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

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для організації та планування маршрутів та поїздок

Виконав студент IV курсу, групи ІТ-04
(шифр групи)
Качук Владислав Дмитрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст.викл. Марченко Олена Іванівна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

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

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Качуку Владиславу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для організації та планування маршрутів та поїздок

керівник проєкту ст.викл. Марченко Олена Іванівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» травня 2024 р. №2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Передпроєкте обстеження предметної області: аналіз предметної області, аналіз існуючих рішень, опис бізнес-процесів.
2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, розроблення функціональних вимог, розроблення нефункціональних вимог.
3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, обґрунтування вибору засобів розробки, Конструювання програмного забезпечення.
4) Аналіз якості тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.
5) Розгортання та супровід програмоного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема бази даних _____

2) Схема структурна класів програмного забезпечення _____

3) Креслення вигляду екраних форм _____

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «11» березня 2024 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	18.03.2024	
3	Постановка та формалізація задачі	20.03.2024	
4	Розробка інформаційного забезпечення	27.03.2024	
5	Алгоритмізація задачі	02.04.2024	
6	Обґрунтування вибору використаних технічних засобів	10.04.2024	
7	Розробка програмного забезпечення	18.04.2024	
8	Налагодження програми	12.05.2024	
9	Виконання графічних документів	20.05.2024	
10	Оформлення пояснювальної записки	25.05.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Владислав КАЧУК

(підпис) (ініціали, прізвище)

Керівник _____ Олена МАРЧЕНКО

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 29 таблиць, 20 рисунків та 14 джерел – загалом 64 сторінки.

Дипломний проєкт присвячений розробці та впровадженню програмного забезпечення для планування подорожей – TripPlanner.

Мета проєкту полягає у створенні зручного інструменту для користувачів, який дозволяє планувати подорожі, зберігати маршрути та отримувати рекомендації щодо оптимальних маршрутів і зупинок.

Об'єкт дослідження: процес планування подорожей з використанням веб-додатків.

Предмет дослідження: методи та засоби розробки програмного забезпечення для планування подорожей, включаючи архітектурні рішення, бази даних та інтеграцію з API.

У розділі 1 розглянуто аналіз існуючих рішень у сфері планування подорожей та обрані інструменти для реалізації проєкту.

Розділ 2 присвячений аналізу бізнес-процесів, постановці задачі та опису вимог до програмного забезпечення.

У розділі 3 детально описано процес конструювання та розробки програмного забезпечення, включаючи архітектурні патерни та компоненти системи.

Розділ 4 охоплює тестування програмного забезпечення, аналіз якості та опис контрольного прикладу.

Розділ 5 описує процес розгортання та супроводу програмного забезпечення TripPlanner, включаючи налаштування на платформі Azure та автоматизацію процесу публікації нових версій.

Програмне забезпечення впроваджено на хмарній платформі Azure для забезпечення надійності та масштабованості.

КЛЮЧОВІ СЛОВА: IDE, API, UI, HTTP, JSON, ОС, LINQ, БД, DI, SQL.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 29 tables, 20 figures and 14 sources – in total 64 pages.

The purpose of the diploma project is devoted to the development and implementation of travel planning software - TripPlanner.

The goal of the project is to create a convenient tool for users that allows them to plan trips, save routes and receive recommendations on optimal routes and stops.

The object of the study: the process of travel planning using web applications.

Research topic: Travel planning software development methods and tools, including architectural solutions, databases, and API integration.

Chapter 1 deals with the analysis of existing solutions in the field of travel planning and selected tools for project implementation.

Chapter 2 is devoted to the analysis of business processes, problem statement and description of software requirements.

Chapter 3 details the software design and development process, including architectural patterns and system components.

Chapter 4 covers software testing, quality analysis, and benchmarking.

Chapter 5 describes the process of deploying and maintaining the TripPlanner software, including configuring it on the Azure platform and automating the process of publishing new versions.

The software is implemented on the Azure cloud platform for reliability and scalability.

KEYWORDS: IDE, API, UI, HTTP, JSON, OC, LINQ, DB, DI, SQL.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9406.045440.01.91	Технічне завдання	12	
3	А4	КПІ.ІТ-9406.045440.02.81	Пояснювальна записка	64	
4	А4	КПІ.ІТ-9406.045440.03.12	Текст програми	35	
5	А4	КПІ.ІТ-9406.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9406.045440.05.34	Керівництво користувача	9	
7	А3	КПІ.ІТ-9406.045440	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ. ІТ-9406.045440.00.90		
Розроб.		Качук В.Д.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Марченко О.І.					
Н.контр.		Родіонов П.Ю.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-04	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

Програмне забезпечення для організації та планування маршрутів та поїздок

Технічне завдання

КПІ.ІТ-9406.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена МАРЧЕНКО

Нормоконтроль:

_____ Павло РОДІОНОВ

Виконавець:

_____ Владислав КАЧУК

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для організації та планування маршрутів та поїздок.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення веб-застосунку для організації та планування маршрутів та поїздок [045440].

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку для організації та планування маршрутів та поїздок є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для створення веб-застосунку TripPlanner, який дозволяє користувачам планувати подорожі, зберігати маршрути, отримувати пропозиції щодо міст, які можна відвідати під час поїздки, та зручно взаємодіяти з усією необхідною інформацією для подорожей. Цей додаток орієнтований на користувачів, які активно подорожують та хочуть мати інструмент для ефективного планування та управління своїми поїздками.

Метою розробки є забезпечення користувачів інтуїтивно зрозумілим інтерфейсом для планування подорожей, зберігання важливої інформації про поїздки, та покращення загального досвіду подорожей шляхом надання актуальних і зручних функцій. Додаток також спрямований на інтеграцію з різними зовнішніми сервісами для надання розширених можливостей, таких як автоматичне пропонування цікавих місць та розрахунок маршрутів з урахуванням особистих уподобань користувача.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

- Забезпечення реєстрації та авторизації користувачів.
- Введення та збереження інформації про подорожі.
- Відображення маршрутів та пропозицій по містах.
- Інтуїтивно зрозумілий інтерфейс для управління подорожами.
- Пошук та фільтрація збережених подорожей.

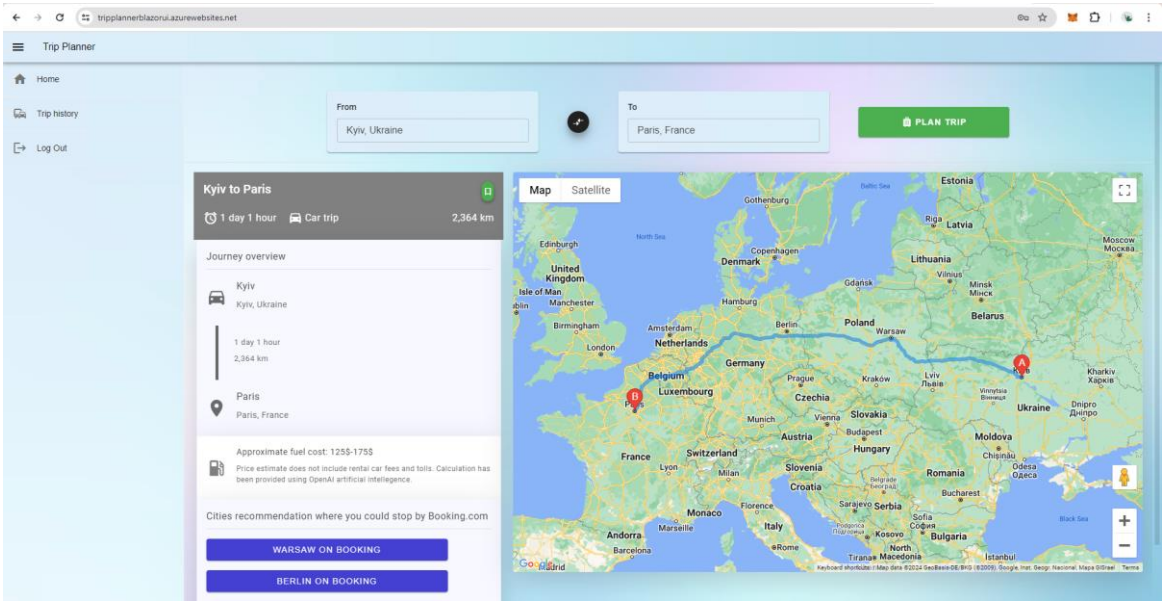


Рисунок 4.1 – Прототип екранної форми головного меню

4.1.2 Для користувача:

- Реєстрація та авторизація користувачів.
- Створення, редагування та видалення подорожей.
- Отримання пропозицій щодо міст для відвідування.
- Збереження та перегляд історії подорожей.

– Зручна навігація по додатку.

4.1.3 Додаткові вимоги:

Підтримка інтеграції з зовнішніми сервісами для отримання додаткових даних.

Забезпечення високої продуктивності при великих обсягах даних.

Забезпечення безперебійної роботи системи.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Передбачити відновлення після збоїв, включаючи час відновлення системи, наявність контрольних точок та резервних копій даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

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

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i7;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в форматі JSON, містити інформацію про користувачів, подорожі, міста та маршрути.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в форматі JSON, включаючи інформацію про успішні дії користувачів, деталі подорожей та запропоновані маршрути.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C#.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі .NET Core з використанням середовища розробки Visual Studio.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторію на GitHub з відповідною документацією.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

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

– пояснювальна записка;

– технічне завдання;

– текст програми;

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

– схема бази даних;

– схема структурна класів програмного забезпечення;

– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

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

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	11.03	
2.	Розробка технічного завдання	20.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	22.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	02.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	18.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	12.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	25.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	25.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	25.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

Пояснювальна записка
до дипломного проєкту

на тему: «**Програмне забезпечення для організації та планування маршрутів та поїздок**»

КПІ.ІТ-9406.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
UI	–	User Interface – користувацький інтерфейс.
HTTP	–	Hypertext Transfer Protocol – протокол передачі гіпертексту.
JSON	–	JavaScript Object Notation – нотація об'єктів JavaScript.
ОС	–	Операційна система.
LINQ	–	Language Integrated Query – інтегрований мова запитів.
БД	–	База даних
DI	–	Dependency Injection – впровадження залежностей.
SQL	–	Structured Query Language – мова структурованих запитів.

ВСТУП

У сучасному світі автомобільні подорожі стають все більш популярними завдяки своїй гнучкості, комфорту та можливості дослідження нових місць у зручний для подорожуючих час. З розвитком цифрових технологій виникає потреба в інструментах, які допоможуть ефективно планувати маршрути, враховувати різноманітні фактори та забезпечувати безпеку подорожей. Розробка веб-додатку для планування автомобільних подорожей є актуальною, оскільки вона дозволяє спростити процес планування, зменшити витрати часу та ресурсів, а також підвищити комфорт і безпеку подорожуючих.

На сьогоднішній день існує безліч веб-додатків та мобільних застосунків для планування автомобільних подорожей, які пропонують різноманітні функції, від простих карт до комплексних інструментів для планування маршрутів. Провідні компанії, такі як Google, Apple, та Microsoft, активно розробляють і вдосконалюють свої навігаційні сервіси. Разом з тим, існують численні стартапи та менші компанії, які спеціалізуються на конкретних аспектах планування подорожей, таких як пошук кращих місць для зупинок, оптимізація витрат на пальне, та забезпечення безпеки на дорозі.

Веб-додаток для планування автомобільних подорожей може бути корисним як для індивідуальних мандрівників, так і для професійних водіїв та транспортних компаній. Для туристів це означає можливість легко планувати свої маршрути, знаходити цікаві місця та оптимізувати час на подорож. Для транспортних компаній та логістичних операторів - це інструмент для оптимізації маршрутів доставки, зменшення витрат на пальне та покращення ефективності операцій.

В цілому, розробка веб-додатку для планування автомобільних подорожей є перспективним напрямком, який відповідає сучасним тенденціям розвитку цифрових технологій та потребам користувачів.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Планування автомобільних подорожей включає в себе комплекс заходів, спрямованих на організацію маршруту, врахування дорожніх умов, часу в дорозі, зупинок, вартості пального та інших ресурсів. Основною метою такого планування є підвищення ефективності та комфорту подорожі, а також забезпечення безпеки.

Згідно з загальними визначеннями, веб-додаток для планування автомобільних подорожей є програмним засобом, який дозволяє користувачам створювати та оптимізувати маршрути для подорожей за допомогою веб-інтерфейсу. Ці додатки зазвичай використовують геоінформаційні системи (ГІС), які забезпечують інтерактивні карти та навігаційні дані в реальному часі.

Напрямки розробок в даній області включають:

- Геолокаційні сервіси.

  Розробка інтерактивних карт та навігаційних систем (наприклад, Google Maps, OpenStreetMap).
- Оптимізація маршрутів.

  Створення алгоритмів, що дозволяють вибирати найбільш ефективні маршрути з урахуванням дорожніх умов, трафіку та інших факторів (наприклад, Waze, MapQuest).
- Аналіз даних та прогнозування.

  Використання великих даних та алгоритмів машинного навчання для передбачення дорожніх умов та заторів (наприклад, INRIX, TomTom Traffic).

На сьогоднішній день знання предметної області активно використовуються в різноманітних веб-додатках та мобільних застосунках. Основні компоненти таких систем включають:

- Інтеграцію картографічних даних, що дозволяє користувачам отримувати інформацію про місцезнаходження та маршрути.
- Застосування методів оптимізації для знаходження найкоротших або найшвидших маршрутів, враховуючи різноманітні обмеження та параметри.

Попри значний прогрес, існують певні недоліки в поточному стані розробок у сфері планування автомобільних подорожей:

- Точність даних.

Дані про дорожні умови та трафік є точними та актуальними, що може призводити до неправильних рішень.

- Інтеграція даних.

  Складнощі в інтеграції даних з різних джерел, що можуть мати різні формати та структури.

Для покращення ситуації з розробками у сфері планування автомобільних подорожей можуть бути використані наступні підходи:

- Використання технологій штучного інтелекту для підвищення точності прогнозування цін на пальне, розумний вибір зупинок та дорожніх умов.
- Покращення інтеграції даних за допомогою стандартних форматів та протоколів обміну даними.

В рамках даного дипломного проєкту було обрано шлях розробки веб-додатку, який буде використовувати сучасні технології для інтеграції геоінформаційних даних, алгоритмів оптимізації маршрутів, штучний інтелект від OpenAI для отримання більш гнучких даних при плануванні подорожі. Особлива увага буде приділятись забезпеченню конфіденційності даних користувачів.

Для досягнення цієї мети планується використовувати:

- Google MAP API - API для провідних геолокаційних сервісів.
- OpenAI ChatGTP Turbo 3.5 – API для підключення AI асистента до серверної частини застосунку.
- Використання шифрування алгоритма SHA256 при забезпечені стандартів безпеки та конфіденційності для захисту даних користувачів.

Цей підхід дозволить створити ефективний та надійний веб-додаток, який буде корисним для широкого кола користувачів.

1.2 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-додатку для планування автомобільних подорожей TripPlanner. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

Однією з ключових задач у плануванні автомобільних подорожей є оптимізація маршрутів. Існує декілька основних алгоритмів, які широко використовуються для цієї мети:

1. Алгоритм Дейкстри:
    - Використовується для знаходження найкоротших шляхів у графах з невід'ємними вагами ребер.
    - Підходить для задачі знаходження оптимальних маршрутів у міських умовах, де кожен відрізок дороги може бути представлений як ребро графу з певною вагою (відстанню або часом).
    - Використовується в таких сервісах, як Google Maps

2. Методи машинного навчання:
    - Використовуються для вирішення складних задач оптимізації, де класичні алгоритми не можуть дати задовільного результату.
    - Можуть адаптуватися до зміни умов та навчатися на основі певних даних.
    - Використовується в таких сервісах, як OpenAI API

Для розробки веб-додатку «АвтоПланер» будуть використані наступні готові програмні рішення та технології:

1. .NET 8 Web API:
    - Потужний фреймворк для створення веб-апі, що забезпечує високу продуктивність та масштабованість.
    - Дозволяє створювати RESTful сервіси для обробки запитів, взаємодії з базами даних та інтеграції з іншими сервісами.
2. Blazor Server Side UI:
    - Фреймворк для створення інтерфейсів користувача за допомогою C# та Razor.
    - Підтримує розробку інтерактивних та динамічних веб-додатків, що виконуються на сервері.
3. Google Maps API:
    - Надає інструменти для роботи з інтерактивними картами, планування маршрутів, отримання даних про трафік та інші геолокаційні сервіси.
    - Забезпечує точні та актуальні дані, що допоможуть користувачам планувати свої подорожі з урахуванням дорожніх умов та місць інтересу.
4. Swagger:
    - Інструмент для документування та тестування API.
    - Дозволяє автоматично генерувати документацію для Web API, що спрощує розробку та інтеграцію клієнтських додатків.

5. SQL Server:
   - Реляційна база даних для зберігання інформації про користувачів, заплановані поїздки та інші необхідні дані.
   - Підтримує функції авторизації та реєстрації користувачів, а також зберігання і управління даними про маршрути.
6. OpenAI ChatGPT 3.5 Turbo:
   - Використовується для навчання та отримання динамічних даних.
   - Може надавати користувачам рекомендації щодо планування подорожей, інформацію про цікаві місця та відповіді на інші питання, пов'язані з поїздками.

Ці технології та інструменти забезпечать створення надійного, функціонального та зручного у використанні веб-додатку для планування автомобільних подорожей. Вони дозволяють інтегрувати потужні алгоритми оптимізації маршрутів, забезпечити взаємодію з користувачами через інтуїтивний інтерфейс та підтримувати актуальні дані про дорожні умови і місця інтересу.

1.2.1 Аналіз відомих програмних продуктів

У сфері планування автомобільних подорожей існує кілька готових програмних продуктів, які частково чи повністю реалізують функціонал, описаний у технічному завданні для веб-додатку «TripPlanner». Одним з найбільш відомих і популярних рішень є сервіс Rome2Rio.

Rome2Rio – це комплексний сервіс, який надає інформацію про маршрути між двома пунктами, включаючи різні види транспорту, такі як автомобільні подорожі, авіарейси, поїзди, автобуси, пороми та оренда автомобілів. Сервіс дозволяє користувачам планувати подорожі, надаючи інформацію про відстань, час у дорозі, вартість та можливі пересадки.

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проєкт TripPlanner	Веб-додаток Rome2Rio	Пояснення
Типи транспорту	Автомобіль, авіа, поїзд, автобус, паром, оренда авто	Автомобіль	Rome2Rio підтримує різні види транспорту, тоді як TripPlanner зосереджений на автомобільних подорожах.
Оптимізація маршрутів	Так	Так	Обидва сервіси пропонують функції оптимізації маршрутів для покращення подорожей.
Інформація про вартість подорожі	Так, обмежена	Так, з використанням адаптивної моделі штучного інтелекту	Обидва сервіси надають інформацію про вартість подорожі, що допомагає користувачам планувати бюджет.
Авторизація та реєстрація користувачів	Ні	Так (SQL Server)	TripPlanner підтримує авторизацію та реєстрацію користувачів для

			персоналізації досвіду та зберігання даних.
Рекомендації на основі штучного інтелекту	Ні	Так (OpenAI ChatGPT 3.5 Turbo)	TripPlanner використовує OpenAI ChatGPT для надання рекомендацій та динамічної інформації, що покращує користувацький досвід.
Інтеграція з іншими сервісами	Так (агрегатори подорожей, бронювання квитків, оренда авто)	Так (Google Maps API, OpenAI ChatGPT)	Обидва сервіси інтегруються з іншими сервісами, але АвтоПланер використовує сучасні технології, такі як OpenAI ChatGPT.

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Для реалізації веб-додатку «TripPlanner» розглянемо відомі технічні рішення, що включають архітектури програмного забезпечення, архітектурні паттерни, платформи та інструменти. Проведемо порівняльний аналіз цих рішень та визначимо ті, що будуть використані у розробці.

Багатошарова архітектура є одним з найбільш поширених підходів у розробці складних додатків [1]. Вона включає кілька шарів, кожен з яких відповідає за окремий аспект функціональності додатку:

1. Шар презентації:
   - Відповідає за взаємодію з користувачем.

   - Використовує Blazor Server Side для створення компонентів інтерфейсу користувача та інтеграцію з JavaScript для роботи з Google Maps API.
2. Шар бізнес-логіки:
   - Містить основні правила та логіку роботи додатку.
   - Реалізується з використанням принципів ООП та SOLID, що забезпечують модульність та підтримуваність коду.
3. Шар доступу до даних:
   - Відповідає за взаємодію з базою даних.
   - Використовує CodeFirst підхід з Entity Framework для роботи з SQL Server.
4. Шар сервісів:
   - Включає сервіси та API для взаємодії з зовнішніми системами.
   - Використовує Dependency Injection для підвищення гнучкості та зручності тестування.

Архітектурні паттерни:

1. Dependency Injection (DI):
   - Паттерн, що дозволяє інжектувати залежності в класи, замість створення їх всередині класу.
   - Покращує тестованість та підтримуваність коду.
2. OpenAPI:
   - Специфікація для створення документації та тестування RESTful API.
   - Використання Swagger для автоматичної генерації документації.
3. CodeFirst підхід:
   - Методологія, при якій спочатку створюються класи моделі, а потім на їх основі генерується структура бази даних.
   - Забезпечує гнучкість при внесенні змін до структури даних.
4. Test-Driven Development (TDD):

- Підхід, при якому спочатку пишуться тести, а потім код, що їх проходить.
- Забезпечує високу якість та надійність коду.

5. SOLID принципи:
   - Набір принципів для створення гнучких та підтримуваних систем.
   - Включають єдину відповідальність, відкритість/закритість, підстановку Лісков, розділення інтерфейсів та інверсію залежностей.

Таблиця 1.2 – Порівняльний аналіз технічних рішень

Технічне рішення	Переваги	Недоліки
Багатошарова архітектура	Чітке розділення відповідальностей, легкість у підтримці	Висока складність реалізації
Dependency Injection	Підвищена гнучкість, покращена тестованість	Підвищена складність налаштування
OpenAPI	Автоматична генерація документації, полегшення інтеграції	Додаткові витрати часу на налаштування
CodeFirst підхід	Гнучкість при зміні структури даних	Потребує знання ORM
Test-Driven Development	Висока якість та надійність коду	Збільшення часу розробки
SOLID принципи	Модульність, легкість у підтримці та розширенні	Вимагає додаткових зусиль на дотримання принципів, збільшення кодової бази

1.3 Опис бізнес-процесів

Веб-додаток «TripPlanner» призначений для планування автомобільних подорожей та надання користувачам зручного інтерфейсу для реєстрації, авторизації, планування маршрутів, збереження подорожей та управління ними. У цьому підрозділі описано основні бізнес-процеси додатку та їх наочне представлення за допомогою моделювання BPMN.

Опис послідовності створення облікового запису користувача:

– користувач переходить на сторінку реєстрації;

– користувач заповнює поля реєстрації;

– якщо введені поля, не відповідають шаблону заповнення на клієнтській та серверній стороні, відповідні поля підсвічуються помилкою.

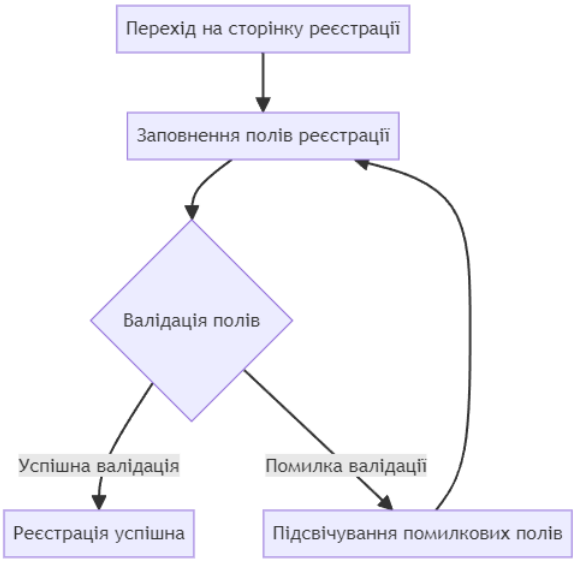


Рисунок 1.2 – Механізм створення облікового запису користувача

Опис авторизації в обліковий запис користувача:

– користувач переходить на сторінку авторизації;

– користувач заповнюю поля авторизації;

– якщо дані правильні, користивач потрапляє до головної сторінка додатку;
– якщо дані неправильні, сповіщення про помилку введення пари логін-пароль;

Рисунок 1.3 – Механізм авторизації в обліковий запис користувача

Опис планування подорожі на головному меню:

– Користувач обирає зі списку місце старту;
– Користувач обирає зі списку місце фінішу;
– Натискає на кпопку планування;
– Йде запит про персональні калькуляції;
– Якщо погоджено користувач вводе додаткові дані про поїздку;
  – Користувач отримує намальований на мапі маршрут подорожі в залежності від персональних калькуляцій, маршрут може бути оптимізований або ні.
  – Користувач бачить всі деталі поїздки, міста, де запропоновано зупинитись на перепочинок та орієнтовні ціни на пальне

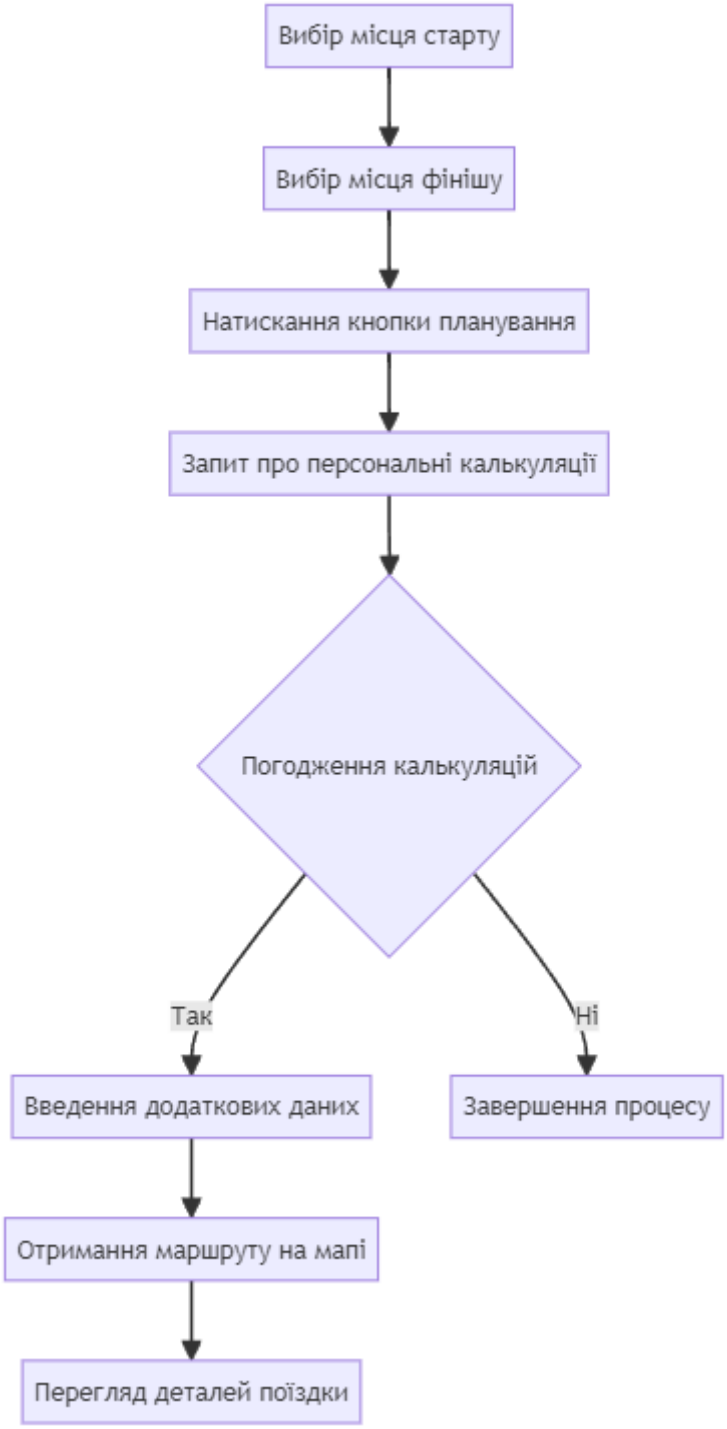


Рисунок 1.4 – Механізм планування подорожі на головному меню

Опис збереження подорожі:

- Користувач планує подорож;
- На вікні про деталі подорожі натискає на іконку збереження;
- Подорож зберігається на серверній стороні;

– Іконка збереження змінюється;

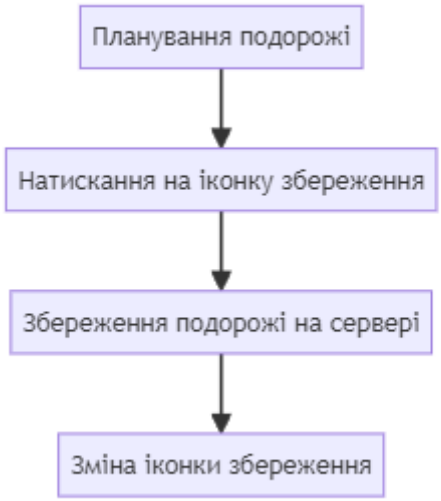


Рисунок 1.5 – Механізм збереження подорожі

Опис таблиці збережених подорожей:

– Користувач в боковому вікні обирає “Saved Trips”
– Користувач бачить всі збережені поїздки;
– Для отримання деталей потрібно натиснути на кнопку “Details” на карточці певної подорожі;

Рисунок 1.6 – Механізм таблиці збережених подорожей

Опис механізму виходу з облікового запису користувача:

– Користувач авторизується під своїм обліковим записом користувача;
– Користувач вікриває бокове меню;
– Натискає на кнопку “Log Out”;
– Користувач потрапляє на вікно авторизації;

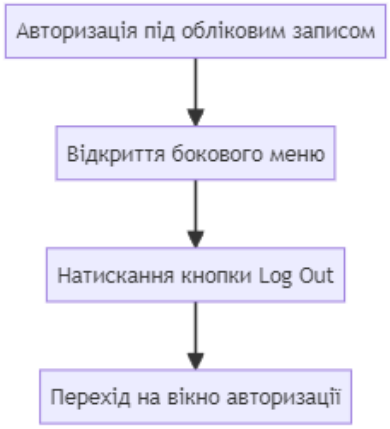


Рисунок 1.7 – Механізм виходу з облікового запису користувача

1.4 Постановка задачі

Мета розробки веб-додатку "TripPlanner" полягає в створенні зручного та ефективного інструменту для планування автомобільних подорожей. Основними завданнями проекту є наступні:

1. Реєстрація та авторизація користувачів:

   Розробка системи реєстрації та авторизації, яка забезпечить безпеку та конфіденційність даних користувачів.

