ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: « Модель веб-платформи для аналізу погоди з підтримкою технології python »

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми _____
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Микола СІРОШТАНОВ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-42_
Микола СІРОШТАНОВ
Ім'я, ПРІЗВИЩЕ

Керівник: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти Бакалавр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою
Інформаційних систем та технологій

_____ Каміла Сторчак
«_____» _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сіроштанову Миколі Дмитровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Модель веб-платформи для аналізу погоди з підтримкою технології Python
керівник кваліфікаційної роботи Андрій БОНДАРЧУК *д.т.н., професор*
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02.2025р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: параметри моделі веб-платформи для аналізу погодних умов, обробки метеорологічних даних, використання сучасних веб-технологій, робота з API.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   − Аналіз існуючих веб-платформ для прогнозу погоди та обґрунтування необхідності створення нового сервісу з персоналізованими рекомендаціями.
   − Розробка програмного продукту на основі технологій Python і Flask із використанням зовнішніх API та реалізація функціональних модулів.

− Демонстрація результатів роботи веб-платформи, оцінка її ефективності та перспектив розвитку.

5. Перелік графічного матеріалу: *презентація*

1. Схема роботи веб-платформи, що демонструє етапи взаємодії користувача з системою.
2. Архітектура веб-платформи, яка включає серверну частину на Python з використанням Flask, фронтенд на HTML/CSS/JavaScript, а також взаємодію з OpenWeather API та базою даних SQLite.
3. Модель веб-платформи з відображенням ключових компонентів: API-запити, обробка даних, система рекомендацій.
4. Приклад бази даних SQLite із рекомендаціями.

6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз джерел з теми цифрового PR та сучасних веб-технологій	21.04-25.04.25	
2	Вивчення функціональних вимог до платформи	26.04-30.04.25	
3	Проєктування архітектури системи та структури бази даних	01.05-06.05.25	
4	Розробка клієнтської частини (інтерфейс, взаємодія з API, UX/UI дизайн)	07.05-13.05.25	
5	Розробка серверної частини та REST API	14.05-18.05.25	
6	Інтеграція інтелектуального помічника та чат-функціоналу	19.05-22.05.25	
7	Тестування системи, усунення помилок та оптимізація	23.05-27.05.25	
8	Підготовка та розробка демонстраційних матеріалів	28.05-29.05.25	
8	Оформлення роботи: вступ, висновки, реферат	29.05-30.05.25	

Здобувач вищої освіти _____ Микола СІРОШТАНОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 76 стор., 72 рис., 2 табл., 25 джерела.

Мета роботи – розробка моделі веб-платформи, яка забезпечує не лише прогноз погоди, а й формує персоналізовані рекомендації щодо збереження здоров'я на основі метеорологічних даних.

Об'єкт дослідження – процес формування інформаційно-аналітичної моделі погодного сервісу.

Предмет дослідження – методика аналізу погодних даних і генерування рекомендацій з охорони здоров'я з використанням відкритих API, веб-технологій і мови програмування Python.

Короткий зміст роботи: У кваліфікаційній роботі проаналізовано сучасні погодні сервіси (OpenWeatherMap, Gismeteo, Sinoptik тощо), виявлено їхні недоліки щодо персоналізації рекомендацій. Запропоновано нову модель веб-платформи, яка інтегрує метеорологічні дані з медичними порадами. Для реалізації було використано Python, Flask, HTML, CSS, JavaScript, SQLite, а також OpenWeather API. Система дозволяє користувачу обрати місто, переглянути прогноз погоди, побачити карту з геолокацією і отримати рекомендації щодо дій у певних погодних умовах, зокрема для людей із хронічними захворюваннями, алергіків або метеозалежних. Платформа має адаптивний інтерфейс, інтерактивні елементи та базу даних порад, що формуються автоматично. Результати тестування показали високу ефективність та зручність використання веб-платформи.

КЛЮЧОВІ СЛОВА: ПОГОДА, PYTHON, FLASK, OPENWEATHER, ВЕБ-ПЛАТФОРМА, РЕКОМЕНДАЦІЇ, API, ЗДОРОВ'Я, ІНТЕРАКТИВНІСТЬ, МЕТЕОДАНІ.

ABSTRACT

Text part of the qualification work for the degree of Bachelor: 76 pages, 72 figures, 2 tables, 25 sources.

Purpose of the work – development of a web platform model that provides not only weather forecasting but also generates personalized health preservation recommendations based on meteorological data.

Object of study – the process of forming an information-analytical model of a weather service.

Subject of study – methodology for analyzing weather data and generating health protection recommendations using open APIs, web technologies, and the Python programming language.

Brief summary of the work: The qualification work analyzes modern weather services (OpenWeatherMap, Gismeteo, Sinoptik, etc.) and identifies their shortcomings regarding personalization of recommendations. A new web platform model is proposed that integrates meteorological data with medical advice. For implementation, Python, Flask, HTML, CSS, JavaScript, SQLite, and the OpenWeather API were used. The system allows users to select a city, view the weather forecast, see a map with geolocation, and receive recommendations for actions under specific weather conditions, particularly for people with chronic illnesses, allergies, or weather sensitivity. The platform features a responsive interface, interactive elements, and a database of automatically generated recommendations. Testing results demonstrated high efficiency and user-friendliness of the web platform.

KEYWORDS: WEATHER, PYTHON, FLASK, OPENWEATHER, WEB PLATFORM, RECOMMENDATIONS, API, HEALTH, INTERACTIVITY, METEOROLOGICAL DATA.

ЗМІСТ

ВСТУП

На сьогоднішній день ми маємо величезні зміни в кліматі. Сучасне суспільство все більше залежить від погодних умов у повсякденному житті. Метеорологічні фактори, такі як температура, вологість, атмосферний тиск та рівень ультрафіолетового випромінювання, мають прямий вплив на здоров'я населення, працездатність, безпеку пересування та інші сфери життєдіяльності. Особливо вразливими до змін погодних умов є діти, літні люди, особи з хронічними захворюваннями, алергіки та метеозалежні. Тому виникає потреба в більш точному аналізі погодних даних та формуванні персоналізованих порад для користувачів.

Сьогодні існує велика кількість сайтів, що надають прогнози погоди. Проте більшість із них зосереджена на візуальному представленні даних – температури, опадів, швидкості вітру тощо, та взагалі не приділяє уваги здоров'я людини. Практично відсутні рішення, які б адаптували інформацію з урахуванням індивідуальних або групових потреб користувачів. Наприклад, попередження алергіків про концентрацію пилку, поради метеозалежним при різких перепадах тиску або ж рекомендації щодо поведінки під час спекотних чи холодових хвиль.

Саме тому дана дипломна робота присвячена розробці моделі веб-платформи для аналізу погоди, яка не лише відображає метеорологічні дані, а й надає рекомендації з охорони здоров'я, орієнтовані на користувачів різних категорій. Основною особливістю цієї платформи є застосування алгоритмів аналізу погоди на основі відкритого API (наприклад, OpenWeather) та формування порад залежно від погодної ситуації, з урахуванням медичних ризиків.

