04.2021	
6.	*Оформлення пояснювальної записки*	21.04.2021-23.05.2021	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Анотація

Основна ідея даної роботи — розробка веб-застосунку для полегшення контролю знань в ході навчального процесу та автоматизація перевірки такого контролю, що допомагало б знизити навантаження на викладача та студента та спростити їхню взаємодію. Головна мета проекту -- розробити робочий прототип такої платформи, яка була б інтуїтивно зрозумілою у користуванні і достатньо гнучкою для пристосування під той чи інший тип контролю знань, а також фіксує результати поточного контролю для зручнішого підведення підсумків.

У результаті роботи було створено прототип платформи, що відповідає поставленим вимогам та дозволяє організувати ефективну взаємодію користувачів.

Annotation

The main idea of the study is the development of a web app for academic performance rating in the educational process and the automation of the control of such rating, which helped to reduce some academic load of assigning a student and to facilitate the process of teachers and students' interaction. The main goal of the project is to develop a working prototype of such a platform, which would be intuitive and user-friendly, and sufficiently adjustive for attaching to this type of academic performance rating, as well as fixing the results of in-line control for the data.

As the result of the study, we have built a prototype of such a platform, so that we can supply it to the users and allow organizing the effective interaction of the users.

Опис альбому
до дипломного проєкту

на тему: «Система створення й проведення
тестів з перевірки знань»

Київ – 2021 року

№ рядка	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
1	***А4***	***ІАЛЦ.466538.001 ОА***	Система створення й	1		
			проведення тестів з перевірки			
			знань			
			Опис альбому			
2	***А4***	***ІАЛЦ.466538.002 ТЗ***	Система створення й			
			проведення тестів з перевірки			
			знань			
			Технічне завдання	3		
3	***А4***	***ІАЛЦ.466538.003 ПЗ***	Система створення й			
			проведення тестів з перевірки			
			знань			
			Пояснювальна записка	69		

Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*	***ІАЛЦ.467800.001 ОА***		
Розроб.		Авдієнко І.О.			*Система створення й проведення тестів з перевірки знань Опис альбому*	*Літ.*	*Арк.*	*Аркушів*
Перев.		Сімоненко В. П.					1	1
						“КПІ імені Ігоря Сікорського” ФІОТ ІО-71		

Технічне завдання
до дипломного проєкту

на тему: «Система створення й проведення
тестів з перевірки знань»

Київ – 2021 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Це технічне завдання поширюється на розробку система створення й проведення тестів з перевірки знань. Область застосування: організація тестування під час навчального процесу.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проєкту професійної програми “Комп'ютерні системи та мережі” спеціальності 123 “Комп'ютерна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки даного програмне забезпечення призначене для людей, які бажають мати можливість створювати й проводити тести, а також для зручного проходження цих тестів людьми.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, технічна документація, публікації в Інтернеті на цю тему.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- реєстрація та авторизація користувачів;
- перегляд, створення, редагування та видалення тестів викладачами-адміністраторами;
- можливість проходження тестів студентами;
- збереження оцінок користувача за тести;

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows XP, MS Windows Vista, MS Windows 7, MS Windows 8/8.1, MS Windows 10.
- Python 3.9+

5.3. Вимоги до апаратної частини

- Комп’ютер на базі Intel Pentium 4 і вище.
- Обсяг оперативної пам’яті не менше 256 Мбайт.

6. ЕТАПИ РОЗРОБКИ

	Дата
6.1 Вивчення необхідної літератури	18.02.2021
6.2 Складання і узгодження технічного завдання	07.03.2021
6.3 Написання вступної частини та огляд рішень	24.03.2021
6.4 Розробка архітектури додатку	06.04.2021
6.5 Написання програмної частини	10.04.2021
6.6 Тестування та виправлення помилок	06.05.2021
6.7 Оформлення документації дипломного проєкту	19.05.2021

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Токен — це згенероване програмою значення, електронно-цифровий ключ, що забезпечує доступ до певної інформації.

JWT — стандарт створення токенів для надання доступу до певного набору інформації з використанням JSON формату.

API — програмний інтерфейс застосування, сукупність правил і засобів, які вможливлюють взаємодію між різними складовими частинами програмного забезпечення.

Багаторівнева архітектура — клієнт-серверна архітектура, в якій розділяються функції представлення, обробки і зберігання даних.

Вебзастосунок (також вебпрограма) — прикладне програмне забезпечення, що виконуються на спеціальному мережевому ресурсі (сервері), а клієнтом є веббраузер користувача. Дані від клієнту передаються з браузера до вебсерверу та/або частково оброблюються на стороні клієнта.

Фреймворк (англіцизм, неологізм від framework — остов, каркас, структура) — інфраструктура програмних рішень, що полегшує розробку складних систем; комплексна бібліотека, що задає правила структури проєкту.

Патерни Програмування (іноді також шаблони програмування, ПП, ШП) — це напрацьовані ефективні підходи, техніки та правила розв’язання задач, що виникають під час розробки ПЗ. Такі підходи є універсальними і мовонезалежними та ефективно транслюються в програмний код будь-якою мовою програмування.

UX дизайн — це комплекс заходів, таких як підходи, методи, техніки та процеси створення зручних у використанні цифрових продуктів, що включає

в себе інтерактивний та візуальний дизайн, інформаційну архітектуру та взаємодію комп’ютер-людина.

ВСТУП

Інформатизація освіти є невід’ємною складовою інформатизації суспільства. Вона виступає як складова загальної тенденції глобалізації світових процесів розвитку та істотно впливає на організаційні форми, зміст та методи навчання й керування навчально-пізнавальною діяльністю. Це, у свою чергу, призводить якісної та всеохоплюючої трансформації діяльності усіх учасників освітнього процесу — викладачів, студентів, керівників навчальних закладів.

Одним з найбільш визначальних напрямків інформатизації освіти є автоматизація створення та проведення тестів, яка породжує необхідність в розробці освітніх платформ та систем автоматизованної перевірки знань. Оцінювання знань в письмовій та усній формі забирає багато часу, як у студентів, так і у викладачів.

У зв'язку з цим виникає питання про створення системи тестування що автоматизує процес складання проміжкових поточних контрольних робіт, підведення підсумків, і надає викладачеві необхідну інформацію про результати здачі вищеназваних робіт студентами. Система повинна забезпечувати ефективну взаємодію користувачів з метою спрощення здачі предмета, а також зменшити навантаження викладача по оцінюванню знань студентів.

Розділ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальний огляд педагогічного тестування і класифікація типів тестів

Тести для перевірки знань є одним з варіантів реалізації педагогічної діагностики. Педагогічна діагностика — це підрозділ педагогіки, що вивчає принципи і методи розпізнавання та встановлення ознак, що характеризують нормальний (або з відхилом від норм) перебіг педагогічного процесу, його стан і результати процесу навчання, і що дає змогу на цій основі прогнозувати можливі відхилення, визначати шляхи їх попередження, а також коригувати процес навчання з метою підвищення якості його результату.

Предметом педагогічної діагностики є ознаки та індикатори компонентів педагогічних (навчальних) процесів, симптоми індивідуально-педагогічних особливостей особистості, групи. [1]

Мета педагогічної діагностики: розробка методів всебічного вивчення ознак, симптомів педагогічного процесу та забезпечення валідності, надійності та достовірності їхніх результатів.

Завдання педагогічної діагностики — розробка методів розпізнавання стану особистості (або групи) шляхом фіксації його найважливіших (визначальних) параметрів та зіставлення виявлених параметрів із законами і тенденціями педагогіки для прогнозу поведінки досліджуваного об'єкта, прийняття рішення про вплив на його поведінку в наміченому напрямі.

Педагогічна діагностика стосується до різних розділів навчально-виховного процесу: до вивчення особистості індивіда, групи і колективу людей, навчальної та виховної діяльності, до системи управління освітою. Її

застосовують у всіх типах дошкільних та позашкільних установ і навчальних закладів.

Педагогічне тестування — це форма вимірювання знань учнів, заснована на застосуванні педагогічних тестів. Включає в себе підготовку якісних тестів, власне проведення тестування і подальшу обробку результатів, яка дає оцінку навченості осіб, що проходять дане тестування.

Педагогічний тест — це інструмент оцінювання навченості учнів, що складається з системи тестових завдань, стандартизованої процедури проведення, обробки та аналізу результатів. [2]

Тести можна класифікувати за різними ознаками:

- за програмними цілями — інформаційні, діагностичні, навчальні, мотиваційні, атестаційні;
- за процедурою створення — стандартизовані, що не стандартизовані;
- за способом формування завдань — детерміновані, стохастичні, динамічні;
- за технологією проведення — паперові, в тому числі паперові з використанням оптичного розпізнавання, натурні, з використанням спеціальної апаратури, комп'ютерні;
- за формою завдань — закритого типу, відкритого типу, встановлення відповідності, упорядкування послідовності;
- по наявності зворотного зв'язку — традиційні та адаптивні.

Традиційний тест містить список питань і різні варіанти відповідей. Кожне питання оцінюється в певну кількість балів. Результат традиційного тесту залежить від кількості питань, на які було дано правильну відповідь.

Традиційний тест — система завдань, що пред'являється в порядку збільшення складності в один і той же час, з однаковою системою оцінювання для всіх осіб, що проходять тест.

Адаптивним тестом називається особливий вид тесту, в якому кожне наступне завдання вибирається залежно від відповідей на попередні завдання. Послідовність завдань і їх кількість в такому вигляді тесту визначається динамічно. Самими значущими перевагами комп'ютерного адаптивного тестування перед традиційним є можливість адаптації під рівень знань особи, що проходить тест (не доведеться відповідати на занадто складні або занадто прості питання) та економія часу і сил за рахунок скорочення кількості завдань (довжина тесту може бути зменшена до 60%) без втрати рівня достовірності.

Тестове завдання — складова частина педагогічного тесту, що відповідає вимогам технологічності, форми, змісту і, крім того, зі статистичними вимогам:

- заданим рівнем складності;
- достатньою варіації тестових балів;
- позитивної кореляцією балів завдання з балами по всьому тесту.

Тестування в педагогіці виконує три основні взаємопов'язані функції: діагностичну, навчальну і виховну. Діагностична функція полягає у виявленні рівня знань, умінь, навичок учня. Це основна і найочевидніша функція тестування. За об'єктивності, широті і швидкості діагностування, тестування перевершує всі інші форми педагогічного контролю. [3]

Навчальна функція тестування полягає в мотивуванні учня до активізації роботи по засвоєнню навчального матеріалу. Для посилення навчальної функції тестування можуть бути використані додаткові заходи стимулювання студентів, такі як: роздача викладачем примірного переліку питань для

самостійної підготовки, наявність в самому тесті навідних запитань і підказок, спільний розбір результатів тесту.

Виховна функція проявляється в періодичності та неминучості тестового контролю. Це дисциплінує, організовує і направляє діяльність учнів, допомагає виявити і усунути прогалини в знаннях, формує прагнення розвинути свої здібності.

Очевидно, що у порівнянні з іншими формами контролю знань тестування, як і її аналоги, має свої переваги і недоліки. Серед переваг часто виділяють:

- Тестування є більш якісним і об'єктивним способом оцінювання, його об'єктивність досягається шляхом стандартизації процедури проведення, перевірки показників якості завдань і тестів цілком.
- Тестування — більш справедливий метод, воно ставить всіх учнів в рівні умови, як в процесі контролю, так і в процесі оцінки, практично виключаючи суб'єктивізм викладача. За даними Британської асоціації NEAB, що займається підсумковою атестацією учнів Великобританії, тестування дозволяє знизити кількість апеляцій більш ніж в три рази, зробити процедуру оцінювання однаковою для всіх учнів незалежно від місця проживання, типу і виду освітньої установи, в якому займаються учні.
- Тести це більш об'ємний інструмент, оскільки тестування може включати в себе завдання з усіх тем курсу, в той час як на усний іспит зазвичай виноситься 2-4 теми, а на письмовий — 3-5. Це дозволяє виявити знання учня по всьому курсу, виключивши елемент випадковості при витягуванні квитка. За допомогою

тестування можна встановити рівень знань учня по предмету в цілому і по окремих його розділів.

- Тест це більш точний інструмент, так, наприклад, шкала оцінювання тесту з 20 питань, складається з 20 пунктів, в той час, як звичайна шкала оцінки знань — тільки з чотирьох.
- Тестування більш ефективно з економічної точки зору. Основні витрати при тестуванні припадають на розробку якісного інструментарію, тобто мають разовий характер. Витрати ж на проведення тесту значно нижче, ніж при письмовому або усному контролі. Проведення тестування і контроль результатів в групі з 30 чоловік займає півтори-дві години, усний або письмовий іспит — не менше чотирьох годин.
- Тестування — це більш м'який інструмент, вони ставлять всіх учнів в рівні умови, використовуючи єдину процедуру і єдині критерії оцінки, що призводить до зниження передекзаменаційні нервових напружень. [4]

Серед недоліків тестування як засобу контролю освітнього процесу можна визначити:

- Розробка якісного тестового інструментарію — тривалий, трудомісткий і дорогий процес. Стандартні набори тестів для більшості дисциплін ще не розроблені, а розроблені зазвичай мають дуже низьку якість.
- Дані, одержувані викладачем в результаті тестування, хоча і включають в себе інформацію про прогалини в знаннях по

конкретних розділах, але не дозволяють судити про причини цих прогалин.

- Тест не дозволяє перевіряти і оцінювати високі, продуктивні рівні знань, пов'язані з творчістю, тобто імовірнісні, абстрактні і методологічні знання.
- Широта охоплення тем в тестуванні має і зворотну сторону. Учень при тестуванні, на відміну від усного або письмового іспиту, не має достатньо часу для скільки-небудь глибокого аналізу теми.
- Забезпечення об'єктивності і справедливості тесту вимагає прийняття спеціальних заходів щодо забезпечення конфіденційності тестових завдань. При повторному застосуванні тесту бажано внесення в завдання змін.
- У тестуванні присутній елемент випадковості. Наприклад, учень, що не відповів на просте питання, може дати правильну відповідь на більш складний. Причиною цього може бути, як випадкова помилка в першому питанні, так і вгадування відповіді у другому.

Тести мають задовольняти основним властивостям, порушення будь-якого з них робить непридатним тест:

- Валідність — відповідність вимірюваним знанням, умінням;
- Складність — обсяг розумових зусиль для вибору відповіді;
- Надійність — правильність і адекватність відображення рівня знань;
- Стійкість — рівнозначність для різних груп випробовуваних;
- Репрезентативність — повнота охоплення навчального матеріалу;
- Значимість — актуальність включення в тест;

- Достовірність — відповідність сучасному стану науки і методики навчання;

Будь-які тести повинні бути валідними, досить складним, надійними, об'єктивними, стійкими, репрезентативними, значимими, достовірними, науковими та несуперечливими. [5]

1.2. Аналіз відомих програмних продуктів

Наявно багато прикладів програмного забезпечення, що створене для перевірки результатів освітнього процесу. У кожного з них є свої переваги та недоліки. В рамках освітнього процесу важлива уніфікація, тож розроблювана мною платформа має однаково підходити для створення тестових завдань з будь-якого предмету. Для аналізу було обрано декілька прикладів програмного забезпечення, були розглянуті як і ПЗ, що запускається в браузері (вебзастосунки), так і десктопні (тобто такі, що запускаються на локальній машині в користувача і позбавлені клієнт-серверної логіки взаємодії). Такий підхід — тобто аналіз доволі різних підходів до розв’язання одного і того ж питання — дозволяє змістовно оцінити переваги й недоліки кожного підходу та вивести оптимальну конфігурацію розв’язання необхідного питання, спираючись на отриманні нижче результати.