2. Планування маршрутів:

   Реалізація інтерактивного інтерфейсу для вибору місця старту та фінішу, обрання проміжних пунктів, врахування додаткових параметрів подорожі (наприклад, тип транспорту, переваги маршруту тощо).

3. Візуалізація маршрутів на мапі:

   Інтеграція з Google Maps API для надання користувачам можливості візуалізації запланованих маршрутів на мапі з можливістю зміни масштабу, додавання маркерів та отримання інформації про маршрут.

4. Збереження та управління подорожами:

   Розробка механізму для збереження запланованих подорожей користувачів на сервері, можливість перегляду та управління збереженими подорожами.

Ці завдання вирішуються шляхом розробки веб-додатку з використанням .NET 8 Web API для серверної частини, Blazor Server Side UI для клієнтської частини, а також інтеграції з Google Maps API та OpenAI ChatGTP 3.5 Turbo. Вихідними даними є потреба користувачів у зручному інструменті для планування подорожей, а на виході отримати функціональний веб-додаток, що задовольняє ці вимоги та забезпечує позитивний досвід користувачів.

Висновки до розділу

У даному розділі було проведено аналіз предметної області, де розглянуто загальні положення та сучасний стан розробок у галузі планування автомобільних подорожей. Виконано порівняльний аналіз існуючих 7рішень, зокрема в порівнянні з розроблюваним веб-додатком.

Розглянуті відомі технічні рішення, що допоможуть у реалізації проекту, такі як архітектурні патерни, платформи, та архітектурні принципи. Основна архітектурна концепція включає в себе багатошарову архітектуру, використання Dependency Injection, OpenAPI, CodeFirst підхід, TestDriven Development, SOLID принципи, та об'єктно-орієнтований підхід. Були описані бізнес-процеси розробки, які включають послідовності створення облікового запису користувача, авторизації, планування подорожі, збереження та управління подорожами, а також вихід з облікового запису користувача. Кожен процес був наочно представлений за допомогою моделювання BPMN.

У постановці задачі були визначені мета розробки та завдання, що потрібно реалізувати для досягнення цієї мети. Розробка веб-додатку "TripPlanner" спрямована на створення зручного інструменту для планування автомобільних подорожей з використанням сучасних технологій та сервісів.

Загальною метою цього розділу є створення базового фундаменту для подальшої розробки веб-додатку, який задовольнятиме потреби користувачів у зручному інструменті для планування подорожей.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є надати можливість користувачам планувати свої подорожі, з гнучкістю та детальним прорахуванням, більше функцій можна побачити на рисунку 2.1, який знаходиться у додатку.

В таблицях 2.1 - 2.9 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	1. Користувач переходить на сторінку реєстрації. 2. Вводяться відповідні дані: логін, пароль в системі. 3. Після заповнення даних користувач натискає кнопку реєстрації. 4. З’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу

Таблиця 2.2 - Варіант використання UC-2

Use case name	Авторизація користувача
Use case ID	UC-02
Goals	Авторизація користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає увійти в систему
Pre-conditions	-
Flow of Events	1. Користувач переходить на сторінку авторизації. 2. Вводиться пошта користувача та пароль. 3. Користувач натискає кнопку входу.
Extension	В разі неправильного введення логіну або паролю виводиться повідомлення про помилку.
Post-Condition	Вхід в обліковий запис користувача, перехід на головну сторінку додатку.

Таблиця 2.3 - Варіант використання UC-3

Use case name	Планування подорожі
Use case ID	UC-03
Goals	Планування маршруту подорожі користувача
Actors	Зареєстрований користувач
Trigger	Користувач бажає спланувати свою подорож
Pre-conditions	Авторизація в системі
Flow of Events	1. Користувач обирає місце старту та фінішу зі списку. 2. Обирає проміжні точки маршруту. 3. Натискає на кнопку планування. 4. Вводить додаткові дані про поїздку, якщо потрібно. 5. Після підтвердження користувач отримує намальований на мапі маршрут подорожі.
Extension	В разі некоректного вводу даних виводиться повідомлення про помилку.
Post-Condition	Отримання маршруту подорожі на мапі та можливість додаткового перегляду деталей поїздки.

Таблиця 2.4 - Варіант використання UC-4

Use case name	Збереження подорожі
Use case ID	UC-04
Goals	Збереження запланованої подорожі користувачем
Actors	Зареєстрований користувач
Trigger	Користувач бажає зберегти свою заплановану подорож
Pre-conditions	Авторизація в системі
Flow of Events	1. Користувач планує подорож за допомогою функціоналу планування подорожі. 2. На сторінці деталей подорожі користувач натискає кнопку "Зберегти". 3. Подорож зберігається на серверній стороні. 4. Кнопка "Зберегти" змінює свій вигляд для вказання, що подорож збережена.
Extension	У разі, якщо під час спроби зберегти подорож виникає помилка під час збереження, виводиться повідомлення з поясненням про проблему.
Post-Condition	Збереження запланованої подорожі користувачем з можливістю подальшого перегляду та управління.

Таблиця 2.5 - Варіант використання UC-5

Use case name	Перегляд збережених подорожей
Use case ID	UC-05
Goals	Перегляд списку та деталей збережених подорожей
Actors	Зареєстрований користувач
Trigger	Користувач бажає переглянути свої збережені подорожі
Pre-conditions	Авторизація в системі
Flow of Events	1. Користувач переходить до розділу "Збережені подорожі". 2. Переглядає список збережених подорожей.

	3. Для отримання деталей певної подорожі користувач натискає на кнопку "Деталі" біля потрібного запису. 4. Відображаються деталі обраної подорожі.
Extension	У разі, якщо список збережених подорожей порожній, виводиться повідомлення про відсутність збережених подорожей.
Post-Condition	Перегляд списку та деталей збережених подорожей користувачем.

Таблиця 2.6 - Варіант використання UC-6

Use case name	Вихід з облікового запису користувача
Use case ID	UC-06
Goals	Вихід користувача з облікового запису
Actors	Зареєстрований користувач
Trigger	Користувач бажає переглянути свої збережені подорожі
Pre-conditions	Авторизація в системі
Flow of Events	1. Користувач у вікні додатку відкриває бокове меню. 2. Натискає на кнопку "Вийти" або аналогічний пункт меню. 3. Користувач перенаправляється на сторінку авторизації.
Extension	Підтвердження виходу з облікового запису попереднім підтвердженням або виведенням додаткового підтвердження на вибір користувача.
Post-Condition	Вихід користувача з облікового запису та перенаправлення на сторінку авторизації.

Таблиця 2.7 - Варіант використання UC-7

Use case name	Перегляд додаткових деталей поїздки
Use case ID	UC-07
Goals	Перегляд детальної інформації про заплановану поїздку
Actors	Зареєстрований користувач

Trigger	Користувач бажає отримати більше інформації про свою заплановану поїздку
Pre-conditions	Авторизація в системі
Flow of Events	1. Користувач переходить до списку збережених подорожей. 2. Обирає певну поїздку зі списку. 3. Натискає на кнопку "Деталі" або аналогічний елемент для перегляду додаткових деталей.
Extension	У випадку, якщо під час завантаження деталей поїздки виникає помилка, виводиться повідомлення про це.
Post-Condition	Перегляд детальної інформації про заплановану поїздку користувачем.

Таблиця 2.8 - Варіант використання UC-8

Use case name	Видалення подорожі
Use case ID	UC-08
Goals	Видалення запланованої подорожі користувачем
Actors	Зареєстрований користувач
Trigger	Користувач бажає видалити певну заплановану подорож
Pre-conditions	Авторизація в системі
Flow of Events	1. Користувач переходить до списку збережених подорожей. 2. Обирає певну подорож зі списку. 3. Натискає на кнопку "Видалити" або аналогічний елемент для видалення обраної подорожі.
Extension	У випадку підтвердження видалення, виводиться підтвердження та подальше видалення подорожі зі списку.
Post-Condition	Видалення обраної подорожі зі списку користувача з можливістю підтвердження дії.

Таблиця 2.9 - Варіант використання UC-9

Use case name	Отримання інформації про пальне
Use case ID	UC-09
Goals	Отримання інформації про ціни на пальне під час планування поїздки
Actors	Зареєстрований користувач
Trigger	Користувач бажає дізнатися ціни на пальне в місці планованої зупинки
Pre-conditions	Авторизація в системі
Flow of Events	1. Під час планування поїздки користувач вводить маршрут та додаткові дані. 2. Після цього система автоматично отримує дані про ціни на пальне у місці запланованої зупинки. 3. Ці дані відображаються користувачеві під час планування поїздки.
Extension	У випадку неможливості отримання інформації про прогноз погоди, користувачеві надається повідомлення про недоступність цієї інформації.
Post-Condition	Отримання інформації про прогноз погоди на маршрут

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.10 наведено загальну модель вимог, а в таблиці 2.11 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.1.

Таблиця 2.10 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики

1	Перегляд головної сторінки	FR-01	Високий	Середній
2	Реєстрація користувача	FR-02	Високий	Високий
3	Авторизація користувача	FR-03	Високий	Високий
4	Вихід з облікового запису користувача	FR-04	Середній	Низький
5	Планування автомобільних подорожей	FR-05	Високий	Середній
6	Збереження запланованих поїздок	FR-06	Високий	Середній
7	Перегляд збережених поїздок	FR-07	Середній	Низький
8	Видалення запланованих поїздок	FR-08	Високий	Середній
9	Отримання інформації про ціни на пальне	FR-09	Високий	Середній

Таблиця 2.11 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-01	Перегляд головної сторінки Незареєстрований користувач бачить головну сторінку додатку, де може отримати інформацію про можливості реєстрації або авторизації.
FR-02	Реєстрація користувача Користувач може створити обліковий запис у системі, вводячи необхідні дані та підтверджуючи їх.
FR-03	Авторизація користувача Користувач може увійти до свого облікового запису, використовуючи вже існуючі дані авторизації.

Продовження таблиці 2.11

FR-04	Вихід з облікового запису користувача Користувач може вийти зі свого облікового запису для безпеки та приватності.
FR-05	Планування автомобільних подорожей Користувач може планувати свої майбутні автомобільні подорожі, вибираючи маршрути, зупинки та інші параметри.
FR-06	Збереження запланованих поїздок Користувач може зберігати свої заплановані поїздки для подальшого використання.
FR-07	Перегляд збережених поїздок Користувач може переглядати список своїх збережених поїздок.
FR-08	Видалення запланованих поїздок Користувач може видаляти свої заплановані поїздки зі свого списку.
FR-09	Отримання інформації про ціни на пальне Користувач може отримувати інформацію про ціни на пальне у різних місцях.

	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07	FR-08	FR-09
UC-01	+								
UC-02		+		+					
UC-03			+					+	+
UC-04				+					
UC-05					+				
UC-06						+			
UC-07						+	+		
UC-08								+	
UC-09									+

Рисунок 2.1 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

У веб-застосунку, що розроблятиметься в рамках дипломного проєктування, виділимо наступні нефункціональні вимоги:

– система має світлий інтерфейс;

– система має інтерфейси з читабельними шрифтами зі стандартної Windows бібліотеки.

– система має інферфейс англійською мовою;

– система є пристосованою до підтримування та відновлення;

– система є надійною; – система повинна працювати на різних платформах та різних за розміром девайсах без зміни зручності її використання.

Висновки до розділу

У цьому розділі були розглянуті загальні положення, було описано та проаналізовано предметну область. Були проаналізовані відомі технічні рішення: архітектури програмного забезпечення, архітектурні патерни.

Були наведені їхні переваги та обґрунтування, чому саме вони використовуватимуться в роботі. Було проаналізовано програмні засоби й засоби розробки, які найкраще використовувати для застосунку, та обґрунтовано вибір мов програмування, бібліотек, фреймворків, текстового редактора та інтегрованого середовища розробки.

Було проведено порівняльний аналіз застосунку з іншим програмним продуктом, який майже повністю реалізує його функціонал. Було проведено аналіз вимог до програмного забезпечення: розроблено варіанти використання, функціональні та нефункціональні вимоги, створено матрицю трасування вимог. Відбулися постановка задачі та аналіз вимог до програмного забезпечення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Розроблене програмне забезпечення для планування автомобільних подорожей базується на багатошаровій архітектурі, яка включає чотири основні шари: Веб API (контролери), бізнес-логіка, база даних та основний проект (core) з головними контрактами та моделями. Такий підхід забезпечує чіткий поділ обов'язків між різними частинами системи, що підвищує її масштабованість, підтримуваність та тестованість.

Для розробки програмного забезпечення було обрано наступний технологічний стек:

- Back-end:
  - Мова програмування: C#
  - Фреймворк: .NET 8
  - Архітектурний патерн: Багатошарова архітектура
  - Бібліотеки: Entity Framework Core, OpenAI nuget, Swagger
  - База даних: SQL Server
- Front-end:
  - UI фреймворк: Blazor Server Side
  - Бібліотеки: Blazor, JavaScript для інтеграції з Google Maps API
- Інтеграції:
  - API: Google Maps API для картографічних функцій
  - OpenAI API: Для отримання динамічних даних

Контролери є верхнім шаром архітектури і відповідають за прийом HTTP-запитів від клієнтів та повернення відповідних HTTP-відповідей. Вони є точкою входу до системи і взаємодіють із сервісами бізнес-логіки для обробки запитів.

Контролери:

- Отримують HTTP-запити.
- Викликають відповідні методи сервісів бізнес-логіки.
- Повертають результати клієнту у вигляді HTTP-відповідей.

Контролери не містять бізнес-логіки, що дозволяє тримати їх "тонкими" та легко підтримуваними. Вони виконують роль посередників між клієнтами і бізнес-логікою.

Другий шар – це шар бізнес логіки, містить сервіси, які обробляють запити, надані контролерами, виконують бізнес-операції і взаємодіють з репозиторіями для доступу до даних [2].

Сервіси:

- Виконують бізнес-операції.
- Взаємодіють з репозиторіями для отримання та збереження даних.
- Інкапсулюють бізнес-логіку, забезпечуючи її повторне використання та тестованість.

Шар доступу до даних відповідає за взаємодію з базою даних. Він реалізує патерн Repository, що забезпечує абстракцію над прямими операціями з базою даних [3].

Репозиторії:

- Забезпечують методи для отримання, збереження, оновлення та видалення даних.
- Використовують Entity Framework з підходом Code First для створення та управління схемою бази даних.
- Кожна таблиця має свій репозиторій, що дозволяє чітко організувати доступ до даних.

Шар ядра містить основні контракти, моделі та інтерфейси, які використовуються у всіх інших шарах. Він визначає структуру даних та основні правила взаємодії між компонентами системи.

Моделі:

- Визначають структуру даних, які використовуються в системі.
- Відповідають за визначення сутностей, що зберігаються у базі даних.

Інтерфейси:

- Визначають контракти для сервісів та репозиторіїв.
- Забезпечують інверсію залежностей, що дозволяє легко замінювати реалізації компонентів.

Для управління базою даних використовується Entity Framework з підходом Code First. Це дозволяє визначати моделі даних у вигляді класів, а Entity Framework автоматично створює та підтримує схему бази даних відповідно до цих моделей [4].

Code First:

- Моделі даних визначаються у вигляді класів C#.
- Entity Framework створює схему бази даних на основі цих класів.
- Міграції дозволяють автоматично оновлювати схему бази даних при зміні моделей.

Патерн Repository забезпечує абстракцію над прямими операціями з базою даних, що дозволяє легко замінювати або змінювати спосіб збереження даних без зміни бізнес-логіки [5].

Репозиторії:

- Кожен репозиторій відповідає за певну сутність (таблицю) в базі даних.
- Включають методи для отримання, збереження, оновлення та видалення даних.
- Сервіси використовують репозиторії для доступу до даних, не взаємодіючи з контекстом бази даних напряму.

Для керування залежностями між компонентами використовується механізм Dependency Injection (DI), що дозволяє впроваджувати залежності через конструктори або методи [6].

Dependency Injection:

- Забезпечує інверсію залежностей, дозволяючи легко замінювати реалізації компонентів.
- Полегшує тестування, дозволяючи використовувати макети (mock) замість реальних реалізацій.

Для обробки помилок у додатку використовується Middleware, що дозволяє централізовано перехоплювати та обробляти всі виключення.

Middleware:

- Перехоплює виключення, що виникають у додатку.
- Логує помилки та повертає користувачам дружні повідомлення про помилки.
- Забезпечує єдиний механізм обробки помилок у всьому додатку.

Для динамічного отримання даних використовується інтеграція з OpenAI API за допомогою NUGET бібліотеки OpenAI.

- Інтеграція з OpenAI:
  - Використовується для отримання динамічних даних під час планування подорожей.
  - Забезпечує зв'язок із зовнішніми сервісами для надання додаткових функціональних можливостей користувачам [8].
- Принципи SOLID та ООП

При розробці додатку дотримуються принципів SOLID та об'єктно-орієнтованого програмування (ООП), що забезпечує якісну та легко підтримувану архітектуру, а саме:

- Single Responsibility Principle (Принцип єдиної відповідальності)
- Open/Closed Principle (Принцип відкритості/закритості)

- Liskov Substitution Principle (Принцип підстановки Лісков)
- Interface Segregation Principle (Принцип розділення інтерфейсів)
- Dependency Inversion Principle (Принцип інверсії залежностей)

А також принципи ООП:

- Використання класів та об'єктів для організації коду.
- Інкапсуляція, наслідування, поліморфізм та абстракція для створення модульного та повторно використовуваного коду.

3.2 Обґрунтування вибору засобів розробки

Для розробки серверної частини веб-додатку було обрано мову програмування C# та фреймворк .NET 8. Це рішення було прийнято з огляду на наступні переваги:

- Стабільність і продуктивність: .NET є потужним фреймворком, який забезпечує високу продуктивність і стабільність веб-додатків.
- Підтримка багатошарової архітектури: .NET легко підтримує багатошарову архітектуру, що важливо для побудови масштабованого та добре структурованого додатку.
- Розширюваність: .NET має багатий набір бібліотек і пакетів, які дозволяють швидко інтегрувати сторонні сервіси та розширювати функціональність додатку.
- Спільнота і підтримка: Велика спільнота розробників та хороша документація дозволяють швидко знаходити рішення для різних задач і проблем [9].

Для роботи з базою даних було обрано Entity Framework Core з підходом Code First. Це обрання обґрунтовано наступними причинами:

- Автоматичне створення схем баз даних: Підхід Code First дозволяє визначати моделі даних у вигляді класів, а Entity Framework автоматично створює відповідну схему бази даних [10].

- Міграції: Entity Framework Core підтримує міграції, що дозволяє легко оновлювати схему бази даних при зміні моделей.
- Простота використання: Entity Framework надає інтуїтивно зрозумілий API для роботи з базою даних, що зменшує час на розробку та налаштування.

Для реалізації інтерфейсу користувача було обрано Blazor Server Side. Цей вибір обґрунтований такими факторами:

- Єдина мова програмування: Blazor дозволяє використовувати C# для розробки як серверної, так і клієнтської частини додатку, що спрощує процес розробки [11].
- Компонентна архітектура: Blazor підтримує компонентний підхід, що дозволяє створювати повторно використовувані та ізольовані компоненти UI.
- Інтеграція з JavaScript: Blazor легко інтегрується з JavaScript, що дозволяє використовувати сторонні бібліотеки, такі як Google Maps API.
- Реактивність: Blazor забезпечує високий рівень інтерактивності та швидкість роботи користувацького інтерфейсу.

3.3 Конструювання програмного забезпечення

Оригінальні алгоритми

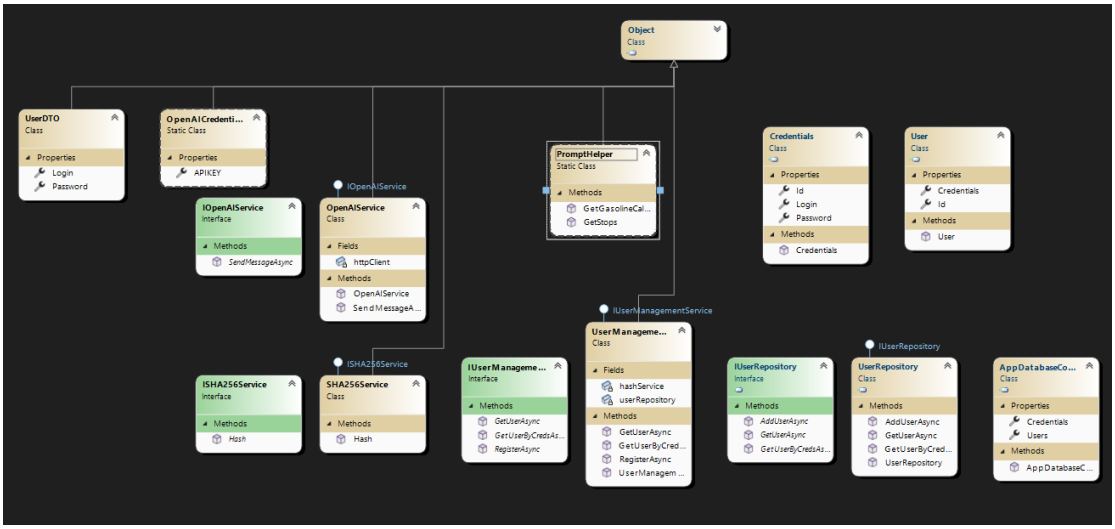
1. Алгоритм планування маршруту:
   - Визначає оптимальний маршрут з точки А до точки Б з урахуванням різних параметрів, таких як відстань, час в дорозі та зупинки в запропонованих містах.
   - Використовує Google Maps API для отримання даних про дороги, трафік та інші фактори, що впливають на маршрут.

- Враховує індивідуальні побажання користувача та оптимізує маршрут відповідно до них.

2. Алгоритм збереження та управління подорожами:
   - Зберігає заплановані подорожі користувача в базі даних.
   - Надає можливість редагування та видалення збережених подорожей.
   - Відслідковує історію збережених подорожей та дозволяє користувачу переглядати деталі кожної з них.

4 Модифікації існуючих рішень

- Інтеграція з OpenAI API:
  - Використання OpenAI ChatGPT для динамічного отримання інформації та рекомендацій щодо подорожей [12].
  - Модифікація стандартних запитів до API для врахування специфічних потреб користувачів у контексті планування подорожей.



- Рисунок 3.1 – Схема структурна класів

В якості системи управління базами даних використовується SQL SERVER. База даних серверу призначена для зберігання користувачів, а також даних про їх збережі подорожі Опис таблиць бази даних наведено у таблицях 3.1 - 3.5. Модель бази даних наведена на рисунку 3.2.

Таблиця 3.1 – Опис таблиці Users

Назва поля	Тип даних	Опис
id	int	Ідентифікаційний номер користувача
credentials_id	int	Посилання на запис у таблиці credentials, де зберігаються облікові дані користувача

Таблиця 3.2 – Опис таблиці Credentials

Назва поля	Тип даних	Опис
id	int	Ідентифікаційний номер облікових даних
login	varchar	Логін користувача
password	varchar	Пароль користувача

Таблиця 3.3 – Опис таблиці SavedTrips

Назва поля	Тип даних	Опис
id	int	Ідентифікаційний номер збереженої подорожі
saved_date	datetime	Дата збереження подорожі
trip_details_id	int	Посилання на запис у таблиці trip_details, де зберігаються деталі подорожі

Таблиця 3.4 – Опис таблиці TripDetails

Назва поля	Тип даних	Опис
id	int	Ідентифікаційний номер деталей подорожі

Продовження таблиці 3.4

from	varchar	Початкова точка подорожі
destination	varchar	Пункт призначення подорожі
distance	int	Відстань між початковою точкою та пунктом призначення (в км)
duration	time	Тривалість подорожі

Таблиця 3.5 – Опис таблиці ProposedCities

Назва поля	Тип даних	Опис
id	int	Ідентифікаційний номер запропонованого міста
city_name	varchar	Назва міста
trip_details_id	int	Посилання на запис у таблиці trip_details

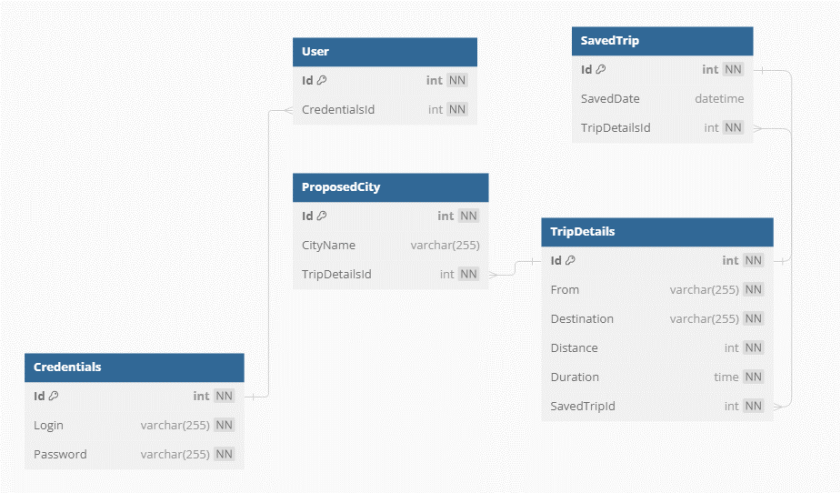


Рисунок 3.2 – Діаграма бази даних

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.6.

