МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« 01 » 06 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ
ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *WEB-додаток пошуку та бронювання авіаквитків*

Група: *321* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти _____ *Руслан СІГАЄВ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Андрій КРАВЧАТИЙ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення
пояснювальної записки _____ *Оксана ОСАДЧА*
(підпис) (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК* (ім'я, ПРІЗВИЩЕ)
« *01* » *03* 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

СІГАЄВУ Руслану Артемовичу
(прізвище, ім'я, по батькові)

1.Тема роботи *WEB-додаток пошуку та бронювання авіаквитків*

Керівник роботи *Кравчатий Андрій Володимирович, викладач*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « *04* » *04* 2025 року № *50-ст*

2. Строк подання здобувачем освіти роботи з _____ по _____

3. Вихідні дані до роботи *Онлайн-пошук авіаквитків, системи бронювання CRS і GDS, технології Python, Flask, SQLite, функції реєстрації та бронювання.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз механізму онлайн-бронювання квитків
Обґрунтування вибору програмного забезпечення для створення веб-застосунку
Розробка веб-додатку пошуку та бронювання авіаквитків

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження технічного завдання з керівником кваліфікаційної роботи*	*04.04.2025-07.04.2025*	*виконано*
2	*Підбір та вивчення науково-технічної літератури за темою кваліфікаційної роботи*	*08.04.2025-14.04.2025*	*виконано*
3	*Аналіз механізму онлайн-бронювання квитків*	*15.04.2025-21.04.2025*	*виконано*
4	*Обґрунтування вибору програмного забезпечення для створення веб-застосунку*	*22.04.2025-28.04.2025*	*виконано*
5	*Розробка веб-додатку пошуку та бронювання авіаквитків*	*29.04.2025-23.05.2025*	*виконано*
6	*Написання та оформлення пояснювальної записки*	*26.05.2025-30.05.2025*	*виконано*
7	*Попередній захист кваліфікаційної роботи*	*09.06.2025-12.06.2025*	*виконано*
8	*Захист кваліфікаційної роботи*		

Здобувач освіти _____ *Руслан СІГАЄВ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Андрій КРАВЧАТИЙ*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 6/4/2025
Дата редагування —

Документ прийнятий

Звіт подібності

метадані

Назва організації
Ukrainian national aviation university
Заголовок
КПІ_2025_123_Сігаєв
Автор Науковий керівник / Експерт
СігаєвКравчатий А.
підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

5.73% 5.73% КП 1

2.68% 2.68% КЦ

25
Довжина фрази для коефіцієнта подібності 2

7245
Кількість слів

56213
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		2
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		33

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розроблення застосунку для управління персоналом аутстаф-компанії [redacted] 5/23/2024 Kharkiv National University of Economics named after S.Kuznets (KNUE) (KNUE)	21 0.29 %
2	ФКПІ_2024_421_Мільченко_А.Ю [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	16 0.22 %

РЕФЕРАТ

Кваліфікаційна робота «WEB-додаток пошуку та бронювання авіаквитків» містить _44_ сторінки основного тексту, _27_ рисунків, _1_ таблицю, _7_ додатків, _11_ використаних джерел.

ВЕБ-ДОДАТОК, АВІАКВИТКИ, ОНЛАЙН-БРОНЮВАННЯ, *FLASK*, БАЗА ДАНИХ, КЛІЄНТ-СЕРВЕР.

Кваліфікаційна робота присвячена створенню веб-додатку для онлайн-пошуку та бронювання авіаквитків. Такий сервіс є актуальним у сучасних умовах цифровізації авіаперевезень і сприяє зручному плануванню подорожей, покращенню сервісу та взаємодії між клієнтами й перевізниками.

У першому розділі проаналізовано розвиток онлайн-бронювання та вплив цифрових технологій на галузь. Розглянуто принципи роботи систем резервування (*CRS*) і глобальних систем дистрибуції (*GDS*), що дозволило зрозуміти структуру сучасних сервісів бронювання.

У другому розділі обґрунтовано вибір стеку технологій. Для бекенду використано *Python* і *Flask*, шаблонізатор *Jinja*, базу даних *SQLite*. Клієнтську частину реалізовано на *HTML*, *CSS* і *JavaScript*. Такий підхід дозволив створити простий, функціональний веб-додаток.

У третьому розділі описано реалізацію функцій: реєстрація, пошук рейсів, бронювання та обробка даних. Проведено тестування, усунено помилки, оптимізовано інтерфейс і логіку.

Результатом стала ефективна система з базовим функціоналом, яку можна розширити онлайн-оплатою, фільтрами, кабінетом користувача тощо. Робота підтверджує практичну підготовку автора та перспективність застосованих рішень.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - *application programming interface*

ARS - *Airlines Reservation System*

CRS - *Computer Reservations System*

GDS - *Global Distribution System*

БД - база даних

ПЗ - програмне забезпечення

ВСТУП

Світ пережив найбільший технологічний бум за останні п'ятдесят років. Інновації в кожній галузі стали можливими зараз, щоб зробити життя людини простішим та комфортнішим. Авіаційна галузь перетворилася на одну з найскладніших та найцікавіших галузей сьогодення. Авіаційна галузь втілила мрію людей про польоти в реальність менш ніж за століття. Сьогодні мільйони людей літають щодня. Це зміцнило не лише економіку місць, а й поєднало людей та культури. Розвиток технологій призвів до великого прогресу в системі бронювання авіаквитків протягом багатьох років.

Квитки – це документи, що підтверджують покупку та гарантують місце на літак для обраної подорожі. Квитки потрібні як підтвердження для отримання посадкового талона в аеропорту, який потрібен для посадки на літак. Традиційні квитки на зорі авіаперельотів були паперовими та їх потрібно було отримати в туристичних агентствах або офісах авіакомпаній після покупки. Разом з глобалізацією та розвитком авіаційної галузі змінився також процес придбання квитків. Зі швидким зростанням та використанням Інтернету з 2000-х років придбання квитків стало можливим онлайн. Все менше людей зараз використовують традиційні паперові квитки, тоді як майже всі великі авіакомпанії надали можливість онлайн-квитків, широко відомих як електронні квитки. Сьогодні квиток містить інформацію про ім'я пасажира, дату подорожі, номер рейсу, пункт призначення та відправлення подорожі, тариф, податки, інформацію про багаж, правила змін та повернення коштів, форму оплати та термін дії квитка.

У даній дипломній роботі буде розроблено систему пошуку та онлайн-бронювання квитків, яку можуть використовувати, зокрема, авіакомпанії, а також буде продемонстровано робочий додаток для прикладу онлайн-бронювання. Для збору матеріалів для дипломної роботи будуть використні різні джерела, такі як електронні книги, онлайн-статті та веб-сайти.

РОЗДІЛ 1
АНАЛІЗ МЕХАНІЗМУ ОНЛАЙН-БРОНЮВАННЯ КВИТКІВ

1.1 Вступ

З появою інтернету онлайн-бронювання квитків на авіаперельоти стає дедалі популярнішим. Сьогодні авіакомпанії зосереджуються переважно на задоволенні потреб клієнтів. Компанії роблять це, роблячи подорожі можливими в повністю мобільному та соціальному середовищі з інтелектуальним використанням величезних обсягів даних для забезпечення реальних покращень обслуговування та операційної діяльності. Очікується, що персоналізоване використання мобільних телефонів та Інтернету клієнтами різко зросте, і, за оцінками, до 2026 року 99% авіакомпаній пропонуватимуть мобільну реєстрацію [1].

1.2 Розвиток електронної комерції

Електронна комерція – це комерційні операції з товарами та послугами, що здійснюються через Інтернет. З появою Інтернету електронна комерція стрімко розвивається протягом останніх десятиліть. Сьогодні неможливо уявити світ без електронної комерції. Багато великих компаній по всьому світу пропонують своїм клієнтам можливість вибирати та купувати товари та послуги, не виходячи з дому. В електронній комерції використовуються різні набори правил комунікації та взаємодії у вигляді передачі файлів, електронної пошти та кошиків для покупок. Електронна комерція дозволяє купувати товари та послуги цілодобово, що можливо з будь-якої точки світу через комп'ютер, підключений до Інтернету. Онлайн-шопінг не тільки зручніший, ніж традиційний спосіб фізичного відвідування магазинів, але й пропонує ширший вибір товарів та гарну доступність [2].

Електронна комерція використовує систему електронних платежів для здійснення транзакцій. Безпаперова монетарна система повністю змінила обличчя

світової торгівлі завдяки простим та зручним можливостям, мінімальній паперовій роботі з рахунками-фактурами та мінімальним витратам на оплату праці та адміністрування. Найпоширенішими способами оплати онлайн є кредитні картки, дебетові картки, банківський переказ та через інші компанії, такі як *PayPal*. Завдяки зростанню електронної комерції та її простій та безпроблемній системі електронних платежів, багато підприємств, включаючи авіаційну галузь, постійно зростали протягом багатьох років.

Онлайн-бронювання квитків – одна з багатьох функцій електронної комерції. Все більшій кількості людей літати стало простіше, ніж будь-коли, а проста система онлайн-купівлі квитків є одним з основних факторів збільшення кількості пасажирів, які користуються авіаперельотами. Сьогодні, щоб забронювати авіаквиток у будь-яку точку світу, потрібно лише увійти на веб-адресу авіакомпанії (іноді інші веб-сайти, які шукають найкращі рейси для клієнтів), здійснити пошук та придбати квиток. Реєстрацію також можна пройти онлайн у багатьох великих авіакомпаніях, а посадковий талон можна роздрукувати безпосередньо перед поїздкою до аеропорту [3].

1.3 Система пошуку та бронювання авіаквитків

Система бронювання авіаквитків (*ARS, Airlines Reservation System*) – це вдосконалена комп'ютеризована функція бронювання авіаквитків. *ARS* допомагає в систематичній та ефективній організації бронювань, цін, розкладів та даних клієнтів. Система бронювання авіаквитків сьогодні перетворилася на комп'ютерну систему бронювання (*CRS, Computer Reservations System). ARS,* інтегрована з Глобальною системою розподілу (*GDS, Global Distribution System*), може використовуватися кількома каналами розподілу, такими як туристичні агентства, які потім можуть використовувати її для оренди готелів, бронювання авіаквитків, оренди автомобілів, а також для заходів та турів через єдину систему. *ARS* складається з кількох областей, таких як управління запасами, відображення наявності місць та бронювання, а також котирування тарифів та квитків [4].

1.4 Механізм роботи бронювання авіаквитків

Користувачі можуть легко придбати електронний квиток, перейшовши на веб-сайт продажу квитків, здійснивши пошук та вибір пункту призначення, ввівши такі дані, як ім'я, спосіб подорожі, інформацію про багаж та дати, і, нарешті, здійснивши оплату за допомогою банківських карток, банківського переказу або через компанії онлайн-платежів. Потім електронний квиток надсилається електронною поштою або текстовим повідомленням на телефон клієнта. Якщо раніше туристичні агенти та авіакомпанії допомагали клієнтам у купівлі квитків, то сьогодні, завдяки вдосконаленій інтернет-системі, бронювати авіаквитки самостійно стає все легше. Після того, як клієнт здійснює покупку, електронний запис та деталі квитка зберігаються в базі даних авіакомпанії. БД інтегрована із системою обслуговування пасажирів, яка потім підключається до аеропортів, авіакомпаній, туристичних агентств для обміну інформацією в режимі реального часу.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТОРЕННЯ ВЕБ-ЗАСТОСУНКУ

2.1 Мова програмування *Python*

Python [5] — це мова програмування на основі скриптів. Хоча системи програмування, такі як *C++, Java* тощо, дуже популярні в шкільних та коледжних програмах, *Python* — це просунута мова програмування на основі скриптів, яку можна використовувати в багатьох видах програм, вона проста у вивченні та має простіший синтаксис і семантику. Скриптові мови особливо добре підходять для вступної послідовності програмування. Скриптові мови, як правило, також мають простіший синтаксис і семантику, ніж системні мови. Оскільки *Python* містить прості правила синтаксису, він сприяє створенню складних програм за короткий час.

Python поєднує в собі риси таких парадигм програмування, як об'єктно-орієнтована, імперативний та процедурний стиль. Особливо для нових програм, *Python* уникає використання фігурних дужок, крапки з комою та інших символів, необхідних для синтаксису в інших мовах програмування. *Python* створений для легкого читання та зручного вигляду.

Python ретельний для користувача та використовує прості ключові слова англійською мовою. *Python* використовує пробіли для відступів блоків. Використання функцій та класів не є обов'язковим, тому прості програми можна писати з меншою кількістю рядків коду. Цикли, керуючі структури та використання змінних також значно простіші в *Python*. Сьогодні існує кілька інтерпретаторів *Python,* які можна використовувати в різних операційних системах. В останні роки *Python* є популярною мовою програмування, і доступні різноманітні матеріали для читання та вправ.

Для розробки веб-застосунків на *Python* створено спеціалізовані бібліотеки, які значно спрощують процес програмування. Однією з найпопулярніших є *Flask*

— легкий мікрофреймворк, що дозволяє швидко створювати прості веб-застосунки та *REST API*. Для складніших проєктів використовується *Django*, який має вбудовані засоби для роботи з базами даних, формами, автентифікацією та адмініструванням. Сучасним і високопродуктивним рішенням є *FastAPI*, який підтримує асинхронне програмування та автоматично створює документацію для *API*. Крім того, існують інші корисні інструменти, такі як *Tornado* для роботи в реальному часі та Bottle, який підходить для невеликих або вбудованих веб-застосунків.

Django — це веб-фреймворк, призначений для розробки простих, але водночас і складних веб-додатків. *Django* був створений для сприяння швидкій розробці та чистому, прагматичному дизайну. Деякі з ключових особливостей *Django* — це сприяння швидкій веб-розробці, надійна безпека та вражаюча масштабованість. *Django* є продуктом розробників *World Online* Адріана Головатого та Саймона Віллісона у 2003 році. Коли вони почали розробляти веб-сайти за допомогою *Django*, вони постійно додавали нові функції, які допомагали створювати багаті, інтерактивні веб-сайти все швидше і швидше.

