МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ СТАТИЧНИХ ВЕБСАЙТІВ»**

Виконав: студент 4 курсу, групи 6.1212-пі-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.П. Сидоренко
(ініціали та прізвище)
Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Сидоренку Дмитру Павловичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтелектуальної системи для автоматичної генерації статичних вебсайтів

керівник роботи Лісняк Андрій Олександрович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 01.06.2026 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Відомості про технології React, FastAPI, Supabase.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Технічне завдання.
2. Проєктування вебзастосунку.
3. Програмна реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 09.12.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.12.2025	
2.	Збір вихідних даних.	26.01.2026	
3.	Обробка методичних та теоретичних	27.02.2026	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2026	
5.	Розробка третього розділу.	20.05.2026	
6.	Оформлення та нормоконтроль	25.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2026	

Студент _____ Д.П. Сидоренко
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтелектуальної системи для автоматичної генерації статичних вебсайтів»: 61 с., 19 рис., 2 табл., 26 джерел, 3 додатки.

ПОРТФОЛІО, FASTAPI, HTML, OPENAI API, PDF, POSTGRESQL, REACT, SUPABASE, TAILWIND CSS, VITE.

Об’єкт дослідження – процес автоматизованого створення персонального портфоліо засобами вебтехнологій та штучного інтелекту.

Мета роботи – розробити вебзастосунок для генерації персонального портфоліо на основі текстового опису користувача з можливістю збереження та експорту у PDF і HTML.

Методи дослідження – аналіз предметної області, проєктування вебзастосунків, клієнт-серверна взаємодія, робота з базами даних, інтеграція зовнішніх API та тестування.

У роботі розроблено вебзастосунок, який дозволяє користувачу зареєструватися, увійти до системи, ввести інформацію про себе, навички, досвід, проєкти та контакти, після чого система формує структуроване персональне портфоліо. Клієнтську частину реалізовано з використанням React, Vite та Tailwind CSS, серверну частину – на основі FastAPI. Для авторизації та збереження даних використано Supabase Auth і PostgreSQL. Розроблений вебзастосунок підтримує вибір стилю портфоліо, українську й англійську мови, світлу та темну теми, збереження результатів, перегляд і видалення портфоліо, а також експорт у PDF та HTML. Результатом роботи є функціональний вебзастосунок для автоматизованого створення, покращення, збереження та експорту персонального портфоліо.

SUMMARY

Bachelor’s qualifying paper “Development of an Intelligent System for Automatic Generation of Static Websites”: 61 pages, 19 figures, 2 table, 26 references, 3 supplements.

PORTFOLIO, FASTAPI, HTML, OPENAI API, PDF, POSTGRESQL, REACT, SUPABASE, TAILWIND CSS, VITE.

The object of the study is the process of automated creation of a personal portfolio using web technologies and artificial intelligence.

The aim of the study is to develop a web application for generating a personal portfolio based on a user’s text description with the ability to save results and export them in PDF and HTML formats.

The research methods include domain analysis, web application design, client-server interaction, database interaction, external API integration, and software testing.

The paper presents the development of a web application that allows a user to register, log in, enter information about skills, experience, projects, education, and contacts, after which the system generates a structured personal portfolio. The frontend part was implemented using React, Vite, and Tailwind CSS, while the backend part was developed using FastAPI. Supabase Auth and PostgreSQL were used for authentication and data storage. Portfolio generation and improvement were implemented through the OpenAI API. The developed web application supports portfolio style selection, Ukrainian and English languages, light and dark themes, saving results, viewing and deleting portfolios, and exporting the final version in PDF and HTML formats. As a result, a functional web application was developed for automated creation, improvement, storage, and export of a personal portfolio.

ЗМІСТ

Завдання на кваліфікаційну роботу ……2
Реферат 4
Summary 5
Вступ 8
1 Технічне завдання 11
1.1 Функціональні вимоги 11
1.1.1 Призначення і мета створення системи 12
1.1.2 Функціональні можливості системи 12
1.2 Нефункціональні вимоги 13
1.3 Опис предметної області 14
1.4 Опис системи 15
1.5 Огляд та порівняння аналогів 16
2 Проєктування 18
2.1 ER-модель бази даних та опис сутностей системи 18
2.2 Діаграма варіантів використання 19
2.2.1 Опис варіантів використання 20
2.3 Діаграма діяльності основного бізнес-процесу 21
2.4 Діаграма класів основних компонентів системи 23
2.5 Діаграма послідовності генерації портфоліо 24
2.6 Діаграма розгортання 25
2.7 Висновки до розділу 26
3 Програмна реалізація та тестування 27
3.1 Опис інструментів розробки 27
3.2 Основні компоненти вебзастосунку 28
3.3 Реалізація клієнтської частини 29
3.4 Реалізація серверної частини 30
3.5 Реалізація бази даних та авторизації 32

3.6 Інтеграція AI-модуля генерації портфоліо 34
3.7 Реалізація експорту портфоліо у PDF та HTML 35
3.8 Тестування вебзастосунку 37
3.9 Керівництво користувача 39
3.9.1 Реєстрація та вхід до системи 39
3.9.2 Генерація персонального портфоліо 40
3.9.3 Покращення портфоліо 40
3.9.4 Збереження та видалення портфоліо 41
3.9.5 Експорт портфоліо у PDF та HTML 42
3.9.6 Зміна мови та теми інтерфейсу 43
Висновки 45
Перелік посилань 47
Додаток А Фрагменти програмного коду frontend-частини 49
Додаток Б Фрагменти програмного коду backend-частини 53
Додаток В Фрагменти програмного коду експорту та роботи з базою даних 57

ВСТУП

У сучасних умовах розвитку інформаційних технологій персональне портфоліо є важливим інструментом для представлення професійних навичок, досвіду, виконаних проєктів та контактної інформації. Воно використовується студентами, розробниками, дизайнерами, фрилансерами та іншими спеціалістами для демонстрації власних можливостей потенційним роботодавцям або замовникам. Разом з тим, створення якісного портфоліо потребує часу, навичок структурування інформації, написання професійного опису та оформлення результату у зручному вигляді.

Актуальність роботи зумовлена необхідністю автоматизації процесу створення персонального портфоліо. Багато користувачів мають практичний досвід, навички та виконані проєкти, однак не завжди можуть правильно подати цю інформацію. Використання технологій штучного інтелекту дозволяє спростити цей процес, оскільки система може аналізувати текстовий опис користувача, виділяти основну інформацію та формувати структуроване портфоліо.

У межах кваліфікаційної роботи розглядається розробка вебзастосунку, який дозволяє користувачу створювати персональне портфоліо на основі звичайного текстового опису. Користувач вводить інформацію про себе, свої навички, освіту, досвід, проєкти та контакти, після чого система за допомогою AI-модуля формує готову структуру портфоліо. Додатково передбачено можливість покращення згенерованого тексту, вибору стилю портфоліо, збереження результатів та експорту фінальної версії у форматах PDF і HTML.

Метою кваліфікаційної роботи є розробка вебзастосунку для автоматизованої генерації персонального портфоліо з використанням технологій штучного інтелекту, механізмів авторизації, збереження даних та експорту результатів.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

- проаналізувати предметну область і сучасні засоби створення портфоліо;
- визначити функціональні вимоги до вебзастосунку;
- спроєктувати архітектуру системи;
- розробити клієнтську та серверну частини;
- реалізувати авторизацію користувачів і збереження портфоліо в базі даних;
- інтегрувати AI-модуль генерації та покращення портфоліо;
- реалізувати експорт результату у PDF та HTML;
- виконати тестування і розгортання вебзастосунку.

Об’єктом дослідження є процес автоматизованого створення персонального портфоліо користувача засобами вебтехнологій та штучного інтелекту.

Предметом дослідження є методи та програмні засоби розробки вебзастосунку для генерації, покращення, збереження та експорту персонального портфоліо.

У роботі використано методи аналізу предметної області, проєктування вебзастосунків, клієнт-серверної взаємодії, роботи з базами даних, інтеграції зовнішніх API та тестування програмного забезпечення.

Практичне значення роботи полягає у створенні функціонального вебзастосунку, який може бути використаний для швидкого формування персонального портфоліо без необхідності ручного написання всіх розділів та самостійного оформлення результату. Розроблений застосунок підтримує українську та англійську мови, світлу й темну теми інтерфейсу, збереження створених портфоліо та експорт у зручних для користувача форматах.

Підходи до проєктування, реалізації та тестування програмного забезпечення розглянуто у працях [1, 2, 10, 11].

Для реалізації frontend-частини використано React, Vite та Tailwind CSS. Backend-частину створено з використанням FastAPI. Для авторизації користувачів і збереження даних використано Supabase Auth та PostgreSQL.

Генерація текстового контенту виконується за допомогою OpenAI API. Розгортання вебзастосунку виконано з використанням Vercel для клієнтської частини та Render для серверної частини.

1 ТЕХНІЧНЕ ЗАВДАННЯ

У цьому розділі визначено основні вимоги до вебзастосунку для генерації персонального портфоліо з використанням технологій штучного інтелекту. Розглянуто призначення системи, її функціональні можливості, нефункціональні вимоги, предметну область, загальний опис системи та порівняння з наявними аналогами.

Розроблювана система призначена для автоматизації процесу створення персонального портфоліо. Користувач вводить текстовий опис про себе, свої навички, освіту, досвід, проєкти та контакти, після чого система формує структуроване портфоліо. Додатково передбачено можливість покращення згенерованого тексту, збереження портфоліо, експорту у PDF та HTML, а також зміни мови й теми інтерфейсу.

1.1 Функціональні вимоги

Під час формування вимог до системи було враховано загальні підходи до інженерії вимог [3].

Функціональні вимоги визначають перелік можливостей, які має надавати вебзастосунок користувачу.

Для системи генерації персонального портфоліо ці вимоги пов’язані з реєстрацією, авторизацією, введенням текстових даних, генерацією портфоліо за допомогою AI-модуля, збереженням результатів, експортом файлів та налаштуванням інтерфейсу.

У межах системи передбачено два основні стани користувача: гість і авторизований користувач.

Гість має доступ до форми реєстрації та входу. Авторизований користувач отримує доступ до основного функціоналу вебзастосунку.

1.1.1 Призначення і мета створення системи

Функціональне призначення системи – надати користувачу зручний інструмент для швидкого створення персонального портфоліо на основі звичайного текстового опису.

Експлуатаційне призначення – система використовується студентами, розробниками, фрилансерами та іншими спеціалістами, які потребують готового портфоліо для демонстрації професійних навичок, досвіду та виконаних проєктів.

Мета створення системи – розробити вебзастосунок, який автоматизує процес формування персонального портфоліо, дозволяє покращувати його зміст за допомогою штучного інтелекту, зберігати результати та експортувати фінальну версію у зручних форматах.

1.1.2 Функціональні можливості системи

Система має надавати гостю такі можливості:

– створити обліковий запис;
– підтвердити електронну пошту;
– увійти до облікового запису;
– переглянути стартову сторінку вебзастосунку.

Авторизованому користувачу система має надавати такі можливості:

– вийти з облікового запису;
– ввести текстовий опис про себе, навички, освіту, досвід, проєкти та контакти;
– вибрати стиль майбутнього портфоліо;
– згенерувати персональне портфоліо за допомогою AI-модуля; покращити вже створене портфоліо;
– переглянути результат генерації у вебінтерфейсі;

– зберегти портфоліо у базі даних;
– переглянути список збережених портфоліо;
– відкрити раніше збережене портфоліо;
– видалити непотрібне портфоліо;
– експортувати портфоліо у формат PDF;
– експортувати портфоліо у формат HTML;
– змінити мову інтерфейсу;
– перемкнути світлу і темну тему.

Основний сценарій роботи системи полягає в тому, що користувач авторизується, вводить початкову інформацію, обирає стиль портфоліо та запускає генерацію. Після отримання результату користувач може покращити текст, зберегти портфоліо або експортувати його у файл.

1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають загальні характеристики вебзастосунку, які впливають на якість його роботи, зручність використання, безпеку, продуктивність, надійність і можливість подальшого розвитку.

Вимоги до інтерфейсу користувача. Інтерфейс системи має бути простим, зрозумілим і зручним для користувачів, які не мають спеціальних технічних знань. Основні дії мають бути доступні з головної сторінки після входу до системи.

Вимоги до адаптивності. Вебзастосунок має коректно відображатися на персональних комп’ютерах, ноутбуках, планшетах і мобільних пристроях. Елементи інтерфейсу повинні зберігати зручність використання на різних розмірах екрана.

Вимоги до продуктивності. Система повинна швидко реагувати на дії користувача. Генерація портфоліо може займати певний час через звернення до зовнішнього AI API, тому користувачу має відображатися стан завантаження.

