МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ВЕБЗАСТОСУНОК ДЛЯ ПІДБОРУ ТУРИСТИЧНИХ ПОДОРОЖЕЙ З ІНТЕГРОВАНИМ ЧАТБОТОМ

Виконала: студентка групи 2П-22
Спеціальності
121 Інженерія програмного забезпечення
Анна АНТОНЕНКО
Керівник:
Вікторія НЕМЧЕНКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп’ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

(підпис)

«_____» _____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Антоненко Анні Яківні

1. Тема кваліфікаційної роботи «Вебзастосунок для підбору туристичних подорожей з інтегрованим чатботом»

Керівник роботи Немченко Вікторія Юріївна, викладач другої категорії затверджені наказом закладу фахової передвищої освіти

від «06» жовтня 2025 року №84/1-у

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи мова програмування JavaScript, бібліотека React, середовище збірки Vite, серверне середовище Node.js, фреймворк Express.js, база даних MongoDB Atlas, бібліотека Mongoose, механізм авторизації JWT, зовнішній сервіс Gemini API, середовище розробки Visual Studio Code, інструменти тестування Jest, Supertest, Playwright, k6.

4. Зміст кваліфікаційної роботи аналіз онлайн-сервісів для підбору туристичних подорожей; аналіз існуючих цифрових рішень у сфері туризму; постановка задачі розроблення вебзастосунку; визначення функціональних і нефункціональних вимог; вибір архітектури та технологічного стеку; проєктування серверної частини, бази даних і клієнтського інтерфейсу; реалізація каталогу турів, авторизації, бронювання, адміністративної панелі, відгуків та інтегрованого чатбота; тестування основних функціональних сценаріїв і backend API.

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1. Аналіз предметної галузі та постановка задачі	09.12.2025	
3	Розділ 2. Проєктування та реалізація вебзастосунку	10.03.2026	
4	Розділ 3. Тестування програмного продукту	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	10.06.2026	

Студент _____ Анна АНТОНЕНКО

(підпис)

Керівник роботи _____ Вікторія НЕМЧЕНКО

(підпис)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробленню вебзастосунку для підбору туристичних подорожей з інтегрованим чатботом. У роботі проаналізовано туристичні онлайн-сервіси, їхні функціональні можливості та підходи до пошуку, фільтрації, бронювання й персоналізованої підтримки користувача.

У межах роботи реалізовано вебзастосунок, що забезпечує перегляд каталогу турів, пошук і фільтрацію пропозицій, відкриття детальної сторінки туру, додавання турів в обране, створення заявки на бронювання, авторизацію користувачів, роботу з профілем, відгуками та адміністративною панеллю. Окремо реалізовано чатбота, який допомагає користувачу уточнити параметри подорожі та отримати рекомендації на основі турів, збережених у базі даних.

Клієнтську частину розроблено з використанням React, Vite, Tailwind CSS, React Router та Axios. Серверну частину реалізовано на Node.js та Express.js. Для збереження даних використано MongoDB Atlas і Mongoose. Авторизацію побудовано на JWT-токенах, а чатбот інтегровано з використанням Gemini API.

Працездатність системи перевірено за допомогою модульного, інтеграційного, функціонального, end-to-end, API, навантажувального та ручного тестування. Результати тестування підтвердили коректність основних сценаріїв взаємодії користувача із системою.

Ключові слова: ВЕБЗАСТОСУНОК, ТУРИСТИЧНІ ПОДОРОЖІ, ЧАТБОТ, БРОНЮВАННЯ, КАТАЛОГ ТУРІВ.

ANNOTATION

The qualification work is devoted to the development of a web application for selecting tourist trips with an integrated chatbot. The paper analyzes online travel services, their functionality, and approaches to search, filtering, booking, and personalized user support.

The developed web application provides a tour catalog, search and filtering of travel offers, detailed tour pages, favorites, booking requests, user authentication, user profile functionality, reviews, and an administrative panel. A chatbot was also implemented to help users clarify travel preferences and receive recommendations based on tours stored in the database.

The client side was developed using React, Vite, Tailwind CSS, React Router, and Axios. The server side was implemented with Node.js and Express.js. MongoDB Atlas and Mongoose were used for data storage. Authentication is based on JWT tokens, and the chatbot is integrated using the Gemini API.

The system functionality was verified through unit, integration, functional, end-to-end, API, load, and manual testing. The testing results confirmed the correctness of the main user interaction scenarios.

Keywords: WEB APPLICATION, TOURIST TRIPS, CHATBOT, BOOKING, TOUR CATALOG

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ..... 3

ВСТУП.....4

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ.....7

1.1 Особливості функціонування онлайн-сервісів для підбору туристичних подорожей.....7

1.2 Аналіз існуючих рішень та їх функціональних можливостей.....15

1.3. Постановка задачі на основі аналізу предметної галузі.....23

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ.....26

2.1 Визначення вимог до вебзастосунку.....26

2.2 Вибір архітектури та технологічного стеку..... 33

2.3 Проєктування серверної частини, бази даних та клієнтського інтерфейсу... 39

2.4 Реалізація та інтеграція чатбота у вебзастосунок.....51

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ..... 59

3.1 Вибір методів тестування.....59

3.2 Розроблення тест-плану вебзастосунку.....61

3.3 Проведення тестування та аналіз отриманих результатів.....66

ВИСНОВКИ..... 75

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 76

ДОДАТКИ.....79

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

AI	Artificial Intelligence, штучний інтелект.
API	Application Programming Interface, програмний інтерфейс взаємодії між клієнтською та серверною частинами вебзастосунку.
BPNM	Business Process Model and Notation, нотація моделювання бізнес-процесів.
CSS	Cascading Style Sheets, мова стилізації вебсторінок.
E2E	End-to-End, наскрізне тестування користувацьких сценаріїв.
HTTP	HyperText Transfer Protocol, протокол передавання даних у вебсередовищі.
JSON	JavaScript Object Notation, текстовий формат обміну структурованими даними.
JWT	JSON Web Token, токен для автентифікації та авторизації користувача.
SPA	Single Page Application, односторінковий вебзастосунок.
ШІ	штучний інтелект

ВСТУП

Сучасна туристична сфера активно використовує цифрові технології, оскільки значна частина процесів, пов’язаних із пошуком і вибором подорожей, поступово переходить у вебсередовище. Користувачі все частіше звертаються до онлайн-сервісів, щоб переглянути доступні туристичні пропозиції, порівняти напрями, вартість, умови проживання, тривалість туру та інші характеристики. Такий формат є зручним, адже дає змогу самостійно обирати подорож у будь-який час без обов’язкового звернення до офісу туристичної агенції.

Водночас велика кількість доступних пропозицій не завжди полегшує процес вибору. Навіть за наявності фільтрів користувач часто змушений переглядати багато варіантів, порівнювати їх між собою та самостійно визначати, який тур найбільше відповідає його очікуванням. Особливо складною така задача стає тоді, коли людина ще не має чітко сформованого запиту, а лише приблизно розуміє бажаний бюджет, напрям, кількість туристів або формат відпочинку. У таких умовах звичайного каталогу може бути недостатньо, оскільки користувачу потрібна не лише інформація про тури, а й допомога у виборі.

Актуальність теми зумовлена потребою у створенні зручних цифрових рішень для підбору туристичних подорожей, які поєднують класичний пошук із персоналізованою підтримкою користувача. Одним зі способів удосконалення такого процесу є інтеграція чатбота у вебзастосунок. Чатбот може виконувати роль первинного цифрового консультанта: приймати запити у вільній формі, ставити уточнювальні запитання, визначати основні параметри подорожі та пропонувати релевантні варіанти з наявного каталогу. Це дозволяє частково автоматизувати процес, який зазвичай виконує менеджер або туроператор, скоротити час пошуку туру та зробити взаємодію із сервісом більш індивідуальною.

Об’єктом дослідження є програмна система підбору туристичних подорожей у вебсередовищі.

Предметом дослідження є технології, методи та програмні засоби розробки вебзастосунку для підбору туристичних подорожей з інтегрованим чатботом.

Метою кваліфікаційної роботи є створення вебзастосунку для підбору туристичних подорожей з можливістю пошуку, бронювання та отримання персоналізованих рекомендацій через чатбота.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати туристичні вебсервіси та використання чатботів для підбору подорожей;

– розглянути технології веброзробки й обґрунтувати вибір архітектури та технологічного стеку;

– сформувати вимоги до системи та спроєктувати її структуру, базу даних, серверну частину й інтерфейс;

– реалізувати вебзастосунок із пошуком, переглядом, бронюванням турів, авторизацією, адмін-панеллю та чатботом;

– протестувати роботу вебзастосунку й перевірити основні сценарії взаємодії користувача із системою.

Практичне значення одержаних результатів полягає у тому, що розроблений вебзастосунок може бути використаний як основа для туристичного онлайн-сервісу або сайту туристичної агенції. Система дозволяє зручно представити туристичні пропозиції, спростити процес вибору туру для користувача та забезпечити адміністратора інструментами для керування турами й заявками. Інтегрований чатбот підвищує зручність взаємодії, оскільки користувач може не лише самостійно переглядати каталог, а й отримувати рекомендації відповідно до власних побажань.

Очікуваним результатом кваліфікаційної роботи є функціональний вебзастосунок для підбору туристичних подорожей, який містить клієнтську та серверну частину, базу даних, механізм авторизації, каталог турів, систему

бронювання, особистий кабінет користувача, адміністративну панель і модуль чатбота. Розроблена система має забезпечувати пошук турів за параметрами, перегляд детальної інформації про подорожі, створення та обробку заявок, а також рекомендацію турів через інтегрований чатбот.

Апробація кваліфікаційної роботи. Результати, наведені в межах поточної роботи, були представлені на XVIII Всеукраїнській науково-практичній конференції «Тенденції розвитку ІТ-технологій в Україні»

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Особливості функціонування онлайн-сервісів для підбору туристичних подорожей

Онлайн-сервіси для підбору туристичних подорожей є веборієнтованими програмними системами, що забезпечують пошук, перегляд, порівняння, попереднє замовлення або бронювання туристичних пропозицій. Їхнє функціонування ґрунтується на перетворенні великого масиву туристичної інформації на структурований набір даних, доступний для користувача через вебінтерфейс. У межах таких сервісів туристична пропозиція подається не як довільний опис подорожі, а як запис із визначеними параметрами: країна, місто або курорт, дата виїзду, тривалість, вартість, кількість туристів, тип харчування, категорія готелю, умови проживання, транспортне забезпечення, наявність додаткових послуг і можливість оформлення заявки. Саме така структуризація створює технічну основу для пошуку, фільтрації, сортування та подальшого добору варіантів [3; 12].

Розвиток туристичних онлайн-сервісів є наслідком цифрової трансформації туристичної галузі. Процеси, які раніше виконувалися переважно через безпосередню взаємодію з менеджером туристичної агенції, поступово переносяться у вебсередовище. Користувач може самостійно переглядати напрями, порівнювати ціни, аналізувати умови проживання, перевіряти доступні дати, надсилати заявку або ініціювати бронювання без обов’язкового звернення до консультанта на першому етапі вибору. Унаслідок цього туристичний вебсервіс виконує функцію цифрового посередника між користувачем і туристичним продуктом, а також частково автоматизує інформаційні та організаційні процеси, пов’язані з добором подорожей [1; 4; 7].

Основним компонентом онлайн-сервісу для підбору туристичних подорожей є каталог пропозицій [12]. Він забезпечує централізоване зберігання та відображення інформації про доступні тури. Каталог виконує не лише інформаційну, а й навігаційну функцію, оскільки саме через нього користувач отримує первинний доступ до туристичних варіантів. Типова картка туру містить стислий набір характеристик: назву, напрям, зображення, ціну, тривалість, категорію готелю, тип харчування та короткий опис. Такий формат дає змогу швидко оцінити пропозицію й вирішити, чи варто переходити до детального перегляду. На рис. 1.1 зображена діаграма активності, яка показує можливі взаємодії між клієнтом та системою [12].

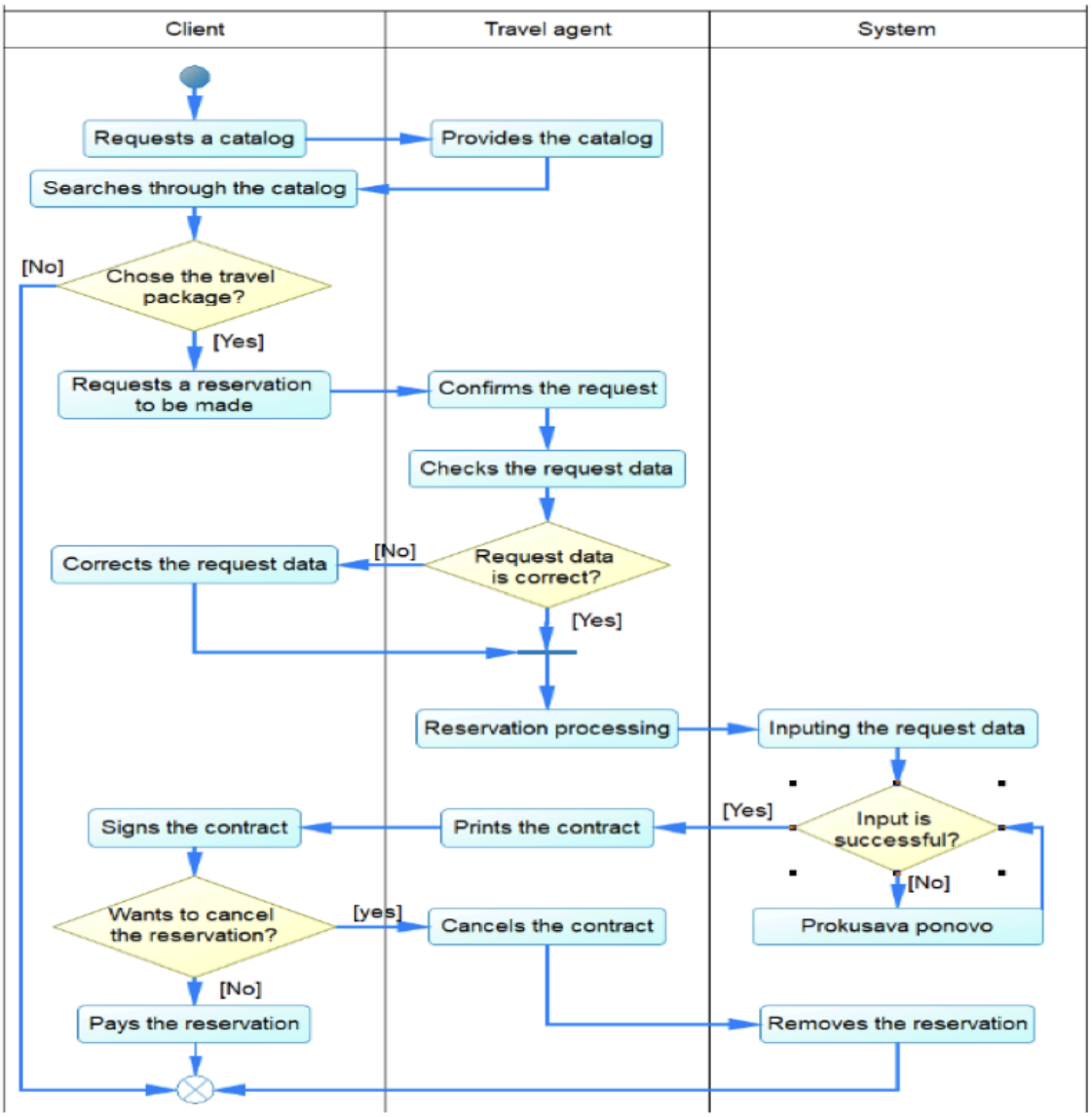

Рисунок 1.1 – Діаграма активності взаємодії між клієнтом та системою

Перевагою каталогу є можливість подати значну кількість туристичних пропозицій в одному інтерфейсі. Це скорочує час первинного пошуку, оскільки користувач не потребує перегляду розрізнених джерел інформації. Водночас каталог сам по собі не гарантує якісного добору подорожі. Якщо система містить багато пропозицій, але не має зручної логіки пошуку, користувач стикається з інформаційним перевантаженням. У такому випадку вебсервіс формально надає доступ до великого обсягу даних, однак не завжди допомагає користувачу прийняти рішення. Отже, кількість пропозицій у каталозі не є достатнім показником ефективності туристичного онлайн-сервісу.

Пошуково-фільтраційний механізм є основним способом зменшення інформаційного навантаження [3; 12]. Пошук дає змогу знайти пропозиції за назвою країни, міста, курорту або ключовими словами, а фільтрація – обмежити результати за ціною, датою, тривалістю, кількістю туристів, типом харчування, категорією готелю або форматом відпочинку. Сортування результатів за вартістю, популярністю, датою виїзду чи іншими критеріями додатково впорядковує процес вибору. Такі інструменти є необхідними для будь-якого туристичного вебсервісу, оскільки без них каталог перетворюється на пасивний перелік пропозицій.

Разом із тим фільтраційний підхід має суттєве обмеження: він ефективний лише тоді, коли користувач уже розуміє, що саме шукає. Для застосування фільтрів потрібно заздалегідь визначити країну, бюджет, дати, тривалість, тип харчування або рівень готелю. Якщо користувач має нечіткий запит, наприклад бажає «спокійний відпочинок біля моря», «недорогу подорож для двох» або «теплий напрям восени», традиційна система фільтрів не завжди допомагає швидко сформувати конкретний варіант. У цьому полягає одна з ключових проблем туристичних онлайн-сервісів: система добре працює з формалізованими параметрами, але гірше підтримує етап формування самого запиту [8; 10].

Туристична подорож є складним об’єктом вибору, оскільки рішення користувача залежить від сукупності взаємопов’язаних критеріїв. Вартість туру

не може оцінюватися окремо від тривалості, умов проживання, харчування, розташування готелю, сезону, складу туристів і мети поїздки. Дешевша пропозиція може бути менш доцільною через незручні дати, віддаленість готелю від інфраструктури або невідповідний тип харчування. Водночас дорожчий варіант може краще відповідати очікуванням користувача через оптимальну тривалість, зручне розташування, вищий рівень сервісу або відповідність формату відпочинку. Тому механічне сортування за ціною не забезпечує повноцінного підбору подорожі [3].

Сторінка детального перегляду туру частково компенсує обмеження короткої картки пропозиції. Вона має містити розширений опис подорожі, інформацію про готель, умови проживання, тип харчування, маршрут або місце відпочинку, фотографії, доступні дати, вартість і правила оформлення заявки. На цьому етапі користувач переходить від поверхневого перегляду до змістовного аналізу конкретного туристичного продукту. Проте навіть детальна сторінка не повністю розв’язує проблему вибору, оскільки вона подає інформацію, але не завжди пояснює, наскільки конкретна пропозиція відповідає індивідуальній ситуації користувача.

Критично важливою характеристикою туристичного онлайн-сервісу є актуальність даних. Туристичні пропозиції швидко змінюються через сезонність, попит, наявність місць, зміну цін, умови перевізників, оновлення готельної інформації та зовнішні економічні фактори. Якщо каталог не оновлюється своєчасно, користувач може отримати застарілу інформацію про вартість, доступність або умови туру. Це знижує довіру до сервісу й ускладнює подальше оформлення заявки. Тому онлайн-сервіс для підбору подорожей повинен мати не лише зручний користувацький інтерфейс, а й адміністративні механізми підтримки актуальності інформації [1; 2].

Адміністративна частина забезпечує керування вмістом вебсервісу. Через неї можуть виконуватися додавання нових турів, редагування описів, оновлення цін, зміна дат, завантаження зображень, видалення неактуальних пропозицій і перегляд отриманих заявок. Без такого функціоналу сервіс залишається

технічно обмеженим, оскільки будь-які зміни потребують прямого втручання у програмний код або базу даних. Для туристичної сфери це є недоцільним, оскільки інформація про тури має регулярно оновлюватися. Отже, адміністративний модуль є не допоміжним, а необхідним елементом практично придатного туристичного вебзастосунку.

Оформлення заявки або бронювання є завершальним етапом базового користувацького сценарію. Після вибору туру користувач передає системі контактні дані, кількість туристів, обрану пропозицію та додаткові побажання. На цьому етапі вебсервіс виконує організаційну функцію, оскільки фіксує зацікавленість користувача й створює підставу для подальшого опрацювання запиту адміністратором або менеджером. Якщо заявка не зберігається у базі даних або не має статусу опрацювання, система не забезпечує повноцінного циклу взаємодії. Тому функціонал заявок є важливим переходом від простого інформаційного каталогу до прикладного сервісу взаємодії з клієнтом [12].

На рис. 1.2 зображена узагальнена BPMN-модель процесу бронювання туристичної подорожі, що відображає перехід від вибору туру до створення заявки в системі [11].

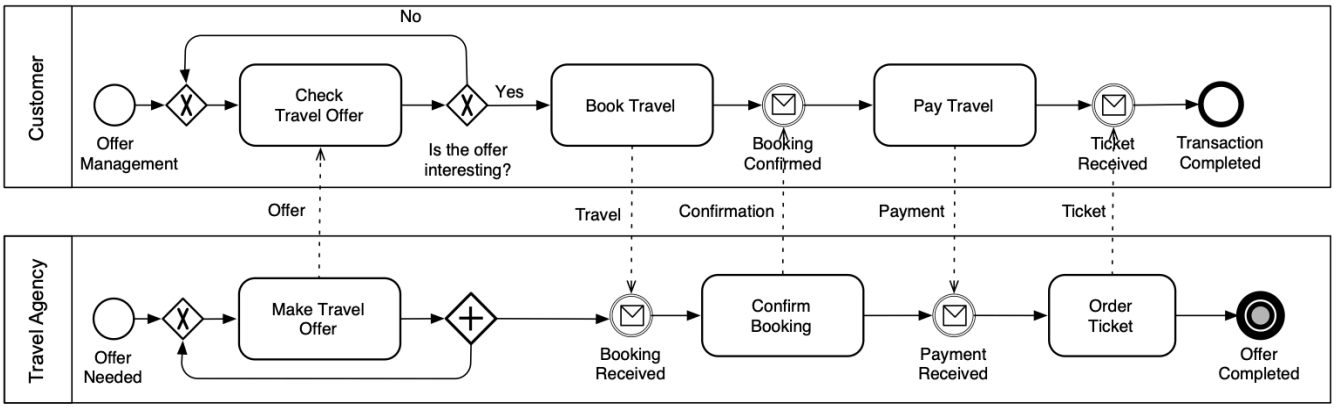

Рисунок 1.2 – BPMN-діаграма процесу бронювання туристичної подорожі

Користувацький профіль розширює можливості персоналізованої взаємодії. Він може містити основні дані користувача, історію заявок, збережені

тури або попередні дії в системі. Такий функціонал спрощує повторне використання сервісу, оскільки користувачеві не потрібно щоразу вводити однакову інформацію. Крім того, профіль створює основу для персоналізації: система може враховувати попередні перегляди, обрані напрями, бюджетні межі або типи відпочинку. Проте персоналізація потребує відповідального ставлення до даних користувача, оскільки вона пов’язана зі зберіганням персональної інформації та історії взаємодії [12].

