Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного засобу з планування подорожей»

Виконав: студент 4 курсу
групи ЗПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Веретковський Д.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ЗІ Лукічов В.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ Романюк О.Н.
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Веретковському Денису Володимировичу

1. Тема роботи – «Розробка програмного засобу з планування подорожей»

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: формалізація методу формування маршруту подорожі; визначення швидкості обробки запитів під час генерації маршрутів подорожей – не більше 12 секунд на запит; суб'єктивне задоволення користувачів під час час роботи з інтерфейсом програмного засобу: не менше 7 показників з 10; визначення продуктивності процесу генерації маршрутів подорожей – не менше 60 маршрутів за годину.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку програм для планування подорожей; розробка алгоритмів роботи застосунку; розробка програмного застосунку; тестування розробленого програмного застосунку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета,

об’єкт та предмет дослідження; задачі дослідження, наукова новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів, використані технології, тестування, апробація результатів та публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О.М., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку програм для планування подорожей	14.03.24 – 24.03.24	виконано
2	Порівняльний аналіз аналогів	24.03.24 – 06.04.24	виконано
3	Розробка структури застосунку	06.04.24 – 11.04.24	виконано
4	Розробка інтерфейсу користувача	12.04.24 – 29.04.24	виконано
5	Тестування застосунку	29.04.24 – 11.05.24	виконано
6	Оформлення матеріалів до захисту БДР	12.05.24– 30.05.24	виконано

Студент _____ (підпис) Веретковський Д.В. (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) Хошаба О.М. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Веретковський Д.В. Розробка програмного засобу з планування подорожей: бакалаврська дипломна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 93 с.

У бакалаврській дипломній роботі проведено аналіз стану питання розробки застосунку. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного модуля клієнта.

Виконане обґрунтування вибору мови та середовища програмування, а також технологій, які були використані для розробки застосунку для планування подорожей.

Розроблено програмний продукт для планування подорожей.

Робота реалізована за допомогою мови програмування Python. В якості середовища для розробки програмного забезпечення було обрано PyCharm.

Отримані в бакалаврській дипломній роботі результати можна використати для планування подорожей, бронювання готелів, планування витрат, перегляду прогнозу погоди, отримування рекомендацій для подорожі залежно від вказаного користувачем маршруту.

Ключові слова: подорожі, програмний засіб, планування.

ABSTRACT

UDC 004.912.032.26

Veretkovsky D.V. Development of travel planning software: bachelor's thesis in the specialty 121 – Software engineering, educational program – software engineering. Vinnytsia: VNTU, 2024. 93 p.

In the bachelor's thesis, an analysis of the state of the application development issue was carried out. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the expediency of developing the client's own module was proven.

A rationale for the choice of programming language and environment, as well as the technologies used to develop the travel planning application, is done.

A travel planning software product has been developed.

The work is implemented using the Python programming language. PyCharm was chosen as the software development environment.

The results obtained in the bachelor's thesis can be used for travel planning, hotel reservations, expense planning, viewing the weather forecast, receiving travel recommendations depending on the route specified by the user.

Keywords: travel, software, planning.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Подорожі є однією з найпопулярніших форм відпочинку та саморозвитку. Люди подорожують з різних причин: від відпочинку на пляжі та екстремальних пригод до пізнавальних турів та культурного обміну. Останнім часом спостерігається тенденція до зростання вартості подорожей через збільшення цін на паливо, проживання, харчування та екскурсії [1].

Одним з способів оптимізувати витрати на подорожі є використання спеціалізованих програм для планування подорожей. Ці програми дозволяють користувачам знайти найкращі варіанти авіарейсів, готелів, екскурсій та інших послуг, а також зберігати всю необхідну інформацію про подорож в одному місці.

Однією з найпопулярніших програм для планування подорожей є TripIt. Вона дозволяє користувачам створювати маршрути подорожей, додавати інформацію про рейси, готелі, автопрокат та інші послуги, а також ділитися своїми планами з друзями та родиною. TripIt також має функцію сповіщень, яка інформує користувачів про зміни в розкладі рейсів, скасування або затримки.

Іншою популярною програмою є Kayak. Вона дозволяє користувачам шукати та порівнювати ціни на авіарейси, готелі, автопрокат та інші послуги від різних постачальників. Kayak також має функцію сповіщень, яка інформує користувачів про зміни цін на обрані послуги.

Крім цих програм, існує багато інших інструментів для планування подорожей, таких як Google Trips, Booking.com, Airbnb та інші. Вони дозволяють користувачам знайти найкращі варіанти проживання, транспорту та розваг, а також зберігати всю необхідну інформацію про подорож в одному місці.

Отже, використання програм для планування подорожей допомагає економити час та гроші, а також робить подорож більш комфортною та безпечною[2].

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного

забезпечення.

Мета та завдання дослідження. Метою роботи є покращення процесу планування подорожі шляхом розробки спеціалізованого програмного застосунку.

Відповідно до поставленої мети необхідно виконати такі завдання:

- проаналізувати предметну область;
- провести порівняльний аналіз аналогів;
- провести аналіз методів розв'язання задачі;
- розробити модель застосунку;
- розробити метод формування маршруту подорожі;
- розробити алгоритми роботи програми;
- провести варіантний аналіз засобів реалізації застосунку;
- розробити функціонал застосунку;
- розробити інтерфейс користувача;
- провести аналіз методів тестування;
- сформувати системні вимоги для роботи програмного засобу;
- провести тестування розробленого застосунку.

Об'єктом дослідження є процеси розробки програмного засобу з планування подорожей.

Предметом дослідження є методи і засоби реалізації програмного засобу з планування подорожей.

Головною задачею є розробка програмного засобу з планування подорожей, який дозволятиме користувачам краще планувати подорожі та отримувати рекомендації щодо напрямків подорожі.

Методи дослідження. У процесі досліджень використовувались такі методи:

- метод формування маршруту подорожчі;
- методи тестування програмного застосунка;
- методи розробки інтерфейсу користувача.

Новизна отриманих результатів.

1. Здобув модального розвикту метод формування маршруту подорожі, у якому, на відміну від існуючих, використовується штучний інтелект для

формування маршруту подорожі між двома містами, рекомендацій щодо визначних місць, які можна відвідати на маршруті подорожі, що дозволяє полегшити процес планування подорожі.

2. Здобула подальшого розвитку модель системи, у якій, на відміну від існуючих, міститься комплексний функціонал для планування подорожей з використанням Google Palm API та Visual Weather Crossing API, що дозволяє підвищити інформативність програмного засобу та краще спланувати подорож користувачем.

Практична цінність отриманих результатів. Практична цінність одержавних результатів полягає у можливостів використання розробленого програмного засобу для планування подорожей, бронювання готелів, планування витрат, перегляду прогнозу погоди, отримування рекомендацій для подорожі залежно від вказаного користувачем маршруту.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто.

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використано 20 літературних джерел.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ

1.1 Аналіз стану питання розробки

Для проведення аналізу стану питання розробки програмного засобу з планування подорожей необхідно розглянути кілька ключових аспектів:

1. Ринкова ситуація та попит:

- Туристична галузь є величезною і постійно зростаючою. Люди все частіше подорожують і потребують зручних інструментів для планування своїх мандрівок.

- На ринку вже присутні численні додатки та веб-сервіси для планування подорожей, проте вони зазвичай зосереджуються на окремих аспектах, таких як бронювання квитків чи готелів.

- Існує попит на комплексні рішення, які об'єднують різні компоненти планування подорожі в одному місці [3].

2. Функціональність та особливості:

- Основними функціями такого програмного засобу повинні бути: пошук і бронювання авіаквитків, готелів, оренда автомобілів, планування маршрутів, створення маршрутів та графіків подорожі.

- Додатковими корисними можливостями можуть бути: рекомендації місць для відвідування, інтеграція з соціальними мережами для обміну досвідом, оффлайн-доступ до заплановані маршрутів тощо.

- Важливими факторами є зручний та інтуїтивно зрозумілий інтерфейс, можливість персоналізації та адаптації під індивідуальні потреби користувачів.

3. Конкуренція та диференціація:

- На ринку вже присутні великі гравці, такі як Expedia, TripAdvisor, Kayak та інші.

- Для успішної конкуренції необхідно запропонувати унікальні особливості, кращу користувацьку взаємодію або зосередитися на специфічній ніші (наприклад, подорожі для певної демографічної групи або типу активності) [4].

- Важливим фактором є стратегія маркетингу та просування програмного

забезпечення на ринку.

Отже, розробка ефективного та конкурентоспроможного програмного засобу для планування подорожей вимагає ретельного аналізу ринку, визначення унікальних переваг, використання сучасних технологій та інструментів.

1.2 Порівняльний аналіз аналогів

Expedia [5] – одна з найбільших і найуспішніших онлайн-платформ для бронювання подорожей у світі.

Заснована в 1996 році, Expedia стала піонером у галузі онлайн-бронювання подорожей. Компанія розпочала свою діяльність як підрозділ Microsoft, а згодом була виділена в окреме підприємство. Сьогодні Expedia – це глобальний бренд, який працює більш ніж у 70 країнах та пропонує широкий спектр послуг з планування та бронювання подорожей.

Основними продуктами та послугами Expedia є бронювання авіаквитків, бронювання готелів, оренда автомобілів, організація круїзів та пакетних турів. Окрім власного веб-сайту, компанія володіє низкою інших брендів, таких як Hotels.com, Travelocity, Orbitz, Wotif Group та інші. Ця диверсифікована екосистема дозволяє Expedia охопити різноманітні сегменти ринку подорожей.

Експедія відома своїм зручним та інтуїтивно зрозумілим користувацьким інтерфейсом, який дозволяє легко знайти та забронювати квитки, готелі та інші послуги для подорожей. Компанія також пропонує програму лояльності під назвою Expedia Rewards, яка нагороджує клієнтів знижками та спеціальними пропозиціями за регулярне користування послугами.

Одним з ключових факторів успіху Expedia є її широка глобальна присутність та партнерські стосунки з авіакомпаніями, готельними мережами та постачальниками послуг з усього світу. Завдяки цьому користувачі можуть знайти привабливі пропозиції незалежно від місця призначення. Крім того, Expedia активно інвестує в технології та інновації, щоб покращити досвід користувачів та підвищити ефективність своїх операцій.

Незважаючи на жорстку конкуренцію з боку інших онлайн-турагентств,

Expedia продовжує зберігати лідерські позиції на ринку завдяки своєму визнаному бренду, широкому асортименту послуг та зручності використання. Компанія постійно розширює свою присутність у різних регіонах світу та впроваджує нові інноваційні рішення для задоволення мінливих потреб сучасних мандрівників.

Застосунок Expedia продемонстровано на рисунку 1.1.

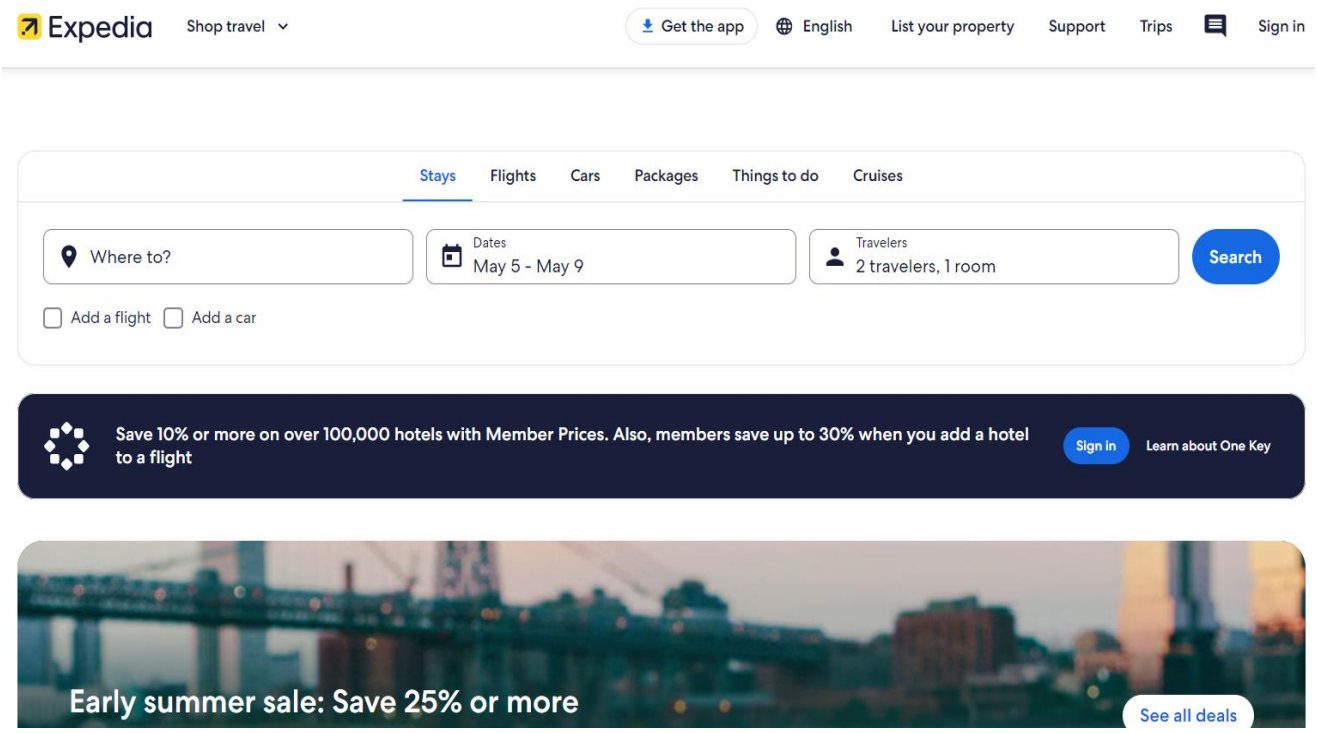


Рисунок 1.1 – Expedia

TripAdvisor [6] є однією з найпопулярніших платформ для подорожей у світі, що допомагає мандрівникам знайти корисну інформацію та почитати відгуки від реальних користувачів про готелі, ресторани, визначні пам'ятки та інші туристичні об'єкти. Заснована в 2000 році, компанія швидко перетворилася на всесвітньовідомий онлайн-ресурс, що акумулює безліч відгуків та оцінок від подорожніх.

Головною особливістю TripAdvisor є величезна база з мільйонів відгуків про місця відпочинку, готелі, ресторани та пам'ятки, написаних реальними людьми. Ця унікальна система відгуків дозволяє користувачам отримати неупереджену інформацію безпосередньо від інших мандрівників, допомагаючи приймати більш

обґрунтовані рішення щодо їхніх майбутніх подорожей.

Окрім відгуків, TripAdvisor також пропонує функції пошуку та бронювання готелів, авіаквитків, турів та оренди автомобілів. Користувачі можуть легко порівнювати ціни та знаходити найкращі пропозиції для своєї подорожі. Платформа також надає детальну інформацію про визначні пам'ятки, заходи та розваги в різних місцях по всьому світу.

Однією з ключових переваг TripAdvisor є її глобальна присутність та локалізація контенту для різних країн та мов. Завдяки цьому користувачі з будь-якої частини світу можуть легко знайти актуальну та корисну інформацію про своє місце призначення. Крім того, платформа постійно вдосконалюється, додаючи нові функції, такі як інтерактивні карти, планувальники подорожей та мобільні додатки.

TripAdvisor [7] широко відомий і визнаний як надійне джерело інформації для мандрівників завдяки своїй величезній спільноті користувачів, що діляться досвідом та думками. Незважаючи на конкуренцію з боку інших туристичних сайтів та додатків, TripAdvisor продовжує зберігати свої лідерські позиції, постійно розширюючи свої можливості та покращуючи користувацький досвід.

Веб-сайт TripAdvisor зображено на рисунку 1.2.

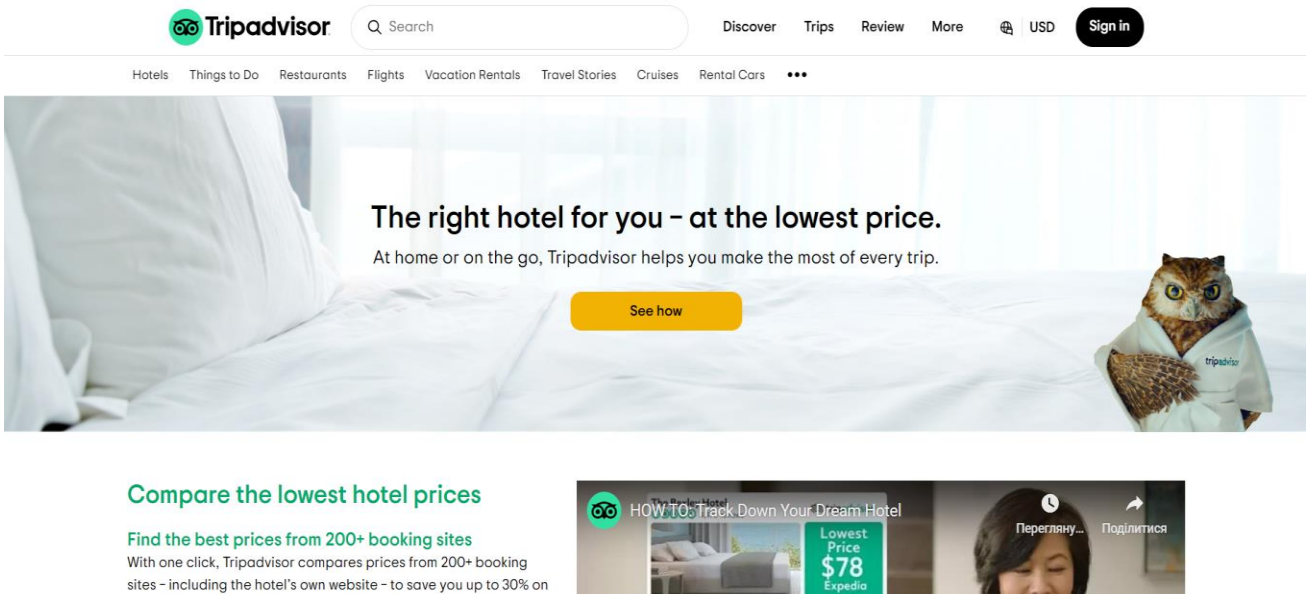


Рисунок 1.2 – TripAdvisor

Kayak є одним з провідних онлайн-сервісів пошуку та порівняння цін на авіаквитки, готелі, оренду автомобілів та інші туристичні послуги. Заснована у 2004 році, ця платформа стала піонером у сфері метапошукових систем для подорожей, допомагаючи користувачам знаходити найвигідніші пропозиції від різних постачальників послуг на одному веб-сайті.

Головною перевагою Kayak є його здатність одночасно сканувати сотні сайтів авіакомпаній, туристичних агентств та систем бронювання, щоб знайти найкращі ціни та варіанти. Це дозволяє користувачам уникнути необхідності перевіряти кожен сайт окремо, економлячи їхній час та зусилля. Інтуїтивний інтерфейс Kayak також спрощує пошук та бронювання, пропонуючи зручні фільтри та параметри.

Окрім пошуку авіаквитків і готелів, Kayak також надає корисні інструменти для планування подорожей, такі як прогнози цін, інформацію про погодні умови, рекомендації щодо найкращих часових періодів для подорожей та багато іншого. Платформа також має мобільні додатки для iOS та Android, що дозволяє користувачам шукати та бронювати з будь-якого місця.

Незважаючи на жорстку конкуренцію з боку інших пошукових систем подорожей, Kayak продовжує утримувати лояльну базу користувачів завдяки своєму зручному інтерфейсу, широкому охопленню постачальників послуг та надійності результатів пошуку. Компанія постійно працює над вдосконаленням своїх алгоритмів та інструментів для забезпечення максимально точних та актуальних даних.

Kayak також активно розширює свої послуги, пропонуючи такі додаткові можливості, як пакетні тури, бронювання екскурсій та заходів, а також партнерські програми з іншими брендами подорожей. Ця диверсифікація допомагає Kayak залишатися конкурентоспроможним та відповідати мінливим потребам сучасних мандрівників в епоху цифрових технологій.

Веб-сайт Kayak продемонстровано на рисунку 1.3.

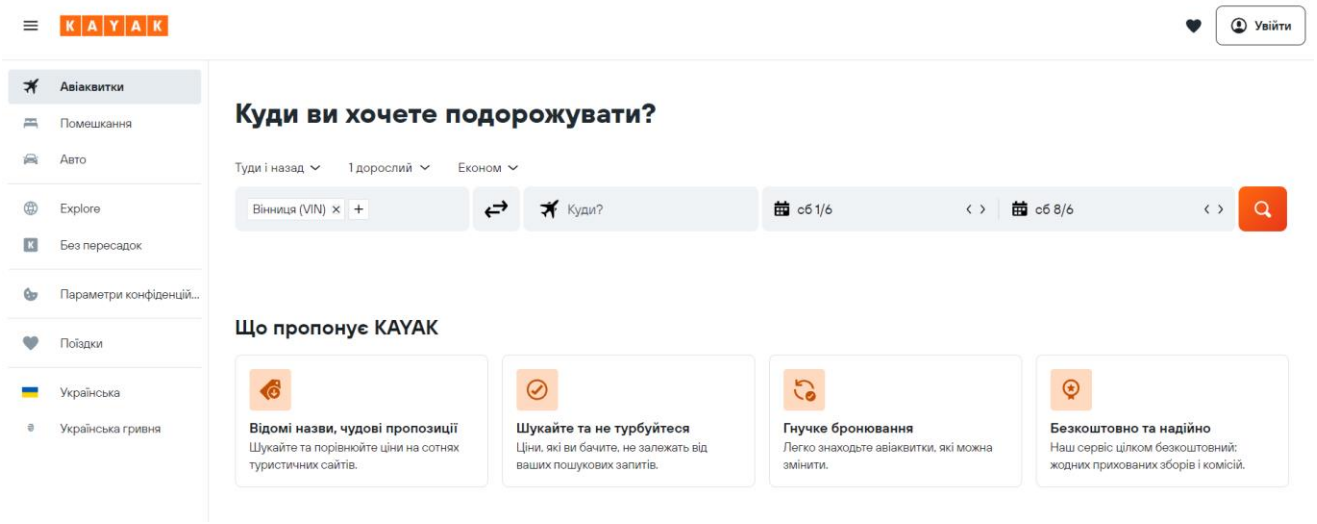


Рисунок 1.3 – Kayak

В результаті аналізу аналогів, було сформовано таблицю 1.1, в якій проведено порівняння характеристик аналогів з власною розробкою.

Таблиця 1.1 – Порівняльний аналіз аналогів

Характеристика	Expedia	TripAdvisor	Kayak	Власна розробка
Бронювання готелів	1	1	0	1
Бронювання авіаквитків	1	0	1	1
Відстеження цін	1	1	1	1
Пошук за фільтрами	0	1	1	1
Сумарний бал	3	3	3	4

Таким чином, власна розробка випереджає аналоги на 25% (100% - ¾*100% = 25%). Отже, власна розробка є актуальною.

1.3 Аналіз методів розв’язання задачі

Для розробки ефективного програмного засобу з планування подорожей необхідно розглянути і проаналізувати різноманітні методи та підходи:

1. Архітектура та масштабованість

- Мікросервісна архітектура може бути ефективним рішенням для забезпечення масштабованості, модульності та незалежного розгортання компонентів [8].

Приклад мікросервісної архітектури наведено на рисунку 1.4.

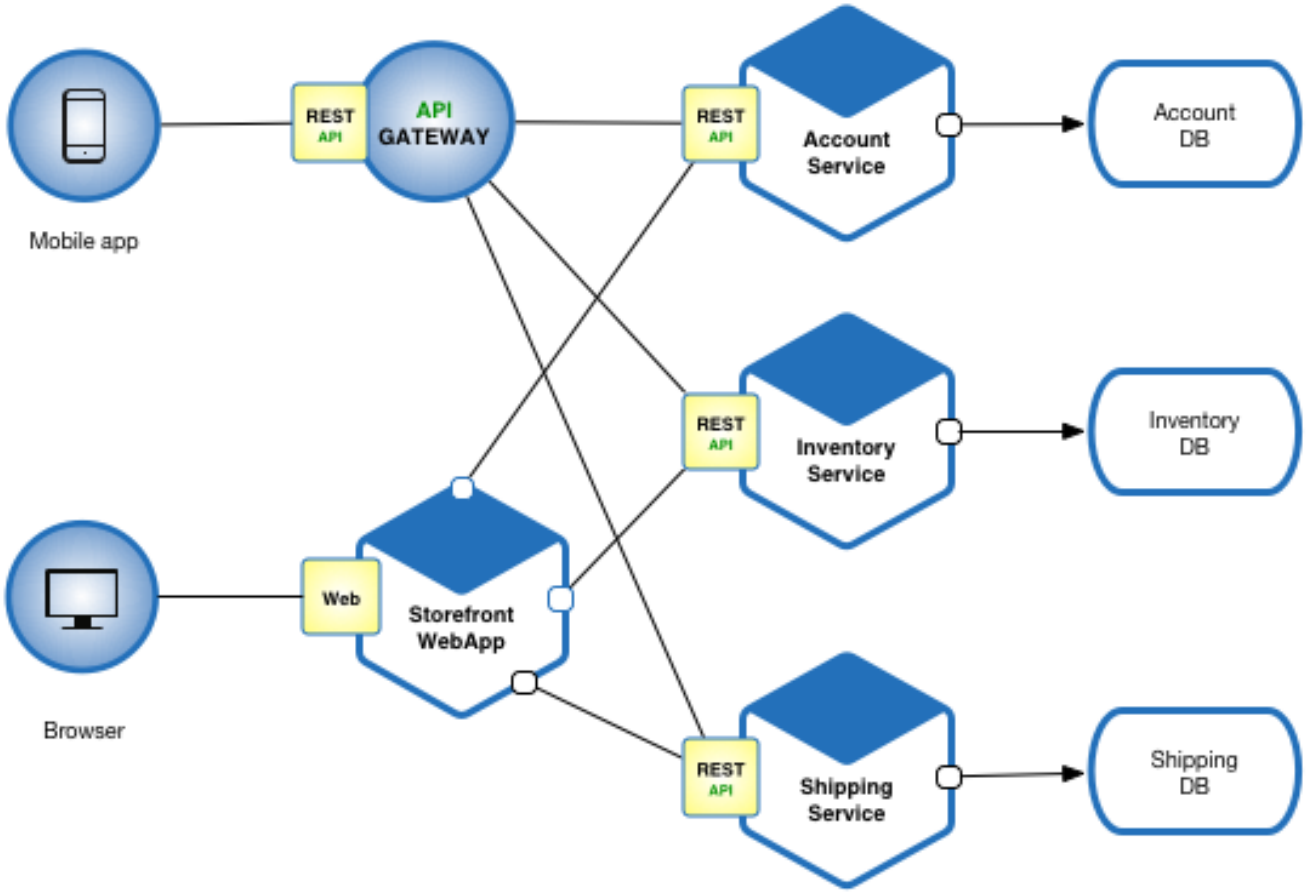


Рисунок 1.4 – Приклад мікросервісної архітектури

- Використання хмарних технологій та контейнеризації (Docker, Kubernetes) полегшить розгортання та масштабування системи.