Актуальність теми

У зв'язку з глобальними кліматичними змінами кількість днів із аномальною спекою або холодом постійно зростає. За даними Всесвітньої організації охорони здоров'я, екстремальні погодні умови щороку спричиняють десятки тисяч випадків передчасної смерті. Крім того, забруднення повітря та збільшення концентрації алергенів (зокрема пилку) негативно впливають на якість життя мільйонів людей. У таких умовах технологічні інструменти, які б не лише повідомляли про прогноз

погоди, а й допомагали приймати обґрунтовані рішення щодо збереження здоров’я, є надзвичайно актуальними.

Створення веб-платформи, яка аналізує погодні умови і генерує медичні поради для користувачів – це новий крок до персоналізованих цифрових сервісів, що об’єднують метеорологію, медицину та інформаційні технології.

Мета та завдання роботи

Метою цієї дипломної роботи є розробка моделі веб-платформи для аналізу погодних умов з використанням мови програмування Python, яка на основі отриманих даних формуватиме рекомендації з охорони здоров’я для різних груп користувачів.

Об’єкт і предмет дослідження

Об’єктом дослідження є процес формування інформаційно–аналітичної моделі погодного сервісу.

Предметом дослідження виступає методика аналізу погодних даних та генерування рекомендацій, що сприяють охороні здоров’я користувачів.

1 АНАЛІЗ ІСНУЮЧИХ ВЕБ-ПЛАТФОРМ ДЛЯ АНАЛІЗУ ПОГОДИ

1.1 Огляд популярних сервісів прогнозу погоди

У сучасному цифровому просторі доступно багато веб-сайтів, які пропонують користувачам детальну інформацію про погодні умови. Найбільш відомими з них є:

1) OpenWeatherMap (**https://openweathermap.org/**)

Це один з найпопулярніших сервісів, який надає потужне API для доступу до погодних даних у реальному часі, включаючи температуру, вологість, тиск, вітер, та багато іншого. Сайт підтримує інтерактивну карту, мобільні додатки та можливість вибору мови. Проте платформа не надає персоналізованих порад чи попереджень, які враховують стан здоров'я користувача або особливості його організму (алергії, метеозалежність тощо).

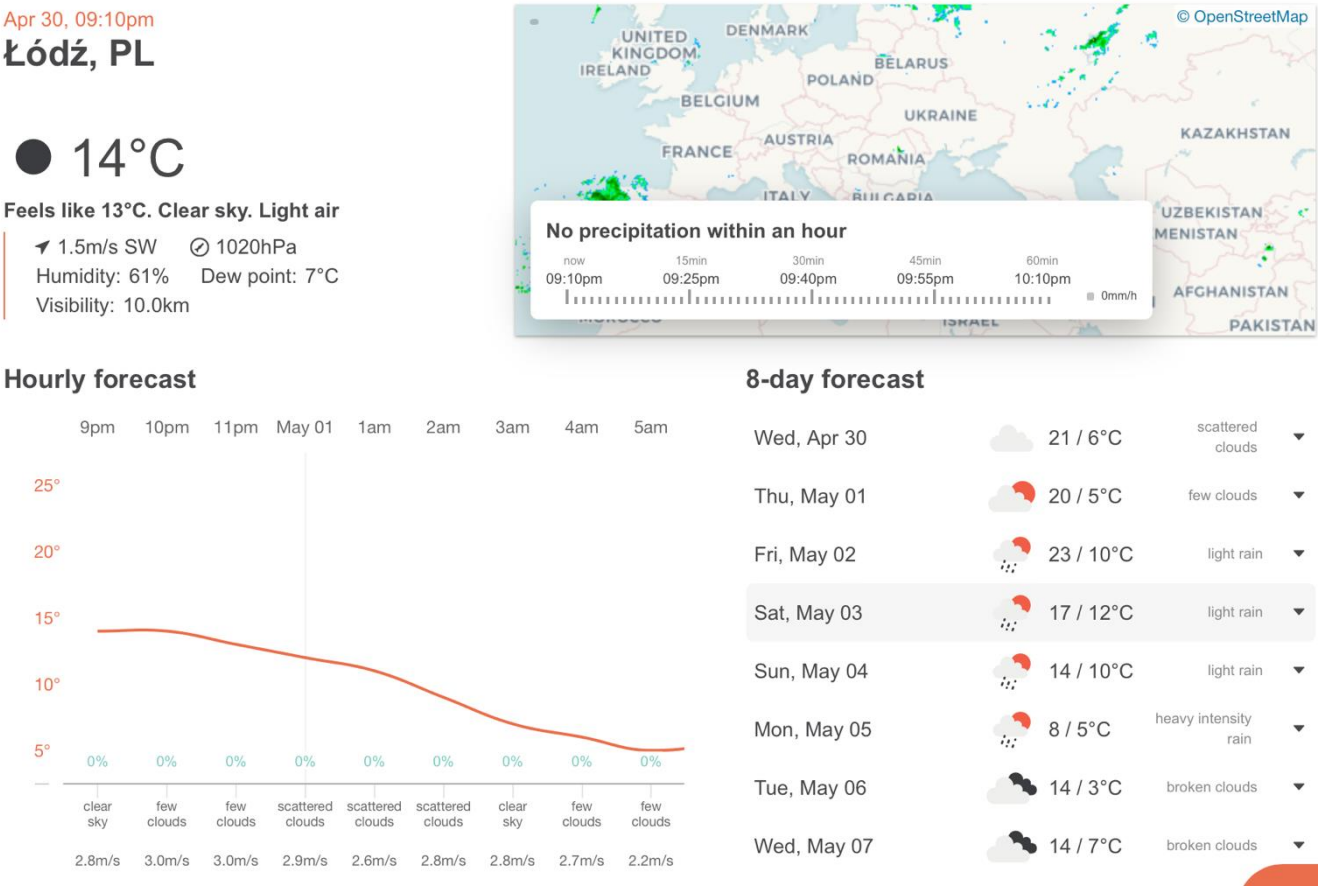


Рисунок 1.1 Погода OpenWeatherMap

2) Gismeteo (https://www.gismeteo.ua/)

Gismeteo — один із найпопулярніших погодних порталів в Україні, який

надає актуальні дані про погоду, включаючи температуру, вологість, тиск, швидкість вітру та інші метеорологічні показники. Платформа пропонує інтерактивні графіки, які дозволяють користувачам відстежувати зміни погодних умов у реальному часі. Однак, як і інші подібні сервіси, Gismeteo обмежується лише наданням базових метеорологічних даних. Користувачеві необхідно самостійно інтерпретувати ці дані, що може бути складним для людей без спеціальних знань. Крім того, на платформі відсутні медичні рекомендації або попередження щодо потенційних ризиків для здоров'я, як це буде в твоєму проєкті, що робить Gismeteo менш корисним для користувачів, які шукають індивідуальні поради, враховуючи їхній стан здоров'я чи метеозалежність.