Безпека є одним із критичних аспектів функціонування туристичних онлайн-сервісів. Такі системи можуть обробляти імена, номери телефонів, електронні адреси, дані авторизації, історію заявок і потенційно платіжну інформацію. Через це вебзастосунок має забезпечувати захищене зберігання паролів, контроль доступу до особистих даних, розмежування ролей користувача й адміністратора, а також коректну обробку помилок. Недостатній рівень безпеки може призвести не лише до технічних збоїв, а й до втрати довіри користувачів. Тому в туристичному сервісі зручність інтерфейсу повинна поєднуватися з базовими принципами захисту даних.

Окремою проблемою туристичних онлайн-сервісів є розрив між формальними інструментами пошуку та реальною логікою прийняття рішення користувачем. Інтерфейс фільтрів передбачає, що користувач уже знає, які параметри йому потрібні. Натомість у реальному процесі планування подорожі людина часто починає з нечіткого наміру: відпочити біля моря, знайти недорогий напрям, підібрати комфортний варіант для сім’ї або обрати подорож на певний сезон. Такий намір потребує уточнення, пояснення та поступового перетворення на конкретні критерії. Саме тут традиційний каталог і фільтри демонструють функціональне обмеження.

Комунікаційна підтримка є способом подолання цього обмеження. У традиційній туристичній агенції її виконує менеджер, який ставить уточнювальні запитання, з’ясовує бюджет, кількість туристів, бажані дати, тип відпочинку та рівень комфорту. У вебсередовищі така підтримка може реалізовуватися через контактні форми, онлайн-чат, довідкові розділи або

чатбота. Контактна форма дає змогу залишити запит, але не забезпечує миттєвого уточнення. Довідковий розділ містить статичну інформацію, але не враховує індивідуальний контекст. Онлайн-чат із менеджером є ефективним, проте залежить від робочого часу й людського ресурсу. Чатбот частково усуває ці обмеження, оскільки може підтримувати первинну взаємодію автоматизовано [8; 9].

Чатбот у структурі туристичного вебзастосунку може виконувати роль первинного цифрового консультанта. Його функція полягає в уточненні параметрів подорожі, відповіді на типові запитання, поясненні доступних варіантів і спрямуванні користувача до релевантних пропозицій. Практична цінність такого інструменту полягає не в заміні всіх функцій менеджера, а в автоматизації початкового етапу комунікації. Якщо користувач не знає, який напрям обрати, бот може послідовно уточнити бюджет, кількість осіб, бажаний формат відпочинку, сезон, тривалість і рівень комфорту. Після цього загальний намір перетворюється на параметри, які вже можуть бути використані для пошуку або рекомендації [10].

Водночас інтеграція чатбота має власні обмеження. Якщо бот працює ізольовано від каталогу турів, його відповіді можуть залишатися загальними й не приводити користувача до конкретної дії. Такий бот створює ефект комунікації, але не забезпечує повноцінного підбору подорожі. Функціонально значущим є лише той чатбот, який пов’язаний із даними вебзастосунку, враховує параметри наявних пропозицій і може спрямувати користувача до конкретних варіантів. Отже, якість чатбота визначається не кількістю згенерованих відповідей, а його здатністю працювати як частина загального сценарію підбору туру [10].

Розвиток штучного інтелекту в туризмі посилює роль персоналізованої взаємодії. Інтелектуальні інструменти можуть використовуватися для аналізу запитів, формування рекомендацій, автоматизації підтримки, роботи з користувацькими даними та покращення сервісних процесів [5; 6; 13]. Проте їх застосування потребує критичного підходу. ШІ-компонент не повинен

створювати неперевірені або відірвані від каталогу рекомендації. Для туристичного вебзастосунку важливо, щоб відповіді чатбота були короткими, зрозумілими, прив’язаними до реального функціоналу системи й не підміняли фактичну інформацію про тури довільними порадами. Тому доцільною є така модель, у якій чатбот допомагає сформулювати запит, а остаточний вибір спирається на структуровані дані каталогу [8;10].

Критичний аналіз функціонування туристичних онлайн-сервісів дає змогу виокремити кілька системних обмежень. Каталог турів є необхідним елементом такої системи, однак його наявність не забезпечує якісного добору подорожі без зручної навігації, фільтрації та пояснення параметрів. Фільтраційний механізм ефективно працює з уже сформованими критеріями, проте не завжди підтримує користувача на етапі нечіткого запиту, коли потреба ще не перетворена на конкретні параметри пошуку. Сторінка детального перегляду надає розширену інформацію про тур, але не гарантує, що користувач зможе правильно співвіднести ці дані зі своїми індивідуальними очікуваннями. Актуальність туристичних пропозицій залежить від якості адміністративного керування, своєчасного оновлення цін, дат, описів і доступності турів. Персоналізація та чатботи мають практичну цінність лише тоді, коли вони інтегровані із загальною логікою роботи вебзастосунку й пов’язані з реальними даними каталогу.

Отже, онлайн-сервіси для підбору туристичних подорожей функціонують як комплексні вебсистеми, що поєднують каталог пропозицій, пошук, фільтрацію, сортування, детальний перегляд, оформлення заявки, користувацький профіль, адміністративне керування та комунікаційну підтримку. Їхня ефективність визначається не лише кількістю розміщених турів, а й здатністю системи допомогти користувачу перейти від загального наміру до конкретного рішення. Найбільш проблемним залишається етап первинного уточнення потреб, оскільки традиційні фільтри не завжди здатні працювати з нечіткими запитами. Саме тому перспективним напрямом удосконалення таких сервісів є поєднання структурованого каталогу з інтегрованим чатботом, який

автоматизує початкову консультацію, допомагає сформувати параметри пошуку та підвищує зручність користувацької взаємодії.

1.2 Аналіз існуючих рішень та їх функціональних можливостей

Аналіз існуючих цифрових рішень у сфері туризму дає змогу визначити, які функції вже реалізовані на ринку, які підходи використовуються для підтримки користувача під час планування подорожі та які обмеження залишаються актуальними для розроблення вебзастосунку. Для дослідження обрано чотири сервіси: Tripadvisor, Priceline, Airbnb та Viator. Вони представляють різні моделі туристичних онлайн-платформ: інформаційно-рекомендаційну, транзакційну, платформу короткострокового проживання та сервіс бронювання туристичних активностей.

Tripadvisor є туристичною платформою, основою якої є пошук інформації, аналіз відгуків, перегляд рейтингів, добір готелів, ресторанів, пам’яток, активностей і формування ідей для майбутньої подорожі. Функціональність сервісу орієнтована насамперед на підтримку користувача на етапі попереднього вибору, коли ще потрібно визначити напрям, місце проживання, заклади харчування або активності. Важливою перевагою Tripadvisor є використання великого масиву користувацького контенту: відгуків, оцінок, фотографій і рекомендацій. Завдяки цьому користувач отримує не лише офіційний опис туристичного об’єкта, а й оцінку реального досвіду інших мандрівників [15;16].

На рис. 1.3 зображено головну сторінку сервісу Tripadvisor, яка демонструє орієнтацію платформи на пошук туристичних об’єктів, перегляд категорій подорожі та використання пошукового рядка як основного інструмента взаємодії користувача із сервісом [15;16].

Окремою функціональною перевагою Tripadvisor є розвиток інструментів планування подорожі. Сервіс дає змогу створювати добірки збережених місць, організовувати ідеї для поїздки, співпрацювати з іншими користувачами та

використовувати ШІ для отримання індивідуальних рекомендацій [15;16]. ШІ-інструменти Tripadvisor є прикладом діалогового підходу, що допомагає користувачу сформулювати запит природною мовою. Це важливо для туристичної сфери, оскільки користувач часто починає планування не з точних параметрів, а з опису бажаного досвіду: сімейний відпочинок, романтична подорож, готель із певними умовами або локальні рекомендації.

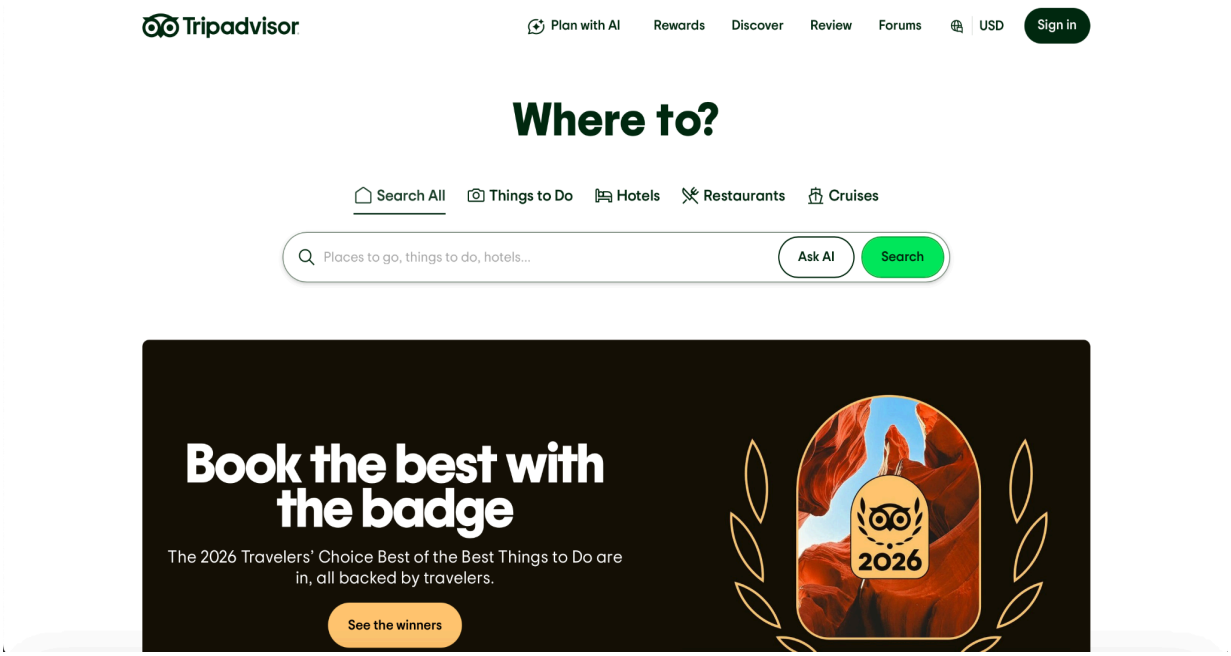

Рисунок 1.3 – Головна сторінка туристичної платформи Tripadvisor

На рис. 1.4 зображено інтерфейс ШІ-асистента Tripadvisor, який демонструє можливість формування туристичного запиту у діалоговій формі та отримання персоналізованих ідей для планування подорожі [15;16].

Перевагою Tripadvisor є поєднання соціального доказу, рекомендаційної логіки та ШІ-підтримки. Платформа допомагає користувачу оцінити туристичні об’єкти через досвід інших людей і зменшує час пошуку завдяки рекомендаційним інструментам. Водночас Tripadvisor більше орієнтований на інформаційне планування, маршрути, рекомендації та добір об’єктів для подорожі, ніж на оформлення готового туристичного пакета. Через це його

доцільно розглядати як сильний інструмент попереднього планування, а не як повноцінну систему підбору й бронювання пакетних турів [15;16].

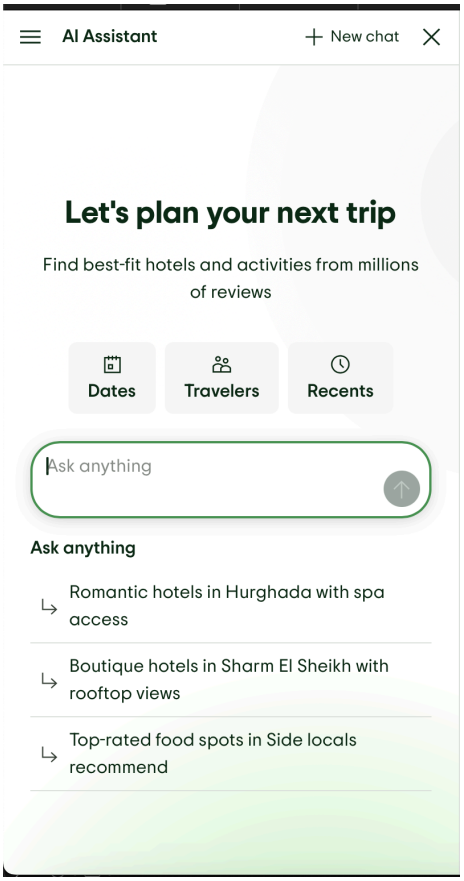

Рисунок 1.4 – Інтерфейс ШІ-асистента Tripadvisor

Priceline має іншу функціональну спрямованість. Якщо Tripadvisor переважно підтримує інформаційний та рекомендаційний етап, то Priceline орієнтований на пошук і бронювання конкретних туристичних послуг: готелів, авіаквитків, автомобілів, круїзів, пакетів і туристичних вражень. Його структура побудована навколо швидкого переходу від вибору категорії до конкретної пропозиції та бронювання. Така модель є транзакційною, оскільки головною дією користувача є не лише отримання інформації, а завершення бронювання [17].

На рис. 1.5 зображено головну сторінку сервісу Priceline, яка демонструє орієнтацію платформи на швидкий пошук і бронювання конкретних туристичних послуг: готелів, авіаквитків, пакетів, автомобілів і круїзів [17].

Важливою особливістю Priceline є ШІ-асистент Penny. Офіційні матеріали Priceline описують Penny як AI-powered travel assistant, який допомагає користувачам у процесі планування і бронювання подорожі [17]. У 2024 році сервіс представив оновлення Trip Intelligence suite, що охоплює понад 30 ШІ-функцій, спрямованих на спрощення планування і бронювання, а також розширення можливостей Penny [17]. Це свідчить про те, що штучний інтелект у Priceline використовується як частина ширшої стратегії автоматизації користувацького сценарію.

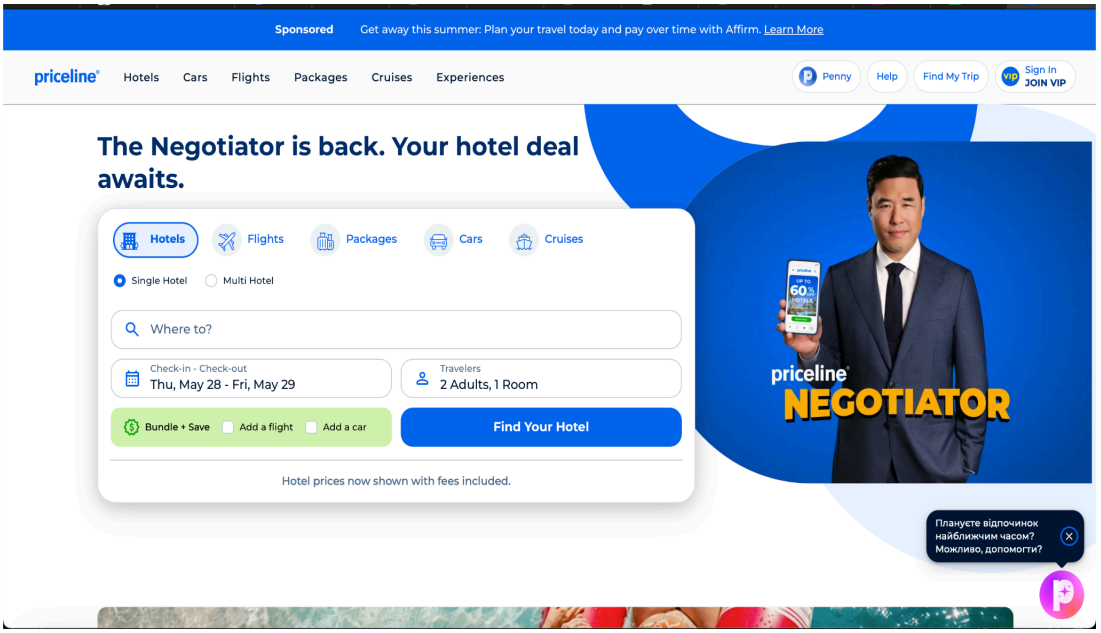

Рисунок 1.5 – Головна сторінка туристичного сервісу Priceline

На рис. 1.6 зображено інтерфейс ШІ-асистента Penny у сервісі Priceline, який підтримує користувача під час пошуку напрямів, готелів, авіарейсів, вигідних пропозицій і туристичної підтримки [17]

Функціональність Penny має практичну спрямованість. Асистент може допомагати з пошуком готелів, авіарейсів, напрямів, вигідних пропозицій і підтримкою користувача під час поїздки. На відміну від рекомендаційних ШІ-інструментів, що переважно генерують ідеї, Penny інтегрована ближче до процесу бронювання. Це означає, що користувач може не лише поставити загальне запитання, а й перейти до конкретної дії в межах сервісу. Перевагою

Priceline є поєднання ШІ-підтримки з реальними бронювальними функціями, однак якість такого сценарію залежить від того, наскільки точно намір користувача співвідноситься з наявними пропозиціями сервісу.

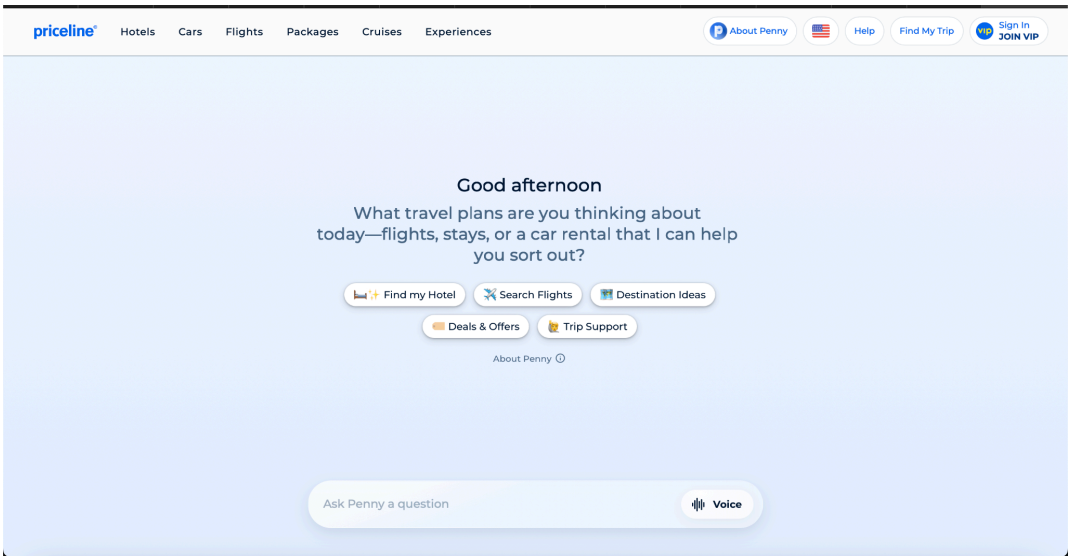

Рисунок 1.6 – Інтерфейс ШІ-асистента Penny у сервісі Priceline

Airbnb представляє модель туристичного сервісу, основою якої є короткострокова оренда житла, взаємодія між гостем і господарем, а також розширення платформи в напрямі локальних послуг і туристичних вражень. У 2025 році Airbnb представив оновлену платформу, що включає Airbnb Services, Airbnb Experiences і новий застосунок. Це свідчить про зміну функціонального фокусу: сервіс уже не обмежується наданням житла, а прагне охопити ширший контекст перебування користувача під час подорожі [18; 19].

На рис. 1.7 наведено інтерфейс сервісу Airbnb, що демонструє пошук короткострокового житла за локацією, датами, кількістю гостей і типом помешкання [18; 19].

Основною перевагою Airbnb є гнучкість вибору проживання. Користувач може шукати житло за локацією, ціною, типом помешкання, зручностями, правилами проживання, рейтингом і відгуками. На відміну від стандартизованих готельних пропозицій, Airbnb дає змогу обрати житло з індивідуальними характеристиками: квартиру, будинок, кімнату, унікальний

об’єкт або помешкання в конкретному районі міста. Розширення Airbnb Services та Airbnb Experiences посилює роль сервісу як платформи туристичного досвіду, де користувач може поєднувати проживання з додатковими послугами й активностями [18; 19].