- Горизонтальне масштабування та розподілена система кешування допоможуть впоратися з пікові навантаження під час високого сезону подорожей (рисунок 1.5).

2. Інтеграція з зовнішніми системами

- Необхідно розробити стратегію інтеграції з численними API постачальників послуг (авіакомпанії, готелі, оренда авто тощо).

- Використання шаблонів проектування, таких як адаптер та фасад, може полегшити інтеграцію з різними системами.

- Важливо забезпечити надійність і відмовостійкість під час запитів до зовнішніх систем.

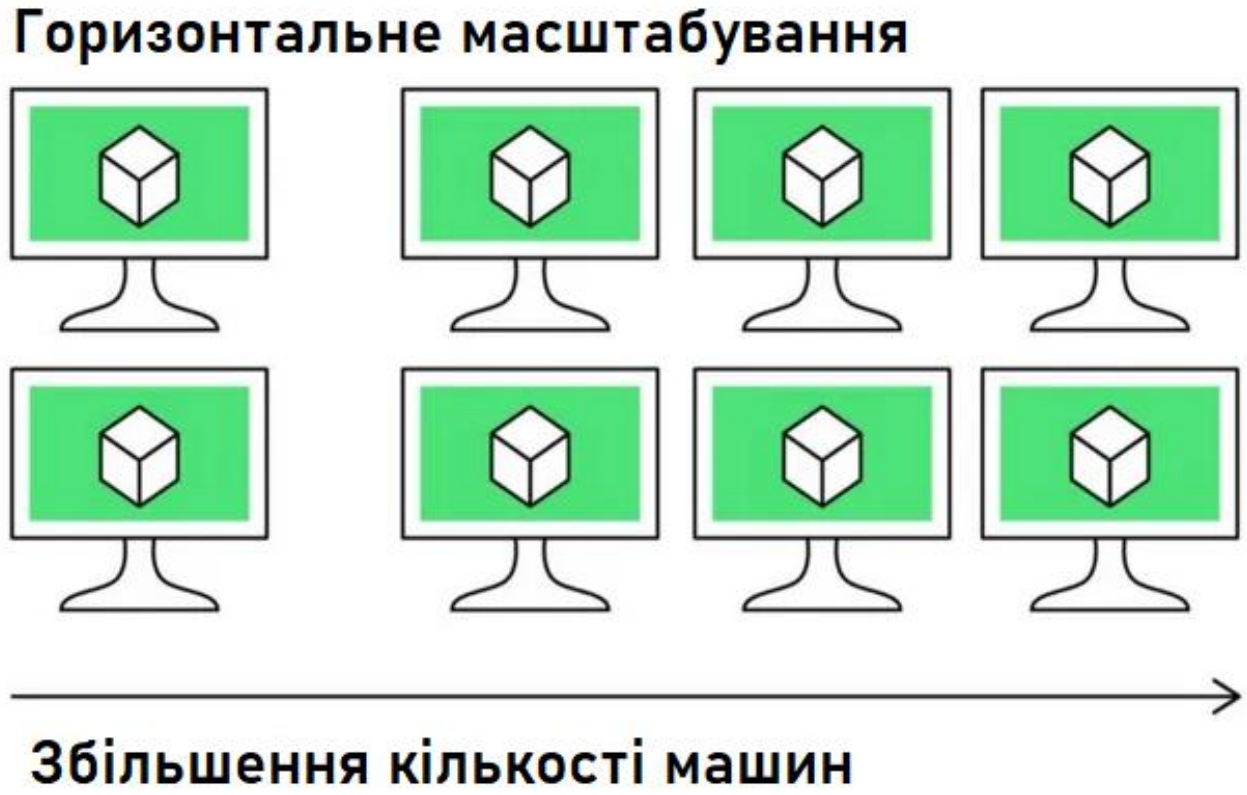


Рисунок 1.5 – Горизонтальне масштабування

3. Обробка та аналіз даних:

- Алгоритми машинного навчання можуть використовуватися для персоналізації рекомендацій та аналізу користувацьких переваг.

- Обробка великих обсягів даних (Big Data) може бути необхідною для аналізу тенденцій та оптимізації цін [9].

4. Безпека та конфіденційність

- Захист персональних даних користувачів та інформації про платежі є критично важливим для забезпечення довіри та дотримання нормативних вимог.

- Регулярні перевірки на безпеку та відповідність нормативним вимогам

повинні бути частиною процесу розробки.

5. Тестування та забезпечення якості

- Автоматизоване тестування (модульне, інтеграційне, системне, UI) має бути невід'ємною частиною циклу розробки [10].

- Безперервна інтеграція та безперервне розгортання допоможуть підвищити якість та швидкість реагування на зміни (рисунок 1.6).

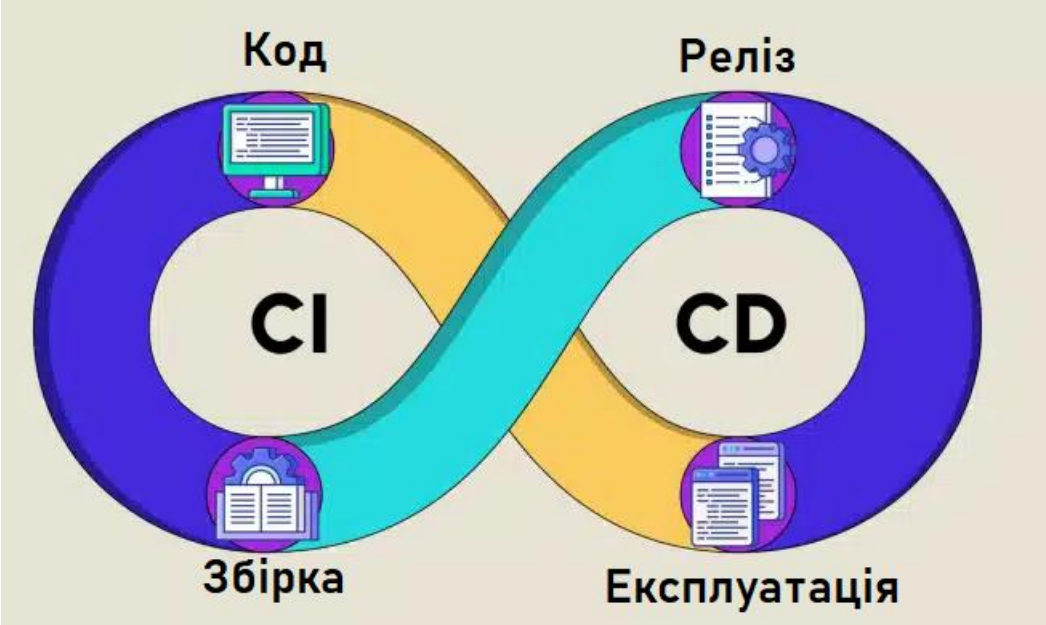


Рисунок 1.6 – Життєвий цикл безперервної інтеграції та безперервного розгортання

- Навантажувальне тестування та тестування продуктивності повинні проводитися для забезпечення стабільності системи під високим навантаженням.

Ретельне планування і врахування цих аспектів у процесі розробки програмного засобу для планування подорожей допоможе забезпечити високу якість, масштабованість, безпеку та конкурентні переваги на ринку туристичних послуг.

1.4 Постановка задач дослідження

В результаті аналізу стану питання розробки програмного засобу з планування подорожей, порівняння можливостей існуючих аналогів та аналізу

методів розв’язання задачі розробки програмних засобів з планування подорожей, було поставлено такі задачі дослідження:

- проаналізувати предметну область;
- провести порівняльний аналіз аналогів;
- провести аналіз методів розв’язання задачі;
- розробити модель застосунку;
- розробити метод формування маршруту подорожі;
- розробити алгоритми роботи програми;
- провести варіантний аналіз засобів реалізації застосунку;
- розробити функціонал застосунку;
- розробити інтерфейс користувача;
- провести аналіз методів тестування;
- сформувати системні вимоги для роботи програмного засобу;
- провести тестування розробленого застосунку.

1.5 Висновки

В цьому розділі, було проведено аналіз стану питання розробки застосунку для планування подорожей, проведено порівняльний аналіз аналогів. Проведено аналіз методів розв’язання задачі та на основі цього проведено постановку задач дослідження.

2 РОЗРОБКА АЛГОРИТМІВ РОБОТИ ЗАСТОСУНКУ

2.1 Аналіз інформаційного забезпечення

При роботі застосунків для планування подорожей обробляються, отримуються та генеруються різні типи даних:

1. Особисті дані користувачів. Це може включати ім'я, електронну пошту, номер телефону, дату народження та іншу інформацію, яку користувач надає при реєстрації або використанні додатку.

2. Дані про подорож. Це може включати інформацію про призначення, дати подорожі, місця проживання, транспортні засоби, бронювання квитків та інше.

3. Фінансові дані. Якщо додаток дозволяє користувачам купувати квитки або бронювати готелі, він може обробляти фінансові дані, такі як номер кредитної картки, дату терміну дії та CVV-код.

4. Дані про пересування. Додатки для планування подорожей можуть відстежувати місцезнаходження користувачів, якщо вони дали на це дозвіл. Це дозволяє додатку надавати рекомендації, засновані на поточному місці розташування користувача.

5. Дані про взаємодію. Це може включати інформацію про те, як користувач взаємодіє з додатком, наприклад, які функції він використовує найчастіше, скільки часу проводить в додатку та інше.

6. Генерування даних. Додатки для планування подорожей також можуть генерувати дані, такі як маршрути подорожей, рекомендації щодо місць для відвідування, прогнози погоди та інше.

2.2 Розробка структури застосунку

Розробка структури файлів розробляється з кількох важливих причин.

Розділення файлів за типами (HTML, CSS, JavaScript, зображення тощо) та функціональністю (шаблони, статичні файли, конфігурації бази даних) забезпечує кращу організацію та модульність проекту. Це полегшує навігацію, пошук та редагування відповідних файлів.

Окремі директорії для HTML-шаблонів, CSS-стилів, JavaScript-скриптів та інших компонентів дозволяють розробникам працювати паралельно над різними частинами проекту без конфліктів.

Структуровані проекти легше масштабувати, оскільки нові функції та компоненти можна додавати у відповідні директорії без порушення існуючої організації.

Було розроблено структуру проекту, яка зображена на рисунку 2.1.

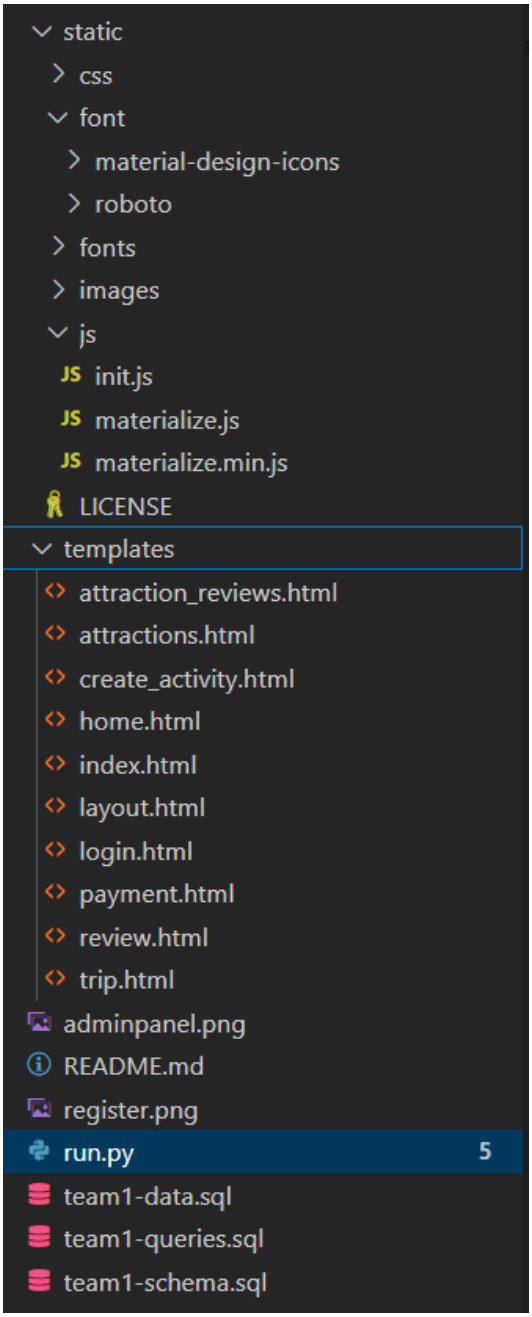


Рисунок 2.1 – Структура проекту

Ця структура є деревоподібним представленням файлів та директорій в проекті. Вона містить наступні елементи:

- js/ :
  - init.js – JavaScript-файл для ініціалізації.
  - materialize.js – бібліотека Materialize для CSS-фреймворкую
  - materialize.min.js – мінімізований варіант бібліотеки Materialize.

templates/:

- attraction_reviews.html – шаблон для відгуків про атракціони.
- attractions.html – шаблон для списку атракціонів.
- create_activity.html – шаблон для створення нової активності.
- home.html – домашня сторінка.
- layout.html – базовий шаблон макета.
- login.html – сторінка входу.
- payment.html – сторінка для обробки платежів.
- review.html – шаблон для відгуків.
- trip.html – шаблон для подорожей.

run.py – головний Python-файл для запуску додатку.

Ця структура описує функціонал, що стосується атракціонів, подорожей та активностей, з можливостями входу, оплати та написання відгуків.

2.3 Розробка моделі застосунку

Розробка моделі системи у вигляді діаграми компонентів має ключове значення для багатьох аспектів розробки програмного забезпечення. Перш за все, вона дозволяє візуалізувати структуру системи. Діаграми компонентів пропонують наочний спосіб представлення того, як система складається з різних компонентів і як вони взаємодіють між собою. Це допомагає розробникам краще розуміти систему в цілому та зменшує ймовірність помилок через непорозуміння.

Крім того, діаграми компонентів полегшують розуміння взаємозв'язків між різними частинами системи. Вони показують, як дані та функціональність передаються між компонентами, що важливо для управління залежностями та

планування тестування. Це допомагає розробникам передбачати, як зміни в одному компоненті можуть вплинути на інші, і уникати непередбачених наслідків.

Спрощення процесу розробки є ще однією перевагою діаграм компонентів. Розбиття системи на менші компоненти дозволяє розробляти та тестувати кожен компонент незалежно. Це полегшує розподіл роботи між командою розробників, зменшує складність завдань та збільшує ефективність розробки.

Діаграми компонентів також сприяють покращенню підтримки та оновлень системи. Вони допомагають визначити, які частини системи можна модифікувати або оновити без впливу на інші частини. Це важливо для підтримки та оновлення системи протягом її життєвого циклу, оскільки зменшує ризик помилок та знижує витрати на оновлення.

Модель застосунку для планування подорожей наведена на рисунку 2.2.

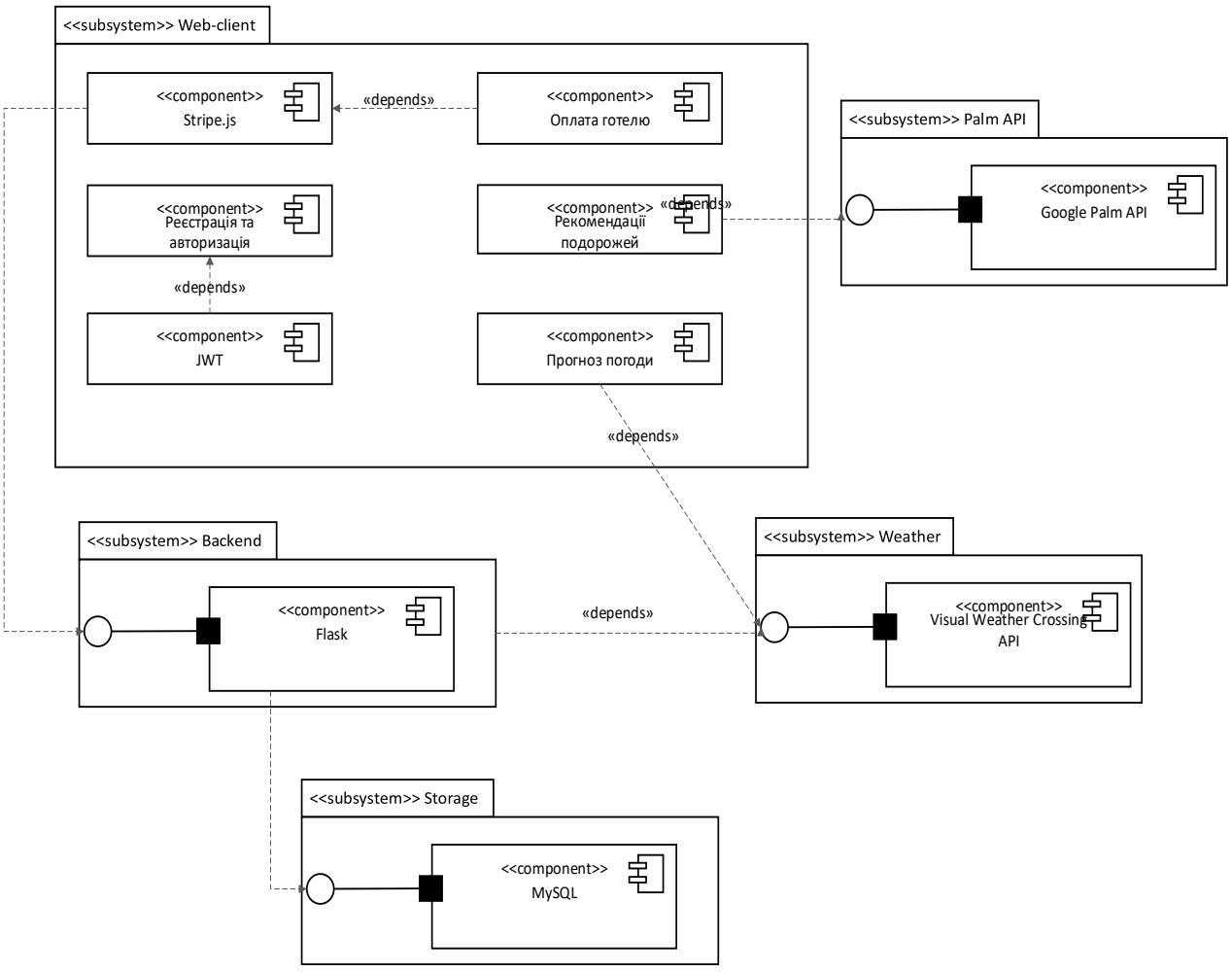


Рисунок 2.2 – Модель системи

Stripe.js використовується для оплати в застосунку, наприклад, при бронюванні готелю.

JWT (JSON Web Tokens) є стандартом індустрії для безпечного передавання інформації між сторонами, особливо корисною в середовищі веб-розробки. Він дозволяє сторонам обмінюватися даними з надійним рівнем автентифікації та авторизації. JWT складається з трьох частин: заголовка, платної частини та підпису.

JWT використовується для безпеки паролів користувача при реєстрації та авторизації.

Для формування рекомендацій щодо подорожей використовується Google Palm API, а для прогнозу погоди – Visual Weather Crossing API.

Для бази даних використовується СКБД MySQL. Фреймворк Flask використовується для серверної частини застосунка.

2.4 Розробка методу формування маршруту подорожі

Метод формування маршруту подорожі формує рекомендації для користувача застосунку за заданими ним початковою та кінцевою точкою подорожі. Також, користувач вказує період часу, протягом якого планує подорожувати.

Метод формування маршруту подорожі складається з таким етапів:

1. Отримати вхідні дані: міста для відвідування, дати подорожі; бюджет/вподобання.

2. Обробити вхідні дані та визначити контекст: витягти ключові слова (міста, активності), визначити загальний контекст (тривала подорож містами Європи).

3. Згенерувати загальну структуру маршруту: визначити послідовність міст та тривалість перебування в кожному, згенерувати рекомендації щодо транспорту між містами.

4. Для кожного міста в маршруті: згенерувати рекомендації щодо проживання, запропонувати відповідні готелі в даному місті, згенерувати рекомендації щодо визначних пам'яток та активностей. Використати дані з туристичних гідів та відгуків, підібрати популярні та затребувані локації.

5. Згенерувати підсумковий опис для кожного міста, об'єднати інформацію

про проживання, транспорт та активності.

6. Об'єднати всі частини для створення кінцевого тексту рекомендацій маршруту: впорядкувати блоки по містах, додати вступні/заключні речення за потреби.

7. Вивести згенерований текст рекомендацій маршруту.

Блок-схема алгоритму роботи методу формування маршруту подорожі наведена на рисунку 2.3.

Рисунок 2.3 – Блок-схема алгоритму роботи методу формування маршруту подорожі

Метод реалізовується в програмному коді шляхом виконання таких кроків:

1. Імпортуються необхідні модулі: google.generativeai як palm для взаємодії з API Google Generative AI, os для взаємодії з операційною системою та dotenv для завантаження змінних середовища.

2. Завантажуються змінні середовища з файлу .env за допомогою функції load_dotenv() з бібліотеки dotenv.

3. Отримується ключ API Google Generative AI (Palm) зі змінної середовища PALM_API_KEY.

4. Налаштовується бібліотека palm з отриманим ключем API за допомогою функції palm.configure(api_key=palm_api_key).

5. Створюється екземпляр моделі GenerativeModel з назвою "gemini-pro", яка є однією з моделей, доступних у Google Generative AI.

6. Визначається функція generate_itinerary, яка приймає п'ять аргументів: source (місце відправлення), destination (місце призначення), start_date (дата початку), end_date (дата завершення) та no_of_day (кількість днів).

7. Усередині функції generate_itinerary створюється рядок prompt, який містить інструкцію для моделі згенерувати персоналізований маршрут подорожі на основі наданих аргументів.

8. Викликається метод model.generate_content(prompt), який надсилає запит до API Google Generative AI з заданим prompt. Результат, який є згенерованим маршрутом подорожі, зберігається в змінній response.

9. Функція generate_itinerary повертає текст згенерованого маршруту подорожі як рядок за допомогою response.text.

Виконання цих кроків дозволяє генерувати персоналізовані маршрутів подорожі на основі заданих параметрів, таких як місце відправлення, місце призначення, дати та кількість днів. Згенеровані маршрути можна потім відображати або зберігати для подальшого використання.

Блок-схема алгоритму реалізації методу формування маршруту подорожі наведено на рисунку 2.4.

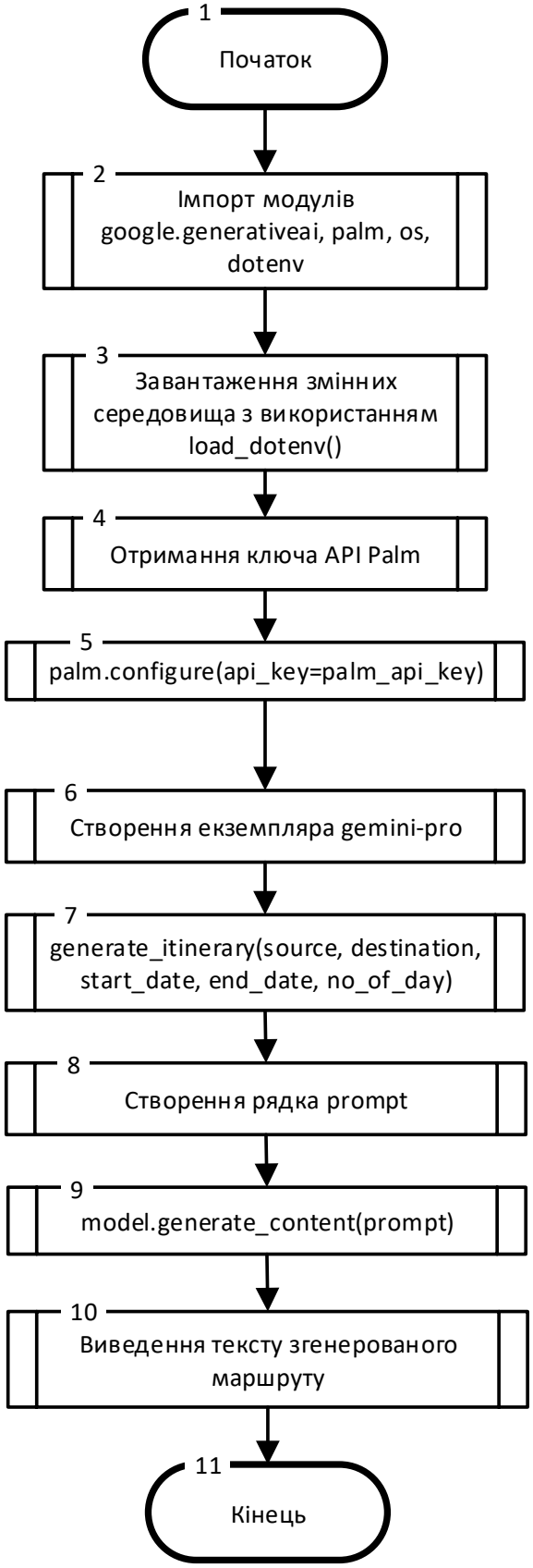


Рисунок 2.4 – Блок-схема алгоритму реалізації методу формування маршруту подорожі

2.5 Розробка алгоритмів роботи програми

Програмний засіб з планування подорожей повинен виконувати такі функції:

1. **Введення даних про подорож.** Користувач вводить початкове та кінцеве місто. Система зберігає ці дані та використовує їх для побудови маршруту.
2. **Побудова маршруту**. Система генерує маршрут між двома містами, враховуючи можливі варіанти шляху. Маршрут може бути оптимізований за критеріями, такими як найшвидший, найкоротший або мальовничий шлях.
3. **Візуалізація маршруту**. Система відображає згенерований маршрут на інтерактивній карті. Користувач може переглядати деталі маршруту, включаючи відстані та час у дорозі.
4. **Пошук визначних місць.** Система знаходить визначні місця вздовж маршруту, використовуючи зовнішні API або базу даних.
5. **Відображення визначних місць**. Система відображає інформацію про визначні місця, включаючи опис, фотографії, години роботи та вартість входу. Користувач може додавати визначні місця до свого маршруту.
6. **Налаштування рекомендацій**. Користувач може фільтрувати визначні місця за категоріями (історичні пам'ятки, природні об'єкти, музеї тощо). Система дозволяє користувачу додавати та видаляти визначні місця зі свого маршруту.
7. **Пошук готелів.** Система інтегрується з сервісами бронювання готелів для пошуку доступних варіантів вздовж маршруту.
8. **Відображення готелів**. Система відображає список доступних готелів з інформацією про ціну, рейтинг, зручності та відстань від визначних місць.
9. **Бронювання**. Користувач може обрати готель та здійснити бронювання безпосередньо через веб-застосунок. Система зберігає інформацію про бронювання та надає користувачу підтвердження.
10. **Введення витрат.** Користувач може вводити витрати на різні аспекти подорожі, такі як транспорт, проживання, їжа та розваги.
11. **Пошук прогнозу погоди.** Система інтегрується з сервісами прогнозу погоди для отримання актуальної інформації про погоду на період подорожі.
12. **Відображення прогнозу погоди.** Система відображає прогноз погоди для

всіх міст маршруту, включаючи температуру, опади та інші метеорологічні дані.