Django поєднує в собі використання інших проектів з відкритим кодом, таких як *Apache*, *Python* і *PostgreSQL*. *Django* — це ефективний інструмент програмування для створення багатофункціональних веб-сайтів. Оскільки *Django* — це веб-фреймворк, він економить час розробника, який потребує створення окремих фрагментів з нуля. *Django* — це набір бібліотек, написаних мовою *Python*, тому для розробки на *Django* потрібне кодування на *Python*. *Django* — це програмне забезпечення (ПЗ) з відкритим вихідним кодом, і сьогодні існує спільнота волонтерів, які керують *Django*. Користувачі можуть постійно робити внесок та вдосконалювати *Django*, тому він сьогодні дуже популярний. Його використовують такі відомі веб-додатки, як *Instagram*, *Pinterest*, *Mozilla* та *Pitchfork*.

2.2 Мова розмітки гіпертексту

Мова гіпертекстової розмітки (*HTML*) [6] – це популярна мова розмітки, яка використовується на веб-сторінках. *HTML* можна легко написати в текстовому

редакторі та протестувати за допомогою веб-браузера. Писати мовою *HTML* легко; за допомогою *HTML* також можна додавати медіафайли та зображення на веб-сторінку. HTML містить спеціальні теги розмітки, такі як *<title>, <h>, <p>* тощо. Наприклад, щоб оголосити заголовок сторінки, заголовок має бути включений до тегів *title*. Аналогічно, абзаци, заголовки та інший вміст на сторінках веб-сайту мають бути включені до відповідних тегів *HTML*.

Зберігати HTML-код легко у простому текстовому файлі з назвою файлу та розширенням *.html* або *.htm*. *HTML* надає розробникам можливість створювати розділи в документі. Як тег *<title>* задає заголовок веб-сторінки, *<H1>*, наприклад, визначає основний вміст веб-сторінки. Аналогічно, за допомогою тегів *H2, H3* тощо, *HTML* створює другорядний вміст. Існують теги для інших функцій, таких як абзаци *(<p>)*, стиль шрифту *(bold)* та таблиці *(<table>)* тощо.

2.3 Мова шаблонів *Jinja*

Jinja [7] — це зручна мова шаблонів, яка використовує *Python*. *Jinja* використовується як основа для високоінтерактивних веб-сторінок. Вона пропонує зручні функції, такі як виконання в ізольованому середовищі, успадкування шаблонів та налаштовуваний синтаксис. Синтаксис *Jinja* можна записати в *HTML*-шаблоні. *Jinja* походить подібно до *Django*. Простота — одна з найкращих особливостей *Jinja*. Відомі компанії, такі як *Mozilla,* використовували *Jinja* для своєї розробки. *Jinja* можна просто записати в текстовому файлі будь-якого формату. Змінні та вирази в шаблоні замінюються значеннями під час рендерингу шаблону.

2.4 Мова програмування *Javascript* та *CSS*

Javascript [8] є однією з найпопулярніших мов сценаріїв для веб-сторінок сьогодні. Javascript (JS) – це об'єктно-орієнтована мова, яка підтримує такі функції, як імперативне та функціональне програмування. Синтаксис схожий на інші

об'єктно-орієнтовані мови, такі як *Java* та *C++*, тому його легко вивчати людям, які знають ці мови. *JS* використовується для додавання інтерактивних функцій, таких як кнопки, анімація, ігри тощо. *JS* був винайдений Брендоном Айхом. *Javascript* здатний створювати багато різних функцій, від початківців до просунутих, таких як *2D* та *3D* елементи на веб-сайті. (Основи *JavaScript 2015*.)

CSS [8] — це мова програмування, розроблена у 1992-1993 роках. *CSS* допомагає в розробці кожного елемента мови розмітки, такої як *HTML*, надаючи дизайнеру повний контроль. У той час як елементи *HTML* дозволяють дизайнерам веб-сторінок додавати бажаний контент, *CSS* дає змогу відображати контент користувачеві. *CSS* охоплює такі області, як кольори, макет, розширені позиції елементів, шрифти, а також дозволяє адаптувати контент до різних пристроїв, таких як телефони, планшети, великі екрани та принтери. *CSS* може працювати незалежно, а також використовуватися з будь-якими мовами розмітки, заснованими на *XML*.

CSS використовує прості, повсякденні англійські слова та має простий синтаксис. *CSS* є вирішальним у просунутому веб-дизайні, оскільки він дає контроль над макетом та пропонує численні методи, щоб зробити веб-сторінку вишуканою. Наразі основні функції *CSS* підтримуються всіма основними браузерами, такими як *Internet Explorer*, *Safari*, *Opera*, *Chrom* та *Faerfox*. Для розробки зразка системи онлайн-бронювання для цієї роботи на веб-сторінках буде використано *CSS* для позиціонування, макета, полів та кольорів елементів *HTML*.

2.5 *Flask* як засіб реалізації серверної частини вебзастосунку

Flask [11] — це легкий веб-фреймворк для мови програмування *Python*, який дозволяє швидко створювати веб-застосунки будь-якої складності. Він побудований на основі *WSGI*-бібліотеки *Werkzeug* та шаблонізатора *Jinja2*. Основною перевагою *Flask* є його мінімалістичність та модульність, що дозволяє розробникам гнучко обирати необхідні компоненти для реалізації конкретного функціоналу.

Однією з ключових особливостей *Flask* є те, що він не нав'язує структуру проєкту, дає змогу розробнику самостійно організувати маршрути, шаблони, обробку запитів, взаємодію з базами даних тощо. Це забезпечує високу швидкість розробки, що особливо актуально для невеликих і середніх за обсягом проєктів.

Переваги *Flask*:

- простота у використанні: Лаконічний синтаксис та зрозумілий *API* дають змогу легко розпочати роботу навіть початківцям.
- гнучкість: Відсутність суворої структури проєкту дозволяє підлаштовувати фреймворк під потреби конкретного застосунку.
- можливість масштабування: Попри свою простоту, *Flask* підходить для створення як невеликих, так і складних веб-систем.
- розвинене ком’юніті: Наявність численних розширень, наприклад *Flask-SQLAlchemy* (ORM), *Flask-Login* (аутентифікація), *Flask-WTF* (форми), дозволяє розширювати функціонал без суттєвих витрат часу.
- інтеграція з *Jinja2*: Зручне відображення даних на *HTML*-сторінках із можливістю використання циклів, умов, макросів тощо.

У межах цього проєкту планується використання саме *Flask* для реалізації серверної логіки: обробки *HTTP*-запитів від клієнта, взаємодії з базою даних *SQLite*, динамічного формування *HTML*-сторінок за допомогою шаблонів *Jinja*, а також забезпечення маршрутизації та базової обробки помилок.

Вибір саме *Flask* як основи серверної частини обумовлений його простотою, достатнім функціоналом для поставлених задач і гарною підтримкою *Python*-екосистеми.

2.6 База даних *SQLite*

Бази даних (БД) – це колекції подібних даних. БД використовуються для організованого збору та зберігання подібних даних для подальшого використання для певних цілей. БД містить таблиці з рядками та стовпцями, заповненими об'єктами, що відображає зв'язок між ними. БД виступає спільним ресурсом для

програм, які можуть використовувати інформацію з бази даних. Багато підприємств сьогодні покладаються на БД для систематичного зберігання широкого спектру інформації. БД використовуються майже скрізь: у невеликих компаніях, які можуть використовувати базу даних для збереження інформації клієнтів, а також для більш просунутих наукових та військових галузей. БД значно спрощують завдання пошуку сотень і тисяч записів, зберігаючи їх організовано. Серед багатьох програм для роботи з базами даних, доступних сьогодні.

Для збереження та обробки даних у *Web*-застосунках часто використовують вбудовану базу даних *SQLite,* яка є легкою та не потребує окремого серверного ПЗ. Вона ідеально підходить для невеликих проєктів, прототипів або настільних застосунків, де не потрібна складна система управління базами даних. У *Python* доступ до *SQLite* забезпечує стандартний модуль sqlite3, який дозволяє створювати таблиці, виконувати *SQL*-запити та обробляти результати без додаткових бібліотек. Завдяки простому синтаксису і сумісності з *SQL*, *SQLite* легко інтегрується у веб-застосунки, особливо при використанні *Flask* чи *Django*. Ця база даних забезпечує надійність, збереження даних у звичайному файлі та зручну роботу в середовищах з обмеженими ресурсами. БД *SQLite* буде використана в розробці цієї роботи.

2.7 Висновок до другого розділу

У цьому розділі було обґрунтовано вибір технологій та програмного забезпечення, необхідного для створення веб-застосунку системи онлайн-бронювання авіаквитків. Основною мовою програмування обрано *Python* завдяки її простому синтаксису, широкій підтримці бібліотек та зручності для розробки серверної частини веб-застосунків. Для побудови інтерфейсу користувача використано *HTML* для структурування вмісту, *CSS* для оформлення та адаптивного дизайну, а також *JavaScript* для додавання інтерактивності.

В якості шаблонізатора було обрано Jinja, який органічно інтегрується з *Python* та забезпечує гнучке формування *HTML*-сторінок. Для реалізації серверної логіки обрано *Flask* — мікрофреймворк, що дозволяє швидко створювати веб-

застосунки з *REST API,* має низький поріг входження, добре масштабується та легко налаштовується.

Для збереження даних застосунок використовує *SQLite* — легку реляційну базу даних, яка не потребує окремого сервера, ідеально підходить для невеликих або середніх проєктів, та безпосередньо підтримується *Python* через модуль *sqlite3*.

Загалом, вибрані інструменти є взаємодоповнювальними, забезпечують зручність розробки, достатню продуктивність, легкість у розгортанні та підтримку ключових функціональних можливостей, необхідних для реалізації поставленого завдання.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-ДОДАТКУ ПОШУКУ ТА БРОНЮВАННЯ АВІАКВИТКІВ

3.1 Визначення структури проекту

У рамках дипломної роботи було розроблено *WEB*-додаток пошуку та бронювання авіаквитків, що реалізований за допомогою сучасних веб-технологій. Додаток дозволяє користувачам здійснювати пошук доступних авіарейсів за вказаними параметрами (місто відправлення, місто прибуття, дата) та бронювати квитки онлайн.

Для реалізації веб-додатка було використано наступні інструменти та технології:

- мова програмування: *Python*;
- фреймворк: *Flask* — мікрофреймворк для створення веб-застосунків;
- система керування базами даних: *SQLite*;
- мова розмітки: *HTML*;
- стилі оформлення: *CSS*;
- шаблони: *Jinja2* — механізм шаблонів, вбудований у *Flask*;
- *JavaScript* — базова функціональність взаємодії з інтерфейсом.

Структура проєкту представлена на рисунку 3.1.

```
flight_search_app/
├── app.py
├── init_db.py
├── /data/
│   └── flights.db
├── /templates/
│   ├── index.html
│   ├── results.html
│   └── booking.html
├── /static/
│   ├── style.css
│   └── /img/
│       └── bg.jpg
```

Рисунок 3.1 – Структура проєкту *WEB*-додатку

Файл app.py реалізує маршрутизацію, логіку обробки запитів користувача, підключення до бази даних та рендеринг *HTML*-шаблонів.

Файл *init_db.py* використовується для ініціалізації бази даних та наповнення її тестовими записами про рейси.

Каталог *templates*/ містить шаблони сторінок, які динамічно формуються залежно від введених користувачем даних.

Каталог *static*/ містить статичні файли, які використовуються для візуального оформлення: *CSS*-стилі, фонове зображення тощо.

3.2 Розробка керуючих програм на *Python*

3.2.1 Розробка головного модулю *app.py*

Основна логіка роботи веб-додатку реалізована у файлі app.py, який є точкою входу до застосунку. У цьому модулі налаштовується *Flask*-додаток, описуються маршрути (*routes*), обробники запитів, робота з базою даних та інтеграція з модулем бронювання.

Імпорт бібліотек та налаштування показано на рисунку 3.2.

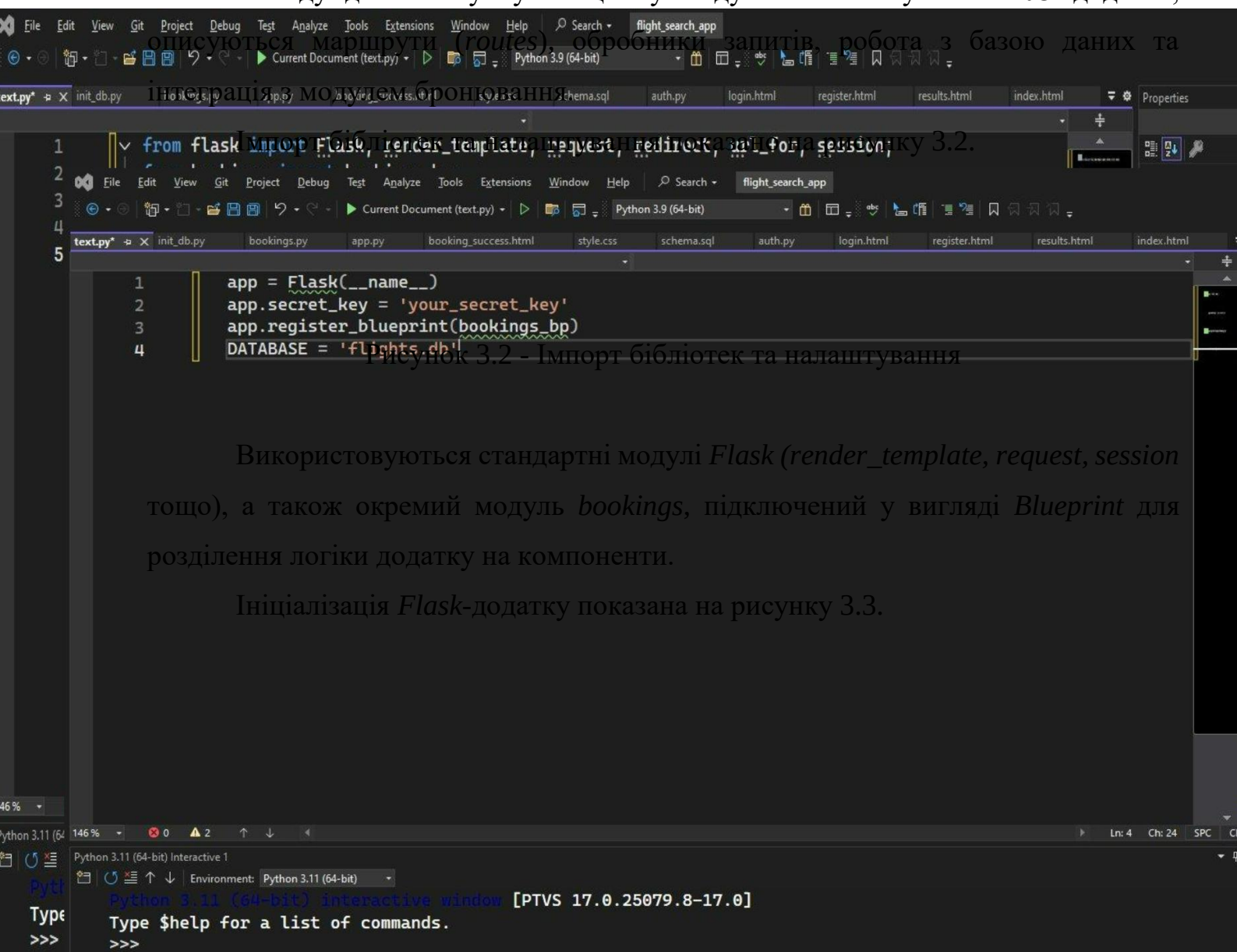


Рисунок 3.2 - Імпорт бібліотек та налаштування

Використовуються стандартні модулі *Flask (render_template, request, session* тощо), а також окремий модуль *bookings*, підключений у вигляді *Blueprint* для розділення логіки додатку на компоненти.