1.1.1 **Аналіз програмного забезпечення «Айрен»**

Айрен — це безкоштовна програма, що дозволяє створювати тести для перевірки знань та проводити тестування в локальній мережі, через інтернет або на одиночних комп'ютерах. [6]

Тести можуть включати в себе завдання різних типів: з вибором однієї або кількох правильних відповідей, з введенням відповіді з клавіатури, на встановлення відповідності, на впорядкування і на класифікацію.

При мережевому тестуванні викладач бачить на своєму комп'ютері докладні відомості про успіхи кожного з учнів. Після закінчення роботи ці дані зберігаються в архіві, де їх в подальшому можна переглядати і аналізувати за допомогою вбудованих в програму засобів.

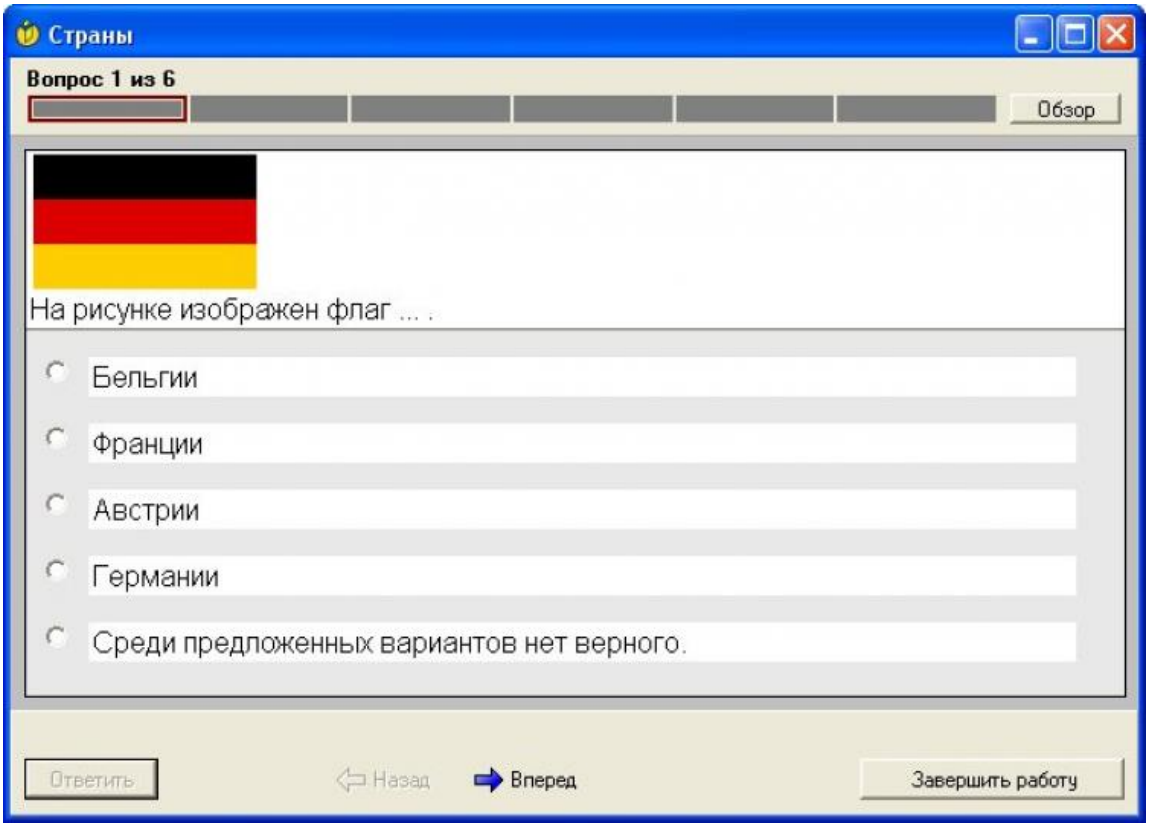


Рис.1.1 ПЗ «Айрен» в режимі тестування.

Серед переваг ПЗ «Айрен» можна виділити наступні:

- Базування і робота в LAN (local area network, локальній мережі)

- Можливість створення тестових завдань різних типів, що включають в себе вибір однієї або кількох правильних відповідей (завдання закритого типу та завдання закритого типу з декількома варіантами відповідей), а також з введенням відповіді з клавіатури (завдання відкритої форми)
- Можливість навігації між запитаннями тесту в обидва боки (можна не зациклюватися на одному запитанні, а йти до наступних, та повернутися, аби переробити вже пройдене тестове завдання)

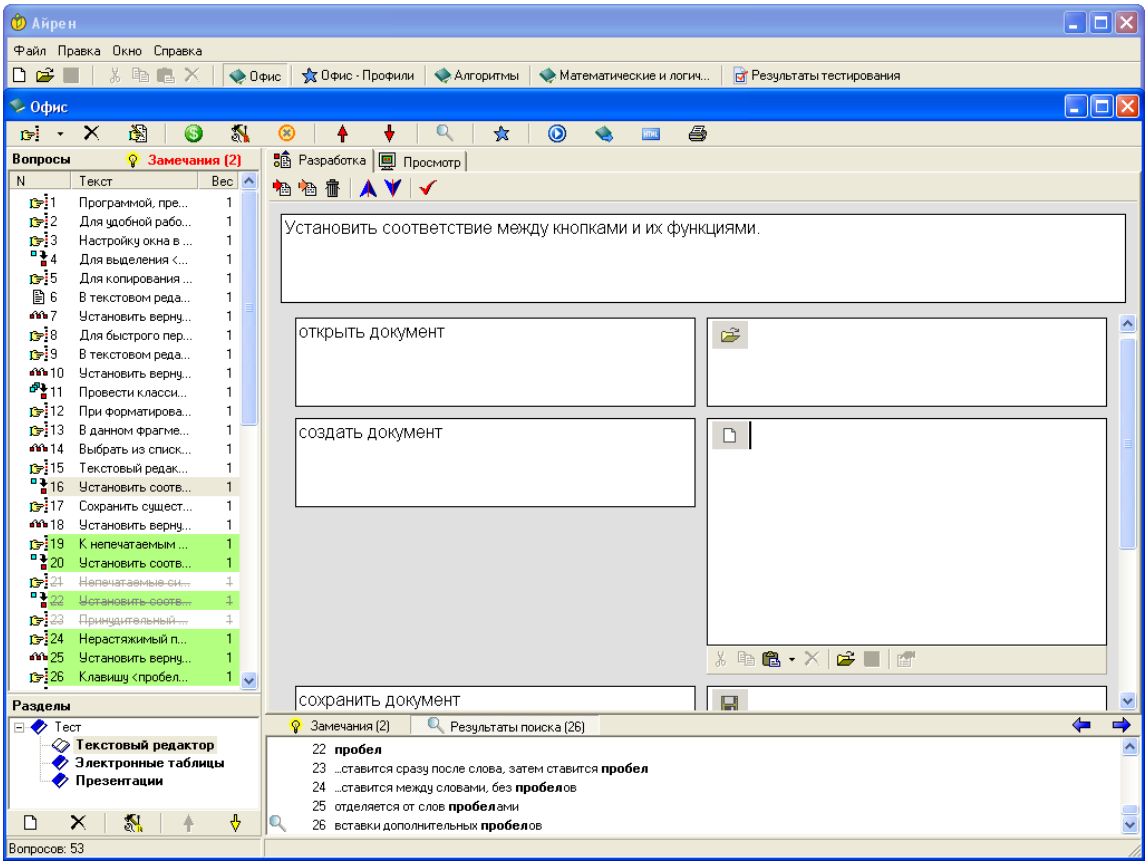


Рис.1.2. ПЗ «Айрен» в режимі створення тестів.

Недоліками даного ПЗ я вважаю наступні його риси:

- Для проходження тестів необхідно встановлення додаткової клієнтської частини ПЗ
- Застарілий UX/UI-дизайн, який не відповідає сучасним нормам, тенденціям, набутим технікам та методологіям створення продуктів в даній сфері
- Перевірка результатів проходження тесту на стороні клієнта, і як наслідок — недосконаліть мережевого коду: таке ПЗ вразливе для мережевих атак та атак підміни (мережеві пакети з оцінкою, що йдуть на сервер, можна «підмінити» на потрібні)

1.1.2 **Аналіз програмного забезпечення «Google Forms»**

Програмне забезпечення Google Forms — одна зі складових частин пакету офісних вебзастосунків Google Workspace, створених компанією Google. Google Форми доступні лише як вебзастосунок, який дозволяє користувачам створювати та редагувати опитування в Інтернеті, співпрацюючи з іншими користувачами в режимі реального часу. Зібрану інформацію можна автоматично вносити в електронну таблицю. [7]

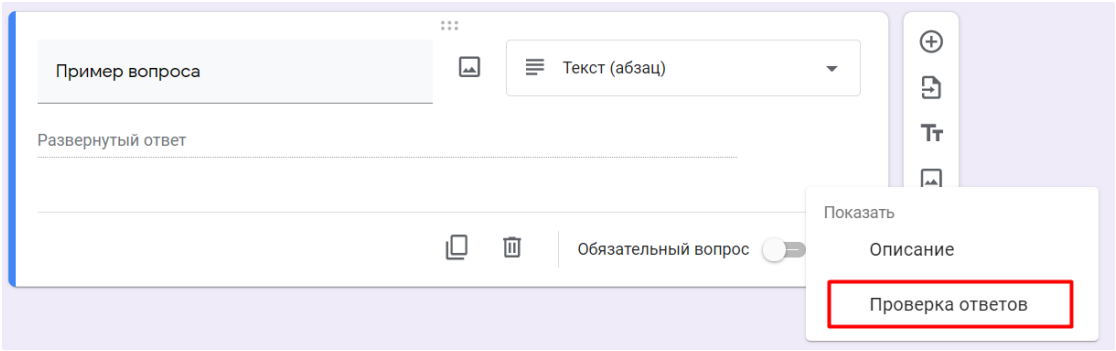


Рис. 1.4. ПЗ Google Forms в режимі створення тестів.

Серед функцій, що доступні користувачам, наявні рандомізоване (випадкове) перемішування питань, обмеження кількості відповідей один раз на користувача, вбудоване скорочування посилання на тест, користувацькі теми, автоматичне формування пропозицій відповідей під час створення форм, та «Завантажити файл» — опція для користувачів для відповіді на запитання, що вимагають від користувачів певний вміст чи файли зі свого комп'ютера чи Google Drive.

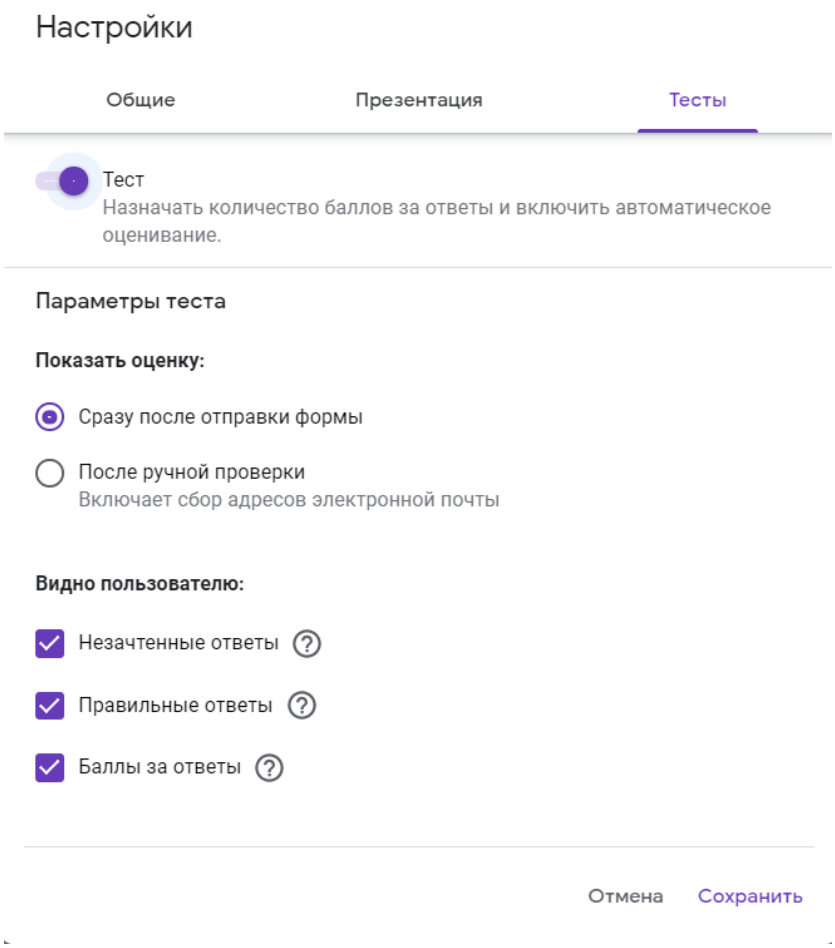


Рис. 1.3. ПЗ Google Forms в режимі налаштування тестування.

Перевагами програмного забезпечення Google Forms можна назвати:

- Сучасний UI/UX дизайн, що забезпечує зручну та зрозумілу взаємодію з даними продуктом, реалізовану за принципом WYSIWYG (What You See Is What You Get, «що бачиш, те і отримаєш») з урахуванням найпередовіших здобутків у сфері проєктування користувацьких інтерфейсів.
- Зберігання введених адміністратором тесту правильних відповідей на стороні сервера (в виділеному на обліковий запис адміністратора розділі хмарного сховища серверів Google), що дозволяє повторне використання одного й того ж самого тесту.
- Перевірка результатів проходження тесту на серверній стороні: користувач надсилає лише свої відповіді, які звіряються з правильними, збереженими в хмарі, і отримує результат. Даний підхід унеможливлює більшість способів втручання в перевірку результатів, такі як підміна пакетів чи перегляд коду сторінки. [8]

Серед недолків ПЗ Google Forms можна назвати:

- Базування в мережі Інтернет: за умови відсутності підключення до цієї мережі або проблемами з ним використання даного ресурсу є неможливим.
- Базування на віддалених серверах Google: за умови проблем на стороні серверу, які немає можливості вирішити адміністраторами тесту та користувачами, використання даної платформи може бути неможливим.

1.3. Аналіз вимог до програмного забезпечення

В системі передбачаються наступні ролі: адміністратор (також екзаменатор) та користувач (тестований).