Таблиця 3.6 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Radzen Blazor Studio	Головне середовище розробки проектування клієнтської частини у Blazor при використанні Radzen Component Library.
2	Visual Studio Code	Універсальний текстовий редактор, використовувався для розробки клієнтського коду.
3	Visual Studio 2022	Інтегроване середовище розробки, використовувалось для розробки клієнтської та серверної частини проекту.
4	SQL Server Management Studio	Графічний інтерфейс для роботи з базою даних SQL Server, використовувався для управління та адміністрування бази даних.
5	Postman	Програмне забезпечення для тестування REST запитів та взаємодії з API інтерфейсами.

Висновки до розділу

У даному розділі було детально розглянуто процес конструювання та розроблення програмного забезпечення для планування автомобільних подорожей. Зокрема, було розроблено архітектурний патерн на основі багатошарової архітектури, що включає WebAPI (контролери), бізнес-логіку, базу даних та основний проект з головними контрактами (моделями) [13]. Для

доступу до даних використовується патерн Repository, де кожна таблиця має свій репозиторій. Було застосовано принципи Dependency Injection для забезпечення гнучкості та тестованості коду. Контролери мають тонку реалізацію, звертаючись лише до сервісів без виконання бізнес-логіки безпосередньо. Реалізовано глобальну обробку помилок через Middleware, використовуючи принципи ООП та SOLID.

Деталізовано основні компоненти бекенд-частини додатку, включаючи контролери, сервіси, репозиторії та моделі. Пояснено інтеграцію з зовнішніми API, зокрема з OpenAI API, за допомогою бібліотеки OpenAI NuGet. Представлено структури даних для сутностей: Credentials, ProposedCity, SavedTrip, TripDetails, User. Наведено детальні описи таблиць бази даних із зазначенням типів даних і їх призначення [14]. Описано концептуальну модель бази даних, яка відображає зв'язки між сутностями та їх атрибутами.

У розробці використовувалися такі інструменти, як Radzen Blazor Studio, Visual Studio Code, Visual Studio 2022, SQL Server Management Studio та Postman. Вказано їх призначення та роль у процесі розробки програмного забезпечення. Надано опис алгоритмів, що використовуються для планування маршрутів, обробки даних та інтеграції з зовнішніми сервісами.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Метою тестування є наступне:

- Перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- Перевірка збереження даних;
- Перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox та інші);
- Перевірка сумісності застосунку з різними операційними системами (Windows, Linux та інші);
- Знаходження проблем, помилок і недоліків з метою їх усунення;
- Перевірка зручності графічного інтерфейсу.

Метриками для оцінки якості ПЗ обрано наступні:

- Швидкість завантаження сторінок.

  Час, необхідний для повного завантаження сторінки після запиту користувача. Важливим є забезпечення швидкого доступу до функціональності додатку.

- Коректність роботи функціональних вимог.

  Відсоток успішно виконаних тестових випадків, які перевіряють відповідність функціональних вимог. Висока коректність вказує на правильну роботу додатку згідно з ТЗ.

- Надійність збереження даних.

  Перевірка цілісності даних після операцій запису, оновлення та видалення. Ця метрика оцінюється через ручне та автоматизоване тестування збережених даних у базі даних.

– Сумісність з браузерами.

Кількість успішно виконаних тестів на різних браузерах. Додаток має коректно працювати в останніх версіях Chrome, Opera, Firefox та інших популярних браузерах.

– Сумісність з операційними системами.

Кількість успішно виконаних тестів на різних операційних системах. Перевірка проводиться на Windows, Linux та інших ОС для забезпечення максимальної доступності додатку.

– Кількість знайдених і усунутих помилок.

Число помилок, виявлених під час тестування, і тих, які були виправлені. Ця метрика допомагає оцінити загальну стабільність та якість додатку.

– Зручність графічного інтерфейсу.

Оцінка користувачами зручності використання інтерфейсу. Зручність інтерфейсу оцінюється за допомогою анкетування та тестування з залученням кінцевих користувачів.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1– 4.10.

Таблиця 4.1 – Тест 1.1 Реєстрація користувача

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Логін, пароль
Опис проведення тесту	У відповідні поля вводяться: логін, який до цього не була зареєстрована в системі, пароль з англійскими

	символами. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Авторизація користувача

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Логін, пароль
Опис проведення тесту	У відповідні поля вводяться: коректний логін, який зареєстрована в системі, та правильний пароль. Після цього натискається кнопка входу.
Очікуваний результат	Користувач успішно авторизується і перенаправляється на головну сторінку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Вихід з облікового запису

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Немає
Опис проведення тесту	Користувач відкриває бокове меню та натискає кнопку "Log Out".
Очікуваний результат	Користувач виходить з облікового запису і перенаправляється на сторінку авторизації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.4 Планування подорожі

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Місце старту, місце фінішу
Опис проведення тесту	Користувач обирає зі списку місце старту та місце фінішу, натискає кнопку "Plan Trip".
Очікуваний результат	Система відображає на мапі маршрут подорожі з деталями поїздки, містами для зупинок та орієнтовними цінами на пальне.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.5 Збереження подорожі

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Немає
Опис проведення тесту	Користувач натискає на іконку збереження на вікні деталей подорожі.
Очікуваний результат	Подорож зберігається на серверній стороні, іконка збереження змінюється.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.6 Перегляд збережених подорожей

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**

Вхідні дані	Немає
Опис проведення тесту	Користувач відкриває бокове меню та обирає "Saved Trips".
Очікуваний результат	Система відображає всі збережені поїздки користувача.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.7 Перегляд деталей збереженої подорожі

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Немає
Опис проведення тесту	Користувач натискає на кнопку "Details" на карточці певної подорожі.
Очікуваний результат	Система відображає деталі обраної подорожі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 1.8 Перевірка наявності помилок при реєстрації

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Некоректна логін, пароль
Опис проведення тесту	У відповідні поля вводяться: некоректний логін, пароль, який не відповідає.
Очікуваний результат	Система підсвічує некоректні поля та виводить повідомлення про помилки.

Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 1.9 Перевірка наявності помилок при авторизації

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Некоректний логін, неправильний пароль
Опис проведення тесту	У відповідні поля вводяться: логін, який не зареєстрований в системі, та неправильний пароль.
Очікуваний результат	Система виводить повідомлення про помилку введення пари логін-пароль.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 1.10 Перевірка сумісності з браузерами

Початковий стан системи	**Користувач знаходиться на сторінці реєстрації**
Вхідні дані	Немає
Опис проведення тесту	Користувач відкриває додаток у різних браузерах (Chrome, Opera, Firefox, Edge, Safari).
Очікуваний результат	Додаток коректно відображається та працює у всіх зазначених браузерах.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Контрольний приклад: Планування та збереження подорожі у додатку TripPlanner

Крок 1: Авторизація користувача

Початковий стан: Користувач знаходиться на сторінці авторизації.

Дії: Введіть логін та пароль у відповідні поля. Натисніть кнопку "Log In".

Очікуваний результат: Користувач успішно авторизується і перенаправляється на головну сторінку.

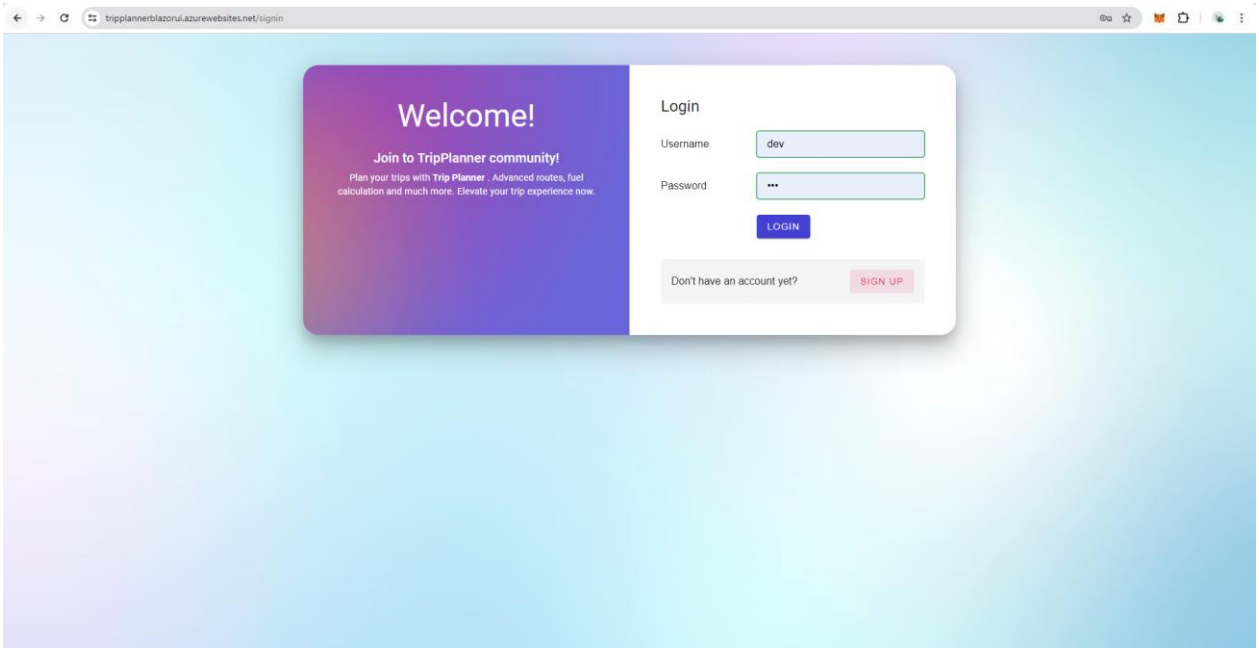


Рисунок 4.1 – Виконання першого кроку контрольного прикладу

Крок 2: Планування нової подорожі

Початковий стан: Користувач авторизований і знаходиться на головній сторінці.

Дії: Натисніть на кнопку "Plan a Trip". Введіть місце старту та місце призначення у відповідні поля. Натисніть кнопку "Plan".

Очікуваний результат: На мапі відображається маршрут подорожі з деталями, включаючи відстань, тривалість подорожі, рекомендовані зупинки та орієнтовні витрати на пальне.

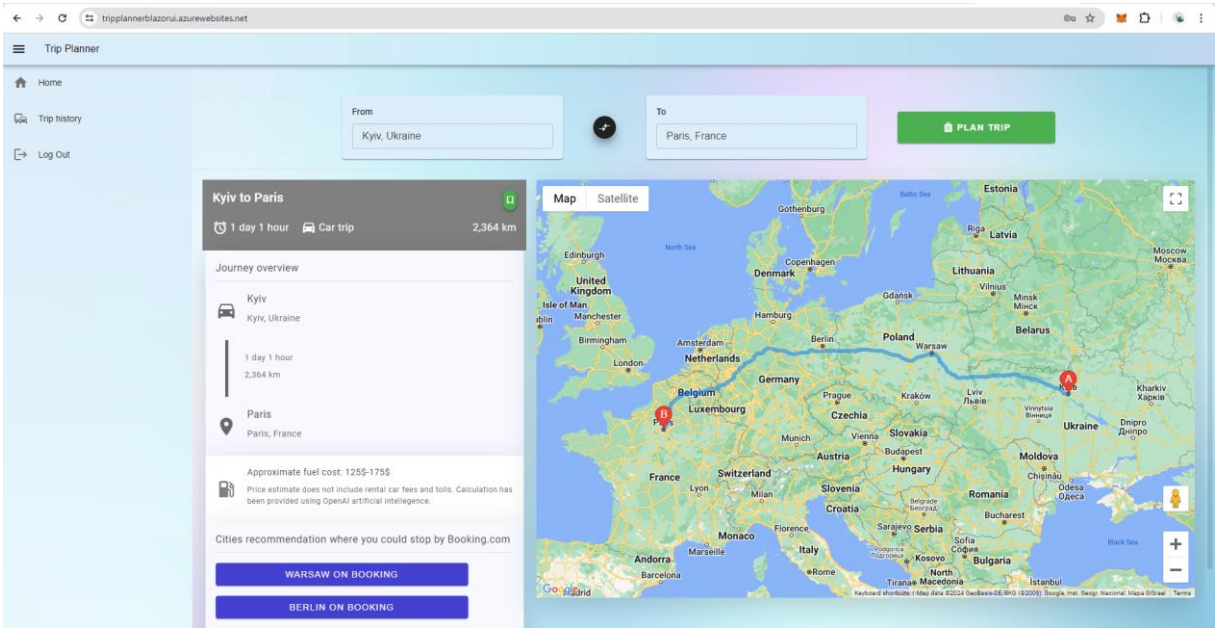


Рисунок 4.2 – Виконання другого кроку контрольного прикладу

Крок 3: Перегляд деталізованої інформації про подорож

Початковий стан: Користувач бачить запланований маршрут на мапі.

Дії: Натисніть на деталі подорожі для перегляду додаткової інформації, такої як рекомендовані міста для зупинок та інші деталі маршруту.

Очікуваний результат: Відображається детальна інформація про подорож.

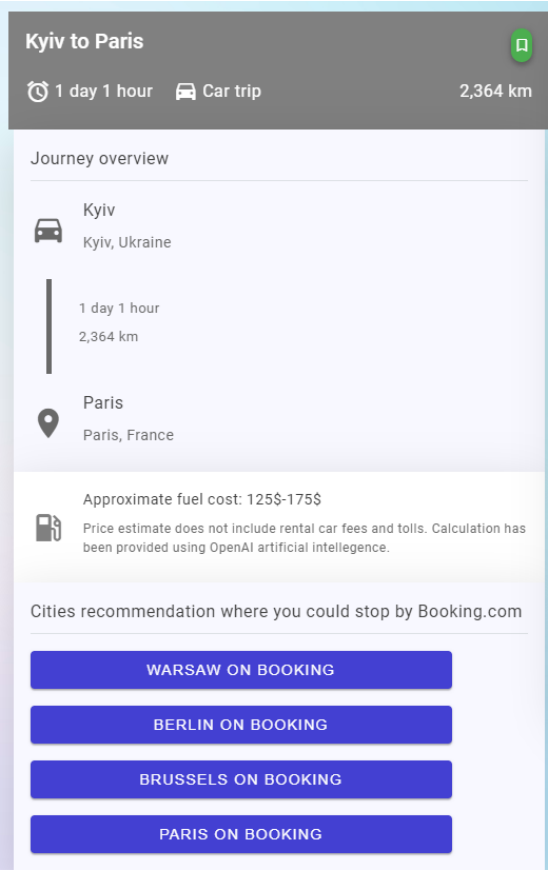


Рисунок 4.3 – Виконання третього кроку контрольного прикладу

Крок 4: Збереження подорожі

Початковий стан: Користувач переглядає деталі запланованої подорожі.

Дії: Натисніть на кнопку "Save Trip".

Очікуваний результат: Подорож зберігається у профілі користувача. З'являється повідомлення про успішне збереження.

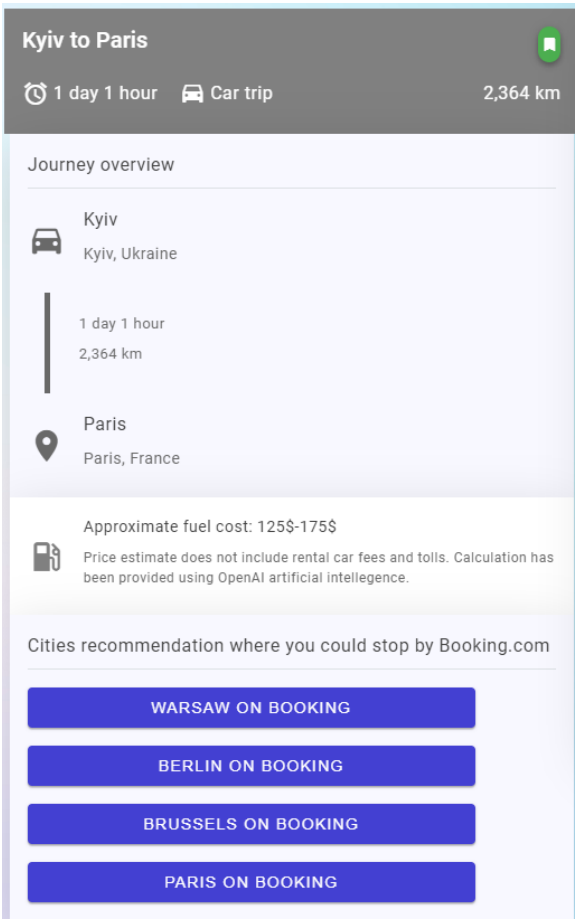


Рисунок 4.4 – Виконання четвертого кроку контрольного прикладу

Крок 5: Перегляд збережених подорожей

Початковий стан: Користувач авторизований і знаходиться на головній сторінці.

Дії: Відкрийте бокове меню. Оберіть "Saved Trips".

Очікуваний результат: Відображається список всіх збережених подорожей користувача.

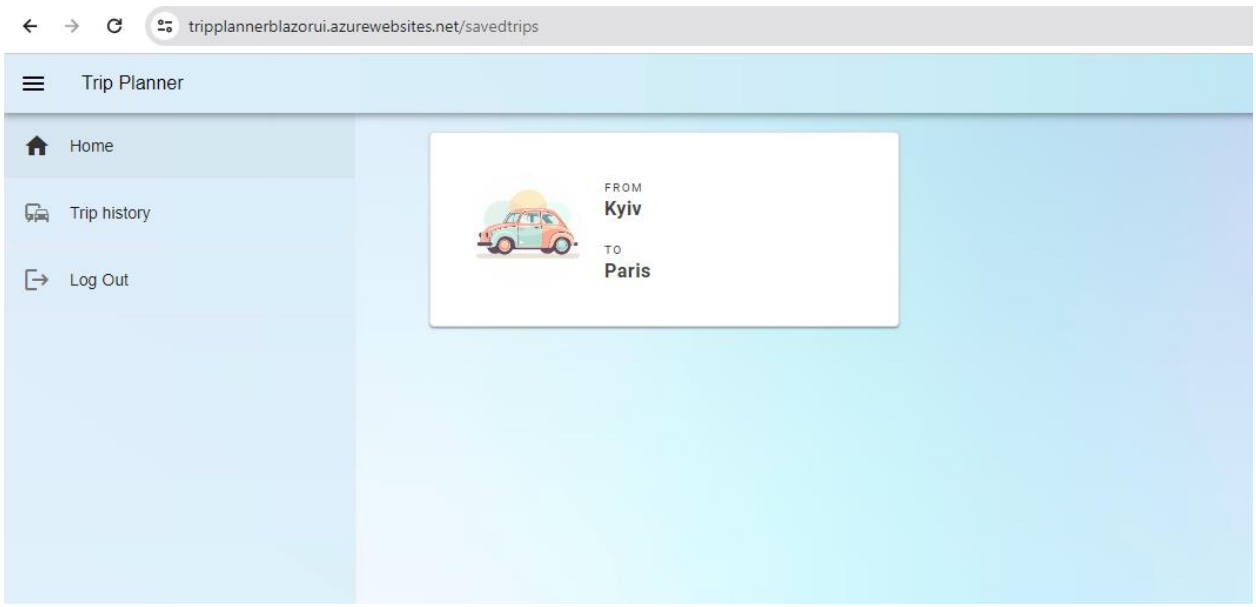


Рисунок 4.5 – Виконання п’ятого кроку контрольного прикладу

Крок 6: Перегляд деталей збереженої подорожі

Початковий стан: Користувач знаходиться на сторінці збережених подорожей.

Дії: Натисніть на кнопку "Details" поруч з потрібною подорожжю.

Очікуваний результат: Відображаються деталі обраної подорожі.

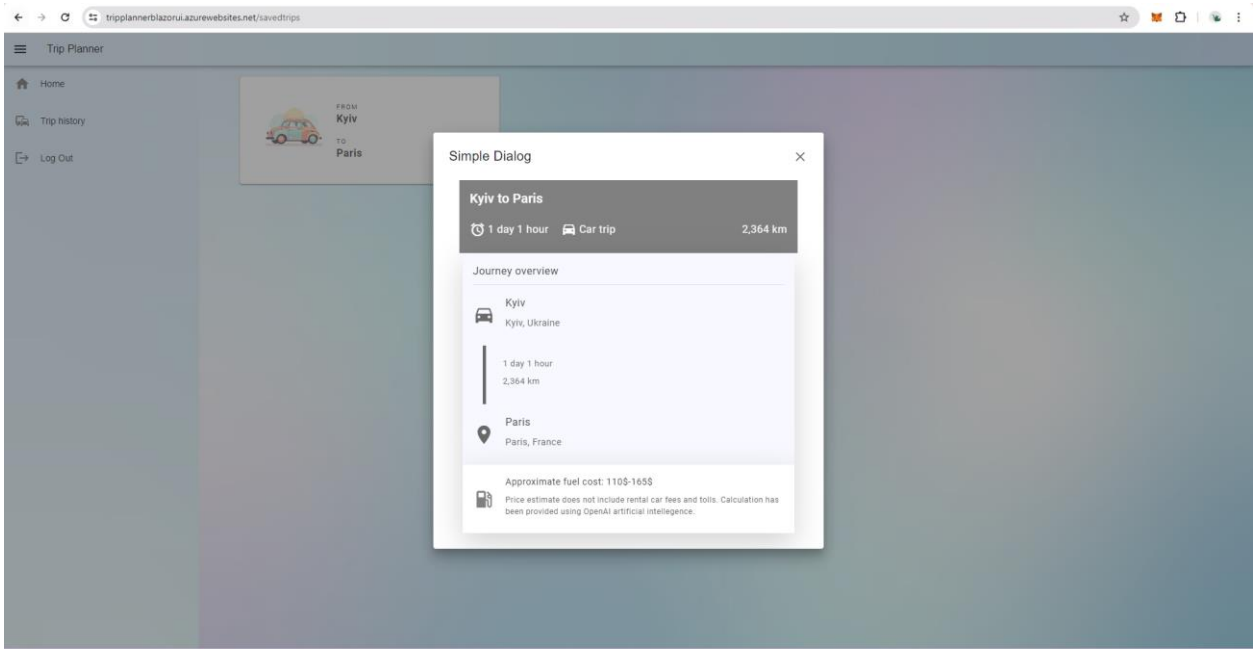


Рисунок 4.6 – Виконання шостого кроку контрольного прикладу

Висновки до розділу

У цьому розділі було здійснено аналіз якості програмного забезпечення TripPlanner, що включало проведення тестування для оцінки відповідності функціональних вимог, збереження даних та сумісності з різними браузерами і операційними системами. Метою тестування було перевірити коректність роботи додатку, забезпечити зручність користування графічним інтерфейсом та виявити можливі проблеми для їх усунення.

Було розглянуто метрики для оцінки якості ПЗ, зокрема швидкість виконання, стабільність, сумісність, зручність використання та інші. За результатами тестування, додаток продемонстрував високу відповідність заявленим функціональним вимогам, забезпечуdючи стабільну роботу і зручний інтерфейс для користувачів.

Проведено розробку та тестування кількох тестових сценаріїв, які охоплювали основні функції додатку, такі як реєстрація користувача, авторизація, планування подорожі, збереження та перегляд збережених подорожей. Для кожного сценарію було створено відповідні таблиці з початковими умовами, вхідними даними, описом проведення тесту, очікуваними та фактичними результатами. Всі тестові випадки були успішно виконані, що підтверджує правильність та надійність роботи додатку.

Було детально описано контрольний приклад використання додатку TripPlanner, що включає всі можливі розгалуження та особливості. Кроки були доповнені ілюстраціями для наочності. Цей приклад демонструє, як користувач може взаємодіяти з додатком для планування, збереження та перегляду подорожей, забезпечуючи високу якість обслуговування та зручність використання.

Проведені тестування та контрольні приклади підтвердили, що додаток TripPlanner є функціональним, надійним та зручним у використанні. Додаток успішно виконує всі заявлені функції, відповідаючи сучасним вимогам до програмного забезпечення.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Розгортання додатку TripPlanner виконувалось на платформі Azure, використовуючи ресурсну групу "diploma-rg". Цей процес включає в себе кілька етапів, кожен з яких важливий для забезпечення коректної та надійної роботи додатку.

Першим кроком було створення ресурсної групи "diploma-rg", яка служить контейнером для всіх ресурсів, пов'язаних з додатком. Ресурсна група дозволяє централізовано керувати всіма ресурсами, що спрощує процеси розгортання, моніторингу та обслуговування.

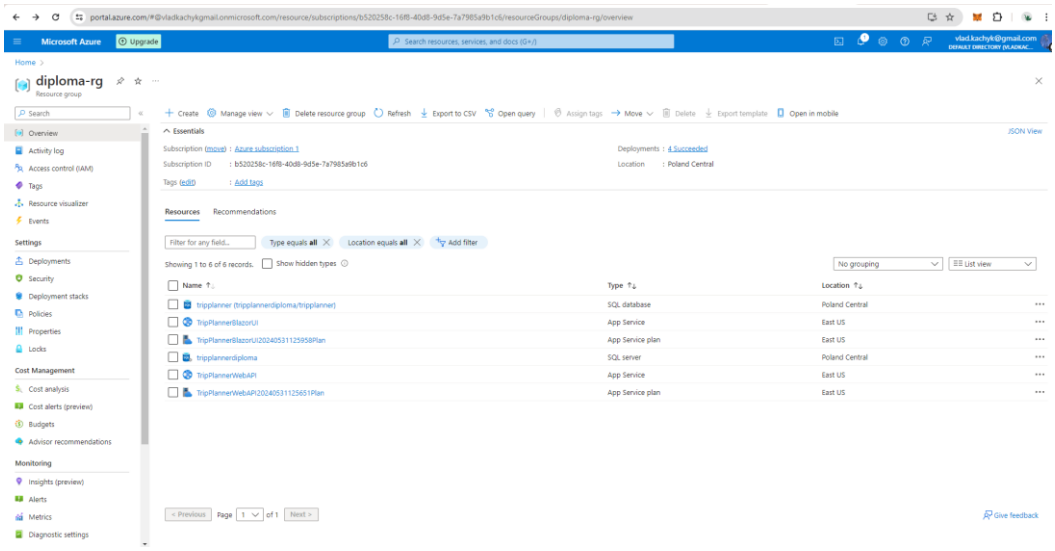


Рисунок 5.1– Створена ресурс група на Azure

Далі, в межах цієї ресурсної групи, було створено сервер SQL. Сервер SQL забезпечує управління базою даних, яка зберігає всі необхідні дані для роботи додатку. На сервері SQL було розгорнуто SQL Database, яка безпосередньо взаємодіє з додатком, зберігаючи інформацію про користувачів, подорожі та інші дані.