Ініціалізація *Flask*-додатку показана на рисунку 3.3.

Рисунок 3.3 - Ініціалізація *Flask*-додатку

Створюється екземпляр додатку *app*. Встановлюється секретний ключ для сесій. Підключається додатковий маршрут (*blueprint*) для бронювання квитків. Вказується шлях до файлу бази даних.

Підключення до бази даних показано на рисунку 3.4.

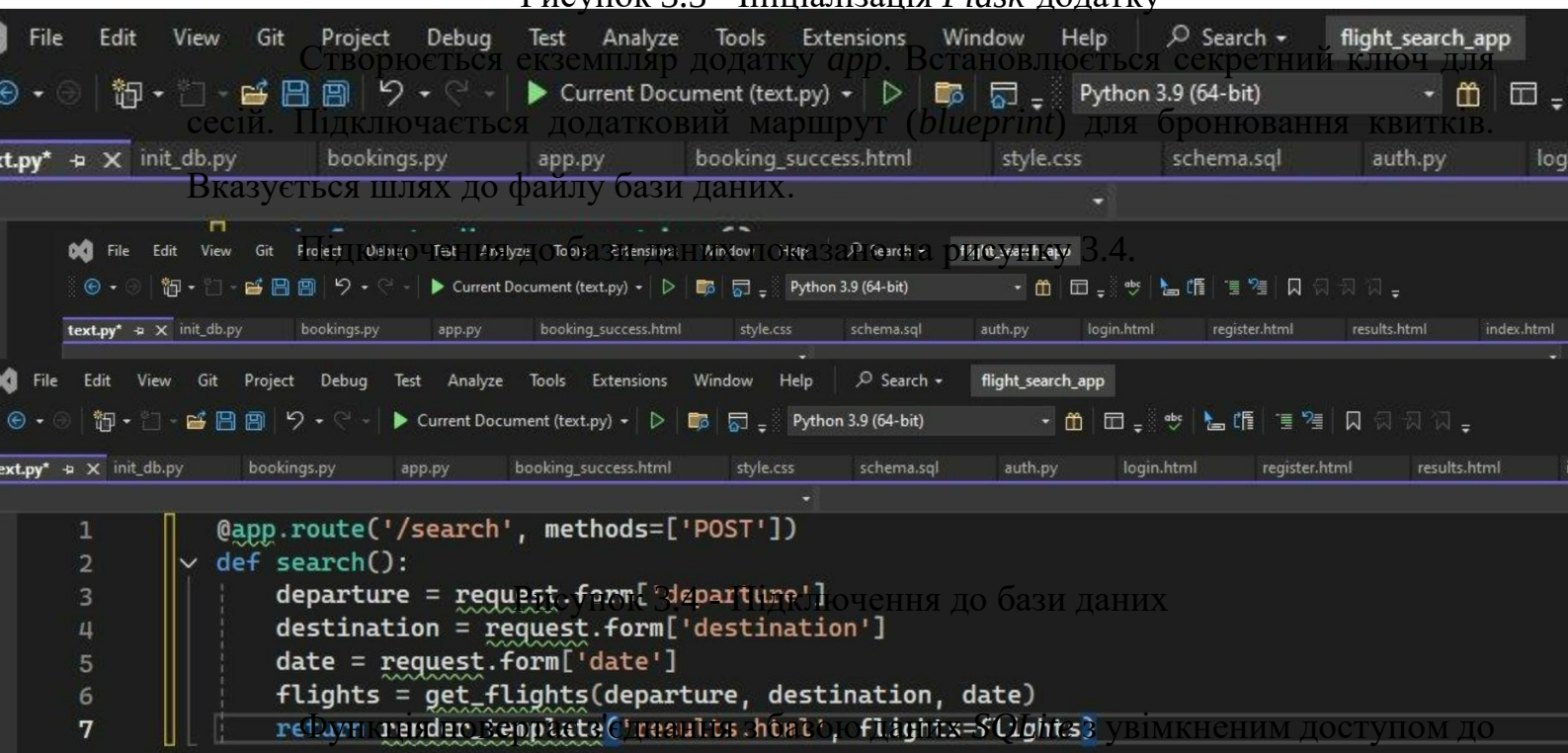


Рисунок 3.4 - Підключення до бази даних

Функція `get_db()` відкриває з'єднання з увімкненим доступом до полів за назвами стовпців.

Основні маршрути веб-додатку показано на рисунку 3.5.

Рисунок 3.5 - Основні маршрути веб-додатку

Відображає стартову сторінку з формою пошуку авіарейсів.

Пошук рейсів (*/search*) (рисунок 3.6):

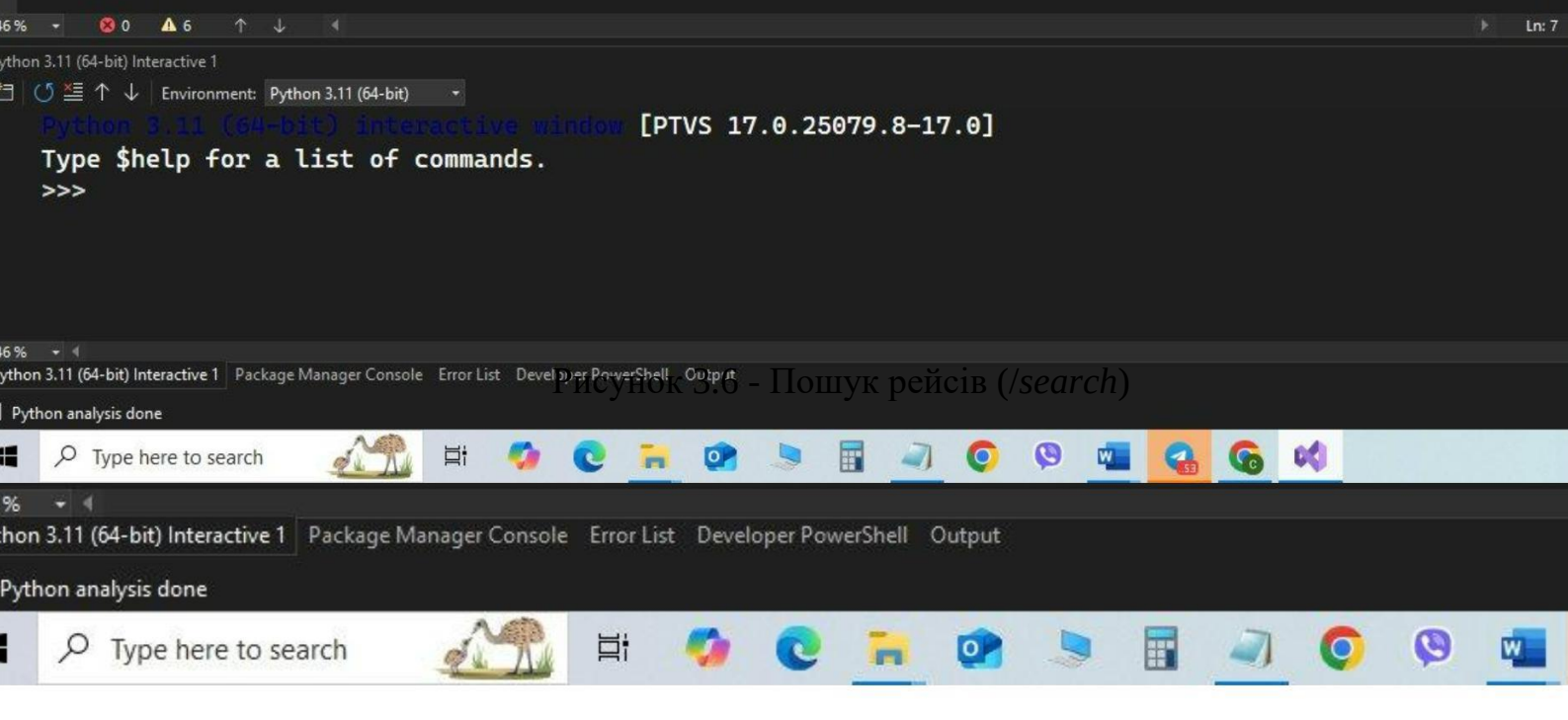


Рисунок 3.6 - Пошук рейсів (*/search*)

Обробляє дані форми пошуку, викликає функцію get_flights() і передає знайдені результати в шаблон results.html.

Пошук у базі даних (рисунок 3.7).

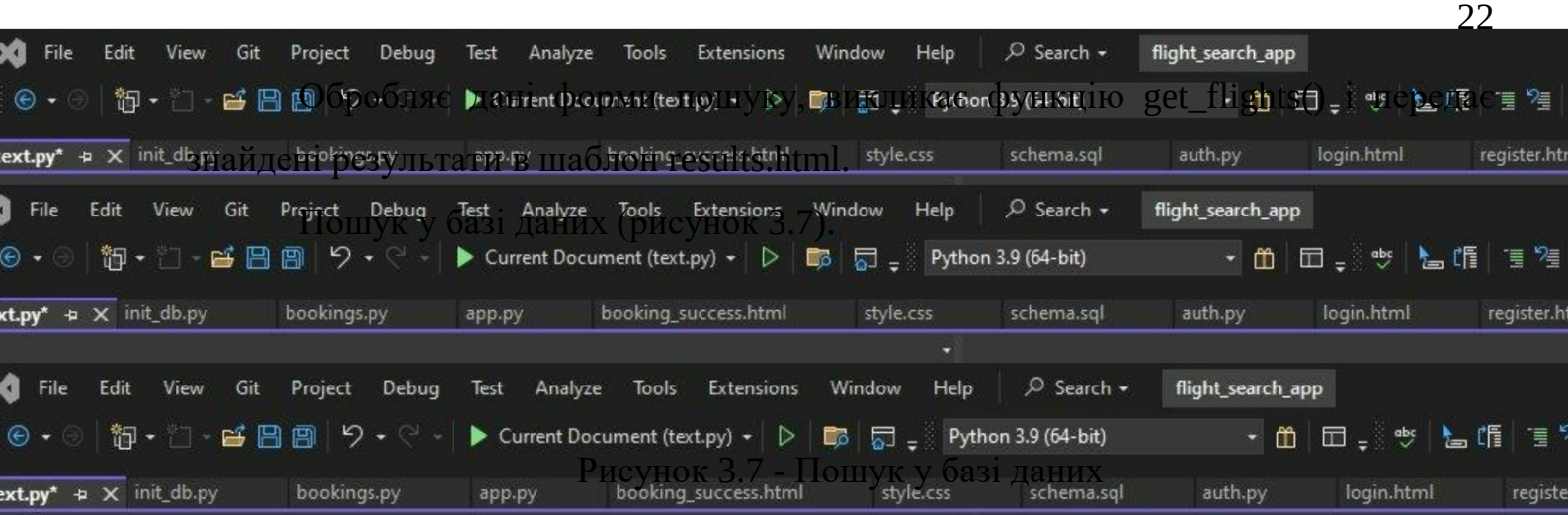
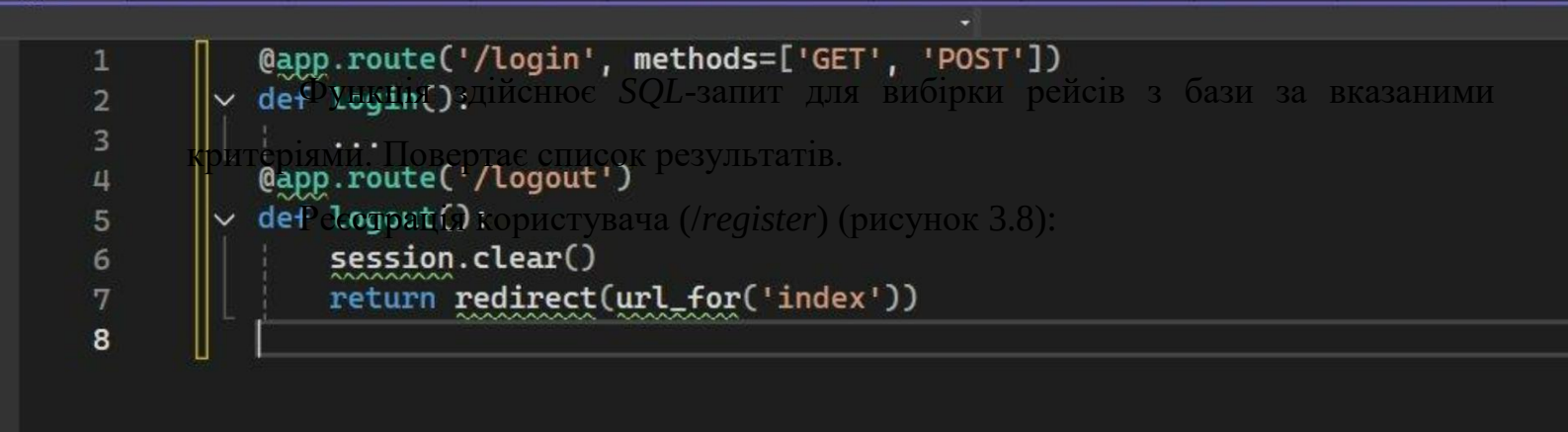


Рисунок 3.7 - Пошук у базі даних

Функція здійснює *SQL*-запит для вибірки рейсів з бази за вказаними критеріями. Повертає список результатів.