Для виконання цих вимог, було розроблено структуру класів, яку оформлено у вигляді діаграми класів на рисунку 2.5.

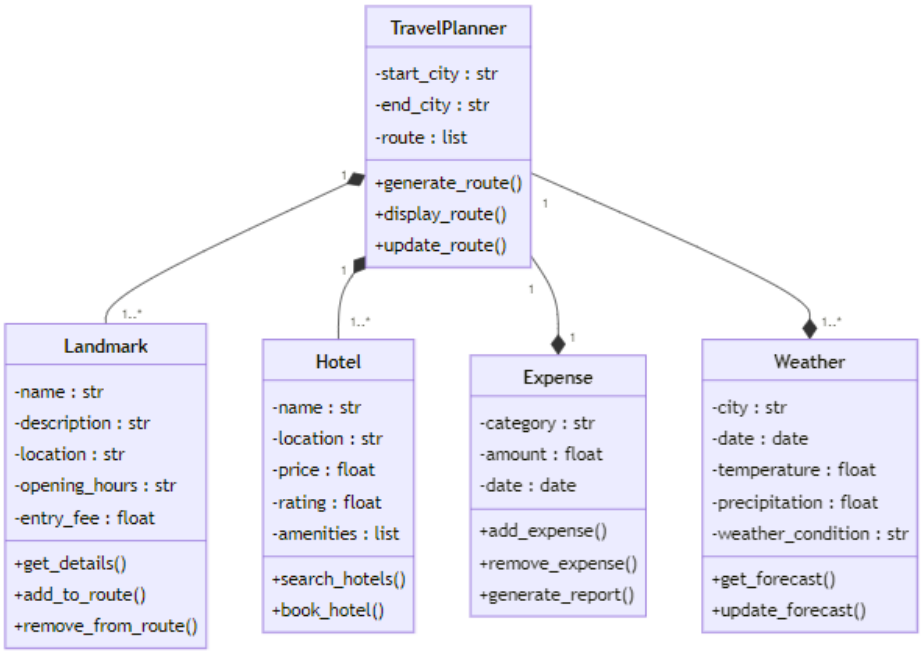


Рисунок 2.5 – Діаграма класів програмного засобу для планування подорожей

Одне планування подорожі (TravelPlanner) може включати кілька пам'яток (Landmark), які необхідно відвідати. Пам'ятки є невід'ємною частиною плану подорожі, тому тут використовується відношення композиції. Якщо планування подорожі видаляється, то всі пов'язані з ним пам'ятки також повинні бути видалені.

Одне планування подорожі (TravelPlanner) може включати кілька готелів (Hotel), в яких необхідно зупинитися. Готелі також є невід'ємною частиною плану подорожі, тому тут використовується відношення композиції. Якщо планування подорожі видаляється, то всі пов'язані з ним готелі також повинні бути видалені.

Одне планування подорожі (TravelPlanner) має одні загальні витрати (Expense), які необхідно відстежувати. Витрати не є невід'ємною частиною плану подорожі, тому тут використовується відношення агрегації. Якщо планування подорожі видаляється, то витрати можуть існувати окремо (наприклад, для

звітності або аналізу).

Одне планування подорожі (TravelPlanner) може перевіряти погодні умови (Weather) для одного або кількох міст, які входять до маршруту. Погодні умови не є невід'ємною частиною плану подорожі, тому тут використовується відношення агрегації. Якщо планування подорожі видаляється, то дані про погоду можуть існувати окремо (наприклад, для інших цілей).

Було розроблено блок-схему алгоритму процесу реєстрації та авторизації (рисунок 2.5).

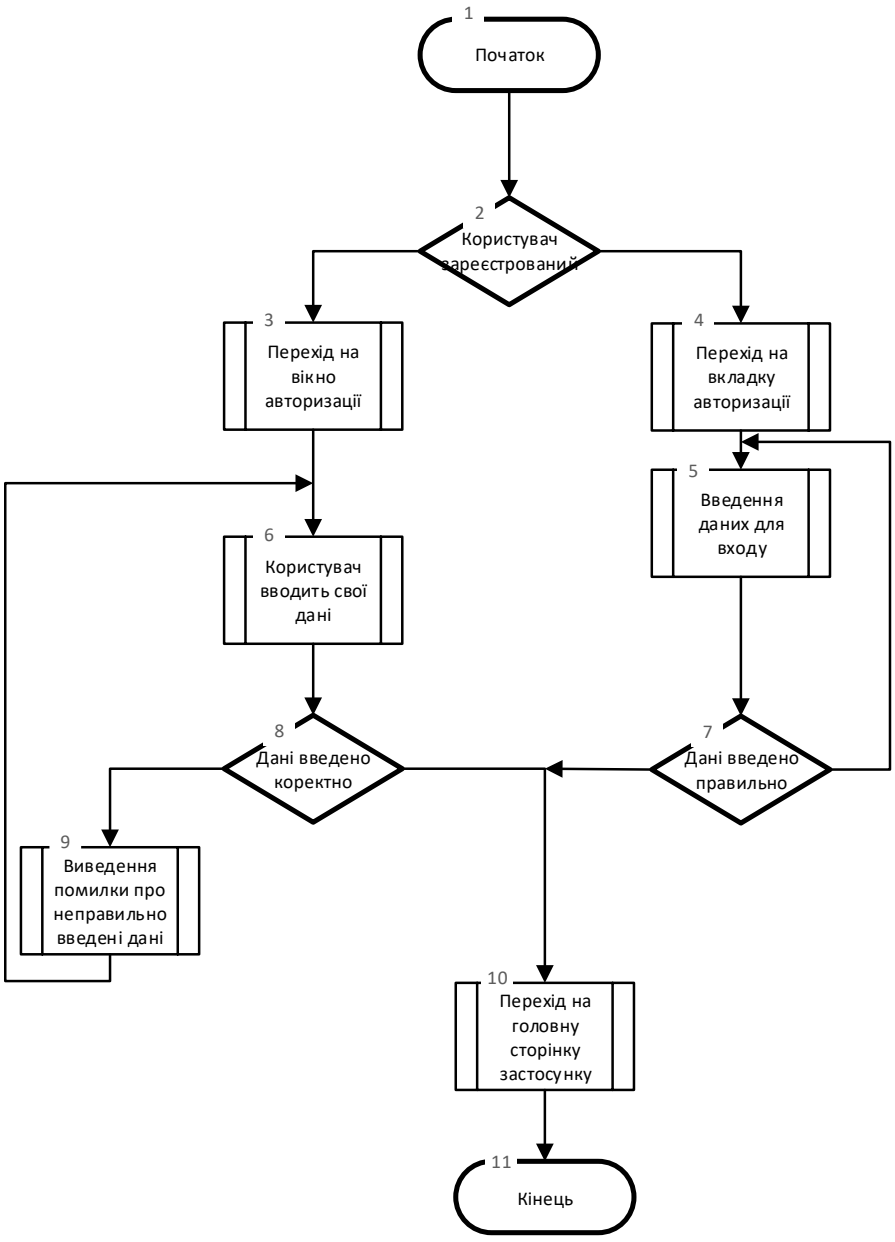


Рисунок 2.5 – Блок-схему процесу реєстрації та авторизації

Під час реєстрації та авторизації проводиться перевірка на коректне введення логіну та пароля, у разі неправильних даних виводиться відповідне повідомлення.

Розроблено блок-схему алгоритму роботи процесу формування прогнозу погоди, який наведено на рисунку 2.6. Для формування прогнозу погоди використовується Visual Weather Crossing API, який збирає дані про погоду, аналізує її, формує таблицю з прогнозом погоди та демонструє її користувачу.

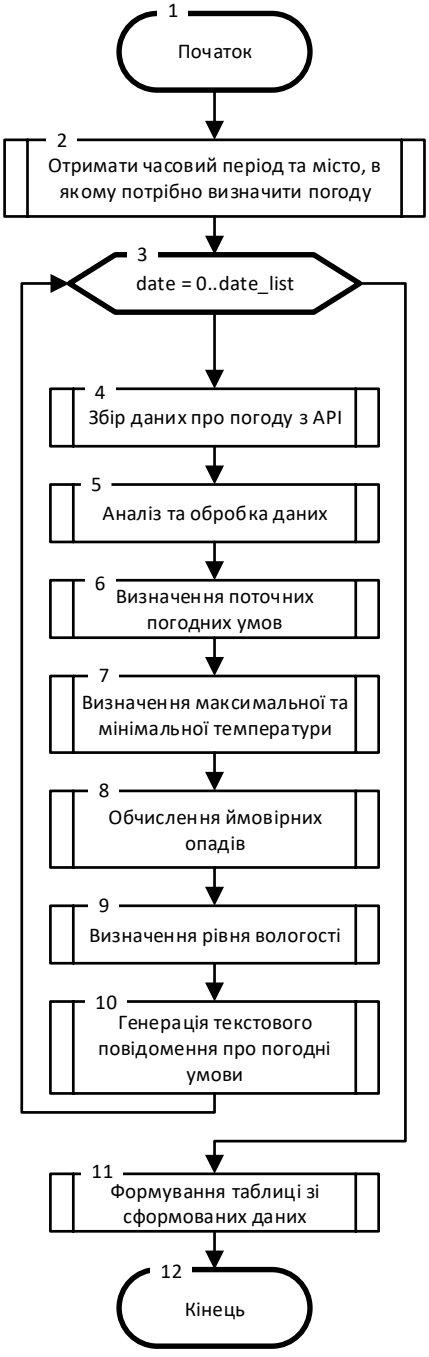


Рисунок 2.6 – Блок-схема алгоритму процесу формування прогнозу погоди

Було розроблено блок-схему алгоритму бронювання готелю. Під час цього процесу, користувач вводить початкові дані для пошуку, такі як країна і місто. Після цього він також здійснює вибір серед готелів та номерів в межах готелю. Якщо він пройшов усі процеси вибору успішно, він резервує номер в готелі (рисунок 2.7).

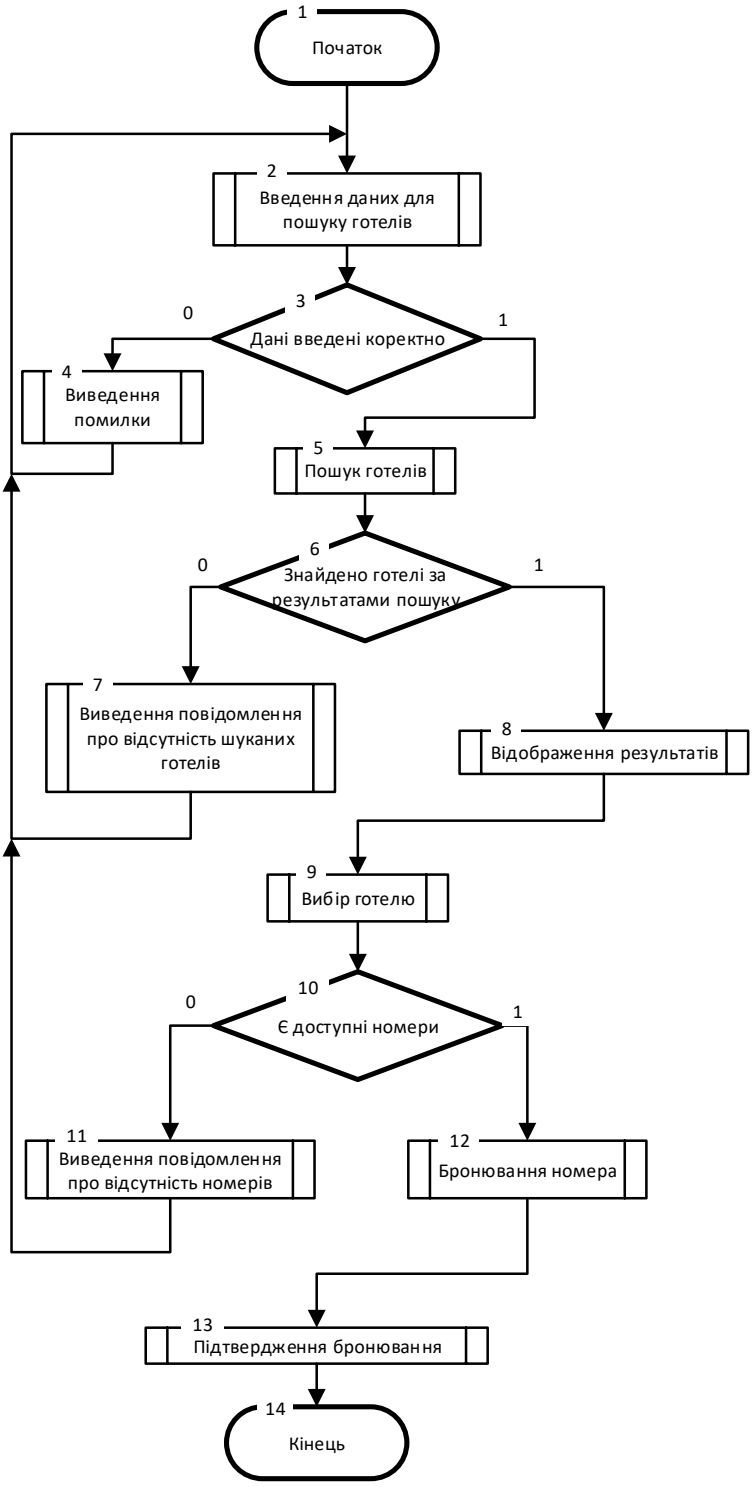


Рисунок 2.7 – Блок-схема алгоритму процесу бронювання номеру в готелі

Програмний засіб повинен генерувати маршрут і надавати рекомендації щодо визначних місць не більше ніж за 12 секунд після введення початкових та кінцевих пунктів.

Програмний засіб повинна підтримувати одночасну роботу не менше 100 користувачів без значного погіршення продуктивності. Система повинна мати можливість горизонтального масштабування для обробки збільшеної кількості користувачів.

Програмний засіб повинна мати середній час безвідмовної роботи (MTBF) не менше 98% на рік. У разі збою програма повинна автоматично відновлюватися без втрати даних користувачів.

Особисті дані користувачів повинні бути шифровані як у процесі передачі (SSL/TLS), так і при зберіганні. Система повинна відповідати вимогам GDPR та інших міжнародних стандартів захисту даних.

Програмний засіб повинна використовувати безпечну аутентифікацію (OAuth, JWT) для доступу користувачів.

Інтерфейс повинен бути інтуїтивно зрозумілим та легким у використанні для користувачів будь-якого рівня технічної підготовки.

Дизайн повинен бути адаптивним для роботи на різних пристроях (десктопи, планшети, мобільні телефони).

2.6 Висновки

В цьому розділі, було проведено аналіз інформаційного забезпечення, розроблено структуру застосунку, розробено модель застосунку, розроблено алгоритми роботи програми.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Варіантний аналіз та вибір мови програмування

JavaScript [11] – це універсальна сценарна мова, яка спочатку була розроблена для створення інтерактивних веб-сторінок. Вона дозволяє додавати динамічну функціональність на веб-сайти, таку як анімовані графічні елементи, інтерактивні форми, оновлення контенту в реальному часі та багато іншого. JavaScript працює на стороні клієнта, що означає, що код виконується безпосередньо у веб-браузері користувача, а не на веб-сервері.

Архітектура JavaScript наведена на рисунку 3.1.

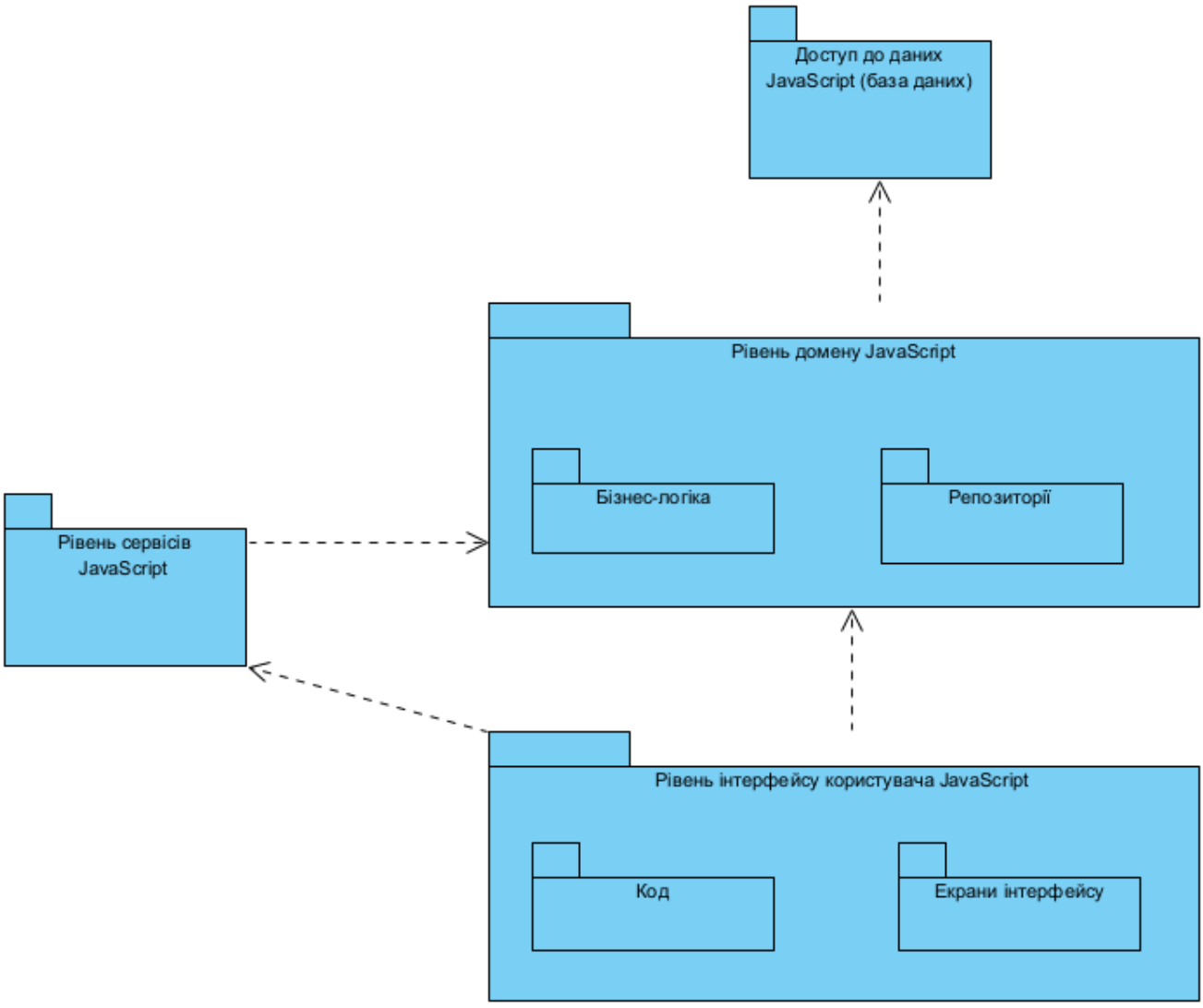


Рисунок 3.1 – Архітектура JavaScript

JavaScript – це об'єктно-орієнтована мова програмування, яка використовує прототипне успадкування. Вона має синтаксис, схожий на Java, але має іншу модель об'єктної системи, засновану на прототипах, а не на класах. JavaScript підтримує різні парадигми програмування, такі як функціональне та імперативне програмування.

Крім використання у веб-браузерах, JavaScript також може працювати на серверній стороні за допомогою таких платформ, як Node.js, що дозволяє створювати повноцінні веб-додатки на одній мові програмування. JavaScript широко використовується в розробці одностраничних додатків (SPA) та фреймворків, таких як React, Angular та Vue.js.

У сучасній веб-розробці JavaScript відіграє життєво важливу роль. Він дозволяє створювати багатий та інтерактивний користувацький інтерфейс, маніпулювати DOM (Об'єктна модель документа), відправляти та отримувати дані з сервера та обробляти події у браузері. JavaScript також має велику спільноту та величезну екосистему бібліотек та інструментів, які полегшують розробку.

PHP (Hypertext Preprocessor) [12] – це поширена скриптова мова загального призначення, яка широко використовується для створення динамічних веб-сторінок та веб-додатків. Код PHP вбудовується безпосередньо в HTML-код, що робить його ідеальним для розробки веб-сайтів. Він працює на стороні сервера, тобто код PHP виконується на веб-сервері, а результати відправляються назад клієнту у вигляді звичайного HTML-коду.

PHP є мовою з відкритим вихідним кодом і має велику спільноту розробників, які сприяють її розвитку та підтримці. Він підтримує безліч баз даних, таких як MySQL, PostgreSQL, Oracle та інші, що робить його потужним інструментом для створення динамічних веб-додатків, заснованих на роботі з базами даних.

Однією з найважливіших переваг PHP є його відносна простота у вивченні. Його синтаксис нагадує C, C++ та Java, що полегшує перехід для розробників з досвідом у цих мовах. PHP також має безліч вбудованих функцій для виконання поширених задач, таких як робота з файлами, обробка форм, відправка електронної пошти та багато іншого.

PHP підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам створювати більш модульний і легко підтримуваний код. Він також має велику екосистему готових бібліотек і фреймворків, таких як Laravel, Symfony, CodeIgniter та інших, які спрощують розробку складних веб-додатків.

Архітектура PHP наведена на рисунку 3.2.

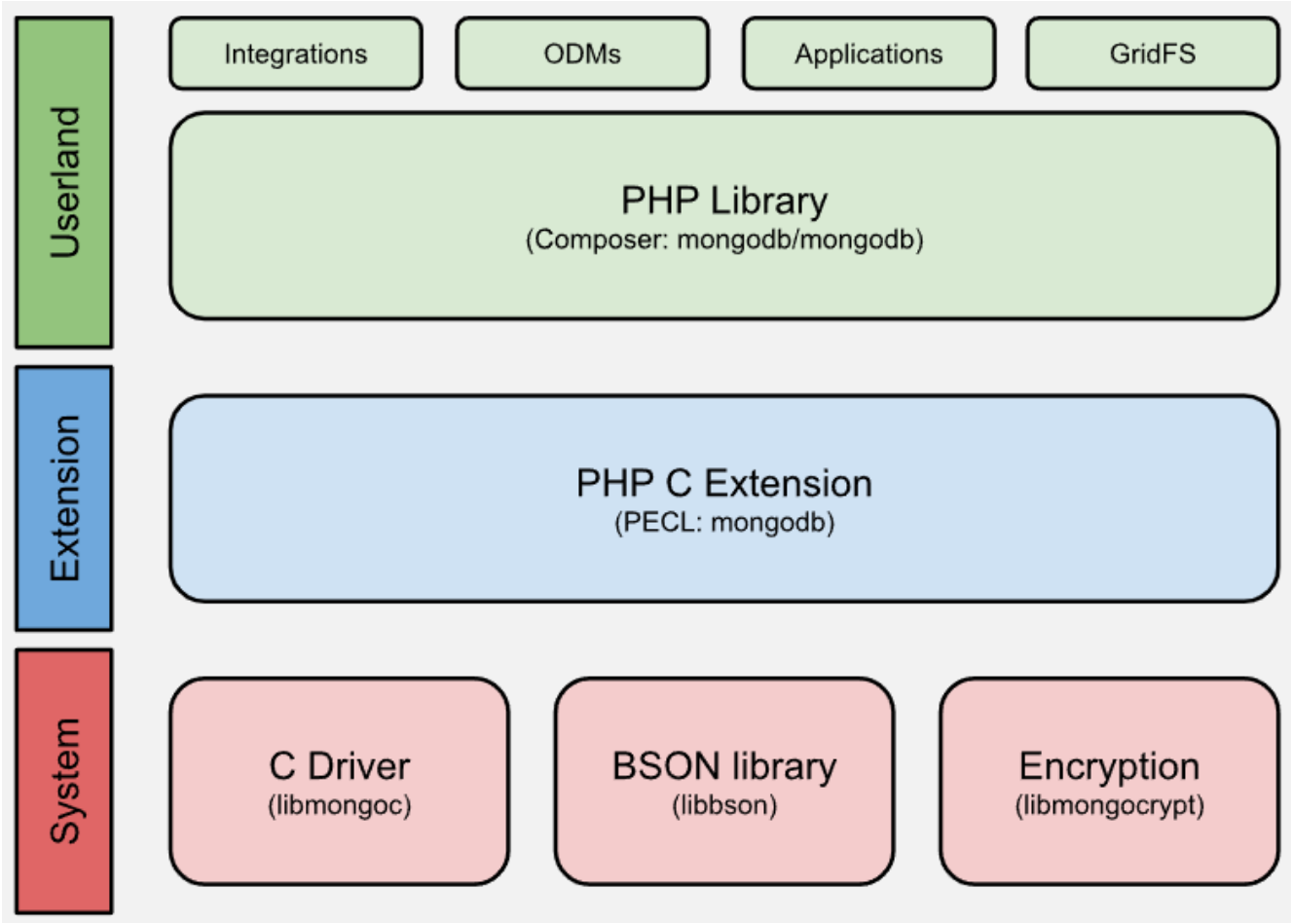


Рисунок 3.2 – Архітектура PHP

Завдяки своїй гнучкості, продуктивності та величезній спільноті, PHP залишається однією з найпопулярніших мов для розробки веб-додатків, особливо для створення блогів, інтернет-магазинів, систем керування контентом (CMS) та багатьох інших типів веб-сайтів.