Вимоги до безпеки. Доступ до збережених портфоліо має бути доступний лише авторизованому користувачу. Пароль користувача повинен відповідати мінімальним вимогам складності. Конфіденційні ключі доступу до зовнішніх сервісів мають зберігатися у змінних середовища та не повинні потрапляти до відкритого коду.

Вимоги до надійності. Система має коректно обробляти типові помилки: неправильні дані авторизації, відсутність інтернет-з'єднання, помилки генерації, помилки збереження або експорту. Користувач повинен отримувати зрозуміле повідомлення про проблему.

Вимоги до локалізації. Інтерфейс має підтримувати українську та англійську мови. Обрана мова повинна зберігатися після оновлення сторінки.

Вимоги до масштабованості. Архітектура системи має передбачати можливість подальшого розширення, зокрема додавання нових стилів портфоліо, форматів експорту, шаблонів оформлення та додаткових AI-функцій.

1.3 Опис предметної області

Предметною областю розроблюваної системи є автоматизоване створення персонального портфоліо засобами вебтехнологій і штучного інтелекту. Персональне портфоліо використовується для представлення професійних навичок, досвіду, освіти, проєктів, контактних даних та інших відомостей про користувача.

Портфоліо особливо важливе для спеціалістів у сфері інформаційних технологій, дизайнерів, фрілансерів, студентів і початківців, які хочуть показати власні практичні результати. На відміну від звичайного резюме, портфоліо дозволяє не лише перелічити навички, а й продемонструвати виконані проєкти, технології та професійний напрям розвитку.

Основною проблемою предметної області є те, що користувач не завжди може самостійно правильно структурувати інформацію про себе. Часто

складно сформулювати професійний опис, виділити ключові навички, описати проєкти та оформити результат у зручному вигляді. Через це процес створення портфоліо може займати багато часу.

Використання AI-модуля дозволяє автоматизувати частину цього процесу. Користувач вводить звичайний текстовий опис, а система перетворює його на структуроване портфоліо. Такий підхід спрощує створення портфоліо та робить систему доступною для користувачів без навичок дизайну або професійного написання текстів.

У межах розроблюваної системи портфоліо розглядається як структурований набір даних, що містить ім’я користувача, професію, короткий опис, навички, проєкти, освіту, досвід, мови, досягнення, контакти та додаткові відомості.

1.4 Опис системи

Розроблювана система є клієнт-серверним вебзастосунком для автоматизованої генерації персонального портфоліо. Вона дозволяє користувачу створити обліковий запис, увійти до системи, ввести опис про себе та отримати готову структуру портфоліо.

Клієнтська частина системи реалізована з використанням React, Vite та Tailwind CSS. Вона відповідає за відображення інтерфейсу, введення даних користувача, вибір стилю портфоліо, перемикання теми й мови, перегляд результату, збереження портфоліо та експорт у файли.

Серверна частина реалізована з використанням FastAPI. Вона приймає запити від клієнтської частини, передає дані до OpenAI API та повертає результат генерації у структурованому вигляді. Серверна частина також використовується для функції покращення портфоліо.

Для авторизації користувачів і збереження даних використовується Supabase. Авторизація забезпечує доступ користувача до власних даних, а база даних PostgreSQL використовується для збереження створених портфоліо.

Основними модулями системи є:

– модуль авторизації;
– модуль введення користувацького опису;
– модуль генерації портфоліо;
– модуль покращення портфоліо;
– модуль збереження портфоліо;
– модуль перегляду збережених портфоліо;
– модуль експорту у PDF;
– модуль експорту у HTML;
– модуль зміни мови;
– модуль зміни теми інтерфейсу.

Основний бізнес-процес системи полягає в тому, що користувач вводить початкову інформацію про себе, обирає стиль портфоліо та натискає кнопку генерації. Після цього система формує структуроване портфоліо. Користувач може покращити отриманий результат, зберегти його або експортувати у PDF чи HTML.

1.5 Огляд та порівняння аналогів

На ринку існує багато сервісів, які частково вирішують задачу створення портфоліо або резюме. До таких рішень можна віднести Canva, Wix, Notion, LinkedIn та ChatGPT. Кожен із цих інструментів має власні переваги, однак не всі вони забезпечують повний цикл створення портфоліо з AI-генерацією, збереженням результатів і експортом у кілька форматів.

Canva дозволяє створювати резюме та портфоліо на основі готових шаблонів. Її перевагою є велика кількість візуальних шаблонів, однак користувач має самостійно писати текст і вручну редагувати структуру.

Wix дозволяє створювати повноцінні сайти-портфоліо. Проте такий підхід потребує більше часу на налаштування сторінок, вибір шаблонів, структурування блоків і публікацію сайту.

Notion може використовуватися для створення простого портфоліо або сторінки з інформацією про користувача. Його перевагою є гнучкість, однак він не орієнтований саме на автоматичну генерацію портфоліо.

LinkedIn дозволяє користувачу представити професійний профіль, досвід і навички, але не створює окремий файл портфоліо у форматах PDF або HTML за введеним описом.

ChatGPT може генерувати текст для портфоліо, однак сам по собі не забезпечує авторизацію, збереження результатів, готовий інтерфейс перегляду та експорт портфоліо.

У таблиці 1.1 наведено порівняння аналогів із розробленою системою.

Таблиця 1.1 – Порівняння аналогів

Критерій	**Canva**	**Wix**	**Notion**	**LinkedIn**	**ChatGPT**	**Система**
Створення портфоліо	+	+	+	частк.	частк.	+
AI-генерація	частк.	частк.	–	–	+	+
Авторизація користувача	+	+	+	+	+	+
Збереження портфоліо	+	+	+	+	частк.	+
Покращення тексту	–	частк.	–	–	+	+
Експорт у PDF	+	частк.	+	частк.	–	+
Експорт у HTML	–	частк.	частк.	–	–	+
Підтримка мов	+	+	+	+	+	+
Простота	+	середня	+	+	середня	+

Отже, аналіз аналогів показав, що більшість наявних рішень або орієнтована на ручне створення портфоліо, або забезпечує лише генерацію тексту без повного циклу роботи з результатом. Розроблювана система поєднує AI-генерацію, покращення тексту, авторизацію, збереження портфоліо та експорт у PDF і HTML.

2 ПРОЄКТУВАННЯ

У цьому розділі розглянуто проєктування вебзастосунку для генерації персонального портфоліо. Проєктування системи включає визначення структури бази даних, основних сутностей, сценаріїв взаємодії користувача із системою, а також побудову UML-діаграм, які відображають логіку роботи вебзастосунку.

Для моделювання структури та поведінки системи використано підходи UML-моделювання [4, 5].

Розроблювана система побудована за клієнт-серверною архітектурою. Клієнтська частина забезпечує взаємодію з користувачем, серверна частина відповідає за обробку запитів до AI-модуля, а база даних використовується для збереження створених портфоліо та зв’язку їх із конкретним користувачем.

2.1 ER-модель бази даних та опис сутностей системи

Для збереження даних у системі використовується база даних Supabase PostgreSQL. Основними сутностями системи є користувач та портфоліо. Користувач створюється засобами Supabase Auth, а створені ним портфоліо зберігаються в окремій таблиці portfolios.

Сутність auth.users використовується для збереження даних зареєстрованих користувачів. Вона містить унікальний ідентифікатор користувача, електронну пошту та службові дані, необхідні для авторизації.

Сутність portfolios призначена для збереження створених користувачем портфоліо. Основними полями таблиці є:

– id – унікальний ідентифікатор портфоліо;
– user_id – ідентифікатор користувача, якому належить портфоліо;
– title – назва портфоліо;

– data – повна структура портфоліо у форматі JSON;

– created_at – дата й час створення запису.

Використання типу JSONB для поля data є доцільним, оскільки структура портфоліо може містити різні секції: ім’я, професію, опис, навички, проєкти, освіту, досвід, мови, досягнення, контакти та інші дані. Такий підхід дозволяє гнучко зберігати результат AI-генерації без створення великої кількості додаткових таблиць.

Між сутностями auth.users та portfolios існує зв’язок один-до-багатьох. Один користувач може мати декілька збережених портфоліо, але кожне портфоліо належить лише одному користувачу.

На рисунку 2.1 наведено ER-модель бази даних вебзастосунку.

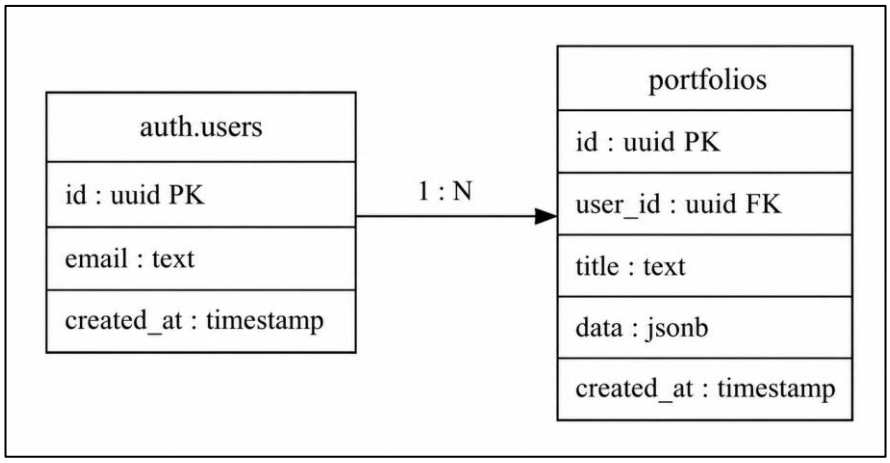


Рисунок 2.1 – ER-модель бази даних вебзастосунку

2.2 Діаграма варіантів використання

Діаграма варіантів використання відображає основні можливості системи з точки зору користувача. У розроблюваному вебзастосунку основним актором є користувач. Він може зареєструватися, увійти до системи, згенерувати портфоліо, покращити його, зберегти результат, переглянути раніше створені портфоліо, видалити непотрібні записи та виконати експорт у PDF або HTML.

На рисунку 2.2 наведено діаграму варіантів використання вебзастосунку.

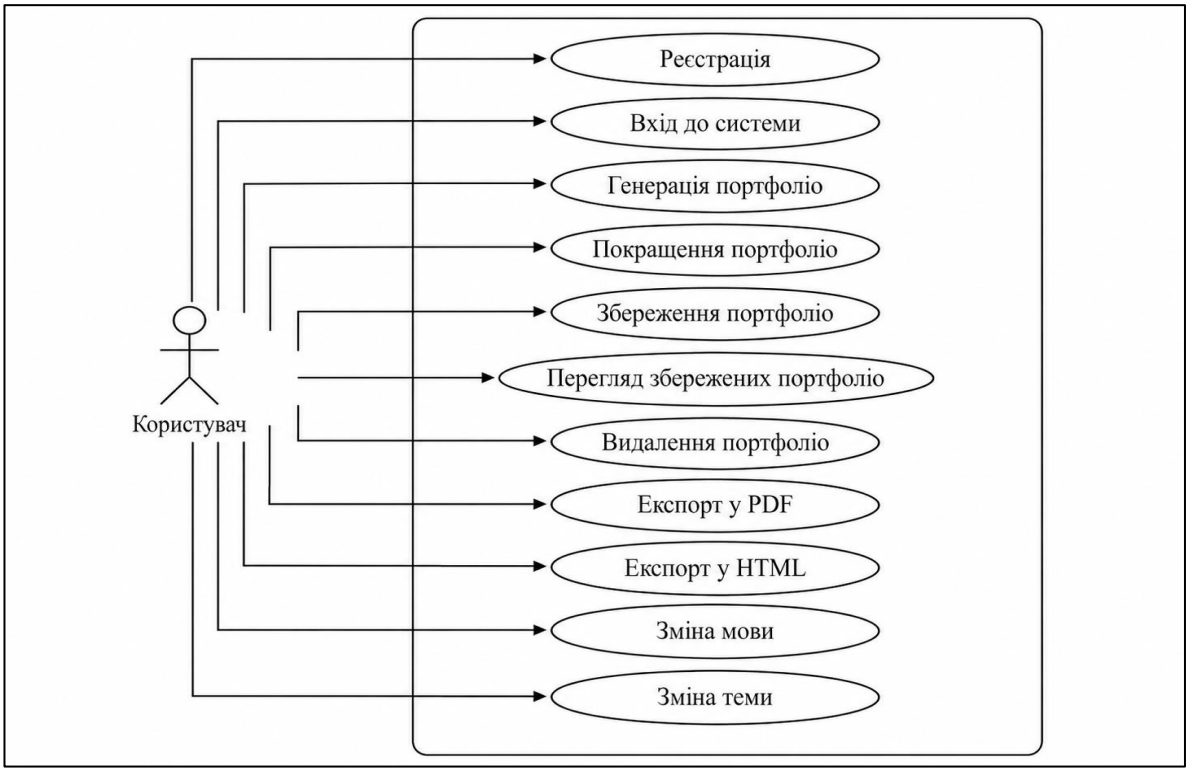


Рисунок 2.2 – Діаграма варіантів використання вебзастосунку

2.2.1 Опис варіантів використання

Основними варіантами використання системи є реєстрація користувача, вхід до системи, генерація портфоліо, покращення портфоліо, збереження результату, перегляд збережених портфоліо, видалення портфоліо, експорт у PDF, експорт у HTML, зміна мови та зміна теми інтерфейсу.