Реєстрація користувача (*/register*) (рисунок 3.8):

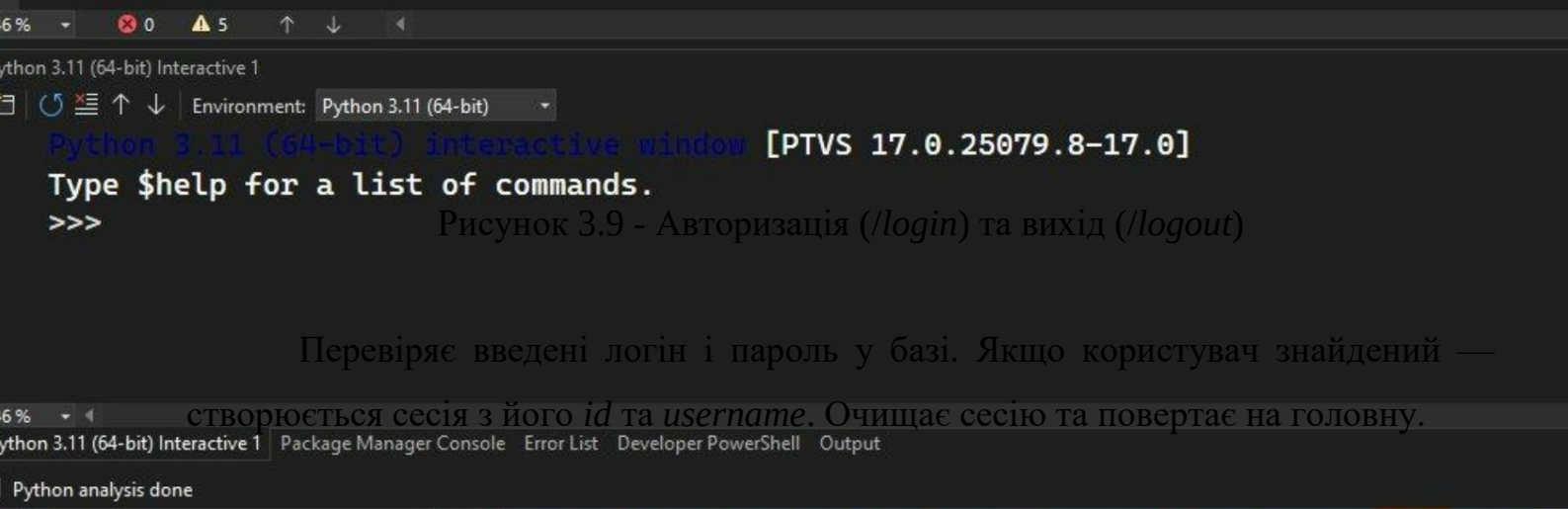


```
@app.route('/login', methods=['GET', 'POST'])
def login():
    ...
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))
```

Рисунок 3.8 - Реєстрація користувача (*/register*)

Якщо запит *GET*, виводить форму реєстрації. Якщо *POST* — намагається зберегти нового користувача до таблиці *users*. У разі дубліката логіна — виводиться повідомлення про помилку.

Авторизація (*/login*) та вихід (*/logout*) (рисунок 3.9).



```
Python 3.11 (64-bit) Interactive window [PTVS 17.0.25079.8-17.0]
Type $help for a list of commands.
>>>
```

Рисунок 3.9 - Авторизація (*/login*) та вихід (*/logout*)

Перевіряє введені логін і пароль у базі. Якщо користувач знайдений — створюється сесія з його *id* та *username*. Очищає сесію та повертає на головну.

Бронювання квитка (*/book/<int:flight_id>*) (рисунок 3.10):

```
if __name__ == '__main__':
    if not os.path.exists(DATABASE):
        print("База даних не знайдена...")
    app.run(debug=True)
```

Рисунок 3.10 - Бронювання квитка (*/book/<int:flight_id>*)

Перевіряє, чи користувач авторизований. Якщо так — перевіряє, чи вже є бронювання цього рейсу. Якщо ні — додає запис до таблиці *bookings*.

Запуск додатку:

```
from flask import Blueprint, render_template, redirect, url_for, session, request
import sqlite3
bookings_bp = Blueprint('bookings', __name__)
DATABASE = 'flights.db'
```

Рисунок 3.11 - Запуск додатку

На етапі запуску перевіряється наявність бази даних. Якщо файл відсутній, виводиться підказка щодо його створення.

3.2.2. Розробка *Python*-модуля бронювання авіаквитків *bookings.py*

У даному підрозділі розглянуто реалізацію функціоналу бронювання рейсів користувачами, які виконали вхід до системи. Весь функціонал винесено у окремий модуль, реалізований з використанням *Blueprint* у *Flask*, що дозволяє організувати код у вигляді логічно відокремлених компонентів.

Створення *Blueprint* (рисунок 3.12):

Рисунок 3.12 - Створення *Blueprint*

Створено *Blueprint* з ім'ям *'bookings'*, який відповідає за обробку маршрутів, пов'язаних із бронюванням.

Задано назву бази даних, у якій зберігаються рейси та інформація про бронювання (*flights.db*).

Функція з'єднання з базою даних (рисунок 3.13).

```
def get_db_connection():
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    return conn
```

Рисунок 3.13 - Функція з'єднання з базою даних

Забезпечується централізоване відкриття з'єднання з базою даних. Встановлення *row_factory = sqlite3.Row* дозволяє звертатися до стовпців через імена, а не індекси, що покращує читабельність коду.

Обробка бронювання відбувається за допомогою команди *@bookings_bp.route ('/book/<int:flight_id>', methods=['GET', 'POST']) def book(flight_id).*

Функція book відповідає за бронювання квитків для обраного рейсу. Вона реалізує такі кроки:

- перевірка авторизації за допомогою команди *if 'user_id' not in session: return redirect(url_for('login')).*

Якщо користувач не авторизований — його перенаправляють на сторінку входу. Таким чином, бронювання доступне лише зареєстрованим користувачам.

- отримання даних про рейс (рисунок 3.14);

Рисунок 3.14 - Отримання даних про рейс

Перевіряється, чи існує рейс з переданим *flight_id*. У разі відсутності відповідного запису повертається повідомлення про помилку (*HTTP-код 404*).

- перевірка наявності бронювання (рисунок 3.15):

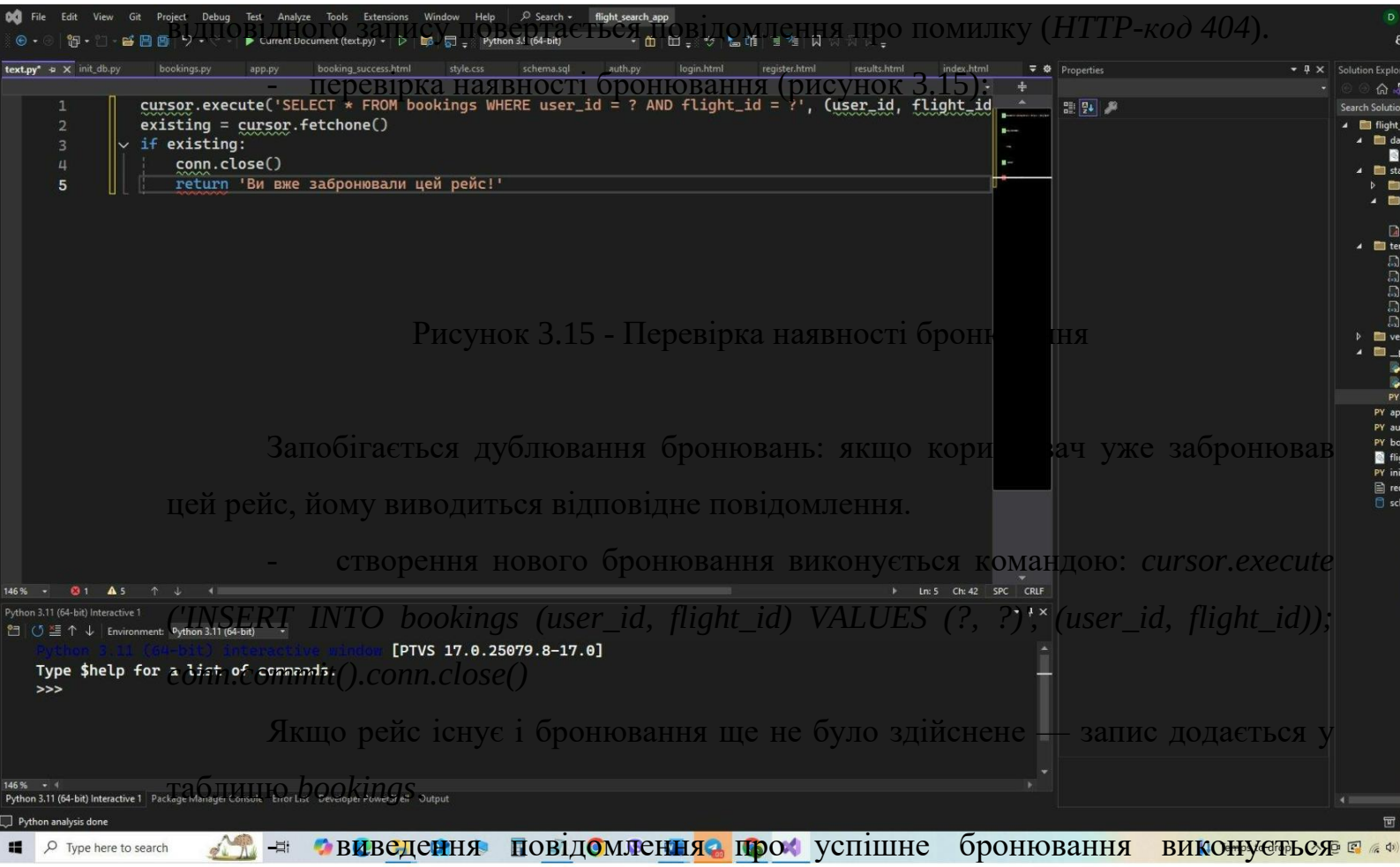


Рисунок 3.15 - Перевірка наявності бронювання

Запобігається дублювання бронювань: якщо користувач уже забронював цей рейс, йому виводиться відповідне повідомлення.

- створення нового бронювання виконується командою: *cursor.execute ('INSERT INTO bookings (user_id, flight_id) VALUES (?, ?)', (user_id, flight_id)); conn.commit().conn.close()*

Якщо рейс існує і бронювання ще не було здійснене — запис додається у таблицю *bookings*.

- виведення повідомлення про успішне бронювання виконується командою: *return render_template('booking_success.html', flight=flight).*

Після успішного бронювання користувачеві виводиться сторінка підтвердження, що містить деталі забронюваного рейсу.

Завдяки використанню *Blueprint*, функціонал бронювання реалізовано окремо від основного коду додатку, що забезпечує модульність та зручність розширення. Передбачено:

- перевірку автентифікації користувача;
- валідацію рейсу;
- перевірку дублювання бронювання;
- захист від несанкціонованих операцій.

Це дозволяє забезпечити базовий рівень безпеки та зручності взаємодії з системою для кінцевого користувача.

3.2.2. Розробка *Python* модуля для створення тестової бази даних *init_db.py*

У зв’язку з повномасштабною військовою агресією російської федерації авіаційне сполучення в Україні тимчасово призупинене, що унеможливлює використання актуальних даних про реальні рейси. Для забезпечення повноцінного тестування функціоналу веб-додатку було створено тестову таблицю з умовними даними про рейси. Це дозволяє перевірити працездатність пошуку, бронювання та інших ключових функцій системи. Тестова база імітує реальні сценарії взаємодії користувача з додатком, забезпечуючи відповідність логіки роботи заданим вимогам.

Даний модуль призначений для створення та ініціалізації тестової бази даних *flights.db,* яка зберігається у директорії *data*/. Він виконує низку операцій, необхідних для підготовки таблиці рейсів перед використанням веб-додатком:

Створення директорії – за допомогою функції *os.makedirs('data', exist_ok=True')* створюється папка *data*, якщо вона ще не існує. Це забезпечує правильне зберігання бази даних у виділеному місці проєкту.

Підключення до бази даних – здійснюється з'єднання з файлом *data/flights.db* за допомогою *SQLite*. Якщо файл не існує, він буде створений автоматично.

Очищення попередніх даних – команда *DROP TABLE IF EXISTS flights* видаляє таблицю *flights*, якщо вона вже існує. Це дозволяє запускати модуль багаторазово для тестування, не накопичуючи старі або дубліковані записи.

Створення таблиці *flights* – створюється нова таблиця з такими полями: id (автоматично зростаючий первинний ключ), *flight_number, departure, destination, date, time, price*. Всі поля обов’язкові для заповнення.

Вставка тестових записів – у таблицю додаються кілька попередньо визначених рейсів з умовними напрямками, датами, часом і цінами. Це дозволяє відразу працювати з веб-додатком, перевіряючи функціонал пошуку та бронювання рейсів.

Завершення роботи – усі зміни зберігаються командою *conn.commit(),* після чого з’єднання з базою даних закривається.