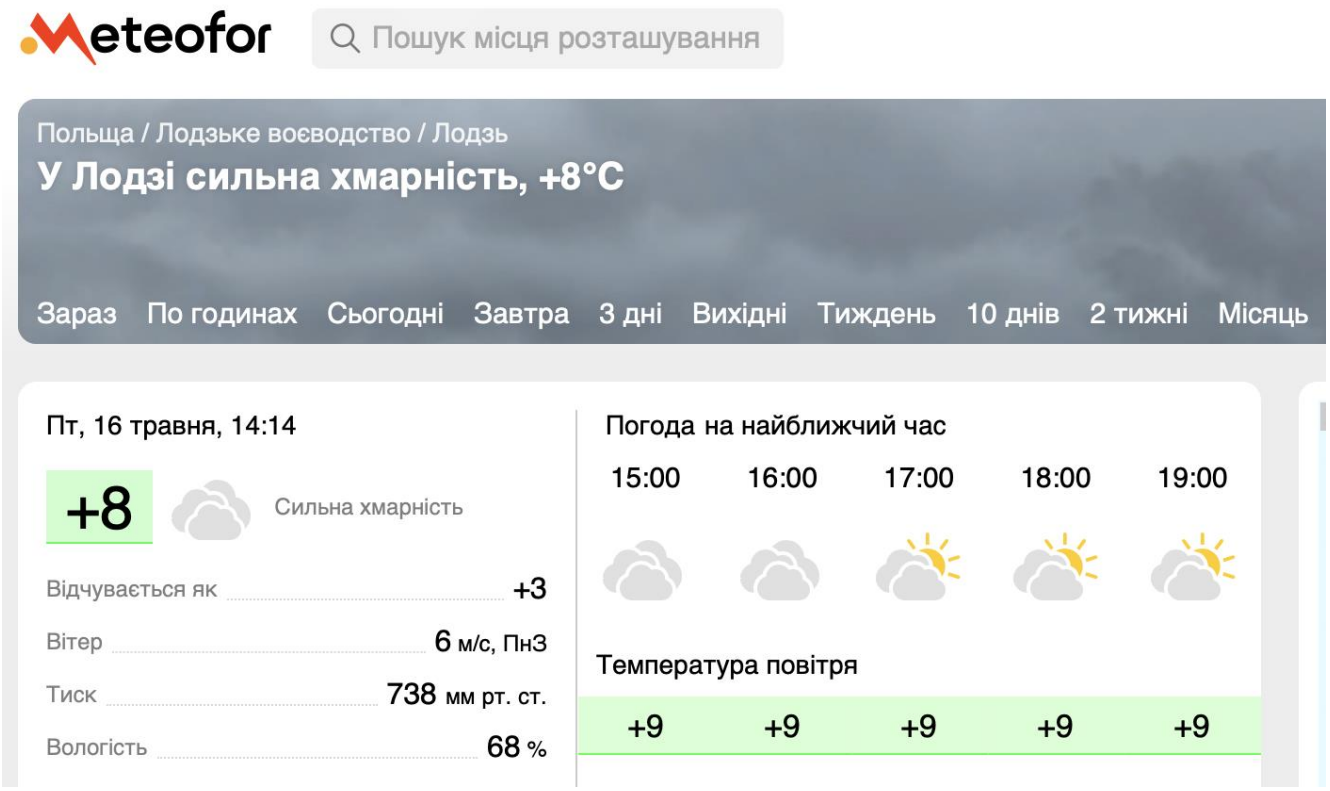


Рисунок 1.2 Погода Meteofor

3) Sinoptik.ua (https://sinoptik.ua/)

Sinoptik.ua — це популярний погодний ресурс, орієнтований на українську аудиторію, який надає актуальну метеорологічну інформацію, довгострокові прогнози та історичні погодні дані для різних регіонів України. Платформа дозволяє користувачам переглядати прогнози на кілька днів уперед, зокрема дані

про температуру, опади, вітер, вологість і тиск. Однак, Sinoptik.ua не надає медичних порад або попереджень щодо потенційних ризиків для здоров’я, таких як рівень забруднення повітря чи індекс ультрафіолетового випромінювання. Відсутність персоналізованих рекомендацій для користувачів робить цей ресурс менш орієнтованим на індивідуальні потреби, які можуть бути важливими для людей з метеозалежністю або алергіями. Тому цей сайт не ідеальний для тих, хто шукає погоду, яка враховує особливості здоров'я користувача, як це буде в твоєму проєкті.

Рисунок 1.3 Погода Sinoptik

4) AccuWeather (https://www.accuweather.com/)

AccuWeather — це глобальна погодна платформа, яка надає точні та детальні прогнози для мільйонів локацій по всьому світу. Платформа спеціалізується на наданні погодної інформації, зокрема на прогнозах температури, опадів, швидкості вітру, вологості та інших метеорологічних даних. AccuWeather пропонує розділ

"Health & Activities", де можна побачити індекс для алергіків, рівень забруднення повітря, а також ризики теплового удару. Однак, ці дані не є персоналізованими і не включають індивідуальних рекомендацій щодо того, як користувачам слід поводитися в залежності від їхнього здоров'я чи інших особливостей. Сайт також включає інтерактивні карти, відео, новини та мобільні додатки, що дозволяє отримувати погодну інформацію на будь-якому пристрої в реальному часі. AccuWeather також доступний на понад 100 мовах і дозволяє користувачам отримувати точні дані по містах, країнах та навіть за географічними координатами.