Варіант використання «Реєстрація» дозволяє новому користувачу створити обліковий запис у системі. Під час реєстрації користувач вводить електронну пошту та пароль. Система перевіряє коректність введених даних і створює новий обліковий запис.

Варіант використання «Вхід до системи» дозволяє зареєстрованому користувачу отримати доступ до основних функцій вебзастосунку. Після

успішного входу користувач може створювати, зберігати та експортувати портфоліо.

Варіант використання «Генерація портфоліо» є основним сценарієм роботи системи. Користувач вводить текстовий опис про себе, обирає стиль портфоліо та запускає генерацію. Система передає дані до AI-модуля та формує структуроване портфоліо.

Варіант використання «Покращення портфоліо» дозволяє користувачу покращити вже створений результат. AI-модуль обробляє поточне портфоліо та робить текст більш професійним, зберігаючи основну структуру.

Варіант використання «Збереження портфоліо» дозволяє користувачу зберегти створений результат у базі даних. Після збереження портфоліо з’являється у списку збережених результатів.

Варіант використання «Перегляд збережених портфоліо» дозволяє користувачу переглядати раніше створені портфоліо та відкривати їх для повторної роботи.

Варіант використання «Видалення портфоліо» дозволяє користувачу видалити непотрібне портфоліо зі списку збережених результатів.

Варіант використання «Експорт у PDF» дозволяє сформувати файл портфоліо у форматі PDF. Такий файл може бути використаний як резюме або документ для надсилання роботодавцю.

Варіант використання «Експорт у HTML» дозволяє сформувати HTML-файл, який можна відкрити у браузері як окрему сторінку портфоліо.

Варіанти використання «Зміна мови» та «Зміна теми» забезпечують персоналізацію інтерфейсу. Користувач може обрати українську або англійську мову, а також світлу або темну тему.

2.3 Діаграма діяльності основного бізнес-процесу

Діаграма діяльності відображає послідовність дій, які виконуються під час основного бізнес-процесу системи – генерації персонального портфоліо.

Цей процес починається з авторизації користувача та введення початкової інформації про себе. Після цього користувач обирає стиль портфоліо та натискає кнопку генерації.

Система перевіряє наявність введеного опису. Якщо текст відсутній або містить недостатньо інформації, користувач отримує повідомлення про необхідність заповнення поля. Якщо дані введені коректно, frontend-частина передає запит до backend-частини. Backend формує запит до OpenAI API, отримує структуровану відповідь та повертає її клієнтській частині.

Після отримання результату користувач може переглянути сформоване портфоліо, покращити його, зберегти в базі даних або експортувати у PDF чи HTML. Таким чином, діаграма діяльності дозволяє описати логіку взаємодії користувача із системою під час створення портфоліо.

На рисунку 2.3 наведено діаграму діяльності основного бізнес-процесу

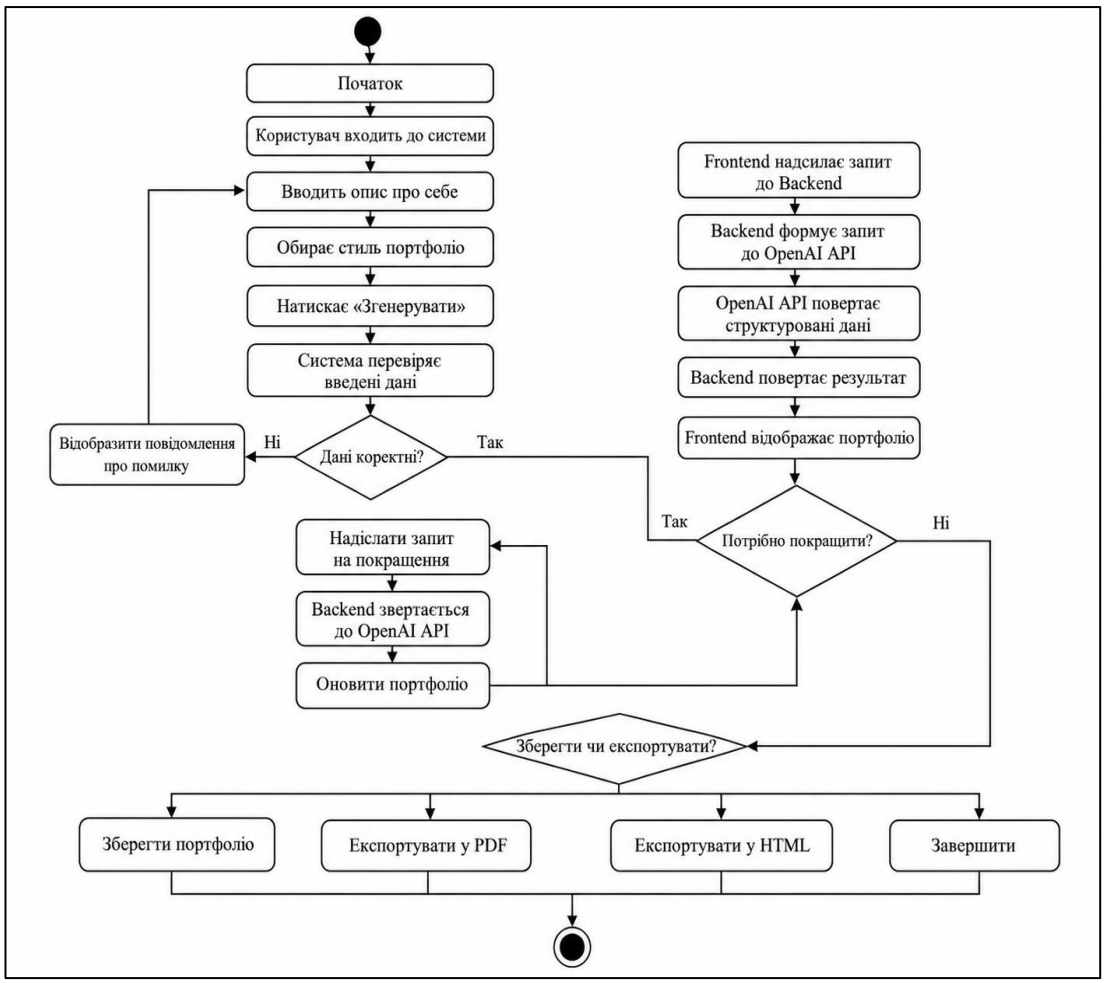


Рисунок 2.3 – Діаграма діяльності основного бізнес-процесу

2.4 Діаграма класів основних компонентів системи

Діаграма класів використовується для відображення основних структурних компонентів системи та зв’язків між ними. Оскільки розроблюваний вебзастосунок побудовано на основі клієнт-серверної архітектури, доцільно виділити основні компоненти, які відповідають за авторизацію, генерацію портфоліо, збереження даних та експорт результатів. На рисунку 2.4 наведено діаграму класів основних компонентів системи.

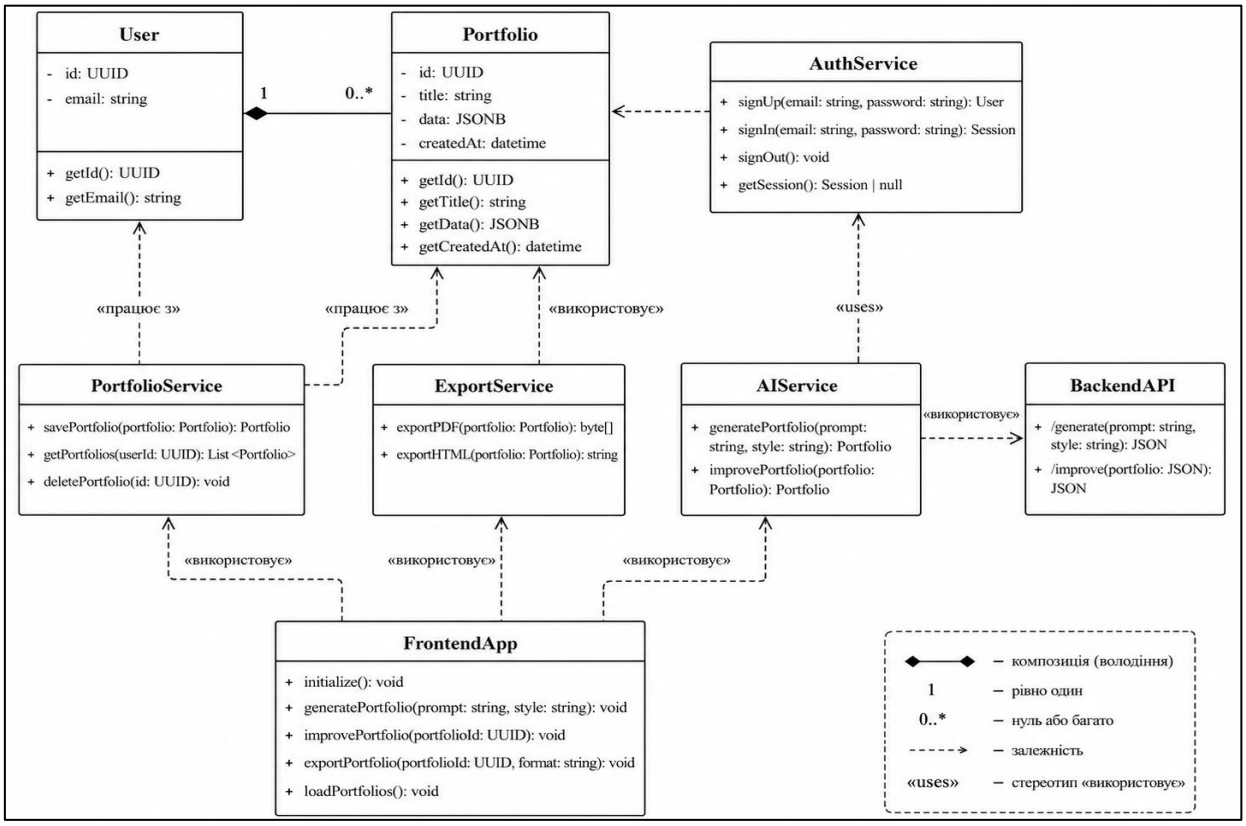


Рисунок 2.4 – Діаграма класів основних компонентів системи

До основних компонентів системи належать:

– AuthService – відповідає за реєстрацію, вхід, вихід користувача та перевірку поточної сесії;
– PortfolioService – забезпечує збереження, завантаження та видалення портфоліо;
– AIService – відповідає за взаємодію з backend-частиною та AI-модулем;

– ExportService – забезпечує експорт портфоліо у PDF та HTML;
– Portfolio – модель даних, що описує структуру персонального портфоліо;
– User – модель користувача системи.

Клас Portfolio містить основні поля портфоліо: ім’я, професію, опис, навички, проєкти, освіту, досвід, мови, досягнення та контакти. Клас User використовується для зв’язку збережених портфоліо з конкретним користувачем.

2.5 Діаграма послідовності генерації портфоліо

Діаграма послідовності відображає порядок взаємодії між користувачем, клієнтською частиною, серверною частиною, OpenAI API та базою даних під час генерації портфоліо (див. рис. 2.5).