Користувач має змогу зареєструватися в системі, проходити тести, отримувати оцінку за нього та переглядати свої результати за минулі тести. Кожен користувач має бути прив’язаним до його студентської групи.

Функціонал адміністратора-екзаменатора значно більший: окрім прав на вищезазначені операції, також він має можливість здійснити пакетну реєстрацію, імпортувавши список групи, керувати тестами (створювати нові та видаляти старі), переглядати результати проходження тестів усіма користувачами, а також експортувати результати усіх користувачів.

Популярним та зручним способом задання вимог є діаграми використання (use-case diagrams), вона наведена у додатку «Схема структурних варіантів

використань». Нижче наведено таблицю із ідентифікатором використання та найменуванням.

Таблиця 1.1 — Варіанти використання

Ідентифікатор	Назва
UC0	Реєстрація
UC1	Авторизація
UC2	Імпорт списку групи
UC3	Експорт оцінок групи
UC4	Перегляд оцінок користувача

Продовження таблиці 1.1

UC5	Створення тесту
UC6	Редагування тесту
UC7	Перегляд тесту
UC8	Проходження тесту

1.4. Розробка функціональних вимог

З урахуванням наведених вище варіантів використання було сформовано нищезазначені функціональні вимоги.

Таблиця 1.2 — Опис функціональних вимог дипломного проєкту.

Ідентифікатор	**Опис**	**Пріоритет**
REQ0	Система має надавати можливість усім користувачам реєструватися	Високий
REQ1	Система має передбачати можливість «пакетної» реєстрації декількох учасників імпортуванням списку групи	Низький
REQ2	Система має дозволяти зареєстрованому користувачу авторизуватися в системі	Високий

Продовження таблиці 1.2

REQ3	Система має надавати можливість авторизованому адміністратору переглядати усі результати тестів серед групи	Середній
REQ4	Система має надавати можливість авторизованому адміністратору експортувати результати тестів усієї групи студентів в окремий документ	Низький
REQ5	Система має дозволяти адміністратору редагувати профіль користувача, змінюючи особисті дані його профілю	Низький
REQ6	Система має надавати можливість авторизованому користувачеві переглядати власні результати минулих тестів	Середній
REQ7	Система має дозволяти авторизованому адміністратору створювати нові тести	Високий
REQ8	Система має дозволяти авторизованому адміністратору редагувати вже створені тести	Високий

Продовження таблиці 1.2

REQ9	Система має дозволяти авторизованому адміністратору обирати тест, що буде доступний для проходження користувачам	Середній
REQ10	Система має надавати можливість авторизованому користувачеві проходити тест	Високий
REQ11	Система має перевіряти надіслані користувачем відповіді і формувати оцінку	Високий
REQ12	Система має надавати можливість експортувати тести у спеціально обумовленому форматі для подальшої передачі будь-яким засобом комунікації	Низький

Висновки до розділу 1

У першому розділі було розглянуто поняття педагогічного тестування, класифікація освітніх тестів і кілька прикладів програмного забезпечення, призначення яких є схожим на тему мого дипломного проєкту. Проаналізувавши існуюче програмне забезпечення для імплементації комп’ютеризованого освітнього тестування, було зроблено висновок, що незважаючи на вдалу реалізацію ними деяких чудових особливостей та способів використання, більшість з них не реалізовують їх в одному проєкті.

Метою розробки програмного забезпечення за темою мого бакалаврського дипломного проєкту є створення системи, що впроваджувала комбінувала б усі здобутки і вдалі архітектурні, дизайнерські, функціональні та інші риси оглянутого ПЗ зі зведенням до мінімуму недоліків функціоналу, що здався мені невдалим.

Розділ 2

АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕЛІЗАЦІЇ ПРОЄКТУ

2.1. Моделювання архітектури програмного забезпечення

Для розробки описаного програмного забезпечення було вибрано форму вебзастосунку. Вебзастосунок — застосунок, заснований на принципі клієнт-серверної взаємодії. Клієнт взаємодіє з вебсервером за допомогою браузера, надсилаючи на нього дані на зберігання та обробку за певно заданою логікою. Перевагою даного методу є кроссплатформеність отримуваних застосунків, оскільки вся взаємодія відбувається через браузер, і клієнту необхідна лише коректна роботу певного браузеру під певною операційною системою. Вебзастосунок являє собою вебсайт, де розташовані сторінки з незакінченим вмістом. Відвідувачу вебсайту необхідно запросити сторінку з вебсервера, щоб отримати остаточний сформований вміст. Динамічною називається сторінка, зміст якої залежить від запиту користувача. В випадку, розглянутому в даній роботі, використання вебдодатку є дуже доречним, оскільки користувачі (як і організатори тесту, так і ті, хто його проводять) зможуть проводити й проходити тести на будь-якій зручній для них платформі, не встановлюючи додаткового ПЗ.

Для реалізації цілей користувачів вебдодатку «Платформа для тестування» також необхідно розробити користувацький інтерфейс створення самих тестів, що буде зручним та компетентним з точки зору UX та уможливлювати створення тестів різної конфігурації (тести з варіантами відповіді, тести з доповненням слів тощо). Адміністратори-викладачі зможуть вносити в систему тести, де останні мають зберігатися на час проведення.

Окрім цього, користувацький інтерфейс розв’язання тестів також має відповідати базовим стандартам UX-дизайну. Для користувачів, що проходять тести, необхідно розробити повнофункціональну і зручну систему, що надаватиме чітку і однозначну відповідь на результати користувацького вводу.

2.2. Опис розглянутих рішень

Для створення серверної частини застосунку була обрана мова програмування Python. PyCharm — інтегроване середовище розробки для Python, що надає засоби для аналізу коду, графічний зневаджувач, інструмент для запуску юніт-тестів і підтримує веброзробку на з використанням найпопулярніших вебфреймворків цієї мови програмування. В рамках даної бакалаврської роботи будуть розглянуті та порівняні за ключовими для даного проєкту принципами та функціоналами два наступні вебферймворки: Flask та Django.

Flask — мікрофреймворк для вебдодатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Поширюється відповідно до умов ліцензії BSD.

Flask називається мікрофреймворком, оскільки він не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх бібліотек. Однак, Flask має підтримку розширень, які надають додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю

процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку. [8]

Серед переваг даного вебфреймворку можна визначити наступні:

- Містить сервер для розробки та відлагоджувач
- Вбудована підтримка юніт-тестів
- Управління запитами RESTful
- Використовує шаблони Jinja2
- Має підтримку безпечних cookies (сесії на стороні клієнта)
- 100% відповідність WSGI 1.0
- Підтримка Unicode
- Сумісність з Google App Engine
- Наявність розширень для забезпечення бажаної поведінки

Django — це безкоштовний фреймворк вебдодатків на Python, який підтримується Django Software Foundation. Зазвичай проєкти, що розроблені з використанням даного вебфреймворку складається з одного або декількох додатків, тобто він орієнтований на розробку мікросервісів. Це одна з основних архітектурних відмінностей цього фреймворку від інших (таких як Ruby on Rails), і є реалізацію принципи DRY (“Don’t Repeat Yourself”, не повторюйся).

На відміну від інших фреймворків, обробники URL-адрес Django використовують обробку регулярними виразами. Для роботи з базою даних Django використовує власну ORM, в якій модель даних описується класами Python, а з неї генерується схема бази даних.

Архітектура Django схожа на модель Model-View-Controller (MVC). Елемент керування класичною моделлю MVC приблизно відповідає рівню,

який вона називає Django view, а логіка представлення View у Django реалізована на рівні Template. З цієї причини архітектуру Django часто називають Model-Template-View (MTV). [9]

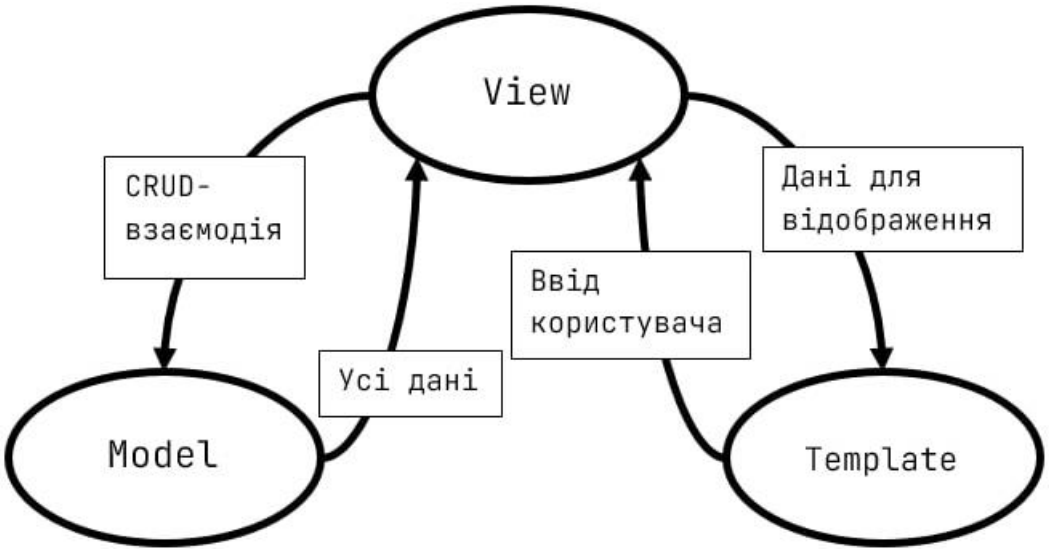


Рис 2.1 Схематичне зображення архітектури MVT

Початковий розвиток Django обумовив те, що в першу чергу цей вебфреймворк пропонує ряд інструментів, які значно спрощують та пришвидшують розробку інформаційного вебсайти. Наприклад, розробник не повинен створювати елементи керування та сторінки для адміністративної частини сайту, Django має вбудовану програму управління вмістом (також CMS), яка належить кожному сайту, створеному в Django, і може керувати кількома сайтами одночасно на одному сервері. Додаток для адміністрування дозволяє створювати, модифікувати та видаляти будь-який об'єкт вмісту вебсайту, реєструвати всі виконані дії та забезпечувати інтерфейс для управління користувачами та групами (шляхом надання прав кожному об'єкту).

Дистрибутив Django також включає програми для коментування, RSS та Atom, "статичні сторінки" (якими можна маніпулювати без написання елементів керування та переглядів), переспрямування URL-адрес тощо.

Деякі особливості Django:

- ORM з підтримкою транзакцій API доступу до бази даних
- вбудований інтерфейс адміністрування з існуючими перекладами на багато мов
- регулярний вираз url-диспетчера
- розширювана система шаблонів із мітками та успадкуванням
- кеш-система
- інтернаціоналізація
- "General views" — шаблони для функцій контролера
- автентифікація та авторизація, підключення зовнішніх модулів автентифікації: LDAP, OpenID тощо.
- система фільтрації (middleware, "проміжне програмне забезпечення") для створення додаткових обробників запитів, таких як фільтри в наборі розподілу для кешування, стиснення, нормалізації URL-адреси та підтримки анонімних сеансів
- бібліотека для роботи з формами (побудова форм відповідно до існуючої моделі бази даних)
- вбудована автоматична документація для міток шаблонів та моделей даних, доступна через програму адміністрування

Окремі елементи рами вільно з’єднані між собою, тому їх можна легко замінити на подібні. Наприклад, замість вбудованих шаблонів можна використовувати Mako або Jinja.

2.3. Детальний аналіз та розробка архітектури застосунку

При розробці вебзастосування було вирішено використовувати чотирирівневу архітектуру, яка включає в себе чотири загальних рівня вебдодатків:

- Рівень представлення (PL)
- Рівень обслуговування даних (DSL)
- Рівень бізнес-логіки (BLL)
- Рівень доступу до даних (DAL)

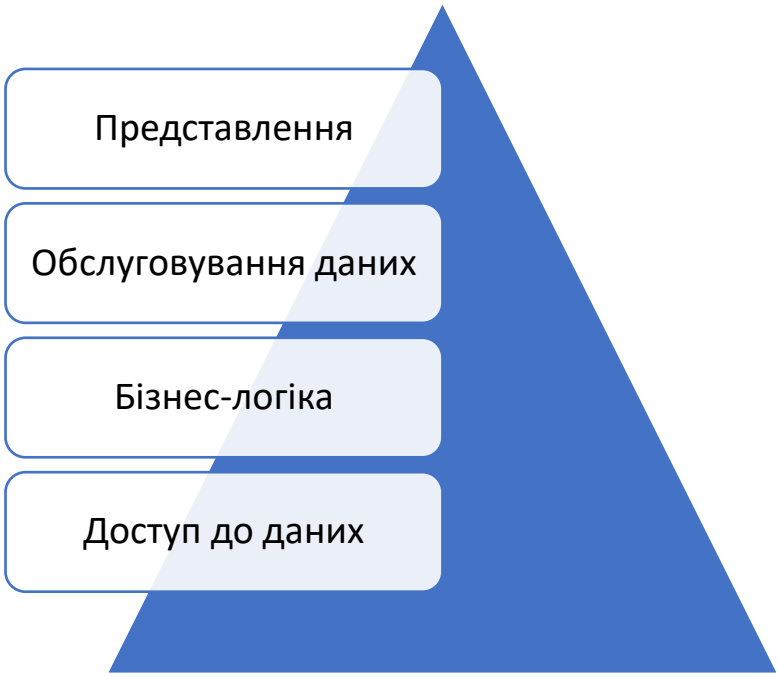


Рис 2.2 Схематичне представлення рівня архітектури вебдодатку

Рівень представлення, або PL, відображає призначений для користувача інтерфейс і спрощує взаємодію з користувачем. Рівень представлення доступний для користувачів через браузер і складається з компонентів

інтерфейсу користувача та компонентів процесів інтерфейсу, які підтримують взаємодію з системою і візуалізують в зручній для людини формі дані для користувачів. Також існують компоненти користувальницького процесу, які задають взаємодія з користувачем. PL надає всю необхідну інформацію клієнтській стороні. Основна мета рівня уявлення — отримати вхідні дані, обробити запити користувачів, відправити їх в службу даних і показати результати. При розробці рівня представленням найважливішим є наступні технологі: HTML, CSS та JavaScript. Використання цих технологій буде розглянуте трохи згодом.

Окрім цього, вебзастосунки за принципами поданням можуть бути розділеним на односторінкові (Single-Page Application, SPA) та багатосторінкові (Multiple-Page Application, MPA) застосунки.

Односторінковий застосунок під час роботи в браузері не вимагає перезавантаження сторінки та додаткового часу на очікування. Сторінку не потрібно оновлювати, оскільки вміст завантажується автоматично. Типовими прикладами односторінкових застосунків є Facebook, Trello, Gmail, Google Maps, Facebook або GitHub.

SPA самостійно запитує розмітку та дані та рендерить (відображає) сторінки прямо в браузері, що зазвичай відбувається через використання завдяки фреймворкам та бібліотекам функцій JavaScript. [10]

Односторінкові вебсайти допомагають утримувати користувача в одному зручному вебпросторі, де вміст представляється користувачеві просто, легко та ефективно.