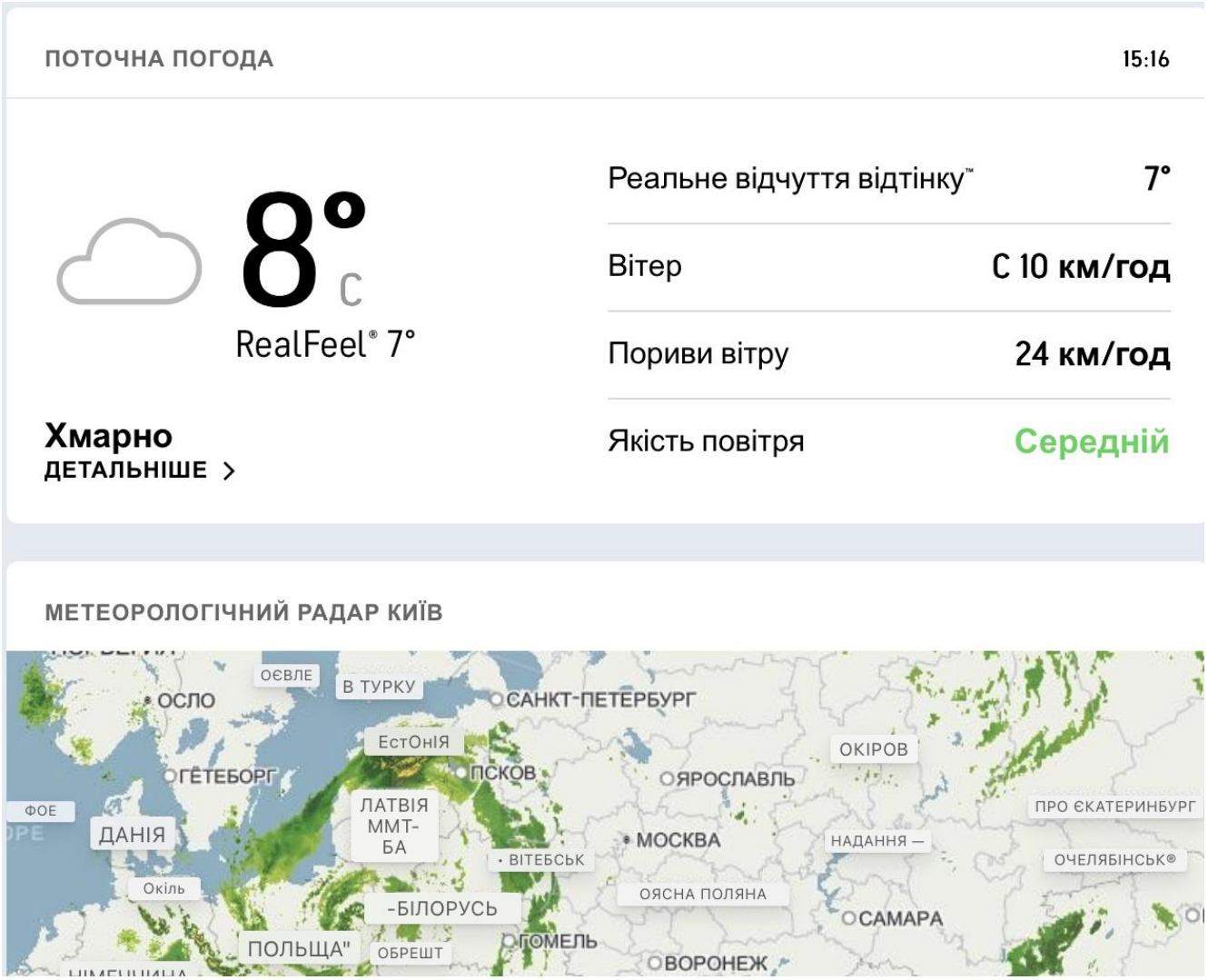


Рисунок 1.4 Погода AccuWeather

5) Weather.com (https://weather.com/)

Weather.com — це офіційний сайт американської компанії The Weather Channel, який є одним із найвідвідуваніших погодних ресурсів у світі. Сервіс надає

метеорологічну інформацію в режимі реального часу для різних регіонів планети, включаючи поточну температуру, опади, швидкість і напрям вітру, вологість, атмосферний тиск, а також детальний прогноз на кілька днів уперед. Крім базової метеоінформації, платформа пропонує спеціалізовані індекси здоров'я, як-от рівень УФ-випромінювання, ризики для алергіків, умови для людей з метеозалежністю, а також попередження про небезпечні погодні явища. Сайт підтримує інтерактивні погодні карти, радари та новинні розділи, де публікуються актуальні повідомлення про кліматичні події, шторми, урагани та інші природні явища. Weather.com також доступний у вигляді мобільних додатків для Android та iOS, що дозволяє користувачам отримувати персоналізовану інформацію з урахуванням їхнього місцезнаходження.

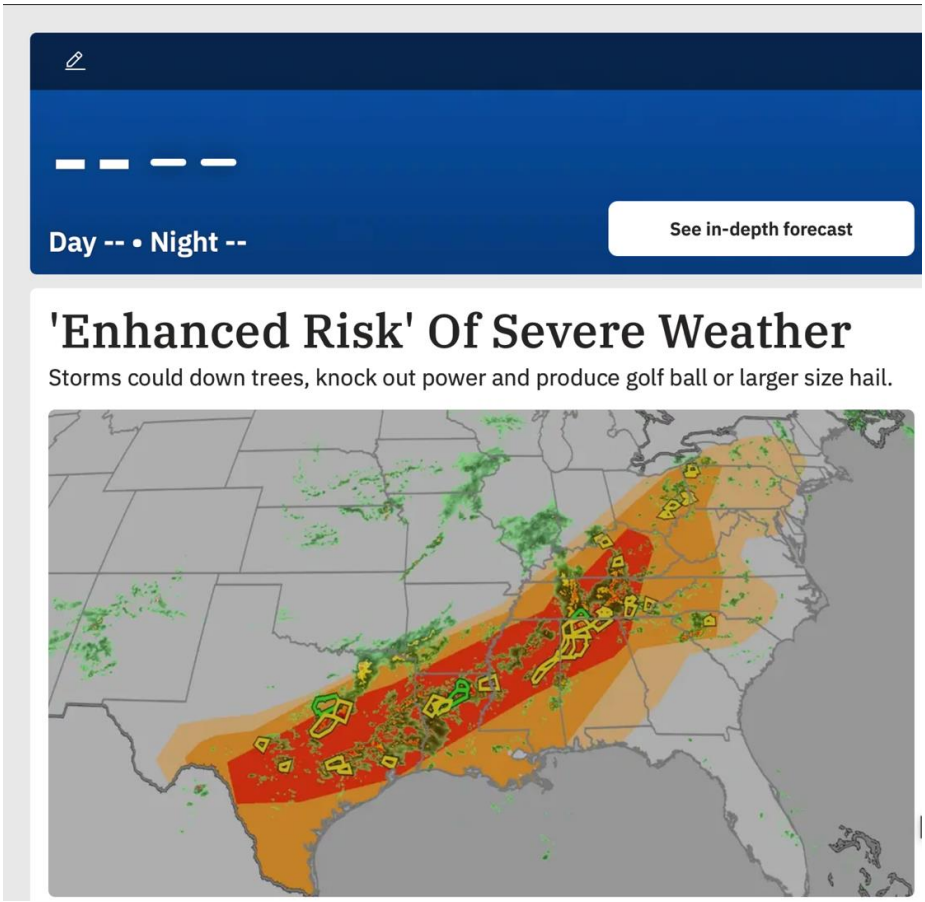


Рисунок 1.5 Погода Weather.com

1.2 Основні недоліки наявних рішень

Попри функціональність перелічених платформ, більшість з них мають схожі обмеження:

- Вони не враховують стан здоров’я або індивідуальні потреби користувача.
- Відсутні поради або рекомендації, які могли б допомогти метеозалежним людям, алергікам чи хронічно хворим.
- Дані часто подаються в складній або надто технічній формі без пояснення впливу на організм.

1.3 Обґрунтування доцільності створення нової платформи

В цьому пункті приведено приклади чому саме моя платформа розробки буде краще ніж інші.

1) Індивідуальні рекомендації для здоров’я.

   Платформа буде аналізувати не лише температуру чи вологість, а також інші важливі параметри, які можуть спричиняти дискомфорт у повягденному житті.
2) Користувач буде отримувати інформацію не лише про погоду, а й про потенціні ризики. Наприклад: “сьогодні моливі мігрені у метозалежних людей” або “погіршення повітря – ризики загострення броніальної астми”.
3) Простота інтерфейсу.

   Платформа розрахована на звичайних користувачів, які не мають спеціальної підготовки. Інформація буде подаватися у вигляді коротких зрозумілих повідомлень.
4) Можливість в майбутномоу вдосконалити веб-сайт з медичними системами.

1.4 Технічний аналіз реалізації подібних рішень

Більшість сучасних погодних веб-платформ реалізовані за класичною архітектурою «клієнт–сервер» та загалом використовують відкритих або комерційних метео–API. Нижче написані ключові технічні компоненти, які зазвичай використовуються в таких сервісах:

Фронтенд (інтерфейс користувача):

- Використовується HTML, CSS, JavaScript, фреймворки типу React, Angular або Vue.
- Інтерфейс адаптується під мобільні пристрої.
- Візуалізація даних через діаграми або інтерактивні карти (наприклад, на базі Leaflet.js чи Mapbox).