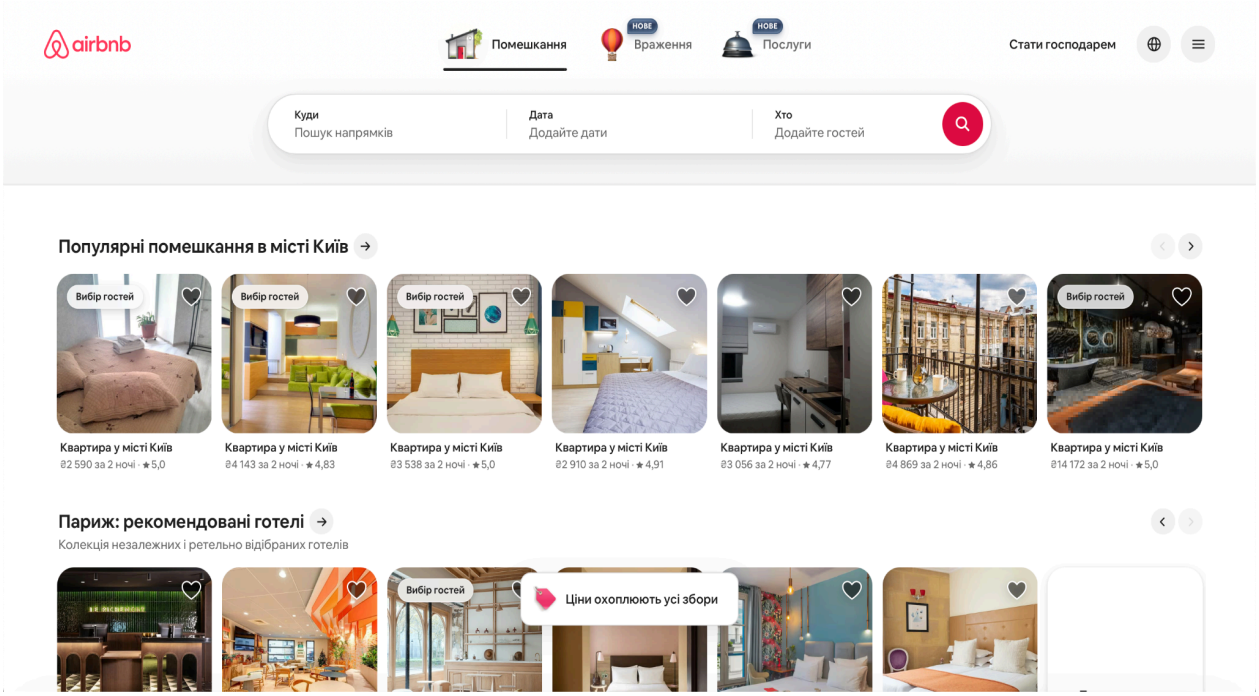

Рисунок 1.7 – Інтерфейс пошуку житла в сервісі Airbnb

Водночас Airbnb не є системою підбору пакетних туристичних пропозицій. Користувач може самостійно зібрати подорож із кількох елементів, але для цього потрібні час, досвід і здатність оцінити ризики. Тому Airbnb є ефективним сервісом для самостійних мандрівників, які хочуть індивідуалізувати проживання та досвід перебування, однак його модель не повністю відповідає задачі комплексного добору туру.

Viator є спеціалізованим сервісом для пошуку і бронювання екскурсій, турів, квитків, активностей і туристичних вражень. Його функціональна модель орієнтована не на проживання чи транспорт, а на змістове наповнення подорожі. Користувач, який уже обрав напрям або перебуває в певному місті, може знайти екскурсії, оглядові маршрути, гастрономічні тури, культурні програми,

пригодницькі активності або квитки на туристичні об’єкти. На офіційному сайті Viator зазначено можливість перегляду і бронювання значної кількості туристичних активностей, а також наявність безкоштовного скасування й гнучких варіантів оплати [20].

На рис. 1.8 зображено головну сторінку сервісу Viator, яка відображає спеціалізацію платформи на пошуку та бронюванні екскурсій, активностей і туристичних вражень [20].

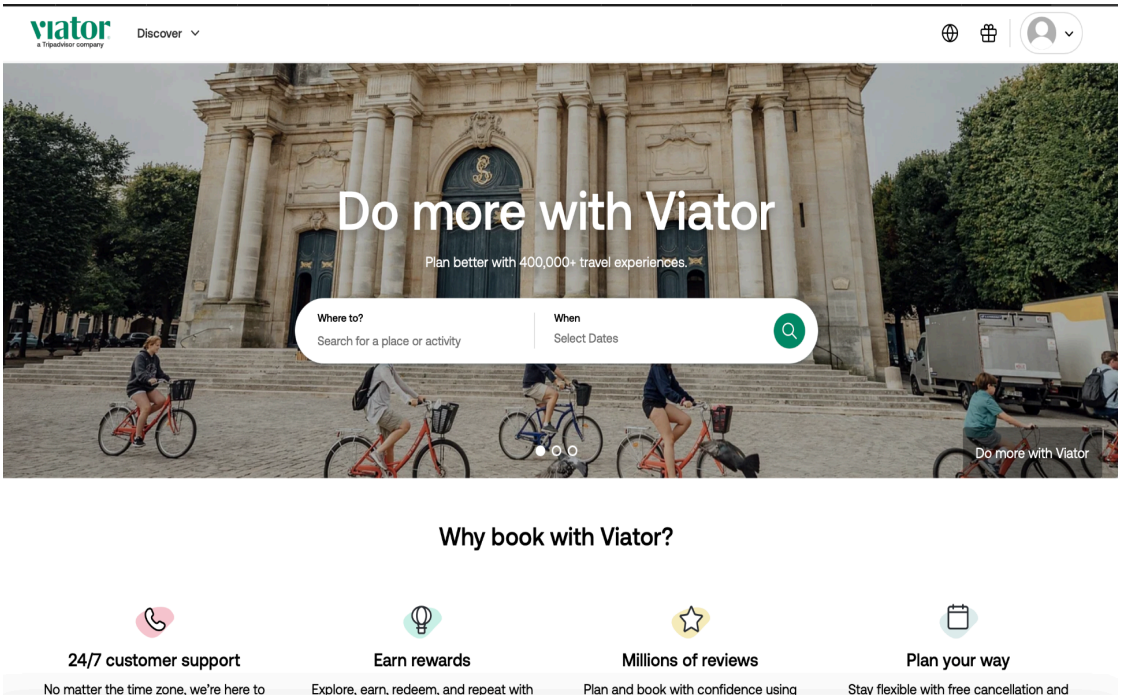

Рисунок 1.8 – Головна сторінка сервісу бронювання туристичних активностей Viator

Перевагою Viator є вузька спеціалізація. Сервіс акумулює багато пропозицій у конкретній категорії та дозволяє порівнювати їх за описом, ціною, тривалістю, рейтингами й умовами скасування. Мобільний застосунок Viator також підтримує перегляд турів і активностей до та під час подорожі, керування бронюваннями й доступ до квитків в офлайн-режимі . Це робить сервіс корисним саме на етапі практичного використання туристичних послуг у процесі поїздки. Обмеження Viator полягає в тому, що він не вирішує питання

первинного вибору напряму, проживання, транспорту та загального бюджету подорожі [20].

Порівняння Tripadvisor, Priceline, Airbnb і Viator дає змогу визначити чотири різні підходи до цифрового обслуговування туриста. Tripadvisor орієнтований на інформацію, відгуки, рейтинги, маршрути й ШІ-підтримку планування. Priceline зосереджується на бронюванні та використовує ШІ-асистента для спрощення пошуку, порівняння й оформлення туристичних послуг. Airbnb забезпечує гнучкий вибір проживання, локальні послуги та враження, але залишає користувачу значну частину самостійного планування. Viator спеціалізується на екскурсіях і активностях, доповнюючи вже сформовану подорож.

Найбільш показовими для цієї роботи є Tripadvisor і Priceline, оскільки саме в цих сервісах діалогові ШІ-інструменти використовуються як засіб спрощення взаємодії з користувачем. Наявність ШІ-асистентів демонструє ефективніший підхід до роботи з нечітким користувацьким запитом порівняно з моделлю, у якій користувач самостійно проходить усі етапи пошуку через фільтри, категорії та списки пропозицій. Діалоговий інтерфейс дає змогу сформулювати потребу природною мовою, уточнити параметри подорожі та швидше перейти до релевантних варіантів [5; 12].

Перевага Tripadvisor полягає в тому, що ШІ-інструмент може працювати з великим масивом відгуків, рейтингів і рекомендацій, допомагаючи користувачу оцінити туристичні об’єкти з позиції реального досвіду. Перевага Priceline полягає в інтеграції ШІ-асистента Penny ближче до практичного сценарію бронювання. Отже, Tripadvisor демонструє цінність ШІ для планування й рекомендацій, а Priceline – цінність ШІ для переходу від запиту до конкретної дії.

Розглянута тенденція свідчить про доцільність поєднання традиційних механізмів туристичного онлайн-сервісу з діалоговими інструментами підтримки користувача. Каталог, фільтри та сторінки детального перегляду залишаються необхідними, оскільки забезпечують структуроване подання

туристичних пропозицій. Водночас чатбот підвищує зручність взаємодії, надаючи користувачу можливість не лише переглядати готові варіанти, а й уточнювати свій запит у діалоговій формі.

Отже, аналіз існуючих рішень підтверджує доцільність розроблення вебзастосунку, який поєднує каталог туристичних пропозицій, пошук, фільтрацію, детальний перегляд, заявку на бронювання та інтегрований чатбот. Досвід Tripadvisor і Priceline показує, що ШІ-асистенти можуть підвищувати ефективність туристичного сервісу, оскільки допомагають користувачу формулювати запит природною мовою, уточнювати критерії вибору та швидше переходити до практичного результату. Саме тому чатбот у вебзастосунку доцільно розглядати не як декоративний елемент, а як функціональний модуль первинної консультації, пов’язаний із каталогом туристичних пропозицій і механізмом оформлення заявки.

1.3. Постановка задачі на основі аналізу предметної галузі

Аналіз туристичних онлайн-сервісів показав, що сучасні вебплатформи забезпечують основні етапи роботи з туристичними пропозиціями: перегляд каталогу, пошук, фільтрацію, сортування, відкриття детальної інформації, оформлення заявки або бронювання. Така структура є базовою для цифрового підбору подорожей, оскільки дає змогу користувачу самостійно працювати з великим обсягом пропозицій і поступово звужувати вибір відповідно до власних параметрів. Водночас ефективність сервісу залежить не лише від кількості розміщених турів, а й від того, наскільки швидко користувач може перейти від загального бажання подорожі до конкретного варіанта [3; 14].

Проблема полягає в тому, що на початковому етапі користувач не завжди має чітко сформований запит. Він може орієнтуватися лише на загальний формат відпочинку, приблизний бюджет, сезон або склад туристів. Наприклад, користувач може шукати недорогий відпочинок біля моря, спокійну подорож

для двох, сімейний тур або теплий напрям на певний період. Для класичних фільтрів такого формулювання недостатньо, оскільки система зазвичай очікує конкретні параметри: країну, дати, ціну, тривалість, кількість осіб, тип харчування або рівень готелю. Через це виникає розрив між природною логікою користувацького запиту та формальною структурою пошуку [8; 9].

Особливе значення має використання чатботів та ШІ-асистентів. Діалоговий інструмент дає змогу формулювати запит природною мовою, а не лише обирати значення у фільтрах. Для туристичної сфери це важливо, оскільки подорож обирається за сукупністю критеріїв: бюджетом, датами, складом туристів, форматом відпочинку, рівнем комфорту, напрямом і додатковими очікуваннями. Чатбот може поступово уточнювати ці параметри та допомагати користувачу перетворити нечіткий намір на конкретні умови пошуку [8; 12].

На основі проведеного аналізу постає задача створення вебзастосунку для підбору туристичних подорожей, який поєднує традиційні механізми туристичного онлайн-сервісу з діалоговою підтримкою користувача. Такий застосунок має забезпечувати перегляд туристичних пропозицій, пошук і фільтрацію, відкриття детальної інформації про тур, оформлення заявки та взаємодію з чатботом. Основна ідея полягає в тому, щоб користувач міг не лише самостійно переглядати каталог, а й отримувати допомогу в уточненні параметрів подорожі.

Чатбот у межах поставленої задачі має виконувати роль первинного цифрового консультанта. Його призначення полягає в уточненні бажаного напряму, бюджету, кількості туристів, тривалості, дат, типу відпочинку та рівня комфорту. Після цього користувач має бути спрямований до релевантних пропозицій у каталозі або до подальшого перегляду турів. Важливо, щоб чатбот був пов’язаний із логікою вебзастосунку, а не працював окремо від основного функціоналу [8; 10].

Окремим аспектом задачі є підтримка актуальності туристичних пропозицій. Інформація про тури швидко змінюється: оновлюються ціни, дати, описи, фотографії, доступність місць та умови бронювання. Тому вебзастосунок

має передбачати адміністративну частину, через яку можна керувати турами та заявками. Такий модуль забезпечує можливість додавання, редагування й видалення пропозицій, а також опрацювання звернень користувачів [1; 12].

З технічного погляду поставлена задача передбачає створення цілісної вебсистеми, у якій клієнтська частина відповідає за взаємодію користувача з інтерфейсом, серверна частина – за обробку запитів і бізнес-логіку, база даних – за збереження інформації про тури, користувачів і заявки, а чатбот – за діалогову підтримку. Такий підхід дає змогу організувати роботу системи як послідовний користувацький сценарій: від уточнення запиту або перегляду каталогу до вибору конкретної пропозиції та оформлення заявки.

Отже, на основі аналізу предметної галузі сформульовано задачу розроблення вебзастосунку для підбору туристичних подорожей з інтегрованим чатботом. Її сутність полягає в поєднанні каталогу туристичних пропозицій, механізмів пошуку та фільтрації, детального перегляду туру, оформлення заявки, адміністративного керування та діалогової підтримки користувача. Такий підхід спрямований на розв’язання проблеми переходу від нечіткого туристичного наміру до конкретних параметрів пошуку. Деталізацію функціональних і нефункціональних вимог до системи доцільно подати в наступному розділі під час опису проєктування та реалізації вебзастосунку.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

2.1 Визначення вимог до вебзастосунку

Визначення вимог до вебзастосунку є початковим етапом практичного проєктування, оскільки саме на цьому рівні встановлюється, які задачі має виконувати система, які ролі користувачів вона повинна підтримувати та які обмеження необхідно врахувати під час реалізації. Система має поєднувати функції класичного туристичного каталогу, механізмів пошуку та фільтрації, оформлення заявки на бронювання, адміністративного керування та діалогової підтримки користувача.

Основною задачею вебзастосунку є надання користувачу зручного інструменту для пошуку й добору туристичних подорожей. Для цього система повинна забезпечувати перегляд каталогу турів, відкриття детальної сторінки конкретної пропозиції, фільтрацію за основними параметрами, додавання турів в обране, створення заявки на бронювання та отримання рекомендацій через інтегрований ШІ-чатбот. Такий підхід дає змогу реалізувати не лише інформаційний перегляд туристичних пропозицій, а й повноцінний користувацький сценарій: від первинного інтересу до конкретної заявки.

У системі передбачено три основні ролі: гість, зареєстрований користувач і адміністратор. Гість повинен мати доступ до відкритої частини вебзастосунку: головної сторінки, каталогу турів, сторінок детального перегляду та чатбота. Це забезпечує можливість первинного ознайомлення із сервісом без обов’язкової реєстрації. Зареєстрований користувач отримує розширений функціонал: створення заявки на бронювання, додавання турів в обране, перегляд профілю, перегляд власних заявок і скасування заявок, які ще не були опрацьовані адміністратором. Адміністратор виконує функції керування системою: додає, редагує та видаляє туристичні пропозиції, переглядає заявки, змінює їхні статуси та аналізує загальну статистику.

Для наочного подання основних ролей системи та доступних їм функцій побудовано діаграму варіантів використання, зображено на рис. 2.1.

Рисунок 2.1 – Діаграма варіантів використання вебзастосунку для підбору туристичних подорожей

До функціональних вимог користувацької частини належить реалізація головної сторінки, яка має виконувати навігаційну й презентаційну функцію. Вона повинна містити основний блок із пошуком, елементи переходу до каталогу, популярні напрями або тематичні блоки, а також доступ до чатбота. Головна сторінка має швидко пояснювати призначення системи й спрямовувати користувача до основної дії – пошуку або добору подорожі.

Каталог турів є центральною частиною вебзастосунку. Він повинен відображати туристичні пропозиції у вигляді карток із ключовою інформацією: фото, тип туру, рейтинг, назва, країна або місто, готель, короткий опис,

тривалість, кількість гостей, транспорт, тип харчування та ціна. Така структура картки дає змогу користувачу швидко оцінити пропозицію без відкриття повної сторінки. Крім того, кожна картка має містити можливість переходу до детального перегляду та додавання туру в обране.

Пошук і фільтрація повинні забезпечувати добір турів за основними параметрами, що впливають на вибір подорожі. До таких параметрів належать країна, місто, тип подорожі, транспорт, тип харчування, тривалість, кількість осіб, рівень готелю, доступність, мінімальна та максимальна ціна, а також текстовий пошук. Наявність фільтрації є необхідною, оскільки каталог може містити значну кількість пропозицій, а користувач повинен мати змогу швидко звузити перелік варіантів відповідно до власних потреб. Сортування за ціною, популярністю або іншими критеріями додатково підвищує зручність роботи з каталогом.

Сторінка детального перегляду туру повинна надавати розширену інформацію про конкретну туристичну пропозицію. На ній доцільно відображати назву туру, країну, місто, тип подорожі, доступність, ціну, рейтинг, опис, тривалість, кількість гостей, транспорт, харчування, дати, інформацію про готель, фото, включені та невключені послуги, активності й відгуки. Така сторінка виконує функцію поглибленого ознайомлення з пропозицією та допомагає користувачу прийняти рішення перед оформленням заявки. Також на цій сторінці має бути передбачена кнопка переходу до бронювання або звернення до AI-асистента для уточнення питань.

Окремою функціональною вимогою є можливість створення заявки на бронювання. Система повинна дозволяти авторизованому користувачу обрати тур, вказати ім’я, електронну адресу, номер телефону, кількість мандрівників, бажану дату та додатковий коментар. Перед створенням заявки серверна частина має перевіряти коректність даних: наявність туру, його доступність, відповідність кількості туристів допустимій місткості та входження бажаної дати в доступний період. Після успішного створення заявка повинна зберігатися

в базі даних зі статусом pending, тобто очікувати подальшого опрацювання адміністратором.

Для роботи із заявками необхідно передбачити систему статусів. Нова заявка має отримувати статус pending; після перевірки адміністратор може підтвердити її, відхилити або залишити на опрацюванні. Користувач повинен мати можливість скасувати лише ту заявку, яка ще не була підтверджена або відхилена. Такий підхід дає змогу відобразити реальну логіку обробки бронювання: користувач створює запит, а остаточне рішення щодо його підтвердження виконується адміністратором.

Послідовність основних дій під час створення та опрацювання заявки на бронювання туру зображено на рис. 2.2.

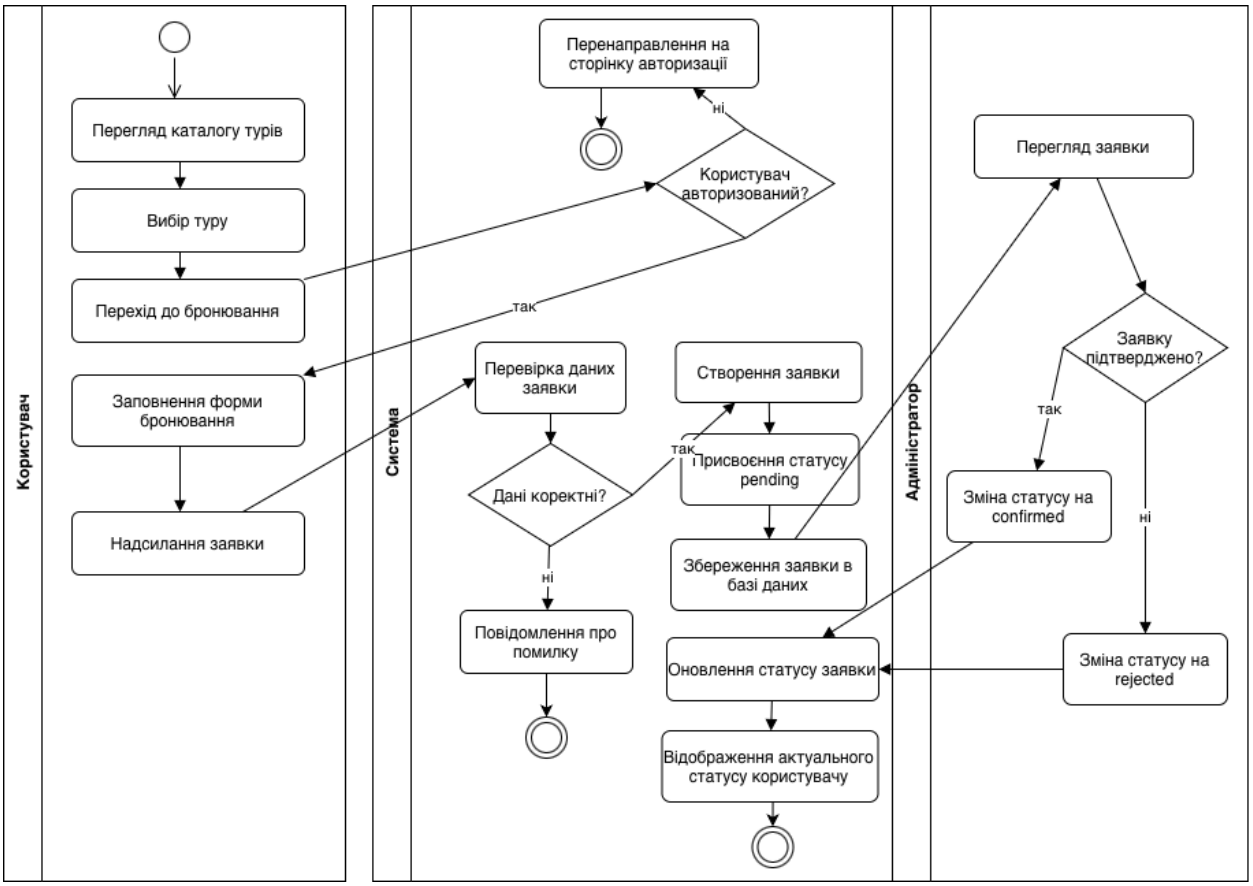

Рисунок 2.2 – Діаграма діяльності процесу створення заявки на бронювання туру

Функціонал профілю користувача має забезпечувати перегляд персональної інформації, власних заявок, обраних турів і базової статистики взаємодії із системою. Наявність профілю підвищує зручність повторного використання вебзастосунку, оскільки користувач може швидко повернутися до збережених пропозицій або перевірити стан своїх бронювань. Також профіль створює основу для подальшої персоналізації системи, наприклад для врахування попередніх дій користувача під час добору турів.