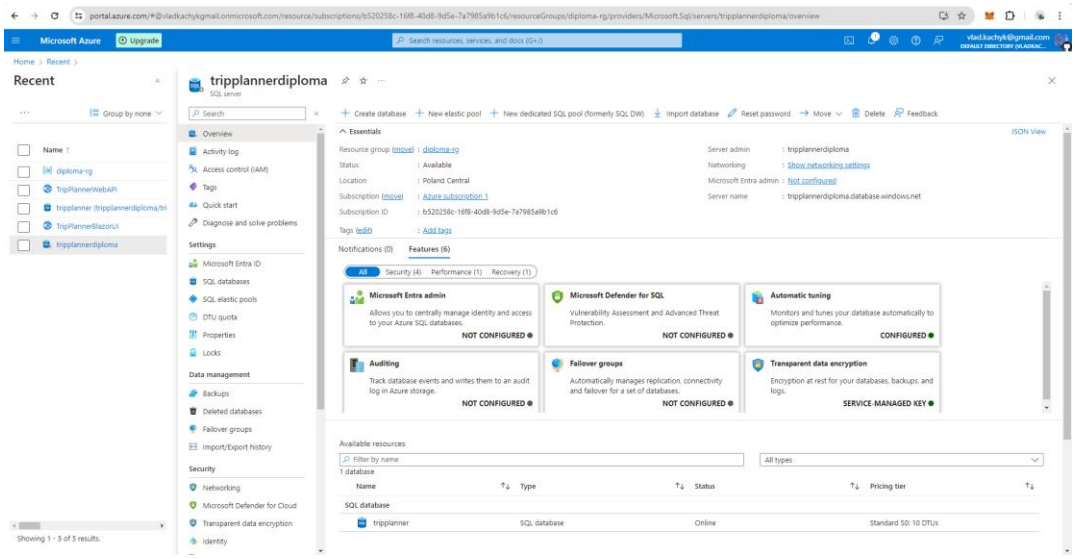


Рисунок 5.2– Створений хмарний SQL Server

Наступним етапом було розгортання API та UI частин додатку. Вони були розгорнуті окремо як Linux Web App для забезпечення високої продуктивності та масштабованості. API забезпечує обробку запитів від користувачів та взаємодію з базою даних, а UI відповідає за інтерфейс користувача та взаємодію з API.

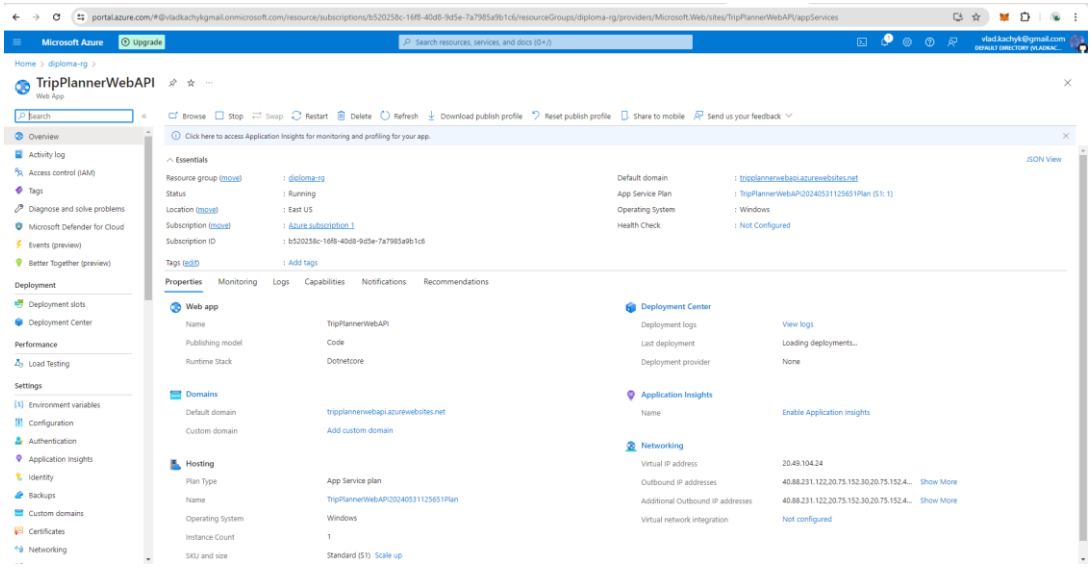


Рисунок 5.3– Розгорнутий .NET Web API модуль в хмарі

Процес розгортання включав налаштування кожного з компонентів, включаючи конфігурацію мережі, безпеки та масштабування. Використання Azure дозволяє автоматизувати багато з цих процесів, забезпечуючи високу надійність та доступність додатку. Було налаштовано моніторинг та логування для відстеження стану додатку та швидкого реагування на можливі проблеми. Azure надає інструменти для моніторингу продуктивності, використання ресурсів та безпеки, що дозволяє підтримувати додаток у належному стані.

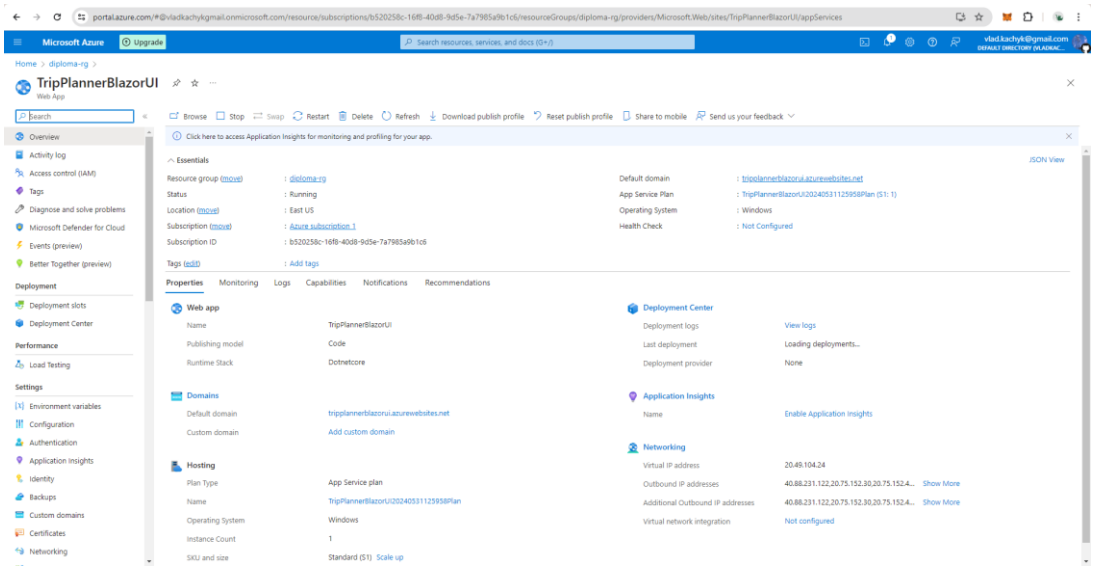


Рисунок 5.4– Розгорнута Blazor UI модуль в хмарі

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Висновки до розділу

У рамках цього розділу було детально описано процес розгортання застосунку на хмарних платформах, таких як Azure. Було створено ресурсну групу та налаштовано різні компоненти, такі як сервер баз даних та веб-додатки, для ефективного розгортання застосунку.

Було розглянуто практичні аспекти супроводу програмного забезпечення, зокрема процес створення та публікації нових версій. Використовуючи GitHub Actions, забезпечено автоматизований процес створення випусків, що дозволяє користувачам отримувати нові версії програмного забезпечення регулярно та безперервно.

ВИСНОВКИ

У результаті виконання дипломного проєкту було спроєктовано і реалізовано веб-додаток для планування автомобільних подорожей. Основною метою проєкту було створення зручного інструменту, який би дозволяв користувачам планувати маршрути подорожей, зберігати їх та отримувати рекомендації щодо зупинок і витрат на пальне. В ході виконання проєкту було досягнуто наступних наукових і практичних результатів:

1. Оцінка одержаних результатів і їх відповідність сучасному рівню наукових і технічних знань:
   - Розроблене програмне забезпечення відповідає сучасним вимогам до веб-додатків, що використовують багатошарову архітектуру та принципи SOLID. Використання патернів Repository і Dependency Injection забезпечує високу гнучкість та тестованість коду.
   - Застосування технологій, таких як .NET 8 Web API, Blazor Server Side UI, Google Maps API, та інтеграція з OpenAI API, свідчить про сучасний підхід до розробки веб-додатків.
2. Ступінь впровадження та можливі галузі або сфери використання результатів роботи:
   - Розроблений додаток може бути використаний туристичними агентствами, транспортними компаніями та приватними користувачами для планування подорожей.
   - Висока масштабованість та модульність системи дозволяє легко адаптувати її для інших типів подорожей, включаючи велосипедні, піші та інші види маршрутів.
3. Наукова, науково-технічна, соціально-економічна значущість роботи:

- Наукова новизна роботи полягає у використанні сучасних технологій та архітектурних підходів для створення зручного та ефективного інструменту планування подорожей.
- Соціально-економічна значущість проєкту полягає в поліпшенні процесу планування подорожей, що може сприяти розвитку туризму, економії часу та коштів для користувачів.

4. Доцільність продовження досліджень за відповідною тематикою:
   - В подальшому дослідження можуть бути спрямовані на розширення функціональності додатку, включаючи інтеграцію з іншими сервісами, такими як бронювання готелів, оренда автомобілів та ін.
   - Можливі також дослідження в напрямку покращення алгоритмів планування маршрутів, використання машинного навчання для персоналізації рекомендацій та прогнозування витрат.

Стан вирішення усіх поставлених у дипломному проєктуванні задач відповідає очікуванням. В якості середовища розробки обрано Visual Studio 2022 та Radzen Blazor Studio, що забезпечило ефективний процес розробки та налагодження коду. У якості бази даних використано SQL Server, що забезпечує надійне зберігання та доступ до даних. Після реалізації застосунок був протестований з використанням Postman для перевірки API-запитів, а також на різних пристроях для забезпечення сумісності та коректного відображення на різних екранах.

Наукова новизна роботи полягає в наступному: вперше реалізовано інтеграцію планування автомобільних подорожей з використанням API OpenAI для динамічної обробки даних та генерації рекомендацій. Це дозволило значно підвищити точність та корисність наданих користувачам рекомендацій.

Таким чином, дипломний проєкт досяг своєї мети, продемонстрував високий рівень відповідності сучасним вимогам та показав потенціал для подальшого розвитку і впровадження у різних сферах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Block, Glenn, et al. Designing Evolvable Web APIs with ASP.NET: Harnessing the Power of the Web. 1st ed., O'Reilly Media, 2014.
2. Sarkar, Tanmoy. Building Modern Serverless Web APIs: Develop Microservices and Implement Serverless Applications with .NET Core 3.1 and AWS Lambda (English Edition). BPB Publications, 2021.
3. Rick-Anderson. "Tutorial: Create a Web API with ASP.NET Core." Microsoft Docs, 3 June 2022, docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio.
4. Normesta. "Azure Blob Storage Documentation." Microsoft Docs, docs.microsoft.com/en-us/azure/storage/blobs. Accessed 10 June 2022.
5. Amundsen, Mike. Design and Build Great Web APIs: Robust, Reliable, and Resilient. 1st ed., Pragmatic Bookshelf, 2020.
6. Gaitatzis, Tony. Learn REST APIs: Your Guide to How to Find, Learn, and Connect to the REST APIs That Powers the Internet of Things Revolution. BackupBrain Press, 2019.
7. January 02, SmartBear 2020. "SOAP vs REST. What's the Difference?" SmartBear.Com, smartbear.com/blog/soap-vs-rest-whats-the-difference. Accessed 10 June 2022.
8. Dependency Injection. (2019, September 23). Tutorials Teacher. https://www.tutorialsteacher.com/ioc/dependency-injection
9. N. (2022, April 13). Designing the infrastructure persistence layer. Microsoft Docs. https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design
10. Richardson, L., Amundsen, M., & Ruby, S. (2013). RESTful Web APIs: Services for a Changing World (1st ed.). O'Reilly Media.
11. Marcano, A. (2021). PROGRAMMING ASP.NET CORE 5 MVC AND WEB API: Examples in C#. Independently published.

12. Geewax, J. J. (2021). API Design Patterns. Manning.
13. H. (2021). MDN Web Docs. Mozilla MDN Web Docs. https://developer.mozilla.org/ru/
14. Jain, Jagdeep. Learn API Testing: Norms, Practices, and Guidelines for Building Effective Test Automation. 1st ed., Apress, 2022.

ДОДАТОК А

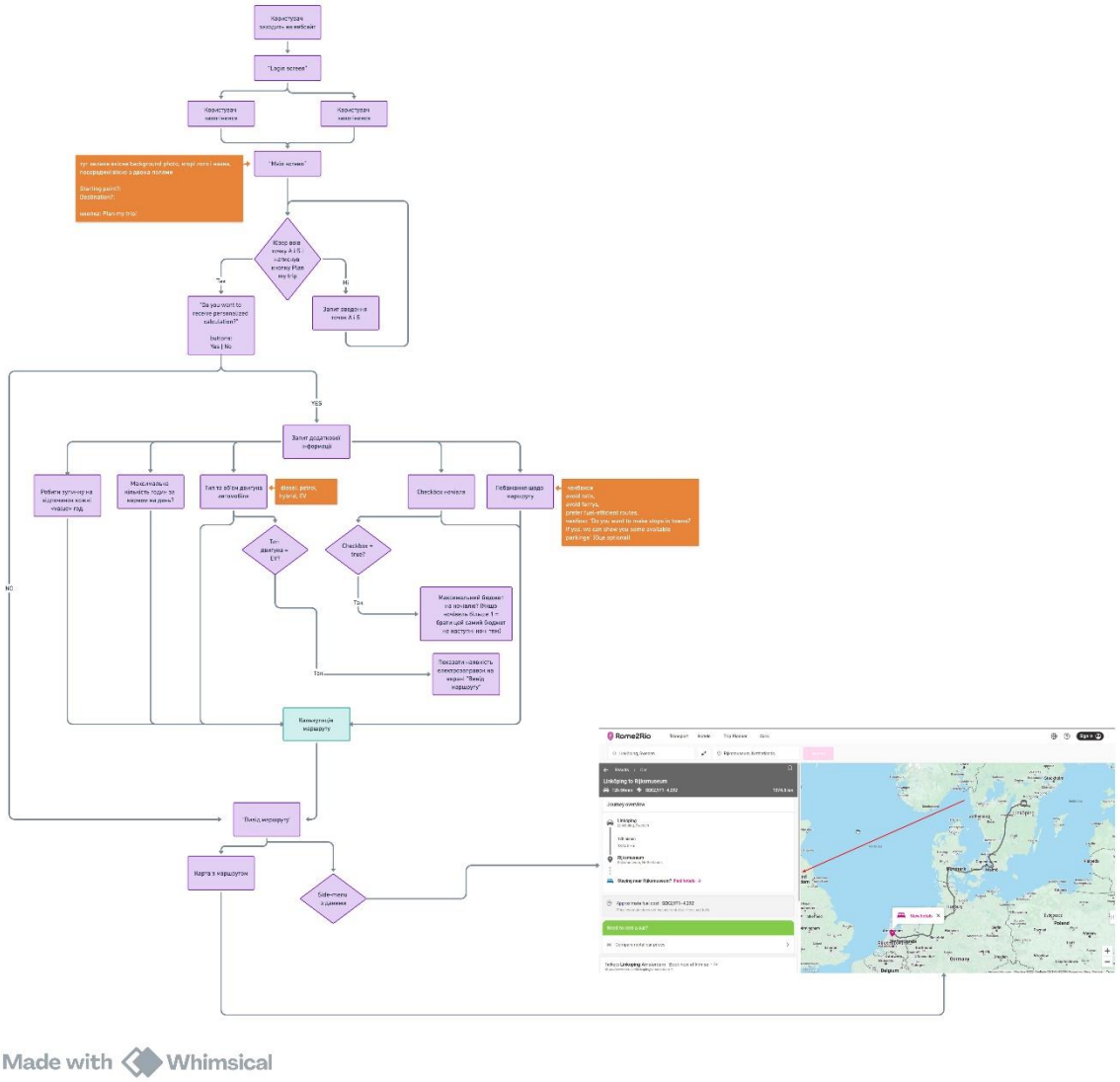


Рисунок 2.1 – Діаграма END-TO-END Flow додатку

ДОДАТОК Б

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016322935

Дата перевірки:
06.06.2024 00:20:12 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 08:39:12 EEST

ID користувача:
76913

Назва документа: **IT-04_Качук_ПЗ**

Кількість сторінок: **61** Кількість слів: **7471** Кількість символів: **62900** Розмір файлу: **3.18 MB** ID файлу: **1016121456**

20.2%
Схожість

Найбільша схожість: **8.12%** з джерелом з Бібліотеки (ID файлу: **1016116296**)

4.78% Джерела з Інтернету	212	Сторінка 63
20% Джерела з Бібліотеки	392	Сторінка 64

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	3

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Програмне забезпечення для організації та планування маршрутів та поїздок

Текст програми

КПІ.ІТ-9406.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Павло РОДІОНОВ

Виконавець:
_____ Владислав КАЧУК

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/shoxtears/TripPlanner

Файл TripPlanner.Application/DTOs/UserDTO.cs

```
namespace TripPlanner.Application.DTOs
{
    public class UserDTO
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл TripPlanner.Application/OpenAI/Config/OpenAICredentials .cs

```
namespace TripPlanner.Application.OpenAI.Config
{
    public static class OpenAICredentials
    {
        public static string APIKEY { get; set; }
    }
}
```

Файл TripPlanner.Application/OpenAI/Services/Interfaces/IOpenAIService.cs

```
namespace TripPlanner.Application.OpenAI.Services.Interfaces
{
    public interface IOpenAIService
    {
        Task<string> SendMessageAsync(string prompt);
    }
}
```

Файл TripPlanner.Application/OpenAI/Services/OpenAIService.cs

```
using Newtonsoft.Json;
using System.Text;
using TripPlanner.Application.OpenAI.Config;
using TripPlanner.Application.OpenAI.Services.Interfaces;

namespace TripPlanner.Application.OpenAI.Services
{
    public class OpenAIService : IOpenAIService
    {
        private readonly HttpClient httpClient;

        public OpenAIService(HttpClient httpClient)
        {
            this.httpClient = httpClient;
        }
```

```
        public async Task<string> SendMessageAsync(string prompt)
        {
            httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer
{OpenAICredentials.APIKEY}");

            var requestBody = new
            {
                model = "gpt-4o",
                messages = new[] { new { role = "user", content = prompt } },
                temperature = 0.7
            };

            var jsonBody = JsonConvert.SerializeObject(requestBody);
            var httpContent = new StringContent(jsonBody, Encoding.UTF8,
"application/json");

            var response = await
httpClient.PostAsync("https://api.openai.com/v1/chat/completions", httpContent);

            var responseBody = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                throw new Exception($"OpenAI API error: {response.StatusCode}");
            }

            dynamic result = JsonConvert.DeserializeObject(responseBody);

            return result.choices[0].message.content.ToString();
        }
    }
}
```

Файл TripPlanner.Application/OpenAI/Services/PromptService.cs

```
namespace TripPlanner.Application.OpenAI.Services
{
    public static class PromptHelper
    {
        public static string GetGasolineCalculationPrompt(string from, string dest)
        {
            return $"Calculate in dollars how much gasoline I will spend on the way
from {from} to {dest}, take the prices that you know, even though they are not
current, car consumption is 10 liters per 100 km. Important! The answer came in the
format of a range in dollars Important! do not write the details of the answer, only
two numbers (range). I need some sample data. Multiply start and stop values by 1.5.
Write only final answer as json object (i will use it in deserealization, don't
write unused symbols) with fields: \"start\", \"stop\"";
        }

        public static string GetStops(string from, string dest, string duration, int
hours)
        {
            return $"I am planning a trip from {from} to {dest}, total duration
{duration}. I want to stop every {hours} hours to rest. Give one city name for each
stop, the cities should be along the route. Please provide your answer in JSON list
of object  (i will use it in deserealization, don't write unused symbols) in type
for example: value: 'Barcelona'; countryCode: 'ES'. if hours == 0 send as response
empty json list";
        }
    }
```

```
}
```

Файл TripPlanner.Application/Services/Intefaces/ISHA256Service.cs

```
namespace TripPlanner.Application.Services.Interfaces
{
    public interface ISHA256Service
    {
        string Hash(string value);
    }
}
```

Файл TripPlanner.Application/Services/Intefaces/IUserManagementService.cs

```
using TripPlanner.Application.DTOs;
using TripPlanner.Infrastructure.Entities;

namespace TripPlanner.Application.Services.Interfaces
{
    public interface IUserManagementService
    {
        Task<int> RegisterAsync(UserDTO userDTO);
        Task<User> GetUserAsync(int id);
        Task<User> GetUserByCredsAsync(string login, string password);
    }
}
```

Файл TripPlanner.Application/Services/SHA256Service.cs

```
using System.Security.Cryptography;
using System.Text;
using TripPlanner.Application.Services.Interfaces;

namespace TripPlanner.Application.Services
{
    public class SHA256Service : ISHA256Service
    {
        public string Hash(string value)
        {
            byte[] passwordBytes = Encoding.UTF8.GetBytes(value);

            using (SHA256 sha256 = SHA256.Create())
            {
                byte[] hashBytes = sha256.ComputeHash(passwordBytes);

                StringBuilder hexString = new StringBuilder(hashBytes.Length * 2);

                foreach (byte b in hashBytes)
                {
                    hexString.AppendFormat("{0:x2}", b);
                }
```

```
                return hexString.ToString();
            }
        }
    }
}
```

Файл TripPlanner.Application/Services/UserManagementService.cs

```
using TripPlanner.Application.DTOs;
using TripPlanner.Application.Services.Interfaces;
using TripPlanner.Infrastructure.Entities;
using TripPlanner.Infrastructure.Repositories.Interfaces;

namespace TripPlanner.Application.Services
{
    public class UserManagementService : IUserManagementService
    {
        private readonly IUserRepository userRepository;
        private readonly ISHA256Service hashService;

        public UserManagementService(IUserRepository userRepository, ISHA256Service
hashService)
        {
            this.userRepository = userRepository;
            this.hashService = hashService;
        }

        public async Task<User> GetUserAsync(int id)
        {
            var foundedUser = await this.userRepository.GetUserAsync(id);

            if (foundedUser is null)
            {
                throw new KeyNotFoundException(nameof(id));
            }

            return foundedUser;
        }

        public async Task<User> GetUserByCredsAsync(string login, string password)
        {
            var foundedUser = await this.userRepository.GetUserByCredsAsync(login,
this.hashService.Hash(password));

            return foundedUser;
        }

        public async Task<int> RegisterAsync(UserDTO userDTO)
        {
            var user = new User()
            {
                Id = 0,
                Credentials = new Credentials()
                {
                    Id = 0,
                    Login = userDTO.Login,
                    Password = this.hashService.Hash(userDTO.Password)
                }
            };

            var userId = await this.userRepository.AddUserAsync(user);
```

```
            return userId;
        }
    }
}
```

Файл TripPlanner.Infrastructure/Entities/Credentials.cs

```
namespace TripPlanner.Infrastructure.Entities
{
    public class Credentials
    {
        public required int Id { get; set; }
        public required string Login { get; set; }
        public required string Password { get; set; }
    }
}
```

Файл TripPlanner.Infrastructure/Entities/SavedTrip.cs

```
namespace TripPlanner.Infrastructure.Entities
{
    public class SavedTrip
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public User User { get; set; }

        public int TripDetailId { get; set; }
        public TripDetail TripDetail { get; set; }
    }
}
```

Файл TripPlanner.Infrastructure/Entities/TripDetail.cs

```
namespace TripPlanner.Infrastructure.Entities
{
    public class TripDetail
    {
        public int Id { get; set; }
        public string From { get; set; }
        public string FromCountryCode { get; set; }
        public string Destination { get; set; }
        public string DestinationCountryCode { get; set; }
        public string Duration { get;set; }
        public string Distance { get; set; }
        public int FuelCostStart { get; set; }
        public int FuelCostStop { get; set; }
    }
}
```

Файл TripPlanner.Infrastructure/Entities/User.cs

```
namespace TripPlanner.Infrastructure.Entities
{
    public class User
    {
        public required int Id { get; set; }
        public required Credentials Credentials { get; set; }
    }
}
```

Файл TripPlanner.Infrastructure/Repositories/Interfaces/IUserRepository.cs

```
using TripPlanner.Infrastructure.Entities;

namespace TripPlanner.Infrastructure.Repositories.Interfaces
{
    public interface IUserRepository
    {
        Task<int> AddUserAsync(User user);
        Task<User> GetUserAsync(int id);
        Task<User> GetUserByCredsAsync(string login, string password);
    }
}
```

Файл TripPlanner.Infrastructure/Repositories/UserRepository.cs

```
using Microsoft.EntityFrameworkCore;
using TripPlanner.Infrastructure.Entities;
using TripPlanner.Infrastructure.Repositories.Interfaces;

namespace TripPlanner.Infrastructure.Repositories
{
    public class UserRepository : IUserRepository
    {
        private readonly AppDatabaseContext context;

        public UserRepository(AppDatabaseContext context)
        {
            this.context = context;
        }
        public async Task<int> AddUserAsync(User user)
        {
            await context.Users.AddAsync(user);

            await context.SaveChangesAsync();

            return user.Id;
        }

        public async Task<User> GetUserAsync(int id)
        {
            return await context.Users.Include(x =>
x.Credentials).AsNoTracking().FirstOrDefaultAsync(x => x.Id == id);
        }

        public async Task<User> GetUserByCredsAsync(string login, string password)
```

```
        {
            return await context.Users.Include(x =>
x.Credentials).AsNoTracking().FirstOrDefaultAsync(x => x.Credentials.Login == login
&& x.Credentials.Password == password);
        }
    }
}
```

Файл TripPlanner.Infrastructure/AppDatabaseContext.cs

```
using Microsoft.EntityFrameworkCore;
using TripPlanner.Infrastructure.Entities;

namespace TripPlanner.Infrastructure
{
    public class AppDatabaseContext : DbContext
    {
        public AppDatabaseContext(DbContextOptions<AppDatabaseContext> options) :
base(options)
        {

        }

        public DbSet<User> Users { get; set; }
        public DbSet<Credentials> Credentials { get; set; }
        public DbSet<SavedTrip> SavedTrips { get; set; }
        public DbSet<TripDetail> TripDetails { get; set; }
    }

}
```

Файл TripPlanner.Infrastructure/DependencyInjection.cs

```
using Microsoft.Extensions.DependencyInjection;
using TripPlanner.Infrastructure.Repositories;
using TripPlanner.Infrastructure.Repositories.Interfaces;

namespace TripPlanner.Infrastructure
{
    public static class DependencyInjection
    {
        public static IServiceCollection RegisterInfrastructure(this
IServiceCollection services)
        {
            services.AddScoped<IUserRepository, UserRepository>();

            return services;
        }
    }
}
```

Файл TripPlanner.WebAPI/Controllers/OpenAIController.cs

```
using Microsoft.AspNetCore.Mvc;
using TripPlanner.Application.OpenAI.Services;
using TripPlanner.Application.OpenAI.Services.Interfaces;

namespace TripPlanner.WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OpenAIController : ControllerBase
    {
        private readonly IOpenAIService openAIService;

        public OpenAIController(IOpenAIService openAIService)
        {
            this.openAIService = openAIService;
        }

        [HttpGet("{from}/{dest}")]
        public async Task<IActionResult> GasolineCalculation(string from, string
dest)
        {
            var prompt = PromptHelper.GetGasolineCalculationPrompt(from, dest);
            return Ok(await openAIService.SendMessageAsync(prompt));
        }


        [HttpGet("{from}/{dest}/{duration}/{hours}")]
        public async Task<IActionResult> GetStops(string from, string dest, string
duration, int hours)
        {
            var prompt = PromptHelper.GetStops(from, dest, duration, hours);
            return Ok(await openAIService.SendMessageAsync(prompt));
        }
    }
}
```

Файл TripPlanner.WebAPI/Controllers/TripController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using TripPlanner.Infrastructure;
using TripPlanner.Infrastructure.Entities;

namespace TripPlanner.WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TripController : ControllerBase
    {
        private readonly AppDatabaseContext appDatabaseContext;

        public TripController(AppDatabaseContext appDatabaseContext)
        {
            this.appDatabaseContext = appDatabaseContext;
        }

        [HttpPost]
        public async Task<IActionResult> Save(SavedTrip trip)
        {
            trip.User = null!;
```

```
            await appDatabaseContext.SavedTrips.AddAsync(trip);
            await appDatabaseContext.SaveChangesAsync();

            return Ok();
        }

        [HttpDelete]
        public async Task<IActionResult> Delete(int savedTripId)
        {
            var tripToDelete = appDatabaseContext.SavedTrips.FirstOrDefault(x =>
x.Id == savedTripId);

            appDatabaseContext.SavedTrips.Remove(tripToDelete);
            await appDatabaseContext.SaveChangesAsync();

            return Ok();
        }

        [HttpDelete("last")]
        public async Task<IActionResult> DeleteLast()
        {
            var tripToDelete = appDatabaseContext.SavedTrips.OrderByDescending(x =>
x.Id).FirstOrDefault();

            appDatabaseContext.SavedTrips.Remove(tripToDelete);
            await appDatabaseContext.SaveChangesAsync();

            return Ok();
        }

        [HttpGet("{userId}")]
        public async Task<IActionResult> Get(string userId)
        {
            return Ok(await appDatabaseContext.SavedTrips.Where(x => x.UserId ==
Convert.ToInt32(userId)).Include(x => x.TripDetail).ToListAsync());
        }
    }
}
```