Модуль завершує свою роботу повідомленням у консоль: «Базу даних ініціалізовано успішно.», що свідчить про успішну ініціалізацію таблиці рейсів. Код модуля завершення наведено у додатку А.

3.3 Розробка *frontend*-частини додатку

Фронтенд — це частина веб-додатку, яку бачить і з якою взаємодіє користувач. Він відповідає за зовнішній вигляд сайту, розташування елементів, оформлення, кнопки, форми та динамічну поведінку сторінок. Фронтенд працює безпосередньо у веббраузері на стороні клієнта (користувача). Основними технологіями для фронтенду є *HTML* (структура сторінки), *CSS* (оформлення стилів) і *JavaScript* (інтерактивність). Сучасні фреймворки, як-от *React, Vue.js* чи *Angular*, значно спрощують створення складних інтерфейсів. Фронтенд тісно взаємодіє з бекендом (розроблено у п.п 3.2 даної дипломної роботи): від нього надходять дані (наприклад, рейси чи користувачі), які відображаються на сторінці. Він також забезпечує зручність та доступність для користувача, адаптацію до різних екранів (через адаптивну верстку) і швидкість реакції. У веб-додатку для пошуку авіаквитків фронтенд буде реалізовано через *HTML/CSS/JS* у поєднанні з шаблонами *Jinja2*, що генеруються на сервері за допомогою *Flask*. Це дозволяє створити зручний інтерфейс для введення даних, перегляду результатів пошуку та бронювання квитків.

3.3.1. Створення таблиці стилів *style.css*

Код таблиці стилів *style.css* наведено у додатку Б.

Ця таблиця стилів *CSS* визначає зовнішній вигляд веб-сторінки з акцентом на зручність та візуальну привабливість:

- *body* — встановлює фонове зображення (*bg.jpg*. Створено автором дипломної роботи з декількох графічних елементів, що мають відкриту ліцензію),

яке не повторюється та покриває весь екран, створюючи приємний візуальний ефект;

- *.container* — центральний блок з напівпрозорим темним фоном, білим текстом, закругленими кутами та тінню. Він використовується для розміщення основного вмісту (форми, повідомлення тощо) у центрі сторінки;
- *.container a* — стилізує посилання всередині контейнера, роблячи їх білими та прибираючи підкреслення. При наведенні — підкреслення з'являється;
- *input, button* — поля введення та кнопки мають однаковий стиль: відступи, ширина, закруглення та відсутність рамок;
- *button* — має яскравий жовтий фон, жирний шрифт та курсор "*pointer*" при наведенні, що вказує на можливість натискання;
- *table* — таблиця має білий напівпрозорий фон, темний текст і виглядає охайно завдяки межам і відступам;
- *th, td* — комірки таблиці мають внутрішні відступи та нижню рамку для кращої читабельності;
- *.footer* — напис, зафіксований у нижньому лівому куті екрану, стилізований білим кольором і меншим розміром шрифту.

3.3.2. Створення гіпертекстової розмітки головної сторінки *index.html*

Код *HTML* реалізації *index.html* наведено у додатку В:

Шаблон *index.html* є головною сторінкою веб-додатку для пошуку авіаквитків. Його структура та функціональність побудовані з використанням *HTML* і *Jinja2* (шаблонізатор *Flask*). Опис основних частин:

- <*head*>: містить метаінформацію та підключення *CSS*-стилів з файлу style.css, який розташований у папці *static*;
- <*div class="container"*>: основний візуальний блок з напівпрозорим фоном (стилізований у *CSS*), який розміщується по центру сторінки;
- заголовок *h1*: назва системи — "Система пошуку авіаквитків";
- авторизаційна логіка: Якщо користувач увійшов у систему (*session.get('user_id')*), відображається привітання з іменем користувача та

посилання на вихід. Якщо не увійшов, показуються посилання на сторінки реєстрації та входу;

- форма пошуку рейсів: Містить поля для введення міста відправлення, міста призначення та дати. Після натискання кнопки "Знайти рейси", форма надсилає дані методом *POST* на маршрут /*search*;
- відображення знайдених рейсів: Якщо перелік рейсів передано у змінній *flights*, відображається список результатів. Для кожного рейсу виводяться: маршрут, дата, ціна, а також кнопка "Забронювати" — якщо користувач авторизований4
- підпис: наприкінці сторінки зазначено автора проєкту — "*Developed by*: *Sigaev Ruslan*. *321*. *2025*".

Цей шаблон забезпечує інтерактивність між користувачем і системою бронювання авіаквитків, надаючи базові можливості для пошуку та замовлення рейсів.

3.3.3. Створення гіпертекстової розмітки результату пошуку *results.html*

Код *HTML* реалізації *results.html* наведено у додатку Г:

Цей *HTML*-шаблон відображає сторінку результатів пошуку авіарейсів у веб-додатку. Він побудований з використанням *HTML* і *Jinja2*-шаблонізатора (для *Flask*), і виконує такі функції:

- <*head*>:
  1. Встановлює кодування *UTF-8*.
  2. Вказує заголовок сторінки "Результати пошуку".
  3. Підключає зовнішній *CSS*-файл для оформлення інтерфейсу (*style.css* з папки *static*);
- <*div class="container*">:
  1. Основна стилізована область, в якій відображаються знайдені рейси.
  2. Має заголовок Знайдені рейси;
- таблиця результатів:
  1. Виводиться тільки якщо є дані *({% if flights %})*.

2. Таблиця містить такі стовпці: номер рейсу, звідки, куди, дата, час, ціна, дія.

3. Для кожного рейсу створюється рядок із відповідними даними.

4. У стовпці "Дія" є посилання "Забронювати", яке веде на маршрут бронювання, передаючи *flight_id*;

- якщо рейсів не знайдено, виводиться повідомлення: Рейсів не знайдено;
- посилання "Повернутися на головну" дає змогу перейти назад на головну сторінку пошуку.

Цей шаблон є важливою частиною користувацького інтерфейсу, оскільки надає зрозумілий і зручний спосіб перегляду доступних рейсів і переходу до бронювання.

3.3.4. Створення гіпертекстової розмітки реєстрації користувачів *register.html*

Код *HTML* реалізації *register.html* наведено нижче у додатку Д:

Цей *HTML*-шаблон реалізує сторінку реєстрації користувача у веб-додатку, створеному на *Flask*. Його структура наступна:

- мета-дані та оформлення (<*head*>):
  1. Задається кодування сторінки *UTF-8*.
  2. Встановлюється заголовок сторінки — "Реєстрація користувача".
  3. Підключається таблиця стилів *style.css* для візуального оформлення через *Jinja*-функцію *url_for*;
- основний блок (<*div class="container"*>):
  1. Виводиться заголовок Реєстрація.
  2. Знаходиться форма, яка надсилається методом *POST* на ту саму адресу.
  3. Форма містить два поля введення: ім’я користувача (*username*) та пароль (*password*)

4. Обидва поля мають атрибут *required*, що забезпечує перевірку на заповнення;

- кнопка відправлення форми:

1. Кнопка «Зареєструватися» ініціює обробку даних форми сервером.

2. Посилання для переходу на сторінку входу:

3. Виводиться повідомлення «Вже маєте акаунт?» з посиланням на сторінку входу (*login*), що полегшує навігацію для вже зареєстрованих користувачів.

Цей шаблон забезпечує базову функціональність інтерфейсу реєстрації, є простим і зручним для користувача, та легко інтегрується з серверною логікою *Flask*-додатку.

3.3.5. Створення гіпертекстової розмітки авторизації користувача *login.html*

Код *HTML* реалізації *login.html* наведено у додатку Е.

Цей *HTML*-файл реалізує сторінку входу користувача до веб-додатку, створеного на *Flask*. Його основне призначення — надати користувачу інтерфейс для введення облікових даних з метою авторизації.

Основні елементи:

- *<head>*:

1. Задається кодування *UTF-8*.

2. Назва сторінки — «Вхід».

3. Підключається зовнішній файл стилів *style.css* для оформлення сторінки через *Jinja*-функцію *url_for*;

- * *:

1. Центральний контейнер оформлено зі стилями для приємного відображення.

2. Виводиться заголовок «Вхід до системи»;

- форма входу (*<form method="post">*):

1. Використовується метод *POST* для надсилання даних на сервер. Є два обов’язкові поля введення: Ім’я користувача (*username*) та Пароль (*password*).

2. Кнопка «Увійти» надсилає введені дані на сервер для перевірки;

- посилання на реєстрацію:

1. Якщо користувач ще не має акаунта, йому пропонується перейти за посиланням на сторінку реєстрації (*register*).

Ця сторінка є важливою частиною системи автентифікації і дозволяє захистити функціонал додатку, доступний лише авторизованим користувачам.

3.3.6. Створення гіпертекстової розмітки результату бронювання *booking_success.html*

Код *HTML* реалізації *booking_success.html* наведено у додатку Є.

Цей HTML-файл реалізує сторінку підтвердження успішного бронювання квитка в системі пошуку авіарейсів.

Основні елементи:

- <*head*>:

1. Встановлюється кодування *UTF-8*.

2. Назва сторінки — "Бронювання успішне".

3. Підключається таблиця стилів *style.css* за допомогою *Flask*-функції *url_for*;

- <*body*>:

1. Виводиться заголовок "Ви успішно забронювали квиток!" — повідомлення про успішне виконання дії.

2. За допомогою змінної *flight*, яка передається з *Flask*, відображається інформація про обраний рейс: номер рейсу, маршрут (місто відправлення - місто призначення), дата, час і ціна квитка.

3. Нижче розміщено посилання для повернення на головну сторінку (*index*).

Ця сторінка є завершальним етапом процесу бронювання, надаючи користувачеві підтвердження та деталі обраного авіарейсу. Вона сприяє зручності використання системи та інформує про успішне завершення операції.

3.4 Тестування *WEB*-додаток пошуку та бронювання авіаквитків

3.4.1. Мета тестування

Метою тестування є перевірка коректної роботи основного функціоналу вебдодатку, а саме: реєстрації та авторизації користувачів, пошуку доступних рейсів, бронювання авіаквитків, збереження інформації в базі даних та відображення відповідних повідомлень. Тестування дозволяє переконатися у відсутності критичних помилок, забезпеченні зручності інтерфейсу та правильному відображенні результатів для користувача.

3.4.2. Середовище для тестування

Для тестування використовувалося середовище *Visual Studio 2022* [9] з встановленим *Python*-інтерпретатором та бібліотекою *Flask*. Усі *HTML*-файли розташовано в директорії *templates*, а стилі — у папці *static*. База даних *flights.db* створюється та заповнюється тестовими рейсами за допомогою окремого *Python*-модуля.

Також для перевірки функціональності застосовувався браузер *Google Chrome*. Сервер запускали командою *flask run*, після чого відбувався доступ до додатку через адресу *http://127.0.0.1:5000*.

3.4.3. Початкова ініціалізація бази даних

Перед тестуванням проводиться ініціалізація бази даних за допомогою окремого *Python*-скрипту (див. рисунок 3.16), що створює таблицю *flights* і заповнює її тестовими даними. Це дозволяє виконати перевірку функціоналу у симульованих умовах.

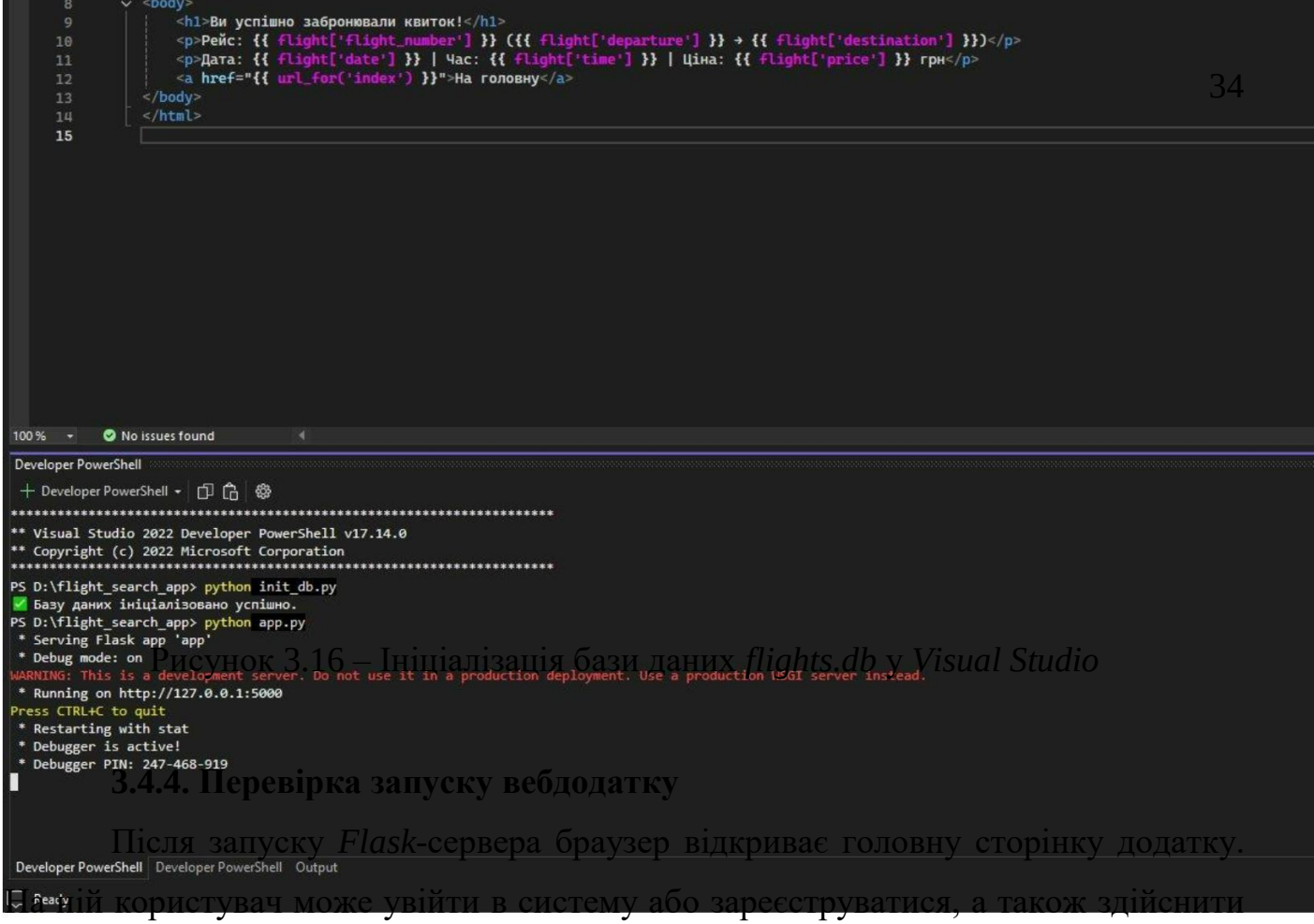


Рисунок 3.16 – Ініціалізація бази даних *flights.db* у *Visual Studio*

3.4.4. Перевірка запуску вебдодатку

Після запуску *Flask*-сервера браузер відкриває головну сторінку додатку. Гість може увійти в систему або зареєструватися, а також здійснити пошук рейсів (рисунок 3.17).