Адміністративна частина повинна забезпечувати керування основними даними системи. До функцій адміністратора належать перегляд статистики, додавання нових турів, редагування наявних пропозицій, видалення неактуальних турів, перегляд заявок, фільтрація заявок за статусом або іншими параметрами, а також підтвердження чи відхилення бронювань. Такі можливості є необхідними, оскільки туристичні пропозиції мають динамічний характер: змінюються ціни, дати, доступність місць, опис готелю, фотографії та умови подорожі. Без адміністративного модуля система не могла б підтримувати актуальність каталогу.

Важливою функціональною вимогою є реалізація чатбота, який повинен допомагати користувачу під час підбору подорожі. На відміну від звичайного інформаційного чату, такий бот має працювати як первинний цифровий консультант. Його завдання полягає в аналізі повідомлення користувача, уточненні параметрів подорожі, визначенні наміру, пошуку відповідних турів у базі даних і формуванні короткої відповіді з рекомендаціями. Чатбот має враховувати країну, місто, бюджет, кількість людей, тривалість, тип відпочинку, транспорт, харчування, сезон, рівень готелю та бажані дати.

Особливе значення має те, що чатбот повинен працювати з реальними туристичними пропозиціями з бази даних, а не створювати довільні варіанти. У межах системи ШІ-модуль аналізує природномовний запит користувача, після чого окремий механізм рекомендацій виконує ранжування турів за відповідністю параметрам. Такий підхід можна визначити як поєднання AI-аналізу повідомлення та rule-based ранжування результатів. Це підвищує

практичну цінність чатбота, оскільки користувач отримує рекомендації, пов’язані з фактичним каталогом, а не загальні поради.

Послідовність взаємодії користувача, чатбота, серверної частини, ШІ-сервісу та бази даних під час підбору туристичної пропозиції зображено на рис. 2.3.

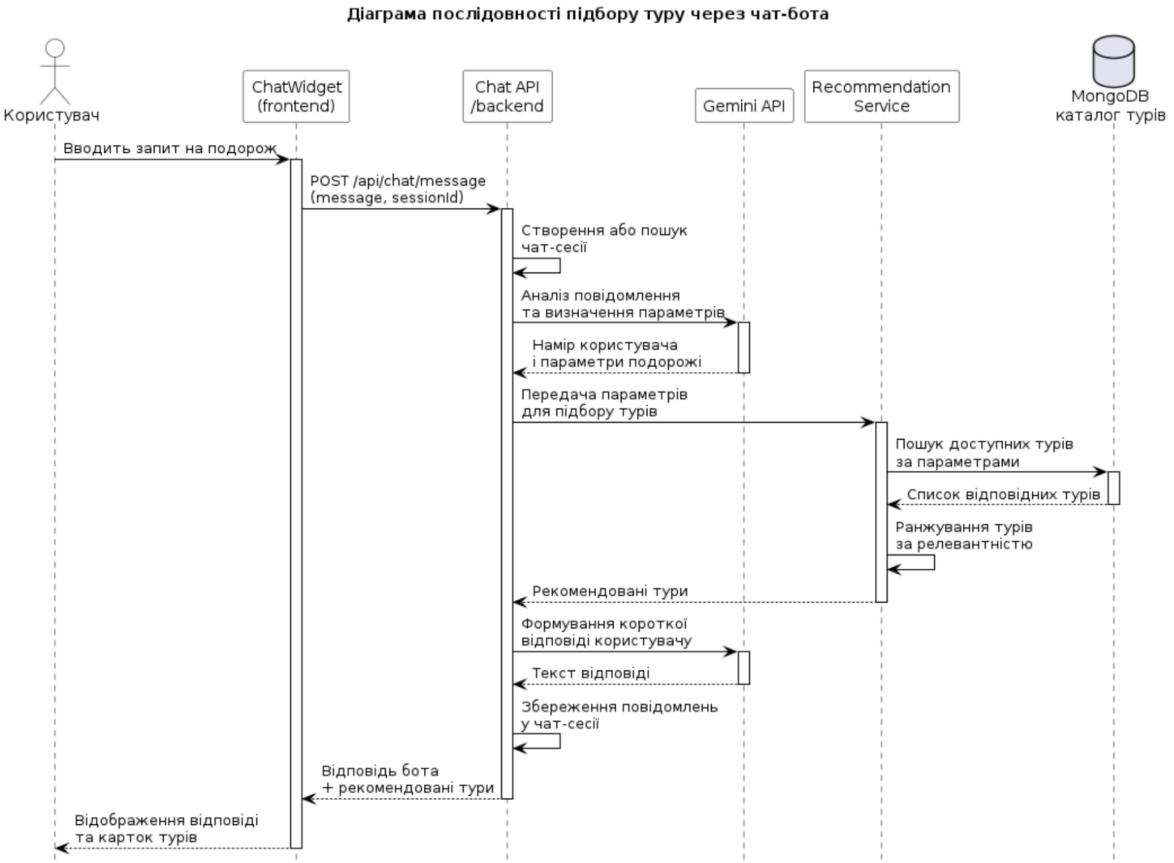

Рисунок 2.3 – Діаграма послідовності підбору туру через чатбота

Для стабільної роботи чатбота необхідно передбачити fallback-логіку. Якщо зовнішній ШІ-сервіс недоступний, перевищено ліміт запитів, виникла помилка API або мережева проблема, система не повинна повністю припиняти взаємодію з користувачем. У такому випадку чатбот має переходити до резервного сценарного режиму, у межах якого базово розпізнаються бюджет, кількість людей, тривалість, тип туру, країна або місто, а користувачу все одно можуть бути запропоновані варіанти з каталогу. Така вимога підвищує

відмовостійкість системи та забезпечує безперервність користувацького сценарію.

До вимог авторизації та безпеки належить реєстрація користувача, вхід у систему, захист пароля, використання JWT-токенів і розмежування доступу за ролями. Паролі мають зберігатися не у відкритому вигляді, а у вигляді хешу. Під час реєстрації доцільно перевіряти складність пароля, зокрема мінімальну довжину, наявність великої літери, цифри та спеціального символу. Захищені маршрути повинні бути доступні лише авторизованим користувачам, а адміністративні функції – лише користувачам із роллю адміністратора. Такий підхід дає змогу відокремити відкриту частину сервісу від дій, які пов’язані з персональними даними, заявками та керуванням системою.

До нефункціональних вимог вебзастосунку належать зручність інтерфейсу, адаптивність, стабільність API, підтримуваність структури проєкту, розширюваність і коректна обробка помилок. Інтерфейс має бути зрозумілим для користувача, а основні дії – доступними без зайвого ускладнення. Адаптивність є важливою, оскільки туристичні сервіси часто використовуються з мобільних пристроїв. Тому сторінки каталогу, детального перегляду, профілю, адміністративної панелі та чатвіджет мають коректно відображатися на різних розмірах екрана.

Стабільність серверної частини забезпечується централізованою обробкою помилок, перевіркою неіснуючих маршрутів, базовою валідацією запитів і контролем навантаження. Підтримуваність досягається через модульну структуру backend-частини, де маршрути, контролери, моделі, middleware та сервіси розділені за призначенням. Розширюваність системи забезпечується тим, що основні функціональні модулі – тури, бронювання, обране, відгуки, чат і адміністративна частина – реалізовані окремо. Це дає змогу в майбутньому додавати нові можливості без повної перебудови архітектури.

Отже, вимоги до вебзастосунку для підбору туристичних подорожей охоплюють користувацькі, адміністративні, безпекові та технічні аспекти. Система повинна забезпечувати перегляд і добір турів, роботу з обраним,

оформлення заявок, керування бронюваннями, адміністрування туристичних пропозицій, збереження відгуків і взаємодію з ШІ-чатботом. Ключовою особливістю є поєднання класичного каталогу з інтелектуальним підбором: чатбот аналізує природномовний запит користувача, а механізм рекомендацій добирає реальні тури з бази даних. Саме така структура дає змогу розглядати вебзастосунок не лише як електронний каталог, а як інтерактивну систему підтримки вибору туристичної подорожі.

2.2 Вибір архітектури та технологічного стеку

Вебзастосунок для підбору туристичних подорожей має підтримувати роботу з каталогом турів, авторизацію користувачів, бронювання, адміністративне керування, збереження даних, обробку відгуків, обрані тури та інтеграцію чатбота. З огляду на ці задачі обрано клієнт-серверну архітектуру з окремою клієнтською та серверною частинами.

Архітектура системи передбачає кілька основних рівнів: користувацький рівень, клієнтський рівень, серверний рівень, рівень даних і рівень зовнішніх сервісів. Користувацький рівень охоплює взаємодію гостя, зареєстрованого користувача та адміністратора із системою. Клієнтський рівень відповідає за відображення інтерфейсу, маршрутизацію між сторінками, роботу форм, каталогів, профілю та адміністративної панелі. Серверний рівень обробляє запити, виконує бізнес-логіку, перевіряє права доступу та забезпечує зв’язок із базою даних. Рівень даних відповідає за збереження користувачів, турів, заявок, відгуків, обраних пропозицій і чат-сесій. Рівень зовнішніх сервісів використовується для підключення сторонніх інструментів, зокрема ШІ-сервісу для роботи чатбота.

Взаємодія між рівнями відбувається послідовно. Користувач виконує дію в інтерфейсі, наприклад відкриває каталог, застосовує фільтр, переходить до сторінки туру або створює заявку. Клієнтська частина формує HTTP-запит і

надсилає його до серверного REST API. Сервер приймає запит, перевіряє його коректність, за потреби перевіряє авторизацію, звертається до відповідного контролера та виконує операцію з базою даних. Після цього backend повертає результат клієнтській частині, а frontend оновлює відображення для користувача. Такий потік даних забезпечує контрольовану взаємодію між інтерфейсом, серверною логікою та збереженими даними.

Контейнерну структуру вебзастосунку зображено на рис. 2.4. Діаграма відображає основних користувачів системи, клієнтський інтерфейс, серверний API, сервіси чатбота та рекомендацій, базу даних і зовнішній ШІ-сервіс.

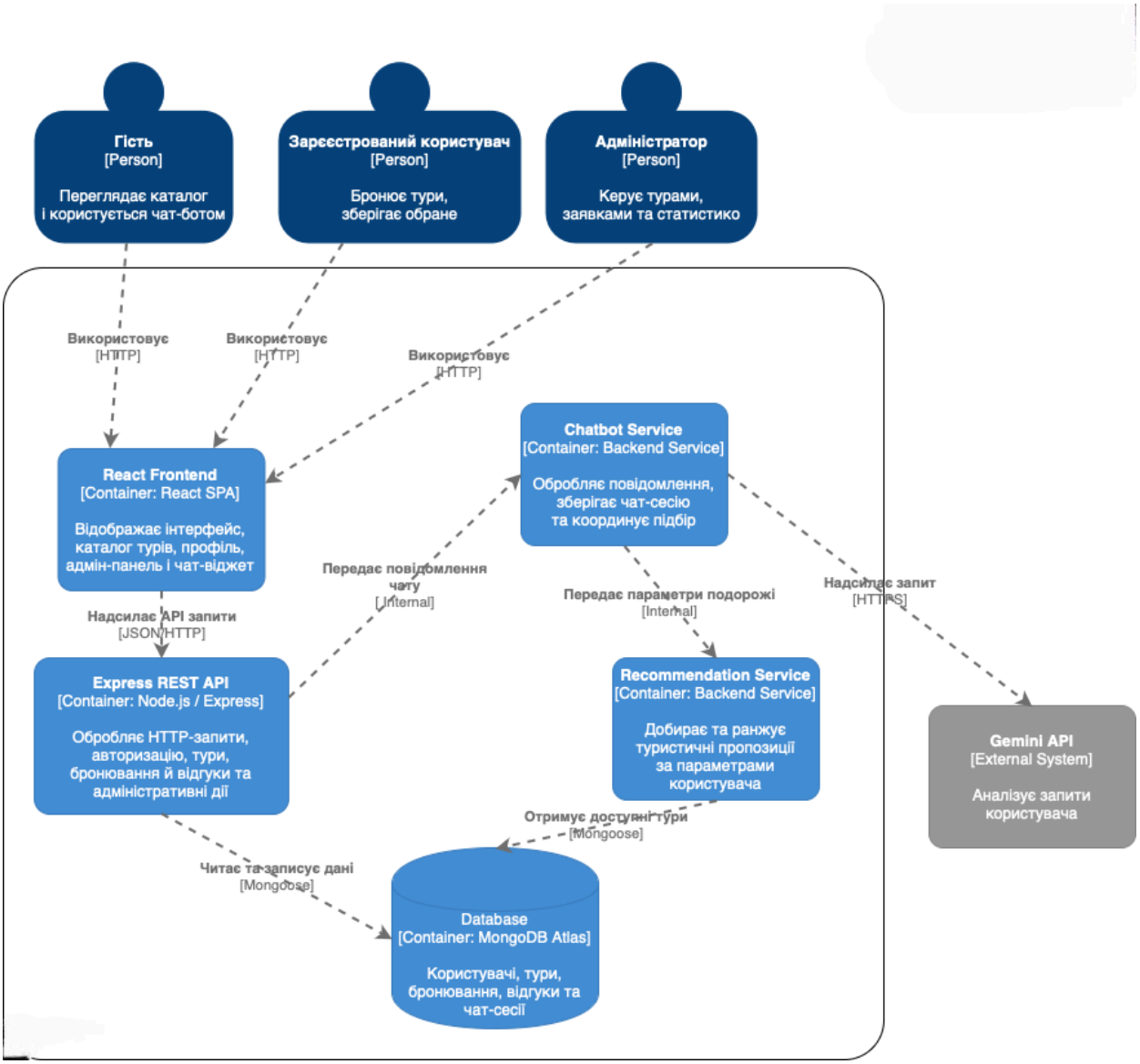

Рисунок 2.4 – Контейнерна C4-діаграма вебзастосунку для підбору туристичних подорожей

Клієнт-серверна архітектура дає змогу розмежувати відповідальність між інтерфейсом користувача та серверною логікою. Клієнтська частина відповідає за відображення сторінок, форм, каталогу, профілю, адміністративної панелі та чат-віджета. Серверна частина обробляє запити, перевіряє авторизацію, працює з базою даних, виконує операції над турами, бронюваннями, відгуками та забезпечує взаємодію з ШІ-сервісом. Такий поділ робить структуру застосунку зрозумілою та спрощує подальшу підтримку, оскільки зміни в інтерфейсі не потребують безпосереднього втручання в логіку роботи сервера.

Backend-частина побудована як монолітний серверний застосунок із внутрішнім модульним поділом. Такий підхід є обґрунтованим для цієї роботи, оскільки всі основні функції системи тісно пов’язані між собою: каталог турів, користувачі, бронювання, відгуки, адміністративна панель і чатбот працюють у межах єдиного програмного середовища. Використання мікросервісної архітектури для такого проєкту було б надмірним, оскільки потребувало б складнішої інфраструктури, окремого розгортання сервісів і додаткової взаємодії між ними. Монолітна backend-архітектура дозволяє реалізувати систему простіше, але водночас зберегти впорядкованість завдяки поділу коду на маршрути, контролери, моделі, middleware та сервіси.

Клієнтська частина спроєктована як SPA-застосунок. Односторінкова архітектура забезпечує взаємодію користувача зі сторінками без повного перезавантаження браузера. Для туристичного сервісу це важливо, оскільки користувач може переходити між головною сторінкою, каталогом, деталями туру, формою бронювання, профілем, обраними пропозиціями та чатботом у межах єдиного інтерфейсу. SPA-підхід підвищує плавність навігації, скорочує відчуття розриву між сторінками та дає змогу повторно використовувати компоненти інтерфейсу [21].

Основною технологією для побудови клієнтської частини є React. Його компонентна модель добре відповідає структурі туристичного вебзастосунку, у якому багато елементів повторюються на різних сторінках: картки турів, кнопки, форми, блоки фільтрації, елементи навігації, повідомлення чатбота та

адміністративні таблиці. Компонентний підхід спрощує підтримку інтерфейсу, оскільки окремий елемент можна змінити в одному місці й повторно використовувати в різних частинах системи [21].

Маршрутизація між сторінками реалізується за допомогою React Router. Ця бібліотека забезпечує переходи між головною сторінкою, каталогом турів, сторінкою детального перегляду, бронюванням, профілем, сторінками авторизації та адміністративною панеллю. Клієнтська маршрутизація відповідає SPA-підходу, оскільки дає змогу змінювати вміст інтерфейсу без повного перезавантаження сторінки.

Обмін даними між frontend і backend виконується через Axios. Цей інструмент використовується для HTTP-запитів до REST API: отримання каталогу турів, передавання параметрів фільтрації, створення заявок, роботи з профілем користувача, обраними турами та повідомленнями чатбота. Централізоване налаштування Axios є доцільним, оскільки дає змогу не дублювати параметри запитів і автоматично додавати токен авторизації до захищених звернень.

Візуальне оформлення інтерфейсу реалізується за допомогою Tailwind CSS. Цей інструмент дає змогу створювати адаптивний інтерфейс через utility-класи, не перевантажуючи проєкт великою кількістю окремих CSS-файлів. Для туристичного вебзастосунку адаптивність має суттєве значення, оскільки користувачі можуть переглядати тури як із комп’ютера, так і зі смартфона. Tailwind CSS дозволяє налаштовувати вигляд сторінок для різних розмірів екрана та підтримувати єдиний стиль інтерфейсу.

Середовище розроблення frontend-частини організоване за допомогою Vite. Цей інструмент забезпечує швидкий запуск проєкту, оперативне оновлення змін під час розроблення та оптимізовану збірку. Для React-застосунку це спрощує перевірку інтерфейсних змін і прискорює процес розроблення.

Серверна частина реалізується на Node.js. Використання Node.js дає змогу застосовувати JavaScript не лише для побудови інтерфейсу, а й для реалізації серверної логіки. Це означає, що одна мова програмування використовується на

двох основних рівнях системи: клієнтському та серверному. Такий підхід спрощує розроблення, оскільки структура даних, принципи роботи з об’єктами, асинхронні операції та загальна логіка обробки запитів залишаються технологічно узгодженими. Для цього проєкту це особливо доцільно, оскільки frontend передає на сервер дані про тури, користувачів, бронювання, фільтри та повідомлення чатбота у форматі, який природно обробляється в JavaScript-середовищі. Крім того, Node.js добре підходить для застосунків, що активно працюють з HTTP-запитами, REST API, базою даних і зовнішніми сервісами, оскільки підтримує неблокувальну модель виконання операцій [22].

Побудова серверного API виконується з використанням Express.js. Цей фреймворк забезпечує організацію REST API, маршрутизацію, middleware, обробку помилок і взаємодію з функціональними частинами системи. Express.js не нав’язує надмірно складної структури, але дає достатню гнучкість для реалізації авторизації, каталогу турів, бронювань, адміністративних операцій, відгуків, обраних турів і чатбота [23].

Як основне сховище даних використовується MongoDB Atlas. Документна модель MongoDB добре відповідає предметній області туристичного сервісу, оскільки туристична пропозиція має багато різнорідних характеристик: назву, країну, місто, опис, готель, фотографії, дати, тривалість, послуги, транспорт, тип харчування, рейтинг і доступність. У реляційній базі такі дані потребували б складнішої структури таблиць і зв’язків, тоді як документна модель дозволяє зберігати складні об’єкти у більш гнучкому форматі. Хмарний формат MongoDB Atlas додатково спрощує підключення до бази даних і не потребує локального адміністрування сервера бази [24].

Робота з MongoDB здійснюється через Mongoose. Він дає змогу описувати структуру документів за допомогою схем, визначати типи полів, задавати зв’язки між сутностями та виконувати базову валідацію. Для системи, де зберігаються користувачі, тури, бронювання, відгуки й чатсесії, такий підхід є зручним, оскільки структура даних залишається контрольованою навіть у межах документної бази [25].

Авторизація користувачів реалізується на основі JWT. Після входу в систему користувач отримує токен, який надалі передається під час звернення до захищених маршрутів. JWT добре підходить для клієнт-серверної архітектури, оскільки сервер може перевірити права користувача без збереження окремої сесії в оперативній пам’яті. Це важливо для профілю, бронювання та адміністративної панелі, де доступ має надаватися лише авторизованим користувачам або адміністраторам [26].

Захист паролів забезпечується за допомогою bcryptjs. Пароль перед збереженням у базі даних перетворюється на хеш, тому він не зберігається у відкритому вигляді. У поєднанні з JWT це дає змогу реалізувати базовий безпечний механізм реєстрації, входу та розмежування доступу за ролями.

Інтеграція чатбота здійснюється через Gemini API з використанням пакета @google/generative-ai. Залучення ШІ-сервісу обґрунтоване особливістю туристичного запиту: користувач часто описує бажаний відпочинок природною мовою, а не одразу задає точні параметри фільтрації. ШІ-модуль допомагає інтерпретувати таке повідомлення, виокремити важливі параметри подорожі та сформувати відповідь у діалоговій формі [27].

У межах обраної архітектури ШІ-сервіс не є самостійним джерелом туристичних пропозицій. Його роль полягає в аналізі повідомлення користувача та підтримці діалогу, тоді як добір варіантів має спиратися на фактичні дані з каталогу. Це принципово важливо, оскільки чатбот не повинен створювати вигадані тури, ціни або дати. Поєднання ШІ-модуля зі структурованими даними бази дає змогу зберегти достовірність рекомендацій і водночас забезпечити гнучкість природномовної взаємодії.

До зовнішніх залежностей системи належать бібліотеки та сервіси, які забезпечують окремі частини функціональності. На клієнтському рівні використовуються React, React Router, Axios, Tailwind CSS і Vite. На серверному рівні застосовуються Node.js, Express.js, Mongoose, JWT, bcryptjs, cors, dotenv, express-rate-limit і пакет @google/generative-ai. На рівні даних використовується MongoDB Atlas, а для ШІ-взаємодії – Gemini API. Такий набір залежностей

забезпечує реалізацію інтерфейсу, API, авторизації, роботи з базою даних, обмеження запитів, конфігурації середовища та інтелектуальної підтримки користувача.