Файл TripPlanner.WebAPI/Controllers/UserManagementController.cs

```
using Microsoft.AspNetCore.Mvc;
using TripPlanner.Application.DTOs;
using TripPlanner.Application.Services.Interfaces;

namespace TripPlanner.WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UserManagementController : ControllerBase
    {
        private readonly IUserManagementService managementService;

        public UserManagementController(IUserManagementService managementService)
        {
            this.managementService = managementService;
        }

        [HttpPost]
        public async Task<IActionResult> Register(UserDTO userDTO)
        {
            return Ok(await managementService.RegisterAsync(userDTO));
        }
```

```
        [HttpGet("{userId}")]
        public async Task<IActionResult> Get(int userId)
        {
            return Ok(await managementService.GetUserAsync(userId));
        }

        [HttpGet("{login}/{password}")]
        public async Task<IActionResult> GetByCreds(string login, string password)
        {
            return Ok(await managementService.GetUserByCredsAsync(login, password));
        }

    }
}
```

Файл TripPlanner.WebAPI/Middlewares/GlobalErrorHandlingMiddleware.cs

```
using System.Net;
using System.Text.Json;
using TripPlanner.WebAPI.Models;

namespace TripPlanner.WebAPI.Middlewares
{
    public class GlobalErrorHandlingMiddleware
    {
        private readonly RequestDelegate _next;

        public GlobalErrorHandlingMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task Invoke(HttpContext context)
        {
            try
            {
                await _next(context);
            }
            catch (Exception ex)
            {
                await HandleExceptionAsync(context, ex);
            }
        }

        private Task HandleExceptionAsync(HttpContext context, Exception exception)
        {
            context.Response.ContentType = "application/json";

            context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

            var errorResponse = new ErrorResponse
            {
                StatusCode = context.Response.StatusCode,
                Message = "An error occurred while processing your request.",
                ExceptionMessage = exception.Message,
                StackTrace = exception.StackTrace
            };

            var serializedResponse = JsonSerializer.Serialize(errorResponse);

            return context.Response.WriteAsync(serializedResponse);
```

```
        }
    }
}
```

Файл TripPlanner.WebAPI/Models/ErrorResponse.cs

```
namespace TripPlanner.WebAPI.Models
{
    public class ErrorResponse
    {
        public int StatusCode { get; set; }
        public string Message { get; set; }
        public string ExceptionMessage { get; set; }
        public string StackTrace { get; set; }
    }
}
```

Файл TripPlanner.WebAPI/appsettings.json

```
{
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning"
        }
    },
    "AllowedHosts": "*",
    "ConnectionStrings": {
        "SqlServerDatabaseConnectionString": "Data
Source=(localdb)\\MSSQLLocalDB;Initial Catalog=TripPlanner2; Integrated
Security=true;TrustServerCertificate=True;"
        //"SqlServerDatabaseConnectionString":
"Server=tcp:tripplannerdiploma.database.windows.net,1433;Initial
Catalog=tripplanner;Persist Security Info=False;User
ID=tripplannerdiploma;Password=Trip1234;MultipleActiveResultSets=False;Encrypt=True;
TrustServerCertificate=False;Connection Timeout=30;"
    },

    "OpenAI": {
        "APIKEY": "sk-proj-FBQQgTXjIhvYSM1zvD6nT3BlbkFJv7QLsFklQdSMlp4TIqUu"
    }
}
```

Файл TripPlanner.WebAPI /Program.cs

```
using Microsoft.EntityFrameworkCore;
using TripPlanner.Application;
using TripPlanner.Application.OpenAI.Config;
using TripPlanner.Infrastructure;

namespace TripPlanner.WebAPI
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);
```

```
            builder.Services.RegisterApplication();
            builder.Services.RegisterInfrastructure();
            builder.Services.AddDbContext<AppDatabaseContext>(
                db =>

db.UseSqlServer(builder.Configuration.GetConnectionString("SqlServerDatabaseConnecti
onString")));

            OpenAICredentials.APIKEY = builder.Configuration["OpenAI:APIKEY"];

            builder.Services.AddControllers();
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddSwaggerGen();

            builder.Services.AddHttpClient();

            var app = builder.Build();

            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger();
                app.UseSwaggerUI();
            }

            app.UseHttpsRedirection();
            app.UseAuthorization();
            app.MapControllers();

            try
            {
                app.Run();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"An error occurred during application startup:
{ex}");
            }
        }
    }
}
```

Файл TripPlanner.BlazorUI/Constants/Controllers.cs

```
namespace TripPlanner.BlazorUI.Data.Constants
{
    public static class Controllers
    {
        public static string UserManagementController => "UserManagement";
        public static string OpenAIController => "OpenAI";
        public static string TripController => "Trip";
    }
}
```

Файл TripPlanner. BlazorUI/Constants/IntegrationDefinitions.cs

```
namespace TripPlanner.BlazorUI.Data.Constants
{
    public static class IntegrationDefinitions
    {
        //public static string TripPlannerAPIUrl =>
"https://tripplannerwebapi.azurewebsites.net/api/";
        public static string TripPlannerAPIUrl => "https://localhost:7061/api/";
    }
}
```

Файл TripPlanner. BlazorUI/UserDTO.cs

```
namespace TripPlanner.BlazorUI.Data
{
    public class UserDTO
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/DTOs/GasPriceRange.cs

```
namespace TripPlanner.BlazorUI.DTOs
{
    public class GasPriceRange
    {
        public int Start { get; set; }
        public int Stop { get; set; }
    }

    public class StopRecomendation
    {
        public string Value { get; set; }
        public string CountryCode { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/Entities/Credentials.cs

```
namespace TripPlanner.BlazorUI.Entities
{
    public class Credentials
    {
        public required int Id { get; set; }
        public required string Login { get; set; }
        public required string Password { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/Entities/SavedTrip.cs

```
namespace TripPlanner.BlazorUI.Entities
{
    public class SavedTrip
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public User User { get; set; }

        public int TripDetailId { get; set; }
        public TripDetail TripDetail { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/Entities/TripDetail.cs

```
namespace TripPlanner.BlazorUI.Entities
{
    public class TripDetail
    {
        public int Id { get; set; }
        public string From { get; set; }
        public string FromCountryCode { get; set; }
        public string Destination { get; set; }
        public string DestinationCountryCode { get; set; }
        public string Duration { get; set; }
        public string Distance { get; set; }
        public int FuelCostStart { get; set; }
        public int FuelCostStop { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/Entities/User.cs

```
namespace TripPlanner.BlazorUI.Entities
{
    public class User
    {
        public int Id { get; set; }
        public Credentials Credentials { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/Pages/_Host.cshtml

```
@page "/"
```

```
@using Microsoft.AspNetCore.Components.Web
@namespace TripPlanner.BlazorUI.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <base href="~/" />
    <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
    <link href="css/site.css" rel="stylesheet" />
    <link href="TripPlanner.BlazorUI.styles.css" rel="stylesheet" />
    <link rel="icon" type="image/png" href="favicon.png"/>
    <link rel="stylesheet" href="_content/Radzen.Blazor/css/material-base.css">
    <component type="typeof(HeadOutlet)" render-mode="ServerPrerendered" />
</head>
<body>
    <component type="typeof(App)" render-mode="ServerPrerendered" />

    <div id="blazor-error-ui">
        <environment include="Staging,Production">
            An error has occurred. This application may no longer respond until
reloaded.
        </environment>
        <environment include="Development">
            An unhandled exception has occurred. See browser dev tools for details.
        </environment>
        <a href="" class="reload">Reload</a>
        <a class="dismiss">✕</a>
    </div>

    <script src="_framework/blazor.server.js"></script>
    <script src="/js/google-autocomplete.js"></script>
    <script src="_content/Radzen.Blazor/Radzen.Blazor.js"></script>
    <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDCYhGKdvjH4J4ORG41iyB_NSExoGV
ElPc&libraries=places"></script>
    <script>
        function openInNewTab(url) {
            window.open(url, '_blank');
        }
    </script>
</body>
</html>
```

Файл TripPlanner. BlazorUI/Pages/DialogSideContent.razor

```
<h3>Trip Preferences</h3>

            <RadzenCard >
                <RadzenCheckBox @bind-Value="checkboxValues[0]" /> Avoid Ferries <br
/>
                <RadzenCheckBox @bind-Value="checkboxValues[1]" /> Avoid Highways
<br />
                <RadzenCheckBox @bind-Value="checkboxValues[2]" /> Avoid Tolls <br
/>
                <RadzenCheckBox @bind-Value="checkboxValues[3]"/> Route
optimization<br />
                <br />
                <RadzenCard>
        <RadzenText TextStyle="TextStyle.Subtitle2" TagName="TagName.H3">
```

```
            How much hours do you plan on driving per day?
            (Please input this if your trip can't be completed in 1 day)
        </RadzenText>
        <RadzenTextBox @bind-Value=@value MaxLength="5" class="w-100" aria-
label="How much hours do you plan on driving per day?
(Please input this if your trip can't be completed in 1 day)" />
                </RadzenCard><br />
                <RadzenButton Click="SendCheckboxState" class="w-100"
ButtonType="ButtonType.Submit" Icon="done" Style="background:green; margin-
top:1rem;">Submit</RadzenButton>
            </RadzenCard>

@code {
    private bool[] checkboxValues = new bool[4];

    [Parameter]
    public EventCallback<bool[]> OnCheckboxChange { get; set; }

    [Parameter]
    public EventCallback<int> HoursStopsChange{ get; set; }

    [Inject]
    public DialogService DialogService { get; set; } = default!;

    string value;

    private async Task SendCheckboxState()
    {
        await OnCheckboxChange.InvokeAsync(checkboxValues);
        await HoursStopsChange.InvokeAsync(Convert.ToInt32(value));
        DialogService.CloseSide();
    }
}
```

Файл TripPlanner. BlazorUI/Pages/Error.cshtml

```
@page
@model TripPlanner.BlazorUI.Pages.ErrorModel

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1.0, user-scalable=no" />
    <title>Error</title>
    <link href="~/css/bootstrap/bootstrap.min.css" rel="stylesheet" />
    <link href="~/css/site.css" rel="stylesheet" asp-append-version="true" />
</head>

<body>
    <div class="main">
        <div class="content px-4">
            <h1 class="text-danger">Error.</h1>
            <h2 class="text-danger">An error occurred while processing your
request.</h2>

            @if (Model.ShowRequestId)
            {
                <p>
                    <strong>Request ID:</strong> <code>@Model.RequestId</code>
                </p>
```

```
            }

            <h3>Development Mode</h3>
            <p>
                Swapping to the <strong>Development</strong> environment displays
detailed information about the error that occurred.
            </p>
            <p>
                <strong>The Development environment shouldn't be enabled for
deployed applications.</strong>
                It can result in displaying sensitive information from exceptions to
end users.
                For local debugging, enable the <strong>Development</strong>
environment by setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment
variable to <strong>Development</strong>
                and restarting the app.
            </p>
        </div>
    </div>
</body>

</html>
```

Файл TripPlanner. BlazorUI/Pages/Index.razor

```
@page "/"

@using TripPlanner.BlazorUI.DTOs
@using TripPlanner.BlazorUI.Data.Constants;
@using TripPlanner.BlazorUI.Data;
@using TripPlanner.BlazorUI.Entities
@using TripPlanner.BlazorUI.Services;
@using Microsoft.JSInterop;

@if (StateContainer.IsUserLogin)
{
    @if (isYesNoPopUpShown)
    {
        <YesNoPopUp/>
    }

    <RadzenStack Orientation="Orientation.Horizontal" AlignItems="AlignItems.Center"
JustifyContent="JustifyContent.Center" Gap="3rem" Wrap="FlexWrap.Wrap" class="rz-p-
12 border-bottom " Style="padding:1rem !important; margin-bottom:1rem; margin-
top:2rem;">
        <RadzenCard Style="width:350px; background-image: url(' /background.jpg');">
            <RadzenText TextStyle="TextStyle.Subtitle2"
TagName="TagName.H3">From</RadzenText>
            <RadzenTextBox @ref="searchTextbox" id="searchTextBox"
Placeholder="Search..." class="w-100" aria-label="TextBox with placeholder"
Style="height:40px; background-image: url(' /background.jpg');" />
        </RadzenCard>

        <RadzenButton Icon="compare_arrows" ButtonStyle="ButtonStyle.Dark"
Size="ButtonSize.Medium" class="rz-border-radius-10 rz-shadow-8"
Click="SwitchTextBoxes" />

        <RadzenCard Style="width:350px; background-image: url(' /background.jpg');">
            <RadzenText TextStyle="TextStyle.Subtitle2"
TagName="TagName.H3">To</RadzenText>
            <RadzenTextBox @ref="searchTextbox2" id="searchTextBox2"
Placeholder="Search..." class="w-100" aria-label="TextBox with placeholder"
Style="height:40px; background-image: url(' /background.jpg');" />
```

```
        </RadzenCard>

        <RadzenButton style="width: 250px; height:50px; font-weight:700;"
Icon="luggage" Text="PLAN TRIP" IconColor="White" ButtonStyle="ButtonStyle.Success"
Click="OnPlanTripClick"/>
    </RadzenStack>



        <RadzenRow JustifyContent="@(isJourneyOverviewShown ? JustifyContent.Normal
: JustifyContent.Center)" AlignItems="AlignItems.Normal" Style="margin-left:1rem;
margin-right:1rem;">
            @if (isJourneyOverviewShown)
            {
                <RadzenColumn Size="4" class="w-100">
                    @if (!spinner)
                    {
                    <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                                 Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12"
                                 Style="padding:1rem !important; background-
color:gray;">
                        <RadzenStack Orientation="Orientation.Horizontal" Gap="4"
class="w-100">
                            <RadzenText TextStyle="TextStyle.H6" Style="color:white;
font-weight:500;">
                                @($"{fromCity} to {destinationCity}")
                            </RadzenText>
                            <RadzenButton @ref="saveButton" Icon="@saveButtonIcon"
ButtonStyle="ButtonStyle.Success" Size="ButtonSize.ExtraSmall" class="rz-border-
radius-6 rz-shadow-4" Style="margin-left:auto; padding:0;"
                                          Click="SaveTrip" />
                        </RadzenStack>
                        <RadzenStack Orientation="Orientation.Horizontal" Gap="4"
class="w-100">
                            <RadzenIcon Icon="alarm" Style="color:white;" />
                            <RadzenText Style="color:white;">
                                @duration
                            </RadzenText>
                            <RadzenIcon Icon="drive_eta" Style="color:white; margin-
left:15px;" />
                            <RadzenText Style="color:white;">
                                Car trip
                            </RadzenText>
                            <RadzenText style="color:white; margin-left:auto">
                                @distance
                            </RadzenText>
                        </RadzenStack>
                    </RadzenStack>

                    <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                                 Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12
shadow-lg border-5 border-dark"
                                 Style="padding:1rem !important; background-
color:ghostwhite; margin-left:0.3rem; margin-right:0.3rem;">
                        <RadzenStack class="border-bottom w-100">
                            <RadzenText style="font-weight:400">
                                Journey overview
                            </RadzenText>
                        </RadzenStack>

                        <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                            <RadzenIcon Icon="drive_eta" Style="font-size:34px;
color:dimgray" />
```

```
                            <RadzenStack Orientation="Orientation.Vertical"
Style="padding:0rem !important;" Gap="0">
                                <RadzenText Style="padding:0rem !important;">
                                    @fromCity
                                </RadzenText>
                                <RadzenText Style="padding:0rem !important; font-
size:14px; color:dimgray">
                                    @($"{fromCity}, {fromCountry}")
                                </RadzenText>
                            </RadzenStack>
                        </RadzenStack>
                        <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                            <div style="border-left: 5px solid dimgray; height:90px;
margin-left:15px;"></div>
                            <RadzenStack Orientation="Orientation.Vertical" Gap="0">
                                <RadzenText Style="font-size:13px; color:dimgray;
margin-left:10px;">
                                    @duration
                                </RadzenText>
                                <RadzenText Style="font-size:13px; color:dimgray;
margin-left:10px;">
                                    @distance
                                </RadzenText>
                            </RadzenStack>
                        </RadzenStack>
                        <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                            <RadzenIcon Icon="place" Style="font-size:34px;
color:dimgray" />
                            <RadzenStack Orientation="Orientation.Vertical"
Style="padding:0rem !important;" Gap="0">
                                <RadzenText Style="padding:0rem !important;">
                                    @destinationCity
                                </RadzenText>
                                <RadzenText Style="padding:0rem !important; font-
size:14px; color:dimgray">
                                    @($"{destinationCity}, {destinationCountry}")
                                </RadzenText>
                            </RadzenStack>
                        </RadzenStack>
                    </RadzenStack>

                    <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                                 Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12
shadow-lg border-5 border-dark"
                                 Style="padding:1rem !important; background-
color:white; margin-left:0.3rem; margin-right:0.3rem;">

                        <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                            <RadzenIcon Icon="local_gas_station" Style="font-
size:34px; color:gray" />
                            <RadzenStack Orientation="Orientation.Vertical"
Style="padding:0rem !important;" Gap="0">
                                <RadzenText Style="padding:0rem !important; font-
size:14px;">
                                    @($"Approximate fuel cost:
{gasPriceRange.Start}$-{gasPriceRange.Stop}$")
                                </RadzenText>
                                <RadzenText Style="padding:0rem !important; font-
size:12px; color:dimgray">
                                    Price estimate does not include rental car fees
and tolls. Calculation has been provided using OpenAI artificial intellegence.
                                </RadzenText>
```

```
                        </RadzenStack>
                    </RadzenStack>
                </RadzenStack>

                @if(isCityPresents)
                {
                    <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                                 Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12
shadow-lg border-5 border-dark"
                                 Style="padding:1rem !important; background-
color:ghostwhite; margin-left:0.3rem; margin-right:0.3rem;">
                        <RadzenStack class="border-bottom w-100">
                            <RadzenText style="font-weight:400">
                                Your trip is estimated to take longer than 1
day. 🚐
                                Please use our suggestions for overnight stays
during your trip, based on how much hours per day you will be driving.
                            </RadzenText>
                        </RadzenStack>
                        @foreach (var item in citiesToStop)
                        {
                            <RadzenButton class="w-100" Text="@(item.Value)"
Click="@(() => OnNavigateToBooking(item))"></RadzenButton>
                        }
                    </RadzenStack>
                }
                }
                else
                {
                    <RadzenStack AlignItems="AlignItems.Center" Class=""
Gap="2rem" Style="padding:10rem;">
                        <RadzenProgressBarCircular Value="100" ShowValue="false"
Mode="ProgressBarMode.Indeterminate" />
                    </RadzenStack>
                }
            </RadzenColumn>
        }
        <RadzenColumn Size="8" class="rz-background-color-success-lighter"
Style="height:570px">
            <div id="map" style="height:70vh;width:100%;" class="shadow-lg p-3
mb-5 bg-white rounded">
            </div>
        </RadzenColumn>
    </RadzenRow>


}
else
{
    <RadzenStack AlignItems="AlignItems.Center" Class="rz-m-12" Gap="2rem"
Style="padding:20rem;">
        <RadzenProgressBarCircular Value="100" ShowValue="false"
Mode="ProgressBarMode.Indeterminate" />
    </RadzenStack>
}

@code {
    [Inject]
    public StateContainer StateContainer { get; set; } = default!;

    [Inject]
    public IJSRuntime JSRuntime { get; set; } = default!;

    [Inject]
    public HttpService HttpService { get; set; } = default!;
```

```
    [Inject]
    public DialogService DialogService { get; set; } = default!;

    [Inject]
    public NotificationService NotificationService { get; set; } = default!;

    [Inject]
    public NavigationManager NavigationManager { get; set; } = default!;

    [Inject]
    public TooltipService TooltipService { get; set; }

    RadzenButton buttonRef;
    private RadzenTextBox searchTextbox;
    private RadzenTextBox searchTextbox2;

    private RadzenButton saveButton;
    private string saveButtonIcon = "bookmark_border";

    private bool spinner = false;
    private bool isCityPresents = false;

    private string fromCity;
    private string fromCountry;
    private string destinationCity;
    private string destinationCountry;
    private string distance;
    private string duration;
    private int stopsHours;

    private GasPriceRange gasPriceRange;

    private StopRecomendation[] citiesToStop;

    private bool isJourneyOverviewShown = false;
    private bool isYesNoPopUpShown = false;

    private bool avoidFerries = false;
    private bool avoidHighways = false;
    private bool avoidTolls = false;
    private bool optimizeWaypoints = false;

    void ShowTooltip(ElementReference elementReference, TooltipOptions options =
null) => TooltipService.Open(elementReference, "Suggested cities for overnight stays
will be proposed on each <x hours user input> of your trip. If there is no suitable
city after specified time – the search will be adjusted for a stop in a slightly
shorter or longer period of time", options);

    protected override async void OnAfterRender(bool firstRender)
    {
        base.OnAfterRender(false);

        if (firstRender && searchTextbox is not null)
        {
            await JSRuntime.InvokeVoidAsync("initialize", null);
        }

        if (!StateContainer.IsUserLogin)
        {
            Thread.Sleep(1500);
            NavigationManager.NavigateTo("signin");
        }
        else
        {
            StateHasChanged();
```

```
        }
    }

    private async Task SaveTrip()
    {
        if (saveButtonIcon == "bookmark_border")
        {
            var savedTrip = new SavedTrip
                {
                    Id = 0,
                    TripDetail = new TripDetail
                    {
                        Id = 0,
                        Destination = destinationCity,
                        From = fromCity,
                        Distance = distance,
                        Duration = duration,
                        FuelCostStart = gasPriceRange.Start,
                        FuelCostStop = gasPriceRange.Stop,
                        FromCountryCode = fromCountry,
                        DestinationCountryCode = destinationCountry
                    },
                    TripDetailId = 0,
                    UserId = StateContainer.User.Id,
                    User = StateContainer.User
                };

            await HttpService.PostAsync<SavedTrip,
object>(Controllers.TripController,
                savedTrip,
                false
            );

            saveButtonIcon = "bookmark";
        }
        else
        {
            await HttpService.DeleteAsync<object>(Controllers.TripController +
"/last", new object[] { }, false);

            saveButtonIcon = "bookmark_border";
        }
    }

    private void OnCheckboxChangeHandler(bool[] checkboxValues)
    {
        avoidFerries = checkboxValues[0];
        avoidHighways = checkboxValues[1];
        avoidTolls = checkboxValues[2];
        optimizeWaypoints = checkboxValues[3];
    }

    private async Task OnNavigateToBooking(StopRecomendation stopRecomendation)
    {
        await JSRuntime.InvokeVoidAsync("openInNewTab",
$"https://www.booking.com/city/{stopRecomendation.CountryCode}/{stopRecomendation.Va
lue}.en-gb.html?aid=304142&label=gen173nr-
1FCAMotgE47QJIM1gEaOkBiAEBmAEhuAEXyAEM2AEB6AEB-
AECiAIBqAIDuALemd6yBsACAdICJDAxOTQzYTBiLWJiZWQtNGQzYS1iYmE5LTRiNDdjYjIwZmIzYdgCBeACA
Q&sid=301d5d325a1084d8c80f5134683bbf55&keep_landing=1&");
    }

    private void OnHoursStopsChangeHandler(int value)
    {
        stopsHours = value;
    }
```

```
    public void SwitchTextBoxes()
    {
        var temp = searchTextbox.Value;
        searchTextbox.Value = searchTextbox2.Value;
        searchTextbox2.Value = temp;

        StateHasChanged();
    }

    public async void OnPlanTripClick()
    {
        try
        {

            var result = await DialogService.Confirm("Would you like to receive a
custom trip preferences?", "Prefences", new ConfirmOptions() { OkButtonText = "Yes",
CancelButtonText = "No" });

            if (result.Value)
            {
                await DialogService.OpenSideAsync<DialogSideContent>("Preferences
panel", options: new SideDialogOptions { CloseDialogOnOverlayClick = false, Position
= DialogPosition.Right, ShowMask = true }, parameters: new Dictionary<string,
object>
                {
                    { "OnCheckboxChange", EventCallback.Factory.Create<bool[]>(this,
OnCheckboxChangeHandler) },
                    { "HoursStopsChange", EventCallback.Factory.Create<int>(this,
OnHoursStopsChangeHandler)}
                });
            }
            else
            {
                OnCheckboxChangeHandler(new bool[] { false, false, false, false });
                OnHoursStopsChangeHandler(0);
            }

            isJourneyOverviewShown = true;
            spinner = true;
            isCityPresents = false;

            var tripDetails = await JSRuntime.InvokeAsync<string[]>("planTripClick",
avoidFerries, avoidHighways, avoidTolls, optimizeWaypoints);
            var additionalDataAboutMarkers = await
JSRuntime.InvokeAsync<string[]>("getCityAndCountry", null);

            // data mapping
            fromCity = additionalDataAboutMarkers[0];
            fromCountry = additionalDataAboutMarkers[1];
            destinationCity = additionalDataAboutMarkers[2];
            destinationCountry = additionalDataAboutMarkers[3];

            distance = tripDetails[0];
            duration = tripDetails[1];

            gasPriceRange = await
HttpService.GetAsync<GasPriceRange>(Controllers.OpenAIController, new object[] {
fromCity, destinationCity });

            citiesToStop = await
HttpService.GetAsync<StopRecomendation[]>(Controllers.OpenAIController, new object[]
{ fromCity, destinationCity, duration, stopsHours });

            saveButtonIcon = "bookmark_border";
```

```
            spinner = false;
            isCityPresents = citiesToStop.Any();
        }
        catch
        {
        }
    }
}
```