Рисунок 3.17 – Головна сторінка вебдодатку до входу в систему

3.4.5. Тестування реєстрації користувача

На сторінці *register.html* перевірено функціональність створення нового облікового запису. Заповнюємо ім’я користувача та пароль, після чого натискаємо

кнопку «Зареєструватися» (рисунок 3.18). Після успішної реєстрації користувач автоматично входить у систему, що підтверджується вітальним повідомленням на головній сторінці.

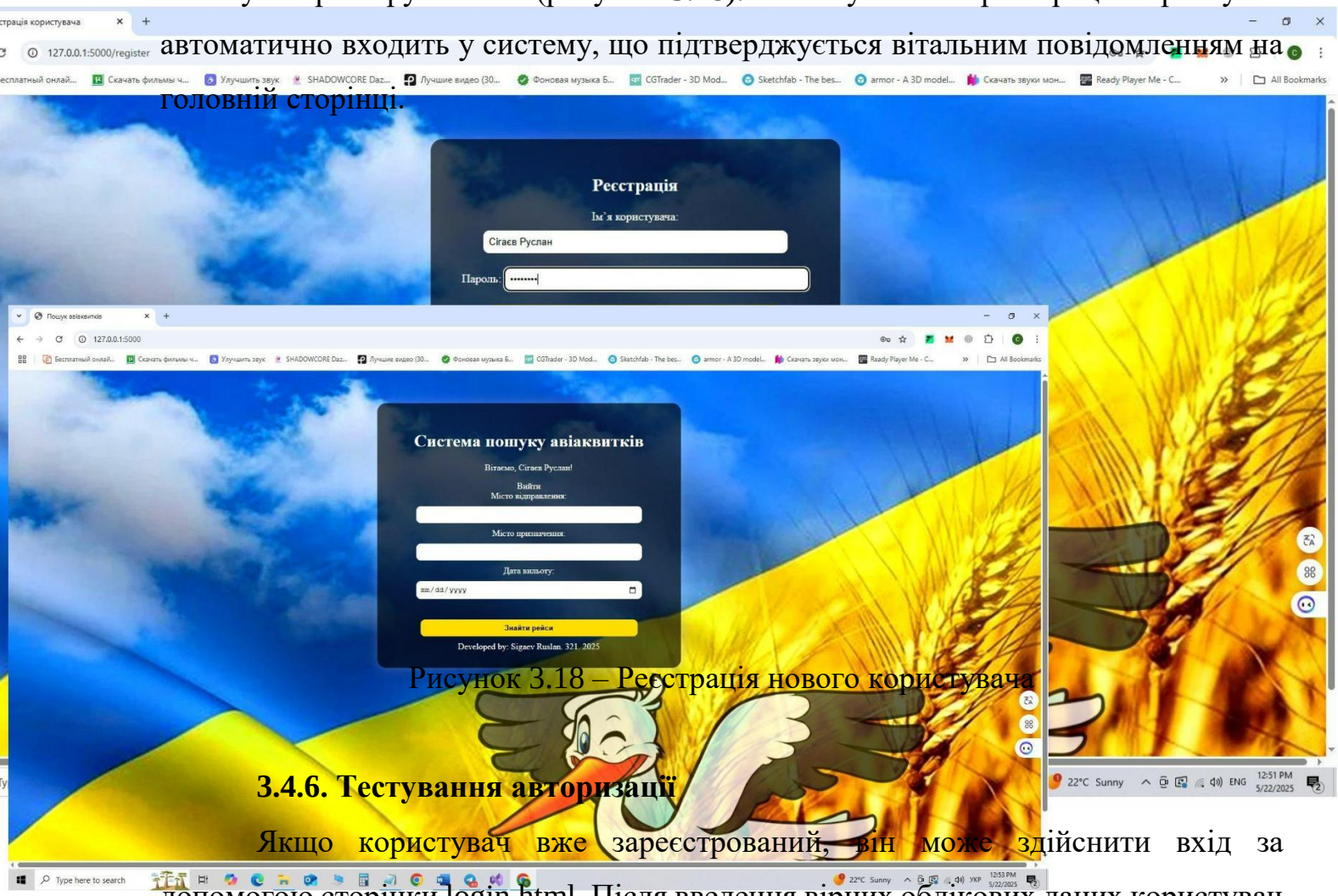


Рисунок 3.18 – Реєстрація нового користувача

3.4.6. Тестування авторизації

Якщо користувач вже зареєстрований, він може здійснити вхід за допомогою сторінки login.html. Після введення вірних облікових даних користувач потрапляє на головну сторінку (рисунок 3.19). У разі введення неправильного пароля система повідомляє про помилку.

Рисунок 3.19 – Вхід користувача в систему

3.4.7. Тестування пошуку рейсів

Наступним етапом тестування є перевірка форми пошуку рейсів. Користувач вводить місто відправлення, місто призначення та бажану дату (рисунок 3.20). Після натискання кнопки «Знайти рейси» сервер обробляє запит, і користувачу відображаються знайдені рейси, які відповідають критеріям пошуку (рисунок 3.21).

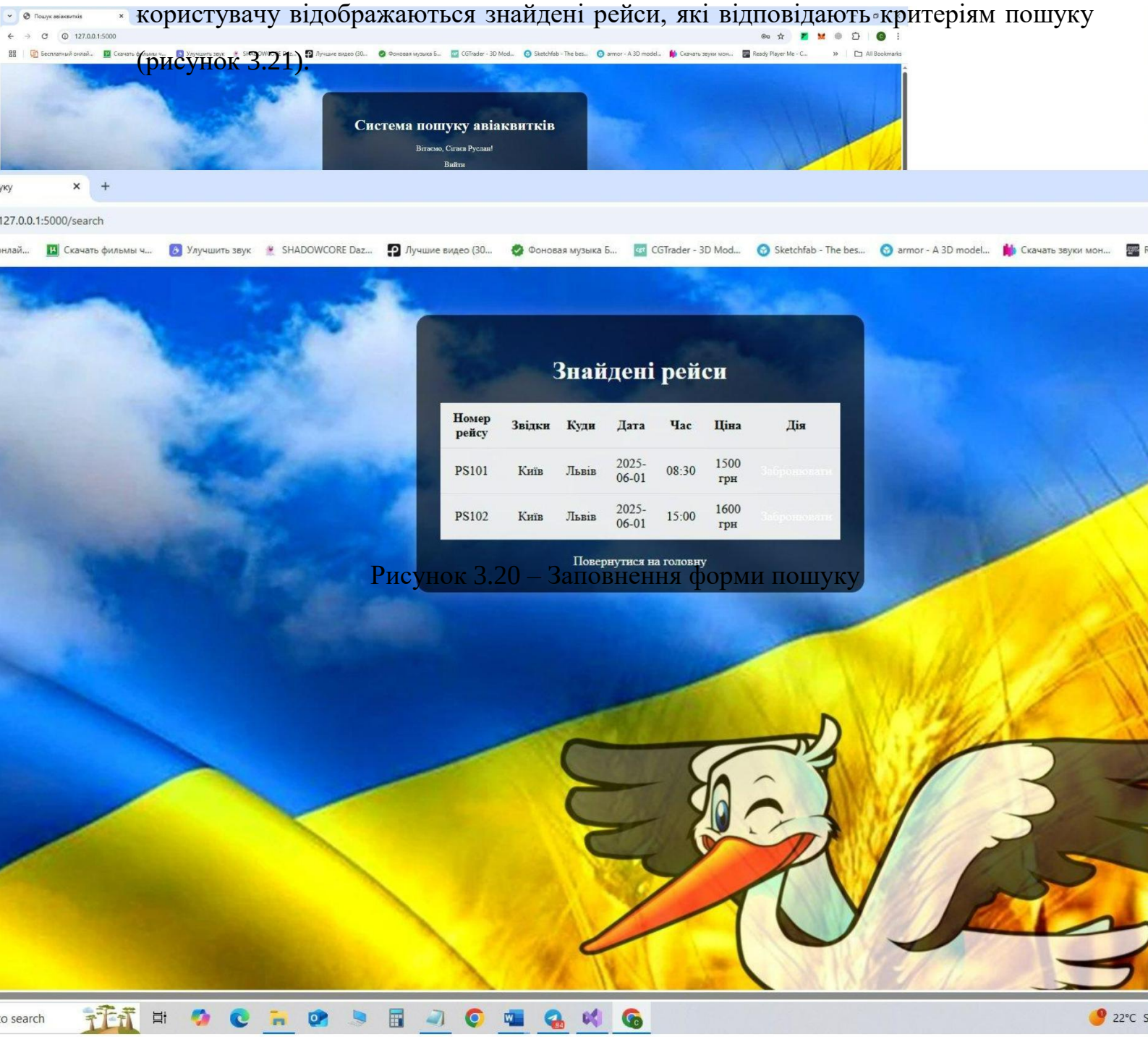


Рисунок 3.20 – Заповнення форми пошуку

Рисунок 3.21 – Результати пошуку доступних рейсів

3.4.8. Перевірка бронювання квитка

Кожен знайдений рейс має кнопку «Забронювати». При натисканні на неї відбувається передача ідентифікатора рейсу до функції *book()*, яка перевіряє, чи вже існує бронювання для цього користувача. У разі успішного бронювання з’являється сторінка *booking_success.html* з підтвердженням (рисунок 3.22).

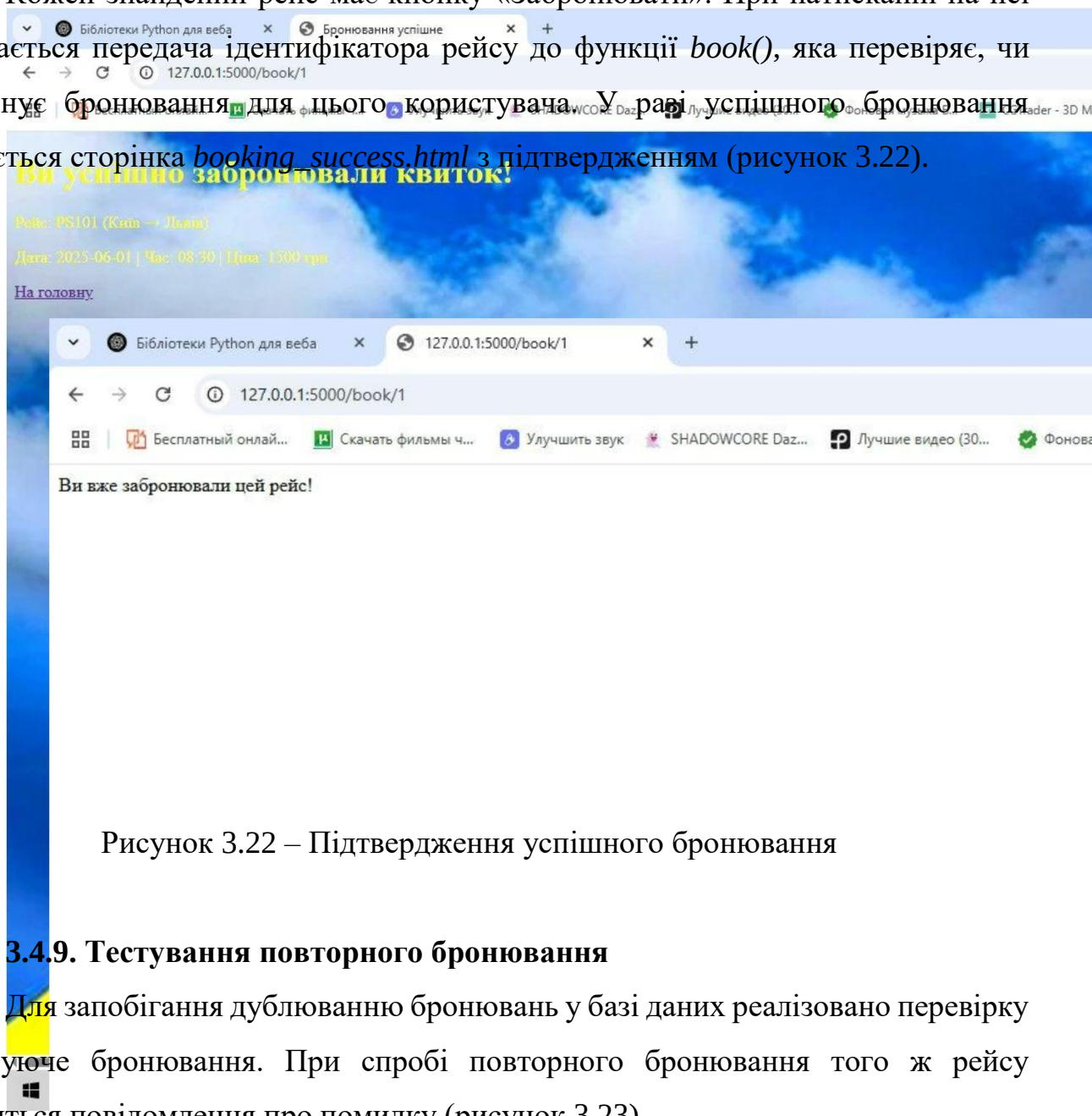


Рисунок 3.22 – Підтвердження успішного бронювання

3.4.9. Тестування повторного бронювання

Для запобігання дублюванню бронювань у базі даних реалізовано перевірку на існуюче бронювання. При спробі повторного бронювання того ж рейсу виводиться повідомлення про помилку (рисунок 3.23).