Додаткові серверні інструменти забезпечують коректну роботу застосунку. cors використовується для налаштування доступу клієнтської частини до серверного API. dotenv дає змогу зберігати конфігураційні параметри, зокрема підключення до бази даних, JWT-секрет і ключ ШІ-сервісу, поза основним кодом. express-rate-limit використовується для обмеження кількості запитів до API, що зменшує ризик надмірного навантаження на сервер.

Обрана архітектура та технологічний стек відповідають функціональним задачам системи. React, React Router, Axios, Tailwind CSS і Vite забезпечують створення інтерактивного та адаптивного frontend-інтерфейсу. Node.js, Express.js, MongoDB Atlas і Mongoose формують серверну та базову частину вебзастосунку. JWT і bcryptjs забезпечують авторизацію й захист паролів. Gemini API додає можливість діалогової підтримки користувача під час підбору подорожі.

Отже, для реалізації вебзастосунку обрано клієнт-серверну архітектуру з SPA-frontend, монолітним backend із модульною внутрішньою структурою, документною базою даних і інтегрованим ШІ-модулем. Такий підхід забезпечує баланс між простотою реалізації, підтримуваністю, розширюваністю та відповідністю функціональним вимогам системи. Детальна взаємодія серверної частини, бази даних, клієнтського інтерфейсу та чатбота розглядається в наступному підрозділі.

2.3 Проєктування серверної частини, бази даних та клієнтського інтерфейсу

Проєктування вебзастосунку для підбору туристичних подорожей передбачає визначення внутрішньої організації системи та взаємозв’язків між її

основними складовими. На цьому етапі розглядається, як серверна частина, база даних і клієнтський інтерфейс забезпечують виконання основних користувацьких та адміністративних сценаріїв. Система повинна підтримувати перегляд туристичних пропозицій, фільтрацію, авторизацію, бронювання, роботу з обраними турами, відгуками та адміністративним керуванням. Тому структура вебзастосунку побудована з урахуванням розподілу відповідальності між окремими рівнями.

Серверна частина виконує роль центрального рівня обробки даних і бізнес-логіки. Вона приймає HTTP-запити від клієнтської частини, перевіряє коректність отриманих даних, виконує авторизацію, звертається до бази даних і повертає результат у форматі API-відповіді. Такий підхід дає змогу виконувати критичну логіку на сервері, контролювати доступ користувачів і забезпечувати цілісність даних.

Backend-частина має впорядковану структуру. У серверній директорії передбачено папки config, controllers, middleware, models, routes, services та utils. Директорія config використовується для налаштувань, зокрема підключення до бази даних. У routes описуються API-маршрути, через які frontend звертається до серверної частини. Директорія controllers містить основну логіку обробки запитів, models – схеми даних для MongoDB, middleware – проміжні функції для перевірки авторизації, ролей, валідації та помилок, а services і utils – допоміжну логіку.

Основними файлами серверної частини є server.js та app.js. Файл server.js відповідає за запуск сервера та підключення до бази даних, тоді як app.js налаштовує Express-застосунок: підключає middleware, визначає маршрути API, задає обробку неіснуючих маршрутів і централізовану обробку помилок. Такий поділ відокремлює запуск сервера від конфігурації застосунку та робить backend-структуру більш зрозумілою.

Логіка роботи серверної частини побудована за послідовністю: маршрут, middleware, контролер, модель або сервіс, база даних. Коли користувач виконує дію в інтерфейсі, клієнтська частина надсилає запит до відповідного endpoint.

Після цього middleware перевіряє доступ до дії, коректність переданих даних і роль користувача. Далі контролер виконує основну логіку: отримує дані, звертається до моделей, створює або оновлює записи в базі даних і повертає результат клієнтській частині.

Для наочного подання програмної структури вебзастосунку побудовано діаграму компонентів, зображену на рис. 2.5.

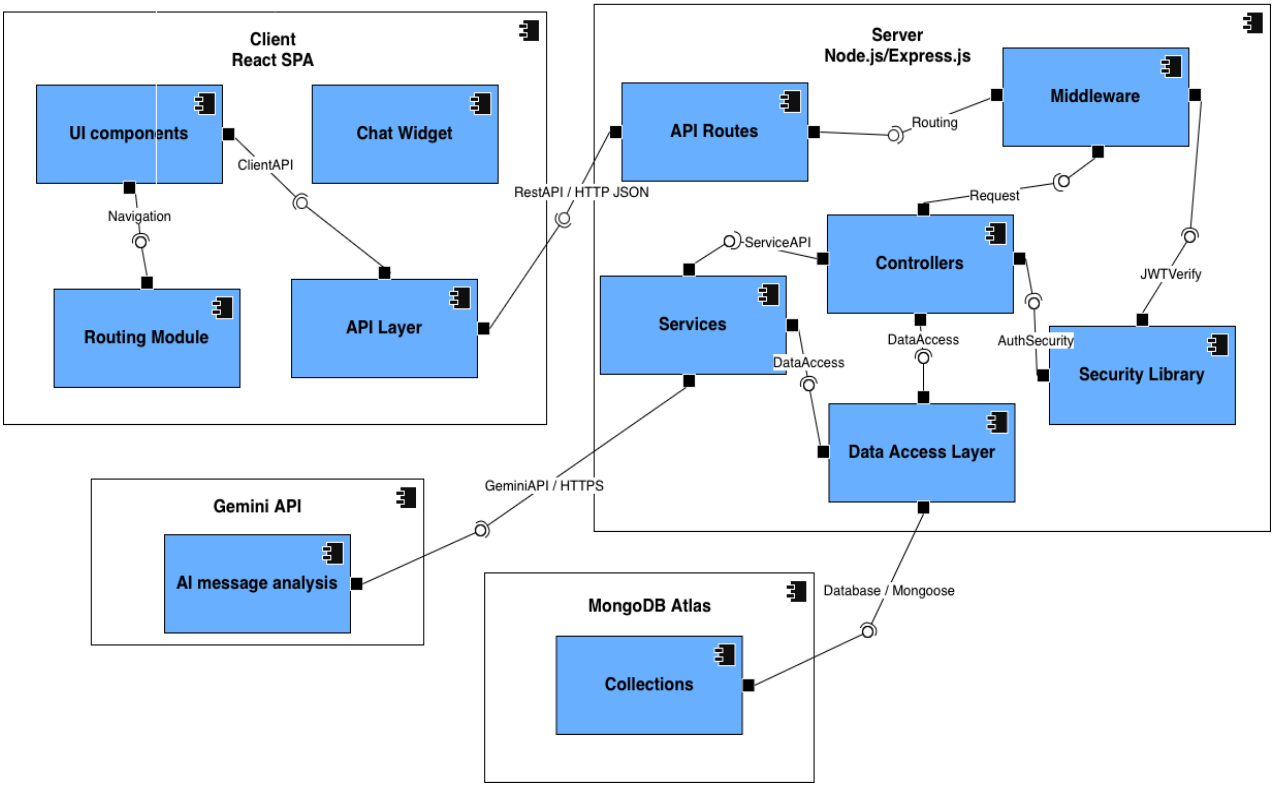

Рисунок 2.5 – Діаграма компонентів вебзастосунку для підбору туристичних подорожей

Логіка авторизації забезпечує реєстрацію, вхід користувача, отримання поточного профілю та оновлення персональних даних. Під час реєстрації пароль не зберігається у відкритому вигляді, а перетворюється на хеш. Після успішного входу користувач отримує JWT-токен, який використовується для подальших звернень до захищених маршрутів. Це дає змогу системі розмежовувати можливості гостя, зареєстрованого користувача й адміністратора.

Робота з туристичними пропозиціями реалізована через окрему групу серверних маршрутів і контролерів. Вона забезпечує отримання списку турів, перегляд конкретного туру за ідентифікатором, фільтрацію, створення, редагування та видалення записів. Операції перегляду доступні всім користувачам, а керування вмістом каталогу обмежене адміністративною роллю. Завдяки цьому звичайний користувач може переглядати каталог і деталі туру, але не має доступу до зміни даних у системі.

Логіка бронювання реалізує сценарій створення заявки на туристичну подорож. Після вибору туру користувач переходить до форми бронювання, вводить контактні дані, кількість мандрівників, бажану дату та коментар. Серверна частина перевіряє, чи існує обраний тур, чи він доступний для бронювання, чи кількість мандрівників не перевищує максимально допустиму, а також чи бажана дата входить у доступний період туру. Після проходження перевірок створюється заявка зі статусом pending.

Система статусів бронювання відображає життєвий цикл заявки. Початковий статус pending означає, що заявка створена й очікує опрацювання. Адміністратор може підтвердити її, після чого вона отримує статус confirmed, або відхилити зі статусом rejected. Користувач може скасувати заявку, якщо вона ще не була остаточно опрацьована; у такому разі статус змінюється на cancelled. Така модель дає змогу контролювати процес бронювання та відображати його стан у профілі користувача й адміністративній частині.

Функціонал обраних турів забезпечує можливість зберігати цікаві туристичні пропозиції для подальшого перегляду. Цей блок пов’язаний із профілем користувача, оскільки список обраного є індивідуальним для кожного авторизованого користувача. Додавання туру в обране не створює бронювання, але дає змогу користувачу сформувати власний список потенційно цікавих варіантів і повернутися до них пізніше.

Робота з відгуками забезпечує збереження оцінок і текстових коментарів до турів. Відгук пов’язується з конкретним користувачем і конкретною туристичною пропозицією. Це дає змогу відображати користувацький досвід на

сторінці детального перегляду туру. Створення відгуку доцільно обмежувати підтвердженим бронюванням, оскільки це підвищує достовірність оцінок і зменшує ризик випадкових або необґрунтованих коментарів.

Адміністративна частина забезпечує керування системою. Адміністратор має доступ до статистики, списку турів, форм створення та редагування туристичних пропозицій, а також до списку бронювань. На рівні backend доступ до адміністративних дій обмежується middleware, який перевіряє роль користувача. Завдяки цьому адміністративні маршрути не можуть бути використані звичайними користувачами.

Проєктування бази даних виконано з урахуванням основних сутностей предметної області. Для збереження даних використовується MongoDB, а робота з документами здійснюється через Mongoose. Основними колекціями є users, tours, bookings, reviews і chatSessions. У цьому підрозділі основну увагу зосереджено на сутностях, які забезпечують роботу каталогу, користувачів, бронювань, відгуків і адміністративної частини.

Оскільки в системі використовується документна база даних MongoDB, класичне приведення до нормальних форм, характерне для реляційних баз даних, не застосовується в повному вигляді. Водночас під час проєктування структури даних було використано принципи нормалізації: основні сутності предметної області винесено в окремі колекції, а зв’язки між ними реалізовано через посилання на ідентифікатори документів. Такий підхід дає змогу уникати надмірного дублювання інформації та підтримувати логічну цілісність даних.

Структура бази даних наближена до третьої нормальної форми на рівні логічної моделі. Дані користувачів зберігаються в колекції users, туристичні пропозиції – у tours, заявки на бронювання – у bookings, відгуки – у reviews, а історія взаємодії з чатботом – у chatSessions. У колекції bookings не дублюється повна інформація про користувача або тур, а зберігаються посилання userId і tourId. Аналогічно, у колекції reviews відгук пов’язується з користувачем і туром через відповідні ідентифікатори. Це відповідає принципу, за яким дані

про одну сутність зберігаються в одному місці, а залежні записи посилаються на неї.

Водночас документна модель MongoDB дає змогу зберігати вкладені дані там, де вони безпосередньо належать до основної сутності. Наприклад, у документі туру можуть зберігатися фотографії, перелік включених і невключених послуг, опис готелю або програма туру. Такі дані є складовими туристичної пропозиції, тому їх розміщення всередині документа Tour не суперечить логіці проєктування. Отже, структура бази даних поєднує принципи нормалізації основних сутностей із гнучкістю документної моделі MongoDB.

Колекція users зберігає інформацію про користувачів системи. До основних полів належать ім’я, email, хеш пароля, роль і список обраних турів. Роль може мати значення user або admin, що дає змогу розмежовувати доступ до звичайного й адміністративного функціоналу. Список обраних турів реалізується через посилання на записи з колекції tours.

Центральною сутністю бази даних є Tour, оскільки саме вона описує туристичну пропозицію. Модель туру містить назву, країну, місто, назву готелю, опис, тип подорожі, ціну, валюту, тривалість, кількість людей, максимальну кількість гостей, дату початку, дату завершення, транспорт, тип харчування, зірковість готелю, рейтинг, доступність, фотографії, опис готелю, програму туру, включені та невключені послуги. Така структура дозволяє використовувати один запис туру для кількох частин системи: картки в каталозі, детальної сторінки, фільтрації, бронювання та адміністративного редагування.

Модель Booking описує заявку на бронювання. Вона пов’язує користувача з конкретним туром через поля userId і tourId. Окрім цих посилань, заявка містить статус, кількість людей, загальну ціну, ім’я клієнта, email, телефон, бажану дату, коментар і джерело заявки. Статус заявки дає змогу відстежувати її поточний стан і відображати його в профілі користувача та адміністративній панелі. Зв’язок із користувачем і туром забезпечує можливість переглянути, хто створив заявку та на яку саме пропозицію вона оформлена.

Модель Review використовується для збереження відгуків. Вона містить посилання на користувача, посилання на тур, числовий рейтинг і текстовий коментар. Такий зв’язок дозволяє відображати відгуки на сторінці відповідного туру та пов’язувати їх із конкретним автором. Крім того, модель відгуку може використовуватися для формування середнього рейтингу туристичної пропозиції.

Зв’язки між сутностями бази даних реалізовано через посилання Mongoose. Колекція bookings містить посилання на користувача та тур, тому один користувач може мати багато заявок, а один тур може бути пов’язаний із багатьма бронюваннями. Колекція reviews пов’язує користувача з конкретним туром, що дає змогу відображати відгуки на сторінці відповідної туристичної пропозиції. Список favorites у колекції users реалізує зв’язок між користувачем і кількома збереженими турами. Така структура відповідає логіці предметної області та забезпечує цілісне зберігання даних без дублювання повної інформації про тур або користувача в кожному пов’язаному документі.

Клієнтський інтерфейс спроєктовано як односторінковий застосунок на React. Його структура включає директорії api, components, context, hooks, layouts, pages і utils. Такий поділ дає змогу розділити сторінки, багаторазові компоненти, глобальні стани, шаблони сторінок, API-запити та допоміжні функції. Завдяки цьому клієнтська частина не є набором ізольованих сторінок, а має впорядковану структуру, придатну для підтримки та розширення.

Маршрутизація клієнтської частини організована через React Router. Основними сторінками є головна сторінка, каталог турів, сторінка детального перегляду туру, сторінка бронювання, профіль користувача, обрані тури, сторінки входу та реєстрації, адміністративна панель, сторінка керування турами й сторінка керування заявками. Такий набір сторінок відповідає основним ролям системи: гість працює з відкритими сторінками, авторизований користувач – із профілем, обраним і бронюваннями, адміністратор – з адміністративною частиною.

Головна сторінка виконує презентаційну та навігаційну функцію. Вона спрямовує користувача до основних можливостей системи: пошуку подорожі, перегляду каталогу або ознайомлення з популярними напрямами. На цьому рівні користувач отримує перше уявлення про призначення сервісу та може перейти до основного сценарію – добору туристичної пропозиції.

Реалізація головної сторінки вебзастосунку зображено на рис. 2.6.

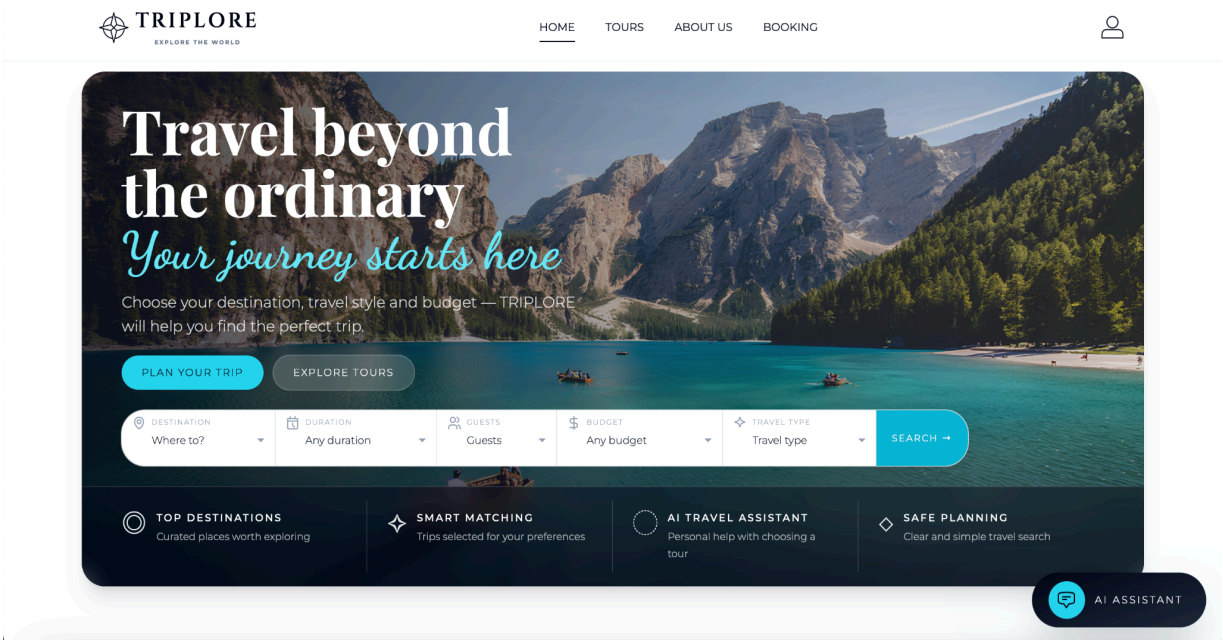

Рисунок 2.6 – Головна сторінка вебзастосунку

Сторінка каталогу турів є одним із ключових екранів інтерфейсу. Вона відображає туристичні пропозиції у вигляді карток. Картка туру містить основну інформацію, необхідну для попереднього порівняння: фото, назву, країну, місто, готель, тип туру, рейтинг, тривалість, кількість гостей, транспорт, харчування та ціну. Користувач може перейти до детальної сторінки або додати тур в обране. Пошук і фільтрація дають змогу звузити список пропозицій відповідно до заданих параметрів.

Сторінку каталогу з блоком фільтрації туристичних пропозицій зображено на рисунку 2.7. Відображення туристичних пропозицій у вигляді карток подано на рис. 2.8.

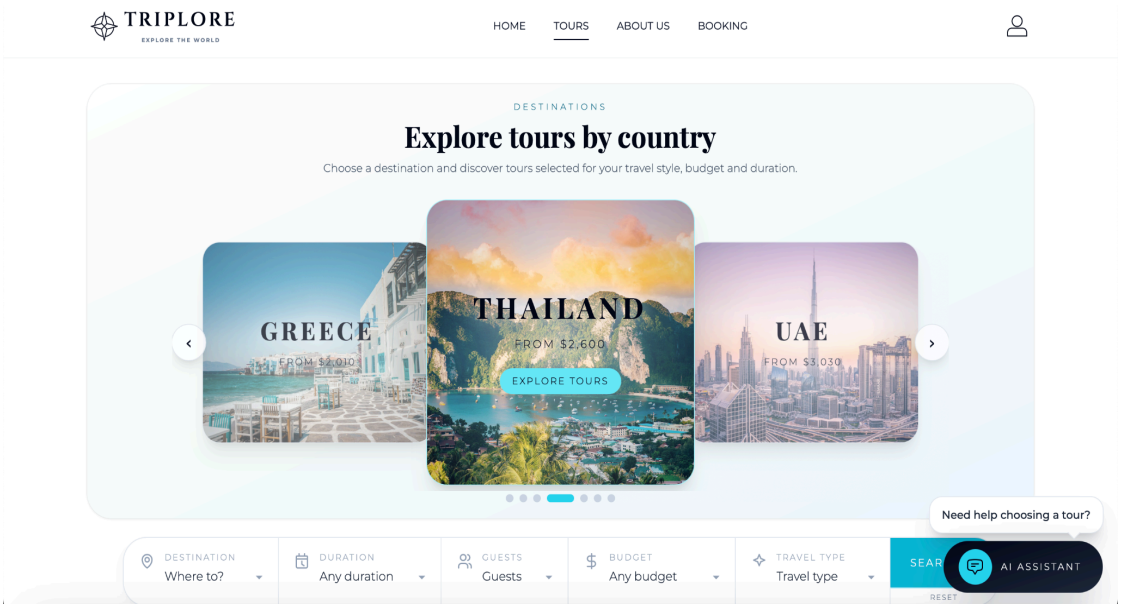

Рисунок 2.7 – Сторінка каталогу туристичних пропозицій

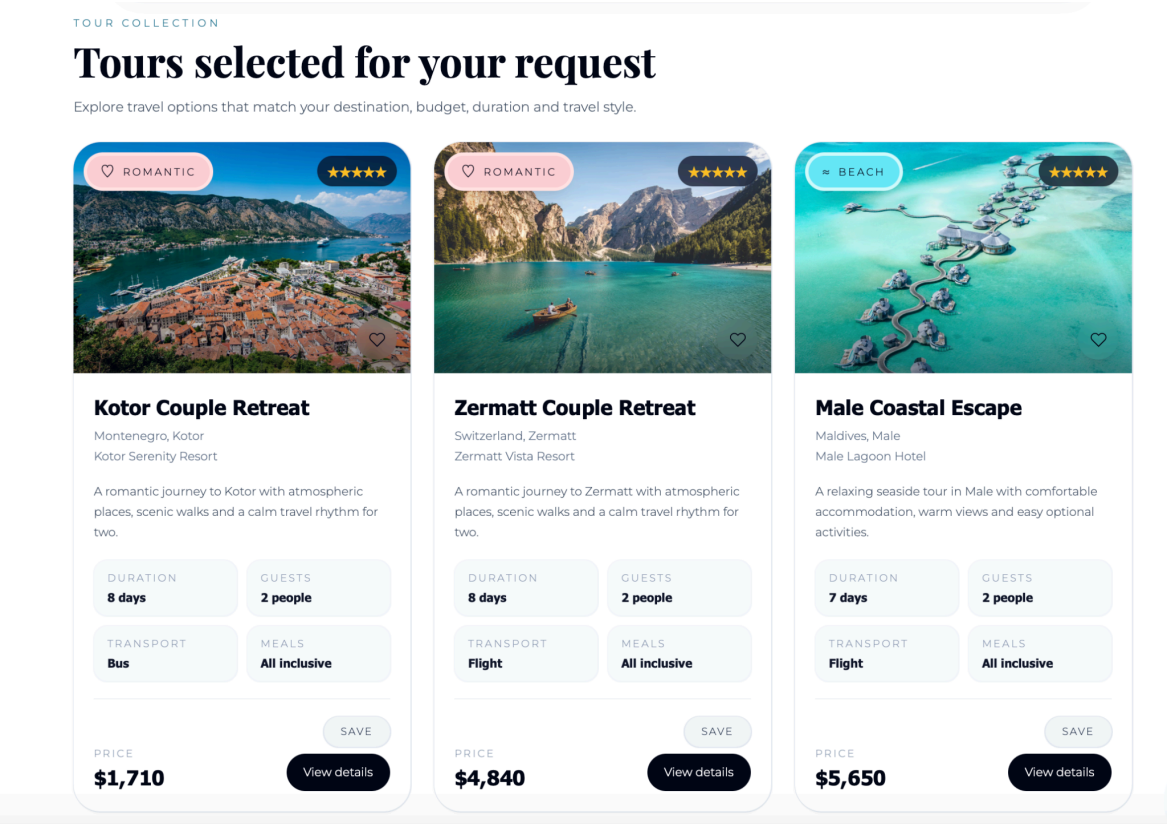

Рисунок 2.8 – Відображення туристичних пропозицій у каталозі

Сторінка детального перегляду туру надає розширену інформацію про конкретну пропозицію. Вона містить головне фото, назву туру, країну й місто, опис, вартість, рейтинг, інформацію про готель, дати, транспорт, харчування, включені та невключені послуги, активності й відгуки. Саме ця сторінка

виконує функцію прийняття рішення, оскільки користувач отримує повний опис туру перед переходом до бронювання.