Даний підхід має наступні легковизначені плюси:

- SPA працює швидко, оскільки більшість ресурсів (HTML + CSS + скрипти) завантажуються лише один раз протягом усього терміну роботи програми. Тільки дані передаються вперед і назад.
- Розробка спрощена та впорядкована. Немає необхідності писати код для відтворення сторінок на сервері. Почати набагато простіше, оскільки зазвичай ви можете розпочати розробку з файлового файлу: // URI, взагалі не використовуючи жодного сервера.
- SPA-служби легко налагодити за допомогою Chrome, оскільки ви можете контролювати мережеві операції, досліджувати елементи сторінки та дані, пов’язані з нею.
- Створити мобільний додаток простіше, оскільки розробник може повторно використовувати той самий серверний код для вебпрограм та власних мобільних додатків.
- SPA може ефективно кешувати будь-яке локальне сховище. Додаток надсилає лише один запит, зберігає всі дані, тоді він може використовувати ці дані і працює навіть в автономному режимі.

Частими недоліками такого підходу можна назвати:

- Зробити SEO-оптимізацію односторінкової програми дуже складним і непростим завданням. Його вміст завантажується AJAX (асинхронний JavaScript та XML) — метод обміну даними та оновлення в додатку без оновлення сторінки.
- Завантаження відбувається повільно, тому що для завантаження клієнта потрібні важкі клієнтські фреймворки.
- Для цього потрібно, щоб JavaScript був присутній і увімкнений. Якщо будь-який користувач вимкне JavaScript у своєму браузері, неможливо буде правильно представити програму та її дії.

- У порівнянні з "традиційним" додатком, SPA менш безпечний. Завдяки міжсайтовому сценарію (XSS) він дозволяє зловмисникам вводити сценарії на стороні клієнта у вебпрограму іншими користувачами.

Багатосторінкові програми працюють «традиційно». Кожна дія, наприклад, уведення користувачем нових даних чи запит інших даних, здійснює запит на сервер з метою відобразити іншу сторінку. Даний тип застосунків є великими, більшими за вищезгадані SPA. Через кількість вмісту ці програми мають багато рівнів інтерфейсу користувача. На щастя, це вже не проблема. Завдяки AJAX нам не потрібно хвилюватися, що великі та складні програми повинні передавати багато даних між сервером та браузером. Це рішення вдосконалюється, і воно дозволяє оновити лише окремі частини програми. З іншого боку, це додає більшої складності, і його важче розробити, ніж односторінковий додаток.

Серед плюсів такого підходу однозначно можна виділити:

- Це ідеальний підхід для користувачів, яким потрібна візуальна карта, куди звернутися в програмі. Цілісна, кількарівнева навігація меню є важливою частиною традиційного багатосторінкового додатка.
- Дуже добре та легко для правильного управління SEO. Це дає кращі шанси оцінити рейтинг за різними ключовими словами, оскільки додаток можна оптимізувати для одного ключового слова на сторінку.

Очевидними недоліками багатосторінкових програм є:

- Немає можливості використовувати той же серверний сервер з мобільними додатками.

- Розробка клієнтської частини програмного забезпечення та її серверної бази тісно пов'язані.
- Розробка стає досить складною. Розробник повинен використовувати фреймворки як для клієнта, так і для сервера. Це призводить до більш тривалого часу розробки додатків.

Рівень бізнес-логіки несе відповідальність за належний обмін даними. Цей рівень визначає логіку бізнес-операцій і правил. Бізнес-логіка в розробці інформаційних систем являє собою сукупність правил, принципів, залежностей поведінки об'єктів предметної області, тобто сфери людської діяльності, яку система підтримує. Інакше можна сказати, що бізнес-логіка — це реалізація правил і обмежень автоматизуються операцій. Бізнес-логіка задає правила, яким підкоряються дані предметної області. Авторизація в системі вебзастосунку — це приклад рівня бізнес-логіки.

Рівень служби даних (також DSL) передає дані, оброблені рівнем бізнес-логіки, на рівень представлення. Цей рівень гарантує безпеку даних, ізолюючи бізнес-логіку з боку клієнта.

Рівень доступу до даних, або DAL, пропонує спрощений доступ до даних, що зберігаються в постійних сховищах, таких як виконавчі файли і файли XML. Рівень доступу до даних також управляє операціями CRUD — створення, читання, оновлення, видалення. Наприклад, DAL може посилатися на об'єкт (з точки зору об'єктно-орієнтованого програмування) з його атрибутами, а не з рядками полів з таблиці бази даних. Це дозволяє створювати модулі клієнти вебзастосунків з вищим рівнем абстракції. Така модель може бути реалізована шляхом створення класу за допомогою з'єднувачів даних, який безпосередньо посилається на відповідний набір дій над базою даних. Можливо, інша програма може отримувати або записувати журнали у файлову

систему або з неї. DAL приховує складність сховища даних для доступа ззовні, реалізуючи механізм інкапсуляції даних.

Крім того, за допомогою команд, таких як "створити", "видалити" або "оновити" в певній таблиці бази даних, ви можете створити клас і багато збережених процедур у базі даних. Ці процедури можна викликати з методу в класі, який повертає об'єкт, що містить необхідні значення. Або команди створення, видалення та оновлення можуть виконуватися за допомогою простих функцій, таких як registerUser або loginUser, що зберігаються на рівні доступу до даних.

Також, важливою проблемою є проблема відображення даних не як окремих скалярних значень-записів таблиці, а як повноцінні об’єкти коду програмного забезпечення. Для подолання цієї проблеми існує підхід, також відомий як ORM, або ж об’єктно-реляційне відображення в інформатиці. Воно являє собою техніку програмування для перетворення даних між системами несумісного типу за допомогою об’єктно-орієнтованих мов програмування.

В об'єктно-орієнтованому програмуванні завдання управління даними діють на об'єкти, які майже завжди є нескалярними значеннями. Це може бути змодельовано в об'єктно-орієнтованій реалізації за допомогою "Особистого об'єкта" з атрибутом / полем для зберігання кожного елемента даних, що містить запис: ім'я людини, список телефонних номерів та список адрес. Список телефонних номерів містив би сам "PhoneNumber objects" тощо. Кожен такий запис адресної книги мовою програмування трактується як єдиний об’єкт (наприклад, на нього може посилатися одна змінна, що містить вказівник на об’єкт). З об’єктом можуть бути пов’язані різні методи, наприклад методи повернення бажаного номера телефону, домашньої адреси тощо.

На відміну від цього, багато популярних продуктів баз даних, таких як системи управління базами даних SQL (СУБД), не є об'єктно-орієнтованими і можуть зберігати та маніпулювати скалярними значеннями, такими як цілі числа та рядки, організовані в таблицях. Програміст повинен або перетворити значення об'єкта на групи простіших значень для зберігання в базі даних (і перетворити їх назад при отриманні), або використовувати лише прості скалярні значення всередині програми. Об’єктно-реляційне відображення реалізує перший підхід.

Найвідомішим ORM-рішенням для мови програмування Python є SQLAchemy. SQLAlchemy — це відкрите програмне забезпечення, розповсюджуване за ліцензією MIT, яке являє собою інструментарій бази даних та реалізації об'єктно-реляційного відображення. SQLAlchemy надає узагальнений інтерфейс для створення та виконання агностичного коду бази даних без необхідності писати оператори SQL. Окрім цього, цей розширюваний модуль надає ядро SQLAlchemy для виконання роботи з базами даних. Дане ядро, яке також називається SQLAlchemy Core, абстраговано від відмінностей у реалізаціях між PostgreSQL, SQLite та інших реляційних СУБД, що забезпечує роботу з будь-якими з них в зручному для розробнику форматі. Таким чином, ORM є лише надбудовою для Core, яка спрощує операції створення, читання, оновлення та видалення даних з бази даних, реалізовуючи їх як відповідні методи об’єктів, що відображають табличні дані.

В ході бакалаврської роботи для порівняння та знайдення найкращої бази даних для цього проєкту мною були розглянуті наступні програмні продукти: PostgreSQL та MongoDB.

Postgres — це безкоштовна реляційна система управління базами даних із відкритим кодом (RDBMS), що підкреслює розширюваність та відповідність SQL. Всі транзакції, що відбуваються в межах даної бази даних, підтримують ACID, себто властивості Atomicity, Consistentity, Isolation, Durability, автоматично оновлювані подання, матеріалізовані подання, тригери даних, зовнішніми ключами та збереженими під процедурами. Він призначений для обробки ряду робочих навантажень, від окремих машин до сховищ даних або вебсервісів із багатьма одночасними користувачами. Це база даних за замовчуванням для сервера macOS, а також доступна для Windows, Linux, FreeBSD та OpenBSD. [11]

PostgreSQL управляє паралельністю через багатоверсійний паралельний контроль (MVCC), який надає кожній транзакції "знімок" бази даних, дозволяючи вносити зміни, не впливаючи на інші транзакції. Це значною мірою усуває необхідність блокування читання та гарантує, що база даних підтримує принципи ACID. PostgreSQL пропонує три рівні ізоляції транзакцій: зчитування, повторне читання та серіалізація. Оскільки PostgreSQL не захищений від брудних читань, запит рівня ізоляції транзакції Read Uncommitted забезпечує замість цього зроблене читання. PostgreSQL підтримує повну серіалізацію за допомогою серіалізованого методу ізоляції знімків (SSI).

PostgreSQL підтримує широкий вибір власних типів даних, включаючи:

- Логічний, або булевий
- Числа довільної точності
- Символьні та текстові
- Двійкові
- Дата / час (позначка часу / час з / без часового поясу, дати, інтервалу)

- Гроші
- Перелік
- Бітові рядки
- Тип пошуку тексту
- Композитний
- Масиви (змінної довжини і можуть бути будь-якого типу даних, включаючи текстові та складені типи) з загальним обсягом пам’яті
- Геометричні примітиви
- Адреси IPv4 та IPv6
- XML, що підтримує запити XPath
- UUID
- JavaScript Object Notation (JSON) та швидший двійковий JSONB

Крім того, користувачі можуть створювати власні типи даних, які зазвичай можна зробити повністю індексованими за допомогою індексів індексування PostgreSQL — GiST, GIN, SP-GiST. Прикладом їх є типи даних географічної інформаційної системи (ГІС) з проєкту PostGIS для PostgreSQL.

В якості альтернативи такому підходу було розглянуто не-реляційну базу даних, що впроваджує NO-SQL підхід для зберігання інформації.

MongoDB зберігає дані у гнучких документах, схожих на JSON, тобто поля можуть відрізнятися від документа до документа, а структура даних може змінюватися з часом.

Модель документа відображається на об'єктах у коді програмного забезпечення одразу, з моменту побудови бази даних, що полегшує роботу з даними та прибирає необхідність в застосуванні складних та будь-яких ORM-рішень. Спеціальні запити, індексація та агрегування в режимі реального часу забезпечують неймовірно швидкі способи доступу до даних та їх

продуктивний аналіз. MongoDB реалізована у вигляді розподіленого програмного забезпечення, що підвищує кінцеву продуктивність та уможливлює такі важливі риси, як високу доступність та горизонтальне масштабування бази даних. Для використання MongoDB у програмних продуктах, написаних мовою програмування Python, написана спеціальна бібліотека PyMongo. [12]

Висновки до 2 розділу

Аналізуючи вищенаведену інформацію, отриману в ході підготовчого дослідження перед проєктуванням програмного забезпечення, обумовленого темою моєї бакалаврської роботи, я дійшов до висновку, що оптимальним рішенням буде використання наступних програмних рішень.

Для зберігання інформації був обраний реляційний підхід з використанням SQL-сумісної бази даних. PostgreSQL з його чіткою структурою підійде для збереження необхідної інформації про користувачів, поля якої зазвичай визначені наперед, а наявність спеціального типу даних для зберігання JSON-ів буде оптимальним та швидшим рішенням при зберіганні створених викладачами-адміністраторами тестів.

Оскільки розроблювана мною система передбачає відсутність серйозних навантажень на сервер, за яким залишається лише авторизація, перевірка надісланих відповідей на правильність, запис результатів в базу даних та обробка запитів на зберігання тестів, то було прийняти рішення для написання серверної частини використовувати вебфреймворк Flask.

Розділ 3

РОЗРОБКА ПРОГРАМНОГО ЗАБЕПЕЧЕННЯ

3.1. Розв’язання питання розгортання вебзастосунку

Для того, аби вебзастосунок був доступним для користування у мережі (не важливо, глобальній чи локальній), його необхідно розгорнути на спеціально виділеному комп’ютері або віртуальній машині, який після розгортання застосунку стане називатися вебсервером. Вебсервер — це комп'ютерне програмне забезпечення та основне обладнання, яке приймає запити через HTTP, мережевий протокол, створений для розповсюдження вебсторінок, або його захищений варіант HTTPS. Ресурс, надісланий з вебсервера, може бути попередньо існуючим файлом, доступним для сервера, або він може бути сформований під час запиту іншою програмою, яка взаємодіє з програмою сервера. Перший часто швидше і легше кешується для повторних запитів, тоді як другий підтримує ширший спектр програм. Вебсайти, які обслуговують створений вміст, зазвичай включають збережені файли, коли це можливо. Найпопулярнішим і найпростішим у налаштуванні програмним забезпеченням є вебсервер NginX.

У NginX робочі процеси обслуговують одночасно безліч з'єднань, мультіплексіруя їх викликами операційної системи select, epoll (Linux) і kqueue (FreeBSD). Робочі процеси виконують цикл обробки подій від дескрипторів, тим самим реалізовується парадигма подієво-орієнтованого програмування. Отримані від клієнта дані розбираються за допомогою кінцевого автомата. Розібраний запит послідовно обробляється ланцюжком модулів, що задається конфігурацією. Відповідь клієнту формується в буферах, які зберігають дані або в пам'яті, або вказують на відрізок файлу. Буфера об'єднуються в ланцюжки, що визначають послідовність, в якій дані будуть передані

клієнтові. Якщо операційна система підтримує ефективні операції введення-виведення, такі, як writev і sendfile, то nginx застосовує їх якнайшвидше.

Конфігурація HTTP-сервера nginx розділяється на віртуальні сервери (директива «server»). Віртуальні сервери поділяються на location'и («location»). Для віртуального сервера можна задати адреси і порти, на яких будуть прийматися з'єднання, а також імена, які можуть включати «*» для позначення довільній послідовності в першій і останній частині, або задаватися регулярним виразом.

location'и можуть задаватися точним URI, частиною URI або регулярним виразом. location'и можуть бути налаштовані для обслуговування запитів з статичного файлу, проксінг на fastcgi / memcached сервер.