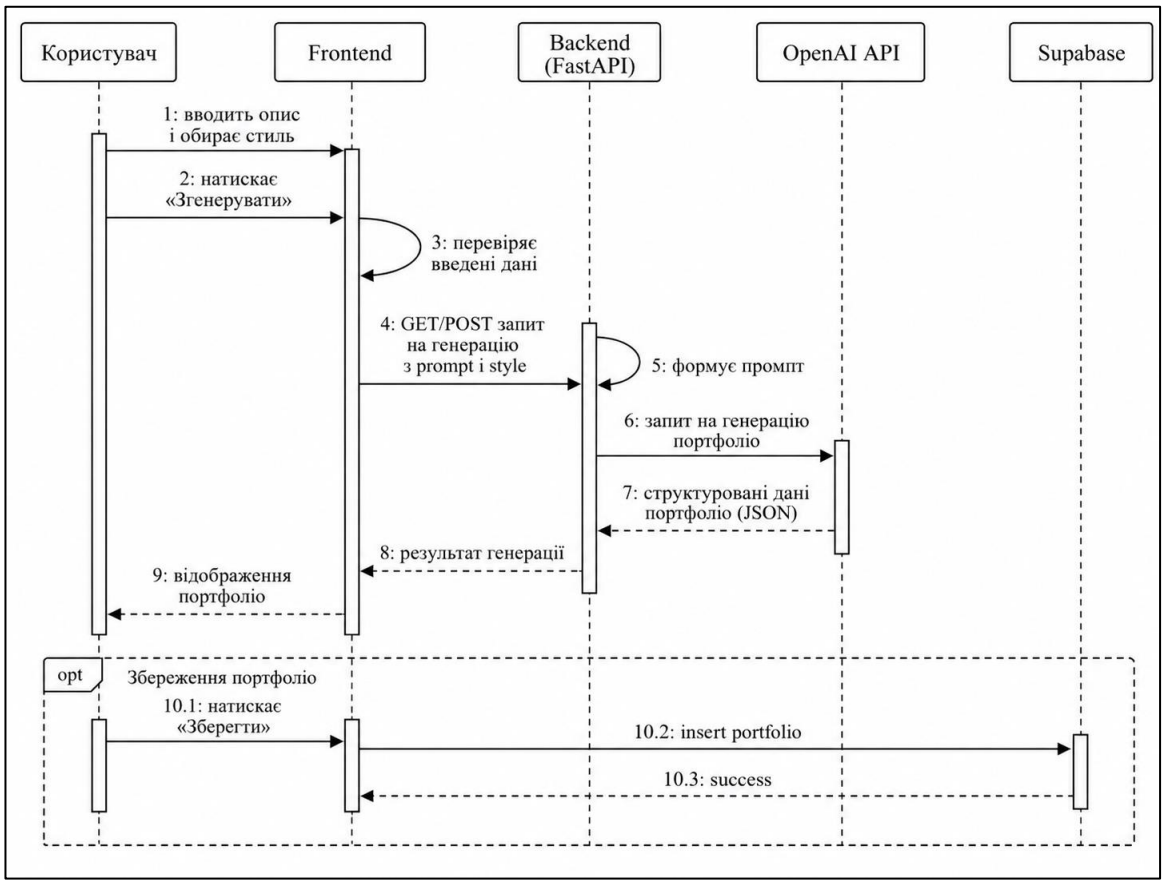


Рисунок 2.5 – Діаграма послідовності генерації портфоліо

Спочатку користувач вводить текстовий опис і натискає кнопку генерації. Frontend-частина формує запит і передає його до backend-частини. Backend обробляє отримані дані, формує звернення до OpenAI API та очікує відповідь. Після отримання згенерованих даних backend повертає їх клієнтській частині у структурованому форматі.

Після відображення результату користувач може зберегти портфоліо. У цьому випадку frontend передає дані до Supabase, де вони зберігаються у таблиці portfolios із прив’язкою до поточного користувача.

2.6 Діаграма розгортання

Діаграма розгортання показує фізичне розміщення основних компонентів системи та їх взаємодію у хмарному середовищі. Розроблюваний вебзастосунок складається з клієнтської частини, серверної частини, бази даних, модуля авторизації та зовнішнього AI API. На рисунку 2.6 наведено діаграму розгортання вебзастосунку.

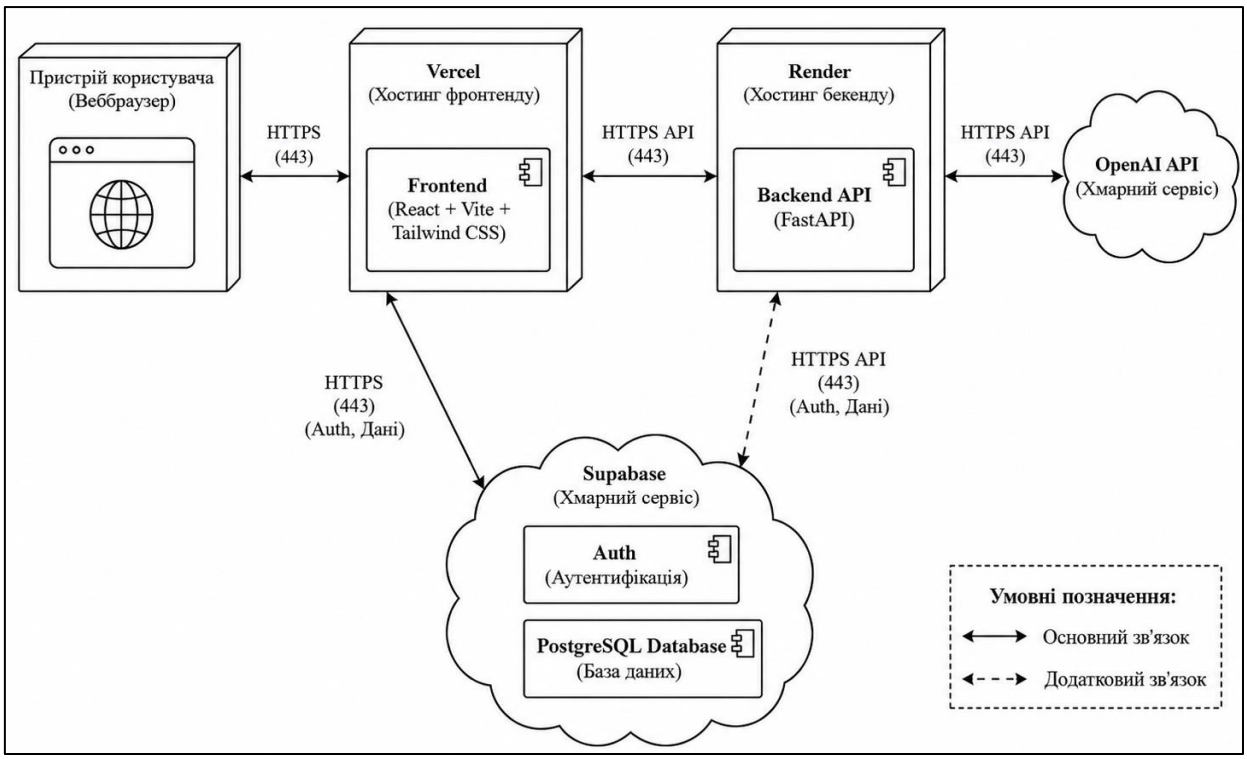


Рисунок 2.6 – Діаграма розгортання вебзастосунку

Клієнтська частина розміщена на платформі Vercel. Вона доступна користувачу через браузер і відповідає за відображення інтерфейсу, введення даних, перегляд портфоліо та виконання дій користувача.

Серверна частина розміщена на платформі Render. Вона реалізована за допомогою FastAPI та відповідає за обробку запитів генерації й покращення портфоліо. Для створення текстового контенту серверна частина звертається до OpenAI API.

Для авторизації користувачів і збереження портфоліо використовується Supabase. Supabase Auth забезпечує реєстрацію та вхід користувачів, а Supabase PostgreSQL використовується для збереження створених портфоліо.

Архітектура системи побудована з урахуванням принципів організації програмних систем [6–9].

2.7 Висновки до розділу

У цьому розділі було виконано проєктування вебзастосунку для генерації персонального портфоліо. Було описано структуру бази даних, основні сутності системи та зв’язок між користувачем і створеними портфоліо.

Також було розглянуто основні сценарії взаємодії користувача із системою, описано варіанти використання, логіку основного бізнес-процесу, структуру основних компонентів, послідовність генерації портфоліо та особливості розгортання системи у хмарному середовищі.

Результати проєктування стали основою для подальшої програмної реалізації клієнтської частини, серверної частини, інтеграції AI-модуля, роботи з базою даних та реалізації експорту портфоліо у PDF і HTML.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

У цьому розділі розглянуто програмну реалізацію вебзастосунку для генерації персонального портфоліо. Описано основні інструменти розробки, структуру клієнтської та серверної частин, роботу з базою даних, авторизацію користувачів, інтеграцію AI-модуля, експорт портфоліо у PDF та HTML, а також тестування основних функцій системи.

Розроблений вебзастосунок складається з клієнтської частини, серверної частини, бази даних, модуля авторизації та інтеграції із зовнішнім AI API. Клієнтська частина відповідає за взаємодію з користувачем, серверна частина виконує обробку запитів до AI-модуля, а база даних забезпечує збереження створених портфоліо.

3.1 Опис інструментів розробки

Для реалізації клієнтської частини вебзастосунку було використано бібліотеку React. Вона дозволяє створювати компонентний інтерфейс користувача та зручно керувати станом застосунку. Для швидкого запуску та збирання frontend-частини використано Vite, який забезпечує швидке середовище розробки.

Для стилізації інтерфейсу використано Tailwind CSS. Цей інструмент дозволяє швидко створювати адаптивний інтерфейс за допомогою готових CSS-класів. Завдяки цьому було реалізовано світлу й темну теми, адаптивне розміщення блоків, кнопки, форми введення та картки портфоліо.

Серверна частина розроблена за допомогою FastAPI. Цей фреймворк використовується для створення API, яке приймає запити від frontend-частини, передає дані до OpenAI API та повертає результат генерації у структурованому форматі.

Для авторизації користувачів і збереження даних використано Supabase. Supabase Auth забезпечує реєстрацію, вхід та керування сесією користувача. Supabase PostgreSQL використовується для збереження створених портфоліо.

Для генерації та покращення портфоліо використано OpenAI API. Він отримує текстовий опис користувача, стиль портфоліо та мову генерації, після чого повертає структурований результат, який відображається у вебінтерфейсі.

Для контролю версій використано Git і GitHub. Розгортання клієнтської частини виконано на платформі Vercel, а серверної частини – на Render.

3.2 Основні компоненти вебзастосунку

Вебзастосунок має модульну структуру, що спрощує підтримку та подальше розширення системи. Основні компоненти системи відповідають за авторизацію, генерацію портфоліо, збереження даних, експорт результатів і налаштування інтерфейсу.

До основних компонентів клієнтської частини належать:

- компонент авторизації, який забезпечує реєстрацію та вхід користувача;
- компонент AI Assistant, у якому користувач вводить опис про себе та обирає стиль портфоліо;
- компонент попереднього перегляду портфоліо, який відображає результат генерації;
- компонент списку збережених портфоліо;
- компонент перемикання мови інтерфейсу;
- компонент перемикання світлої та темної теми;
- функції експорту портфоліо у PDF та HTML.

Серверна частина містить API-маршрути для генерації та покращення портфоліо. Вона отримує запит від клієнтської частини, формує промпт для AI-модуля, звертається до OpenAI API та повертає результат у форматі JSON.

База даних містить таблицю portfolios, у якій зберігаються створені користувачами портфоліо. Кожен запис пов’язаний з конкретним користувачем через поле user_id. Це дозволяє забезпечити доступ користувача лише до власних збережених результатів.

Основна логіка вебзастосунку полягає в тому, що користувач вводить дані, система генерує портфоліо, після чого користувач може покращити його, зберегти або експортувати у файл.

3.3 Реалізація клієнтської частини

Клієнтська частина вебзастосунку реалізована з використанням React, Vite та Tailwind CSS. Вона відповідає за відображення інтерфейсу користувача, обробку введених даних, взаємодію з backend-частиною, роботу із Supabase та формування файлів експорту.

Основним елементом клієнтської частини є головна сторінка застосунку, на якій розміщено блок AI Assistant, область попереднього перегляду портфоліо та список збережених портфоліо. Така структура дозволяє користувачу одночасно вводити опис, бачити результат генерації та працювати зі збереженими даними.

Блок AI Assistant містить поле для введення текстового опису користувача, список вибору стилю портфоліо та кнопку запуску генерації. Користувач може описати свої навички, досвід, освіту, проєкти, контакти та побажання щодо оформлення. Після натискання кнопки генерації дані передаються до серверної частини.

У клієнтській частині реалізовано обробку станів завантаження. Під час генерації портфоліо або покращення тексту відповідні кнопки тимчасово блокуються, а користувач бачить повідомлення про виконання операції. Це запобігає повторному надсиланню однакових запитів і робить роботу із застосунком зрозумілішою.

Для відображення результату використовується компонент попереднього перегляду портфоліо. Він формує структуроване подання даних: ім’я, професію, короткий опис, навички, проєкти, освіту, досвід, мови, досягнення та контакти. Якщо певні секції відсутні у згенерованому результаті, вони не відображаються, що дозволяє уникнути порожніх блоків.

Окремо реалізовано підтримку світлої та темної теми інтерфейсу. Перемикання теми змінює кольорову схему сторінки, карток, кнопок і полів введення. Також реалізовано перемикання мови інтерфейсу між українською та англійською. Обрана мова впливає не лише на підписи елементів, а й на мову генерації портфоліо.

Клієнтська частина також взаємодіє із Supabase. За допомогою Supabase-клієнта виконується авторизація користувача, отримання поточної сесії, збереження портфоліо, завантаження списку збережених результатів і видалення непотрібних записів.