Сторінка бронювання реалізована як форма введення даних. Користувач зазначає контактну інформацію, кількість мандрівників, бажану дату та коментар. На frontend виконуються базові перевірки заповнення полів, а остаточна перевірка відбувається на сервері. Такий поділ є важливим: клієнтська перевірка підвищує зручність користувача, але серверна перевірка гарантує коректність і безпеку створення заявки.

Форму створення заявки на бронювання туру подано на рис. 2.9.

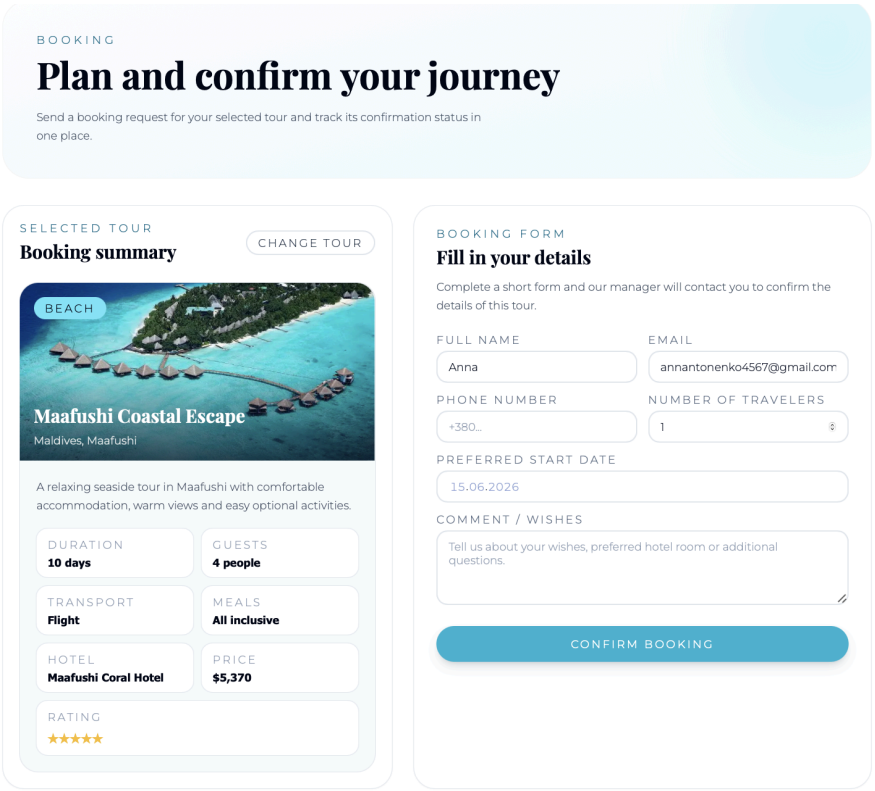

Рисунок 2.9 – Форма створення заявки на бронювання туру

Профіль користувача забезпечує доступ до персональної інформації, власних бронювань, статусів заявок і збережених турів. Такий функціонал дає змогу користувачу контролювати результати своєї взаємодії із системою. Обрані тури винесені в окремий сценарій, що дозволяє повернутися до цікавих пропозицій без повторного пошуку в каталозі.

Інтерфейс профілю користувача з інформацією про заявки та збережені тури подано на рис. 2.10.

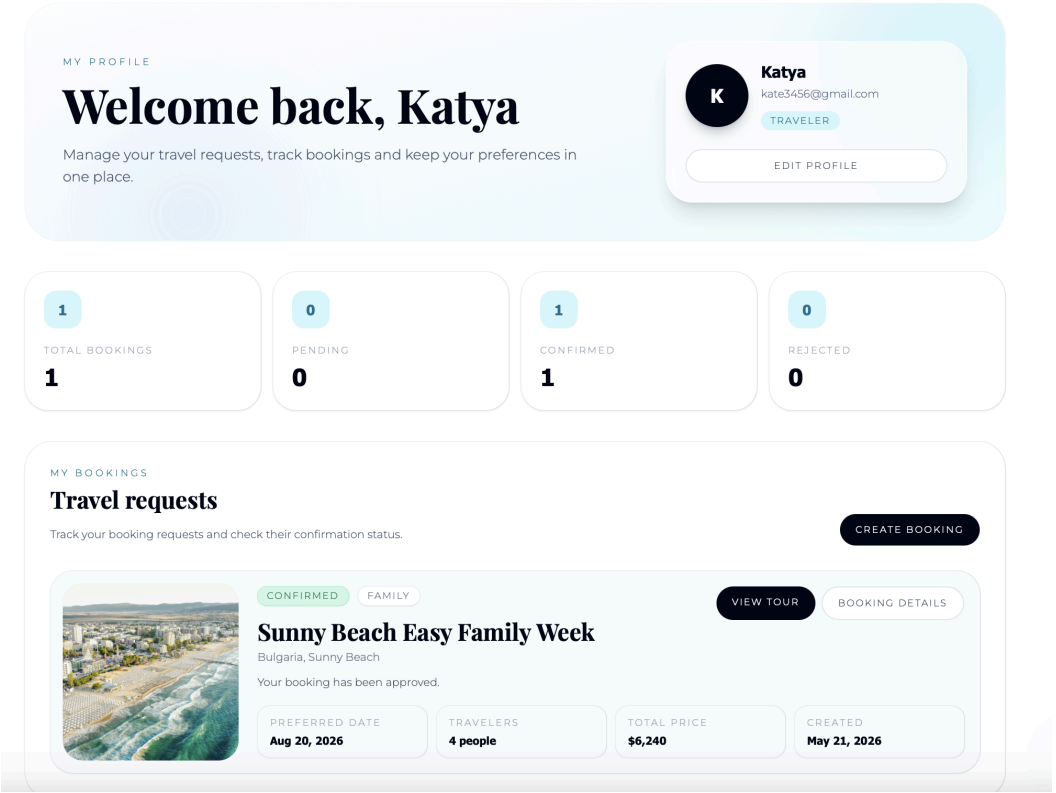

Рисунок 2.10 – Профіль користувача вебзастосунку

Адміністративний інтерфейс має окрему структуру та доступний лише користувачам із роллю адміністратора. Він містить сторінку статистики, сторінку керування турами та сторінку керування бронюваннями. На сторінці керування турами адміністратор може додавати нові пропозиції, редагувати наявні та видаляти неактуальні записи. На сторінці бронювань адміністратор переглядає заявки, фільтрує їх і змінює статуси. Окремий адміністративний layout відокремлює ці дії від звичайного користувацького інтерфейсу.

Адміністративний інтерфейс для керування туристичними пропозиціями та заявками зображено на рис. 2.11.

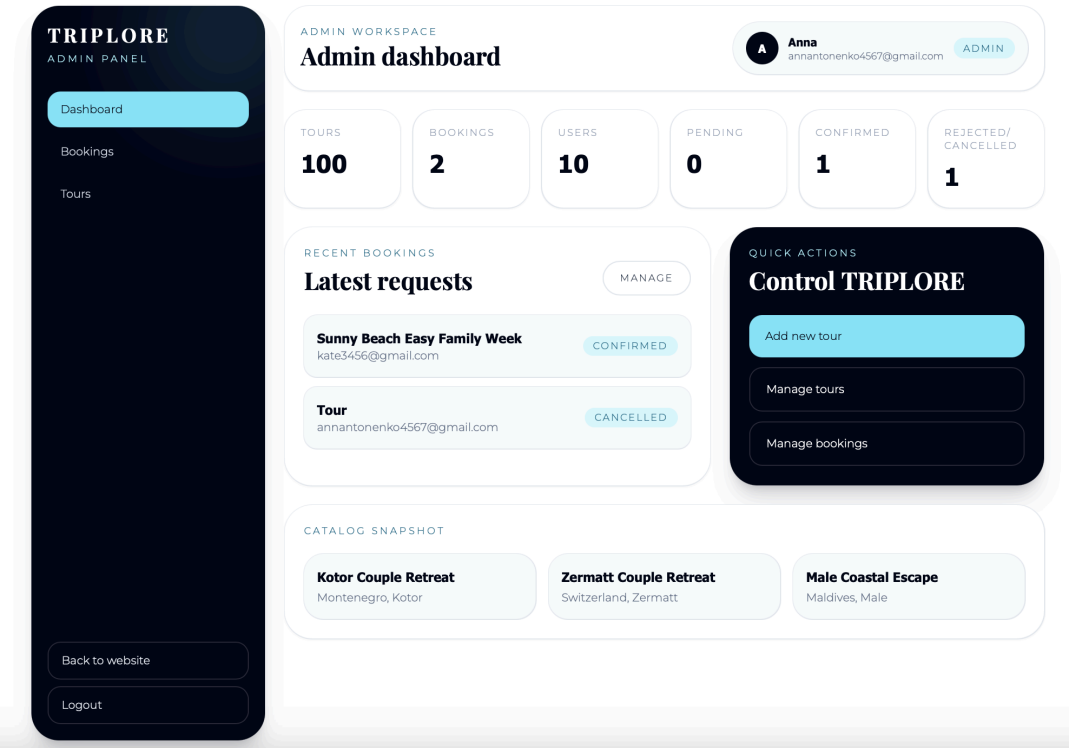

Рисунок 2.11 – Адміністративна панель вебзастосунку

Адаптивність клієнтського інтерфейсу забезпечується за допомогою responsive-класів Tailwind CSS. Сторінки мають коректно відображатися на різних типах пристроїв, оскільки туристичні сервіси часто використовуються не лише з комп’ютера, а й зі смартфона. Мобільне меню, гнучкі сітки карток і адаптивні форми забезпечують доступність основних сценаріїв незалежно від розміру екрана.

Отже, серверна частина, база даних і клієнтський інтерфейс спроєктовані як взаємопов’язані компоненти єдиної системи. Backend забезпечує API, авторизацію, бізнес-логіку, перевірку даних, роботу з турами, бронюваннями, обраним, відгуками та адміністративними операціями. База даних зберігає користувачів, туристичні пропозиції, заявки та відгуки, підтримуючи зв’язки між основними сутностями. Frontend надає користувачу й адміністратору зручний інтерфейс для виконання відповідних дій. Така структура створює основу для стабільної роботи вебзастосунку та подальшого розширення його функціональних можливостей.

2.4 Реалізація та інтеграція чатбота у вебзастосунок

У вебзастосунку для підбору туристичних подорожей реалізовано інтегрований AI-чатбот, призначений для діалогової підтримки користувача під час вибору туру. Його основне завдання полягає в тому, щоб допомогти користувачу сформулювати запит, уточнити параметри подорожі та отримати релевантні рекомендації на основі реальних туристичних пропозицій, збережених у базі даних. На відміну від звичайного довідкового чату, цей інструмент бере участь у процесі підбору: аналізує повідомлення користувача, визначає намір, витягує ключові параметри подорожі, звертається до каталогу турів і повертає структуровану відповідь із рекомендованими варіантами.

Чатбот інтегровано у вебзастосунок на клієнтському та серверному рівнях. На клієнтському рівні він реалізований як React-компонент інтерфейсу, що відповідає за візуальне відображення чату, введення повідомлень, показ відповідей і карток рекомендованих турів. На серверному рівні чатбот працює через окремі API-маршрути, які приймають повідомлення користувача, обробляють їх, взаємодіють із Gemini API, звертаються до MongoDB, виконують підбір турів і повертають результат на frontend.

Загальна логіка роботи чатбота побудована як послідовний обмін даними між користувачем, клієнтською частиною, сервером, AI-сервісом і базою даних. Користувач вводить повідомлення природною мовою, наприклад про бажання знайти пляжний відпочинок для двох осіб на тиждень. Компонент чату на frontend надсилає повідомлення на серверний маршрут POST /api/chat/message. Далі запит обробляється контролером чатбота, який визначає поточну сесію, зберігає повідомлення, аналізує намір користувача, витягує параметри подорожі, виконує пошук відповідних турів у MongoDB і формує відповідь.

На frontend чатбот реалізовано у файлі ChatWidget.jsx. Цей компонент відповідає за відкриття та закриття вікна чату, збереження введеного

повідомлення, надсилання запиту на backend, відображення відповідей, показ індикатора завантаження та виведення рекомендованих турів. Чат працює як плаваючий віджет, тому користувач може звернутися до нього без переходу на окрему сторінку. Це підвищує зручність взаємодії, оскільки консультаційна підтримка доступна під час перегляду основних сторінок сайту.

Чатбот підключено до основного шаблону користувацької частини через MainLayout.jsx. У цьому файлі компонент ChatWidget імпортується та виводиться в інтерфейсі за умови, що користувач не перебуває на сторінках авторизації або реєстрації. Така інтеграція відповідає логіці туристичного сервісу, де потреба в уточненні параметрів може виникати на різних етапах користувацького сценарію.

Інтерфейс чатбота складається з плаваючої кнопки, вікна діалогу, заголовка, області повідомлень, поля введення, кнопки надсилання та блоку рекомендованих турів. Початкове повідомлення бота інформує користувача про можливість отримати допомогу у виборі подорожі та пропонує вказати напрям, бюджет і стиль відпочинку. Повідомлення відображаються через компонент ChatMessage.jsx, який розділяє повідомлення користувача та відповіді бота. Якщо backend повертає рекомендовані тури, вони відображаються у вигляді карток із фото, назвою, країною, містом, тривалістю, ціною, рейтингом, поясненням відповідності запиту та кнопкою переходу до детальної сторінки.

Реалізація інтерфейсу чатбота зображена на рисунку 2.12

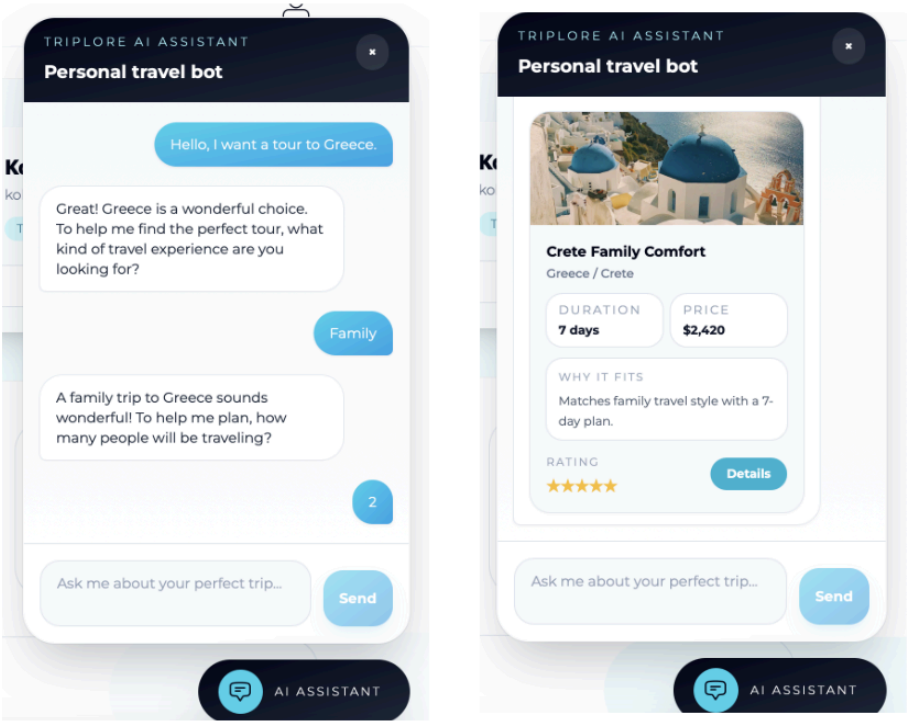

Рисунок 2.12 – Інтерфейс чатбота

Надсилання повідомлення на backend виконується через централізований axiosInstance. Після введення тексту компонент ChatWidget формує запит на маршрут /chat/message, передаючи sessionId і текст повідомлення. sessionId використовується для ідентифікації поточного діалогу, а message містить запит користувача. Оскільки запит проходить через спільний екземпляр Axios, до нього може автоматично додаватися JWT-токен, якщо користувач авторизований.

На backend повідомлення обробляється через маршрут POST /api/chat/message, описаний у chatRoutes.js. Для цього маршруту використовується optionalAuthMiddleware, що дозволяє звертатися до чатбота без обов’язкової авторизації. Якщо користувач авторизований, його ідентифікатор зберігається в чат-сесії. Якщо користувач працює як гість, сесія підтримується через ідентифікатор, збережений у браузері. Такий підхід дає змогу використовувати чатбот як гостям, так і зареєстрованим користувачам.

Для збереження історії діалогу використовується модель ChatSession. Вона містить ідентифікатор користувача, якщо він авторизований, масив повідомлень, витягнуті параметри подорожі, список рекомендованих турів, вибраний тур, стан діалогу та чернетку бронювання. На клієнтському рівні sessionId зберігається в localStorage, тому після перезавантаження сторінки діалог не починається з нуля. Коли користувач знову відкриває чат, frontend може завантажити попередню історію повідомлень.

Зміна станів чат-сесії під час взаємодії користувача з чатботом подана на рисунку 2.13. Діаграма відображає основні етапи роботи бота: очікування запиту, аналіз повідомлення, уточнення параметрів, використання fallback-режиму, пошук турів, формування рекомендацій, вибір туру, оформлення бронювання та завершення діалогу.

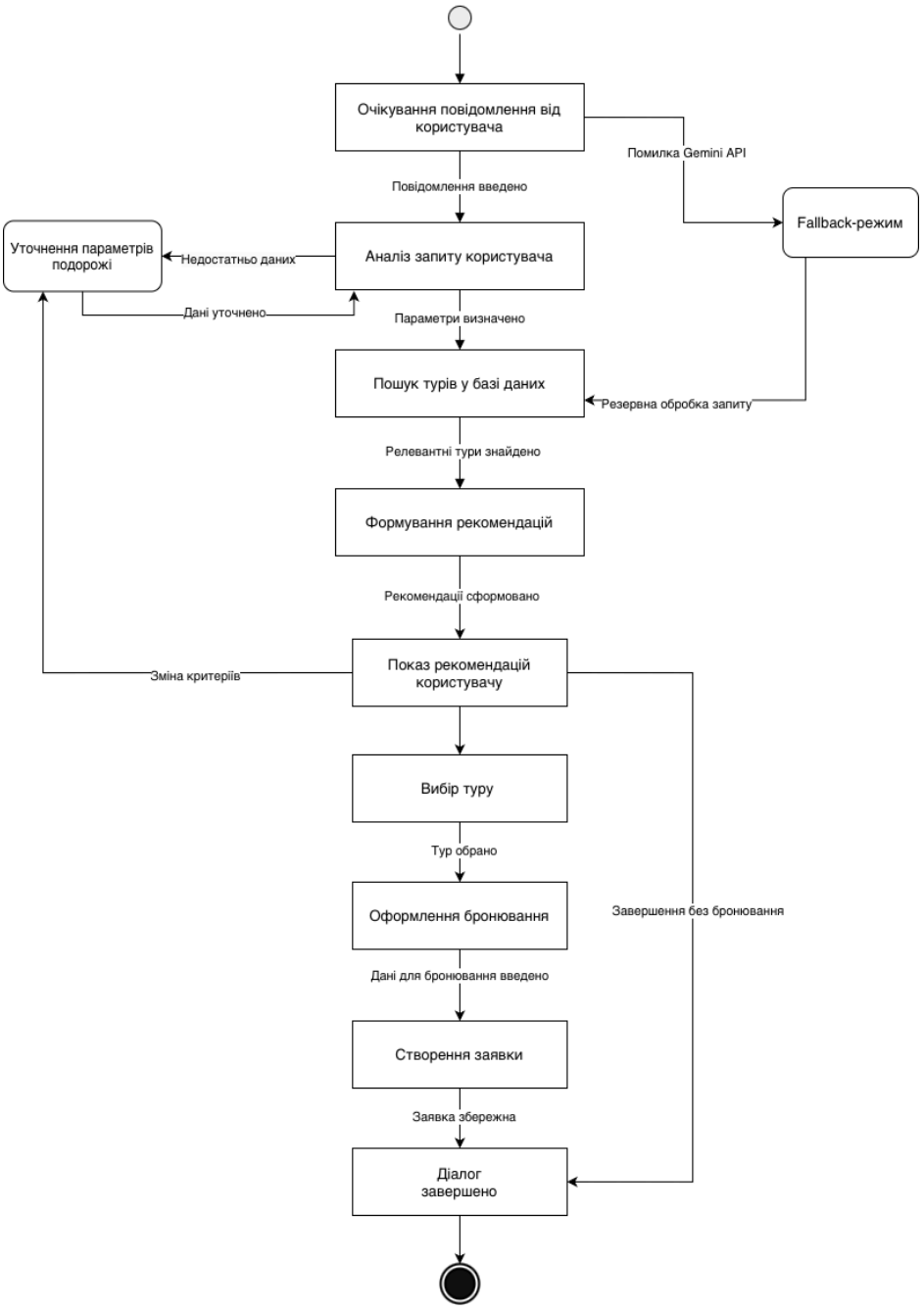

Рисунок 2.13 – Діаграма станів чатбота під час підбору та бронювання туру

Основна серверна логіка чатбота реалізована в chatController.js. Контролер перевіряє повідомлення, знаходить або створює чат-сесію, додає повідомлення користувача в історію, аналізує намір, витягує параметри подорожі, виконує пошук турів у базі даних, ранжує знайдені пропозиції, формує відповідь і зберігає її в історії. Після цього сервер повертає відповідь на frontend разом із рекомендованими турами, якщо такі були знайдені.