Файл TripPlanner. BlazorUI/Pages/SignInPage.razor

```
@page "/signin"
@using TripPlanner.BlazorUI.Data.Constants;
@using TripPlanner.BlazorUI.Entities;
@using TripPlanner.BlazorUI.Services;

<div id="signin-form">
    <RadzenRow Gap="0" Class="rz-my-12 rz-mx-auto rz-border-radius-6 rz-shadow-10"
Style="width: 100%; max-width: 1000px; overflow: hidden;">
        <RadzenColumn Size="12" SizeMD="6">
            <RadzenCard Class="rz-shadow-0 rz-border-radius-0 rz-text-align-center
rz-p-12" Style="height: 100%; background: var(--rz-primary-light) no-repeat 100% 75%
fixed
url('
SIwIDAgMTIwNCAxNDU4IiBmaWxsPSJub25lIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciP
go8ZyBvcGFjaXR5PSIwLjUiIGZpbHRlcj0idXJsKCNmaWx0ZXIwX2ZfNDkzXzEwMTM0KSI+CjxjaXJjbGUgY
3g9IjcyMi4xMjgiIGN5PSI4MzkuMDIiIHI9IjQ4MS40MTkiIGZpbGw9InVybCgjcGFpbnQwX3JhZGlhbF80O
TNfMTAxMzQpIi8+CjwvZz4KPGcgb3BhY2l0eT0iMC41IiBmaWx0ZXI9InVybCgjZmlsdGVyMV9mXzQ5M18xM
DEzNCkiPgo8Y2lyY2xlIGN4PSI0NzAuMzMzIiBjeT0iNTcwLjMzMyIgcj0iNDcwLjMzMyIgZmlsbD0idXJsK
CNwYWludDFfcmFkaWFsXzQ5M18xMDEzNCkiLz4KPC9nPgo8ZyBvcGFjaXR5PSIwLjUiIGZpbHRlcj0idXJsK
CNmaWx0ZXIyX2ZfNDkzXzEwMTM0KSI+CjxjaXJjbGUgY3g9IjY5MS41MTEiIGN5PSI1MjIuMjk3IiByPSIzM
zEuNTAzIiBmaWxsPSJ1cmwoI3BhaW50Ml9yYWRpYWxfNDkzXzEwMTM0KSIvPgo8L2c+CjxnIG9wYWNpdHk9I
jAuNSIgZmlsdGVyPSJ1cmwoI2ZpbHRlcjNfZl80OTNfMTAxMzQpIj4KPGNpcmNsZSBjeD0iNjA4LjI0NCIgY
3k9IjEwNzkuOTciIHI9IjMzMS41MDMiIHRyYW5zZm9ybT0icm90YXRlKC04MS4yMjQ0IDYwOC4yNDQgMTA3O
S45NykiIGZpbGw9InVybCgjcGFpbnQzX3JhZGlhbF80OTNfMTAxMzQpIi8+CjxnIG9wYWNpdHk9IjAuNSI+C
XIgaWQ9ImZpbHRlcjBfZl80OTNfMTAxMzQiIHg9IjE0MC43MDkiIHk9IjI1Ny42MDEiIHdpZHRoPSIxMTYyL
jg0IiBoZWlnaHQ9IjExNjIuODQiIGZpbHRlclVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgY29sb3ItaW50ZXJwb
2xhdGlvbi1maWx0ZXJzPSJzUkdCIj4KPGZlRmxvb2QgZmxvb2Qtb3BhY2l0eT0iMCIgcmVzdWx0PSJCYWNrZ
3JvdW5kSW1hZ2VGaXgiLz4KPGZlQmxlbmQgbW9kZT0ibm9ybWFsIiBpbj0iU291cmNlR3JhcGhpYyIgaW4yP
SJCYWNrZ3JvdW5kSW1hZ2VGaXgiIHJlc3VsdD0ic2hhcGUiLz4KPGZlR2F1c3NpYW5CbHVyIHN0ZERldmlhd
Glvbj0iNTAiIHJlc3VsdD0iZWZmZWN0MV9mb3JlZ3JvdW5kQmx1cl80OTNfMTAxMzQiLz4KPC9maWx0ZXI+C
jxmaWx0ZXIgaWQ9ImZpbHRlcjFfZl80OTNfMTAxMzQiIHg9IjEwLjY3I
iBoZWlnaHQ9IjExNDAuNjciIGZpbHRlclVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgY29sb3ItaW50ZXJwb2xhd
Glvbi1maWx0ZXJzPSJzUkdCIj4KPGZlRmxvb2QgZmxvb2Qtb3BhY2l0eT0iMCIgcmVzdWx0PSJCYWNrZ3Jvd
W5kSW1hZ2VGaXgiLz4KPGZlQmxlbmQgbW9kZT0ibm9ybWFsIiBpbj0iU291cmNlR3JhcGhpYyIgaW4yPSJCY
WNrZ3JvdW5kSW1hZ2VGaXgiIHJlc3VsdD0ic2hhcGUiLz4KPGZlR2F1c3NpYW5CbHVyIHN0ZERldmlhdGlvb
j0iNTAiIHJlc3VsdD0iZWZmZWN0MV9mb3JlZ3JvdW5kQmx1cl80OTNfMTAxMzQiLz4KPC9maWx0ZXI+Cjxma
Wx0ZXIgaWQ9ImZpbHRlcjJfZl80OTNfMTAxMzQiIHg9IjI2MC4wMDgiIHk9IjkwLjc5MzkiIHdpZHRoPSI4N
jMuMDA2IiBoZWlnaHQ9Ijg2My4wMDYiIGZpbHRlclVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgY29sb3ItaW50Z
XJwb2xhdGlvbi1maWx0ZXJzPSJzUkdCIj4KPGZlRmxvb2QgZmxvb2Qtb3BhY2l0eT0iMCIgcmVzdWx0PSJCY
WNrZ3JvdW5kSW1hZ2VGaXgiLz4KPGZlQmxlbmQgbW9kZT0ibm9ybWFsIiBpbj0iU291cmNlR3JhcGhpYyIga
W4yPSJCYWNrZ3JvdW5kSW1hZ2VGaXgiIHJlc3VsdD0ic2hhcGUiLz4KPGZlR2F1c3NpYW5CbHVyIHN0ZERld
mlhdGlvbj0iNTAiIHJlc3VsdD0iZWZmZWN0MV9mb3JlZ3JvdW5kQmx1cl80OTNfMTAxMzQiLz4KPC9maWx0Z
XI+CjxmaWx0ZXIgaWQ9ImZpbHRlcjNfZl80OTNfMTAxMzQiIHg9IjE3Ni42OTQiIHk9IjY0OC40MjMiIHdpZ
HRoPSI4NjMuMSIgaGVpZ2h0PSI4NjMuMSIgZmlsdGVyVW5pdHM9InVzZXJTcGFjZU9uVXNlIiBjb2xvci1pb
nRlcnBvbGF0aW9uLWZpbHRlcnM9InNSR0IiPgo8ZmVGbG9vZCBmbG9vZC1vcGFjaXR5PSIwIiByZXN1bHQ9I
kJhY2tncm91bmRJbWFnZUZpeCIvPgo8ZmVCbGVuZCBtb2RlPSJub3JtYWwiIGluPSJTb3VyY2VHcmFwaGljI
iBpbjI9IkJhY2tncm91bmRJbWFnZUZpeCIgcmVzdWx0PSJzaGFwZSIvPgo8ZmVHYXVzc2lhbkJsdXIgc3RkR
GV2aWF0aW9uPSI1MCIgcmVzdWx0PSJlZmZlY3QxX2ZvcmVncm91bmRCbHVyXzQ5M18xMDEzNCIvPgo8L2Zpb
HRlcj4KPHJhZGlhbEdyYWRpZW50IGlkPSJwYWludDBfcmFkaWFsXzQ5M18xMDEzNCIgY3g9IjIyAiIGN5PSIwI
iByPSIxIiBncmFkaWVudFVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgZ3JhZGllbnRUcmFuc2Zvcm09InRyYW5zb
GF0ZSg3MjIuMTI4IDgzOS4wMikgcm90YXRlKDkwKSBzY2FsZSg0ODEuNDE5KSI+CjxzdG9wIHN0b3AtY29sb
3I9IiNGRjFBNkMiLz4KPHN0b3Agb2Zmc2V0PSIxIiBzdG9wLWNvbG9yPSIjRkYxQTZDIiBzdG9wLW9wYWNpd

```
            spinner = false;
            isCityPresents = citiesToStop.Any();
        }
        catch
        {
        }
    }
}
```

Файл TripPlanner. BlazorUI/Pages/SignInPage.razor

```
@page "/signin"
@using TripPlanner.BlazorUI.Data.Constants;
@using TripPlanner.BlazorUI.Entities;
@using TripPlanner.BlazorUI.Services;

    <RadzenRow Gap="0" Class="rz-my-12 rz-mx-auto rz-border-radius-6 rz-shadow-10"
Style="width: 100%; max-width: 1000px; overflow: hidden;">
        <RadzenColumn Size="12" SizeMD="6">
            <RadzenCard Class="rz-shadow-0 rz-border-radius-0 rz-text-align-center
rz-p-12" Style="height: 100%; background: var(--rz-primary-light) no-repeat 100% 75%
fixed
url('
SIwIDAgMTIwNCAxNDU4IiBmaWxsPSJub25lIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciP
go8ZyBvcGFjaXR5PSIwLjUiIGZpbHRlcj0idXJsKCNmaWx0ZXIwX2ZfNDkzXzEwMTM0KSI+CjxjaXJjbGUgY
3g9IjcyMi4xMjgiIGN5PSI4MzkuMDIiIHI9IjQ4MS40MTkiIGZpbGw9InVybCgjcGFpbnQwX3JhZGlhbF80O
TNfMTAxMzQpIi8+CjwvZz4KPGcgb3BhY2l0eT0iMC41IiBmaWx0ZXI9InVybCgjZmlsdGVyMV9mXzQ5M18xM
DEzNCkiPgo8Y2lyY2xlIGN4PSI0NzAuMzMzIiBjeT0iNTcwLjMzMyIgcj0iNDcwLjMzMyIgZmlsbD0idXJsK
CNwYWludDFfcmFkaWFsXzQ5M18xMDEzNCkiLz4KPC9nPgo8ZyBvcGFjaXR5PSIwLjUiIGZpbHRlcj0idXJsK
CNmaWx0ZXIyX2ZfNDkzXzEwMTM0KSI+CjxjaXJjbGUgY3g9IjY5MS41MTEiIGN5PSI1MjIuMjk3IiByPSIzM
zEuNTAzIiBmaWxsPSJ1cmwoI3BhaW50Ml9yYWRpYWxfNDkzXzEwMTM0KSIvPgo8L2c+CjxnIG9wYWNpdHk9I
jAuNSIgZmlsdGVyPSJ1cmwoI2ZpbHRlcjNfZl80OTNfMTAxMzQpIj4KPGNpcmNsZSBjeD0iNjA4LjI0NCIgY
3k9IjEwNzkuOTciIHI9IjMzMS41MDMiIHRyYW5zZm9ybT0icm90YXRlKC04MS4yMjQ0IDYwOC4yNDQgMTA3O
S45NykiIGZpbGw9InVybCgjcGFpbnQzX3JhZGlhbF80OTNfMTAxMzQpIi8+CjxnIG9wYWNpdHk9IjAuNSI+C
XIgaWQ9ImZpbHRlcjBfZl80OTNfMTAxMzQiIHg9IjE0MC43MDkiIHk9IjI1Ny42MDEiIHdpZHRoPSIxMTYyL
jg0IiBoZWlnaHQ9IjExNjIuODQiIGZpbHRlclVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgY29sb3ItaW50ZXJwb
2xhdGlvbi1maWx0ZXJzPSJzUkdCIj4KPGZlRmxvb2QgZmxvb2Qtb3BhY2l0eT0iMCIgcmVzdWx0PSJCYWNrZ
3JvdW5kSW1hZ2VGaXgiLz4KPGZlQmxlbmQgbW9kZT0ibm9ybWFsIiBpbj0iU291cmNlR3JhcGhpYyIgaW4yP
SJCYWNrZ3JvdW5kSW1hZ2VGaXgiIHJlc3VsdD0ic2hhcGUiLz4KPGZlR2F1c3NpYW5CbHVyIHN0ZERldmlhd
Glvbj0iNTAiIHJlc3VsdD0iZWZmZWN0MV9mb3JlZ3JvdW5kQmx1cl80OTNfMTAxMzQiLz4KPC9maWx0ZXI+C
jxmaWx0ZXIgaWQ9ImZpbHRlcjFfZl80OTNfMTAxMzQiIHg9IjEwLjY3I
iBoZWlnaHQ9IjExNDAuNjciIGZpbHRlclVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgY29sb3ItaW50ZXJwb2xhd
Glvbi1maWx0ZXJzPSJzUkdCIj4KPGZlRmxvb2QgZmxvb2Qtb3BhY2l0eT0iMCIgcmVzdWx0PSJCYWNrZ3Jvd
W5kSW1hZ2VGaXgiLz4KPGZlQmxlbmQgbW9kZT0ibm9ybWFsIiBpbj0iU291cmNlR3JhcGhpYyIga
W4yPSJCYWNrZ3JvdW5kSW1hZ2VGaXgiIHJlc3VsdD0ic2hhcGUiLz4KPGZlR2F1c3NpYW5CbHVyIHN0ZERld
mlhdGlvbj0iNTAiIHJlc3VsdD0iZWZmZWN0MV9mb3JlZ3JvdW5kQmx1cl80OTNfMTAxMzQiLz4KPC9maWx0Z
XI+CjxmaWx0ZXIgaWQ9ImZpbHRlcjNfZl80OTNfMTAxMzQiIHg9IjE3Ni42OTQiIHk9IjY0OC40MjMiIHdpZ
HRoPSI4NjMuMSIgaGVpZ2h0PSI4NjMuMSIgZmlsdGVyVW5pdHM9InVzZXJTcGFjZU9uVXNlIiBjb2xvci1pb
nRlcnBvbGF0aW9uLWZpbHRlcnM9InNSR0IiPgo8ZmVGbG9vZCBmbG9vZC1vcGFjaXR5PSIwIiByZXN1bHQ9I
kJhY2tncm91bmRJbWFnZUZpeCIvPgo8ZmVCbGVuZCBtb2RlPSJub3JtYWwiIGluPSJTb3VyY2VHcmFwaGljI
iBpbjI9IkJhY2tncm91bmRJbWFnZUZpeCIgcmVzdWx0PSJzaGFwZSIvPgo8ZmVHYXVzc2lhbkJsdXIgc3RkR
GV2aWF0aW9uPSI1MCIgcmVzdWx0PSJlZmZlY3QxX2ZvcmVncm91bmRCbHVyXzQ5M18xMDEzNCIvPgo8L2Zpb
HRlcj4KPHJhZGlhbEdyYWRpZW50IGlkPSJwYWludDBfcmFkaWFsXzQ5M18xMDEzNCIgY3g9IjIyAiIGN5PSIwI
iByPSIxIiBncmFkaWVudFVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgZ3JhZGllbnRUcmFuc2Zvcm09InRyYW5zb
GF0ZSg3MjIuMTI4IDgzOS4wMikgcm90YXRlKDkwKSBzY2FsZSg0ODEuNDE5KSI+CjxzdG9wIHN0b3AtY29sb
3I9IiNGRjFBNkMiLz4KPHN0b3Agb2Zmc2V0PSIxIiBzdG9wLWNvbG9yPSIjRkYxQTZDIiBzdG9wLW9wYWNpd
```
```

```
Hk9IjAiLz4KPC9yYWRpYWxHcmFkaWVudD4KPHJhZGlhbEdyYWRpZW50IGlkPSJwYWludDFfcmFkaWFsXzQ5M
18xMDEzNCIgY3g9IjAiIGN5PSIwIiByPSIxIiBncmFkaWVudFVuaXRzPSJ1c2VyU3BhY2VPblVzZSIgZ3JhZ
GllbnRUcmFuc2Zvcm09InRyYW5zbGF0ZSg0NzAuMzMzIDU3MC4zMzMpIHJvdGF0ZSg5MCkgc2NhbGUoNDcwL
jMzMykiPgo8c3RvcCBzdG9wLWNvbG9yPSIjM0FBQ0ZGIi8+CjxzdG9wIG9mZnNldD0iMSIgc3RvcC1jb2xvc
j0iIzNBOTVGRiIgc3RvcC1vcGFjaXR5PSIwIi8+CjwvcmFkaWFsR3JhZGllbnQ+CjxyZWN0IGlkPSJSZWN0Y
CBpZD0icGFpbnQyX3JhZGlhbF80OTNfMTAxMzQiIGN4PSIwIiBjeT0iMCIgcj0iMSIgZ3JhZGllbnRVbml0c
z0idXNlclNwYWNlT25Vc2UiIGdyYWRpZW50VHJhbnNmb3JtPSJ0cmFuc2xhdGUoNjkxLjUxMSA1MjIuMjk3K
SByb3RhdGUoOTApIHNjYWxlKDMzMS41MDMpIj4KPHN0b3Agc3RvcC1jb2xvcj0iIzQ4M0FGRiIvPgo8c3RvcC
CBvZmZzZXQ9IjEiIHN0b3AtY29sb3I9IiM0ODNBRkYiIHN0b3Atb3BhY2l0eT0iMCIvPgo8L3JhZGlhbEdyY
WRpZW50Pgo8cmFkaWFsR3JhZGllbnQgaWQ9InBhaW50M19yYWRpYWxfNDkzXzEwMTM0IiBjeD0iMCIgY3k9I
jAiIHI9IjEiIGdyYWRpZW50VW5pdHM9InVzZXJTcGFjZU9uVXNlIiBncmFkaWVudFRyYW5zZm9ybT0idHJhb
nNsYXRlKDYwOC4yNDQgMTA3OS45Nykgcm90YXRlKDkwKSBzY2FsZSgzMzEuNTAzKSI+CjxzdG9wIHN0b3AtY
29sb3I9IiNGRkM4M0EiLz4KPHN0b3Agb2Zmc2V0PSIxIiBzdG9wLWNvbG9yPSIjRkZDODNBIiBzdG9wLW9wY
WNpdHk9IjAiLz4KPC9yYWRpYWxHcmFkaWVudD4KPC9kZWZzPgo8L3N2Zz4K')">
                <RadzenText TextStyle="TextStyle.DisplayH3" TagName="TagName.H2"
Class="rz-color-white rz-mb-6">Welcome!</RadzenText>
                <RadzenText TextStyle="TextStyle.H6" Class="rz-color-white">Join to
TripPlanner community!</RadzenText>
                <RadzenText TextStyle="TextStyle.Body2" Class="rz-color-white">Plan
your trips with <strong>Trip Planner </strong>. Advanced routes, fuel calculation
and much more. Elevate your trip experience now.</RadzenText>
            </RadzenCard>
        </RadzenColumn>
        <RadzenColumn Size="12" SizeMD="6">
            <RadzenCard Class="rz-shadow-0 rz-border-radius-0 rz-p-12">
                <RadzenText TextStyle="TextStyle.H5" TagName="TagName.H2" Class="rz-
mb-6">
                    Login
                </RadzenText>
                <RadzenTemplateForm Data=@("LoginWithRegister")>
                    <RadzenLogin AllowRegister="true"
                                 AllowResetPassword="false"
                                 Username=@userName
                                 Password=@password
                                 Login=@((args) => OnLogin(args))
                                 Register="@(() => OnRegister())" />
                </RadzenTemplateForm>
            </RadzenCard>
        </RadzenColumn>
    </RadzenRow>
</div>


@code {
    [Inject]
    public StateContainer StateContainer { get; set; }

    [Inject]
    public NavigationManager NavigationManager { get; set; }

    [Inject]
    public HttpService HttpService { get; set; }

    [Inject]
    public NotificationService NotificationService { get; set; }

    private bool sidebarExpanded = true;

    private string userName = default!;
    private string password = default!;

    private async Task OnLogin(LoginArgs args)
    {
        try
        {
```

```
            var user = await
HttpService.GetAsync<User>(Controllers.UserManagementController, new object[] {
args.Username, args.Password });
            StateContainer.User = user;
            StateContainer.IsUserLogin = true;

            ShowNotification(new NotificationMessage { Severity =
NotificationSeverity.Success, Summary = "Success!", Detail = $"Successfully signed
in!\nWelcome {args.Username}!", Duration = 4000 });
            NavigationManager.NavigateTo("");
        }
        catch
        {
            StateContainer.IsUserLogin = false;

            ShowNotification(new NotificationMessage { Severity =
NotificationSeverity.Error, Summary = "Sign In failed!", Detail = "Incorrect
username/password pair!", Duration = 4000 });
        }
    }

    private void ShowNotification(NotificationMessage message)
    {
        NotificationService.Notify(message);
    }

    private void OnRegister()
    {
        NavigationManager.NavigateTo("signup");
    }
}
```

Файл TripPlanner. BlazorUI/Pages/SignUpPage.razor

```
@page "/signup"
@using TripPlanner.BlazorUI.Data.Constants;
@using TripPlanner.BlazorUI.Data;
@using TripPlanner.BlazorUI.Entities;
@using TripPlanner.BlazorUI.Services;

<div id="signin-form">
    <RadzenStack Gap="0" Class="rz-my-12 rz-mx-auto rz-border-radius-6 rz-shadow-10"
Style="width: 100%; max-width: 400px; overflow: hidden;">
        <RadzenCard Class="rz-shadow-0 rz-border-radius-0 rz-p-12" style="text-
align: center; background-color: #4340d2;">
            <RadzenText TextStyle="TextStyle.DisplayH3" TagName="TagName.H2"
Class="rz-color-white rz-mb-0">Sign Up</RadzenText>
        </RadzenCard>
        <RadzenCard Class="rz-shadow-0 rz-p-12">
            <RadzenTemplateForm Data=@("SimpleLogin")>
                <RadzenLogin Username=@userName
                             Password=@password
                             LoginText="Register"
                             Login=@(args => OnSignUp(args))
                             AllowRegister="false"
                             AllowResetPassword="false" />
            </RadzenTemplateForm>
        </RadzenCard>
    </RadzenStack>
</div>
```

```
@code {
    [Inject]
    public StateContainer StateContainer { get; set; }

    [Inject]
    public NavigationManager NavigationManager { get; set; }

    [Inject]
    public NotificationService NotificationService { get; set; }

    [Inject]
    public HttpService HttpService { get; set; }

    private string userName = default!;
    private string password = default!;

    bool sidebarExpanded = true;

    private async Task OnSignUp(LoginArgs args)
    {
        var userId = await HttpService.PostAsync<UserDTO,
int>(Controllers.UserManagementController,
            new UserDTO
                {
                    Login = args.Username,
                    Password = args.Password
                }
        );

        var user = await
HttpService.GetAsync<User>(Controllers.UserManagementController, new object[] {
userId });

        StateContainer.User = user;
        StateContainer.IsUserLogin = true;

        ShowNotification(new NotificationMessage { Severity =
NotificationSeverity.Success, Summary = "Success!", Detail = "Welcome! You've been
successfully registreted!", Duration = 4000 });

        NavigationManager.NavigateTo("");
    }

    private void ShowNotification(NotificationMessage message)
    {
        NotificationService.Notify(message);
    }


}
```

Файл TripPlanner. BlazorUI/Pages/Trips.razor

```
@page "/savedtrips"
@using TripPlanner.BlazorUI.DTOs
@using TripPlanner.BlazorUI.Entities
@using TripPlanner.BlazorUI.Services

<div class="card-container">
    @foreach (var trip in StateContainer.SavedTrips)
    {
```

```
        <RadzenCard Variant="@variant" Class="rz-my-12 rz-mx-auto card"
Style="width: 400px; min-width: 400px; max-width: 400px;" @onclick="@(() =>
ShowInlineDialog(trip))">
            <RadzenStack Orientation="Orientation.Horizontal"
JustifyContent="JustifyContent.Start" Gap="1rem" Class="rz-p-4">
                <RadzenImage Path="images/trip.jpg" Style="width: 100px; height:
100px; border-radius: 50%;" />
                <RadzenStack Gap="0">
                    <RadzenText TextStyle="TextStyle.Overline" class="rz-display-
flex rz-mt-2 rz-my-0">From</RadzenText>
                    <RadzenText
TextStyle="TextStyle.Body1"><b>@(trip.TripDetail.From)</b></RadzenText>
                    <RadzenText TextStyle="TextStyle.Overline" class="rz-display-
flex rz-mt-4 rz-mb-0">To</RadzenText>
                    <RadzenText
TextStyle="TextStyle.Body1"><b>@trip.TripDetail.Destination</b></RadzenText>
                </RadzenStack>
            </RadzenStack>
        </RadzenCard>
    }
</div>

<style>
    .card-container {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(420px, 1fr));
        gap: 1.5rem;
        justify-content: center;
    }

    .card {
        max-width: 100%;
        margin: 0 auto;
        transition: transform 0.3s ease-in-out;
    }

        .card:hover {
            transform: scale(1.05);
        }

    .rz-my-12 {
        margin-top: 1rem !important;
        margin-bottom: 1rem !important;
    }

    .rz-mx-auto {
        margin-left: auto !important;
        margin-right: auto !important;
    }
</style>


<style>
    .rz-row > .rz-col-4 {
        max-width: none !important;
        flex-basis: auto !important;
    }
</style>

@code{
    Variant variant = Variant.Filled;


    [Inject]
    public DialogService DialogService { get; set; } = default!;
```

```
    [Inject]
    public StateContainer StateContainer{get;set;}

    private GasPriceRange gasPriceRange;
    private string fromCountry = "UA";
    private string destinationCountry = "UA";

    async Task ShowInlineDialog(SavedTrip trip)
    {
        var result = await DialogService.OpenAsync("Simple Dialog", ds =>
    @<RadzenRow JustifyContent="JustifyContent.Center"
AlignItems="AlignItems.Normal" Style="margin-left:1rem; margin-right:1rem;">
        <RadzenColumn Size="4" class="w-100">
            <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                         Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12"
                         Style="padding:1rem !important; background-color:gray;">
                <RadzenStack Orientation="Orientation.Horizontal" Gap="4" class="w-
100">
                    <RadzenText TextStyle="TextStyle.H6" Style="color:white; font-
weight:500;">
                        @($"{trip.TripDetail.From} to
{trip.TripDetail.Destination}")
                    </RadzenText>
                </RadzenStack>
                <RadzenStack Orientation="Orientation.Horizontal" Gap="4" class="w-
100">
                    <RadzenIcon Icon="alarm" Style="color:white;" />
                    <RadzenText Style="color:white;">
                        @trip.TripDetail.Duration
                    </RadzenText>
                    <RadzenIcon Icon="drive_eta" Style="color:white; margin-
left:15px;" />
                    <RadzenText Style="color:white;">
                        Car trip
                    </RadzenText>
                    <RadzenText style="color:white; margin-left:auto">
                        @trip.TripDetail.Distance
                    </RadzenText>
                </RadzenStack>
            </RadzenStack>

            <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                         Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12 shadow-lg
border-5 border-dark"
                         Style="padding:1rem !important; background-
color:ghostwhite; margin-left:0.3rem; margin-right:0.3rem;">
                <RadzenStack class="border-bottom w-100">
                    <RadzenText style="font-weight:400">
                        Journey overview
                    </RadzenText>
                </RadzenStack>

                <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                    <RadzenIcon Icon="drive_eta" Style="font-size:34px;
color:dimgray" />
                    <RadzenStack Orientation="Orientation.Vertical"
Style="padding:0rem !important;" Gap="0">
                        <RadzenText Style="padding:0rem !important;">
                            @trip.TripDetail.From
                        </RadzenText>
                        <RadzenText Style="padding:0rem !important; font-size:14px;
color:dimgray">
```

```
                            @($"{trip.TripDetail.From},
{trip.TripDetail.FromCountryCode}")
                        </RadzenText>
                    </RadzenStack>
                </RadzenStack>
                <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                    <div style="border-left: 5px solid dimgray; height:90px; margin-
left:15px;"></div>
                    <RadzenStack Orientation="Orientation.Vertical" Gap="0">
                        <RadzenText Style="font-size:13px; color:dimgray; margin-
left:10px;">
                            @trip.TripDetail.Duration
                        </RadzenText>
                        <RadzenText Style="font-size:13px; color:dimgray; margin-
left:10px;">
                            @trip.TripDetail.Distance
                        </RadzenText>
                    </RadzenStack>
                </RadzenStack>
                <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                    <RadzenIcon Icon="place" Style="font-size:34px; color:dimgray"
/>
                    <RadzenStack Orientation="Orientation.Vertical"
Style="padding:0rem !important;" Gap="0">
                        <RadzenText Style="padding:0rem !important;">
                            @trip.TripDetail.Destination
                        </RadzenText>
                        <RadzenText Style="padding:0rem !important; font-size:14px;
color:dimgray">
                            @($"{trip.TripDetail.Destination},
{trip.TripDetail.DestinationCountryCode}")
                        </RadzenText>
                    </RadzenStack>
                </RadzenStack>
            </RadzenStack>

            <RadzenStack Orientation="Orientation.Vertical"
AlignItems="AlignItems.Start" JustifyContent="JustifyContent.Left"
                         Gap="1rem" Wrap="FlexWrap.Wrap" class="rz-p-12 shadow-lg
border-5 border-dark"
                         Style="padding:1rem !important; background-color:white;
margin-left:0.3rem; margin-right:0.3rem;">

                <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center">
                    <RadzenIcon Icon="local_gas_station" Style="font-size:34px;
color:gray" />
                    <RadzenStack Orientation="Orientation.Vertical"
Style="padding:0rem !important;" Gap="0">
                        <RadzenText Style="padding:0rem !important; font-
size:14px;">
                            @($"Approximate fuel cost:
{trip.TripDetail.FuelCostStart}$-{trip.TripDetail.FuelCostStop}$")
                        </RadzenText>
                        <RadzenText Style="padding:0rem !important; font-size:12px;
color:dimgray">
                            Price estimate does not include rental car fees and
tolls. Calculation has been provided using OpenAI artificial intellegence.
                        </RadzenText>
                    </RadzenStack>
                </RadzenStack>
            </RadzenStack>
        </RadzenColumn>
    </RadzenRow>
```

```
    );
    }
}
```