Під час розробки інтерфейсу враховано загальні принципи зручності користування та проєктування користувацьких інтерфейсів [15–17].

На рисунку 3.1 наведено фрагмент клієнтської частини, що відповідає за ініціалізацію Supabase-клієнта.

```
import { createClient } from "@supabase/supabase-js";

const supabaseUrl = "VITE_SUPABASE_URL";
const supabaseKey = "VITE_SUPABASE_ANON_KEY";

export const supabase = createClient(
  supabaseUrl,
  supabaseKey
);
```

Рисунок 3.1

3.4 Реалізація серверної частини

Серверна частина вебзастосунку реалізована за допомогою FastAPI. Вона використовується як проміжний рівень між клієнтською частиною та

OpenAI API. Такий підхід дозволяє не зберігати ключ OpenAI API у frontend-коді та централізовано обробляти запити генерації.

Основними задачами серверної частини є приймання запитів від клієнта, формування промпта для AI-модуля, звернення до OpenAI API, обробка відповіді та повернення результату у форматі JSON. Серверна частина також відповідає за покращення вже створеного портфоліо. На рисунку 3.2 наведено фрагмент серверної частини, який відповідає за генерацію портфоліо.

```
@app.get("/generate")
def generate_portfolio(prompt: str, style: str = "Minimal", language: str = "uk"):
    output_language = LANGUAGE_NAMES.get(language, "Ukrainian")

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {
                "role": "system",
                "content": f"""
Ти AI-агент для генерації професійного персонального портфоліо.

Завжди повертай ТІЛЬКИ валідний JSON без Markdown, без пояснень, без коментарів і без тексту поза JSON.

Обраний стиль портфоліо: {style}.
Мова портфоліо: {output_language}.

Ураховуй обраний стиль під час генерації всіх секцій портфоліо: bio, skills, projects, education, experience,
achievements, theme та інших полів.

Пиши всі текстові значення на мові: {output_language}.
Не змішуй мови в одному результаті.

Обов'язково:
- заповнюй усі основні поля;
- завжди генеруй масив skills;
- завжди генеруй хоча б один project;
- завжди повертай contacts.email, contacts.github і contacts.phone;
- завжди повертай education, experience, languages, achievements, softSkills, certificates і links;
- якщо для розширених секцій немає даних – використовуй порожній масив;
- якщо контактні дані не вказані – залишай порожній рядок;
- не вигадуй реальні персональні дані, якщо користувач їх не надав;
- структура відповіді має повністю відповідати заданому JSON-шаблону;
- не додавай зайві поля, яких немає у структурі.

Структура JSON:
{PORTFOLIO_JSON_STRUCTURE}
"""
            },
            {
                "role": "user",
                "content": prompt
            }
        ]
    )

    content = response.choices[0].message.content

    return parse_json_response(content)
```

Рисунок 3.2

Для генерації портфоліо клієнтська частина передає текстовий опис користувача, вибраний стиль і мову. Backend формує інструкцію для AI-модуля таким чином, щоб відповідь поверталася у структурованому вигляді. Це дозволяє frontend-частині коректно відображати отримані секції портфоліо.

Під час покращення портфоліо серверна частина отримує вже сформовану структуру даних і передає її до AI-модуля з інструкцією покращити формулювання. Основна задача цієї функції полягає у тому, щоб зробити опис користувача та проєктів більш професійним, не змінюючи загальну структуру портфоліо.

У серверній частині також передбачено обробку можливих помилок. Якщо OpenAI API не повертає коректну відповідь або виникає проблема під час виконання запиту, backend має повернути повідомлення про помилку, яке може бути відображене користувачу в інтерфейсі.

3.5 Реалізація бази даних та авторизації

Для авторизації користувачів і збереження створених портфоліо використано Supabase. Цей сервіс поєднує механізм автентифікації користувачів і базу даних PostgreSQL, що спрощує реалізацію облікових записів та збереження даних.

Авторизація користувача реалізована за допомогою Supabase Auth. Користувач може створити обліковий запис, підтвердити електронну пошту, увійти до системи та вийти з облікового запису. Після входу система отримує поточну сесію користувача та використовує її для доступу до персональних даних.

Під час реєстрації реалізовано перевірку складності пароля. Пароль має містити від 10 до 50 символів, щонайменше одну велику літеру та одну цифру. Це підвищує рівень безпеки облікових записів і зменшує ризик використання занадто простих паролів.

Для збереження портфоліо використовується таблиця portfolios. Кожен запис містить унікальний ідентифікатор, ідентифікатор користувача, назву портфоліо, структуру даних у форматі JSONB та дату створення. Поле user_id дозволяє пов’язати портфоліо з конкретним користувачем. На рисунку 3.3 наведено фрагмент коду збереження портфоліо у базу даних.

```
const savePortfolio = async () => {
  if (!portfolio || !session?.user?.id) {
    return;
  }

  setSaveLoading(true);

  try {
    const title = getPortfolioTitle(portfolio, t);

    const { data, error } = await supabase
      .from("portfolios")
      .insert({
        user_id: session.user.id,
        title,
        data: portfolio,
      })
      .select("id,title,data,created_at")
      .single();

    if (error) {
      throw error;
    }

    setSavedPortfolios((current) => [data, ...current]);
    setSelectedPortfolioId(data.id);
  } catch (error) {
    console.error(error);
    alert(t.saveError);
  } finally {
    setSaveLoading(false);
  }
};
```

Рисунок 3.3

Збереження даних у форматі JSONB дозволяє зберігати повну структуру портфоліо без створення окремих таблиць для кожної секції. Це зручно для системи, у якій структура результату може змінюватися залежно від введеного опису користувача.

Для обмеження доступу до даних використовується зв’язок портфоліо з поточним користувачем. Користувач може переглядати, зберігати та видаляти

лише власні портфоліо. Такий підхід забезпечує базовий рівень захисту персональних даних.

Проєктування структури даних виконано з урахуванням принципів організації реляційних баз даних [12–14].

3.6 Інтеграція AI-модуля генерації портфоліо

Однією з основних функцій вебзастосунку є генерація персонального портфоліо за допомогою AI-модуля. Для цього використано OpenAI API, який отримує текстовий опис користувача та повертає структурований результат.

Користувач вводить інформацію про себе у довільній формі: професійний напрям, навички, освіту, досвід, проєкти, мови, досягнення та контактні дані. Також користувач може обрати стиль портфоліо, наприклад мінімалістичний, сучасний, корпоративний, креативний або професійний. Обраний стиль передається разом із текстом до серверної частини.

Серверна частина формує запит до OpenAI API таким чином, щоб відповідь була повернута у структурованому форматі JSON. Це потрібно для того, щоб клієнтська частина могла автоматично відобразити окремі секції портфоліо: ім’я, професію, опис, навички, проєкти, освіту, досвід, мови, досягнення та контакти.

Окремо реалізовано функцію покращення портфоліо. Вона використовується після первинної генерації результату. Користувач натискає кнопку покращення, після чого поточне портфоліо повторно передається до AI-модуля. У результаті система покращує формулювання, робить опис більш професійним і структурованим, але зберігає загальний зміст та основні секції.

Перевагою такого підходу є те, що користувачу не потрібно самостійно формулювати всі розділи портфоліо. Достатньо ввести загальний опис, а система автоматично перетворює його на готову структуру.

Інтеграція AI-модуля базується на сучасних підходах до штучного інтелекту та обробки природної мови [18–20].

На рисунку 3.4 наведено фрагмент коду, який відповідає за звернення до AI-модуля генерації портфоліо.

```
const generatePortfolio = async () => {
  setLoading(true);

  try {
    const response = await fetch(
      `${API_URL}/generate?prompt=${encodeURIComponent(
        prompt
      )}&style=${encodeURIComponent(
        selectedStyle
      )}&language=${encodeURIComponent(language)}`
    );

    if (!response.ok) {
      throw new Error("Backend returned an error.");
    }

    const data = await response.json();

    setPortfolio(data);
    setSelectedPortfolioId(null);
  } catch (error) {
    console.error(error);
    alert(t.generateError);
  } finally {
    setLoading(false);
  }
};
```

Рисунок 3.4

3.7 Реалізація експорту портфоліо у PDF та HTML

Для зручного використання результатів у вебзастосунку реалізовано експорт створеного портфоліо у формати PDF та HTML. Це дозволяє користувачу не лише переглядати портфоліо у вебінтерфейсі, а й зберігати його як окремий файл.

Експорт у PDF призначений для створення документа, який можна використати як резюме або надіслати роботодавцю. PDF-файл містить основні секції портфоліо: ім’я, професію, опис, навички, проєкти, освіту, досвід, мови,

досягнення та контакти. На рисунку 3.5 наведено фрагмент коду експорту портфоліо у PDF.

```
const handleDownloadPdf = async () => {
  if (!portfolio) {
    return;
  }

  setPdfLoading(true);

  try {
    await downloadPdf(previewPdfRef, t);
  } catch (error) {
    console.error("PDF export error:", error);
    alert(t.pdfCreateError);
  } finally {
    setPdfLoading(false);
  }
};
```

Рисунок 3.5

Для формування PDF використовується окремий HTML-блок, який має світлий стиль оформлення та адаптований вигляд для друку.

Експорт у HTML дозволяє створити окремий файл сторінки портфоліо. Такий файл містить HTML-структуру та CSS-стилі всередині документа, тому його можна відкрити локально у браузері без запуску основного вебзастосунку. Це робить результат більш універсальним, оскільки користувач отримує готову вебсторінку.

Під час формування HTML-файлу система створює документ із базовою структурою: html, head, style та body. У тіло документа додаються дані поточного портфоліо. Після цього створюється об’єкт Blob, формується тимчасове посилання для завантаження та виконується збереження файлу на пристрій користувача.

На рисунку 3.6 наведено фрагмент коду експорту портфоліо у HTML.

```
const handleDownloadHtml = () => {
  if (!portfolio) {
    return;
  }

  setHtmlLoading(true);

  try {
    downloadHtml(portfolio, t, language);
  } catch (error) {
    console.error("HTML export error:", error);
    alert(t.htmlCreateError);
  } finally {
    setHtmlLoading(false);
  }
};
```

Рисунок 3.6

3.8 Тестування вебзастосунку

Після реалізації основних функцій було виконано тестування вебзастосунку. Метою тестування була перевірка коректності роботи авторизації, генерації портфоліо, покращення тексту, збереження результатів, експорту у PDF та HTML, а також зміни мови й теми інтерфейсу.

Тестування виконувалося вручну шляхом перевірки основних сценаріїв роботи користувача. Для кожного сценарію було визначено вхідні дії, очікуваний результат і фактичний результат (див. табл. 3.1).

Таблиця 3.1 – Результати тестування вебзастосунку

№	Сценарій тестування	Дії користувача	Очікуваний результат	Фактичний результат
1	Реєстрація користувача	Ввести email і коректний пароль	Обліковий запис створено	Виконано
2	Перевірка складності пароля	Ввести пароль без великої літери або цифри	Система показує повідомлення про помилку	Виконано

Продовження таблиці 3.1

№	Сценарій тестування	Дії користувача	Очікуваний результат	Фактичний результат
3	Вхід до системи	Ввести правильні дані авторизації	Користувач переходить до головної сторінки	Виконано
4	Генерація портфоліо	Ввести опис і натиснути кнопку генерації	Система формує структуроване портфоліо	Виконано
5	Покращення портфоліо	Натиснути кнопку покращення	Текст портфоліо стає професійнішим	Виконано
6	Збереження портфоліо	Натиснути кнопку збереження	Портфоліо з’являється у списку збережених	Виконано
7	Видалення портфоліо	Натиснути кнопку видалення	Запис видаляється зі списку	Виконано
8	Експорт у PDF	Натиснути кнопку завантаження PDF	Завантажується PDF-файл	Виконано
9	Експорт у HTML	Натиснути кнопку завантаження HTML	Завантажується HTML-файл	Виконано
10	Зміна мови	Обрати UA або EN	Інтерфейс змінює мову	Виконано
11	Зміна теми	Натиснути кнопку зміни теми	Інтерфейс змінює світлу/темну тему	Виконано
12	Адаптивність	Відкрити сайт на меншому екрані	Інтерфейс коректно відображається	Виконано

Результати тестування показали, що основні функції вебзастосунку працюють коректно.

Система дозволяє користувачу зареєструватися, увійти до облікового запису, згенерувати портфоліо, покращити його, зберегти результат, експортувати портфоліо у PDF та HTML, а також змінювати мову й тему інтерфейсу.

3.9 Керівництво користувача

У цьому підрозділі наведено коротке керівництво користувача для роботи з вебзастосунком генерації персонального портфоліо. Користувач може зареєструватися, увійти до системи, створити портфоліо за допомогою AI-модуля, покращити результат, зберегти портфоліо та експортувати його у PDF або HTML.