Бекенд (логіка обробки даних):

- Поширені мови Python (Flask, Django), Node.js, PHP, Java.
- Сервер здійснює запити до API (наприклад, OpenWeather API) та обробляє відповіді.
- Додатково може використовуватись обробка даних за допомогою бібліотек: Pandas, NumPy, Scikit–learn.

API джерела даних:

- OpenWeatherMap API – один із найпопулярніших, підтримує поточну погоду, 5–денний прогноз, історичні дані.
- Дані зазвичай повертаються у форматі JSON.
- Для роботи потрібен ключ доступу (API Key).

Бази даних:

- PostgreSQL, MySQL, MongoDB – зберігають історію запитів, профілі користувачів, результати аналізу.
- У вашому випадку – також можна зберігати симптоми або рекомендації.

1.5 Технічний опис веб-платформ для аналізу погоди

1) OpenWeatherMap

Загальна архітекутра платформи OpenWeatherMap за моделлю клієнт–сервер, де клієнти це – користувачі або програми, які надсилають HTTP-запити до серверів OpeanWeatherMap, які повертають дані у форматі JSON або XML.

Також платформа має API-інтерфейс. Платформа надає RESTful API**,** доступне за допомогою стандартних HTTP-запитів. Користувачі отримують ключ

API після реєстрації, який використовується для авторизації.

Основні типи API:

Current Weather Data API — поточна погода за координатами, містом або ID.

5 Day / 3 Hour Forecast API — прогноз на 5 днів з інтервалом у 3 години.

16 Day Daily Forecast API — щоденний прогноз на 16 днів (доступно в платних тарифах).

Historical Weather Data API — історичні погодні дані.

Air Pollution API — рівень забруднення повітря.

UV Index API — ультрафіолетовий індекс.

Weather Alerts API — погодні попередження (тільки у вибраних регіонах).

Вхідні параментри: API підтримує різні типи параметрів, такі як назва міста, географічні координати, ID міста, поштовий індекс, мова, одиниці виміру.

Формати відповіді: Дані які були надішлені клієнтом повертаються в форматі JSON, XML.

Платформа: Данна платформа має мовну підтримку в якій є понад 40 мов, включно з українською.

Обмеження та тарифи: OpenWeather має кілька тарифних планів:

Free — обмежена кількість запитів/хвилину (~60) та базові дані.

Startup, Professional, Enterprise — включають більший ліміт, розширені дані (погодні карти, історія, попередження).

Обмеження:

На безкоштовному тарифі можуть блокувати ключ при перевищенні ліміту.

Деякі типи даних (наприклад, погодні карти або історія понад 1 рік) доступні лише за платною підпискою.

Інтиграція: OpenWeatherMap легко інтегрується з різними технологіями.

Python, JavaScript, PHP, Java, C# — через HTTP-запити та обробку JSON-відповіді.

Flask, Django, React, Vue, Android, iOS — підтримка сучасних стеків.

Можливість інтеграції у IoT-пристрої (розумні будинки, сенсори).

2) AccuWeather

Загальна архітекутра платформи OpenWeatherMap працює за моделлю клієнт–сервер. Клієнти надсилають HTTP-запити до серверів AccuWeather, які обробляють запити та повертають погодні дані у форматі JSON.

API-інтерфейс: AccuWeather надає RESTful API після реєстрації та отримання ключа доступу (API key). Ключ є обов'язковим для авторизації у більшості запитів.

Основні типи API:

Current Conditions API – поточні погодні умови для заданої локації.

5-Day Forecast API / 12-Hour Forecast API – прогноз на 5 днів або 12 годин з погодинним оновленням.

Severe Weather Alerts API – система попередження про екстремальні погодні явища.

Air Quality API – індекс якості повітря.

Lifestyle Indices API – оцінка впливу погоди на повсякденні активності.

Вхідні параметри: API підтримує передачу параметрів через ID локації (LocationKey), координати, мову відповіді, одиниці виміру.

Формати відповіді: JSON.

Мовна підтримка: Понад 30 мов, включаючи українську.

Обмеження та тарифи: Free Tier — до 50 запитів на день та Developer/Enterprise Plans — доступ до розширених API та збільшених лімітів. Частина API доступна лише у платній підписці.

Інтеграція:

Можлива інтеграція з Python, Java, .NET, JavaScript, PHP.

Підходить для мобільних додатків (Android, iOS), веб-сервісів і IoT-проєктів.

3) Sinoptik.ua

Загальна архітектура має веб-інтерфейс з доступом до погодних даних через HTTP-запити. Проте офіційного відкритого API не надається. Дані можна отримувати лише через парсинг HTML-сторінок або неофіційні API.

API-інтерфейс: Офіційно API не документоване. Частково використовуються внутрішні запити JavaScript, які можуть бути відслідковані через браузер (наприклад, при побудові графіків). Будь-яке використання в сторонніх додатках порушує умови користування.

Типові запити:

Дані на день/тиждень для певного міста.

Архів погоди за минулі роки.

Графіки температури, вологості, тиску.

Формати відповіді: HTML / JSON (використовується на внутрішніх модулях).

Мовна підтримка: Українська, російська, англійська.

Обмеження та тарифи:

Відсутній публічний доступ до API.

Дані використовуються виключно на сайті.

Масовий збір даних через парсинг може бути заблокований.

Інтеграція: Технічно можливо використовувати HTML-парсинг через Python (наприклад, з бібліотекою BeautifulSoup), але така інтеграція неофіційна.

Не рекомендується для проєктів, де потрібна стабільність та легальність.

4) Gismeteo

Загальна архітектура має модель клієнт–сервер. Сервер обробляє запити користувачів та повертає дані про погоду. Має веб-інтерфейс, мобільні додатки та частково доступне API.

API-інтерфейс: Gismeteo має офіційне API, доступне після реєстрації та підтвердження проєкту. Доступ надається через RESTful API з обов’язковим API ключем.

Основні типи API:

Current Weather API — поточні погодні умови.

Forecast API — погодинний та добовий прогноз.

Climate Normals API — кліматичні норми.

Astronomy API — фази місяця, схід/захід сонця.

Historical Weather API — архівні дані (у платному доступі).

Вхідні параметри:

Місто, координати, мова, часовий пояс, формат (JSON).

Формати відповіді: JSON.

Мовна підтримка: Понад 20 мов, зокрема українська.

Обмеження та тарифи: безкоштовна версія має обмеження на кількість запитів.

Платні тарифи надають розширений доступ і підвищені ліміти.

Інтеграція: підтримуються популярні мови: Python, JavaScript, Java.

Можлива інтеграція з Android, iOS, веб-додатками та сенсорними пристроями.

5) Wheather.com

Загальна архітектура працює за класичною моделлю клієнт–сервер: браузери користувачів надсилають HTTP(S)-запити до серверів IBM Weather Company, які у відповідь повертають погодні дані у форматі JSON або HTML. Основна частина даних обробляється на стороні сервера та оновлюється в реальному часі.