Python [13] – це універсальна мова програмування високого рівня з відкритим вихідним кодом. Вона відрізняється читабельним і лаконічним синтаксисом, що робить її легкою для вивчення та використання. Python є інтерпретованою мовою,

що означає, що код виконується безпосередньо без попередньої компіляції.

Python підтримує кілька парадигм програмування, включаючи структурне, об'єктно-орієнтоване, функціональне та імперативне програмування. Це робить його гнучким та придатним для широкого спектру завдань, таких як веб-розробка, наукові обчислення, аналіз даних, машинне навчання, скриптинг та автоматизація.

Одна з ключових особливостей Python – його величезна стандартна бібліотека, яка забезпечує багатий набір модулів і функцій для виконання різноманітних завдань. Це дозволяє економити час розробки, оскільки немає необхідності писати весь код з нуля.

Python має активну та зростаючу спільноту розробників, що сприяє його безперервному вдосконаленню та появі нових бібліотек і фреймворків. Популярні фреймворки, такі як Django для веб-розробки, NumPy та SciPy для наукових обчислень, і TensorFlow для машинного навчання, написані на Python.

Архітектура Python наведена на рисунку 3.3.

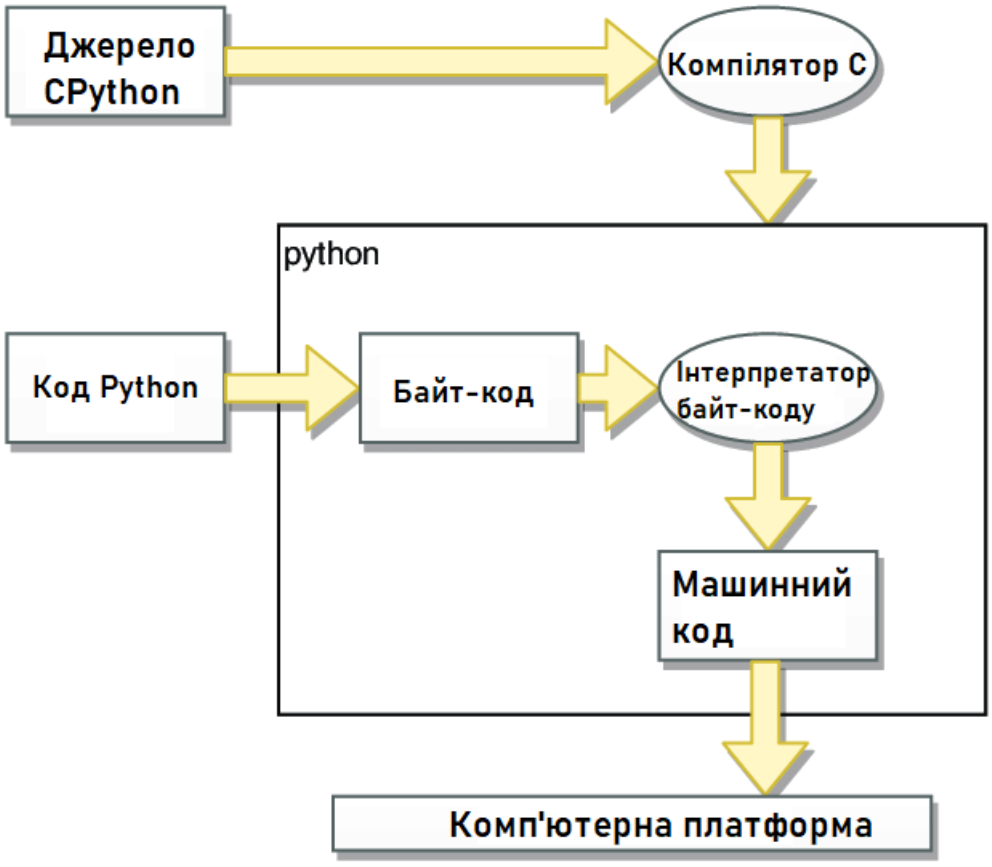


Рисунок 3.3 – Архітектура Python

Python широко використовується у різних галузях, таких як веб-розробка, наука про дані, штучний інтелект, фінанси, автоматизація, безпека та багато інших. Він високо цінується за свою простоту, читабельність та продуктивність, особливо для швидкої розробки прототипів та створення мінімальних робочих зразків.

Python також активно використовується в освіті завдяки своїй простоті та чіткому синтаксису, що робить його доступним для початківців у програмуванні.

Сформуємо таблицю 3.1 порівняння характеристик описаних мов програмування.

Таблиця 3.1 – Порівняння характеристик мов програмування

Функціональність	**Java**	**Python**	**PHP**
Наявність фреймворків для веб-розробки	1	1	1
Підтримка багатопоточності	1	1	1
Інтеграція з API для бронювання готелів	1	1	0
Підтримка кешування для підвищення продуктивності застосунку	1	1	1
Інтеграція з сервісами для аналітики взаємодії користувачів із застосунком	1	1	1
Інтеграція з платіжними системами	0	1	1
Підтримка адаптації під мобільні пристрої	1	1	0
Наявність інструментів для тестування	1	1	1
Сумарний коефіцієнт	7	8	6

Python є чудовим вибором для розробки програмного засобу з планування подорожей з кількох причин.

1. Python має лаконічний та зрозумілий синтаксис, що робить його ідеальним для створення проектів будь-якого розміру. Це також полегшує подальшу підтримку та розширення програмного забезпечення.

2. Python має потужну стандартну бібліотеку, яка пропонує модулі для роботи з веб-скрапінгом, обробкою даних, взаємодією з API та багато іншого. Це може значно спростити інтеграцію з такими сервісами, як пошук авіаквитків, бронювання готелів, пошук маршрутів тощо.

3. Для більш складних функцій планування подорожей, таких як оптимізація маршрутів, рекомендаційні системи та аналітика, Python пропонує потужні бібліотеки для машинного навчання, такі як TensorFlow, Scikit-Learn тощо.

4. Python має кілька популярних веб-фреймворків, таких як Django та Flask, які ідеально підходять для створення веб-додатків або веб-сайтів для планування подорожей.

5. Python є кроссплатформеною мовою, що робить його ідеальним для створення додатків, які працюють на різних операційних системах.

На відміну від JavaScript, який в основному використовується для веб-розробки, або PHP, який зазвичай використовується для серверної частини веб-додатків, Python пропонує більш широкий діапазон можливостей та гнучкість для розробки різноманітних типів програмного забезпечення, включаючи додатки для планування подорожей.

Отже, було обрано Python для розробки програмного засобу з планування подорожей.

3.2 Варіантний аналіз та вибір середовища розробки

PyCharm [14] – це потужне інтегроване середовище розробки (IDE) для Python, створене компанією JetBrains. Воно пропонує багатий набір інструментів та функцій для підвищення продуктивності та зручності розробки Python. PyCharm доступний у двох версіях: безкоштовній Community Edition та платній Professional

Edition з розширеними можливостями.

Одна з головних переваг PyCharm – це його вбудоване розуміння коду Python. Він забезпечує інтелектуальне автодоповнення коду, аналізує помилки та попереджує про потенційні проблеми під час написання коду. Це допомагає зменшити кількість помилок та прискорити процес програмування. PyCharm також підтримує рефакторинг коду, що полегшує внесення змін у великі проекти.

PyCharm має потужні інструменти для налагодження та тестування коду. Він дозволяє встановлювати точки зупинки, відстежувати значення змінних, крокувати код та взаємодіяти з налагоджувальним процесом багатьма іншими способами. Інтеграція з популярними фреймворками для тестування, такими як unittest, pytest та doctest, спрощує процес написання та виконання тестів.

Середовище розробки добре інтегрується з різними системами контролю версій, такими як Git, Mercurial та Subversion. Це дозволяє легко відстежувати зміни в коді, створювати гілки, виконувати злиття та вирішувати конфлікти прямо з PyCharm.

PyCharm також забезпечує чудову підтримку веб-розробки з Python, включаючи інтеграцію з популярними веб-фреймворками, такими як Django, Flask та Pyramid. Він забезпечує можливості налагодження веб-додатків, візуалізацію шаблонів та багато іншого.

Середовище розробки має багато корисних інструментів для підвищення продуктивності, наприклад віртуальні середовища, консолі Python, інструменти для форматування коду відповідно до популярних стилів кодування та багато іншого. PyCharm також пропонує шаблони проектів для різних типів Python-проектів, що полегшує початок роботи.

Отже, PyCharm є потужним та повнофункціональним середовищем розробки для Python, яке пропонує всі необхідні інструменти та функції для професійного програмування на Python. Незважаючи на свою складність, він забезпечує зручний та інтуїтивно зрозумілий інтерфейс користувача.

Інтерфейс PyCharm продемонстровано на рисунку 3.4.

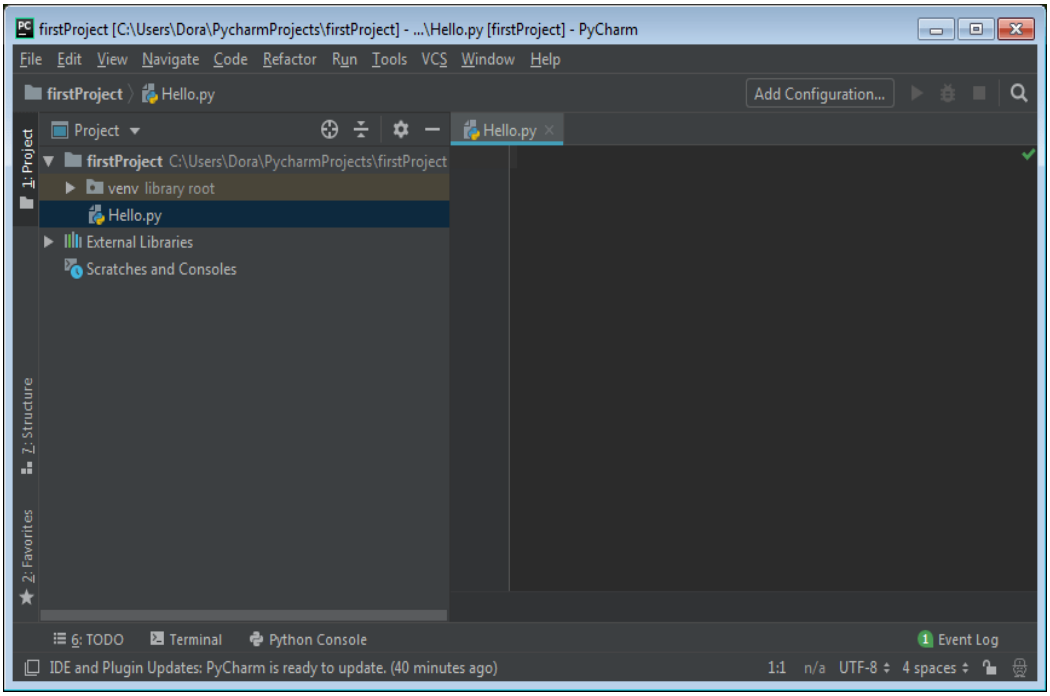


Рисунок 3.4 – PyCharm

Visual Studio Code [15] (VS Code) – це безкоштовний та кросплатформний редактор вихідного коду з відкритим вихідним кодом, розроблений Microsoft. Він набув величезної популярності завдяки своїй зручності, продуктивності та гнучкості.

VS Code має елегантний та інтуїтивно зрозумілий інтерфейс користувача, який легко настроїти відповідно до ваших потреб. Він підтримує численні мови програмування, включаючи Python, завдяки своєму розширеному механізму плагінів. Офіційне розширення Microsoft Python пропонує повну підтримку Python у VS Code.

Одна з головних переваг VS Code – це його швидкість та продуктивність. Він запускається майже миттєво та забезпечує плавну роботу навіть з великими кодовими базами. VS Code використовує власний рушій рендерингу для максимальної оптимізації віртуальної машини.

Редактор підтримує інтелектуальне автодоповнення коду, виділення синтаксису для різних мов програмування та вбудований термінал. Він також пропонує корисні функції, такі як налагодження, інтеграція з системами контролю версій (Git), фрагменти коду та багато іншого.

VS Code має потужну екосистему розширень, що дозволяє користувачам додавати нові функції та інструменти відповідно до своїх потреб. Розширення можуть надавати додаткову підтримку мов програмування, тем оформлення, інструменти розробки та багато іншого.

Для розробки на Python VS Code пропонує всі необхідні функції, включаючи налагодження, перевірку синтаксису, автоматичне форматування коду, інтеграцію з популярними Python-фреймворками, такими як Django та Flask, та багато іншого.

VS Code також підтримує інтеграцію з Jupyter Notebook, що робить його чудовим вибором для аналізу даних та наукових обчислень з Python. Розширення дозволяють запускати, редагувати та налагоджувати Jupyter Notebook прямо у VS Code.

Отже, Visual Studio Code є потужним, гнучким та легким у використанні середовищем розробки для Python та багатьох інших мов програмування. Його відкритий вихідний код, активна спільнота та багата екосистема розширень роблять його одним з найпопулярніших рішень для розробників.

Інтерфейс Visual Studio Code продемонстровано на рисунку 3.5.

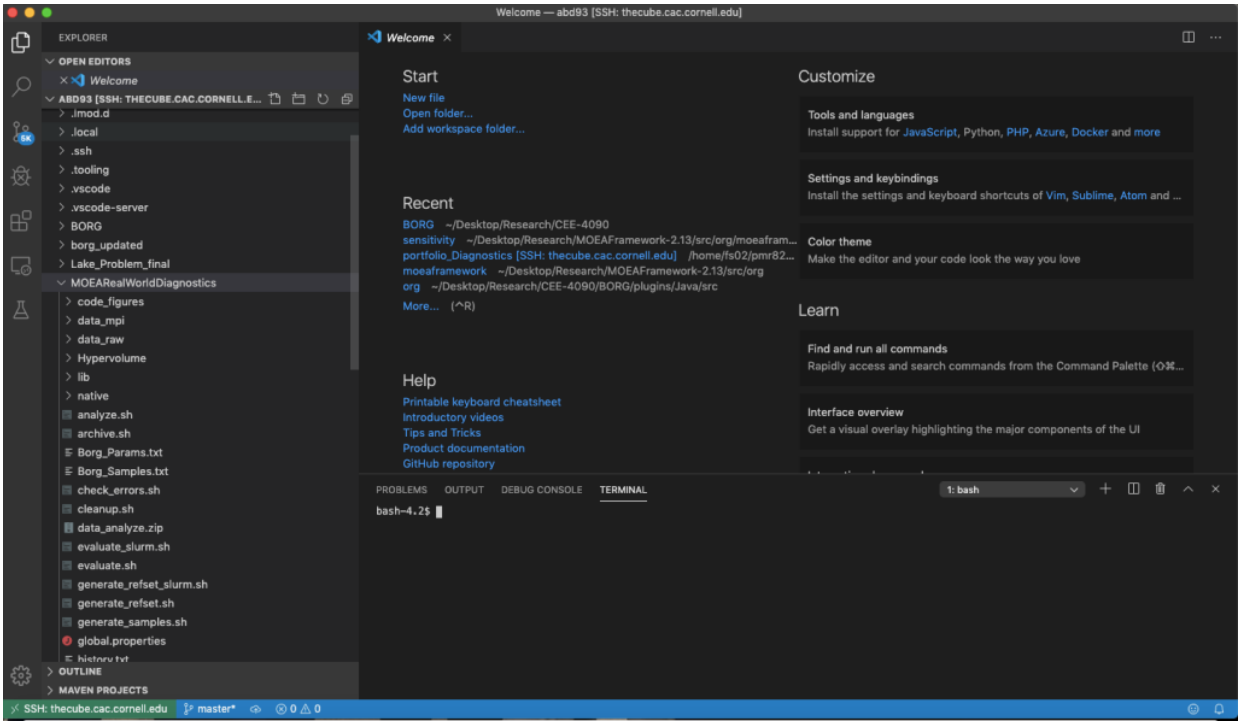


Рисунок 3.5 – Visual Studio Code

PyDev [16] – це потужне розширення для Eclipse IDE, яке забезпечує всебічну підтримку для розробки на Python. Це відкрите та безкоштовне розширення, що робить його доступним для будь-якого розробника Python.

Одна з головних переваг PyDev – це його глибока інтеграція з Eclipse. Після встановлення PyDev стає невід'ємною частиною середовища розробки Eclipse, забезпечуючи безперебійну роботу з Python прямо в ньому. Це економить час та зусилля, оскільки немає потреби переключатися між різними інструментами.

PyDev пропонує всі необхідні функції для ефективного програмування на Python, включаючи автодоповнення коду, виділення синтаксису, навігацію по коду та перевірку помилок. Він також підтримує рефакторинг коду, що допомагає зберігати якість та підтримуваність проекту під час його розвитку.

Налагодження програм є однією з ключових функцій PyDev. Воно дозволяє встановлювати точки зупинки, крокувати код, переглядати значення змінних та взаємодіяти з налагоджувальним процесом різними способами. Це допомагає виявляти та виправляти помилки більш ефективно.

PyDev добре інтегрується з популярними фреймворками для тестування Python, такими як unittest та pytest. Він дозволяє легко запускати тести та відстежувати їх результати безпосередньо в Eclipse, що полегшує процес розробки з тестуванням.

Розширення також забезпечує відмінну підтримку веб-розробки з Python, включаючи інтеграцію з фреймворками, такими як Django, Flask та Pyramid. Воно пропонує спеціальні функції, такі як налагодження веб-додатків, візуалізація шаблонів та багато іншого.

PyDev також має функції, що підвищують продуктивність, такі як інтерактивна Python-консоль, підтримка віртуальних середовищ, форматування коду відповідно до стандартів Python та багато іншого.

Інтерфейс PyDev наведено на рисунку 3.6.

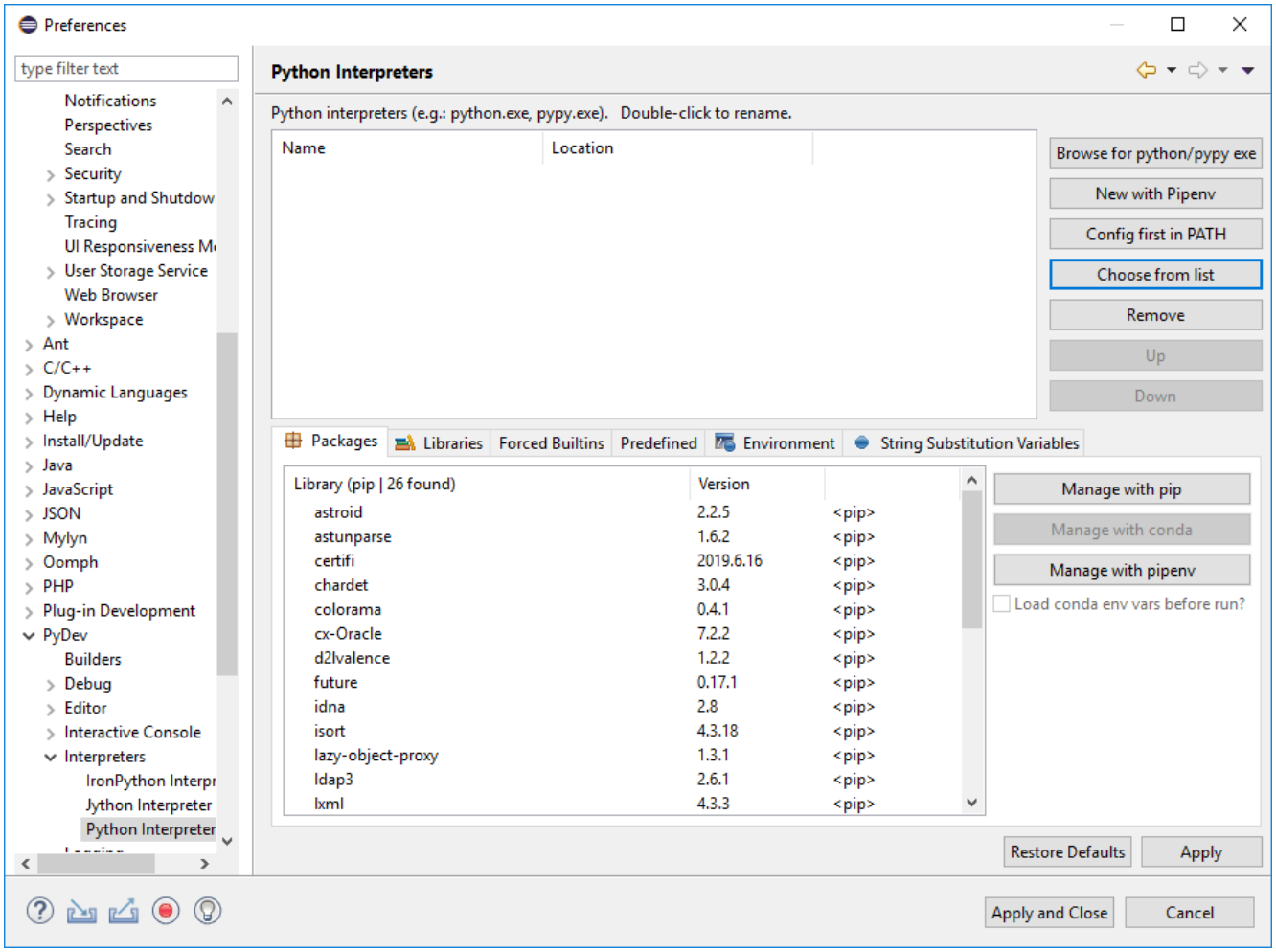


Рисунок 3.6 – Інтерфейс PyDev

Отже, PyDev є потужним розширенням для Eclipse IDE, що забезпечує всебічну підтримку розробки на Python. Його тісна інтеграція з Eclipse, багатий набір функцій та відкритий вихідний код роблять його чудовим вибором для розробників Python, які віддають перевагу Eclipse як основному середовищу розробки.

Складемо таблицю порівняння характеристик цих середовищ розробки (таблиця 3.2).

Таблиця 3.2 – Таблиця порівняння характеристик середовищ розробки

Характеристика	**PyCharm**	**Visual Studio Code**	**PyDev**
Підтримка Python	+	+	+
Автодоповнення коду	+	+	+

Продовження таблиці 3.2

Рефакторинг коду	+	-	+
Підтримка веб-розробки	+	+	+
Екосистема розширень	+	+	-
Вбудована система налагодження	+	+	+
Інтеграція з системами контролю версій	+	+	-
Виконання тестів	+	+	+
Сумарний коефіцієнт	8	7	6

Таким чином, PyCharm є найкращим середовищем для розробки програмного засобу з планування подорожей.

3.3 Розробка інтерфейсу користувача

Структурна схема програмного інтерфейсу є графічним зображенням організації та зв'язків між різними елементами користувацького інтерфейсу програми. Вона допомагає розробникам та дизайнерам уявити та спланувати структуру та спосіб взаємодії користувача з програмою.

Основними компонентами структурної схеми є сторінки, екрани, вікна або модальні вікна, які представляють собою окремі функціональні блоки. Ці блоки пов'язані між собою через переходи, які визначають логіку переміщення користувача. Переходи можуть бути ініційовані користувачем, наприклад, натисканням кнопки, або автоматично, наприклад, при зміні стану програми.

Також у структурну схему можуть входити компоненти, такі як меню, панелі інструментів, форми, повідомлення про помилки і т. д. Ці компоненти можуть бути використані на різних сторінках або екранах, що забезпечує послідовність та зручність користування програмою.

Було розроблено структурні схеми інтерфейсу застосунку для планування подорожей. Структурну схему вікна реєстрації зображено на рисунку 3.7.

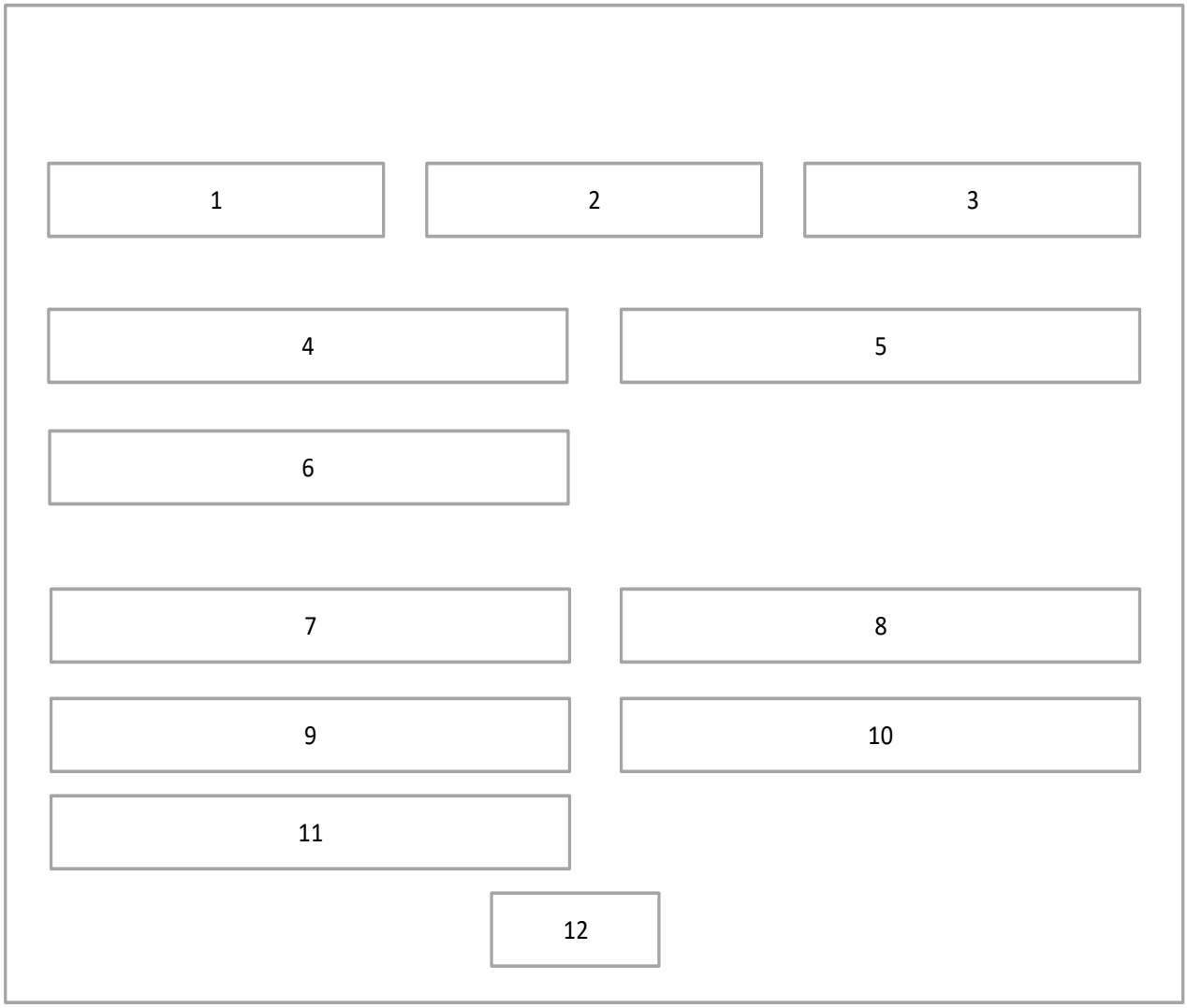


Рисунок 3.7 – Структурна схема вікна реєстрації

Елементи вікна реєстрації:

1. Нікнейм користувача.
2. Пароль.
3. Повторення пароля.
4. Ім’я користувача.
5. Прізвище користувача.
6. Електронна адреса.
7. Вулиця.
8. Місто.
9. Регіон.
10. Країна.