3.9.1 Реєстрація та вхід до системи

Для початку роботи користувач відкриває вебзастосунок у браузері. На стартовій сторінці доступна форма входу та реєстрації. Для створення облікового запису користувач вводить електронну пошту та пароль. Пароль має відповідати вимогам складності: містити від 10 до 50 символів, щонайменше одну велику літеру та одну цифру.

Після реєстрації користувач може увійти до системи, використовуючи вказану електронну пошту та пароль. У разі успішного входу відкривається головна сторінка вебзастосунку.

На рисунку 3.7 наведено форму входу та реєстрації користувача.

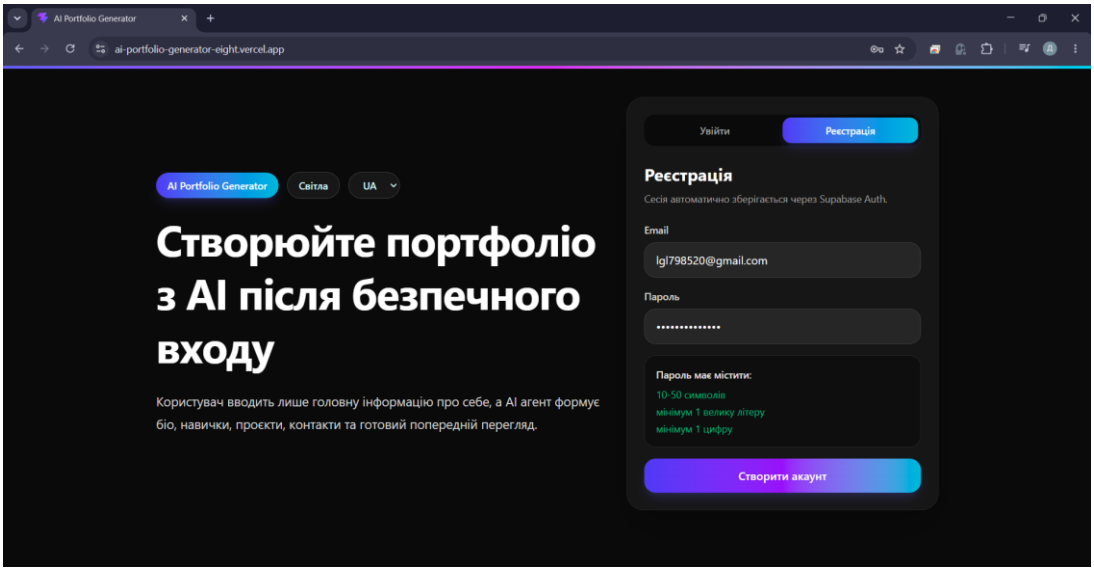


Рисунок 3.7

3.9.2 Генерація персонального портфоліо

Після входу до системи користувач переходить до головного інтерфейсу вебзастосунку. У лівій частині сторінки розміщено блок AI Assistant, у якому користувач вводить опис про себе. У тексті можна вказати ім’я, професію, навички, освіту, досвід, проєкти, мови, досягнення та контактні дані.

Перед запуском генерації користувач може обрати стиль портфоліо. Після цього необхідно натиснути кнопку генерації. Система передає введений опис до AI-модуля та формує структуроване портфоліо.

На рисунку 3.8 показано головну сторінку вебзастосунку з полем введення опису.

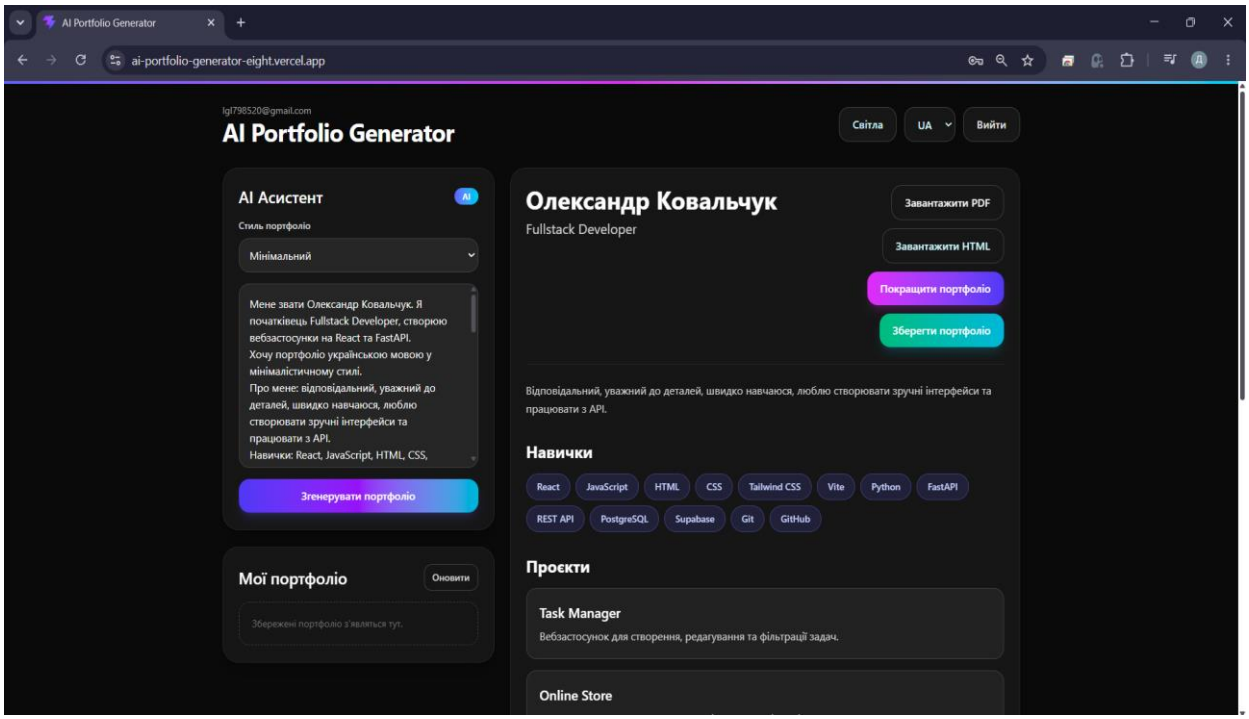


Рисунок 3.8

3.9.3 Покращення портфоліо

Після первинної генерації користувач може скористатися функцією покращення портфоліо. Для цього потрібно натиснути кнопку покращення.

Система повторно обробляє вже створений результат за допомогою AI-модуля.

У результаті покращення текст професійного опису та описів проєктів стає більш структурованим, точним і професійним. При цьому основна структура портфоліо зберігається.

На рисунку 3.9 показано використання функції покращення портфоліо.

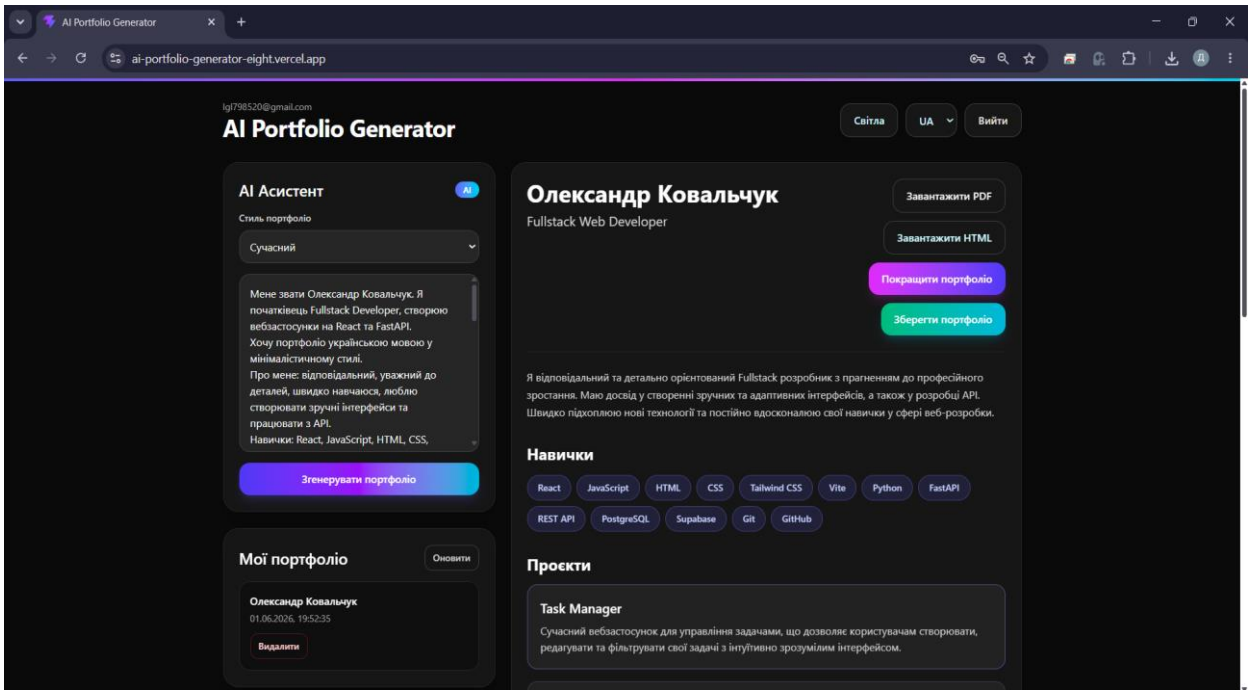


Рисунок 3.9

3.9.4 Збереження та видалення портфоліо

Після створення портфоліо користувач може зберегти його у базі даних. Для цього потрібно натиснути кнопку збереження.

Після успішного збереження портфоліо з’являється у списку збережених результатів.

Користувач може переглядати збережені портфоліо, відкривати їх повторно або видаляти непотрібні записи. Видалення виконується за допомогою відповідної кнопки біля збереженого портфоліо.

На рисунку 3.10 наведено список збережених портфоліо користувача.

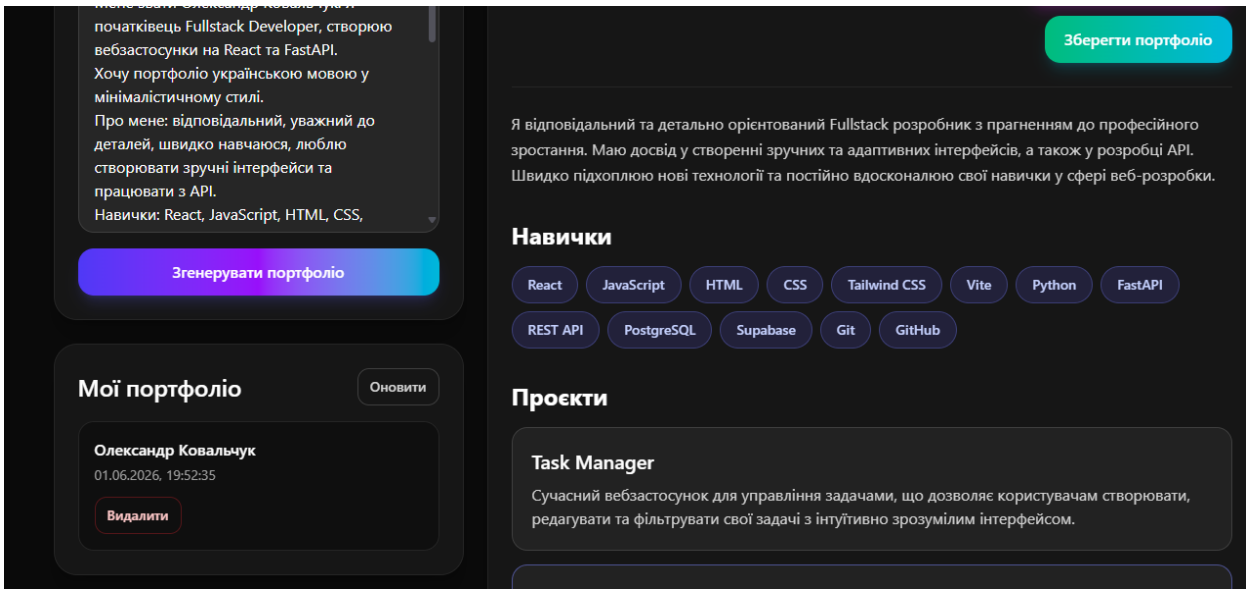


Рисунок 3.10

3.9.5 Експорт портфоліо у PDF та HTML

Для подальшого використання результату користувач може експортувати портфоліо у PDF або HTML. На рисунку 3.11 показано результат експорту портфоліо у PDF. На рисунку 3.12 показано результат експорту портфоліо у HTML.