API-інтерфейс: Платформа не надає відкритого API напряму на сайті Weather.com, але API доступний через IBM Weather Company Data API, що є частиною IBM Cloud. Цей API розрахований на комерційне використання та корпоративні рішення.

Основні типи API (через IBM Cloud):

Current Conditions — поточна погода.

Daily Forecast / Hourly Forecast — прогнози на годину, день або 10 днів.

Weather Alerts — попередження про стихійні явища.

Air Quality — показники якості повітря.

Historical Observations — погодна історія.

Tropical and Storm Tracker — трекінг штормів, ураганів, тайфунів.

Вхідні параметри:

Локація (місто, координати, IP), мова, одиниці виміру, часовий пояс.

Формати відповіді: JSON (для API IBM); HTML (для веб-користувача).

Мовна підтримка: Доступний понад 40 мовами, включаючи українську (на сайті — частково).

Обмеження та тарифи:

Безкоштовний перегляд на сайті — необмежений.

API через IBM Weather доступне на умовах підписки (є безкоштовний trial).

Деякі дані (наприклад, штормові попередження або погодна аналітика) — лише в платному доступі.

Інтеграція:

IBM Weather API інтегрується з хмарними рішеннями (IBM Cloud, Watson).

Підтримуються мови: Python, Java, Node.js, .NET.

Підходить для великих бізнес-проєктів, аналітики, логістики, сільського господарства.

2 ЗАСОБИ І ТЕХНОЛОГІЇ СТВОРЕННЯ ВЕБ-ПЛАТФОРМИ

2.1 Вибір технологій розробки веб-платформи

Для розробки даної платформи інтегроване середовище розробки (IDE) PyCharm а також мова програмування Python з використанням фреймворка Flask. Взявши мову програмування Python та фреймворк Flask за основу та комбінуючи його з HTML, CSS і мовою програмування JavaScript, можна створити цілісне середовище для розробки веб-платформи. Flask і Python відповідатимуть за серверну частину, а HTML і CSS — за зовнішній вигляд сайту та взаємодію з користувачем. JavaScript забезпечуватиме динамічність та інтерактивність веб-платформи.

Давайте розглянемо біль детальніше інструменти для створення веб-платформи.

2.1.1 Основні характеристики Python

Python одна з найпотужніших мов програмування завдяки своїй простоті, читабельності та великій кількості бібліотек і фреймворків для різних задач. Я обрав Python для розробки цієї веб-платформи, оскільки маю значний досвід роботи з цією мовою в університеті. Під час навчання я працював над різними проєктами, що включали обробку даних, автоматизацію процесів, розробку ботів та веб-додатків. Набутий досвід дозволив мені краще зрозуміти можливості Python і був одною .

Оскільки Python широко використовується в області веб-розробки, завдяки фреймворкам, таким як Flask та Django, він є ідеальним вибором для створення сучасних та масштабованих веб-додатків. Легкість інтеграції Python з іншими технологіями, його велика підтримка бібліотеками для роботи з базами даних, API та веб-технологіями також є однією з причин вибору цієї мови для розробки платформи.

Мій досвід роботи з Python дозволяє мені ефективно використовувати всі його переваги, що допомагає оптимізувати процес розробки та забезпечити високу продуктивність веб-додатка. Враховуючи ці фактори, вибір Python для цього проєкту був логічним і обґрунтованим.

Набутий досвід дозволив мені краще зрозуміти можливості Python і був одним із головних факторів, що вплинув на вибір теми моєї дипломної роботи. Зрозумівши, як ефективно використовувати цю мову для вирішення різних завдань, я вирішив застосувати її для створення веб-платформи, яка аналізує погодні умови та надає медичні рекомендації.

2.1.2 Функціональні характеристики Flask

Мною було обрано фреймворк Flask, оскільки він є ідеальним вибором для моєї веб-платформи завдяки своїй простоті та легкій взаємодії з іншими інструментами. Flask дозволяє розпочати розробку з базових функцій і додавати необхідні розширення у разі потреби. Оскільки Flask добре інтегрується з Python, я можу використовувати різні бібліотеки для збору інформації, обробки даних та виконання інших завдань.

Flask — це мікрофреймворк для веб-розробки на Python, який вирізняється простотою та високою гнучкістю. На відміну від більш масштабних фреймворків таких як Django, Pyramid, Tornado, Web2py наприклад, він надає лише базовий набір інструментів, залишаючи розробнику свободу вибору додаткових компонентів. Завдяки цьому Flask особливо добре підходить для невеликих і середніх проєктів, де важливо мати повний контроль над структурою й функціональністю додатка.

Flask є основою програми, яка відповідає за маршрутизацію URL, обробку запитів і відповідей. Він перенаправляє HTTP-запити до відповідних Python-функцій, які обробляючи їх повертають відповіді у вигляді HTML-сторінок, JSON або інших форматів. Flask також підтримує різноманітні розширення, що додають функціональність, автентифікація користувачів та управління базами даних.

Фреймворк може показувати HTML-файли або динамічно створювати HTML-контент за допомогою шаблонів Jinja2, вставляючи змінні та керуючі структури безпосередньо в HTML-код. Він також взаємодіє з базами даних, наприклад, SQLite, MySQL для зберігання, отримання, оновлення та видалення даних, що допомагає легко взаємодіяти або змінювати інформацію у даних.

2.1.3 Середа розробки PyCharm

Для розробки веб-платформи мною було обрано PyCharm як основне середовище розробки. PyCharm є потужним інтегрованим середовищем розробки (IDE), яке спеціально створене для Python і підтримує численні функції, що полегшують програмування та підвищують ефективність роботи. Його переваги — це зручний інтерфейс, інтелектуальна підтримка коду, можливості для налагодження, інтеграція з системами контролю версій та підтримка багатьох популярних бібліотек і фреймворків, таких як Flask, Django т.д.

Однією з ключових причин вибору PyCharm є його зручність для розробки великих проєктів, оскільки середовище підтримує автоматичне доповнення коду, рефакторинг, тестування та налагодження. PyCharm також має вбудовані інструменти для роботи з базами даних, що є важливим для створення веб-платформи з інтеграцією з базою даних.

Ще однією перевагою є те, що як студент, я маю можливість отримати професійну версію PyCharm через студентську пошту, що дає доступ до додаткових функцій. Професійна версія надає можливість роботи з додатковими інструментами для веб-розробки, таких як підтримка фреймворків для розробки на Python, інтеграція з сучасними технологіями веб-розробки, підтримка роботи з великими проєктами та розширений інтерфейс для роботи з базами даних. Завдяки цьому, я отримую ще більше можливостей для оптимізації процесу розробки та роботи з проєктом.

2.1.4 Загальні характеристики HTML та CSS

HTML (Hypertext Markup Language) є основною мовою розмітки для створення веб-сторінок. Він визначає структуру веб-сторінки, надаючи інформацію про елементи, які повинні бути на сторінці, такі як заголовки, параграфи, списки, зображення та інші компоненти. HTML є стандартом для створення та організації контенту в інтернеті.

CSS (Cascading Style Sheets) — це мова стилів, яка використовується для опису вигляду документа, написаного на HTML. CSS дозволяє задавати стиль для елементів на сторінці, таких як кольори, шрифти, розміри, відступи, позиціонування та багато інших властивостей. За допомогою CSS можна створювати адаптивний та зручний інтерфейс для користувачів, підлаштовуючи вигляд веб-сторінки під різні пристрої та екрани.