11. Поштовий індекс.

12. Кнопка підтвердження реєстрації.

Для реалізації цього вікна використовується HTML-шаблон. Він використовує бібліотеку Bootstrap для стилізації та розмітки. Сторінка має форму з полями вводу для імені користувача, електронної пошти, пароля та підтвердження пароля. Також є кнопка для відправлення форми та посилання для переходу на сторінку входу.

Сторінка містить секцію для відображення флеш-повідомлень (наприклад, про успішну реєстрацію або помилки валідації). Ці повідомлення автоматично приховуються через 5 секунд завдяки вбудованому JavaScript-коду.

Сторінка має адаптивний дизайн завдяки використанню CSS-фреймворку Bootstrap.

Крім того, код включає посилання на зовнішні ресурси, такі як шрифти, іконки Font Awesome та CSS-файли Bootstrap.

Елементи вікна авторизації:

1. Електронна адреса.
2. Пароль.
3. Кнопка «Увійти».

Структурна схема вікна авторизації наведена на рисунку 3.8.

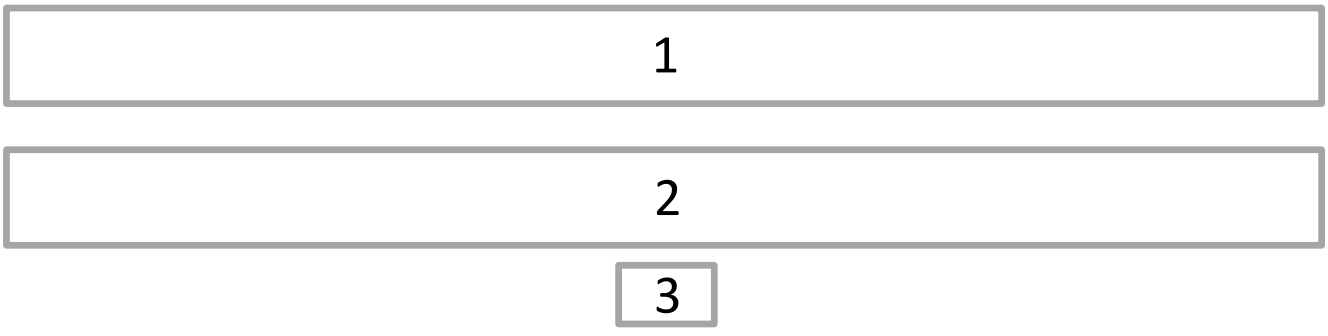


Рисунок 3.8 – Структурна схема вікна авторизації

Ця сторінка використовує HTML-шаблон. Він також використовує бібліотеку Bootstrap для стилізації та розмітки, схожу на шаблон реєстрації.

Сторінка містить форму входу з полями вводу для електронної пошти та пароля. Також є кнопка для відправлення форми та посилання для переходу на сторінку реєстрації.

Як і у випадку зі сторінкою реєстрації, цей шаблон містить секцію для відображення флеш-повідомлень (наприклад, про невірні облікові дані). Ці повідомлення автоматично приховуються через 5 секунд завдяки вбудованому JavaScript-коду.

Сторінка має адаптивний дизайн завдяки використанню CSS-фреймворку Bootstrap. Фоновим зображенням є градієнтний колір та зображення міста.

Крім того, код включає посилання на зовнішні ресурси, такі як шрифти, іконки Font Awesome та CSS-файли Bootstrap.

Основна відмінність між цим шаблоном та шаблоном реєстрації полягає у формі входу, яка містить лише поля для електронної пошти та пароля, а також посилання на сторінку реєстрації замість посилання на сторінку входу.

Розроблено структурну схему вікна планування подорожей. Воно дозволяє складати детальний план подорожі та витрати на нього.

Структурна схема вікна наведена на рисунку 3.9.

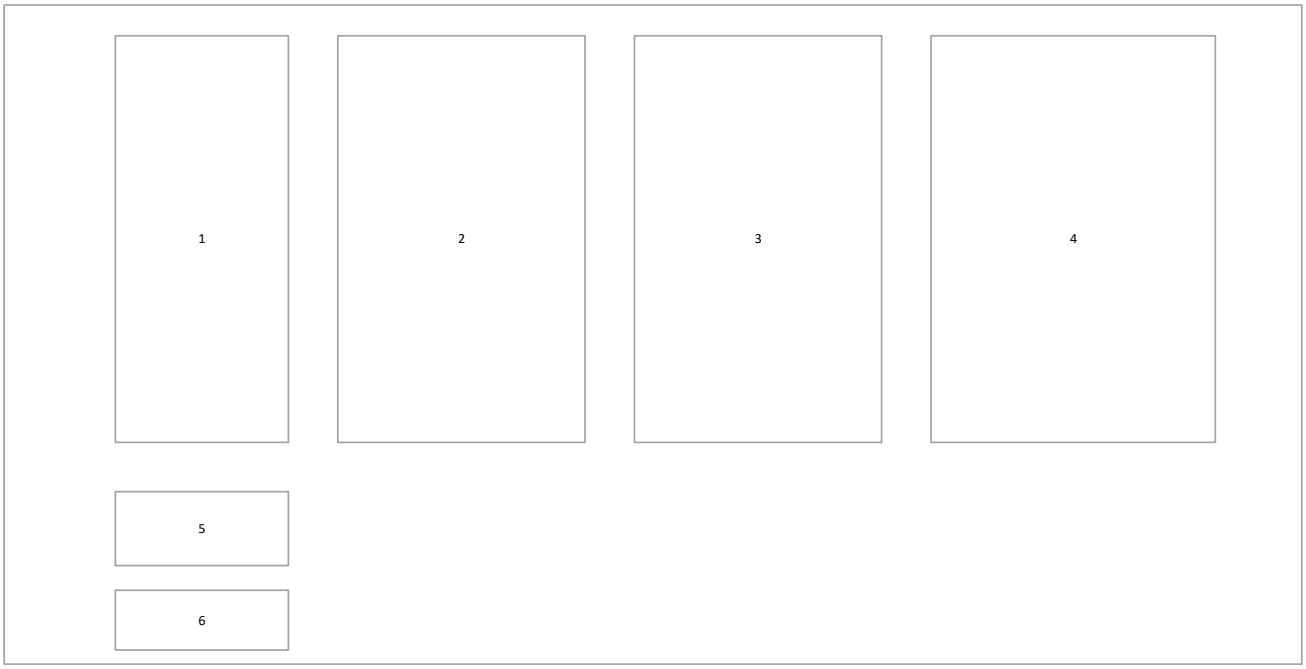


Рисунок 3.9 – Структурна схема вікна планування подорожі

Елементи вікна планування подорожей:

1. Колонка для дати записів.

2. Часовий період.

3. Опис дії.

4. Витрати.

5. Загальна сума витрат.

6. Кнопка «Розпочати подорож».

Цей код є шаблоном для веб-сторінки, що відображає інформацію про погоду та запланований маршрут подорожі:

1. Інформація про погоду. Цей розділ відображає таблицю з даними про погоду для кількох днів. Дані включають дату, поточні погодні умови, максимальну та мінімальну температуру, ймовірність опадів, вологість та попередження про погоду.

2. Запланований маршрут. Цей розділ відображає попередній маршрут подорожі у форматі Markdown. Вміст Markdown конвертується у HTML за допомогою бібліотеки markdown-it та відображається на сторінці.

3. Бронювання готелів та авіаквитків. Цей розділ включає кнопку, яка веде на веб-сайт booking.com для бронювання готелів та авіаквитків.

4. Іконка «поділитися». Цей розділ включає плаваючий контейнер з кнопками для поширення в соціальних мережах. Користувачі можуть поділитися URL-адресою сторінки на різних платформах, таких як Facebook, WhatsApp, Twitter, LinkedIn та Telegram.

5. Футер сторінки. Розділ підвалу включає повідомлення про авторські права та посилання на веб-сайт автора.

6. Завантажити PDF. Код включає скрипт, який дозволяє користувачам завантажити PDF-версію маршруту. Він використовує бібліотеку html2pdf.js для перетворення основного вмісту сторінки у PDF-файл.

7. Рендеринг Markdown. Цей розділ включає скрипт, який використовує бібліотеку markdown-it для конвертації вмісту Markdown в HTML та вставки його на веб-сторінку.

8. Видалення повідомлень сповіщень. У коді є скрипт, який видаляє будь-які повідомлення сповіщень з класом "alert" після визначеного часу (за замовчуванням 5 секунд). Спочатку він встановлює непрозорість повідомлень сповіщень на 0 для ініціалізації переходу, а потім видаляє їх після тривалості переходу (за замовчуванням 1 секунда).

Шаблон розширює базовий шаблон під назвою headers.html і включає контент всередині розділу {% block content %}. Він також включає посилання на різні зовнішні бібліотеки та таблиці стилів, такі як Bootstrap, Font Awesome та Google Fonts.

Загалом, цей шаблон забезпечує зручний інтерфейс для відображення інформації про погоду, запланованого маршруту подорожі та додаткових ресурсів, таких як бронювання готелів та авіаквитків. Він також пропонує функції, такі як поширення в соціальних мережах та завантаження PDF-файлу для маршруту подорожі.

Структурна схема вікна рекомендацій подорожей наведена на рисунку 3.10. Вона відображає популярні напрямки подорожей, які система радить користувачеві.

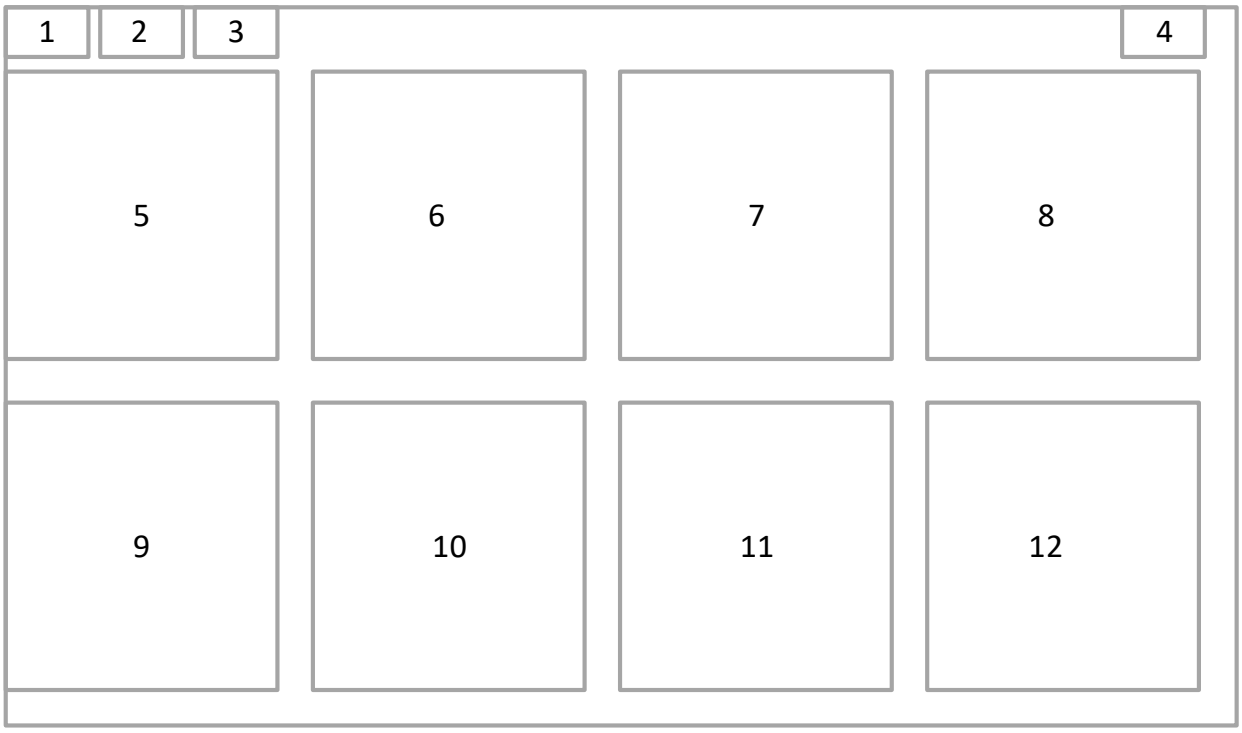


Рисунок 3.10 – Структурна схема вікна рекомендацій подорожей

Елементи вікна рекомендацій подорожей:

1-3. Категорії рекомендацій.

4. Кнопка «Обрати всі категорії».

5-12. Міста, які система рекомендує відвідати.

Для реалізації цього вікна було розроблено компонент під назвою "Featured", який відображає список міст з відповідними зображеннями та кількістю доступних поїздок:

1. Компонент імпортує необхідні модулі, такі як apiClient для здійснення HTTP-запитів, USER_API_ROUTES для отримання відповідного API-маршруту, Image з бібліотеки Next.js для відображення зображень і React-хуки useEffect та useState.

2. Компонент використовує useState для зберігання масиву міст з інформацією про назву, зображення та кількість доступних поїздок.

3. Всередині useEffect знаходиться асинхронна функція getUniqueCities, яка викликає маршрут API USER_API_ROUTES.GET_UNIQUE_TRIP_CITIES за допомогою apiClient. Отримані з API міста зберігаються в стані компонента за допомогою функції setCities.

4. У функції рендеринга компонент виводить заголовок, підзаголовок та сітку карток з інформацією про міста. Картки містять зображення міста (завантажене за допомогою компонента Image з Next.js), назву міста та кількість доступних поїздок.

5. Компонент використовує класи Tailwind CSS для стилізації елементів, таких як заголовки, підзаголовки, сітка карток та самі картки. Картки мають ефект наведення за допомогою класів hover:shadow-lg та transition-all.

6. Картки відображаються за допомогою методу map на масиві міст. Для кожного міста створюється окрема картка з відповідним вмістом.

Розроблено також структурну схему вікна бронювання готелів, яка сформована у вигляді таблиці. Її зображення подано на рисунку 3.11.

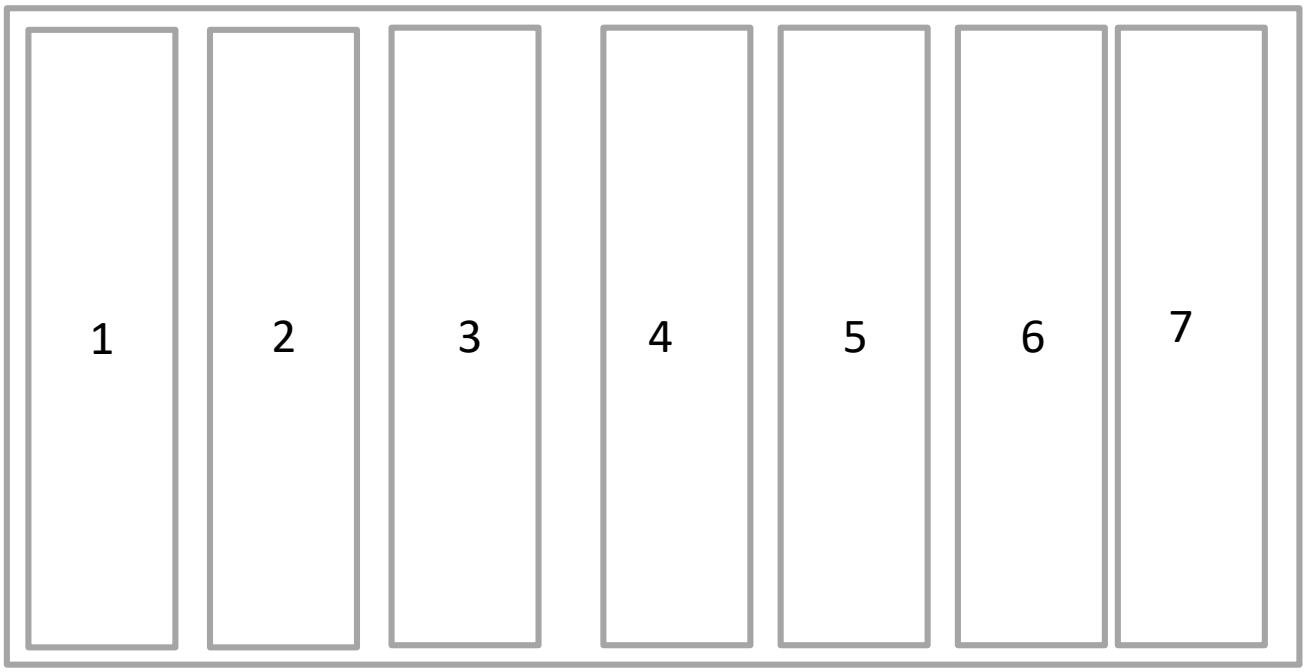


Рисунок 3.11 – Структурна схема вікна бронювання готелів

Елементи вікна:

1. Колонка з назвою готелю.
2. Дата прибуття.
3. Дата відбуття.
4. Тип номера.
5. Кількість проживаючих.
6. Загальна сума витрат.
7. Кнопки «Відмінити бронювання».

Цей код є контролером під назвою ReservationController, який відповідає за операції, пов'язані з бронюванням номерів у готелі. Ось детальний опис різних методів у цьому контролері:

1. index(): Цей метод отримує всі бронювання для автентифікованого користувача, включно з інформацією про пов'язані номер і готель. Він впорядковує бронювання за датою приїзду в зростаючому порядку і передає їх у вигляді dashboard.reservations.

2. create(hotel_id): Цей метод отримує інформацію про готель, включно з його номерами, для заданого $hotel_id. Він передає інформацію про готель у вигляд

dashboard.reservationCreate, який, ймовірно, використовується для створення нового бронювання.

3. store(request): Цей метод обробляє створення нового бронювання. Спочатку він встановлює user_id бронювання рівним ідентифікатору sub автентифікованого користувача з Auth0. Потім він створює новий запис Reservation з даними з запиту.

4. show(reservation): Цей метод отримує вказане бронювання, включно з інформацією про пов'язані номер і готель. Він перевіряє, чи має автентифікований користувач дозвіл на перегляд бронювання, порівнюючи user_id бронювання з ідентифікатором sub автентифікованого користувача. Якщо користувач має дозвіл, він передає бронювання та інформацію про готель у вигляд dashboard.reservationSingle.

5. edit(reservation): Цей метод отримує вказане бронювання, включно з інформацією про пов'язані номер і готель. Він перевіряє, чи має автентифікований користувач дозвіл на редагування бронювання, порівнюючи user_id бронювання з ідентифікатором sub автентифікованого користувача. Якщо користувач має дозвіл, він передає бронювання та інформацію про готель у вигляд dashboard.reservationEdit.

6. update(request, reservation): Цей метод обробляє оновлення існуючого бронювання. Спочатку він перевіряє, чи має автентифікований користувач дозвіл на оновлення бронювання, порівнюючи user_id бронювання з ідентифікатором sub автентифікованого користувача. Якщо користувач має дозвіл, він оновлює властивості бронювання з даними з запиту і зберігає зміни.

7. destroy(reservation): Цей метод обробляє видалення бронювання. Спочатку він отримує вказане бронювання. Потім він перевіряє, чи має автентифікований користувач дозвіл на видалення бронювання, порівнюючи user_id бронювання з ідентифікатором sub автентифікованого користувача. Якщо користувач має дозвіл, він видаляє бронювання.

Контролер використовує моделі Reservation, Hotel та Room для взаємодії з базою даних. Він також використовує систему автентифікації для автентифікації користувачів та отримання їх ідентифікаторів sub з Auth0. Ідентифікатор sub,

ймовірно, є унікальним ідентифікатором для кожного користувача в системі автентифікації Auth0.

Цей контролер забезпечує функціональність для керування бронюваннями номерів у готелі, включно зі створенням, переглядом, оновленням та видаленням бронювань, а також забезпечує, щоб користувачі мали доступ та могли змінювати лише свої власні бронювання.

3.4 Висновки

В цьому розділі, було проведено варіантний аналіз та здійснено вибір середовища розробки застосунку та мови програмування, розроблено структурні схеми та дизайн інтерфейсу. Було обрано мову програмування Python та середовище розробки PyCharm для розробки, аргументовано їхні переваги перед аналогічними засобами.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування

Тестування програмного забезпечення є критично важливим процесом, який гарантує високу якість, безпеку та надійність програмних продуктів. Воно допомагає виявляти та усувати помилки та недоліки на ранніх стадіях розробки, що знижує витрати та прискорює випуск продукту на ринок [17].

Одним із видів тестування є тестування веб-застосунків. Веб-додатки стали невід'ємною частиною нашого повсякденного життя, і їх якість має вирішальне значення для забезпечення позитивного користувацького досвіду.

Складові веб-тестування наведено на рисунку 4.1.

Рисунок 4.1 – Види веб-тестування

Тестування веб-застосунків включає різні типи випробувань, такі як функціональне, навантажувальне, тестування інтерфейсу користувача, безпеки, сумісності та регресійне тестування.

Тестування програм для планування подорожей є особливим видом тестування веб-застосунків. Такі програми мають бути надзвичайно надійними, безпечними та зручними у використанні, оскільки користувачі довіряють їм свої персональні дані та гроші. Будь-які помилки або недоліки можуть мати серйозні наслідки для користувачів та репутації компанії.

Функціональне тестування програм для планування подорожей перевіряє основні функції, такі як пошук авіаквитків, бронювання готелів, оренда автомобілів, створення маршрутів тощо. Воно гарантує, що всі функції працюють коректно і відповідно до вимог замовника та очікувань користувачів [18].

Види функціонального тестування наведено на рисунку 4.2.

Рисунок 4.2 – Види функціонального тестування

Навантажувальне тестування імітує реальні умови використання, коли на веб-застосунок одночасно здійснюється велика кількість запитів. Це допомагає визначити граничні навантаження, при яких система працює стабільно, без збоїв та уповільнень. Для програм для планування подорожей це особливо важливо, оскільки вони можуть зазнавати пікових навантажень, наприклад, під час свят або туристичних сезонів.

Тестування інтерфейсу користувача забезпечує зручність використання програми для планування подорожей. Користувацький інтерфейс має бути інтуїтивно зрозумілим, доступним та привабливим для різних категорій користувачів, включаючи людей з обмеженими можливостями та людей похилого віку .

Критерії тестування інтерфейсу користувача наведені на рисунку 4.3.

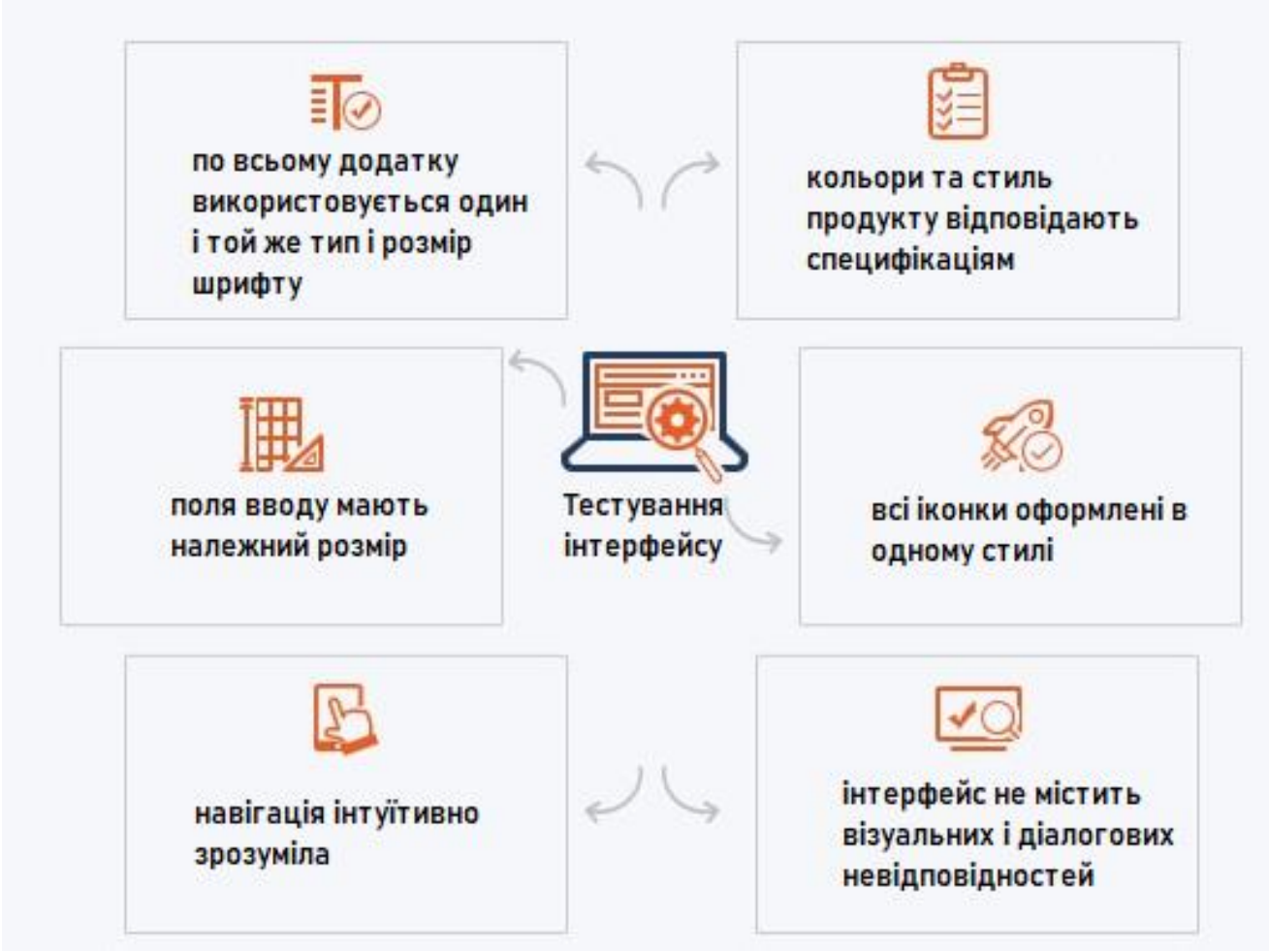


Рисунок 4.3 – Критерії тестування інтерфейсу користувача

Автоматизоване тестування є ефективним способом прискорити процес тестування та зменшити витрати. Воно передбачає використання спеціальних інструментів для створення та виконання тестових сценаріїв, що дозволяє скоротити час, необхідний для ручного тестування, та підвищити ефективність процесу.