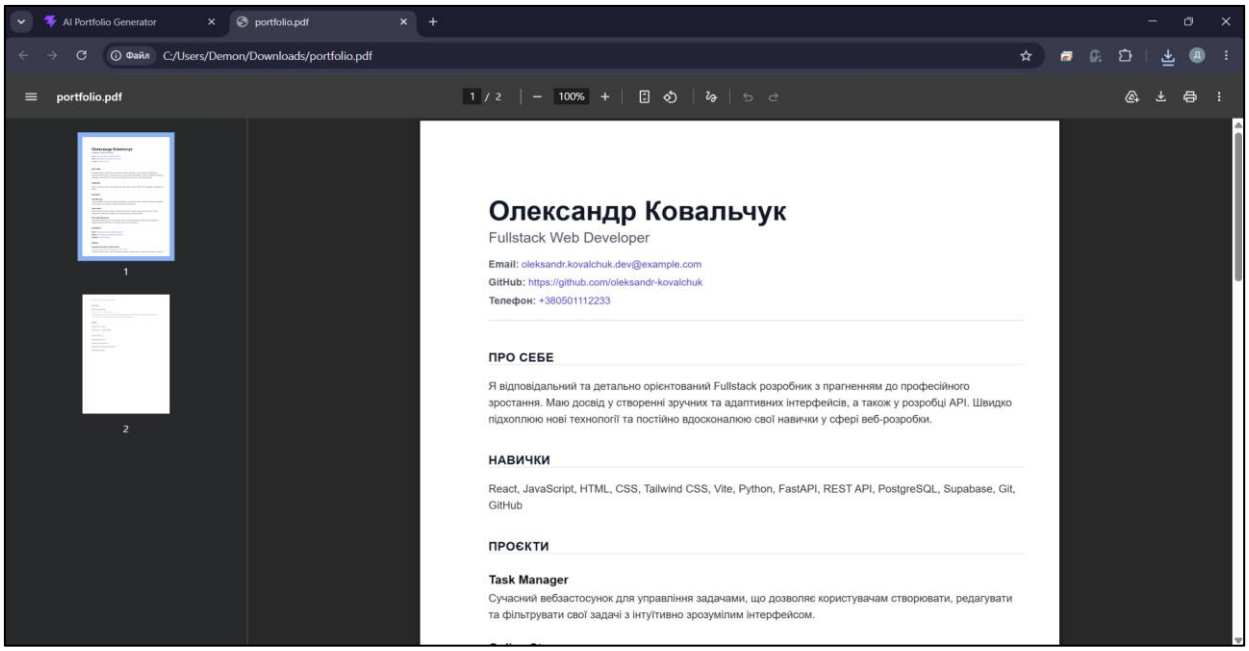


Рисунок 3.11

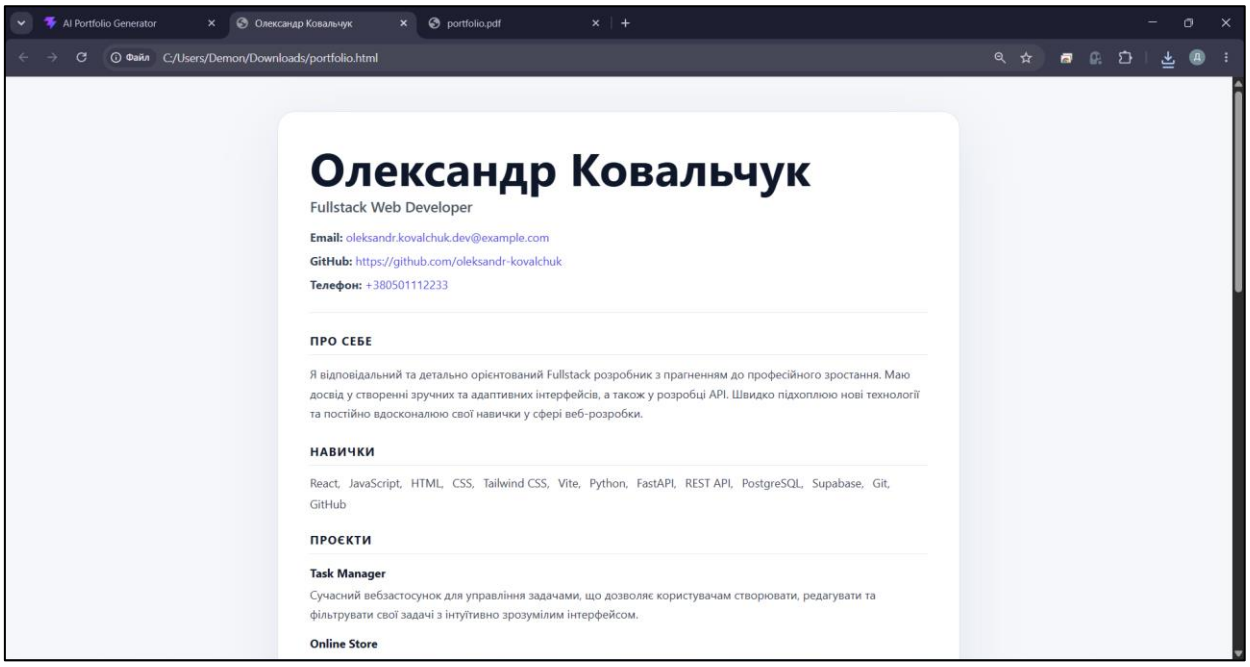


Рисунок 3.12

PDF-файл можна використовувати як резюме або документ для надсилання роботодавцю. HTML-файл відкривається у браузері як окрема вебсторінка.

Для експорту потрібно натиснути відповідну кнопку завантаження. Після цього файл автоматично зберігається на пристрій користувача.

3.9.6 Зміна мови та теми інтерфейсу

У вебзастосунку реалізовано підтримку української та англійської мов. Користувач може змінити мову інтерфейсу за допомогою відповідного перемикача. Обрана мова впливає на підписи елементів інтерфейсу та мову генерації портфоліо.

Також користувач може перемикати світлу та темну тему. Це дозволяє налаштувати зовнішній вигляд вебзастосунку відповідно до власних уподобань.

На рисунку 3.13 наведено приклад відображення вебзастосунку у світлій темі.

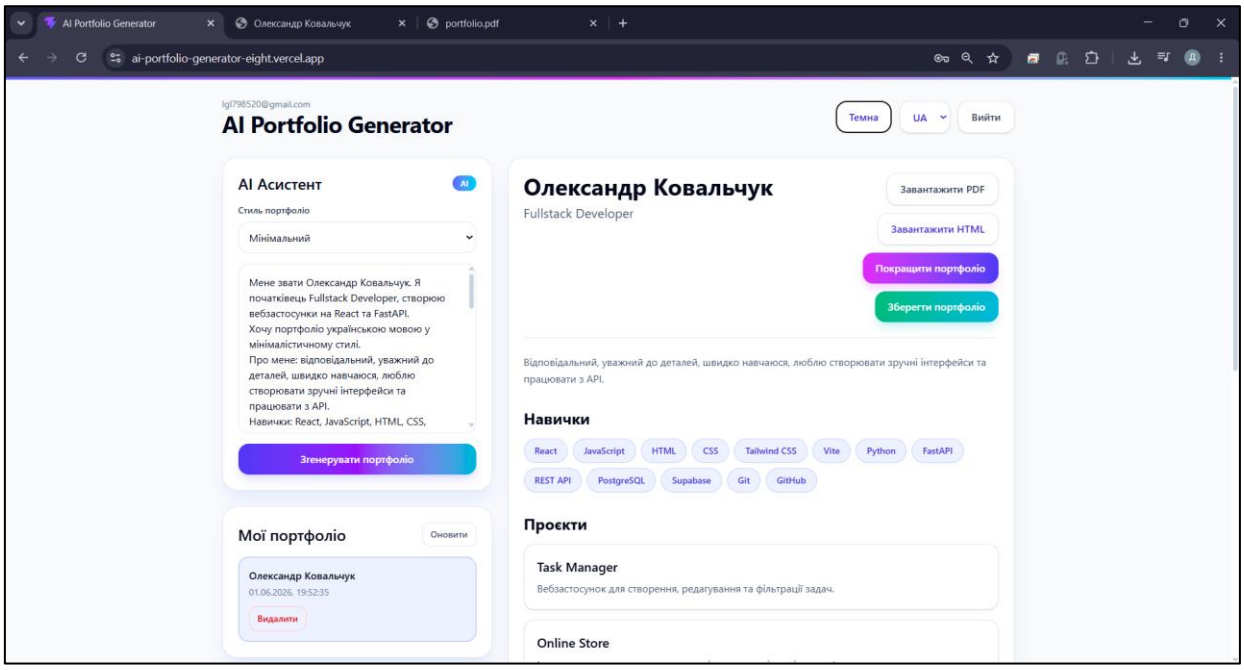


Рисунок 3.13

Таким чином, у результаті було повністю реалізовано та комплексно протестовано адаптивний вебзастосунок для автоматизованої генерації, покращення, збереження та кросплатформного експорту персональних портфоліо на основі інтеграції сучасних AI-технологій та хмарної інфраструктури.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено вебзастосунок для автоматизованої генерації персонального портфоліо з використанням технологій штучного інтелекту. Розроблена система дозволяє користувачу створювати структуроване портфоліо на основі звичайного текстового опису, зберігати результати та експортувати фінальну версію у форматах PDF і HTML.

У першому розділі було сформульовано технічне завдання, визначено функціональні та нефункціональні вимоги до системи. Було описано предметну область автоматизованого створення персонального портфоліо та визначено основні проблеми, які виникають під час ручного формування портфоліо. Також було виконано огляд і порівняння аналогів, що дозволило обґрунтувати доцільність розробки власного вебзастосунку.

У другому розділі було виконано проєктування системи. Було розроблено ER-модель бази даних, описано основні сутності системи та зв’язки між ними. Також було побудовано діаграму варіантів використання, діаграму діяльності основного бізнес-процесу, діаграму класів основних компонентів, діаграму послідовності генерації портфоліо та діаграму розгортання. Це дозволило визначити загальну структуру системи та логіку взаємодії її компонентів.

У третьому розділі було описано програмну реалізацію вебзастосунку. Клієнтську частину реалізовано з використанням React, Vite та Tailwind CSS. Серверну частину створено на основі FastAPI. Для авторизації користувачів і збереження даних використано Supabase Auth та PostgreSQL. Генерацію та покращення текстового наповнення портфоліо реалізовано за допомогою OpenAI API.

У вебзастосунку реалізовано реєстрацію та вхід користувача, перевірку складності пароля, введення опису користувача, вибір стилю портфоліо,

генерацію персонального портфоліо, покращення згенерованого тексту, збереження результатів у базі даних, перегляд і видалення збережених портфоліо. Також реалізовано підтримку української та англійської мов, світлої й темної теми інтерфейсу.

Для подальшого використання результатів було реалізовано експорт портфоліо у форматах PDF та HTML. PDF-файл може бути використаний як готовий документ для надсилання роботодавцю, а HTML-файл можна відкрити у браузері як окрему сторінку портфоліо. Це розширює практичне застосування системи та робить результат генерації зручним для користувача.

Було виконано тестування основних сценаріїв роботи вебзастосунку: реєстрації, входу до системи, генерації портфоліо, покращення тексту, збереження та видалення портфоліо, експорту у PDF та HTML, зміни мови й теми інтерфейсу. За результатами тестування підтверджено коректність роботи основних функцій системи.

Розроблений вебзастосунок було розгорнуто у хмарному середовищі. Клієнтську частину розміщено на платформі Vercel, серверну частину – на Render, а авторизація та база даних працюють на основі Supabase. Завдяки цьому користувач може працювати із системою через браузер як з персонального комп’ютера, так і з мобільного пристрою.

Отже, поставлену мету кваліфікаційної роботи було досягнуто. У результаті створено функціональний вебзастосунок, який забезпечує автоматизоване створення персонального портфоліо, використання AI-модуля для генерації та покращення тексту, збереження результатів, експорт у PDF і HTML та зручну взаємодію користувача із системою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sommerville I. Software Engineering : 10th ed. Boston : Pearson, 2016. 816 p.

2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach : 8th ed. New York : McGraw-Hill Education, 2015. 976 p.

3. Wiegers K., Beatty J. Software Requirements : 3rd ed. Redmond : Microsoft Press, 2013. 672 p.

4. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language : 3rd ed. Boston : Addison-Wesley, 2003. 208 p.

5. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development : 3rd ed. Upper Saddle River : Prentice Hall, 2004. 736 p.

6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

7. Bass L., Clements P., Kazman R. Software Architecture in Practice : 4th ed. Boston : Addison-Wesley, 2021. 464 p.

8. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

9. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Boston : Prentice Hall, 2008. 464 p.