Для ефективного управління пам'яттю nginx використовує пули. Пул — це послідовність попередньо виділених блоків динамічної пам'яті. Довжина блоку варіюється від 1 до 16 кілобайт. Спочатку під пул виділяється тільки один блок. Блок розділяється на зайняту область та незайняту. Виділення дрібних об'єктів виконується шляхом просування покажчика на незайняту область з урахуванням вирівнювання. Якщо незайнятої області у всіх блоках не вистачає для виділення нового об'єкта, то виділяється новий блок. Якщо розмір виділяється об'єкта перевищує значення константи NGX_MAX_ALLOC_FROM_POOL або довжину блоку, то він повністю виділяється з купи. Таким чином, дрібні об'єкти виділяються дуже швидко і мають накладні витрати тільки на вирівнювання.

nginx містить модуль географічної класифікації клієнтів по IP-адресою. В його основу входить база даних відповідності IP-адрес географічних регіонів, представлена у вигляді radix tree (стислий префіксне дерево або стиснений ліс) в оперативній пам'яті. nginx попередньо розподіляє перші

кілька рівнів дерева таким чином, щоб вони займали рівно 1 сторінку пам'яті. Це гарантує, що при пошуку IP-адреси для перших декількох вузлів при трансляції адреси завжди знайдеться запис.

```
while
(
    (
        cnt = read
        (
            read_file_descriptor,
            buffer,
            block_size
        ),
        write
        (
            socket_file_descriptor,
            buffer,
            count
        ) == cnt
    )
)
    byte_count += count;
```

Рис.3.1 Висхідний код nginx, приклад операції читання даних з файлу та запису їх в сокет

Як HTTP-сервер nginx значно спрощує наступні задачі:

- обслуговування незмінних запитів, індексних файлів, автоматичне створення списку файлів, кеш дескрипторів відкритих файлів
- акселерірованное проксінг без кешування, простий розподіл навантаження і відмовостійкість
- підтримка кешування при акселерірованном проксінг і FastCGI
- підтримка FastCGI і memcached-серверів, простий розподіл навантаження і відмовостійкість

- модульність, фільтри, в тому числі стиснення (gzip), byte-ranges (докачка), chunked-відповіді, HTTP-аутентифікація, SSI-фільтр
- кілька підзапитів на одній сторінці, оброблювані в SSI-фільтрі через проксі або FastCGI, виконуються паралельно
- підтримка SSL
- підтримка PSGI, WSGI
- експериментальна підтримка вбудованого Perl

Вебсервер лише приймає запити від клієнтів, у той час як програмне забезпечення, що працює на ньому, має змогу лише обробляти дані та проводити інші операції з ними, такі як читання-запис в базу даних, файли і тд. Для того, аби програмне забезпечення мало змогу обробляти дані, передані на сервер користувачем безпосередньо, між вебсервером і серверною частиною програмного забезпечення вебзастосунку має існувати «шар взаємодії», метою якого було б перекладати отримувані запити у вигляд, зрозумілий для серверу вебзастосунку. Для цього у межах даної лабораторної роботи я використав програмне забезпечення WSGI та бібліотеку для Python Werkzeug .

Специфікація WSGI точно визначає, якими повинні бути параметри виклику певних об'єктів коду програмного забезпечення Python та яким має бути повернене значення, тому кожен вебсервер знає, як розмовляти з кожним серверною частиною програмного забезпечення, і навпаки.

Werkzeug надає купу вже готового службового програмного забезпечення для спрощення розробки програм, сумісних із WSGI. Ці утиліти виконують такі дії, як розбір заголовків, надсилання та отримання файлів cookie, надання доступу до даних форми, генерування переадресацій,

генерування сторінок помилок, коли є виняток, навіть надання інтерактивного налагоджувача, який працює у браузері.

```
server {
    listen 80;
    server_name localhost;

    location / {
        include uwsgi_params;
        uwsgi_pass unix:/home/admin/avditest/avditest.sock;
    }
}
```

Рис.3.2 Приклад конфігураційного файлу для WSGI

Окрім цього, Werkzeug значно спрощує процес відлагодження та зневадження під час розробки серверного програмного забезпечення. Встановлення та налаштування повноцінного вебсервера під час розробки, такого як Apache або Nginx, — це багато зусиль, і являє собою надмірну задачу лише для тестування програми на власному комп’ютері. З цієї причини Werkzeug надає сервер розробки: простий вебсервер, який можна запускати за допомогою однієї команди і майже без конфігурації.

Таким чином, аналізуючи вищенаведені твердження, можна прийти до висновку, що для повноцінного розгортання серверної частини мого вебзастосунку необхідна зв'язка nginx+WSGI+Werkzeug+Flask. Поєднання Werkzeug+Flask являє собою повноцінний вебфреймворк і забезпечують обробку інформації, nginx обробляє тонкощі мережевих з'єднань, отримує запит і надсилає відповідь, а WSGI забезпечує обмін інформацією між вищезазначеними складовими програмного забезпечення. [13]

3.2. Проєктування бази даних під потреби дипломного проєкту

Як вже було зазначено в другому розділі даної пояснювальної записки бакалаврської дипломної роботи, в даному проєкті я обрав рішення використовувати сервер та систему керування баз даних PostgreSQL. Для зручного адміністрування даної бази даних я обрав рішення, рекомендоване самою спільнотою розробників, з відкритим висхідним кодом та зручним графічним інтерфейсом — pgAdmin.

pgAdmin — найпопулярніша платформа адміністрування та розробки з відкритим кодом для PostgreSQL. Дане середовище може використовуватися в Linux, Unix, macOS та Windows для управління PostgreSQL та EDB Advanced Server 9.5 і вище.

Дане середовище є односторінковим вебзастосунком, і саме тому користується неймовірною популярністю. Це програмне забезпечення дозволяє неймовірно зручно створювати нові таблиці, діалекти, функції, типи даних, послідовності та інші схеми прямо у вікні браузеру. Також воно дозволяє переглядати статистику з використання, будує графіки інтенсивності навантаження на сервер та інтенсивності запитів, в ньому можна будувати SQL-запити будь-якої складності, генерувати SQL-скрипти до вже будь-яких вже готових схем та таблиць, а також запускати новостворені скрипти та запити, користуючись спеціальним меню.

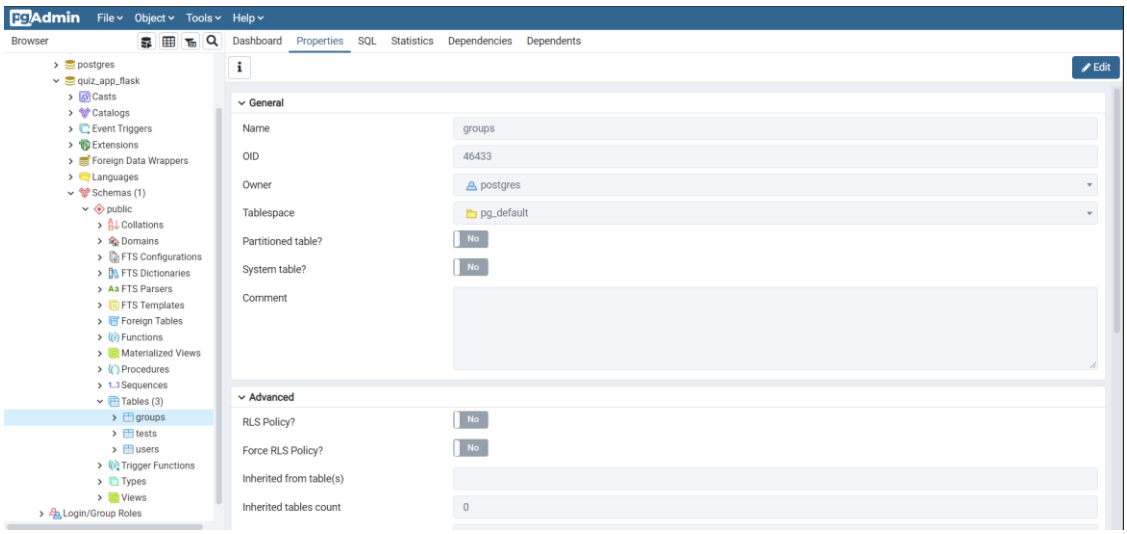


Рис.3.3 Середовище адміністрування pgAdmin

Для спрощення побудови та керування взаємодіями в даній дипломній роботі я використовую реляційний підхід.

Реляційна модель даних (РМД) — логічна модель даних, прикладна теорія побудови бази даних, яка є основним підходом до розв'язання завдання обробки даних, поєднуючи в собі такі різні елементи розділів матетматики, як теорію множин та математичу логіку першого порядку.

Реляційна модель даних включає в себе наступні компоненти:

- Структурний аспект — дані в базі даних представляють собою набір відносин.
- Аспект цілості — відношення відповідають визначеним умовам цілості. РМД підтримує декларативні обмеження цілісності рівня домену (типу даних), рівня відносин та рівня баз даних.
- Аспект обробки (маніпулювання) — РМД підтримує оператори маніпулювання відносинами (реляційна алгебра, реляційне обчислення).

- Теорія нормалізації — процес перетворення переданої до бази даних до виду, що відповідає нормальному формату, тобто таке, що характеризується лаконічністю даних та потенційно не придатне до логічно помилкового результату вибору або зміни даних у великих і сладних відношеннях.

Термін «реляційний» означає, що теорія заснована на математичному понятті відношення (відношення). У якості неформального синоніма терміна «відношення» часто зустрічається слово таблиця. Необхідно помнити, що «таблиця» є поняттям нестрогим і неформальне визначене, і часто означає не «відношення» як абстрактне поняття, візуальне представлення відносин на папері або екрані. Відношення є абстракціями і не можуть бути ні «плоскими», ні «неплоскими».

Для найкращого розуміння РМД слід відзначити три важливі обставини:

- модель є логічною, оскільки її відношення є логічними (абстрактними), а не фізичними структурами;
- для реляційних баз даних дано інформаційний принцип: все інформаційне наповнення баз даних представлено одним і лише одним способом, а саме — явним завданням значених атрибутів у кортежах відносин; зокрема, відсутність ніяких вказівників (адресів), пов'язаних з іншими значеннями;
- наявність реляційної алгебри дозволяє реалізувати декларативне програмування та декларативне опис обмеженої цілості, у доповненні до навігаційного (процедурного) програмування та процедурної перевірки умовного.

```python
from sqlalchemy import create_engine
from sqlalchemy import Column, Integer, String, ForeignKey, Text
from sqlalchemy.dialects.postgresql import ARRAY
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

engine = create_engine("postgresql://%USERNAME%:%PASSWORD%@localhost/quiz_app_flask")

Session = sessionmaker(bind=engine)
session = Session()
```

Рис.3.4 Частина висхідного коду файлу db_manager.py, у якому були визначені ORM-класи та інші необхідні для взаємодії з базою данних об'єкти

У даному дипломному проєкті для спрощення взаємодії з базами даних я скористався функціоналом бібліотеки SQLAlchemy, яка, як вже було описано в другому розділі даної роботи, є ORM-фреймворком для мови програмування Python. Конкретно в даній роботі для кращої інтеграції бази даних з Flask я скористався рішенням flask-sqlalchemy, яке є розширенням вищеназваної бібліотеки для вебфреймворку, що я його використовую в бакалаврському дипломному проєкті.

Для даного проєкту я у висхідному коді імпортував необхідні класси з різних пакетів бібліотеки SQLAlchemy. Всі визначення ORM-класів я виніс в окремий файл db_manager.py. Це я зробив для підвищення модульності проєкту. У разі необхідності дане програмне забезпечення можна буде перенести на окремий рушій бази даних або взагалі на іншу базу даних, можливо навіть іншого типу. Також таке рішення гарантує спрощене зневаження та відлагодження висхідного коду даного проєкту, оскільки повідомлення про помилки у мові програмування Python містять інформацію

про те, в якому саме висхідному коді з якого файлу інтерпретатор Python-у наштовхнувся на помилку.

```
  reraise(exc_type, exc_value, tb)
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\_compat.py", line 39, in reraise
  raise value
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\app.py", line 2447, in wsgi_app
  response = self.full_dispatch_request()
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\app.py", line 1952, in full_dispatch_request
  rv = self.handle_user_exception(e)
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\app.py", line 1821, in handle_user_exception
  reraise(exc_type, exc_value, tb)
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\_compat.py", line 39, in reraise
  raise value
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\app.py", line 1950, in full_dispatch_request
  rv = self.dispatch_request()
File "D:\Python Projects\quiz-app-flask\venv\Lib\site-packages\flask\app.py", line 1936, in dispatch_request
  return self.view_functions[rule.endpoint](**req.view_args)
File "D:\Python Projects\quiz-app-flask\app.py", line 83, in wrap
```

Рис.3.5 Приклад помилки виконання коду інтерпретатором, який явно вказує на програмний модуль, його файл та рядок, де сталась помилка

```
15  class User(Base):
16      __tablename__ = "users"
17
18      id = Column(Integer, primary_key=True)
19      name = Column(String(128), unique=True, nullable=False)
20      username = Column(String(64), unique=True, nullable=False)
21      password = Column(String(64), unique=False, nullable=False)
22
23      def __repr__(self):
24          return f"<User(id={self.id}, username={self.username})>"
```

Рис.3.6 Шматок висхідного коду, що описує ORM-клас абстрактного користувача.

У даній дипломній роботі мною для збереження інформації про користувача були створені декілька класів, що описують об’єкти-сутності. Діаграми взаємодії цих класів та їх ієрархії будуть наведені пізніше, в додатку. Поки що вважаю за необхідне зазначити, що для даного проєкту мною було обрано використовувати декілька класів, приклади вихідного коду яких я зазначу нижче.

Вихідний код даних класів вийшов напрочуд простим та лаконічним, тому вважаю, що використання бібліотеки програмного забезпечення SQLAlchemy показало себе як напрочуд вдале рішення.

```
26  class Student(User):
27      """
28      Regular User without special rights.
29      """
30
31      __tablename__ = "students"
32
33      group = Column(String(10), ForeignKey("groups.name"))
34
35      def __repr__(self):
36          return f"<Student(id={self.id}, username={self.username}, group={self.group})>"
37
38
39  class Admin(User):
40      """
41      User that can create new tests.
42      """
43
44      __tablename__ = "admins"
45
46      def __repr__(self):
47          return f"<Admin(id={self.id}, username={self.username})>"
```

Рис.3.7 Шматок вихідного коду, що описує ORM-класи, успадковані від користувача, з розширенішим функціоналом.

Окрім цього, клас Student містить в собі поле group, яке є прикладом відношення багатьох-до-одного з таблицею Group (багато студентів належать одній групі). Для цієї таблиці у висхідному коді також визначений окремий ORM-клас.

```
50  class Group(Base):
51      """
52      ORM-class for Group. Served for many-to-one relationship
53      with Student.
54      """
55      __tablename__ = "groups"
56
57      id = Column(Integer, primary_key=True)
58      name = Column(String(10), unique=True, nullable=False)
59
60      def __repr__(self):
61          return f"<Group(id={self.id})>"
```

Рис.3.8 Шматок висхідного коду, що характеризує об’єктне відображення таблиці груп.