Застосування передових методологій та інструментів тестування, таких як Test-Driven Development (TDD), Behavior-Driven Development (BDD) та Continuous Integration/Continuous Deployment (CI/CD), може значно підвищити ефективність процесу тестування веб-застосунків та програм для планування подорожей.

Тестування безпеки веб-застосунків, особливо програм для планування подорожей, вимагає постійної пильності та регулярних перевірок, оскільки ландшафт кібербезпеки постійно змінюється, і з'являються нові загрози. Тестувальники повинні бути в курсі найновіших методів атак та вразливостей і застосовувати відповідні заходи для захисту систем [19].

Етапи тестування безпеки застосунків наведено на рисунку 4.4.

Рисунок 4.4 – Тестування безпеки застосунків

Автоматизація тестування може значно підвищити ефективність та скоротити час, необхідний для виконання регресійних та повторюваних тестів.

Отже, тестування відіграє вирішальну роль у забезпеченні високої якості, безпеки та надійності веб-застосунків, включаючи програми для планування подорожей. Ретельне планування, використання сучасних методологій та інструментів, а також тісна співпраця між усіма зацікавленими сторонами є ключовими факторами успіху в галузі тестування програмного забезпечення.

4.2 Тестування програми

Проведемо тестування функціоналу розробленого застосунку та демонстрацію його можливостей.

На рисунку 4.5 зображено вікно реєстрації.

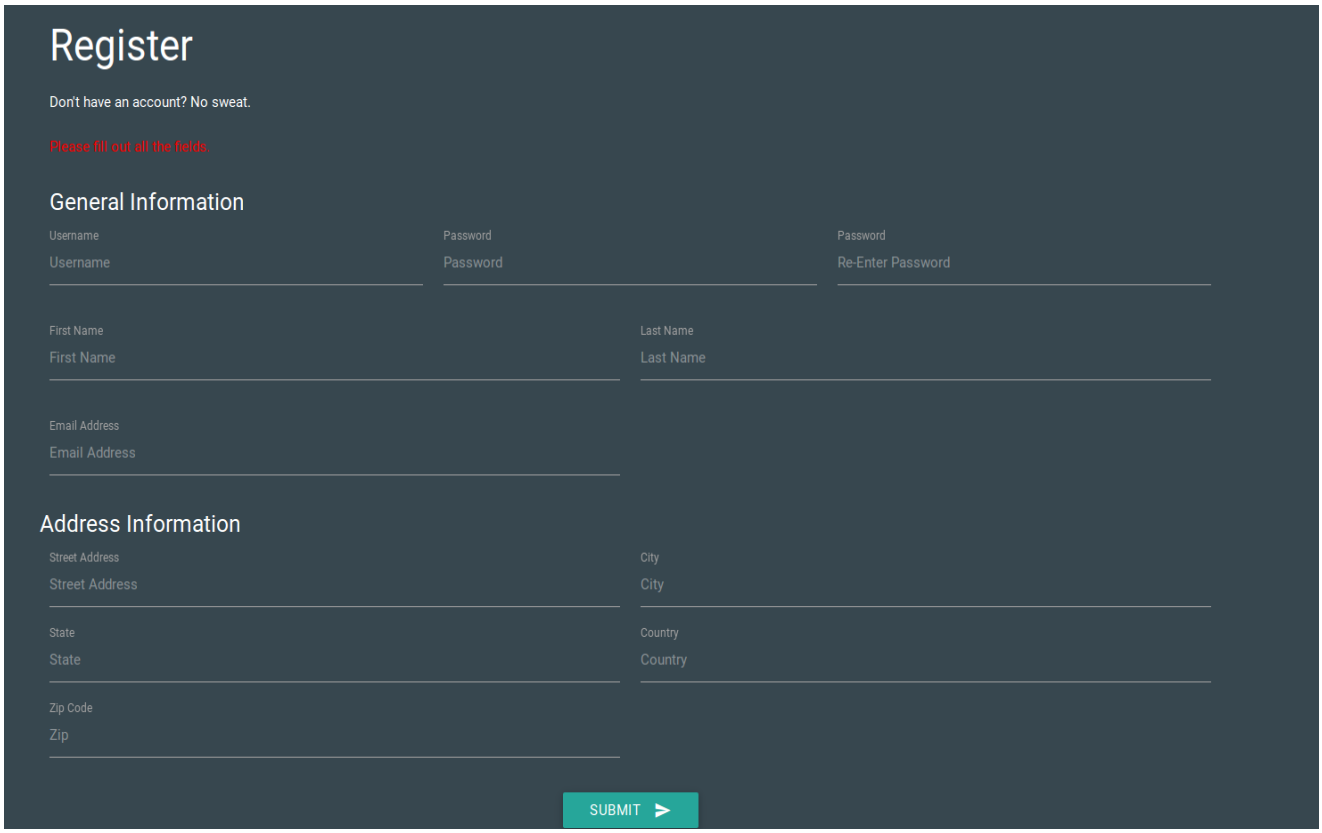


Рисунок 4.5 – Вікно реєстрації

Вікно авторизації зображено на рисунку 4.6.

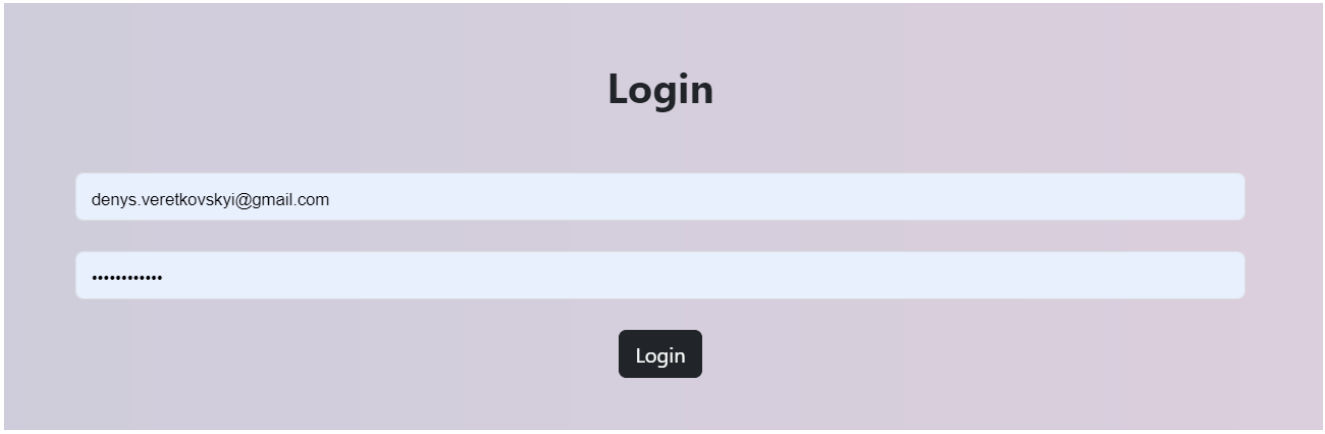


Рисунок 4.6 – Вікно авторизації

На рисунку 4.7 продемонстровано вкладку рекомендацій користувачеві популярних напрямків подорожей та доступних на цих напрямках готелів.

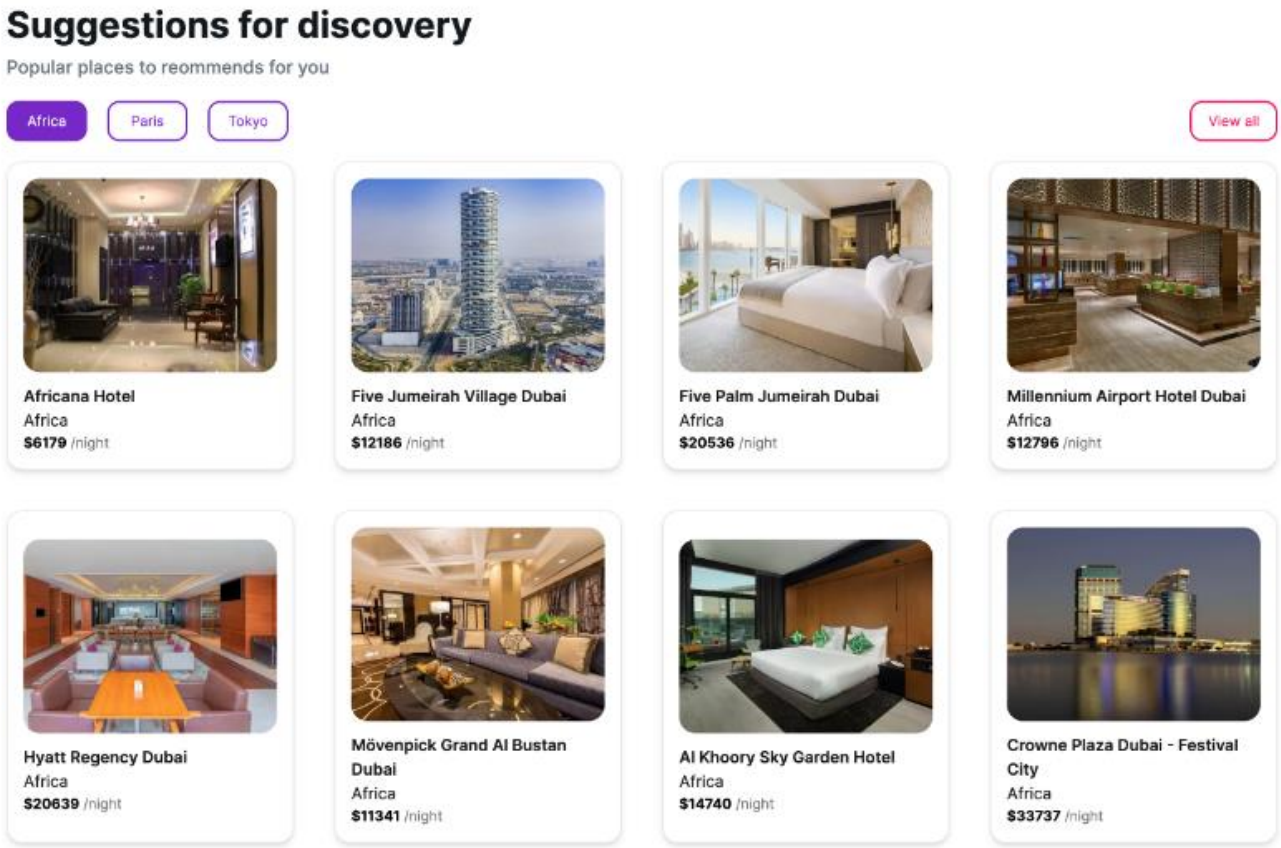
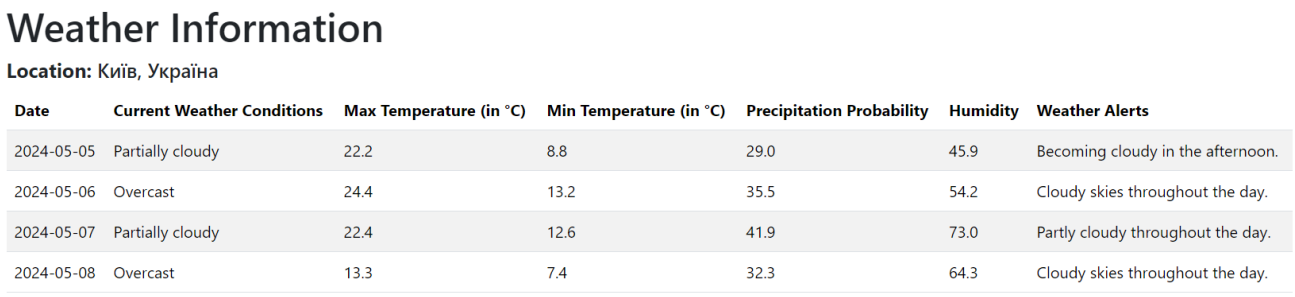


Рисунок 4.7 – Рекомендації готелів

На рисунку 4.8 зображено прогноз погоди у вказаному користувачем місті. На цьому рисунку зображено прогноз погоди для Києва.



Weather Information

Location: Київ, Україна

Date	Current Weather Conditions	Max Temperature (in °C)	Min Temperature (in °C)	Precipitation Probability	Humidity	Weather Alerts
2024-05-05	Partially cloudy	22.2	8.8	29.0	45.9	Becoming cloudy in the afternoon.
2024-05-06	Overcast	24.4	13.2	35.5	54.2	Cloudy skies throughout the day.
2024-05-07	Partially cloudy	22.4	12.6	41.9	73.0	Partly cloudy throughout the day.
2024-05-08	Overcast	13.3	7.4	32.3	64.3	Cloudy skies throughout the day.

Рисунок 4.8 – Прогноз погоди

На рисунку 4.9 зображено меню вибору напрямку маршруту подорожі та періоду подорожі.

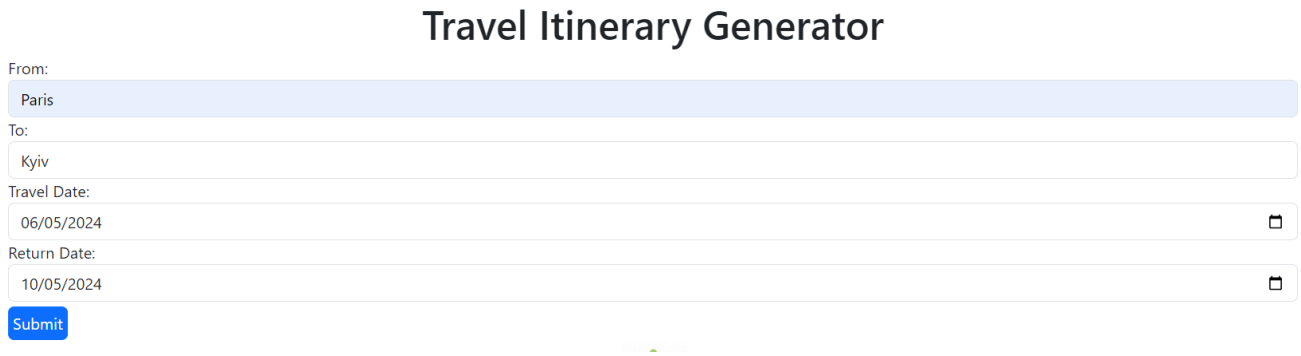


Рисунок 4.9 – Меню вибору напрямку маршруту подорожі

На рисунку 4.10 зображено формування списку витрат на різноманітні активності користувачем. Це дозволяє краще планувати подорож та витрати на неї.

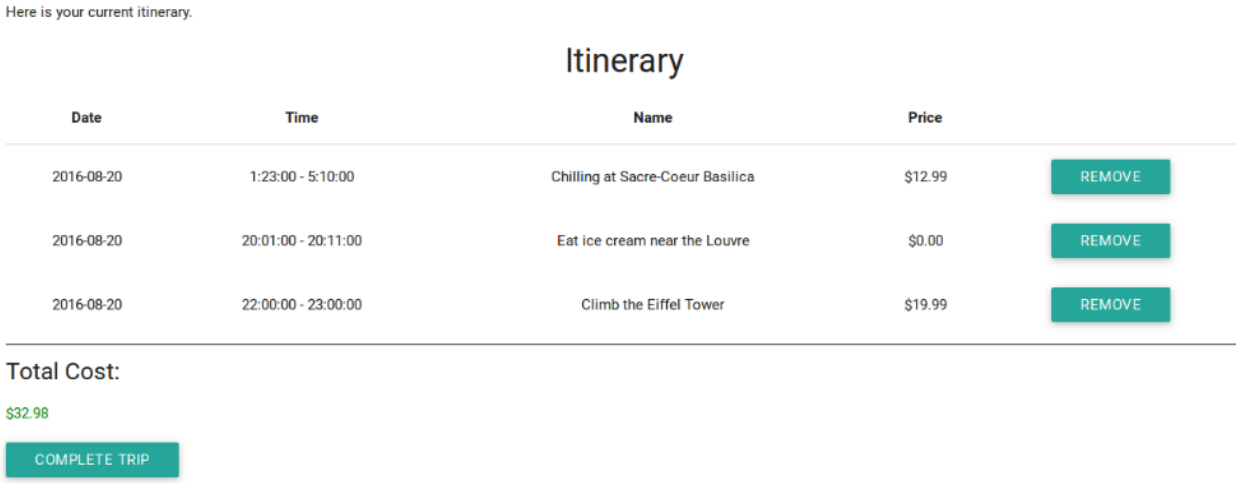


Рисунок 4.10 – Формування списку витрат на подорож

Формування маршруту та рекомендацій щодо подорожі зображено на рисунку 4.11. Тут продемонстровано формування маршруту з Парижа до Києва, в якому описані рекомендації щодо того, які міста можна відвідати на шляху маршруту і які цікаві місця в них знаходяться.

Day 1-7: Paris

- **Accommodation:** Hotel Les Théâtres
- **Transportation:** Arrive in Paris and take a taxi to the hotel
- **Activities:** Visit the Eiffel Tower, Louvre Museum, Arc de Triomphe, Notre Dame Cathedral, and enjoy a cruise along the Seine River.

Day 8-14: Amsterdam

- **Transportation:** Train from Paris to Amsterdam
- **Accommodation:** Hotel Estheréa
- **Activities:** Explore the canals, visit the Anne Frank House, Rijksmuseum, and enjoy the nightlife in Leidseplein.

Day 15-18: Berlin

- **Transportation:** Train from Amsterdam to Berlin
- **Accommodation:** The Ritz-Carlton, Berlin
- **Activities:** Visit the Brandenburg Gate, Reichstag Building, Holocaust Memorial, and explore the vibrant Kreuzberg district.

Day 19-21: Prague

- **Transportation:** Train from Berlin to Prague
- **Accommodation:** The Grand Mark Prague
- **Activities:** Visit Prague Castle, Charles Bridge, Old Town Square, and indulge in traditional Czech cuisine.

Day 22-26: Vienna

Рисунок 4.11 – Формування маршруту подорожі

На рисунку 4.12 продемонстровано бронювання готелів користувачем.

Your Reservations

Hotel	Arrival	Departure	Type	Guests	Price	Manage
Marriott	2019-08-28	2019-08-29	Luxury Suite	7	$980.00	Edit
Marriott	2019-08-28	2019-08-29	Luxury Suite	8	$980.00	Edit
Aria	2020-05-06	2020-05-07	Suite	3	$350.00	Edit
Marriott	2020-05-10	2020-05-12	Double	1	$200.00	Edit
Aria	2020-05-12	2020-05-15	Economy	2	$87.99	Edit
Marriott	2020-05-18	2020-05-29	Luxury Suite	10	$980.00	Edit
Marriott	2020-05-20	2020-05-24	Double	2	$200.00	Edit
Marriott	2021-08-28	2021-08-30	Double	9	$200.00	Edit

Рисунок 4.12 – Бронювання готелів

Таким чином, розроблений застосунок виконує поставлені перед ним на початку розробки задачі.

4.3 Висновки

В цьому розділі було проведено аналіз методів тестування, проведено тестування розробленої програми, розглянуті різні варіанти використання. Було продемонстровано, що розроблений застосунок виконує поставлені на початку розробки задачі.

ВИСНОВКИ

В результаті виконання бакалаврської дипломної роботи, було розроблено програмний засіб з планування подорожей. Він дозволяє користувачам планувати подорожі, бронювати готелі, планувати витрати, переглядати прогноз погоди, отримувати рекомендації для подорожі залежно від вказаного користувачем маршруту.

Для розробки було використано мову програмування Python та середовище розробки PyCharm. Бакалаврська дипломна робота оформлена згідно методичних вказівок [20].

У першому розділі, було проведено аналіз стану питання розробки програмного засобу з планування подорожей, порівняльний аналіз аналогів, аналіз методів розв’язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження.

У другому розділі, було проведено аналіз інформаційного забезпечення, розроблено модель застосунку, метод, загальний алгоритм роботи програми .

У третьому розділі, було проведено варіантний аналіз та вибір мови програмування і середовища розробки розроблено алгоритми роботи застосунку, розроблено структурні схеми інтерфейсу застосунку, розроблено програмний застосунок.

У четвертому розділі, було проведено аналіз методів тестування, проведено тестування розробленого застосунка. Тестування довело його працездатність та коректність роботи. Розроблений застосунок виконує поставлені перед ним на початку розробки задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Top 10 benefits of travelling [Електронний ресурс] – Режим доступу до ресурсу: https://eurama.hu/informations-budapest/top-10-benefits-of-travelling-learn-why-travelling-is-good-for-you/

2. Why it’s important to travel [Електронний ресурс] – Режим доступу до ресурсу: https://learnenglishteens.britishcouncil.org/study-break/youtubers/why-its-important-travel

3. The full guide to travel app development [Електронний ресурс] – Режим доступу до ресурсу: https://www.cleveroad.com/blog/travel-app-development/

4. Essential features of user-centric travel apps [Електронний ресурс] – Режим доступу до ресурсу: https://coaxsoft.com/blog/essential-features-for-user-centric-travel-apps-prioritizing-the-travelers-experience

5. Expedia [Електронний ресурс] – Режим доступу до ресурсу: https://www.expedia.com/

6. TripAdvisor [Електронний ресурс] – Режим доступу до ресурсу: https://www.tripadvisor.com/

7. Kayak [Електронний ресурс] – Режим доступу до ресурсу: https://www.ua.kayak.com/

8. Microservice architecture [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/microservices/

9. Big Data [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/big-data-bolshie-dannye

10. What is automated testing? [Електронний ресурс] – Режим доступу до ресурсу: https://smartbear.com/learn/automated-testing/what-is-automated-testing/

11. What is JavaScript? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/javascript/

12. PHP [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/

13. Python [Електронний ресурс] – Режим доступу до ресурсу:

https://www.python.org/

14. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/

15. PyCharm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/

16. PyDev [Електронний ресурс] – Режим доступу до ресурсу: https://www.pydev.org/

17. The different types of software testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

18. Functional testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.opentext.com/what-is/functional-testing

19. Prem Apte. Software Testing: Job Ready In 80 Hours. Пуне: Prem Apte, 2023. 202с.

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

64

Додаток А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«_12_»__березня__2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного засобу з планування подорожей»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. О.М. Хошаба
«_12_»__березня__2024 року.

Виконав:
студент гр. ЗПІ-20б Веретковський Д.В.
«_12_»__березня__2024 року.

Вінниця – 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного засобу з планування подорожей».

Галузь застосування – веб-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 р.

3. Мета та призначення розробки.

Метою роботи є покращення процесу планування подорожі шляхом розробки спеціалізованого програмного застосунку.

4. Вихідні дані для проведення НДР

1. Top 10 benefits of travelling [Електронний ресурс] – Режим доступу до ресурсу: https://eurama.hu/informations-budapest/top-10-benefits-of-travelling-learn-why-travelling-is-good-for-you/
2. The full guide to travel app development [Електронний ресурс] – Режим доступу до ресурсу: https://www.cleveroad.com/blog/travel-app-development/
3. What is automated testing? [Електронний ресурс] – Режим доступу до ресурсу: https://smartbear.com/learn/automated-testing/what-is-automated-testing/
4. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/

5. Технічні вимоги

Вхідні дані — дані користувача, інформація про заплановані подорожі.

Вихідні дані — маршути подорожі, рекомендації щодо подорожей, прогноз погоди

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним

вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

–пояснювальна записка до бакалаврської дипломної роботи;

–технічне завдання;

–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку програм для планування подорожей	14.03.24 – 24.03.24
2	Порівняльний аналіз аналогів	24.03.24 – 06.04.24
3	Розробка структури застосунку	06.04.24 – 11.04.24
4	Розробка інтерфейсу користувача	12.04.24 – 29.04.24
5	Тестування застосунку	29.04.24 – 11.05.24
6	Оформлення матеріалів до захисту БДР	12.05.24– 30.05.24

10. Порядок контролю та прийняття.

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

67

Додаток Б

Протокол перевірки на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

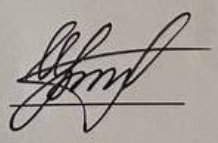
Назва роботи: Розробка програмного засобу з планування подорожей.
Тип роботи: БДР
Підрозділ : кафедра програмного забезпечення, ФІТКІ
Науковий керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.

Оригінальність	92,5 %
Схожість	7,5 %

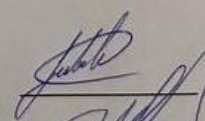
Аналіз звіту подібності

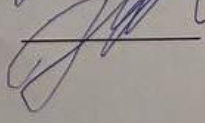
Запозичення в роботі оформлені належним чином і не містять ознак плагіату.

Запозичення, виявлені у роботі, не мають ознак плагіату, але їх велика кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу необхідно доопрацювати.

Особа, відповідальна за перевірку  Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи  Веретковський Д.В.

Керівник роботи  Хошаба О.М.