10. Beck K. Test-Driven Development: By Example. Boston : Addison-Wesley, 2003. 240 p.

11. Myers G. J., Sandler C., Badgett T. The Art of Software Testing : 3rd ed. Hoboken : Wiley, 2011. 256 p.

12. Elmasri R., Navathe S. B. Fundamentals of Database Systems : 7th ed. Boston : Pearson, 2016. 1280 p.

13. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts : 7th ed. New York : McGraw-Hill Education, 2019. 1376 p.

14. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1993. 362 p.

15. Norman D. A. The Design of Everyday Things. Revised and expanded. New York : Basic Books, 2013. 368 p.

16. Russell S., Norvig P. Artificial Intelligence: A Modern Approach : 4th ed. Hoboken : Pearson, 2021. 1168 p.

17. Vaswani A., Shazeer N., Parmar N., Uszkoreit J. et al. Attention Is All You Need. *Advances in Neural Information Processing Systems*. 2017. Vol. 30. P. 5998–6008.

18. Документація бібліотеки React. *React*. URL: https://react.dev/learn (дата звернення: 18.03.2026).

19. Офіційна документація інструмента Vite. *Vite*. URL: https://vite.dev/guide/ (дата звернення: 20.03.2026).

20. Офіційна документація Tailwind CSS. *Tailwind CSS*. URL: https://tailwindcss.com/docs (дата звернення: 22.03.2026).

21. Офіційна документація FastAPI. *FastAPI*. URL: https://fastapi.tiangolo.com/ (дата звернення: 24.03.2026).

22. Офіційна документація платформи Supabase. *Supabase*. URL: https://supabase.com/docs (дата звернення: 27.03.2026).

23. Офіційна документація PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/docs/ (дата звернення: 29.03.2026).

24. Офіційна документація OpenAI API. *OpenAI API*. URL: https://platform.openai.com/docs (дата звернення: 29.03.2026).

25. Документація платформи Render. *Render*. URL: https://render.com/docs (дата звернення: 30.03.2026).

26. AI Portfolio Generator. Вебзастосунок для генерації персонального портфоліо. URL: https://ai-portfolio-generator-eight.vercel.app/ (дата звернення: 25.05.2026).

ДОДАТОК А

Фрагменти програмного коду frontend-частини

У додатку наведено фрагменти програмного коду клієнтської частини вебзастосунку, які демонструють підключення Supabase, збереження портфоліо, завантаження збережених результатів, експорт у PDF та HTML.

А.1 Фрагмент підключення Supabase у клієнтській частині

```
import { createClient } from "@supabase/supabase-js";

const supabaseUrl = "https://ukkjfetmtmixiwphnhbm.supabase.co";

const supabaseKey = "sb_publishable_gRZUh2Fws59AeRjKvCK0zA__aV9jXWK";

export const supabase = createClient(
  supabaseUrl,
  supabaseKey
);
```

А.2 Фрагмент збереження портфоліо у базу даних

```
const savePortfolio = async () => {
  if (!portfolio || !session?.user?.id) {
    return;
  }

  setSaveLoading(true);

  try {
    const title = getPortfolioTitle(portfolio, t);

    const { data, error } = await supabase
```

```
      .from("portfolios")
      .insert({
        user_id: session.user.id,
        title,
        data: portfolio,
      })
      .select("id,title,data,created_at")
      .single();

    if (error) {
      throw error;
    }

    setSavedPortfolios((current) => [data, ...current]);
    setSelectedPortfolioId(data.id);
  } catch (error) {
    console.error(error);
    alert(t.saveError);
  } finally {
    setSaveLoading(false);
  }
};
```

А.3 Фрагмент завантаження збережених портфоліо

```
const loadSavedPortfolios = useCallback(async (userId, options = {}) => {
  if (!userId) {
    return;
  }

  const force = options.force === true;

  if (!force && loadedPortfoliosUserIdRef.current === userId) {
    return;
  }

  loadedPortfoliosUserIdRef.current = userId;
  setListLoading(true);

  try {
```

```
    const { data, error } = await supabase
      .from("portfolios")
      .select("id,title,data,created_at")
      .eq("user_id", userId)
      .order("created_at", { ascending: false });

    if (error) {
      throw error;
    }

    setSavedPortfolios(data || []);
  } catch (error) {
    console.error(error);

    if (loadedPortfoliosUserIdRef.current === userId) {
      loadedPortfoliosUserIdRef.current = null;
    }

    alert(tRef.current.loadError);
  } finally {
    setListLoading(false);
  }
}, []);
```

А.4 Фрагмент експорту портфоліо у PDF

```
import html2pdf from "html2pdf.js";

const downloadPdf = async (pdfRef, t) => {
  const element = pdfRef.current;

  if (!element) {
    alert(t.pdfBlockMissing);
    return;
  }

  const options = {
    margin: 10,
    filename: "portfolio.pdf",
    image: { type: "jpeg", quality: 0.98 },
```

```
    html2canvas: {
      scale: 2,
      useCORS: true,
      backgroundColor: "#ffffff",
    },
    jsPDF: {
      unit: "mm",
      format: "a4",
      orientation: "portrait",
    },
  };

  await html2pdf().set(options).from(element).save();
};
```

А.5 Фрагмент експорту портфоліо у HTML

```
const downloadHtml = (portfolio, t, language) => {
  const html = buildPortfolioHtml(portfolio, t, language);

  const blob = new Blob([html], {
    type: "text/html;charset=utf-8",
  });

  const url = URL.createObjectURL(blob);
  const link = document.createElement("a");

  link.href = url;
  link.download = "portfolio.html";

  document.body.appendChild(link);
  link.click();
  link.remove();

  URL.revokeObjectURL(url);
};
```

ДОДАТОК Б

Фрагменти програмного коду backend-частини

У додатку наведено фрагменти програмного коду серверної частини вебзастосунку, які демонструють реалізацію API для генерації та покращення персонального портфоліо за допомогою AI-модуля.

Б1. Фрагмент підключення FastAPI та налаштування CORS

```
from dotenv import load_dotenv
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

load_dotenv()

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Б.2 Фрагмент endpoint генерації портфоліо

```
@app.get("/generate")
def generate_portfolio(prompt: str, style: str = "Minimal", language: str = "uk"):
    output_language = LANGUAGE_NAMES.get(language, "Ukrainian")

    response = client.chat.completions.create(
```

```
        model="gpt-4o-mini",
        messages=[
            {
                "role": "system",
                "content": f"""
Ти AI агент для генерації професійного портфоліо.

Завжди повертай ТІЛЬКИ валідний JSON.

Обраний стиль портфоліо: {style}.
Мова портфоліо: {output_language}.

Заповнюй усі поля портфоліо:
name, profession, bio, skills, projects, contacts,
education, experience, languages, achievements,
softSkills, certificates, links і theme.

Структура:

{PORTFOLIO_JSON_STRUCTURE}
                """
            },
            {
                "role": "user",
                "content": prompt
            }
        ]
    )

    content = response.choices[0].message.content

    return parse_json_response(content)
```

Б.3 Фрагмент endpoint покращення портфоліо

```
@app.post("/improve")
def improve_portfolio(request: ImproveRequest):
    output_language = LANGUAGE_NAMES.get(request.language, "Ukrainian")

    response = client.chat.completions.create(
```

```
        model="gpt-4o-mini",
        messages=[
            {
                "role": "system",
                "content": f"""
Ти AI редактор професійних портфоліо.

Завжди повертай ТІЛЬКИ валідний JSON.

Обраний стиль портфоліо: {request.style}.
Мова портфоліо: {output_language}.

Покращи існуюче портфоліо:
- зроби bio більш професійним;
- покращи profession title;
- покращи descriptions у projects;
- не видаляй contacts;
- не втрачай skills;
- не втрачай education, experience, languages,
  achievements, softSkills, certificates і links;
- не змінюй структуру JSON.

Структура має залишитися:

{PORTFOLIO_JSON_STRUCTURE}
                """
            },
            {
                "role": "user",
                "content": json.dumps(
                    request.portfolio,
                    ensure_ascii=False
                )
            }
        ]
    )

    content = response.choices[0].message.content

    return parse_json_response(content)
```

Б.4 Фрагмент формування запиту до OpenAI API

```
response = client.chat.completions.create(
   model="gpt-4o-mini",
   messages=[
      {
         "role": "system",
         "content": f"""
Ти AI агент для генерації професійного портфоліо.

Завжди повертай ТІЛЬКИ валідний JSON.

Обраний стиль портфоліо: {style}.
Мова портфоліо: {output_language}.

Заповнюй усі поля портфоліо:
name, profession, bio, skills, projects, contacts,
education, experience, languages, achievements,
softSkills, certificates, links і theme.

Структура:

{PORTFOLIO_JSON_STRUCTURE}
         """
      },
      {
         "role": "user",
         "content": prompt
      }
   ]
)
```

ДОДАТОК В

Фрагменти програмного коду експорту та роботи з базою даних

У додатку наведено фрагменти програмного коду, які демонструють збереження портфоліо у базі даних, завантаження збережених портфоліо, видалення записів, а також експорт фінальної версії портфоліо у форматах PDF та HTML.

В.1 Фрагмент збереження портфоліо у базу даних

```
const savePortfolio = async () => {
  if (!portfolio || !session?.user?.id) {
    return;
  }

  setSaveLoading(true);

  try {
    const title = getPortfolioTitle(portfolio, t);

    const { data, error } = await supabase
      .from("portfolios")
      .insert({
        user_id: session.user.id,
        title,
        data: portfolio,
      })
      .select("id,title,data,created_at")
      .single();

    if (error) {
      throw error;
    }

    setSavedPortfolios((current) => [data, ...current]);
    setSelectedPortfolioId(data.id);
```

```
  } catch (error) {
    console.error(error);
    alert(t.saveError);
  } finally {
    setSaveLoading(false);
  }
};
```

В.2 Фрагмент завантаження збережених портфоліо

```
const loadSavedPortfolios = useCallback(async (userId, options = {}) => {
  if (!userId) {
    return;
  }

  const force = options.force === true;

  if (!force && loadedPortfoliosUserIdRef.current === userId) {
    return;
  }

  loadedPortfoliosUserIdRef.current = userId;
  setListLoading(true);

  try {
    const { data, error } = await supabase
      .from("portfolios")
      .select("id,title,data,created_at")
      .eq("user_id", userId)
      .order("created_at", { ascending: false });

    if (error) {
      throw error;
    }

    setSavedPortfolios(data || []);
  } catch (error) {
    console.error(error);

    if (loadedPortfoliosUserIdRef.current === userId) {
```

```
    loadedPortfoliosUserIdRef.current = null;
   }

   alert(tRef.current.loadError);
  } finally {
   setListLoading(false);
  }
}, []);
```

В.3 Фрагмент видалення портфоліо

```
const deletePortfolio = async (portfolioId) => {
  if (!session?.user?.id) {
   return;
  }

  setDeleteLoadingId(portfolioId);

  try {
   const { error } = await supabase
    .from("portfolios")
    .delete()
    .eq("id", portfolioId)
    .eq("user_id", session.user.id);

   if (error) {
    throw error;
   }

   setSavedPortfolios((current) =>
    current.filter((savedPortfolio) => savedPortfolio.id !== portfolioId)
   );

   if (selectedPortfolioId === portfolioId) {
    setPortfolio(null);
    setSelectedPortfolioId(null);
   }
  } catch (error) {
   console.error(error);
   alert(t.deleteError);
```

```
  } finally {
    setDeleteLoadingId(null);
  }
};
```

В.4 Фрагмент експорту портфоліо у PDF

```
const deletePortfolio = async (portfolioId) => {
  if (!session?.user?.id) {
    return;
  }

  setDeleteLoadingId(portfolioId);

  try {
    const { error } = await supabase
      .from("portfolios")
      .delete()
      .eq("id", portfolioId)
      .eq("user_id", session.user.id);

    if (error) {
      throw error;
    }

    setSavedPortfolios((current) =>
      current.filter((savedPortfolio) => savedPortfolio.id !== portfolioId)
    );

    if (selectedPortfolioId === portfolioId) {
      setPortfolio(null);
      setSelectedPortfolioId(null);
    }
  } catch (error) {
    console.error(error);
    alert(t.deleteError);
  } finally {
    setDeleteLoadingId(null);
  }
};
```

В.5 Фрагмент експорту портфоліо у HTML

```
const downloadHtml = (portfolio, t, language) => {
  const html = buildPortfolioHtml(portfolio, t, language);

  const blob = new Blob([html], {
    type: "text/html;charset=utf-8",
  });

  const url = URL.createObjectURL(blob);
  const link = document.createElement("a");

  link.href = url;
  link.download = "portfolio.html";

  document.body.appendChild(link);
  link.click();
  link.remove();

  URL.revokeObjectURL(url);
};
```