Ці два інструменти є основою для створення веб-сторінок. Вони дозволяють не тільки створити структуру сайту, але й надавати йому естетичний вигляд, забезпечуючи зручне користування інтерфейсом. У процесі розробки веб-платформи HTML був використаний для створення основної структури сторінок, а CSS — для їх стилізації та оптимізації відображення на різних пристроях, що забезпечує кращу взаємодію з користувачем.

2.1.5 Особливості роботи та характеристики JavaScript

JavaScript — це високорівнева мова програмування, яка використовується для створення динамічних і інтерактивних елементів на веб-сторінках. Вона дозволяє додавати до веб-сторінок функціональність, яку не можна реалізувати лише за допомогою HTML та CSS, наприклад, обробку подій, зміну контенту на сторінці без перезавантаження, валідацію форм, анімації тощо.

JavaScript виконується на стороні клієнта, що означає, що код обробляється безпосередньо в браузері користувача. Це дозволяє значно зменшити навантаження на сервер і забезпечити швидку реакцію веб-сторінки на дії користувача. Завдяки асинхронному програмуванню за допомогою таких технологій, як AJAX, JavaScript

дозволяє завантажувати дані та оновлювати сторінки без повного перезавантаження.

В моєму випадку JavaScript використовувався як другорядна мова для обробки стилізації сайту та динамічних елементів, таких як інтерактивні карти, обробка форм, а також для забезпечення взаємодії користувача з веб-сторінкою без необхідності її перезавантаження. У розробці веб-платформи JavaScript використовується для додавання динамічних елементів на сайті, таких як інтерактивні карти, форми з перевіркою даних, а також для забезпечення інтерактивної взаємодії користувача з контентом без необхідності перезавантаження сторінки. Наприклад, для відображення прогнози погоди у реальному часі, зміни елементів на сторінці без її перезавантаження, а також для реалізації інтерфейсу, що реагує на події користувача.

За допомогою JavaScript також можна здійснювати взаємодію з різними API, що дозволяє отримувати та обробляти дані з зовнішніх джерел, таких як сервіси для отримання прогнозу погоди або геокодування.

2.1.6 Основні характеристики SQLlite3

SQLite3 — це легка вбудована система керування базами даних (СКБД), яка зберігає всі дані в одному локальному файлі. Вона не потребує окремого сервера або складної конфігурації, що робить її зручною для невеликих проєктів, десктопних застосунків, мобільних додатків та веб-розробки. SQLite підтримує стандарт SQL, що дозволяє створювати таблиці, виконувати запити, фільтрацію, сортування, агрегацію даних тощо.

SQLite3 виконується безпосередньо в додатку (на стороні сервера або клієнта в залежності від реалізації), що дозволяє уникнути складностей з підключенням до окремої СКБД. Вона є кросплатформною, має відкритий вихідний код і не потребує встановлення, що робить її дуже зручною для швидкого розгортання.

У моєму випадку SQLite3 використовується для зберігання медичних рекомендацій, які залежать від поточної температури. У базі даних зберігаються наперед визначені рекомендації для різних діапазонів температур. Після отримання

актуальних метеоданих система автоматично визначає, в який температурний діапазон потрапляє поточна температура, і на основі цього надає користувачу відповідні поради. Це дозволяє динамічно адаптувати рекомендації до погодних умов у реальному часі.

SQLite3 ідеально підійшла для реалізації прототипу веб-платформи, що працює з погодними умовами, оскільки забезпечила просту й ефективну інтеграцію з Python-кодом без додаткових залежностей.

2.2 Інтеграція з OpenWeather API

OpenWeather API — це зовнішній програмний інтерфейс (Application Programming Interface), який надає доступ до різноманітних метеорологічних даних у режимі реального часу. Даний сервіс дозволяє отримувати не лише поточну погоду, а й прогноз на декілька днів, історичні дані, погодні карти, інформацію про якість повітря, УФ-індекс тощо. Дані надаються у зручному форматі JSON, що спрощує їх обробку та інтеграцію у веб-застосунки.

Мною було використано у межах розробки веб-платформи, OpenWeather API використовується для збору погодної інформації в залежності від міста, яке вводить користувач. Основна мета інтеграції — надати актуальну інформацію про погодні умови, яка надалі використовується як для візуалізації на сайті, так і для формування медичних рекомендацій, орієнтованих на стан погоди.

Основні етапи взаємодії з OpenWeather API:

1. Отримання API-ключа

   Для використання сервісу розробник повинен зареєструватися на офіційному сайті openweathermap.org і отримати унікальний API-ключ, який використовується для автентифікації запитів.

2. Формування HTTP-запиту

   Веб-додаток формує запит до API з урахуванням обраного міста, одиниць вимірювання мови відповіді а також бажаного типу прогнозу.

3. Обробка відповіді у форматі JSON

   У відповідь на запит API повертає структуру у форматі JSON, яка містить

усю потрібну інформацію: температуру, відчутну температуру, вологість, швидкість вітру, погодні умови, піктограму тощо.

4. Парсинг та обробка даних

   Отримані дані обробляються на серверній стороні за допомогою Python. Витягуються лише необхідні значення, які потім передаються до шаблону HTML для виведення на екран.

5. Візуалізація та взаємодія з користувачем

   За допомогою HTML, CSS та JavaScript інформація відображається у зручному вигляді: блоки прогнозу погоди, іконки стану неба, температура, вологість тощо. Дані оновлюються автоматично при запиті нового міста або при оновленні сторінки.

Причини вибору OpenWeather API:

Детальна та зручна документація, яка дозволяє швидко зрозуміти структуру запитів і відповідей.

Можливість безкоштовного використання з базовим обсягом запитів, що повністю покриває потреби дипломного проєкту.

Гнучкість налаштування запитів: можна змінювати мову, формат, кількість днів у прогнозі, тип даних тощо.

Широкий функціонал — крім стандартної погоди, API підтримує історичні дані, погодні карти, геокодування тощо.

Легка інтеграція з Python — просте отримання та парсинг JSON-відповідей, що дозволяє легко реалізовувати обробку на сервері.

Завдяки цьому API вдалося реалізувати динамічну систему, яка надає користувачеві свіжу інформацію про погоду в будь-якому місті світу, а також адаптувати рекомендації залежно від отриманих метеоданих.