Додаток В

Лістинг програми

```
from flask import Flask, flash, render_template, request, redirect, session, url_for, flask_wtf
from flask_wtf import Form
from wtforms import StringField
from wtforms.widgets import TextArea
from wtforms.validators import DataRequired
import hashlib
import locale
import pymysql
import time

app = Flask(__name__)
app.config['SECRET_KEY'] = '8ffe05624dfe0efdf7c7f67288d4f4ce5005e0dfb6a1bc48366ef9906dd0586e'
locale.setlocale( locale.LC_ALL, 'en_CA.UTF-8') # To get money formatting

##########################################################################
#                    SQL Queries                         #
##########################################################################
def view_completed_attractions_query():
 return "select activity_date, attraction.attraction_name, description from activity natural join attraction where activity.username = '" + session['username'] + "' and ((activity_date = CURDATE() and activity_end_time <= CURTIME()) or activity_date < CURDATE());"

def get_trip_cost():
 return "select sum(cost) from activity join trip using (trip_id) where trip_id = " + str(session['current_trip_id']) + ";"

def get_all_activities_in_a_trip():
 return "select activity_date, activity_name, cost, activity_start_time, activity_end_time, activity_id from activity natural join trip where username = '" + session['username'] + "' and is_booked = false;";

def get_current_trip_id():
 return "select trip_id from trip natural join user where trip.is_booked=false and user.username='" + session['username'] + "';"
```

```
def add_attraction_to_trip(attraction_name, activity_name, start_time, end_time, date, cost):
 return "insert into activity (activity_name, activity_start_time, activity_end_time, activity_date, attraction_name, username, trip_id, cost) values ('" + activity_name + "', '" + start_time + "', '" + end_time + "', '" + date + "', '" + attraction_name + "', '" + session['username'] + "', " + str(session['current_trip_id']) + ", " + str(cost) + ");"

#########################################################################
#                    WTF FORMS                          #
#########################################################################
class ReviewForm(Form):
    title = StringField('review_title', validators=[DataRequired()])
    body = StringField('review_body', widget=TextArea(), validators=[DataRequired()])

#########################################################################
#                    INDEX/HOME                         #
#########################################################################

# Visit site for first time. Pictures.
@app.route('/')
@app.route('/index')
def index():
 return render_template('index.html')

# Home page. Displays admin interface if user is admin.
@app.route('/home')
def home():

 # Disallow unlogged in users from requesting homepage.
 if 'username' not in session or session['username'] is '':
  return redirect(url_for('index'))

 return create_trip(no_error=True)

#########################################################################
#                    ADMIN PANEL                        #
#########################################################################
```

```
# Delete user from admin panel
@app.route('/delete-user/<username>')
def delete_user(username):

 cursor = db.cursor()
 cursor.execute("delete from user where username='" + username + "';")
 db.commit()

 return redirect(url_for('home'))

# Suspend user from admin panel
@app.route('/suspend-user/<username>')
def suspend_user(username):

 cursor = db.cursor()
 cursor.execute("select suspended from user where username='" + username + "';")
 if cursor.fetchall()[0][0] == 1:
  cursor.execute("update user set suspended=0 where username='" + username + "';")
 else:
  cursor.execute("update user set suspended=1 where username='" + username + "';")
 db.commit()

 return redirect(url_for('home'))

# Make user an Admin from admin panel
@app.route('/make-admin/<username>')
def make_admin(username):

 cursor = db.cursor()
 cursor.execute("update user set is_admin=1 where username='" + username + "';")
 db.commit()

 return redirect(url_for('home'))

#########################################################################
#                    LOGIN / REGISTRATION                    #
#########################################################################
```

```
# Login/Registration Page. Redirects to home if already logged in.
@app.route('/login-page')
def login_page():

 # Show login page if not logged in. Redirect to home if already logged in.
 if 'username' not in session or session['username'] == '':
  return render_template('login.html')
 else:
  return redirect(url_for('home'))

# On Login Form Submit. Loads home page or shows error.
@app.route('/login', methods=['POST'])
def verify_credentials():

 # Parse user input fields
 name=request.form['login_username']
 password=hashlib.sha256(request.form['login_password'].encode('utf-8')).hexdigest()

 # Query Database
 cursor = db.cursor()
 cursor.execute("select * from user where username = '" + name + "' and password = '" + password + "';")
 rows = cursor.fetchall()
 error = None

 if rows:
  # User found
  if rows[0][7] != 1:
   # Not suspended
   session['username'] = rows[0][0]
   session['email'] = rows[0][2]
   session['is_admin'] = rows[0][3]
   session['name'] = rows[0][4]

   # Set current trip id for this user.
   query = get_current_trip_id()
   cursor.execute(query)
   trip_ids = cursor.fetchall()
```

```
  if len(trip_ids) > 0:
   # There are trips for this user. If no, just make it when they start adding attractions.
   session['current_trip_id'] = trip_ids[0][0]

  return redirect(url_for('home'))
 else:
  # Suspended user
  error='User suspended.'
else:
 # No such user. Login again.
 error = 'Incorrect username or password. Please try again.'
return render_template('login.html', error=error)

# Logs out of system and redirects to pictures.
@app.route('/logout')
def logout():

 # Clear out session variables
 session.clear()
 return redirect(url_for('index'))

# On Register Form Submit. Loads home page.
# TODO: Re-fill out correct fields when registration fails.
@app.route('/register', methods=['POST'])
def register():

 # Parse user input fields
 name=request.form['register_username']
 password1=hashlib.sha256(request.form['register_password'].encode('utf-8')).hexdigest()
 password2=hashlib.sha256(request.form['register_password2'].encode('utf-8')).hexdigest()
 firstname=request.form['register_firstname']
 lastname=request.form['register_lastname']
 email=request.form['register_email']
 street=request.form['register_streetaddress']
 city=request.form['register_city']
 state=request.form['register_state']
 country=request.form['register_country']
 zipcode=request.form['register_zip']
```

```
        # Check if all user fields filled in
        if name == '' or password1 == '' or password2 == '' or firstname == '' or firstname == '' or lastname == ''
or email == '' or street == '' or city == '' or state == '' or country == '' or zipcode == '':
         error = 'Please fill out all the fields.'
         return render_template('login.html', error2=error, scroll="register")

        # Check that passwords match
        if password1 != password2:
         error = 'Passwords do not match.'
         return render_template('login.html', error2=error, scroll="register")

        # Parse street_no from street
        street_no = -1

        # Check that address is valid format
        if len(street.split(" ")) < 3 or not street.split(" ")[0].isdigit():
         error = 'Street Address format not recognized. Please re-enter.'
         return render_template('login.html', error2=error, scroll="register")

        street_no = str(street.split(" ")[0])
        street = street[street.index(" ") + 1:]

        # Write to Database
        cursor = db.cursor()
        cursor.execute("insert into address (street_no, street_name, city, state, country, zip) values ("
          + street_no + ", '" + street + "', '" + city + "', '" + state + "', '" + country + "', '" + zipcode + "');")

        cursor.execute("insert into user (username, password, email, is_admin, first_name, last_name,
address_id, suspended) values ('"
        + name + "', '" + password1 + "', '" + email + "', false, '" + firstname + "', '" + lastname + "', (select
max(address_id) from address), 0);")
        db.commit()

        # Update current user session
        session['username'] = name
        session['name'] = firstname
        session['is_admin'] = 0
```

```
    session['email'] = email

    return redirect(url_for('home'))

    ##########################################################################
    #                        REVIEWS                         #
    ##########################################################################

    # View Reviews
    @app.route('/view-reviews/<attraction_index>')
    def view_review(attraction_index):

     # Get attraction_name from the index.
     cursor = db.cursor()
     cursor.execute("select * from attraction;")
     attractions = [dict(name=row[0]) for row in cursor.fetchall()]
     attraction_name = attractions[int(attraction_index) - 1]['name']

     # Get the attraction details using attraction_name.
     query = "select authored_date, body, username, title from review where attraction_name='" +
attraction_name + "';"
     cursor.execute(query)
     reviews = [dict(authored_date=row[0], body=row[1], username=row[2], title=row[3]) for row in
cursor.fetchall()]

     return render_template('attraction_reviews.html', session=session, reviews=reviews,
attraction_name=attraction_name)

    # Select visisted attraction to review.
    @app.route('/review')
    def reviews():

     # Lists a user's visited attractions with a "Review" button.
     cursor = db.cursor()
     query = view_completed_attractions_query()
     cursor.execute(query)
     attractions = [dict(date=row[0], name=row[1], description=row[2]) for row in cursor.fetchall()]
     return render_template('review.html', items=attractions, session=session)
```

```
# Write a review for attraction by its name.
@app.route('/write-review/<attraction_index>')
def write_review(attraction_index):

 # Get attraction_name from the index.
 cursor = db.cursor()
 query = view_completed_attractions_query()
 cursor.execute(query)
 attractions = [dict(date=row[0], name=row[1], description=row[2]) for row in cursor.fetchall()]
 attraction_name = attractions[int(attraction_index) - 1]['name']

 # Check if you should be writing reviews for this attraction (visited in past).
 valid_review = False

 for attraction in attractions:
  if attraction['name'] == attraction_name:
   valid_review = True
   break

 if valid_review:
  form = ReviewForm()
  return render_template('review.html', items=attractions, attraction_name=attraction_name, review=1, form=form, session=session)
 else:
  error='You must complete a visit to the attraction before you can review it!'
  return render_template('review.html', items=attractions, error=error, session=session)

# Submit user created review into database.
@app.route('/create-review', methods=['POST'])
def create_review():

 # Insert the review into the database.
 cursor = db.cursor()
 query = "insert into review (title, authored_date, body, username, attraction_name) values ('" + request.form['title'] + "', '" + time.strftime('%Y-%m-%d') + "', '" + request.form['body'] + "', '" + session['username'] + "', '" + request.form['attraction_name'] + "');"
 cursor.execute(query)
```

```
        db.commit()

        # Reload the review page, with list of attractions that have been visited by this user.
        query = view_completed_attractions_query()
        cursor.execute(query)
        attractions = [dict(date=row[0], name=row[1], description=row[2]) for row in cursor.fetchall()]
        attraction_name = request.form['attraction_name']

        message = "Created review for " + attraction_name + "!"
        return render_template('review.html', items=attractions, success=message, session=session)

        ##########################################################################
        #                       ATTRACTIONS                                      #
        ##########################################################################

        def get_attractions_data():

        cursor = db.cursor()
        cursor.execute("select * from attraction natural join address;")
        attractions = [dict(name=row[1], description=row[2], nearest_transport=row[3],
         address=(str(row[4]) if row[4] is not None else "") + " " + (row[5] if row[5] is not None else "") + " " +
(row[6] if row[6] is not None else "") + ", " + (row[7] if row[7] is not None else "") + " " + (row[8] if row[8] is
not None else "") + " " + (row[9] if row[9] is not None else "")) for row in cursor.fetchall()]

        for i in range(0, len(attractions)):

         attraction = attractions[i]
         attraction_name = attraction['name']

         # Add hours into attractions list
         cursor.execute("select day, hour_start_time, hour_end_time from hour natural join attraction where
attraction.attraction_name='" + attraction_name + "';")
         hours = [dict(day=row[0], hour_start_time=row[1], hour_end_time=row[2]) for row in cursor.fetchall()]
         attractions[i]['hours'] = hours

         # Add time slots into attractions list

         # 1) Get remaining spots for a time slot.
```

```
            cursor.execute("select timeslot_num_people - sum(reserves_num_people) from timeslot natural join reserves where timeslot.attraction_name = '" + attraction_name + "' group by timeslot_id;")
            num_remaining = cursor.fetchall()

            if len(num_remaining) > 0:

             # 2) Get timeslot information.
             cursor.execute("select timeslot_id, timeslot_start_time, timeslot_end_time, timeslot_num_people from timeslot natural join attraction where attraction.attraction_name='" + attraction_name + "';")
             timeslots = []
             rows = cursor.fetchall()
             for j in range(0, len(num_remaining)):
              row = rows[j]
              timeslot = dict(id=row[0], start_time=row[1], end_time=row[2], num_remaining=num_remaining[j][0])
              timeslots.append(timeslot)

             # 3) Add timeslot information to attractions
             attractions[i]['timeslots'] = timeslots
            return attractions

           # Shows all available attractions.
           @app.route('/attractions')
           def attractions():

            attractions = get_attractions_data()
            return render_template('attractions.html', items=attractions, session=session)

           # Receive attraction data to turn into an activity
           @app.route('/add-to-trip/<attraction_index>', methods=['GET'])
           def add_to_trip(attraction_index):

            # TODO: Check if the attraction_name is on list of attractions

            if 'current_trip_id' not in session or not session['current_trip_id']:
             return create_trip(no_error=False)

            cursor = db.cursor()
```

```
        # Check if it was a reserved attraction.
        if 'is_reserved' in request.form and request.form['is_reserved']:

         # Reserved
         timeslot_id = request.form['timeslot_id']
         num_people = request.form['num_people']

         # Update database
         cursor.execute("insert into reserves (reserves_num_people, timeslot_id, username) values (" + str(num_people) + ", " + str(timeslot_id) + ", '" + session['username'] + "');")

        # Get attraction name from index.
        cursor.execute("select * from attraction;")
        attractions = [dict(attraction_name=row[0], description=row[1], nearest_transport=row[2]) for row in cursor.fetchall()]
        attraction_name = attractions[int(attraction_index) - 1]['attraction_name']
        db.commit()

        return render_template('create_activity.html', session=session, attraction_name=attraction_name)

       # Insert activity into database
       @app.route('/create-activity', methods=['POST', 'GET'])
       def create_activity():

        if 'current_trip_id' not in session or not session['current_trip_id']:
         return create_trip(no_error=False)

        # Get activity field data.
        attraction_name = request.form['attraction_name']
        activity_name = request.form['activity_name']
        start_time = request.form['start_time']
        end_time = request.form['end_time']
        date = request.form['date']
        cost = request.form['cost'][1:]

        # Add attraction to trip
        cursor = db.cursor()
```

```
query = add_attraction_to_trip(attraction_name, activity_name, start_time, end_time, date, cost)
cursor.execute(query)
db.commit()

success = attraction_name + " added to My Trip!"
attractions = get_attractions_data()
return render_template('attractions.html', items=attractions, session=session, success=success)

# Delete an attraction
@app.route('/delete-attraction/<attraction_index>')
def delete_attraction(attraction_index):

 # Get attraction_name
 cursor = db.cursor()
 cursor.execute("select attraction.attraction_name from attraction natural join address;")
 attraction_name = cursor.fetchall()[int(attraction_index) - 1][0]

 # Delete from database
 cursor.execute("delete from attraction where attraction_name='" + attraction_name + "';")
 db.commit()
 return redirect(url_for('home'))

#########################################################################
#                          TRIP                         #
#########################################################################

# Shows current trip itinerary.
@app.route('/trip')
def trip():

 # Create a trip if none exists
 if 'current_trip_id' not in session or not session['current_trip_id']:
  return create_trip(no_error=False)

 # Get activity info for this trip
 cursor = db.cursor()
 query = get_all_activities_in_a_trip()
 cursor.execute(query)
```

```
        activities = [dict(date=row[0], name=row[1], price=locale.currency(row[2], grouping=True), start_time=row[3], end_time=row[4], id=row[5]) for row in cursor.fetchall()] # TODO: Correctly map activity info.

        # Calculate total cost of trip
        query = get_trip_cost()
        cursor.execute(query)
        amount = cursor.fetchall()[0][0]
        total_cost = locale.currency(amount, grouping=True) if amount is not None else locale.currency(0, grouping=True)

        return render_template('trip.html', items=activities, session=session, total_cost=total_cost)

    @app.route('/complete')
    def complete():

        # Create a trip if none exists
        if 'current_trip_id' not in session or not session['current_trip_id']:
         return create_trip(no_error=False)

        # Pay if total cost > 0, else update trip_id record to "booked"
        cursor = db.cursor()
        query = get_trip_cost()
        cursor.execute(query)
        total_cost = cursor.fetchall()[0][0]

        if total_cost is None:
         total_cost = 0

        if total_cost > 0:

         # Check if a credit card is on file for this user.
         query = "select creditcard_id from creditcard, user where user.username='" + session['username'] + "' and user.username=creditcard.username;"
         cursor.execute(query)
         num_cards = len(cursor.fetchall())

         if num_cards is 0:
```

```
            # No credit card on file
            return render_template('payment.html', session=session, total_cost=locale.currency(total_cost, grouping=True))
        else:
            # Already have a credit card; they're fine.
            return redirect(url_for('trip_booked'))
    else:
        return redirect(url_for('trip_booked'))

@app.route('/pay', methods=['POST'])
def pay():

    # Create a trip if none exists
    if 'current_trip_id' not in session or not session['current_trip_id']:
        return create_trip(no_error=False)

    card_number="".join(request.form['card_number'].split('-'))
    first_name=request.form['first_name']
    last_name=request.form['last_name']
    exp_month=request.form['expiration_month']
    exp_year=request.form['expiration_year']

    # Get Address ID
    cursor = db.cursor()
    cursor.execute("select address_id from user where username='" + session['username'] + "';")
    address_id = cursor.fetchall()[0][0]

    # Insert credit card information
    query = "insert into creditcard (card_number, username, first_name, last_name, exp_month, exp_year, address_id) values ('" + card_number + "', '" + session['username'] + "', '" + first_name + "', '" + last_name + "', " + str(exp_month) + ", " + str(exp_year) + ", " + str(address_id) + ");"

    return redirect(url_for('trip_booked'))

# Render home page once a trip has been successfully booked.
# TODO: Return a Trip ID so that a user can view their previous trips
@app.route('/trip-booked')
def trip_booked():
```

```
# Create a trip if none exists
if 'current_trip_id' not in session or not session['current_trip_id']:
 return create_trip(no_error=False)

cursor = db.cursor()
cursor.execute("update trip set is_booked=1 where trip_id=" + str(session['current_trip_id']) + ";")
db.commit()

session['current_trip_id'] = False
trip_booked_message = "Congratulations! You're all set!"

# Query database when user is admin for admin panel
if session['is_admin']:

 # Get user table information.
 cursor = db.cursor()
 cursor.execute("select * from user;")
 users = [dict(is_admin="Yes" if row[3] == 1 else "No", username=row[0], password=row[1], first_name=row[4], last_name=row[5], email=row[2], suspended="Yes" if row[7] == 1 else "No") for row in cursor.fetchall()]

 # Get attraction table information.
 cursor.execute("select * from attraction natural join address;")
 attractions = [dict(name=row[1], description=row[2], nearest_transport=row[3],
  address=(str(row[4]) if row[4] is not None else "") + " " + (row[5] if row[5] is not None else "") + " " + (row[6] if row[6] is not None else "") + ", " + (row[7] if row[7] is not None else "") + " " + (row[8] if row[8] is not None else "") + " " + (row[9] if row[9] is not None else "")) for row in cursor.fetchall()]

 return render_template("home.html", session=session, users=users, attractions=attractions, trip_booked_message=trip_booked_message)

return render_template('home.html', session=session, trip_booked_message=trip_booked_message)

def create_trip(no_error):

# Query database when user is admin for admin panel
if session['is_admin']:
```

```
        # Get user table information.
        cursor = db.cursor()
        cursor.execute("select * from user;")
        users = [dict(is_admin="Yes" if row[3] == 1 else "No", username=row[0], password=row[1], first_name=row[4], last_name=row[5], email=row[2], suspended="Yes" if row[7] == 1 else "No") for row in cursor.fetchall()]

        # Get attraction table information.
        attractions = get_attractions_data()

        if no_error:
         return render_template("home.html", session=session, users=users, attractions=attractions, no_trip="Here, you can start making your first trip!")
        else:
         return render_template("home.html", session=session, users=users, attractions=attractions, no_trip_error="You must first create a new trip!")

       # Not an admin
       if no_error:
        if 'current_trip_id' not in session or not session['current_trip_id']:
         return render_template("home.html", session=session, no_trip="Here, you can start making a new trip!")
        return render_template("home.html", session=session)
       else:
        return render_template("home.html", session=session, no_trip_error="You must first create a new trip!")

      # Create a new current trip id for the user
      @app.route('/new-trip', methods=['POST'])
      def new_trip():

        start_date = request.form['start_date']
        end_date = request.form['end_date']

        cursor = db.cursor()
        query = "insert into trip (is_booked, trip_start_date, trip_end_date, creditcard_id, username) values (0, '" + start_date + "', '" + end_date + "', 1, '" + session['username'] + "');"
        cursor.execute(query)
```

```
db.commit()

# Set current trip id for this user.
query = get_current_trip_id()
cursor.execute(query)
session['current_trip_id'] = cursor.fetchall()[0][0]
return redirect(url_for('trip'))

# Remove an activity from a trip
@app.route('/remove-from-trip/<activity_id>')
def remove_from_trip(activity_id):

 # Find out which activity it is from index.
 cursor = db.cursor()
 cursor.execute("delete from activity where activity_id=" + activity_id + ";")
 db.commit()

 return redirect(url_for('trip'))

##############################################################################
#                    MAIN APPLICATION                        #
##############################################################################

# Run the application
if __name__ == '__main__':

 # Note: If your database uses a different password, enter it here.
 db_pass = 'root'

 # Make sure your database is started before running run.py
 db_name = 'team1'
 db = pymysql.connect(host='localhost', user='root', passwd=db_pass, db=db_name)
 app.run(debug=True)
 db.close()
```

Додаток Г

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка програмного засобу з планування подорожей

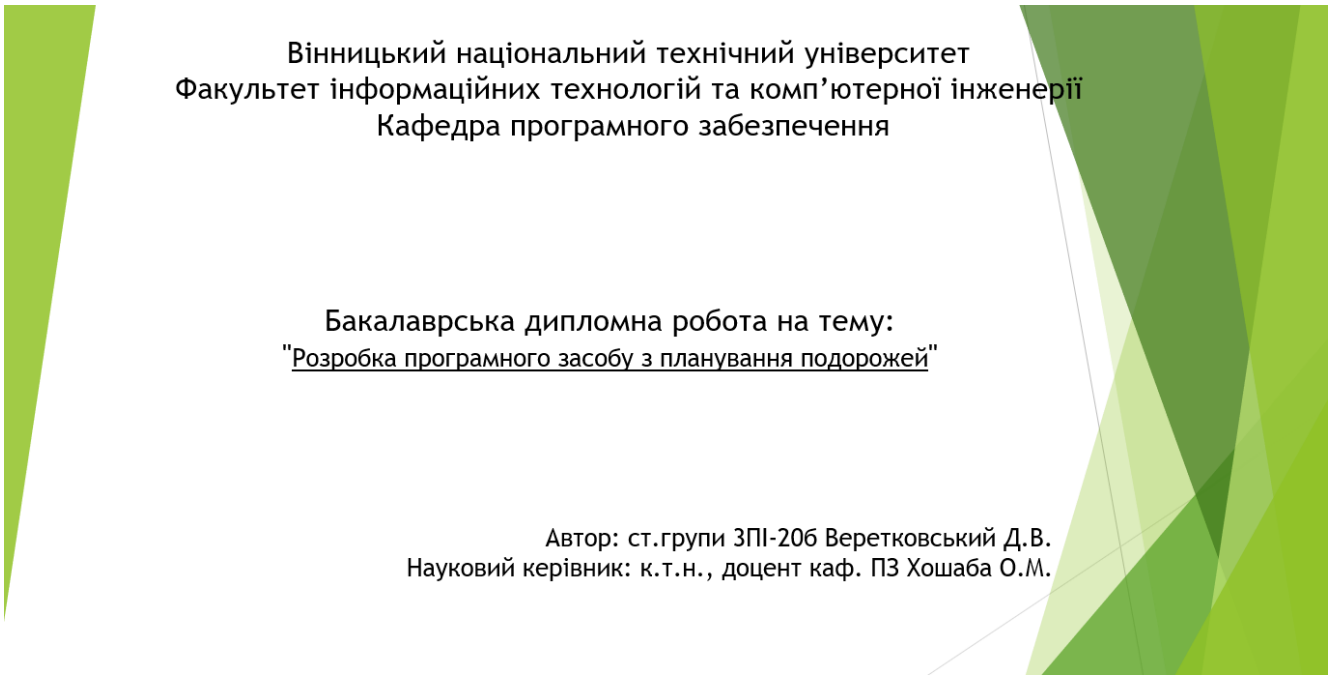


Рисунок Г.1 – Титульний слайд

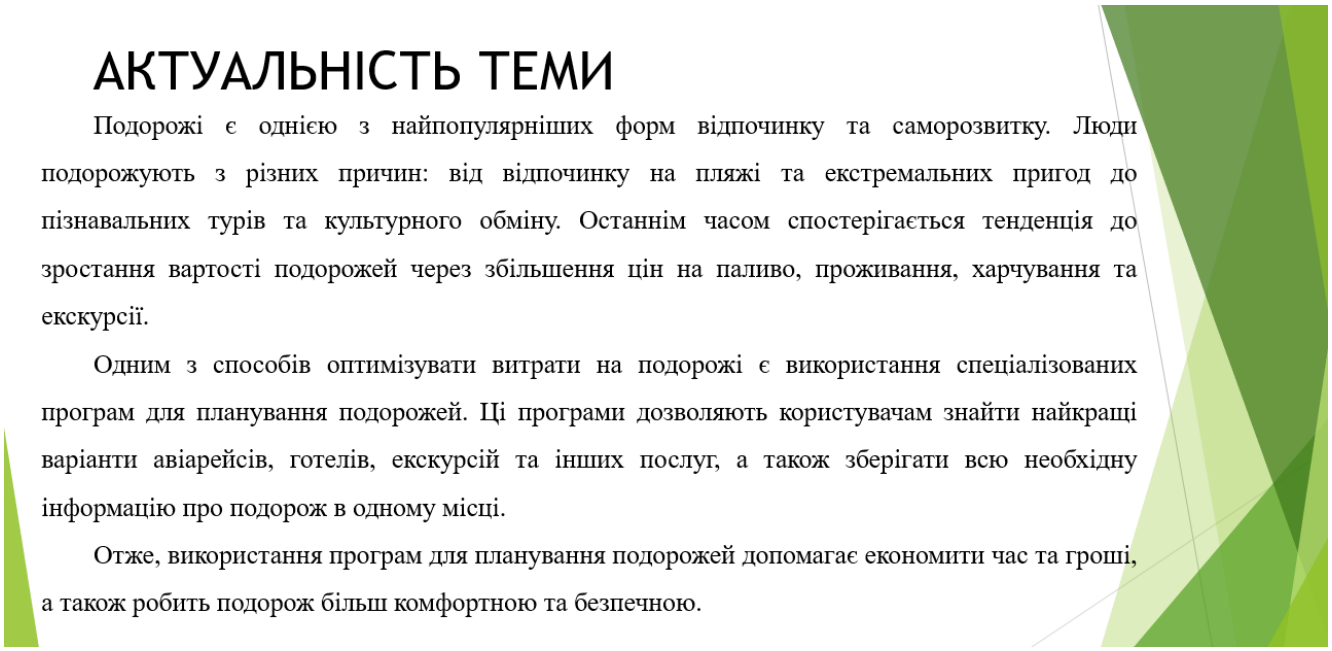


Рисунок Г.2 – Актуальність теми

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Метою роботи є покращення процесу планування подорожі шляхом розробки спеціалізованого програмного застосунку.

Об'єктом дослідження є процеси розробки програмного засобу з планування подорожей.

Предметом дослідження є методи і засоби реалізації програмного засобу з планування подорожей.

Рисунок Г.3 – Мета, об'єкт та предмет дослідження

ЗАВДАННЯ ДОСЛІДЖЕННЯ

Відповідно до поставленої мети необхідно виконати такі завдання:

- проаналізувати предметну область;
- провести порівняльний аналіз аналогів;
- провести аналіз методів розв'язання задачі;
- розробити модель застосунку;
- розробити метод формування маршруту подорожі;
- розробити алгоритми роботи програми;
- провести варіантний аналіз засобів реалізації застосунку;
- розробити функціонал застосунку;
- розробити інтерфейс користувача;
- провести аналіз методів тестування;
- сформувати системні вимоги для роботи програмного засобу;
- провести тестування розробленого застосунку.