Рисунок 3.23 – Повідомлення про існуюче бронювання

3.4.10. Перевірка стилів і дизайну

Було проведено візуальну перевірку стилів *CSS*-файлу. Основний контейнер має напівпрозорий фон, білі елементи тексту, заокруглені поля введення та кнопки. Фонова картинка також успішно підтягується через *background-image*, що підтверджено в інструментах розробника (рисунок 3.24).

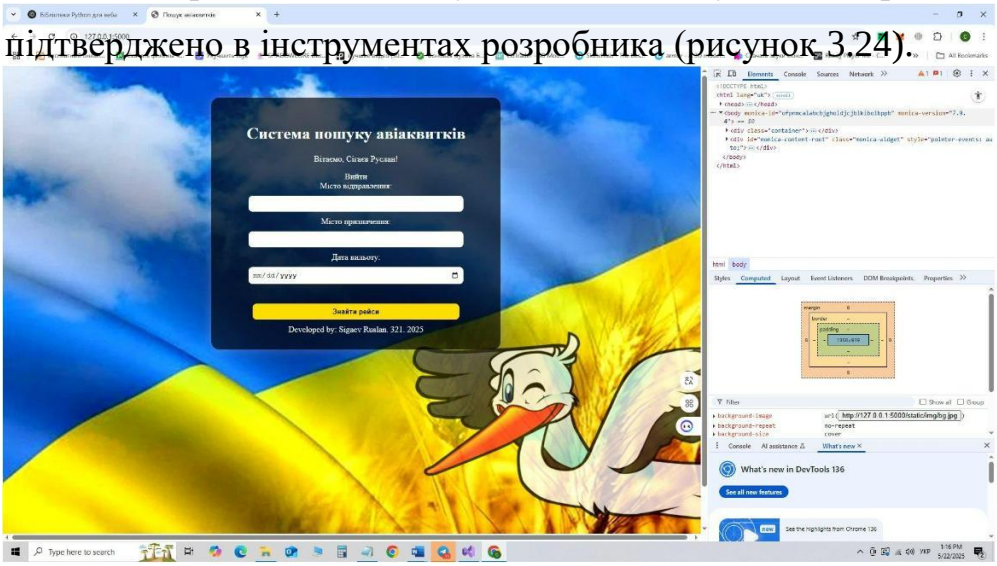


Рисунок 3.24 – Відображення стилів форми в браузері

3.4.11. Робота з базою даних

Тестування передбачало також перевірку змін у базі даних. Після бронювання квитка вручну переглядалася таблиця *bookings* у файлі *flights.db* за допомогою *SQLite*-редактора *DB Browse* (*QLite*) [10]. Було підтверджено, що кожне бронювання додається з правильним *user_id* і *flight_id.*

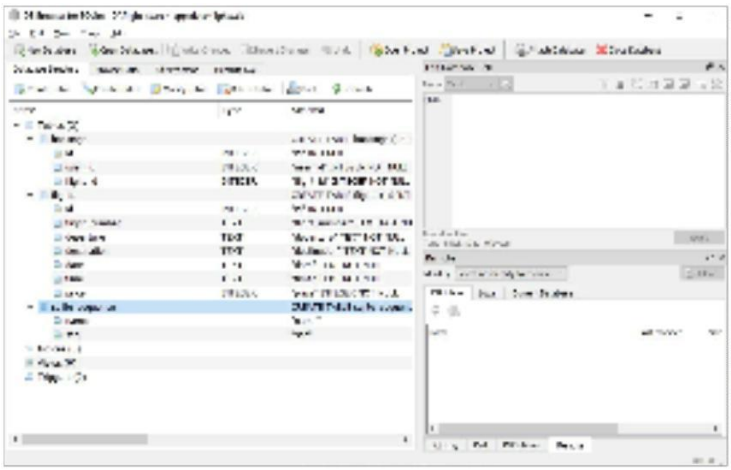


Рисунок 3.25 – Таблиця *bookings* після бронювання

3.4.12. Тестування виходу з системи

Кнопка «Вийти» коректно завершує сесію, видаляючи *user_id* із *session*. Після виходу головна сторінка знову показує посилання на реєстрацію та вхід (рисунок 3.26).

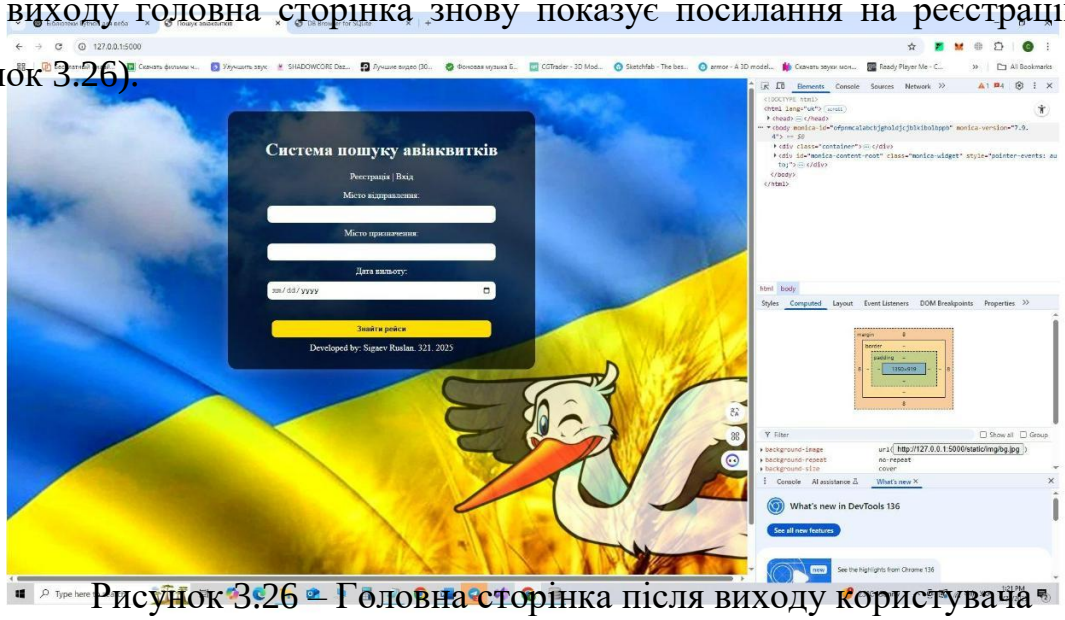


Рисунок 3.26 – Головна сторінка після виходу користувача

3.4.13. Обробка неіснуючих рейсів

Було протестовано обробку випадку, коли користувач намагається перейти на сторінку бронювання неіснуючого рейсу (вручну змінено *flight_id* у адресному рядку). У цьому випадку система повертає повідомлення «Рейс не знайдено» з відповідним *HTTP*-статусом 404.

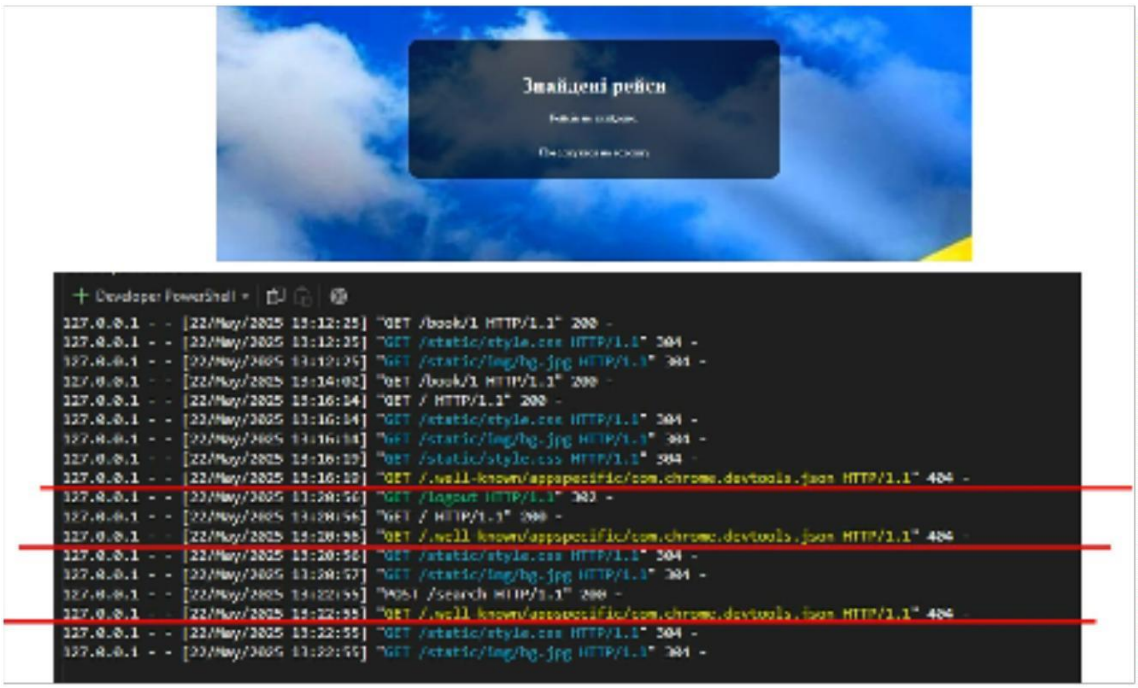


Рисунок 3.27 – Повідомлення про відсутність рейсу

3.4.14. Результати тестування

У результаті ручного тестування всі основні функції вебдодатку було перевірено функції представлені у таблиці 3.1.

Проведене тестування підтвердило працездатність створеного вебдодатку у тестовому середовищі. Усі основні сценарії використання були перевірені вручну, і не було виявлено критичних помилок. Вебінтерфейс є інтуїтивно зрозумілим, а реалізовані перевірки забезпечують стабільність роботи додатку. За потреби функціонал можна доповнити автоматичними тестами на основі бібліотек *Flask-Testing* або *pytest* у майбутньому. Також, передбачається, що додаток може працювати з реальними даними українських авіакомпаній з використанням їх API.

Таблиця 3.1 – Чек-лист перевірки Веб-додатку пошуку та бронювання авіаквитків

Функція	**Результат**
Реєстрація	Працює
Авторизація	Працює
Пошук рейсів	Працює
Вивід рейсів	Працює
Бронювання	Працює
Запобігання повторному бронюванню	Працює
Стилі та дизайн	Відображаються
Вихід із системи	Працює
Обробка помилок	Працює

3.5 Висновки до третього розділу

У результаті розробки було створено повнофункціональний веб-додаток, який дозволяє користувачам здійснювати пошук авіарейсів та оформлювати бронювання в зручному інтерфейсі. Система реалізована з використанням

фреймворку *Flask,* що забезпечило простоту в архітектурі додатку, швидкий розвиток функціоналу та ефективну інтеграцію з базою даних *SQLite*.

Було реалізовано основні функції:

- реєстрація та авторизація користувачів із захистом даних за допомогою сесій;
- пошук рейсів за вказаними параметрами (місто відправлення, прибуття, дата);
- можливість перегляду деталей рейсу та здійснення бронювання;
- перевірка наявності повторного бронювання одним користувачем;
- вивід повідомлення про успішне бронювання із деталями рейсу.

Окрему увагу приділено структурі бази даних, яка включає таблиці для зберігання рейсів та бронювань. Було створено *SQL*-скрипти для ініціалізації таблиць і наповнення їх тестовими даними, що дозволяє швидко відновлювати БД для тестування та демонстрацій.

Під час реалізації було також проведено ручне тестування інтерфейсу, маршрутизації, роботи з формами та базою даних. Виявлені помилки (наприклад, відсутність таблиці бронювань або некоректна обробка *ID* рейсу) були своєчасно виправлені. Внесено стилістичні покращення інтерфейсу для зручності користувача (наприклад, додано кольорове оформлення повідомлень).

Загалом, розроблений додаток демонструє принципи побудови клієнт-серверної архітектури, взаємодії з базами даних, обробки запитів і відображення даних у браузері. У майбутньому можливе розширення функціональності — додавання кабінету користувача, історії бронювань, оплати онлайн, фільтрів для пошуку тощо. Це відкриває перспективи використання додатку як прототипу реальної інформаційно-сервісної системи в авіаційній галузі.

ВИСНОВКИ

У кваліфікаційній роботі було здійснено повноцінне дослідження, проєктування та реалізацію веб-додатку для онлайн-бронювання авіаквитків, що є надзвичайно актуальним в умовах стрімкого розвитку електронної комерції та цифровізації сервісів у сфері пасажирських авіаперевезень. Система бронювання квитків є важливою складовою сучасної авіаційної інфраструктури, оскільки забезпечує зручний доступ користувачів до пошуку та оформлення подорожей через Інтернет, спрощує взаємодію між клієнтом і перевізником та підвищує загальний рівень сервісу.

У першому розділі роботи було проаналізовано історичні та сучасні аспекти розвитку механізмів онлайн-бронювання квитків, зокрема в авіаційній галузі. Розглянуто роль електронної комерції як фундаменту цифрової торгівлі та обґрунтовано, чому онлайн-сервіси, включаючи бронювання квитків, стали стандартом у сфері пасажирських перевезень. Було досліджено структуру систем бронювання — від комп'ютеризованих систем резервування (*CRS*) до інтеграції з глобальними системами дистрибуції (*GDS*). Це дало змогу краще зрозуміти принципи роботи сучасних веб-сервісів авіабронювання, включаючи взаємодію з базами даних, механізмами оплати та сервісами реєстрації.