Окрім цього, окремий ORM-клас також визначений для сутності тест. Інформація, отримувана для цього відображення, розміщується в окремій таблиці та містить в собі поле-масиви, в якому розміщені ідентифікаційний номер користувачів-студентів, які вже пройшли даний тест, а також окреме поле-масив для збереження відомостей про групи, в яких даний тест вже був проведений. Окрім цього, наявне окреме поле, відповідальне за збереження JSON. Для цього з пакету sqlalchemy.dialects.postgresql модуля sqlalchemy був

імпортований спеціальний клас JSONB, який дозволяє керувати JSON-подібними об'єктами і зберігає їх в бінарному представленні.

```
64  class Test(Base):
65      __tablename__ = "tests"
66
67      id = Column(Integer, primary_key=True)
68      title = Column(String(64), unique=True, nullable=False)
69      questions = Column(JSONB, unique=True, nullable=False)
70      users = Column(ARRAY(Integer), unique=False, nullable=False)
71      groups = Column(ARRAY(Integer), unique=False, nullable=False)
72
73      def __repr__(self):
74          return f"<Test(id={self.id})>"
```

Рис.3.9 Код, що описує об’єктне відображення сутності тест.

3.3. Проєктування системи реєстрації та авторизації користувачів

Одним із найбільш необхідних елементів даної бакалаврської роботи є правильна система реєстрації та авторизації користувачів система створення та проведення тестів з перевірки знань. Дана підсистема буде реалізовувати розподілення функціоналу між різними ролями користувачів. Користувачі-студенти зможуть створювати свої облікові записи та авторизуватися, у той час як адміністратори-викладачі матимуть змогу після успішної авторизації перейти в режим створення тестів. Даний режим буде недоступним для звичайних користувачів-студентів.

```python
class RegisterForm(Form):
    name = StringField("Name", [validators.Length(min=3, max=50)])
    username = StringField("Username", [validators.Length(min=4, max=25)])
    group = StringField("Group", [validators.Length(min=5, max=8)])
    password = PasswordField(
        "Password",
        [
            validators.Regexp(
                r"^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$",
                message="Password should contain min 8 characters "
                        "including 1 letter and 1 number.",
            ),
            validators.DataRequired(),
            validators.EqualTo("confirm", message="Password do not match"),
        ],
    )
    confirm = PasswordField("Confirm Password")
```

Рис.3.10 Вихідний код для генерування реєстраційної форми

Реєстраційна форма створена у вигляді окремого класу, який успадковується від класу Form бібліотеки програмного забезпечення WTForms.

WTForms — це гнучка бібліотека для перевірки форм та рендерингу (тобто відображення) для веброзробки з використанням мови програмування Python. Він може працювати з будь-якою вебструктурою та механізмом шаблонів, який ви вибрали. Він підтримує перевірку даних, захист CSRF, інтернаціоналізований переклад та роботу з різними локалями програмного забезпечення згідно зі стандартом I18N тощо. Оскільки WTForms є програмним забезпеченням з відкритим програмним кодом, присутні різні бібліотеки та модулі, написані спільнотою цього проєкту, які забезпечують

більш тісну і зручну інтеграцію з популярними бібліотеками та вебфреймворками, у тому числі і з Flask.

```python
@app.route("/login", methods=["POST"])
def login():
    with session:
        username = request.form["username"]
        student = session.query(Student).filter_by(username=username).one()
        admin = session.query(Admin).filter_by(username=username).one()
        if student or admin:
            if request.form["password"] == admin.password:
                web_session["can_create_tests"] = True
            elif request.form["password"] == student.password:
                web_session["can_create_tests"] = False
            else:
                error = "Invalid password"
                return render_template("login.html", error=error)
            user = admin or student
            web_session["name"] = user.name
            web_session["username"] = username
            web_session["logged_in"] = True
            return redirect(url_for("dashboard"))
        else:
            error = "Username not found"
            return render_template("login.html", error=error)
```

Рис.3.11 Зразок висхідного коду, що описує процес авторизації користувача та його наступне перенаправлення у разі успішної аутентифікації

Алгоритм авторизації користувача доволі простий і виконаний з використанням ініціалізованого раніше об’єкту вебсесії. Отримавши результати, що були введені користувачем в на спеціально відділеній сторінці, та опрацювавши введену інформацію, переглянувши та розібравши переданий

об’єкт форми LoginForm, дана функція формує запит до бази даних з метою знайти користувача з введеним в форму логіном. Якщо такий користувач не був знайденим ні в таблиці студентів, ні в таблиці адміністраторів, система генерує повідомлення про відповідну помилку і знову перенаправляє на сторінку логіну. В іншому ж випадку система переходить до перевірки паролю. Результат запиту в базу даних на минулому кроці був перетворений на об’єкт, в якого наявний атрибут password (пароль від облікового запису користувача), за яким тепер відбувається перевірка. Якщо паролі співпадають — користувач успішно авторизований.

В залежності від того, який користувач авторизувався (адміністратор-викладач чи студент), вебсесії приписується відповідний атрибут, який дозволяє чи унеможливлює перехід в режим створення тестів. Об’єкт вебсесії створюється при кожній авторизації користувача і є сугубо унікальним для нього, незалежним від інших дій інших користувачів об’єктом. Після цього відбувається перенаправлення на головну сторінку вебзастосунку.

Роутінг (себто таблиця перенаправлень) із викликом необхідних функцій при умові введення певного URL чи методу HTTP-запиту в Flask зазвичай задається неявно. Для цього використовується така особливість синтаксису мови програмування Python, як декоратори.

Декоратор — це функція, яка дозволяє обернути іншу функцію для розширення її функціональності без безпосереднього зміни її коду. Ось чому декоратори можна розглядати як практику метапрограмування, коли програми можуть працювати з іншими програмами як зі своїми даними. В Python все є об'єктом, а не тільки об'єкти, які ви створюєте з класів. У цьому сенсі він (Python) повністю відповідає ідеям об'єктно-орієнтованого програмування. Це означає, що в Python все це — об'єкти. Той факт, що все є об'єктами,

уможливлює деякі цікаві практики. Ми можемо зберігати функції в змінні, передавати їх в якості аргументів і повертати з інших функцій. Можна навіть визначити одну функцію всередині іншої.

Іншими словами, вираз @decorator_function є всього лиш синтаксичним цукром, що викликає decorator_function() з іншою функцією як аргумент і привласнює імені переданої функції отриманий перевизначений об'єкт функції.

Декоратори значно спрощують читабельність коду і дозволяють використовувати деякі гарні практики. Так, наприклад, замість постійної громіздкої перевірки, чи є користувач зареєстрованим в системі, в ході виконання даного бакалаврського дипломного проєкту мною був розроблений декоратор @is_logged(), вихідний код якого був наведений нижче:

```
def is_logged(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if "logged_in" in web_session:
            return f(*args, **kwargs)
        else:
            flash("Unauthorized, Please login", "danger")
            return redirect(url_for("login"))

    return wrap
```

Рис.3.12 Вихідний код декоратору @is_logged()

3.4. Проєктування та розробка підсистеми створення тестів

При розробці даного проєкту мною було прийнято рішення розбити розроблений мною вебзастосунок на декілька окремих модулів-підсистем. Такий підхід при проєктуванні програмного забезпеченя вебсерверів називається мікросервісною архітектурою.

Мікросервісна архітектура — варіант сервіс-орієнтованої архітектури програмного забезпечення, спрямований на взаємодію наскільки це можливо невеликих, слабо пов'язаних і легко змінюваних модулів — мікросервісов, що набув поширення в середині 2010-х років у зв'язку з розвитком практик гнучкої розробки і DevOps.

Якщо в традиційних варіантах сервіс-орієнтованої архітектури модулі можуть бути самі по собі досить складними програмними системами, а взаємодія між ними часто покладається на стандартизовані великовагові протоколи (такі, як SOAP, XML-RPC), в мікросервісній архітектурі системи шикуються з компонентів, що виконують щодо елементарні функції, і взаємодіючі з використанням економічних мережевих комунікаційних протоколів (в стилі REST з використанням, наприклад, JSON, Protocol Buffers, Thrift). За рахунок підвищення гранулярності модулів архітектура націлена на зменшення ступеня зачеплення і збільшення зв'язності, що дозволяє простіше додавати і змінювати функції в системі в будь-який час.

Філософія мікросервісов фактично копіює філософію Unix, згідно з якою кожна програма повинна «робити щось одне, і робити це добре» і взаємодіяти з іншими програмами простими засобами: мікросервіси мінімальні і призначаються для єдиної функції. Основні зміни в зв'язку з цим накладаються на організаційну культуру, яка повинна включати

автоматизацію розробки і тестування, а також культуру проєктування, від якої потрібно передбачати обхід колишніх помилок, виняток по можливості успадкованого коду (мікросервіси часто замінюють цілком, оскільки їх функції елементарні).

Властивості, характерні для мікросервісною архітектури:

- модулі можна легко замінити в будь-який час: акцент на простоту, незалежність розгортання та оновлення кожного з мікросервісів;
- модулі організовані навколо функцій: мікросервіс по можливості виконує тільки одну досить елементарну функцію;
- модулі можуть бути реалізовані з використанням різних мов програмування, фреймворків, сполучного програмного забезпечення, виконуватися в різних середовищах контейнеризації, віртуалізації, під керуванням різних операційних систем на різних апаратних платформах: пріоритет віддається на користь найбільшої ефективності для кожної конкретної функції, ніж стандартизації засобів розробки і виконання;
- архітектура симетрична, а не ієрархічна: залежності між мікросервісами однорангові.

Враховуючи вищезазначену інформацію, я прийняв рішення про винесення підсистеми створення тестів в окремий мікросервіс. Йому буде виділіний окремий скрипт, який буде вміщати в себе код, яким буде досягнута обробка необхідного функціоналу.

```
from flask import Flask, request, render_template

# Flask setup
app = Flask(__name__)

try:
    app_root = os.environ["EDWIN_BASE"]
except:
    app_root = "/app/edwin"


@app.route('/', methods=['GET', 'POST'])
def index():
    curval = loaddata(app_root + "/edwin.json")
    return render_template('index.html', curval=json.dumps(curval))
```

Рис.3.13 Частина висхідного коду серверної частини підсистеми створення тестів

Код цього окремого мікросервісу доволі простий: там всього лиш відбувається ініціалізація окремого об’єкту застосунку та наявні декілька методів експорту утворених об’єктів тесту в базу даних.

Об’єкти тесту створюються у вигляді JSONy — спеціального об'єкту серіалізації даних, який підтримують більшість сучасних мов програмування. Ці об’єкти доволі зручно передавати за допомогою мережевого протоколу HTTP. Окрім цього, вони дуже просто конвертуються в об’єкти JavaScript, а отже, такий формат передачі даних є дуже зручним при сервер-клієнтських транзакціях.

```
// stuff for the modal ws window
function display_ws_modal() {
    var id = '#dialog';
    //transition effect
    $('#mask').fadeIn(500);
    $('#mask').fadeTo( a: "slow",  b: 0.8);

    //Get the window height and width
    var winH = $(window).height();
    var winW = $(window).width();

    //Set the popup window to center
    $(id).css('top', winH / 2 - $(id).height() / 2);
    $(id).css('left', winW / 2 - $(id).width() / 2);

    //transition effect
    $(id).fadeIn(1000);
}
```

Рис.3.14 Шматок висхідного коду клієнтської частини програмного забезпечення мікросервісу створення тестів

Структура об’єктів тесту доволі проста:

```
[
  {
    "question1": {
      "text": "Який вебфреймворк Python-у є найкращим?",
      "answers": [
        "Flask",
```

```
        "Django",
        "Pyramid",
        "aiohttp"
      ],
      "correct": [
        0,
        3
      ]
    }
  },
  {
    "question2": {
      "text": "Хто заснував Python?",
      "correct": "Гвідо ван Россум"
    }
  }
]
```


Як було зазначено вище, після закінчення редагування отриманий JSON можна зберегти в базу даних під потрібним іменем. Після цього він буде доступний для проходження іншим користувачам, у тому числі і для

проходження користувачами-студентами. Окрім цього, адміністратори-викладачі зможуть редагувати вже створені в системі тести.

```
$.fn.editable = function(target, options) {

    if ('disable' == target) {
        $(this).data('disabled.editable', true);
        return;
    }
    if ('enable' == target) {
        $(this).data('disabled.editable', false);
        return;
    }
    if ('destroy' == target) {
        $(this)
            .unbind($(this).data('event.editable'))
            .removeData('disabled.editable')
            .removeData('event.editable');
        return;
    }

    var settings = $.extend({}, $.fn.editable.defaults, {
        target: target
    }, options);

    /* setup some functions */
    var plugin = $.editable.types[settings.type].plugin || function() {};
    var submit = $.editable.types[settings.type].submit || function() {};
    var buttons = $.editable.types[settings.type].buttons ||
```

Рис.3.14 Шматок висхідного коду клієнтської частини програмного забезпечення мікросервісу створення тестів

Висновки до розділу 3

В даному розділі мною були описані етапи розробки системи створення та проведення тестів з перевірки знань та впроваджені в ході проектування рішення.

Так, наприклад, як вже було зазначено вище, розроблена мною серверна частина вебзастоунку для доступу в локальній мережі буде розміщуватися на комп’ютері чи віртуальній машині з встановленим програмним забезпеченням вебсерверу. Для свого програмного забезпечення я обрав комплекс програмних рішень, складовими частинами якого є вебсервер nginx, до якого підключив транслятор у виклики серверного скрипту WSGI разом з бібліотекою Werkzeug.

Для роботи з базою даних я підключив бібліотеку SQLAlchemy. Вище у цьому ж розділі була наведене ієрархія класів, які я створив для опису сутностей, що беруть участь у роботі даної системи.

Окрім цього, мною був описаний формат JSONy, який я використовую для збереження тестів в системі. Також я надав зразки висхідного програмного коду, що використовується в клієнтський частині підсистеми створення тестів.