Файл TripPlanner. BlazorUI/Pages/YesNoPopUp.razor

```
@inject DialogService DialogService

<div class="rz-p-12 rz-text-align-center">
    <RadzenButton Text="Confirm dialog" ButtonStyle="ButtonStyle.Secondary"
                  Click=@(args => DialogService.Confirm("Would you like to receive a
custom trip preferences?", "Preferences", new ConfirmOptions() { OkButtonText =
"Yes", CancelButtonText = "No" })) />
</div>
```

Файл TripPlanner. BlazorUI/Services/HttpService.cs

```
using System.Text;
using System.Text.Json;
using TripPlanner.BlazorUI.Data.Constants;

namespace TripPlanner.BlazorUI.Services
{
    public class HttpService
    {
        private readonly HttpClient httpClient;
        public HttpService(HttpClient httpClient)
        {
            this.httpClient = httpClient ?? throw new
ArgumentNullException(nameof(httpClient));
        }

        public async Task<TResponse> PostAsync<TRequest, TResponse>(string
controller, TRequest request, bool withJsonOutput = true)
        {
            var requestJson = JsonSerializer.Serialize(request);
            var content = new StringContent(requestJson, Encoding.UTF8,
"application/json");
            var response = await
httpClient.PostAsync(IntegrationDefinitions.TripPlannerAPIUrl + controller,
content);
            var responseContent = await response.Content.ReadAsStringAsync();
            response.EnsureSuccessStatusCode();

            var responseStream = await response.Content.ReadAsStringAsync();
            if (withJsonOutput)
                return JsonSerializer.Deserialize<TResponse>(responseStream, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });
            else
                return (TResponse)new object();
        }

        public async Task<TResponse> DeleteAsync<TResponse>(string controller,
object[] queryParameters, bool withJsonOutput = true)
        {
            var queryParametersJoined = "/" + string.Join("/", queryParameters);
```

```
            var url = IntegrationDefinitions.TripPlannerAPIUrl + controller +
queryParametersJoined;
            var response = await httpClient.DeleteAsync(url);

            response.EnsureSuccessStatusCode();

            var responseStream = await response.Content.ReadAsStringAsync();
            if (withJsonOutput)
                return JsonSerializer.Deserialize<TResponse>(responseStream, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });
            else
                return (TResponse)new object();
        }

        public async Task<TResponse> GetAsync<TResponse>(string controller, object[]
queryParameters)
        {
            var queryParametersJoined = "/" + string.Join("/", queryParameters);
            var url = IntegrationDefinitions.TripPlannerAPIUrl + controller +
queryParametersJoined;
            var response = await httpClient.GetAsync(url);

            response.EnsureSuccessStatusCode();

            var responseStream = await response.Content.ReadAsStringAsync();
            return JsonSerializer.Deserialize<TResponse>(responseStream, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });
        }

    }
}
```

Файл TripPlanner. BlazorUI/Services/StateContainer.cs

```
using TripPlanner.BlazorUI.Entities;

namespace TripPlanner.BlazorUI.Services
{
    public class StateContainer
    {
        public bool IsUserLogin { get; set; } = false;
        public User User { get; set; } = default!;

        public IEnumerable<SavedTrip> SavedTrips { get; set; }
    }
}
```

Файл TripPlanner. BlazorUI/Shared/MainLayout.razor

```
@using TripPlanner.BlazorUI.Data.Constants
@using TripPlanner.BlazorUI.Entities
@using TripPlanner.BlazorUI.Services;
@inherits LayoutComponentBase

<div id="page">
```

```
    <RadzenLayout>
        <RadzenDialog />
        @if (StateContainer.IsUserLogin)
        {
                <RadzenHeader>
                <RadzenStack Orientation="Orientation.Horizontal"
AlignItems="AlignItems.Center" Gap="0" Style="background-image:
url('/background.jpg');">
                    <RadzenSidebarToggle Click="@(() => sidebarExpanded =
!sidebarExpanded)" Style="color:black" />
                        <RadzenLabel Text="Trip Planner" Style="color:black"/>
                    </RadzenStack>
                </RadzenHeader>
            <RadzenSidebar @bind-Expanded="@sidebarExpanded" Style="background-
image: url('/background.jpg');">
                        <RadzenPanelMenu>
                    <RadzenPanelMenuItem Text="Home" Icon="home" Click="OnHome"
Style="background-image: url('/background.jpg');" />
                    <RadzenPanelMenuItem Text="Trip history" Icon="commute"
Click="OnSavedTrip" Style="background-image: url('/background.jpg');"/>
                    <RadzenPanelMenuItem Text="Log Out" Icon="logout"
Click="OnLogOut" Style="background-image: url('/background.jpg');"/>
                        </RadzenPanelMenu>
                    </RadzenSidebar>
        }
        <RadzenBody Style="padding: 0rem; background-size:cover;background-
position:center;background-image: url('/background.jpg');">
                @Body
            </RadzenBody>
    </RadzenLayout>
</div>

@code {
    bool sidebarExpanded = true;

    [Inject]
    public StateContainer StateContainer { get; set; }

    [Inject]
    public NavigationManager NavigationManager { get; set; }

    [Inject]
    public HttpService HttpService { get; set; }

    private void OnLogOut()
    {
        StateContainer.User = null;
        StateContainer.IsUserLogin = false;

        NavigationManager.NavigateTo("", true);
    }

    private async Task OnSavedTrip()
    {

        StateContainer.SavedTrips = await
HttpService.GetAsync<IEnumerable<SavedTrip>>(Controllers.TripController, new
object[] { StateContainer.User.Id });

        NavigationManager.NavigateTo("savedtrips", true);
    }

    private void OnHome()
    {
        NavigationManager.NavigateTo("", true);
    }
```

```
}
```

Файл TripPlanner. BlazorUI/_Imports.razor

```
@using System.Net.Http
@using Microsoft.AspNetCore.Authorization
@using Microsoft.AspNetCore.Components.Authorization
@using Microsoft.AspNetCore.Components.Forms
@using Microsoft.AspNetCore.Components.Routing
@using Microsoft.AspNetCore.Components.Web
@using Microsoft.AspNetCore.Components.Web.Virtualization
@using Microsoft.JSInterop
@using TripPlanner.BlazorUI
@using TripPlanner.BlazorUI.Shared
@using Radzen.Blazor
@using Radzen
```

Файл TripPlanner. BlazorUI/App.razor

```
<Router AppAssembly="@typeof(App).Assembly">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
    <NotFound>
        <PageTitle>Not found</PageTitle>
        <LayoutView Layout="@typeof(MainLayout)">
            <p role="alert">Sorry, there's nothing at this address.</p>
        </LayoutView>
    </NotFound>
</Router>
```