AI-частина реалізована у файлі geminiService.js із використанням пакета @google/generative-ai. Ключ доступу до Gemini API зберігається на backend у змінних середовища, тому він не потрапляє на клієнтську частину. Gemini використовується для аналізу повідомлення користувача, визначення наміру, витягнення параметрів подорожі та формування відповіді. Така інтеграція дає змогу підтримувати природномовну взаємодію, коли користувач описує бажану подорож не через фільтри, а звичайним текстом [27].

Чатбот може визначати країну, місто, тип туру, бюджет, тривалість, кількість людей, транспорт, тип харчування, дати, сезон, рівень готелю та додаткові побажання. Витягнуті параметри зберігаються в чат-сесії, що дає змогу використовувати їх у наступних повідомленнях. Якщо користувач надав недостатньо інформації, бот може поставити уточнювальне питання щодо бюджету, кількості мандрівників або тривалості подорожі. Завдяки цьому реалізовано не одноразовий пошук, а послідовний діалоговий сценарій.

Після отримання достатньої кількості параметрів backend формує запит до MongoDB. Пошук враховує доступність туру, тип подорожі, країну, місто, бюджет, кількість людей, тривалість, транспорт, харчування, рівень готелю, дати та сезон. Якщо точних збігів не знайдено, система поступово розширює умови пошуку: наприклад, може шукати без прив’язки до конкретного міста або пропонувати доступні тури ширшої категорії. Такий підхід дає змогу не залишати користувача без відповіді навіть у разі вузького запиту.

Ранжування рекомендованих турів виконується в recommendationService.js. Сервіс обчислює умовний показник відповідності для кожної туристичної пропозиції. Бали нараховуються за збіг типу туру, відповідність бюджету, країні, місту, тривалості, кількості людей, транспорту, харчуванню, рівню готелю, сезону, датам і рейтингу. Після цього система повертає найбільш релевантні варіанти. Такий підхід можна визначити як комбіновану модель: AI аналізує природну мову, а backend виконує формальний підбір і ранжування на основі структурованих даних.

Коли список турів сформовано, backend передає відібрані варіанти до Gemini для створення природної відповіді. AI-сервіс отримує не всю базу даних, а лише попередньо знайдені й ранжовані пропозиції. Відповідь має бути короткою, структурованою та зрозумілою для користувача. На frontend ці самі варіанти додатково відображаються у вигляді карток, що поєднує текстову рекомендацію з візуальним представленням.

У чатботі реалізовано fallback-режим. Він використовується у випадках, коли Gemini API недоступний, відсутній або некоректний API-ключ, перевищено квоту, досягнуто rate limit, виникла мережева помилка, отримано порожню відповідь або некоректний JSON. У таких ситуаціях користувач не отримує технічне повідомлення про помилку. Система переходить у спрощений режим і повідомляє, що працює з обмеженою відповіддю, але все одно може допомогти з вибором подорожі.

Fallback-логіка здатна витягувати базові параметри з тексту: бюджет, кількість людей, тривалість, тип подорожі, країну або місто. Це означає, що навіть без активної відповіді Gemini система може поставити уточнювальне питання або запропонувати тури з каталогу. Такий механізм підвищує відмовостійкість чатбота й забезпечує безперервність взаємодії з користувачем.

Чатбот підтримує не лише сценарій підбору, а й сценарій бронювання. Якщо користувач обирає один із запропонованих турів і повідомляє про бажання його забронювати, система переходить у стан bookingInProgress. Далі бот послідовно запитує повне ім’я, телефон, email, кількість людей і бажану дату. Після збору необхідних даних backend створює запис бронювання через модель Booking. Перед створенням заявки виконується перевірка доступності туру, допустимої кількості людей і відповідності дати періоду проведення туру.

Заявки, створені через чатбот, мають джерело chatbot. Це дає змогу відрізняти їх від заявок, оформлених через стандартну форму на сайті. Така ознака може бути корисною для подальшого аналізу ефективності чатбота, оскільки адміністратор може бачити, які бронювання були створені через

діалоговий сценарій. Отже, чатбот не лише рекомендує тури, а й може супроводжувати користувача до практичної дії – створення заявки.

Для захисту чат-сесій використовується логіка, що враховує статус користувача. Гість може користуватися чатботом через сесію, ідентифікатор якої зберігається в браузері. Якщо користувач авторизований, його userIdзберігається в ChatSession, а доступ до такої сесії обмежується відповідним користувачем. Це запобігає несанкціонованому перегляду чужих авторизованих діалогів і водночас зберігає можливість користування чатботом без реєстрації.

У межах системи чатбот виконує роль персонального туристичного консультанта. Він не замінює каталог турів, а доповнює його. Каталог залишається зручним інструментом для ручного пошуку й фільтрації, тоді як чатбот орієнтований на користувачів, які хочуть описати побажання природною мовою. Наприклад, замість ручного вибору фільтрів користувач може написати, що шукає романтичну подорож для двох осіб у межах певного бюджету. Система перетворює такий текст на структуровані параметри та шукає відповідні тури в базі даних.

Отже, чатбот інтегрований у вебзастосунок як повноцінний функціональний компонент. На frontend він реалізований як плаваючий React-віджет, підключений до основного layout сайту. На backend він працює через API-маршрути, контролер, Gemini API, сервіс рекомендацій і модель чат-сесії. Бот аналізує повідомлення користувача, витягує параметри подорожі, шукає реальні тури в MongoDB, ранжує їх і повертає структуровані рекомендації. Завдяки збереженню сесії, fallback-режиму, зв’язку з каталогом турів і можливості створення бронювання чатбот виконує роль практичного інструменту підбору туристичних подорожей.

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Вибір методів тестування

Тестування програмного продукту виконувалося з урахуванням структури вебзастосунку для підбору туристичних подорожей. Система містить клієнтську частину, серверну частину, REST API, механізм авторизації, каталог турів, функціонал бронювання, відгуки, адміністративну панель та інтегрований чатбот. Через це для перевірки застосунку обрано кілька методів тестування, кожен із яких відповідає окремому рівню роботи системи.

Основу перевірки окремих логічних операцій становить модульне тестування. Воно дає змогу перевірити функції, які можуть працювати ізольовано від інтерфейсу, бази даних та інших частин застосунку. У проєкті для цього використовується Jest. На модульному рівні перевіряється логіка, пов’язана з авторизацією та базовою обробкою даних: хешування пароля, перевірка правильного й неправильного пароля, генерація та перевірка JWT-токена, валідація складності пароля, розрахунок загальної вартості бронювання та базова перевірка дати. Такий підхід дозволяє переконатися, що окремі частини програмної логіки працюють правильно незалежно від інших компонентів системи [28].

Для перевірки взаємодії серверних компонентів обрано інтеграційне тестування. У вебзастосунку більшість дій користувача проходить через кілька рівнів: API-маршрут приймає запит, middleware перевіряє авторизацію або роль користувача, контролер виконує основну логіку, а модель працює з даними. Тому важливо перевіряти не лише окремі функції, а й узгодженість їхньої роботи. Для інтеграційного тестування використовуються Jest і Supertest. Ці інструменти дають змогу перевіряти API-запити, коди відповідей і дані, які повертає backend [28; 29].

Окремо застосовується API-тестування, оскільки клієнтська частина взаємодіє із сервером саме через REST API. У межах цього виду тестування перевіряються маршрути авторизації, каталогу турів, бронювання, відгуків та базових адміністративних дій. Основну увагу приділено правильності обробки HTTP-запитів, поверненню очікуваних статусів відповіді, структурі JSON-даних і роботі захищених маршрутів. Для API-перевірок також використовується Supertest у поєднанні з Jest.

Функціональне тестування обрано для перевірки відповідності роботи системи визначеним вимогам. Воно зосереджується на тому, чи правильно виконуються основні можливості вебзастосунку з погляду користувача. До таких можливостей належать реєстрація, авторизація, перегляд каталогу турів, фільтрація, відкриття детальної сторінки туру, перегляд профілю, створення бронювання, робота з відгуками та базові адміністративні дії. Цей вид тестування дає змогу перевірити не внутрішню реалізацію коду, а фактичну поведінку системи.

Для перевірки користувацьких сценаріїв у браузері використовується end-to-end тестування. У проєкті для цього застосовується Playwright, який імітує дії реального користувача: відкриття сторінок, введення даних у форми, натискання кнопок, перехід між сторінками та перевірку відображення результатів. Такий підхід є важливим для односторінкового React-застосунку, оскільки дозволяє перевірити роботу інтерфейсу як цілісного користувацького сценарію.

Для оцінювання стабільності серверної частини обрано навантажувальне тестування. У проєкті для цього використовується k6. Навантажувальні перевірки зосереджені на основних API-запитах, зокрема авторизації та отриманні каталогу турів. Під час такого тестування аналізуються час відповіді, кількість запитів, частка помилок, а також показники P95 і P99. Це дає змогу оцінити, як backend API поводиться під час багаторазових запитів [30].

Ручне тестування використовується як доповнення до автоматизованих перевірок. Воно застосовується для оцінювання зовнішнього вигляду сторінок,

зручності навігації, поведінки форм, роботи адміністративної панелі, чатбота та адаптивності інтерфейсу на різних пристроях. Такий тип перевірки є необхідним, оскільки окремі аспекти користувацького досвіду складно повністю оцінити автоматизованими тестами.

Отже, для тестування програмного продукту обрано комплекс методів, які охоплюють основні рівні роботи вебзастосунку. Jest використовується для модульного й інтеграційного тестування, Supertest – для перевірки API, Playwright – для браузерних користувацьких сценаріїв, k6 – для навантажувального тестування, а ручне тестування – для перевірки інтерфейсу, адаптивності та зручності взаємодії. Такий набір методів дає змогу перейти до розроблення тест-плану та опису конкретних тестових сценаріїв.

3.2 Розроблення тест-плану вебзастосунку

Для перевірки вебзастосунку для підбору туристичних подорожей було розроблено тест-план, який охоплює основні рівні роботи системи: окремі логічні функції, серверні API-сценарії, користувацькі дії в інтерфейсі, навантаження на backend та ручну перевірку елементів, які потребують візуальної оцінки. Тест-план побудовано з урахуванням фактично реалізованих тестів у проєкті, тому він відображає реальний обсяг перевірки програмного продукту.

Основними об’єктами тестування визначено авторизацію, захищені маршрути, каталог турів, фільтрацію, перегляд детальної інформації про тур, бронювання, відгуки, базові адміністративні дії, стабільність API та інтерфейс користувача. Окремо враховано ручну перевірку чатбота й адаптивності, оскільки ці сценарії вимагають не лише технічної перевірки, а й оцінювання зручності взаємодії.

На логічному рівні тестування спрямоване на перевірку операцій, які можуть виконуватися ізольовано від інтерфейсу й бази даних. До них належать хешування пароля, перевірка правильного та неправильного пароля, генерація

JWT-токена, перевірка складності пароля, розрахунок вартості бронювання та валідація дати. Ці перевірки дають змогу підтвердити коректність базових функцій, від яких залежить подальша робота авторизації та бронювання.

Таблиця 3.1 – Розподіл перевірок за рівнями вебзастосунку

№	Рівень перевірки	Об’єкт перевірки	Інструмент	Очікуваний результат
1	Логічний рівень	Паролі, JWT, складність пароля, дата, розрахунок вартості	Jest	Функції повертають очікувані значення
2	Серверний рівень	API авторизації, турів, бронювань, відгуків, admin-дій	Jest, Supertest	Запити обробляються коректно
3	Клієнтський рівень	Реєстрація, вхід, каталог, фільтрація, сторінка туру	Playwright	Сценарії виконуються в браузері без помилок
4	Рівень продуктивності	POST /api/auth/login, GET /api/tours	k6	API працює в межах заданих порогів
5	Користувацький рівень	UI, навігація, адаптивність, чат-бот, адмін-панель	Ручна перевірка	Інтерфейс працює відповідно до очікуваної логіки

Серверний рівень охоплює перевірку API-сценаріїв. У межах цього рівня перевіряються реєстрація користувача, вхід у систему, доступ до захищених маршрутів, отримання каталогу турів, фільтрація за параметрами, створення бронювання, підтвердження заявки адміністратором і створення відгуку після підтвердженого бронювання. Такі перевірки дають змогу оцінити, чи правильно backend обробляє запити та повертає очікувані відповіді.

Клієнтський рівень передбачає перевірку поведінки вебзастосунку в браузері. До тест-плану включено сценарії реєстрації через інтерфейс, входу в систему, переходу до каталогу, відкриття детальної сторінки туру, застосування фільтра за країною, перегляду секції відгуків і перенаправлення неавторизованого користувача на сторінку входу. Ці сценарії показують, чи може користувач виконати основні дії через інтерфейс.

Оскільки вебзастосунок активно використовує API, до тест-плану включено навантажувальну перевірку основних endpoint-ів. Для цього обрано запити авторизації та отримання каталогу турів, оскільки вони належать до базових сценаріїв роботи системи. Критеріями успішності визначено частку помилок, час відповіді та показники P95 і P99.

Окремий блок тест-плану становить ручна перевірка. Вона застосовується для інтерфейсу, адаптивності, чатбота, навігації, форм і адміністративної панелі. Ці елементи потребують безпосередньої перевірки через реальний інтерфейс, оскільки їх якість визначається не лише технічною працездатністю, а й зручністю використання.

Таблиця 3.2 – Перевірка сценаріїв за ролями користувачів

№	Роль у системі	Сценарій перевірки	Спосіб перевірки	Очікуваний результат
1	2	3	4	5
1	Гість	Перегляд головної сторінки та каталогу	Playwright, ручна перевірка	Відкриті сторінки доступні без авторизації
2	Гість	Перехід до профілю	Playwright	Виконується перенаправлення на сторінку входу
3	Користувач	Реєстрація та вхід у систему	Jest, Supertest, Playwright	Користувач створюється й авторизується

Продовження таблиці 3.2

1	2	3	4	5
4	Користувач	Перегляд каталогу та сторінки туру	Supertest, Playwright	Тури відображаються, сторінка деталей відкривається
5	Користувач	Створення заявки на бронювання	Supertest, ручна перевірка	Заявка створюється зі статусом pending
6	Користувач	Перегляд секції відгуків	Playwright	Секція відгуків відображається на сторінці туру
7	Адміністратор	Перегляд статистики	Supertest, ручна перевірка	Адміністратор отримує доступ до службових даних
8	Адміністратор	Підтвердження бронювання	Supertest, ручна перевірка	Статус заявки змінюється на confirmed

Таблиця 3.3 – Перелік автоматизованих тестових сценаріїв

TC ID	Сценарій перевірки	Тип тестування	Очікуваний результат
1	2	3	4
TC001	Хешування пароля	Модульне	Хеш не збігається з початковим паролем
TC002	Перевірка правильного та неправильного пароля	Модульне	Правильний пароль проходить перевірку, неправильний відхиляється
TC003	Генерація JWT-токена	Модульне	Створюється токен із даними id і role

Продовження таблиці 3.3

1	2	3	4
TC004	Валідація складності пароля	Модульне	Перевіряється довжина, велика літера, цифра та спеціальний символ
TC005	Розрахунок вартості та перевірка дати	Модульне	Вартість обчислюється коректно, дата визначається як майбутня
TC006	Реєстрація нового користувача	Інтеграційне / API	Користувач створюється, дублювання email обробляється як помилка
TC007	Автентифікація користувача	Інтеграційне / API	Правильні дані дають успішний вхід, неправильні – помилку
TC008	Перевірка захищених маршрутів	Інтеграційне / API	Без токена доступ заборонено, user не має admin-доступу
TC009	Отримання та фільтрація турів	Інтеграційне / API	Повертається список турів, фільтрація працює за країною та maxPrice
TC010	Створення бронювання	Інтеграційне / API	Авторизований користувач створює заявку зі статусом pending
TC011	Підтвердження бронювання адміністратором	Інтеграційне / API	Адміністратор змінює статус заявки на confirmed
TC012	Створення відгуку	Інтеграційне / API	Відгук створюється після підтвердженого бронювання
TC013	Сценарій: вхід → каталог → сторінка туру	Функціональне / E2E	Користувач входить, бачить каталог і відкриває деталі туру
TC014	Неавторизований доступ до профілю	Функціональне / E2E	Користувач перенаправляється на сторінку входу

Продовження таблиці 3.3

1	2	3	4
TC019	Навантаження на каталог турів	Навантажувальне	Частка помилок не перевищує встановлений поріг
TC021	Навантаження на авторизацію	Навантажувальне	Успішність логіну перевищує 95 %, P95 не перевищує 5000 мс
TC024	Реєстрація через інтерфейс	Функціональне / E2E	Форма реєстрації заповнюється, сценарій завершується успішно
TC025	Пошук туру через фільтр країни	Функціональне / E2E	Після вибору країни відображаються відповідні результати
TC026	Перегляд секції відгуків	Функціональне / E2E	Авторизований користувач бачить секцію відгуків на сторінці туру

Розроблений тест-план охоплює основні частини вебзастосунку та визначає, які компоненти перевіряються автоматизовано, а які – вручну. Автоматизовані тести зосереджені на авторизації, API-сценаріях, каталозі турів, бронюванні, відгуках, базових адміністративних діях і навантаженні на API. Ручна перевірка доповнює тест-план оцінюванням інтерфейсу, адаптивності, чатбота й адміністративної панелі. Результати виконання перевірок наведено в наступному підрозділі.

3.3 Проведення тестування та аналіз отриманих результатів

Тестування вебзастосунку виконувалося відповідно до розробленого тест-плану. Перевірка охоплювала основні сценарії використання системи: реєстрацію, авторизацію, перегляд каталогу турів, фільтрацію, відкриття детальної сторінки, створення бронювання, роботу з відгуками, базові адміністративні дії та стабільність основних API-запитів. Додатково було

проведено ручну перевірку інтерфейсу, адаптивності на різних пристроях і роботи чатбота.

Модульне тестування виконувалося за допомогою Jest. Метою цього виду тестування була перевірка окремих логічних операцій, які не потребують запуску всього вебзастосунку. У межах модульних тестів перевірено хешування пароля за допомогою bcryptjs, порівняння правильного та неправильного пароля, генерацію JWT-токена, валідацію складності пароля, розрахунок загальної вартості туру та перевірку дати. Отримані результати підтвердили, що перевірені функції повертають очікувані значення та коректно працюють в ізольованому середовищі.

На рис. 3.1 зображено результат проходження модульних тестів. Усі перевірки завершено успішно, що підтверджує коректність базової логіки авторизації, роботи з паролями, JWT-токенами та допоміжними обчисленнями.

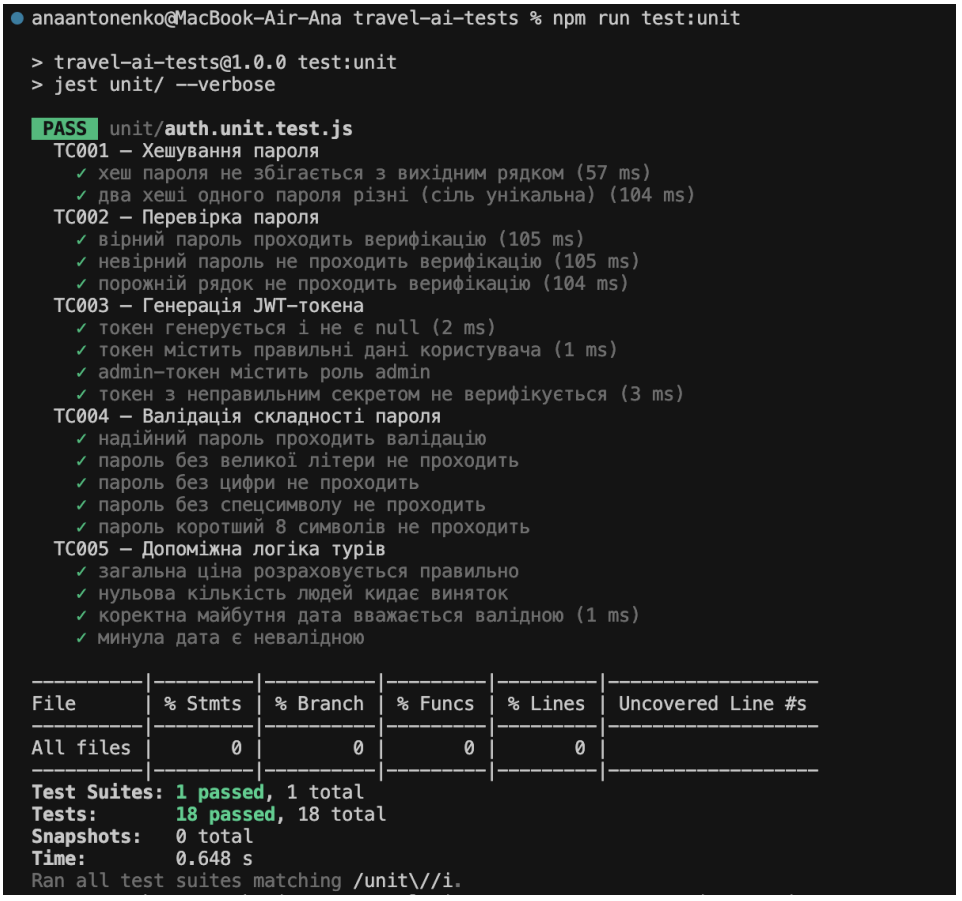

Рисунок 3.1 – Результат проходження модульних тестів

Інтеграційне та API-тестування проводилося з використанням Jest і Supertest. У межах цих перевірок було протестовано серверні сценарії, пов’язані з реєстрацією нового користувача, входом у систему, доступом до захищених маршрутів, отриманням каталогу турів, фільтрацією, створенням бронювання, підтвердженням заявки адміністратором і створенням відгуку після підтвердженого бронювання. Також перевірено, що запити без токена відхиляються, а користувач із роллю user не має доступу до адміністративних дій.