Розділ 4

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНСТРУКЦІЯ ПО РОБОТІ З НИМ

4.1. Перевірка на зручність UX/UI

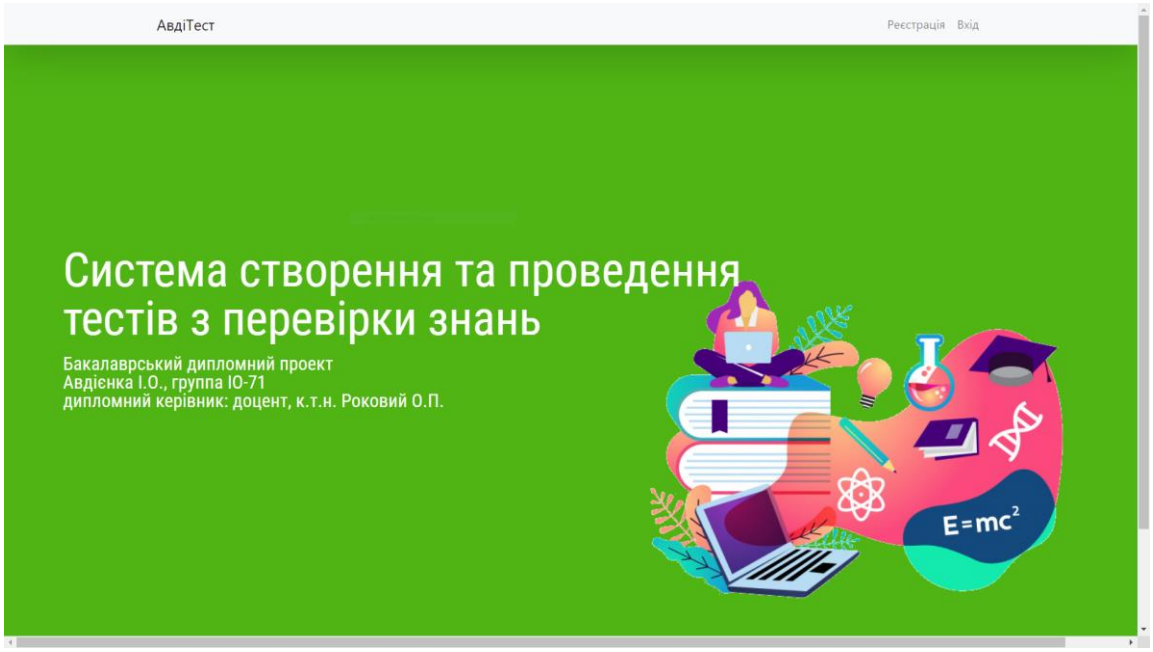


Рис.4.1 Головна сторінка вебзастосунку

Головна сторінка виконана з використанням найновіших тенденцій в дизайні та UX. Був обраний зелений колір, оскільки, згідно з більшістю професійних предметних досліджень, саме цей колір є найбільш сприйнятливим для використання в більшості інтерфейсів. Він є приємним для очей, клітини сітківки ока, що відповідають за сприйняття зеленого кольору, дуже легко збудливі, а отже, це сприяє тому, що очі не будуть швидко втомлюватися. Увесь дизайн даного вебзастосунку був виконаний з дотриманням подібної кольорової гами. Окрім цього, вибір саме цього кольору як провідного при розробці дизайну даного дипломного проєкту обумовлений

ще й тим, що цей колір дуже легко, без особливих проблем реагує на зміну насиченості, світлості, контрастності та інших колірних налаштувань, тим самим збільшуючи діапазон можливих змін в дизайні клієнтської частини даної системи програмного забезпечення, тим самим підвищуючи портованість даної системи.

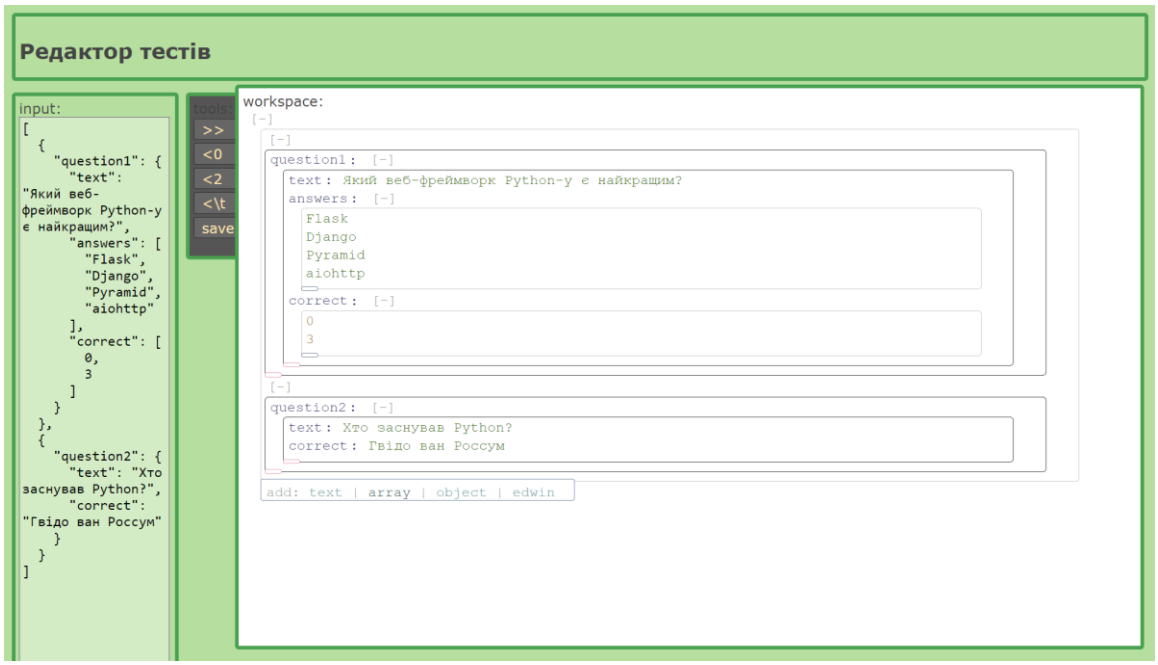


Рис.4.2 Інтерфейс створення та редагування тестів

Інтерфейс форми побудови об’єктів тесту також побудований з урахуванням правильного дизайну. Головний блок, в якому відбувається основне редагування, виділений кольором і місцеположенням, кожний з підпунктів сформованого тесту має окреме виділення. Меню додавання та редагування вже існуючих об’єктів не перенавантажують візуальною інформацією користувача. В цілому даний дизайн можна характеризувати як лаконічний, як і з точки зору кольорової гами, так і зі сторони розташування об’єктів інтерфейсу та способів взаємодіяти з ними. Також варто зазначити, що даний інтерфейс в повній мірі реалізовує принцип WYSIWYG (What You

See Is What You Get, «що бачиш, те і отримаєш»), який формально виступає еталоном при створені інтерфейсів вебзастосунків, що спрямовані на створення користувацького контенту.

4.2. Інструкція до використання

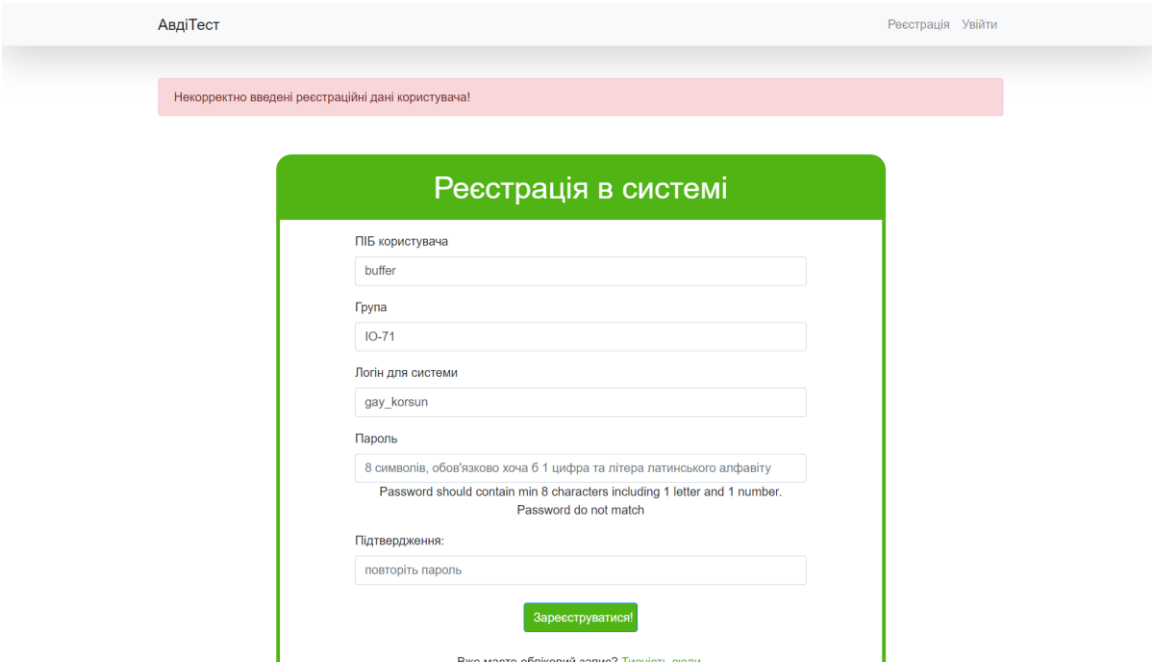


Рис.4.3 Вікно реєстрації в системі

Вікно реєстрації в системі передбачає введення користувачем, що бажає зареєструватися, його ПІБ, групи, логіну для системи, пароль. Також форма вимагає підтвердити пароль за допомогою спеціального поля. Як свідчить підказка, правильний пароль має складатися з восьми (8) символів, містити лише латинські літери, та мати в своєму складі принаймні одну цифру та одну літеру.

По завершенню реєстрації користувач має натиснути на кнопку «Зареєструватися», після чого система отримає та обробить введені користувачем дані та сформує окремий запис у відповідній таблиці баз даних. Вважаю за необхідне зауважити, що в даній версії цієї системи можливість прямої реєстрації користувача як адміністратора-викладача, себто користувача-екзаменатора, який має право створювати тести, відсутня; на даний момент єдиною можливістю додати користувача з такими правами є додати запис про нього напряму в базу даних через систему її адміністрування, оминаючи інтерфейс вебзастосунку.

Також, як помітно з рисунку вище, у разі відсутності введених користувачем полів система не дозволить продовжити реєстрацію і повідомить про неможливість даної дії користувача, залишивши його на цій сторінці, давши змогу виправити необхідні дані.

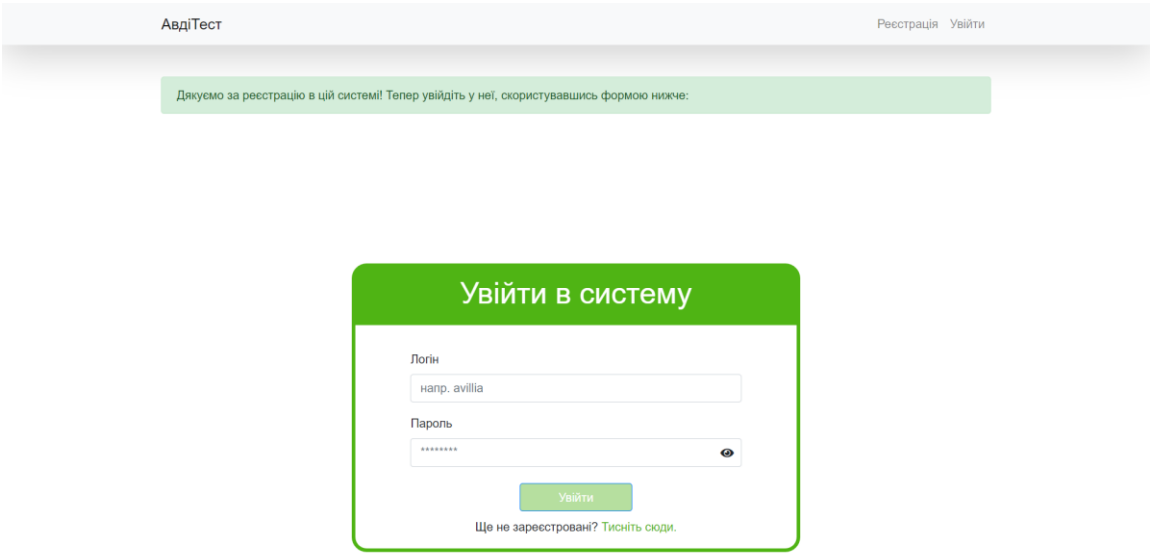


Рис.4.4 Вікно входу в систему

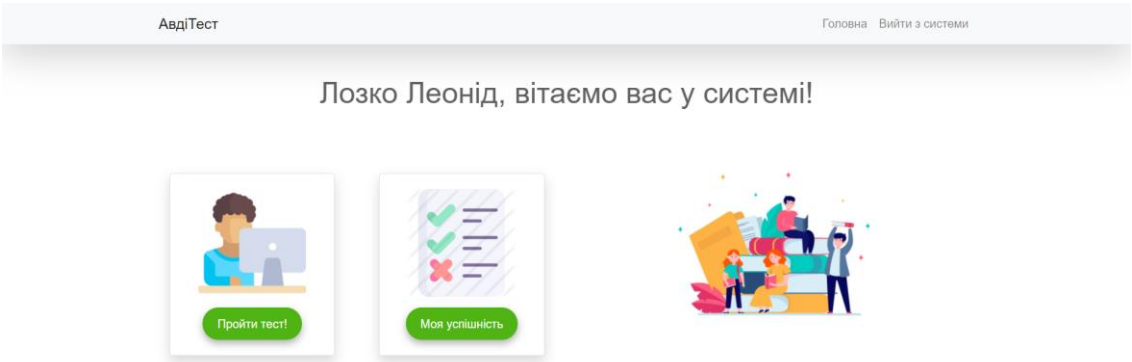


Рис.4.5 Знімок екрану, що демонструє кабінет облікового запису студента-користувача освітньої системи

За умови успішної авторизації (та передуючої ній реєстрації) користувача автоматично перенаправляє на сторінку кабінету його облікового запису. На даній сторінці вебзастосунку користувачу з таким рівнем доступу (себто таким набором визначених прав) доступна можливість обрати опцію перегляду усіх його оцінок за пройдені тести, а також доступна для натискання кнопка-посилання для переходу на сторінку проходження тесту. Також для користувача доступні опції повернення на голову сторінку системи та вихід з неї, що припиняє поточну сесію та чистить дані про неї.

Зауважу, що якщо користувач не має права створювати тести, потрібна опція у нього не з’явиться. Вона стає доступною лише для користувачів зі спеціальним рівнем доступу. Вигляд кабінету облікового запису користувача з правом створювати тести має трошки інший вигляд і зображений на наступному рисунку.

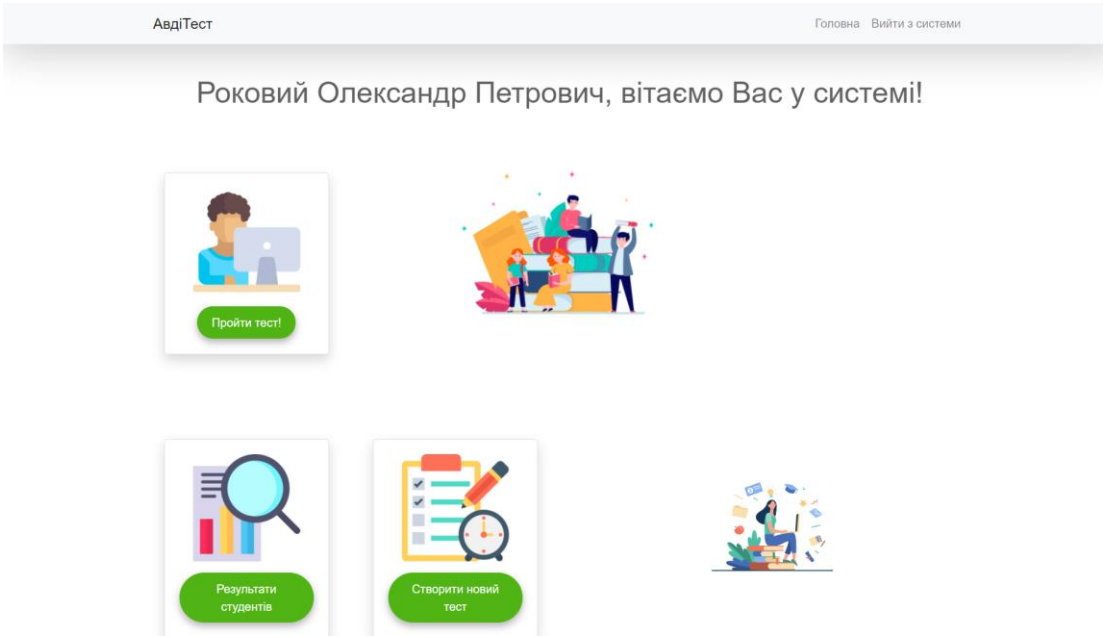


Рис.4.5 Вигляд системи для користувача з правом створення тестів (викладача-адміністратора)

Після натиснення на кнопку «створити новий тест» відбувається перенаправлення на сусідній мікросервіс, де запущений вебзастосунок створення тестів.