Рисунок Г.4 – Завдання дослідження

НОВИЗНА

1. Здобув модального розвикту метод формування маршруту подорожі, у якому, на відміну від існуючих, використовується штучний інтелект для формування маршруту подорожі між двома містами, рекомендацій щодо визначних місць, які можна відвідати на маршруті подорожі, що дозволяє полегшити процес планування подорожі.
2. Здобула подальшого розвитку модель системи, у якій, на відміну від існуючих, міститься комплексний функціонал для планування подорожей з використанням Google Palm API та Visual Weather Crossing API, що дозволяє підвищити інформативність програмного засобу та краще спланувати подорож користувачем.

Рисунок Г.5 – Новизна

ПРАКТИЧНА ЦІННІСТЬ

Практична цінність отриманих результатів. Розроблений програмний засіб можна використати для планування подорожей, бронювання готелів, планування витрат, перегляду прогнозу погоди, отримування рекомендацій для подорожі залежно від вказаного користувачем маршруту.

Рисунок Г.6 – Практична цінність

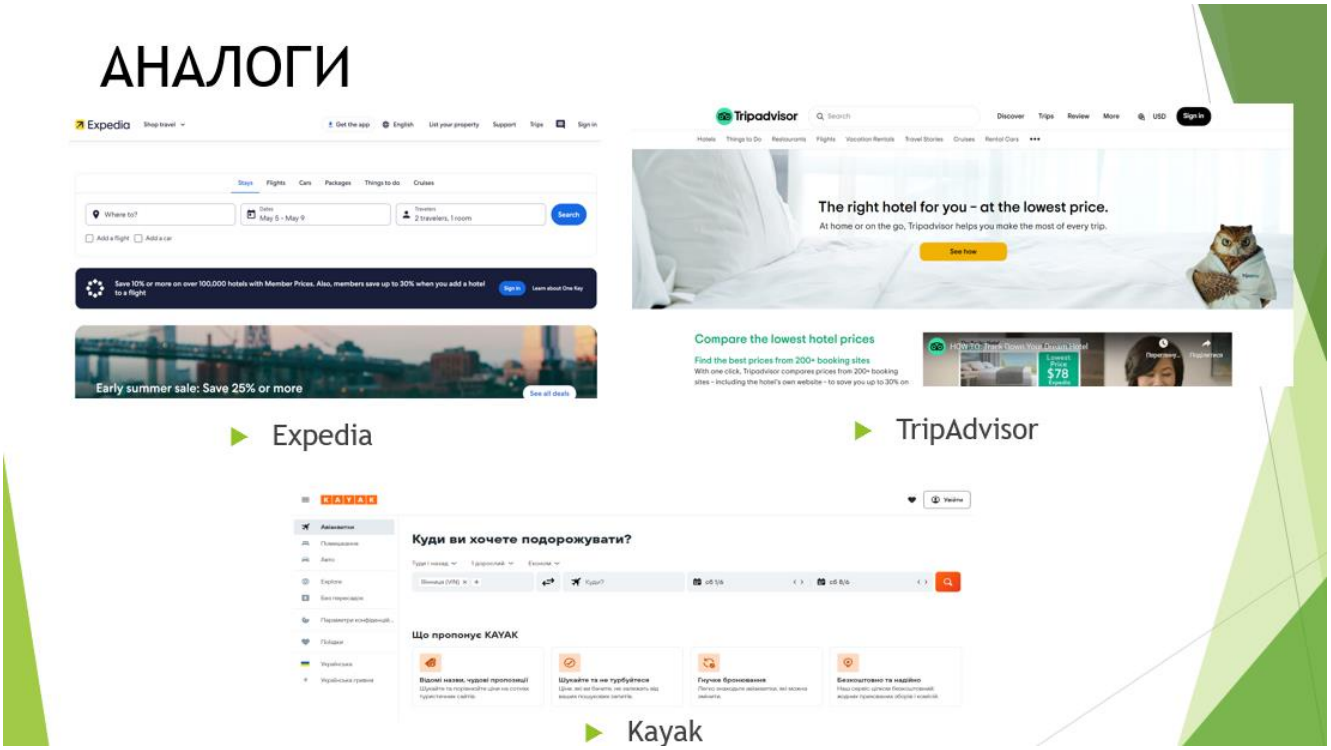


Рисунок Г.7 – Аналоги

Характеристика	Expedia	TripAdvisor	Kayak	Власна розробка
Бронювання готелів	1	1	0	1
Бронювання авіаквитків	1	0	1	1
Відстеження цін	1	1	1	1
Пошук за фільтрами	0	1	1	1
Сумарний бал	3	3	3	4

Рисунок Г.8 – Порівняльний аналіз аналогів

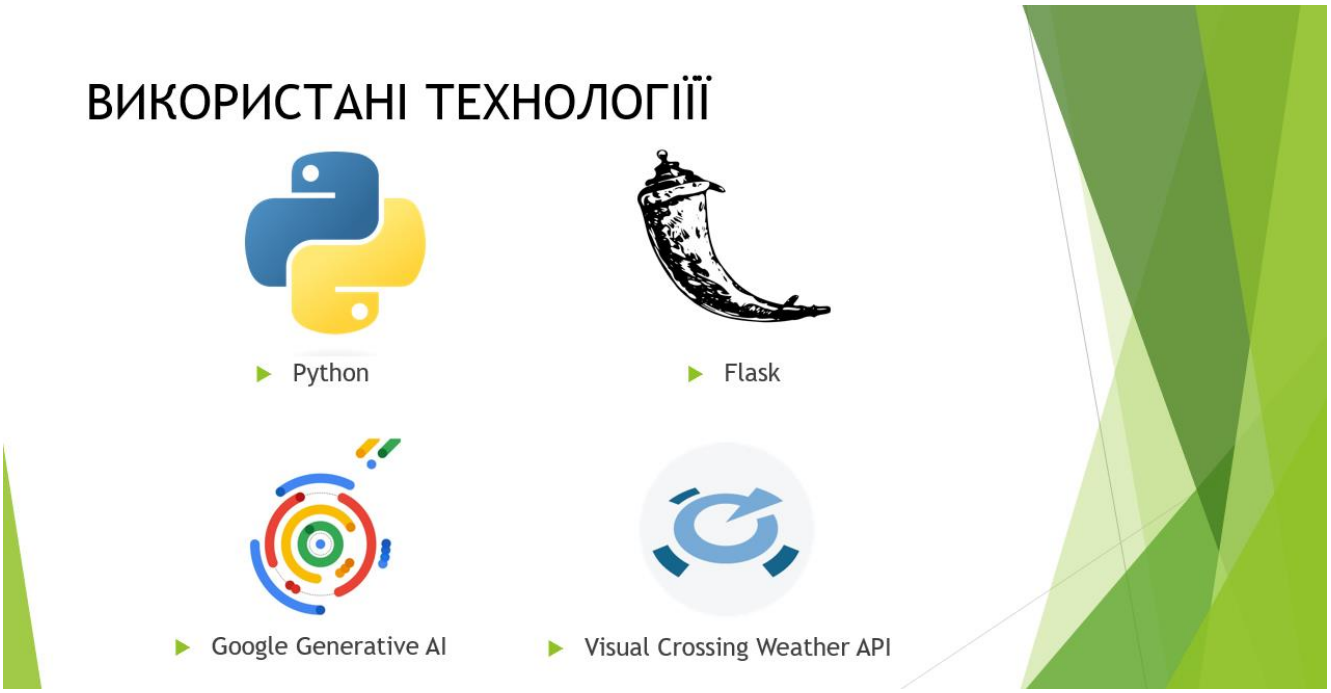


Рисунок Г.9 – Використані технології

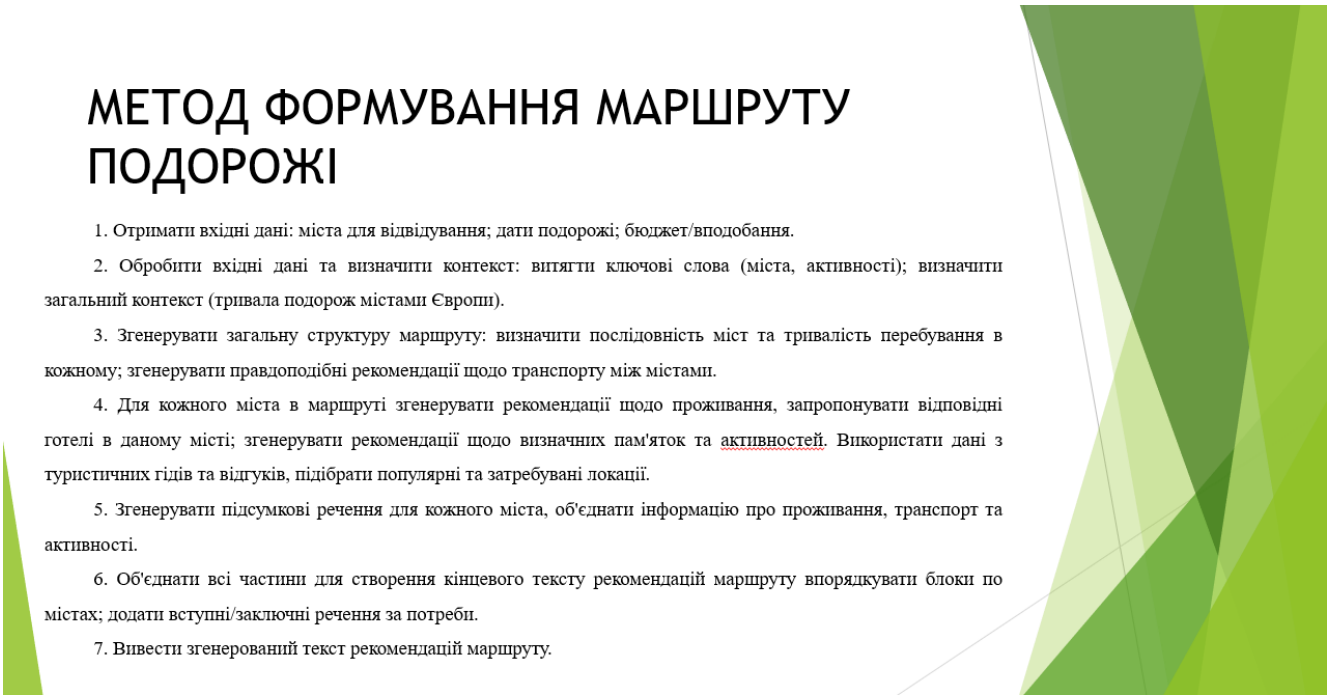


Рисунок Г.10 – Метод формування маршруту подорожі

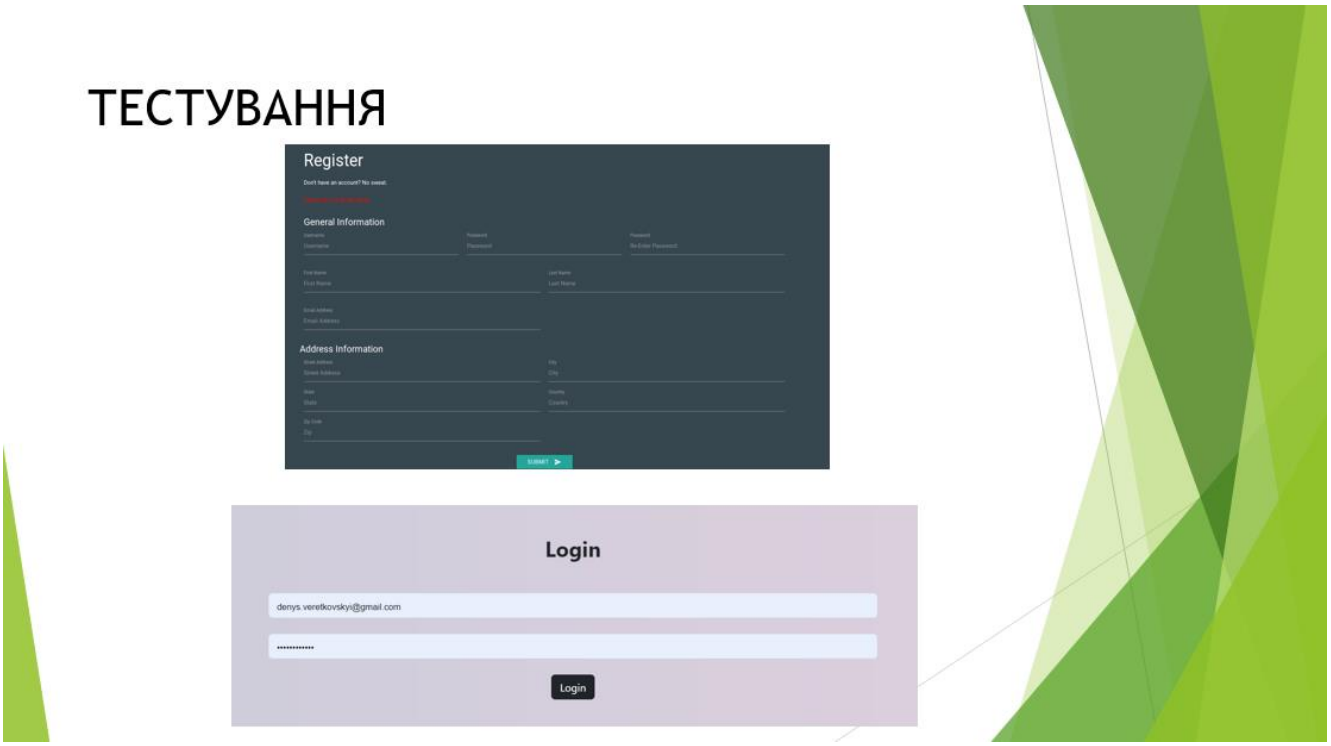


Рисунок Г.11 – Тестування вікон реєстрації та авторизації

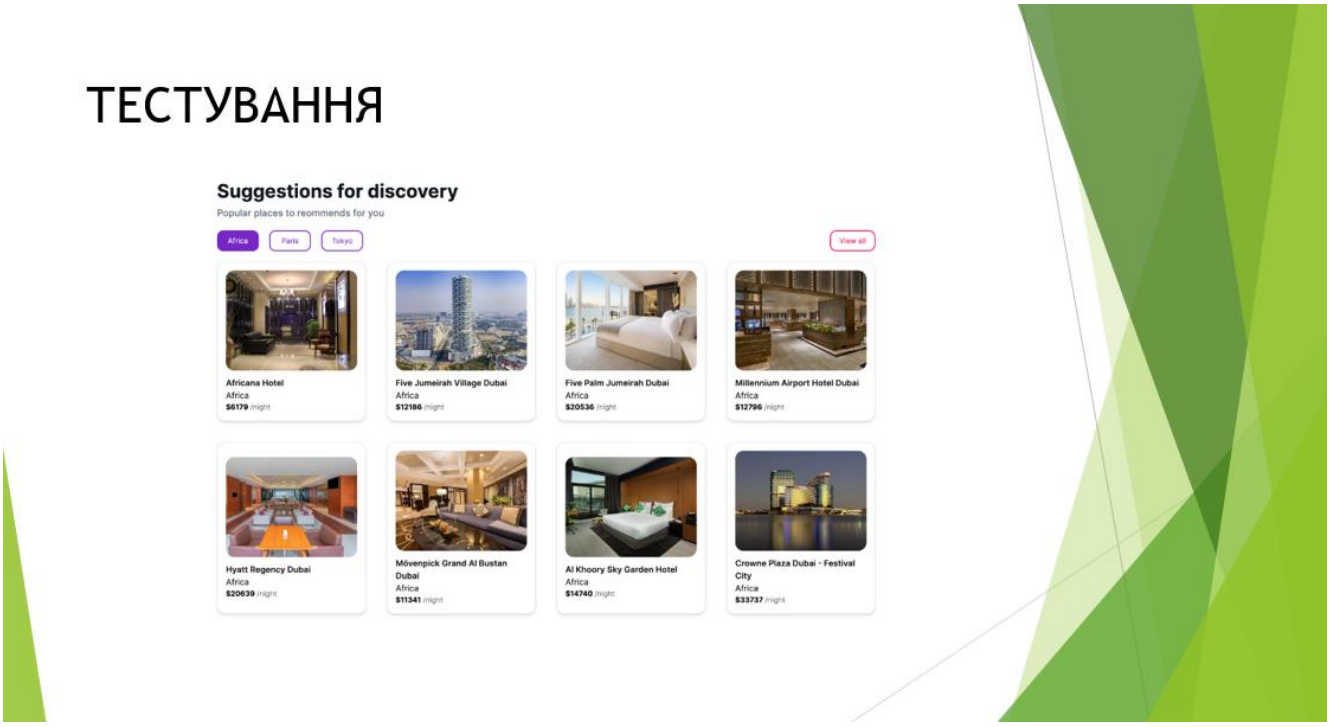


Рисунок Г.12 – Тестування рекомендацій для подорожі

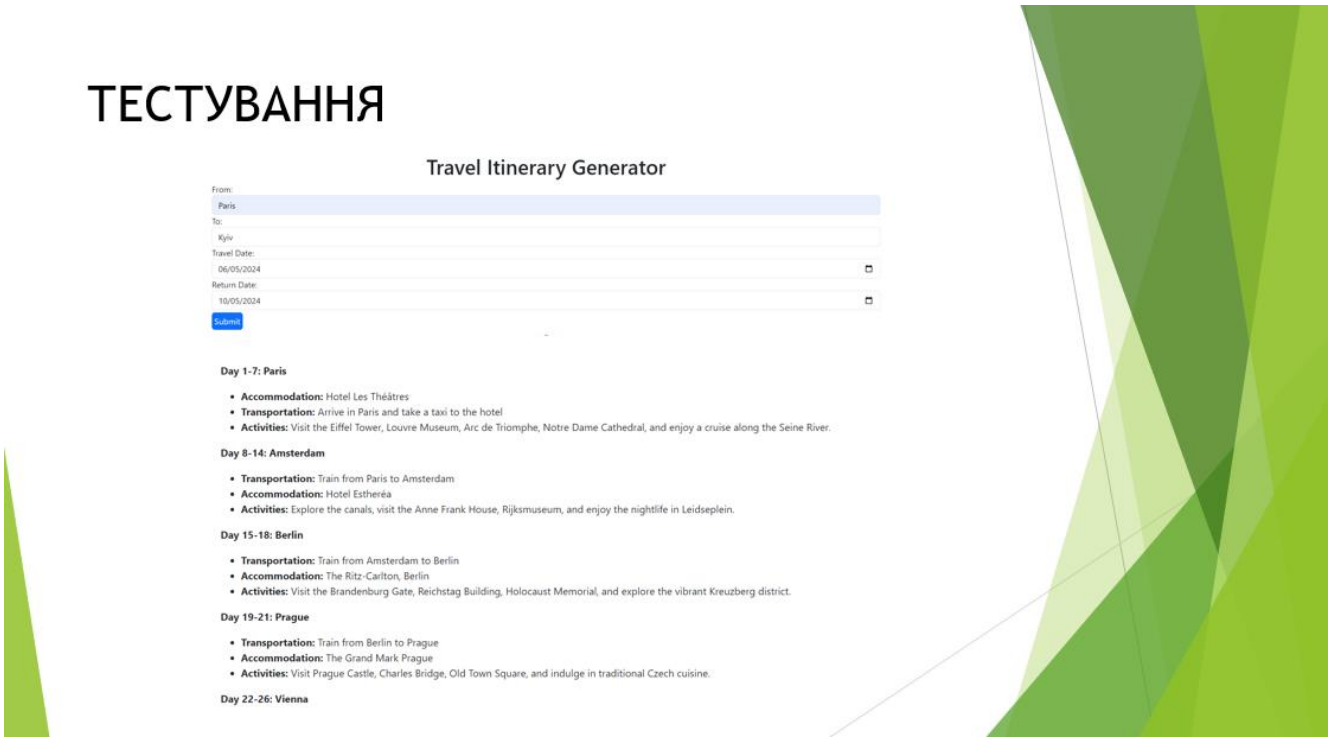


Рисунок Г.13 – Тестування планування подорожі

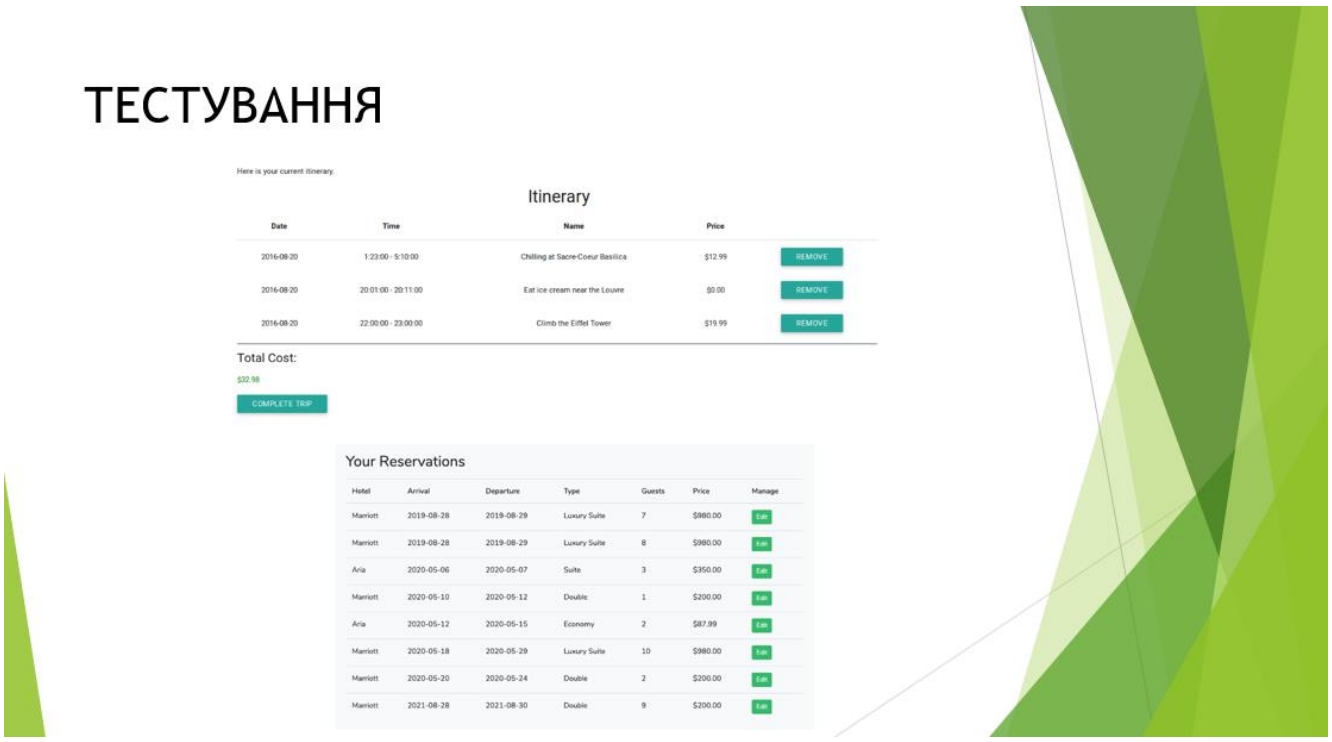


Рисунок Г.14 – Тестування бронювання готелів

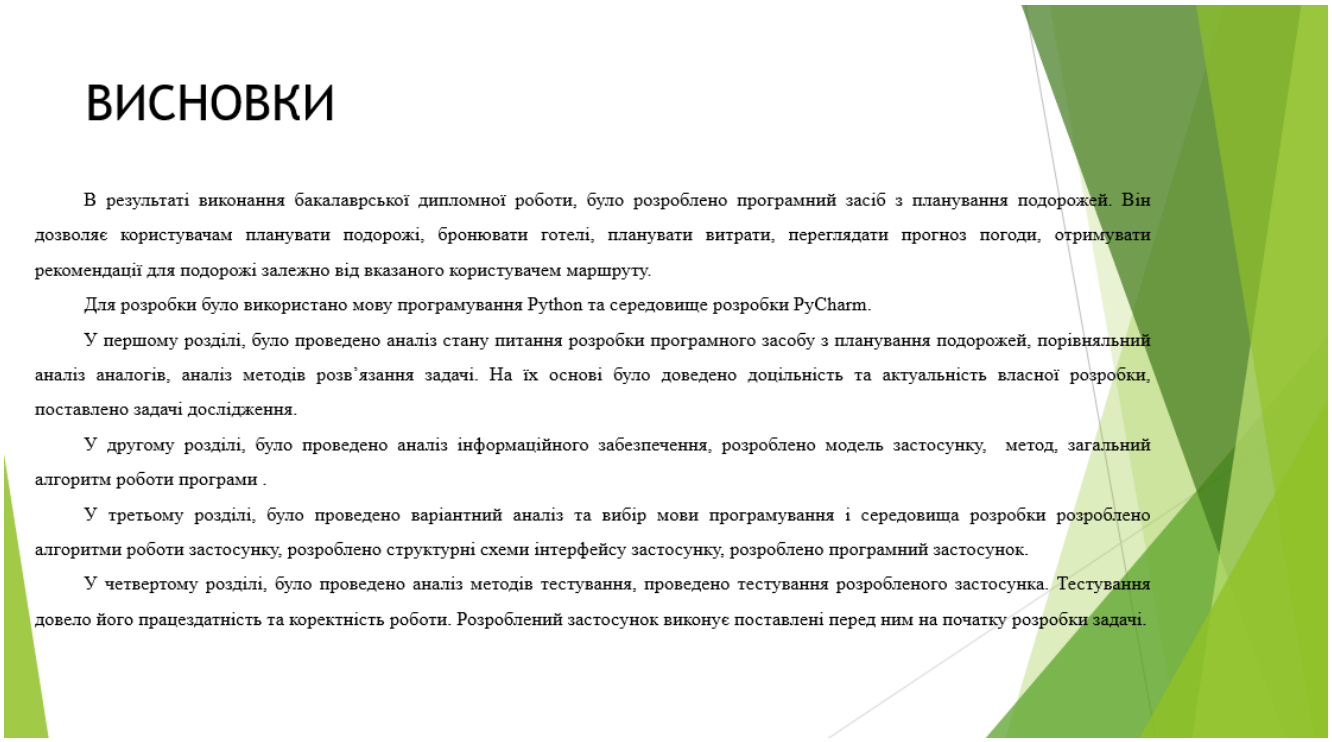


Рисунок Г.15 – Висновки