На рис. 3.2 наведено результат проходження інтеграційних та API-тестів. Результат підтверджує, що основні серверні сценарії виконуються відповідно до очікуваної логіки, а API повертає коректні відповіді для успішних і помилкових запитів.

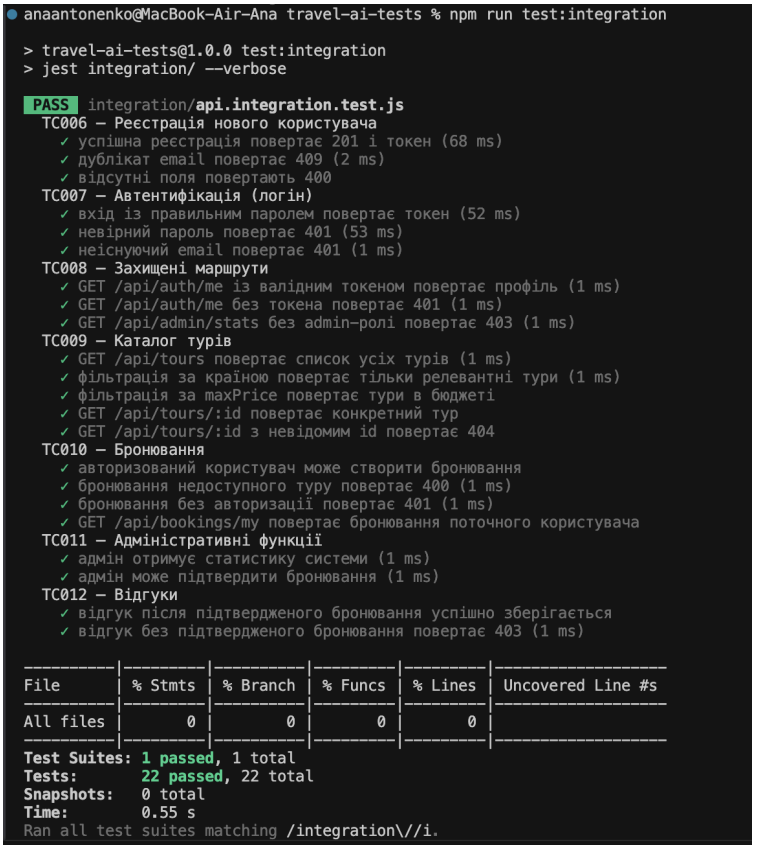

Рисунок 3.2 – Результат проходження інтеграційних та API-тестів

Функціональне та E2E-тестування виконувалося за допомогою Playwright. Ці перевірки імітували поведінку користувача в браузері. У межах тестування перевірено реєстрацію через інтерфейс, вхід у систему, перехід до каталогу турів, відображення карток, відкриття сторінки детального перегляду, фільтрацію за країною, перегляд секції відгуків і перенаправлення неавторизованого користувача зі сторінки профілю на сторінку входу. Такі сценарії дають змогу оцінити працездатність інтерфейсу з позиції кінцевого користувача.

На рис. 3.3 зображено результат проходження функціональних тестів Playwright. Отриманий результат підтверджує, що основні користувацькі сценарії в браузері виконуються без помилок, а переходи між сторінками та реакція інтерфейсу відповідають очікуваній поведінці.

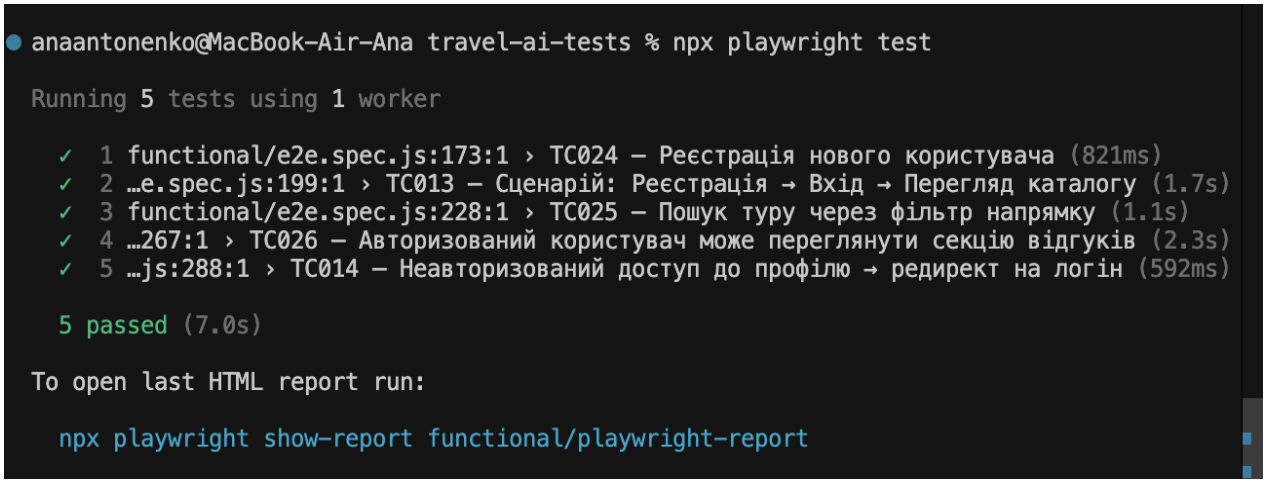

Рисунок 3.3 – Результат проходження функціонального тестування Playwright

Навантажувальне тестування виконувалося за допомогою k6. Перевірка була зосереджена на endpoint-ах POST /api/auth/login і GET /api/tours, оскільки вони належать до базових сценаріїв роботи вебзастосунку. Авторизація забезпечує доступ до персональних функцій, а каталог турів є основним розділом системи, тому стабільність цих запитів є важливою для загальної працездатності застосунку. Під час тестування оцінювалися час відповіді,

кількість запитів, частка помилок, показники median, P95, P99, успішність логіну та стабільність отримання каталогу.

На рис. 3.4 наведено результат навантажувального тестування k6. Результати показують, що основні API-запити працюють у межах заданих критеріїв: частка помилок не перевищує допустиме значення, а час відповіді залишається в межах установлених порогів.

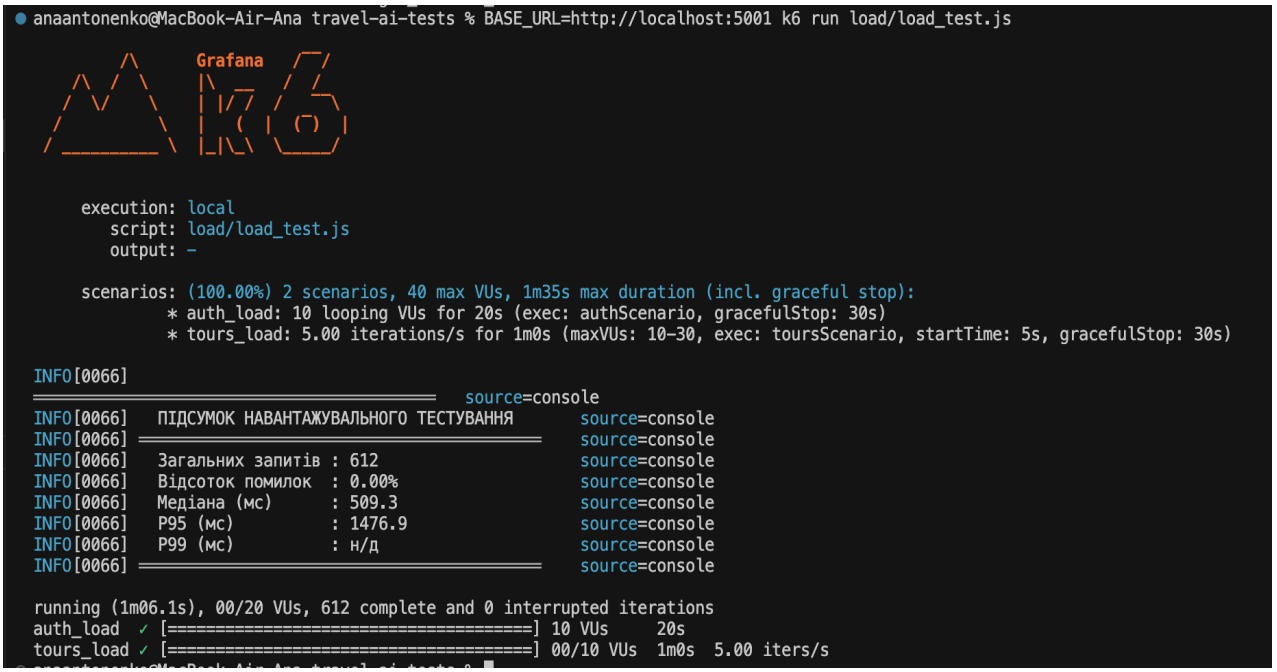

Рисунок 3.4 – Результат навантажувального тестування k6

Ручне тестування застосовувалося для перевірки тих частин системи, які потребують візуальної оцінки або взаємодії з реальним інтерфейсом. Основну увагу було приділено головній сторінці, каталогу турів, сторінці деталей, формі бронювання, профілю користувача, адміністративній панелі, чатботу, навігації та адаптивності на різних пристроях. Результати ручної перевірки наведено в табл. 3.4.

Таблиця 3.4 – Результати ручної перевірки вебзастосунку

ID	Сценарій ручної перевірки	Очікуваний результат	Фактичний результат
1	2	3	4
MT001	Відкрити головну сторінку на desktop-екрані	Hero-блок, пошук, популярні напрями, навігація та footer відображаються коректно	Виконано
MT002	Перейти з головної сторінки до каталогу турів	Відкривається сторінка каталогу, картки турів завантажуються без помилок	Виконано
MT003	Перевірити картку туру в каталозі	На картці відображаються фото, назва, країна, місто, готель, тривалість, ціна та рейтинг	Виконано
MT004	Натиснути кнопку переходу до деталей туру	Відкривається окрема сторінка туру з повним описом, фото, датами, послугами та відгуками	Виконано
MT005	Заповнити форму бронювання авторизованим користувачем	Поля форми доступні для введення, після надсилання створюється заявка на бронювання	Виконано
MT006	Перевірити профіль користувача після створення заявки	У профілі відображаються особисті дані, бронювання, статус заявки та збережені тури	Виконано
MT007	Додати тур в обране та відкрити сторінку обраних турів	Обраний тур зберігається та відображається у відповідному розділі	Виконано
MT008	Увійти як адміністратор і відкрити адміністративну панель	Адміністратор бачить статистику, список турів і заявки на бронювання	Виконано

Продовження таблиці 3.4

1	2	3	4
МТ009	Змінити статус заявки в адміністративній панелі	Статус бронювання змінюється після дії адміністратора та коректно відображається в системі	Виконано
МТ010	Перевірити відкриття чатбота на сторінці каталогу	Плаваючий віджет відкривається, поле введення доступне, повідомлення надсилається	Виконано
МТ011	Надіслати чатботу запит про підбір туру	Бот повертає відповідь і, за наявності відповідних варіантів, показує рекомендовані тури	Виконано
МТ012	Перевірити головну сторінку на мобільному екрані 320–480 px	Контент не виходить за межі екрана, горизонтальний скрол відсутній, блоки перебудовуються коректно	Виконано
МТ013	Перевірити роботу мобільного меню	Меню відкривається кнопкою, пункти навігації доступні для натискання, переходи працюють	Виконано
МТ014	Перевірити каталог турів на мобільному екрані	Картки турів відображаються в одну колонку, текст і кнопки не перекриваються	Виконано
МТ015	Перевірити форму бронювання на мобільному екрані	Поля форми зручно заповнювати, кнопка надсилання доступна, елементи не виходять за межі екрана	Виконано

Продовження таблиці 3.4

1	2	3	4
МТ017	Перевірити чатбот на мобільному екрані	Вікно чату не перекриває критичний контент, поле введення та кнопка надсилання доступні	Виконано
МТ018	Перевірити переходи між головною, каталогом, деталями туру, профілем і адмін-панеллю	Усі переходи виконуються без помилок, користувач потрапляє на відповідні сторінки	Виконано
МТ019	Перевірити повідомлення про помилки у формах	У разі некоректного або неповного введення система показує зрозуміле повідомлення	Виконано

Результати тестування підтвердили працездатність основних функцій вебзастосунку. Користувач може зареєструватися, увійти в систему, переглядати каталог турів, застосовувати фільтрацію, відкривати детальну сторінку туру, створювати заявку на бронювання та переглядати відгуки. Адміністратор має доступ до базових функцій керування, зокрема перегляду статистики та підтвердження бронювання. Серверна частина коректно обробляє основні API-запити, а інтерфейс забезпечує доступ до ключових сценаріїв роботи системи.

Водночас результати тестування показують, що автоматизоване покриття може бути розширене в подальшому. Автоматизованими тестами не повністю охоплено роботу чатбота, повні сценарії адміністративної панелі через інтерфейс, створення бронювання через браузер, адаптивність на різних viewport-ах і взаємодію з реальною MongoDB у тестовому середовищі. Зазначені обмеження не впливають на базову працездатність системи, однак можуть бути враховані під час подальшого розвитку програмного продукту.

Отже, проведене тестування підтвердило, що вебзастосунок для підбору туристичних подорожей працює відповідно до основних сценаріїв використання. Поєднання автоматизованих і ручних перевірок дало змогу оцінити різні рівні системи: логіку авторизації, API, клієнтський інтерфейс, навантаження на серверну частину та зручність взаємодії з основними функціями вебзастосунку.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено вебзастосунок для підбору туристичних подорожей з інтегрованим чатботом.

У процесі роботи проаналізовано туристичні вебсервіси та визначено основні підходи до організації пошуку, фільтрації, перегляду й бронювання туристичних пропозицій. Окремо розглянуто використання чатботів у туристичній сфері та встановлено, що діалогова взаємодія допомагає користувачу точніше сформулювати запит і швидше перейти до вибору релевантної подорожі.

Розглянуто сучасні технології веброзробки та обґрунтовано вибір архітектури й технологічного стеку, які відповідають завданням створення клієнт-серверного вебзастосунку. На основі аналізу предметної галузі сформовано вимоги до системи та спроєктовано її основні складові: структуру вебзастосунку, базу даних, серверну частину й клієнтський інтерфейс.

Практичним результатом роботи стала реалізація вебзастосунку, який забезпечує пошук, перегляд і бронювання турів, авторизацію користувачів, роботу з обраними пропозиціями, адміністративне керування турами й заявками, а також використання чатбота для підтримки користувача під час підбору подорожі.

Проведене тестування підтвердило працездатність основних функцій системи. Перевірено сценарії реєстрації та входу користувача, перегляду каталогу турів, фільтрації, відкриття детальної сторінки, створення заявки на бронювання, роботи адміністративної частини та взаємодії з чатботом. Отримані результати свідчать, що розроблений вебзастосунок виконує визначені завдання та може бути використаний як основа для подальшого розвитку туристичного онлайн-сервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шевелюк М. М. Цифровізація у сфері туризму: інноваційні тренди і пріоритетні напрями розвитку. Питання культурології. 2021. Вип. 38. С. 226–235. DOI: 10.31866/2410-1311.38.2021.245956.

2. Грабар М. В. Основні показники цифрової трансформації міжнародної туристичної індустрії. Проблеми економіки. 2021. № 3(49). С. 10–15. DOI: 10.32983/2222-0712-2021-3-10-15.

3. Information and Communication Technologies in Tourism 2021 / ed. by W. Wörndl, C. Koo, J. L. Stienmetz. Cham : Springer, 2021.

4. Писарева І., Яловнича К. Сталий розвиток туризму в цифрову епоху: розвиток смарт-дестинацій. Економіка та суспільство. 2024. № 69. DOI: 10.32782/2524-0072/2024-69-104.

5. Байрачна О. К. Використання штучного інтелекту та Big Data для оптимізації управління туристичними напрямами. Ukrainian Journal of Applied Economics and Technology. 2024. Т. 9, № 3. С. 252–255.

6. Artificial Intelligence and Tourism: G7. OECD Tourism Papers. 2024. URL: https://www.oecd.org/content/dam/oecd/en/publications/reports/2024/12/artificial-intelligence-and-tourism_41e7f157/3f9a4d8d-en.pdf (дата звернення: 29.05.2026).

7. Digital Transformation. UN Tourism. URL: https://www.unwto.org/digital-transformation (дата звернення: 01.06.2026).

8. Camilleri M. A., Troise C. Chatbot recommender systems in tourism: A systematic review and a benefit-cost analysis. Proceedings of the 2023 7th International Conference on E-Commerce, E-Business and E-Government. 2023. DOI: 10.1145/3589883.3589906.

9. Gatzioufa P., Saprikis V. Chatbots in tourism: A literature review on users' behavioral intention towards their adoption. AIP Conference Proceedings. 2023. DOI: 10.1063/5.0182620.

10. Calvaresi D., Ibrahim A., Calbimonte J.-P., Schegg R., Fragnière E., Schumacher M. I. The evolution of chatbots in tourism: A systematic literature review. Information and Communication Technologies in Tourism 2021. Cham : Springer, 2021.

11. Corradini F., Muzi C., Re B., Tiezzi F. A Classification of BPMN Collaborations based on Safeness and Soundness Notions. Electronic Proceedings in Theoretical Computer Science. 2018. Vol. 276. P. 37–52. DOI: 10.4204/EPTCS.276.5. URL: https://pubblicazioni.unicam.it/retrieve/e0ff0073-4eb4-9bac-e053-1705fe0af019/paper%20%281%29.pdf (дата звернення: 02.06.2026).

12. Литвин В. В., Наум О. М., Висоцька В. А., Дверій М. В. Архітектура системи онлайн-туризму для пошуку та планування подорожей із урахуванням потреб користувача. Information Systems and Networks. 2019. № 6. С. 13–29. DOI: 10.23939/sisn2019.02.013. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2019/nov/19755/191675maket2-15-31.pdf (дата звернення: 29.05.2026).

13. Londhe K., Pawar S., Patil A. Enhanced Travel Experience using Artificial Intelligence. Procedia Computer Science. 2024.

14. Zhou Q. Current Dynamics and Future Growth of Online Travel Agencies: A Systematic Literature Review. 2024.

15. Tripadvisor launches AI-powered travel planning product. Tripadvisor. 2023. URL: https://tripadvisor.mediaroom.com/Tripadvisor-launches-AI-powered-travel-planning-product (дата звернення: 01.06.2026).

16. Free Trip Planner & AI Itinerary Builder. Tripadvisor. URL: https://www.tripadvisor.com/Trips (дата звернення: 02.06.2026).

17. Priceline Debuts New AI-Powered Trip Intelligence Features to Save Consumers Time and Money. Priceline. 2024. URL: https://press.priceline.com/priceline-winter-2024-product-release/ (дата звернення: 29.05.2026).

18. 2025 Summer Release: Now you can Airbnb more than an Airbnb. Airbnb. 2025. URL: https://news.airbnb.com/airbnb-2025-summer-release/ (дата звернення: 01.06.2026).

19. Airbnb 2025 Summer Release. Airbnb. 2025. URL: https://news.airbnb.com/product-releases/airbnb-2025-summer-release/ (дата звернення: 02.06.2026).

20. Viator: Travel Tours, Activities, and Things to Do. Viator. URL: https://www.viator.com/ (дата звернення: 29.05.2026).

21. React Documentation. Quick Start. URL: https://react.dev/learn (дата звернення: 01.06.2026).

22. Node.js Documentation. Introduction to Node.js. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення: 02.06.2026).

23. Express.js Documentation. Node.js web application framework. URL: https://expressjs.com/ (дата звернення: 29.05.2026).

24. MongoDB Documentation. MongoDB Docs. URL: https://www.mongodb.com/docs/ (дата звернення: 01.06.2026).

25. Mongoose Documentation. Getting Started. URL: https://mongoosejs.com/docs/ (дата звернення: 02.06.2026).

26. JSON Web Token. Introduction to JSON Web Tokens. URL: https://jwt.io/introduction (дата звернення: 29.05.2026).

27. Google AI for Developers. Gemini API Documentation. URL: https://ai.google.dev/gemini-api/docs (дата звернення: 01.06.2026).

28. Jest Documentation. Getting Started. URL: https://jestjs.io/docs/getting-started (дата звернення: 02.06.2026).

29. Playwright Documentation. Getting Started. URL: https://playwright.dev/docs/intro (дата звернення: 29.05.2026).

30. Grafana k6 Documentation. Get started with k6. URL: https://grafana.com/docs/k6/latest/get-started/ (дата звернення: 01.06.2026).

ДОДАТКИ

Додаток А – Посилання на репозиторій

Репозиторій з програмною реалізацією, виконаною в межах кваліфікаційної роботи, розташовано за адресою https://github.com/AnnAntonenko07/travel-ai-app.git.