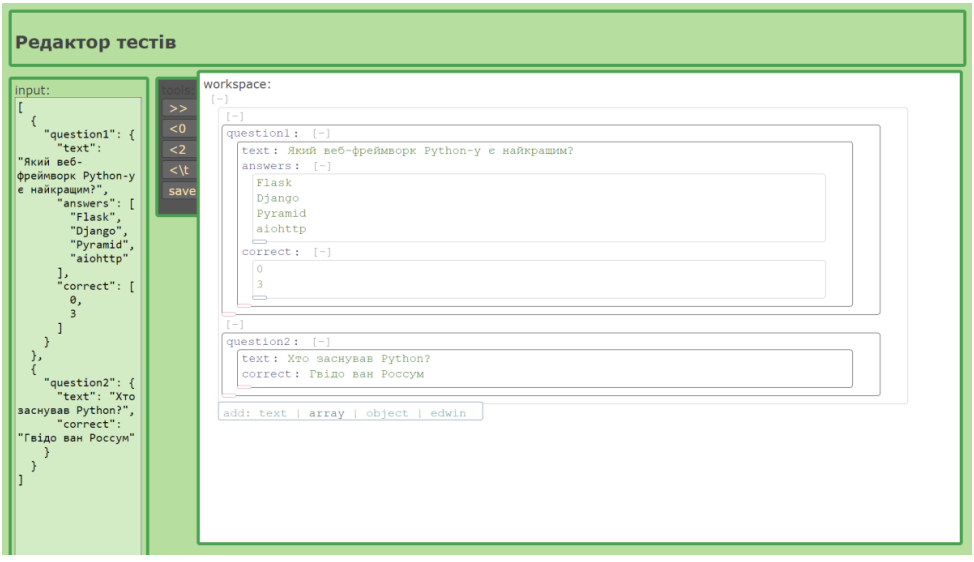


Рис.4.6 Клієнтська частина створення та редагування тестів

Висновок до розділу 4

В даному розділі мною була проаналізована робота створеною мною системи зі створення та перевірки освітніх тестів. Були розглянуті основні способи використання, перевірена працездатність підсистеми реєстрації та авторизації. Дані системи показали себе відмінно. Також було показано, що зовнішній вигляд вебзастосунку та доступний функціонал для користувачів з різними правами відрізняються.

Також моя робота була проаналізована з точки зору визначеної початково мети створення доступного UI/UX. Я вважаю, що створена мною клієнтська частина програмного забезпечення цілком відповідає заявленому стандарту. Звичайно ж, є можливість подальшого вдосконалення, так, наприклад, інтерфейс створення та редагування тестів міг би бути й кращим. Незважаючи на те, що він інтуїтивно зрозумілий і повністю й беззаперечно виконує свою функцію, його зовнішній вигляд не є досконалим.

Вважаю, що поставлені мною в ході розробки вимог до даного бакалаврського дипломного проєкту задачі були успішно виконані.

Список використаної літератури

1. Педагогическая диагностика | Під ред. К. Інгенкампа. *Переклад з нім.*− М.: 1991. − 525 с

2. *Майоров А. Н.* Теория и практика создания тестов для системы образования | Майоров А. Н. – М.: Народное образование, 2000. — 352 с. *Бурлачук Л. Ф.* Словарь справочник по психодиагностике | Бурлачук Л.Ф., Морозов С.М. − СПб.: Питер, 1999.− 528с.

3. *Аванесов В. С.* Определение исходных понятий теории педагогических измерений | Педагогические измерения. – 2005. – № 2. – С. 17-20.

4. *Клайн Пол* Справочное руководство по конструированию тестов. Київ, 1994. — 283 с.

5. Останін С. Айрен: главная страница. Программа тестирования знаний Айрен. URL: https://irenproject.ru/ (дата звернення: 08.05.2021).

6. Create a survey using Google Forms. Docs editors Help. Google Inc. (дата звернення: 09.05.2021).

7. More ways to build and share Google Forms. G Suite Updates. September 29, 2014. (дата звернення: 09.05.2021).

8. Welcome to flask — flask documentation (2.0.x). Flask. URL: https://flask.palletsprojects.com/en/2.0.x/ (date of access: 16.05.2021).

9. The Web framework for perfectionists with deadlines | Django. Django. URL: https://www.djangoproject.com/ (дата звернення: 13.05.2021).

10. Flanagan D. JavaScript: the definitive guide. 2nd ed. Sebastopol, CA : O'Reilly, 1997. 454 p.

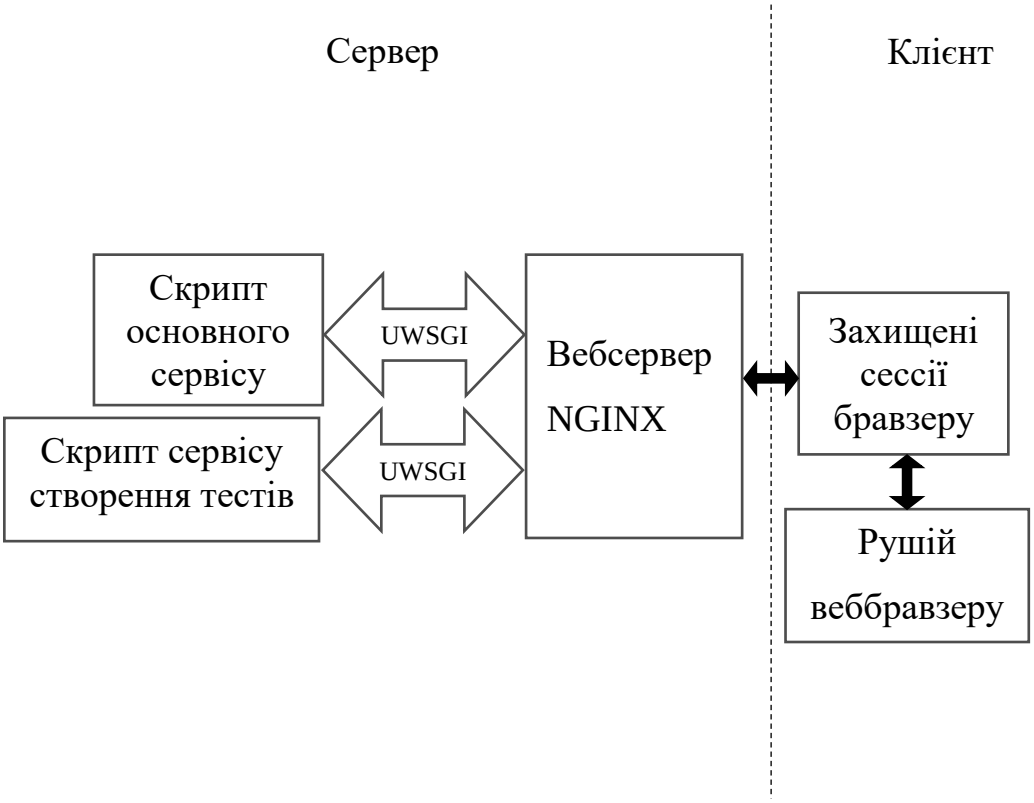
11. PostgreSQL. PostgreSQL. URL: http://www.postgresql.org/ (дата звернення: 20.05.2021).

12. What is NoSQL? NoSQL Databases Explained. MongoDB. URL: https://www.mongodb.com/nosql-explained (дата звернення: 07.05.2021).

13. Ellingwood J., Juell K. Обслуживание приложений Flask с uWSGI и Nginx в Ubuntu 18.04 | DigitalOcean. DigitalOcean. URL: https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-uswgi-and-nginx-on-ubuntu-18-04-ru (дата звернення: 19.05.2021).

Додаток 1
СТРУКТУРНА СХЕМА РОБОТИ ВЕБ-ДОДАТКУ
до дипломного проєкту
на тему: «Система створення та проведення тестів з перевірки знань»

Київ – 2021 р.

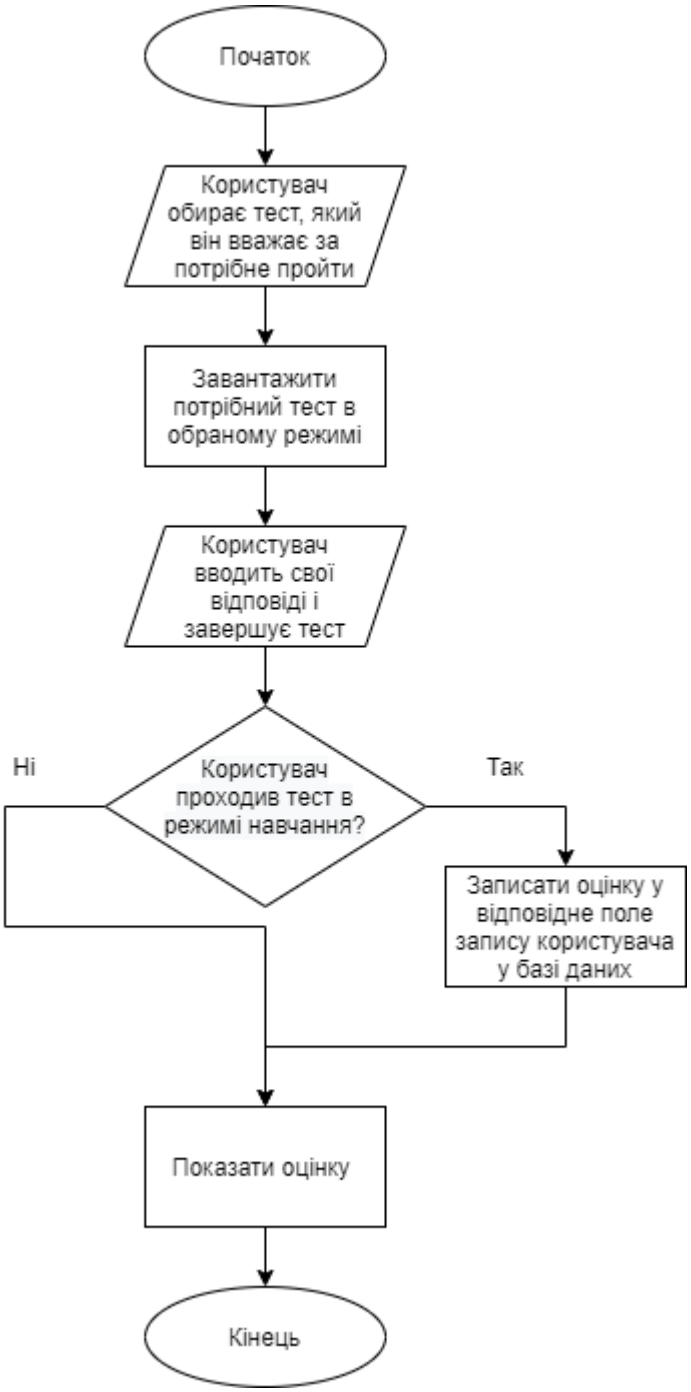


					ІАЛЦ.467200.004 Д1		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розробила		*Авдієнко І.О.*			*Система моделювання клітинних автоматів*	*Літ.*	*Аркуш*	*Аркушів*
Перевірив		*Роковий О.П.*				*Т*	*1*	*1*
					Пояснювальна записка	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-71*		
Н.контр.		*Сімоненко В.П.*						
Затв.		*Роковий О.П.*						

Додаток 2
ПРИНЦИПОВА СХЕМА РОБОТИ ПІДСИСТЕМИ ПРОХОДЖЕННЯ ТЕСТУ

до дипломного проєкту
на тему: «Система створення та проведення тестів з перевірки знань»

Київ – 2021 р.

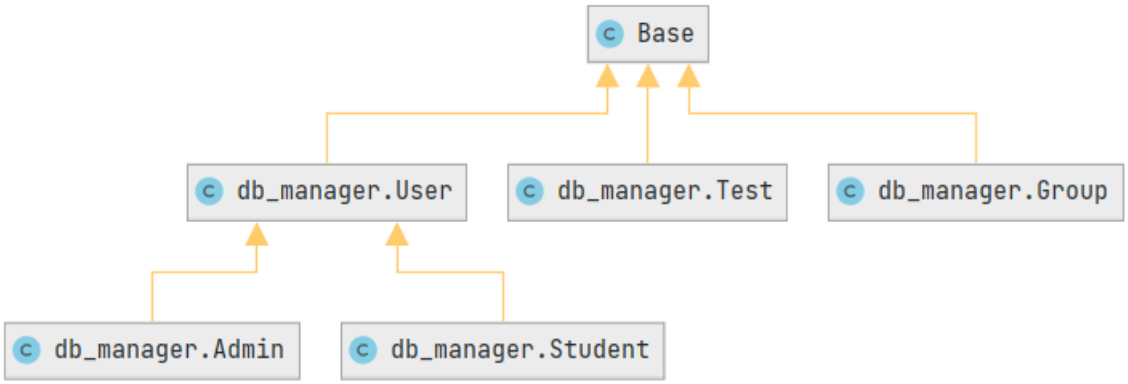


					ІАЛЦ.467200.004 Д2		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розробила		*Авдієнко І.О.*			*Система моделювання клітинних автоматів*	*Літ.*	*Аркуш*	*Аркушів*
Перевірив		*Роковий О.П.*				*Т*	*1*	*1*
Н.контр.		*Сімоненко В.П.*			*Пояснювальна записка*	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-71*	
Затв.		*Роковий О.П.*					

Додаток 3
ФУНКЦІОНАЛЬНА СХЕМА КЛАСІВ ОБ'ЄКТНОГО ВІДОБРАЖЕННЯ БАЗИ ДАННИХ

до дипломного проєкту
на тему: «Система створення та проведення тестів з перевірки знань»

Київ – 2021 р.



					ІАЛЦ.467200.004 ДЗ		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розробила		Авдієнко І.О.			Система моделювання клітинних автоматів	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Роковий О.П.				Т	
Н.контр.		Сімоненко В.П.			Пояснювальна записка	НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-71	
Затв.		Роковий О.П.					