Файл TripPlanner. BlazorUI/appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Файл TripPlanner. BlazorUI/Program.cs

```
using Radzen;
using TripPlanner.BlazorUI.Services;

namespace TripPlanner.BlazorUI
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            builder.Services.AddRazorPages();
            builder.Services.AddServerSideBlazor();

            builder.Services.AddRadzenComponents();
            builder.Services.AddSingleton<StateContainer>();

            builder.Services.AddHttpClient();
            builder.Services.AddScoped<HttpService>();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (!app.Environment.IsDevelopment())
            {
                app.UseExceptionHandler("/Error");
                // The default HSTS value is 30 days. You may want to change this
for production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }

            app.UseHttpsRedirection();

            app.UseStaticFiles();

            app.UseRouting();

            app.MapBlazorHub();
            app.MapFallbackToPage("/_Host");

            app.Run();
        }
    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Програмне забезпечення для організації та планування маршрутів та поїздок

Програма та методика тестування

КПІ.ІТ-9406.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Павло РОДІОНОВ

Виконавець:
_____ Владислав КАЧУК

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення TripPlanner, розроблене для планування подорожей. TripPlanner складається з двох основних компонентів: веб-додатка, який реалізує користувацький інтерфейс (UI), та серверної частини (API), що забезпечує бізнес-логіку та взаємодію з базою даних. ПЗ розроблене з використанням сучасних технологій та архітектурних патернів для забезпечення надійності, масштабованості та високої продуктивності.

Тестування проводиться на таких платформах та середовищах:

- Операційні системи: Windows, Linux.
- Веб-браузери: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera.
- Хмарна платформа: Microsoft Azure, де розгорнуто серверну частину та базу даних.
- Інструменти для тестування: Postman (для тестування API), SQL Server Management Studio (для роботи з базою даних).

Тестування охоплює перевірку функціональних та нефункціональних вимог, включаючи коректність обробки даних, збереження та відновлення даних, сумісність з різними браузерами та операційними системами, а також продуктивність та масштабованість системи.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог.

– перевірка збереження даних.

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, Edge, Safari).

– перевірка сумісності застосунку з різними операційними системами (Windows, Linux).

– знаходження проблем, помилосk і недоліків з метою їх усунення.

– перевірка зручності графічого інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

Статичне тестування – перевірка програми разом з усією документацією, аналіз на предмет дотримання стандартів програмування.

Динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. Під час виконання кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми.

Функціональне тестування – перевірка відповідності реальної поведінки програмного забезпечення очікуваній.

Системне тестування – перевірка всього програмного забезпечення в цілому.

Мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

Тестування «чорної скриньки» – об’єктом тестування є функції, присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних.

Тестування «білої скриньки» – об’єктом тестування є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним.

Тестування «сірої скриньки» – об’єктом тестування є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування (End-to-End, E2E, Chain testing) та написанням unit-тестів, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення, так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- Динамічне тестування на відповідність функціональним вимогам – перевірка кожної функції програми, щоб переконатися в їх відповідності заданим специфікаціям.
- Тестування на різних веб-браузерах – перевірка сумісності додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, Edge).
- Тестування на різних операційних системах – перевірка працездатності додатку на Windows, Linux та MacOS.
- Тестування виведення повідомлень про помилку – перевірка коректності виведення повідомлень про помилки у випадку некоректних дій користувача або проблем з мережею.
- Тестування інтерфейсу користувача – перевірка зручності та інтуїтивності інтерфейсу, відповідності його очікуванням користувача.
- Тестування зручності використання – оцінка користувацького досвіду, виявлення можливих недоліків у взаємодії з програмою.
- Тестування продуктивності – перевірка швидкості завантаження та обробки даних, відповідності вимогам до продуктивності.
- Тестування безпеки – перевірка захищеності додатку від потенційних загроз, таких як SQL-ін'єкції, XSS-атаки тощо.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Програмне забезпечення для організації та планування маршрутів та поїздок

Керівництво користувача

КПІ.ІТ-9406.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Павло РОДІОНОВ

Виконавець:
_____ Владислав КАЧУК

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Програмне забезпечення "TripPlanner" є веб-додатком для планування та організації подорожей. Воно створене для зручного та ефективного планування подорожей користувачами.

Кінцева збірка програмного забезпечення:

- Версія: 1.0.0
- Платформа: Веб-додаток
- Мови програмування: HTML, CSS, JavaScript, C#
- Використані технології: .NET, GoogleMapsAPI, OpenAI Model 4o

Інсталяційна версія програмного забезпечення:

- Підтримка надається безпосередньо через веб-сайт програми. Користувачі можуть отримати доступ до програми, відвідавши веб-сторінку "TripPlanner".

Вміст репозиторію:

- Код веб-додатку зберігається у репозиторії на GitHub.
- Репозиторій містить вихідний код клієнтської та серверної частин, а також документацію та інші додаткові файли, необхідні для розгортання та розробки програми.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність будь-якого веб-браузера.

– наявність доступу до Інтернету;

– для встановлення додатку на мобільному пристрої повинно бути не менше 100 МБ вільної пам’яті.

2.2 Завантаження застосунку

Додаток є доступним на веб-сторінці розгорнутого додатку.

2.3 Перевірка коректної роботи

При переході на веб-сторінку розгорнутого додатку повинен завантажитись інтерфейс користувача у браузері. Стартове вікно неавторизованого користувача – Форма авторизації.

3 ВИКОНАННЯ ПРОГРАМИ

Авторизація користувача

Початковий стан: Користувач знаходиться на сторінці авторизації.

Дії: Введіть логін та пароль у відповідні поля. Натисніть кнопку "Log In".

Очікуваний результат: Користувач успішно авторизується і перенаправляється на головну сторінку.

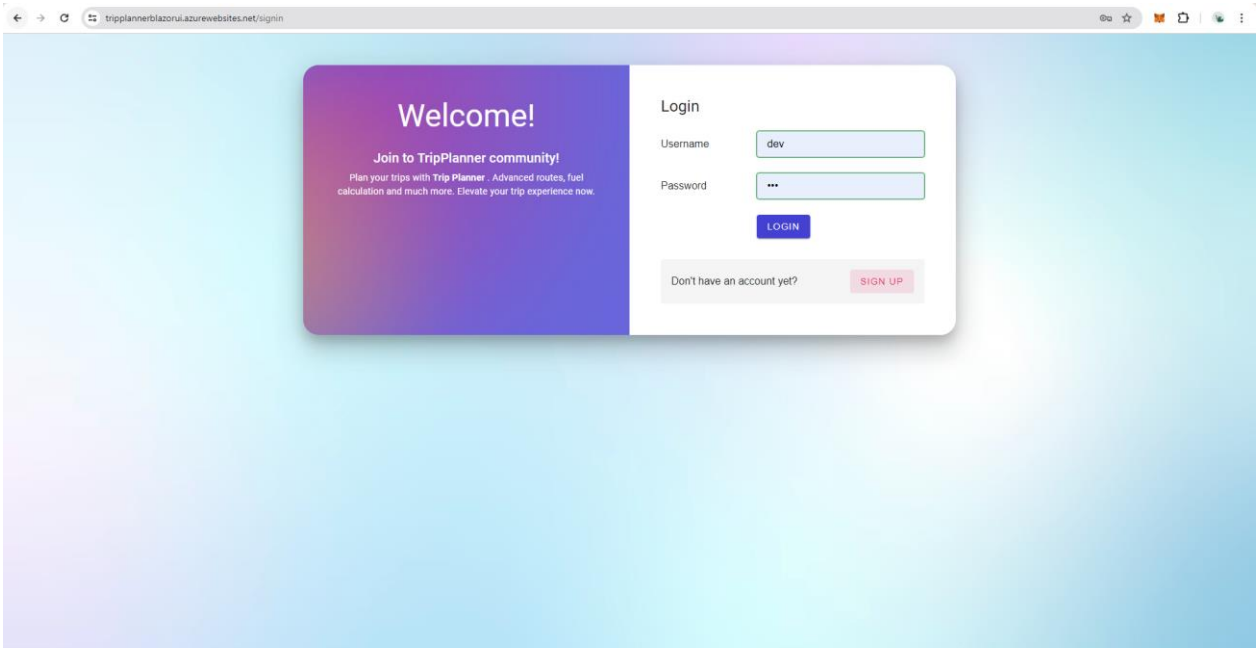


Рисунок 3.1 – Вікно авторизації користувача

Планування нової подорожі

Початковий стан: Користувач авторизований і знаходиться на головній сторінці.

Дії: Натисніть на кнопку "Plan a Trip". Введіть місце старту та місце призначення у відповідні поля. Натисніть кнопку "Plan".

Очікуваний результат: На мапі відображається маршрут подорожі з деталями, включаючи відстань, тривалість подорожі, рекомендовані зупинки та орієнтовні витрати на пальне.

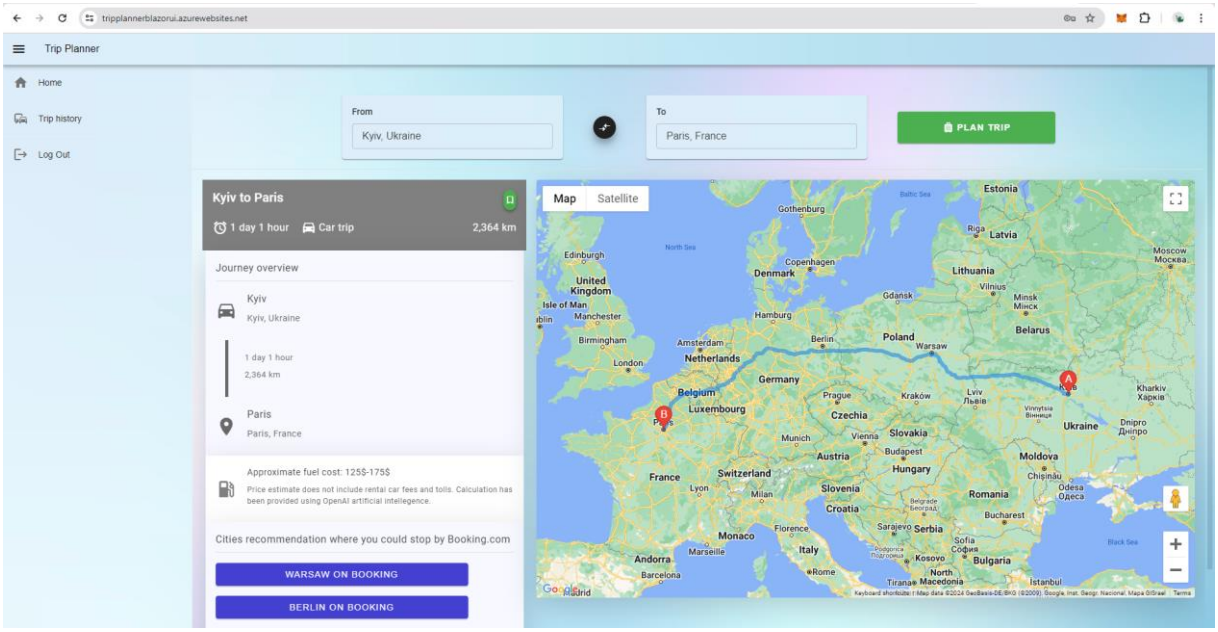


Рисунок 3.2 – Вікно головного меню

Перегляд деталізованої інформації про подорож

Початковий стан: Користувач бачить запланований маршрут на мапі.

Дії: Натисніть на деталі подорожі для перегляду додаткової інформації, такої як рекомендовані міста для зупинок та інші деталі маршруту.

Очікуваний результат: Відображається детальна інформація про подорож.

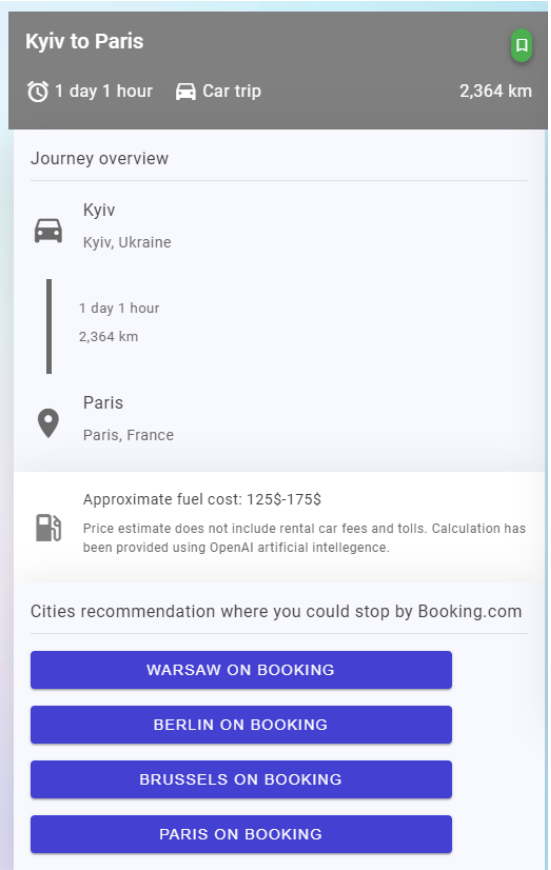


Рисунок 3.3 – Область деталей про подорож

Збереження подорожі

Початковий стан: Користувач переглядає деталі запланованої подорожі.

Дії: Натисніть на кнопку "Save Trip".

Очікуваний результат: Подорож зберігається у профілі користувача. З'являється повідомлення про успішне збереження.

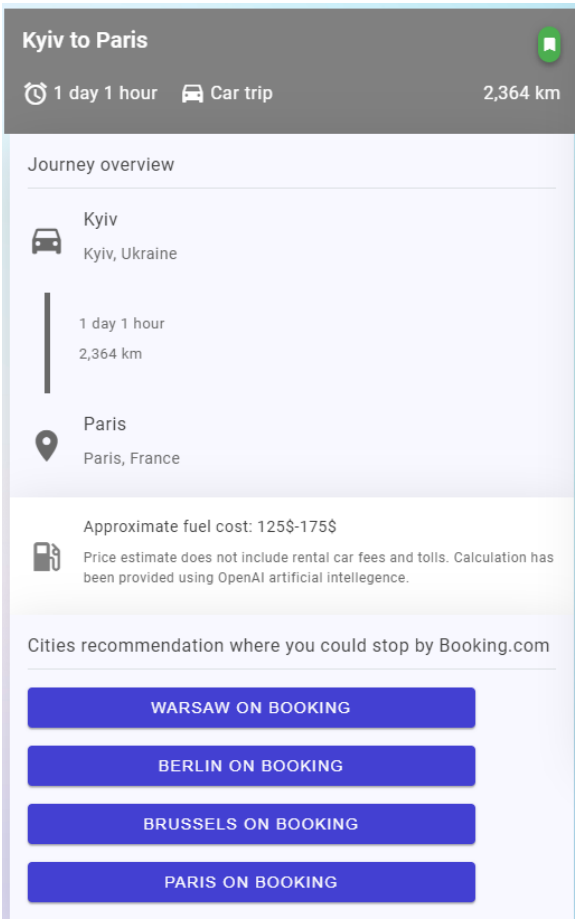


Рисунок 3.4 – Деталі про подорож після збереження

Перегляд збережених подорожей

Початковий стан: Користувач авторизований і знаходиться на головній сторінці.

Дії: Відкрийте бокове меню. Оберіть "Saved Trips".

Очікуваний результат: Відображається список всіх збережених подорожей користувача.

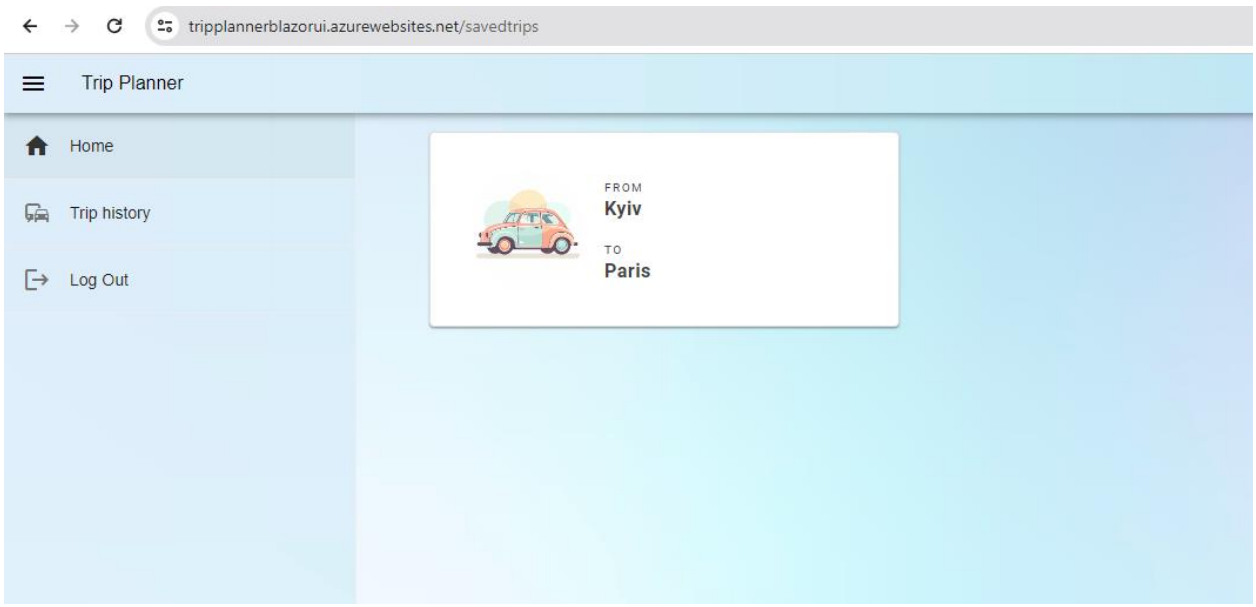


Рисунок 3.5 – Вікно збережених подорожей

Перегляд деталей збереженої подорожі

Початковий стан: Користувач знаходиться на сторінці збережених подорожей.

Дії: Натисніть на кнопку "Details" поруч з потрібною подорожжю.

Очікуваний результат: Відображаються деталі обраної подорожі.

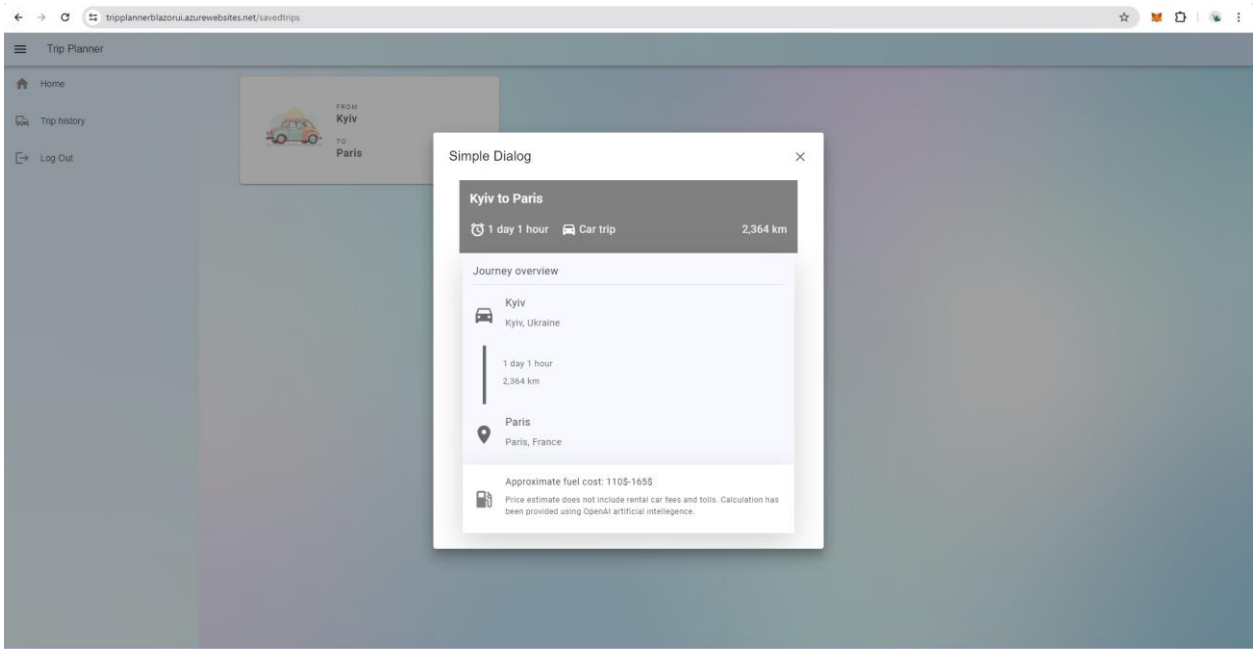


Рисунок 3.6 – Вікно деталей збереженої подорожі

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

Програмне забезпечення для організації та планування маршрутів та поїздок

Графічний матеріал

КПІ.ІТ-9406.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена МАРЧЕНКО

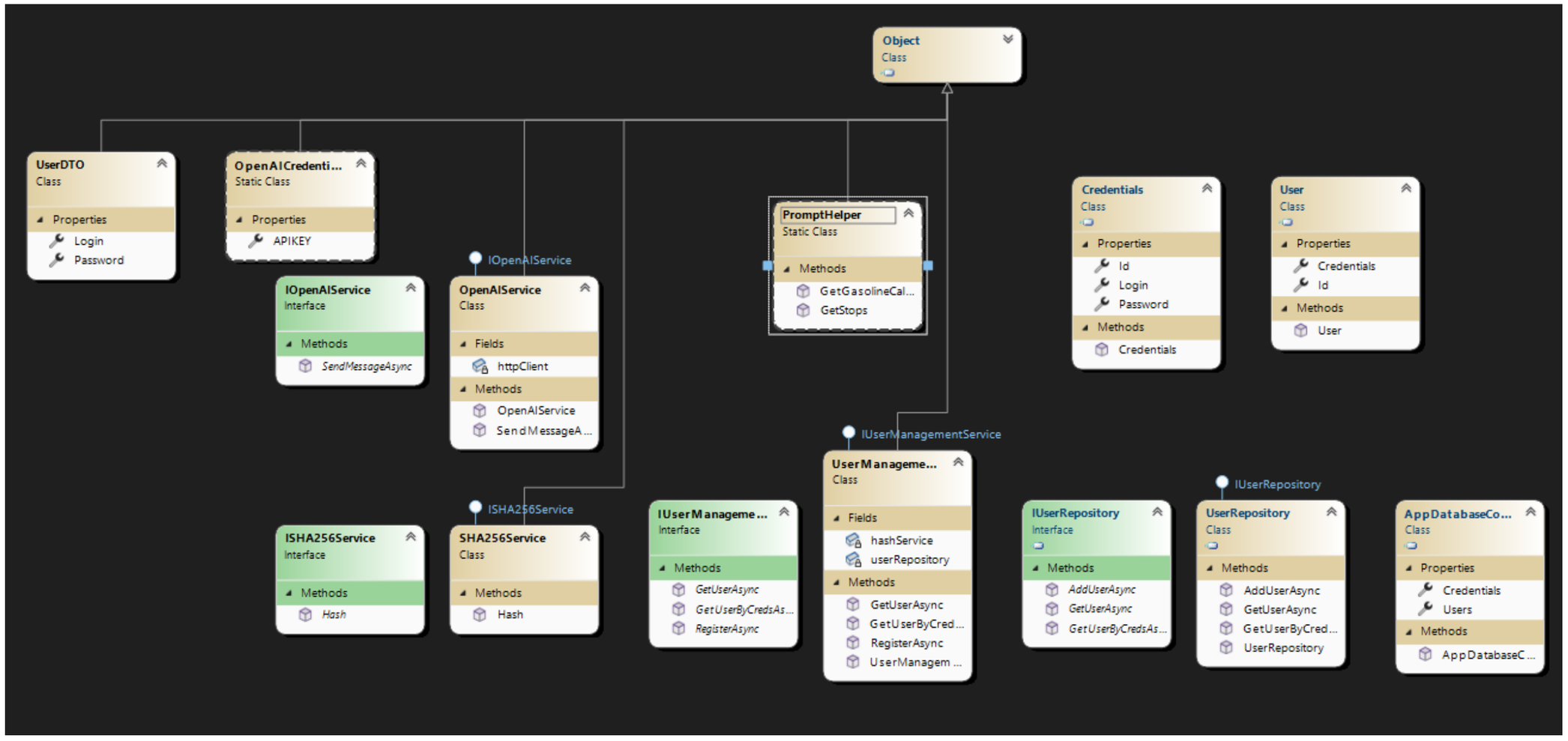
Нормоконтроль:

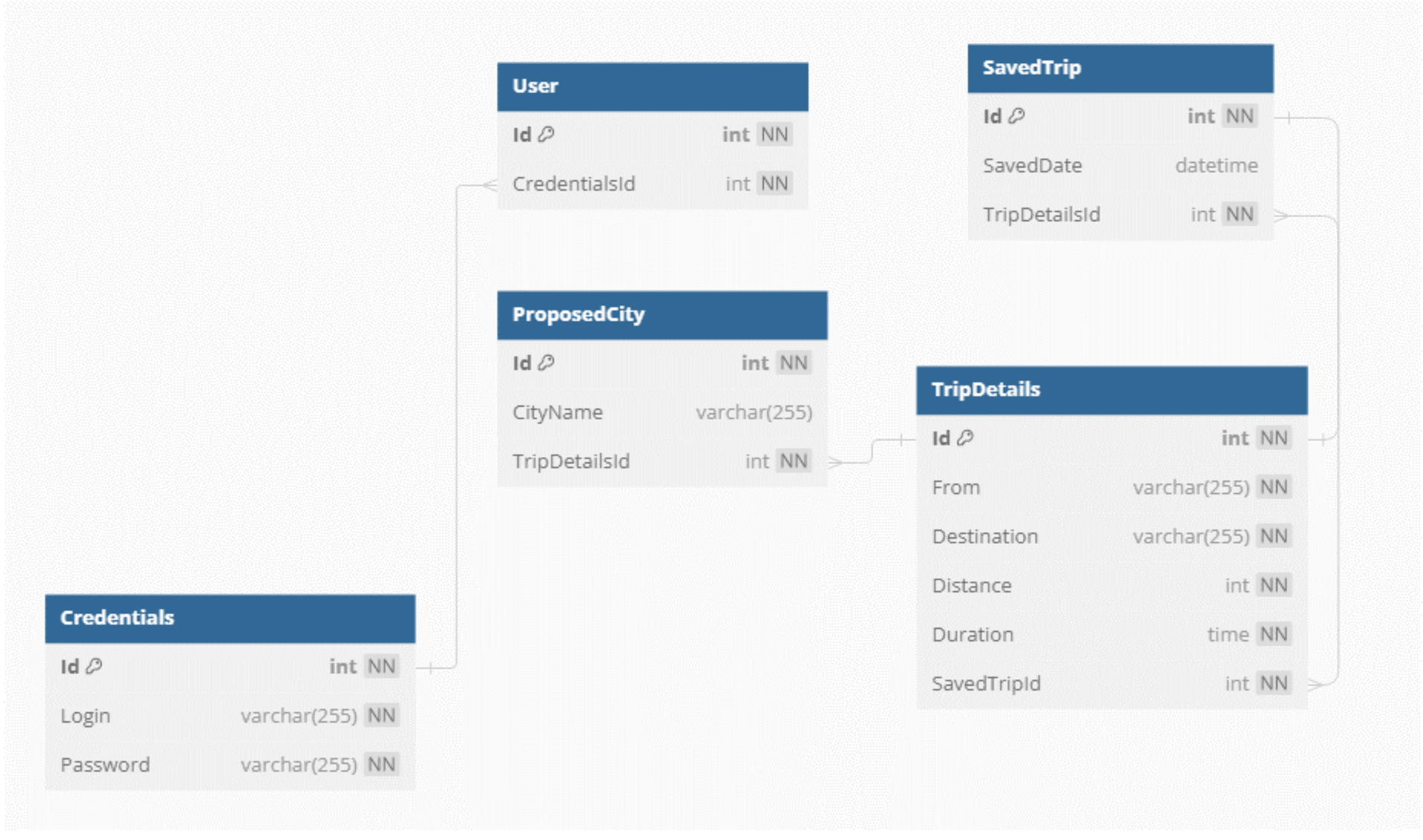
_____ Павло РОДІОНОВ

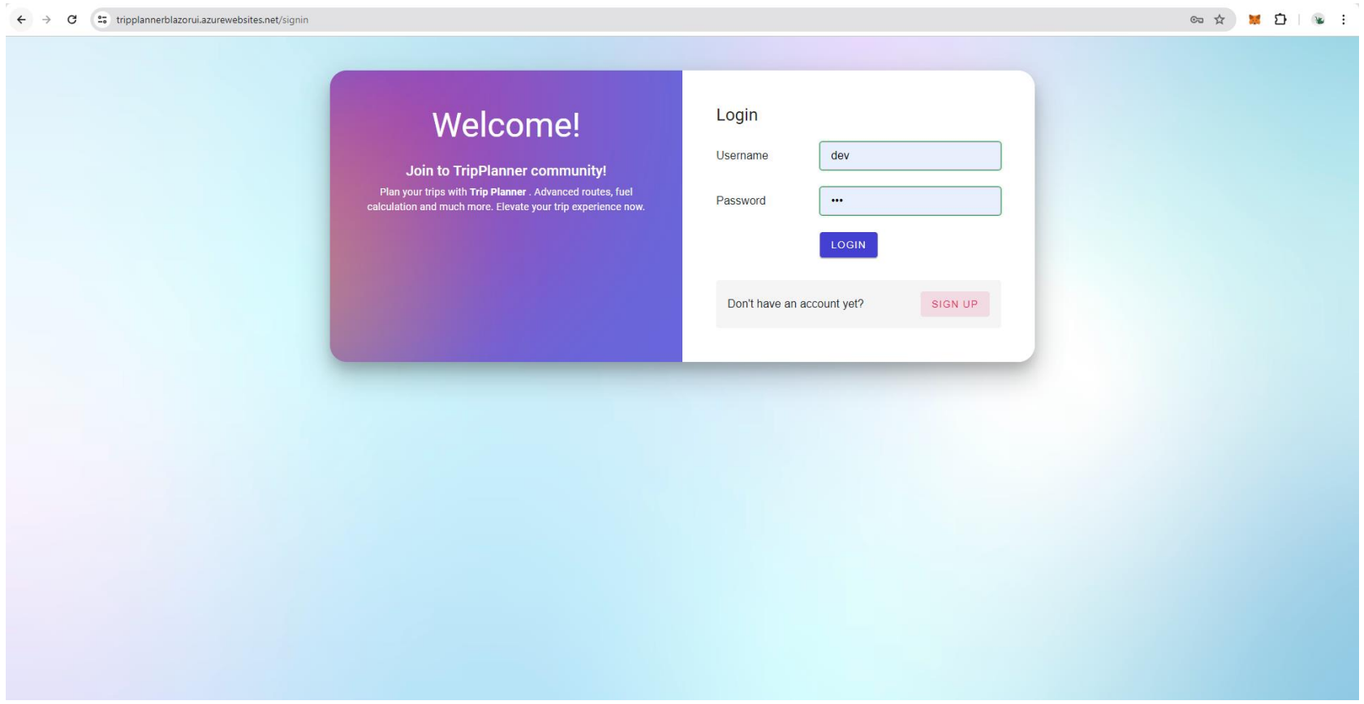
Виконавець:

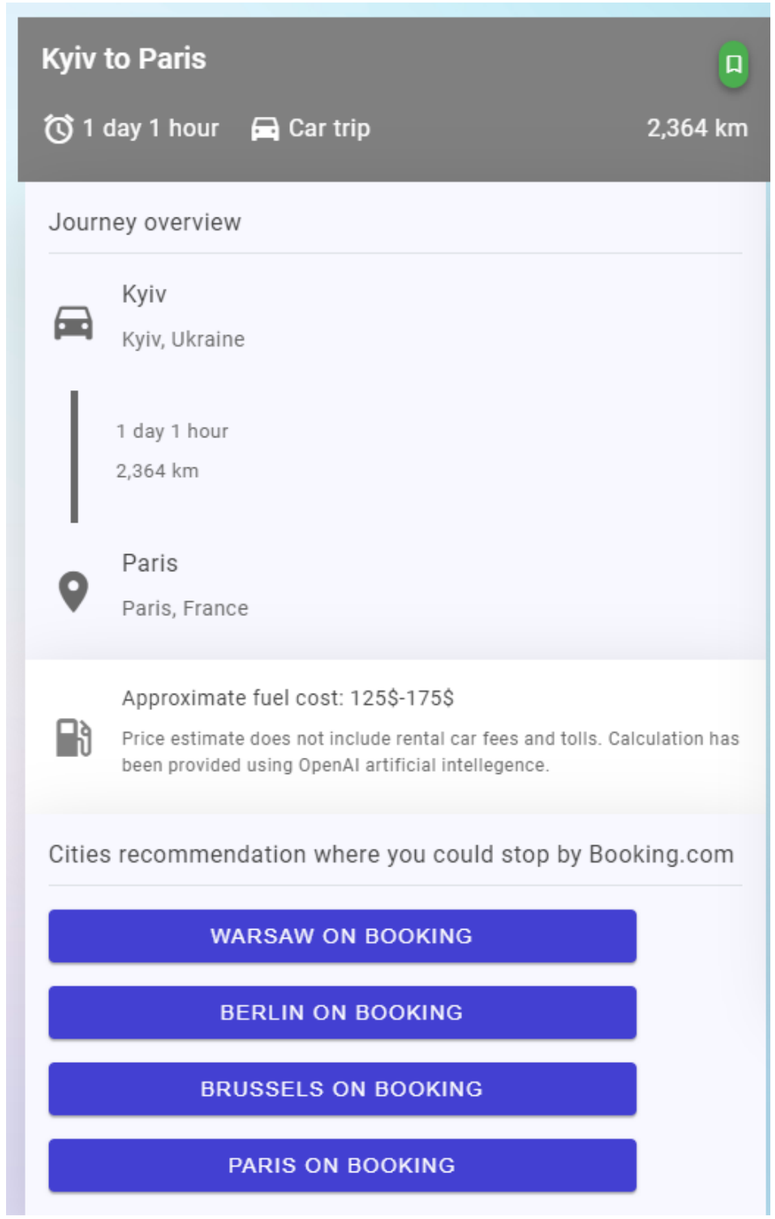
_____ Владислав КАЧУК

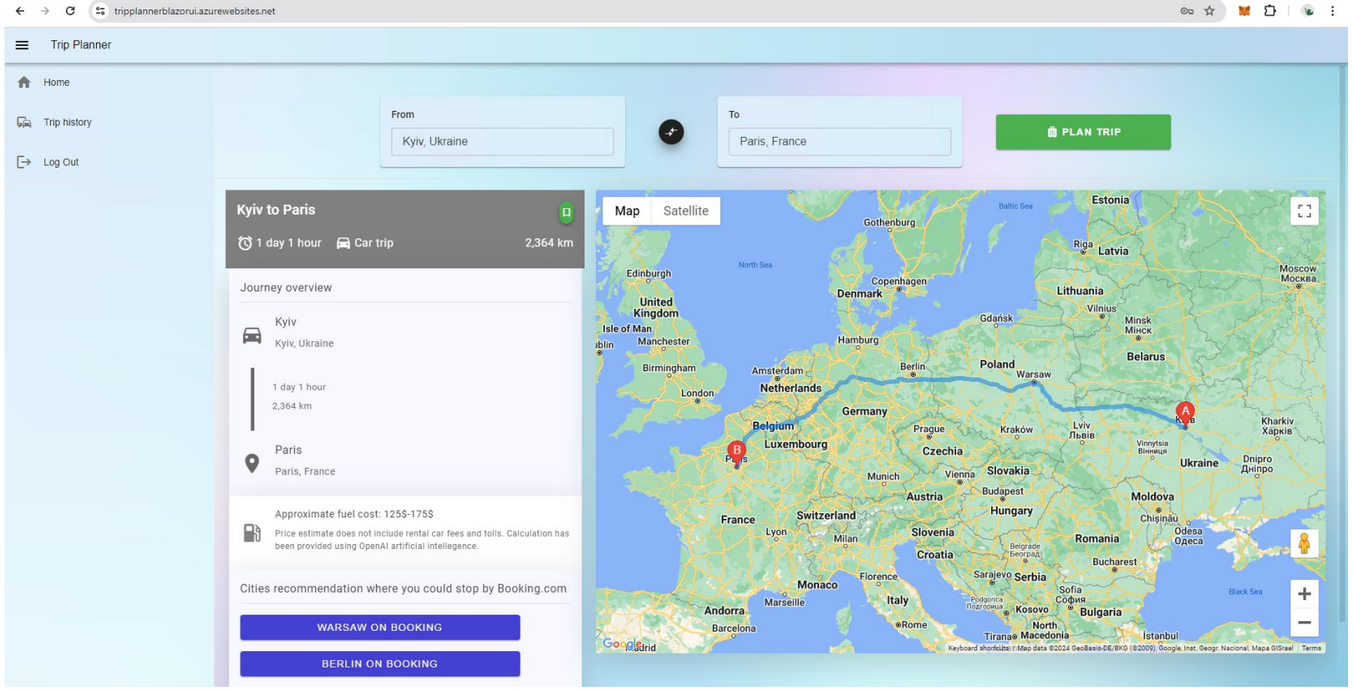
Київ – 2024

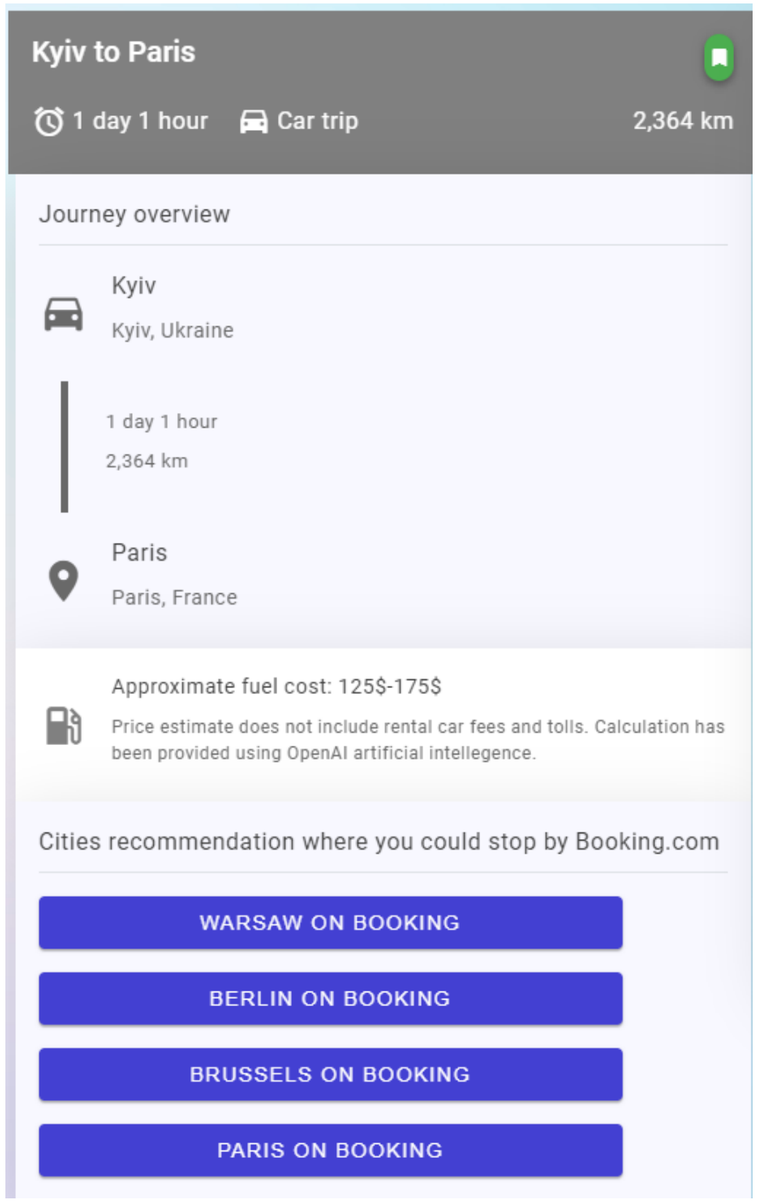

Object
Class
UserDTO
Class
Properties
Login
Password
OpenAICredenti...
Static Class
Properties
APIKEY
IOpenAIService
Interface
Methods
SendMessageAsync
IOpenAIService
OpenAIService
Class
Fields
httpClient
Methods
OpenAIService
SendMessageA...
PromptHelper
Static Class
Methods
GetGasolineCal...
GetStops
Credentials
Class
Properties
Id
Login
Password
Methods
Credentials
User
Class
Properties
Credentials
Id
Methods
User
IUserManagementService
ISHA256Service
Interface
Methods
Hash
ISHA256Service
SHA256Service
Class
Methods
Hash
IUserManageme...
Interface
Methods
GetUserAsync
GetUserByCredsAs...
RegisterAsync
UserManageme...
Class
Fields
hashService
userRepository
Methods
GetUserAsync
GetUserByCred...
RegisterAsync
UserManagem...
IUserRepository
Interface
Methods
AddUserAsync
GetUserAsync
GetUserByCredsAs...
IUserRepository
UserRepository
Class
Methods
AddUserAsync
GetUserAsync
GetUserByCred...
UserRepository
AppDatabaseCo...
Class
Properties
Credentials
Users
Methods
AppDatabaseC...

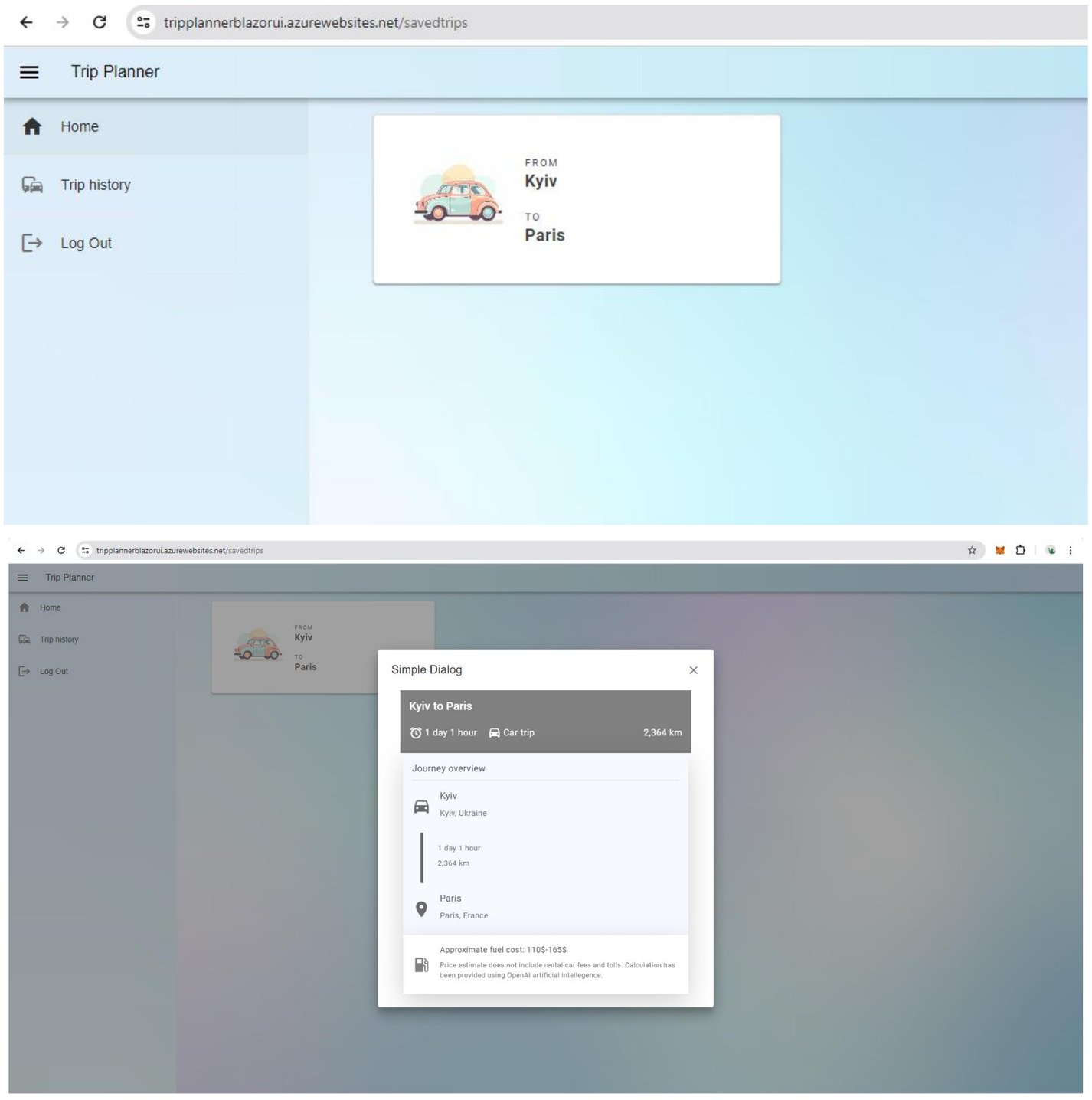

User
Id int NN
CredentialsId int NN
SavedTrip
Id int NN
SavedDate datetime
TripDetailsId int NN
ProposedCity
Id int NN
CityName varchar(255)
TripDetailsId int NN
TripDetails
Id int NN
From varchar(255) NN
Destination varchar(255) NN
Distance int NN
Duration time NN
SavedTripId int NN
Credentials
Id int NN
Login varchar(255) NN
Password varchar(255) NN


tripplannerblazorui.azurewebsites.net/signin
Welcome!
Join to TripPlanner community!
Plan your trips with Trip Planner . Advanced routes, fuel calculation and much more. Elevate your trip experience now.
Login
Username
dev
Password
LOGIN
Don't have an account yet?
SIGN UP


Kyiv to Paris
1 day 1 hour
Car trip
2,364 km
Journey overview
Kyiv
Kyiv, Ukraine
1 day 1 hour
2,364 km
Paris
Paris, France
Approximate fuel cost: 125-175
Price estimate does not include rental car fees and tolls. Calculation has been provided using OpenAI artificial intellegence.
Cities recommendation where you could stop by Booking.com
WARSAW ON BOOKING
BERLIN ON BOOKING
BRUSSELS ON BOOKING
PARIS ON BOOKING


tripplannerblazorui.azurewebsites.net
Trip Planner
Home
Trip history
Log Out
From
Kyiv, Ukraine
To
Paris, France
PLAN TRIP
Kyiv to Paris
1 day 1 hour
Car trip
2,364 km
Journey overview
Kyiv
Kyiv, Ukraine
1 day 1 hour
2,364 km
Paris
Paris, France
Approximate fuel cost: 125-175
Price estimate does not include rental car fees and tolls. Calculation has been provided using OpenAI artificial intellegence.
Cities recommendation where you could stop by Booking.com
WARSAW ON BOOKING
BERLIN ON BOOKING
Map
Satellite


Kyiv to Paris
1 day 1 hour
Car trip
2,364 km
Journey overview
Kyiv
Kyiv, Ukraine
1 day 1 hour
2,364 km
Paris
Paris, France
Approximate fuel cost: 125-175
Price estimate does not include rental car fees and tolls. Calculation has been provided using OpenAI artificial intellegence.
Cities recommendation where you could stop by Booking.com
WARSAW ON BOOKING
BERLIN ON BOOKING
BRUSSELS ON BOOKING
PARIS ON BOOKING


tripplannerblazorui.azurewebsites.net/savedtrips
Trip Planner
Home
Trip history
Log Out
FROM
Kyiv
TO
Paris
Simple Dialog
Kyiv to Paris
1 day 1 hour
Car trip
2,364 km
Journey overview
Kyiv
Kyiv, Ukraine
1 day 1 hour
2,364 km
Paris
Paris, France
Approximate fuel cost: 110-165
Price estimate does not include rental car fees and tolls. Calculation has been provided using OpenAI artificial intellegence.