У другому розділі обґрунтовано вибір технологій для реалізації веб-додатку. Основною мовою програмування було обрано *Python,* завдяки її простоті, читабельності та багатству бібліотек. Фреймворк *Flask* забезпечив гнучку та зручну серверну архітектуру, а *Jinja* дозволив швидко створювати шаблони *HTML*-сторінок. Для зберігання даних обрано *SQLite* — компактну реляційну базу даних, що добре підходить для локальних і демонстраційних проєктів. Також для клієнтської частини використано *HTML, CSS* та *JavaScript,* що дозволило створити простий, але функціональний користувацький інтерфейс.

У третьому розділі розглянуто процес реалізації системи. Було створено веб-додаток, що дозволяє реєструватися, авторизовуватися, здійснювати пошук авіарейсів, переглядати деталі рейсів, оформлювати бронювання та отримувати

підтвердження. Особливу увагу приділено реалізації бази даних — створено таблиці для зберігання інформації про рейси та бронювання, а також *SQL*-скрипти для їх ініціалізації. Проведено ручне тестування всіх функцій, і виявлені помилки були своєчасно виправлені. Зокрема, було доопрацьовано обробку *ID* рейсів, захист від дублювання бронювань та покращено інтерфейс повідомлень для користувача.

Загалом, створений застосунок є ефективним прикладом реалізації веб-системи з архітектурою клієнт-сервер, що взаємодіє з базою даних та забезпечує користувача основними функціями онлайн-сервісу. Отримані результати засвідчують відповідність поставленим цілям і завданням, а також демонструють практичне застосування отриманих знань у сфері веб-програмування, баз даних і систем електронної комерції.

У перспективі система може бути розширена шляхом додавання функціоналу онлайн-оплати, історії бронювань, персонального кабінету користувача, застосування фільтрів пошуку та інших зручностей. Це відкриває можливість інтеграції проєкту в реальні бізнес-процеси авіакомпаній або туристичних сервісів.

Таким чином, дана дипломна робота не лише підтверджує рівень професійної підготовки розробника, але й слугує базисом для подальших досліджень і вдосконалення систем веб-сервісів у сфері пасажирських авіаперевезень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Інтернет-посилання]. Метод доступу: https://www.travelmole.com/news/ (дата звернення: 14.04.2025)
2. *Rouse, M. E-commerce (electronic commerce or EC) definition, Available:* [Інтернет-посилання]. Метод доступу: *http://searchcio.techtarget.com/definition/e-commerce/.* (дата звернення: 14.04.2025).
3. *Atkinson, B. How does online check in work?* [Інтернет-посилання]. Метод доступу: *http://www.travelsupermarket.com/blog/how-does-online-check-in-work/.* (дата звернення: 14.04.2025).
4. *Winston, C., Morrison, S. The Evolution of the Airline Industry. Brookings Institution Press., Washington, DC : Brookings Institution, 1995.*
5. [Інтернет-посилання]. Метод доступу: *https://www.python.org/* (дата звернення: 14.04.2025).
6. [Інтернет-посилання]. Метод доступу: https://vseosvita.ua/lesson/mova-hipertekstovoi-rozmitky-262003.html (дата звернення: 14.04.2025).
7. [Інтернет-посилання]. Метод доступу: https://jinja.palletsprojects.com (дата звернення: 14.04.2025).
8. [Інтернет-посилання]. Метод доступу: https://developer.mozilla.org (дата звернення: 14.04.2025).
9. [Інтернет-посилання]. Метод доступу: https://visualstudio.microsoft.com (дата звернення: 14.04.2025).
10. [Інтернет-посилання]. Метод доступу: https://sqlitebrowser.org/ (дата звернення: 14.04.2025).
11. [Інтернет-посилання]. Метод доступу: https://flask.palletsprojects.com (дата звернення: 14.04.2025).

ДОДАТОК А

Код модуля завершення

```
import sqlite3
import os
# Створити папку, якщо не існує
os.makedirs('data', exist_ok=True)
# Підключення до БД
conn = sqlite3.connect("data/flights.db")
cursor = conn.cursor()
# Видалити таблицю, якщо існувала раніше (для повторного запуску)
cursor.execute('DROP TABLE IF EXISTS flights')
# Створити таблицю з полем id
cursor.execute('''
    CREATE TABLE flights (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        flight_number TEXT NOT NULL,
        departure TEXT NOT NULL,
        destination TEXT NOT NULL,
        date TEXT NOT NULL,
        time TEXT NOT NULL,
        price INTEGER NOT NULL
    )
''')
# Додати тестові дані
cursor.executemany('''
   INSERT INTO flights (flight_number, departure, destination, date, time, price)
    VALUES (?, ?, ?, ?, ?, ?)
''', [
    ('PS101', 'Київ', 'Львів', '2025-06-01', '08:30', 1500),
```

```
    ('PS102', 'Київ', 'Львів', '2025-06-01', '15:00', 1600),
    ('PS201', 'Одеса', 'Харків', '2025-06-01', '12:00', 1800),
    ('PS301', 'Львів', 'Київ', '2025-06-01', '09:45', 1400)
])
# Завершення
conn.commit()
conn.close()
print(" Базу даних ініціалізовано успішно.")
```

ДОДАТОК Б

Код таблиці стилів style.css

```
body {
    width: 100%;
    height: 100vh;
    background-image: url('../static/img/bg.jpg');
    background-repeat: no-repeat;
    background-size: cover;
}
.container {
    background: rgba(0, 0, 0, 0.65);
    max-width: 500px;
    margin: 60px auto;
    padding: 30px;
    border-radius: 16px;
    box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);
    text-align: center;
    color: white; /* робимо весь текст білим */
}
    /* Стиль для посилань всередині контейнера */
    .container a {
        color: white;
        text-decoration: none;
    }
        .container a:hover {
            text-decoration: underline;
        }
input, button {
    margin: 10px 0;
```

```
    padding: 8px;
    width: 80%;
    border-radius: 8px;
    border: none;
}
button {
    background-color: #ffdd00;
    font-weight: bold;
    cursor: pointer;
}
table {
    width: 100%;
    background-color: rgba(255, 255, 255, 0.9);
    color: #000;
    border-collapse: collapse;
    margin-top: 20px;
}
th, td {
    padding: 10px;
    border-bottom: 1px solid #ddd;
}
/* Напис внизу */
.footer {
    position: fixed;
    bottom: 10px;
    left: 10px;
    color: white;
    font-size: 14px;
}
```

ДОДАТОК В

Код *HTML* реалізації *index.html*

```
<!DOCTYPE html>
<html lang=»uk»>
<head>
   <meta charset=»UTF-8»>
   <title>Пошук авіаквитків</title>
   <link rel=»stylesheet» href=»{{ url_for(‘static’, filename=’style.css’) }}»>
</head>
<body>
   <div class=»container»>
      <h1>Система пошуку авіаквитків</h1>
      {% if session.get(‘user_id’) %}
      <p>Вітаємо, {{ session.get(‘username’) }}!</p>
      <a href=»{{ url_for(‘logout’) }}»>Вийти</a>
      {% else %}
      <p>
         <a href=»{{ url_for(‘register’) }}»>Реєстрація</a> |
         <a href=»{{ url_for(‘login’) }}»>Вхід</a>
      </p>
      {% endif %}
      <form action=»{{ url_for(‘search’) }}» method=»post»>
         <label for=»departure»>Місто відправлення:</label><br>
         <input type=»text» name=»departure» required><br>

         <label for=»destination»>Місто призначення:</label><br>
         <input type=»text» name=»destination» required><br>
         <label for=»date»>Дата вильоту:</label><br>
         <input type=»date» name=»date» required><br><br>
```

```
            <button type=»submit»>Знайти рейси</button>
        </form>
        {% if flights %}
        <h2>Знайдені рейси:</h2>
        <ul>
          {% for flight in flights %}
          <li>
            {{ flight[1] }} → {{ flight[2] }}, {{ flight[3] }} —
            {{ flight[4] }} грн
            {% if session.get('user_id') %}
          <form action=»{{ url_for('book', flight_id=flight[0]) }}» method=»post»
style=»display:inline;»>
              <button type=»submit»>Забронювати</button>
            </form>
            {% endif %}
          </li>
          {% endfor %}
        </ul>
        {% endif %}
        Developed by: Sigaev Ruslan. 321. 2025
      </div>
    </body>
    </html>
```

ДОДАТОК Г

Код *HTML* реалізації *results.html*

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Результати пошуку</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>Знайдені рейси</h1>
        {% if flights %}
        <table border="0">
            <tr>
                <th>Номер рейсу</th>
                <th>Звідки</th>
                <th>Куди</th>
                <th>Дата</th>
                <th>Час</th>
                <th>Ціна</th>
                <th>Дія</th>
            </tr>
            {% for flight in flights %}
            <tr>
                <td>{{ flight[1] }}</td>
                <td>{{ flight[2] }}</td>
                <td>{{ flight[3] }}</td>
```

```
                <td>{{ flight[4] }}</td>
                <td>{{ flight[5] }}</td>
                <td>{{ flight[6] }} грн</td>
                <td>
                    <a href="{{ url_for('bookings.book', flight_id=flight[0]) }}">Забронювати</a>
                </td>
            </tr>
            {% endfor %}
        </table>
        {% else %}
        <p>Рейсів не знайдено.</p>
        {% endif %}
        <br>
        <a href="{{ url_for('index') }}">Повернутися на головну</a>
    </body>
    </html>
```

ДОДАТОК Д

Код *HTML* реалізації *register.html*

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Реєстрація користувача</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h2>Реєстрація</h2>
        <form method="post">
            <label for="username">Ім’я користувача:</label>
            <input type="text" name="username" required><br>

            <label for="password">Пароль:</label>
            <input type="password" name="password" required><br>

            <button type="submit">Зареєструватися</button>
        </form>
        <p>Вже маєте акаунт? <a href="{{ url_for('login') }}">Увійти</a></p>
    </div>
</body>
</html>
```

ДОДАТОК Е

Код *HTML* реалізації *login.html*

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Вхід</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h2>Вхід до системи</h2>
        <form method="post">
            <label for="username">Ім’я користувача:</label>
            <input type="text" name="username" required><br>
            <label for="password">Пароль:</label>
            <input type="password" name="password" required><br>
            <button type="submit">Увійти</button>
        </form>
        <p>Ще не зареєстровані? <a href="{{ url_for('register') }}">Реєстрація</a></p>
    </div>
</body>
</html>
```

ДОДАТОК Є

Код *HTML* реалізації *booking_success.html*

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Бронювання успішне</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Ви успішно забронювали квиток!</h1>
    <p>Рейс: {{ flight['flight_number'] }} ({{ flight['departure'] }} → {{ flight['destination'] }})</p>
    <p>Дата: {{ flight['date'] }} | Час: {{ flight['time'] }} | Ціна: {{ flight['price'] }} грн</p>
    <a href="{{ url_for('index') }}">На головну</a>
</body>
</html>
```

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ
на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»
відділення: комп'ютерної та програмної інженерії
циклова комісія: комп'ютерних систем та мереж
Руслана СІГАЄВА
(ім'я, прізвище)

У сучасних умовах цифрової трансформації транспортної галузі розробка веб-сервісів для онлайн-бронювання авіаквитків є надзвичайно затребуваною. Такий застосунок спрощує процес планування подорожей, підвищує зручність для користувачів та забезпечує ефективну взаємодію між клієнтом і перевізником. Я вважаю, що тема «Розробка веб-додатку для онлайн-пошуку та бронювання авіаквитків» є актуальною.

Виконана дипломна робота відповідає затвердженій темі і охоплює повний цикл створення веб-застосунку — від аналізу предметної області до реалізації і тестування. Структура пояснювальної записки логічна з чітким поділом на етапи розробки.

В ході виконання дипломної роботи реалізовано функціональний веб-додаток з базовими можливостями: реєстрація користувача, пошук рейсів, бронювання квитків та обробка введених даних. Система працює на основі популярного стека технологій — Python (Flask), SQLite, HTML/CSS/JS, що свідчить про доцільний вибір інструментів.

Пояснювальна записка оформлена згідно з вимогами. Описано архітектуру системи, принципи функціонування, приклади коду та інтерфейс користувача. Проведене тестування і оптимізація підтверджують відповідальне ставлення до проєкту.

Розроблений застосунок має високу практичну значущість і може слугувати основою для подальшого розвитку повнофункціонального сервісу. Передбачена можливість масштабування (онлайн-оплата, особистий кабінет, фільтри), що свідчить про гнучкість обраної архітектури.

Дипломна робота виконана на високому рівні, демонструє добру підготовку здобувача у сфері веб-програмування та розуміння принципів розробки повноцінних інформаційних систем.

Робота заслуговує на оцінку "відмінно".

Рецензент *викладач*
(науковий ступінь, посада)

«30» 05 2025 р. _____ Сергій РУДИЙ
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Руслан СІГАЄВ
(підпис) (ім'я, прізвище)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ВІДГУК
керівника кваліфікаційної роботи

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Руслан СІГАЄВ
(ім'я, прізвище)

1. Кваліфікаційна робота на тему «WEB-додаток пошуку та бронювання авіаквитків» виконана в ініціативному порядку.

2. Метою кваліфікаційної роботи є проектування сучасного та функціонального веб-додаток пошуку та бронювання авіаквитків.

3. Кваліфікаційна робота відповідає темі, затвердженій наказом начальника коледжу.

4. Кваліфікаційна робота виконана здобувачем освіти самостійно.

5. *Здобувач освіти показав високі вміння роботи з літературними джерелами, аналіз теоретичного та практичного матеріалу, приймання обґрунтованих рішень, застосовування сучасних комп'ютерних інформаційних технологій.*

6. *Руслан СІГАЄВ показав достатній рівень дотримання вимог державних стандартів при виконанні кваліфікаційної роботи в цілому та оформленні пояснювальної записки .*

7. Рівень виконаної кваліфікаційної роботи заслуговує оцінку «добре», відповідає набутих випускником знань, умінь та навичок, вимогам освітньої характеристики фахівця і можливість присвоєння йому освітньо-кваліфікаційного ступеню «Фаховий молодший бакалавр» спеціальності 123 «Комп'ютерна інженерія».

Керівник кваліфікаційної роботи

«10» 06 2025 р. _____ Андрій КРАВЧАТИЙ
(підпис) (ім'я, прізвище)