Давайте розглянемо приклади запиту до OpenWeatherAPI та приклад відповіді у форматі JSON а також обробку та виведення основної інформації Рисунок(2.1, 2.2, 2.3)

```python
import requests

API_KEY = 'ваш_ключ'
city = 'Kyiv'
url = f"https://api.openweathermap.org/data/2.5/weather?q
    ={city}&appid={API_KEY}&units=metric&lang=ua"

response = requests.get(url)
data = response.json()
```

Рисунок 2.1 Приклад запиту до OpenWeather API


```json
{
  "weather": [
    {
      "description": "хмарно з проясненнями",
      "icon": "03d"
    }
  ],
  "main": {
    "temp": 18.53,
    "feels_like": 17.76,
    "humidity": 56,
    "pressure": 1013
  },
  "wind": {
    "speed": 3.6
  },
  "name": "Kyiv"
}
```


Рисунок 2.2 Приклад відповіді у форматі JSON

```
temp = data['main']['temp']
weather_desc = data['weather'][0]['description']
humidity = data['main']['humidity']
wind_speed = data['wind']['speed']
city_name = data['name']

print(f"Погода у місті {city_name}: {weather_desc}")
print(f"Температура: {temp}°C")
print(f"Вологість: {humidity}%")
print(f"Швидкість вітру: {wind_speed} м/с")
```

Рисунок 2.3 Обробка та виведення основної інформації

2.3 Принцип роботи системи збору та обробки метеоданих

Давайте покроково розберемо взаємодію з OpenWeather API у моєму веб-додатку:

1. Користувач вводить назву міста у формі на веб-сторінці (через HTML-інтерфейс).
2. Запит передається на сервер, де обробляється за допомогою Python (фреймворк Flask).
3. Flask-обробник формує HTTP-запит до OpenWeather API, підставляючи назву міста, API-ключ, мову та одиниці вимірювання.
4. OpenWeather API надсилає відповідь у форматі JSON з погодними даними.
5. Python-логіка розбирає (парсить) отриманий JSON, витягує необхідну інформацію (температура, вологість, опис погоди, вітер тощо).
6. Ці дані передаються до HTML-шаблону, де вони динамічно виводяться на сторінку (використовуючи Jinja2 у Flask).
7. Додатково, модуль рекомендацій аналізує ці дані та генерує індивідуальні поради (наприклад, при спеці — уникати виходу на вулицю, пити більше води).
8. Користувач бачить на сайті поточну погоду та рекомендації, пов’язані з погодними умовами.

Також наглядно я показав це схемою (рисунок 2.4)

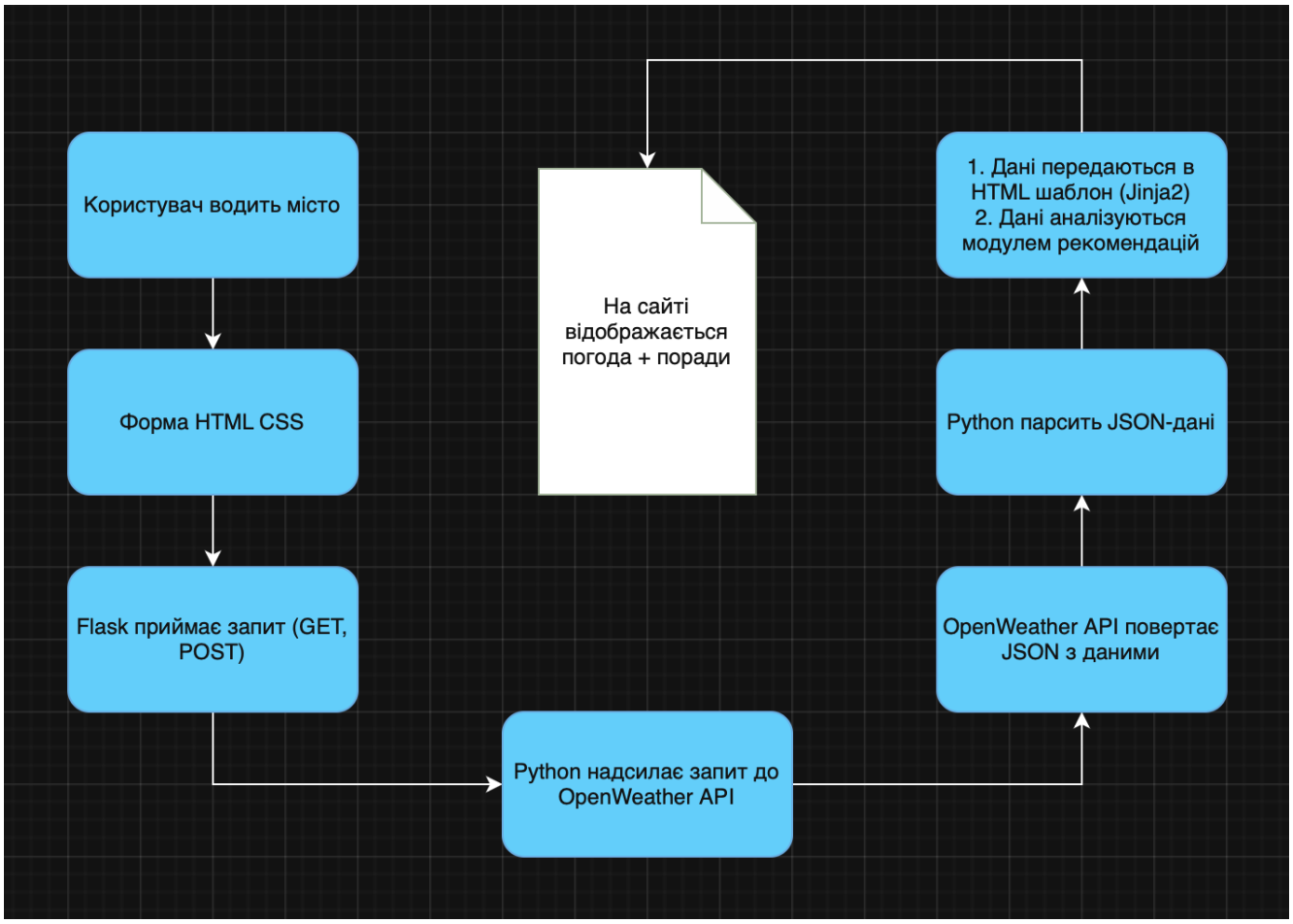


Рисунок 2.4 Схема принципу роботи системи збору та обробки метеоданих

3 РЕАЛІЗАЦІЯ ВЕБ-ПЛАТФОРМИ

3.1 Створення структури

Як уже було зазначено в пункті 2.1, для розробки веб-платформи мною було обрано мову програмування Python та інтегроване середовище розробки (IDE) PyCharm. Першим кроком ми створюємо структуру (дерево).

Дерево в мовах програмування — це структура даних**,** яка організована за ієрархічним принципом, подібно до перевернутого дерева в природі. Далі нам потрібно визначитися, як саме ми хочемо, щоб наша програма працювала. Переважно є вже стандарти, як саме потрібно створювати дерево.

Я створив структуру проєкту Flask (веб-додаток на Python), яка організована за найкращими практиками, щоб код був зрозумілим, масштабованим і зручним для підтримки.

Пояснення:

WheatherProject — це головна папка проєкту (може бути назвою репозиторію або каталогу).

.venv — це віртуальне середовище Python. Тут зберігаються встановлені залежності, щоб вони не конфліктували з глобальними бібліотеками Python. Це віртуальне середовище не додається в репозиторій (вказується у .gitignore**).**

app — це головна папка програми Flask.

static — містить статичні файли (ті, які не змінюються сервером): зображення, CSS, JavaScript.

assets/img — зображення, іконки тощо.

favicon.ico — іконка сайту (що відображається у вкладці браузера).

js/script.js — JavaScript-файл для обробки динамічної взаємодії на клієнті.

styles.css — CSS-файл для стилізації сайту.

templates містить HTML-шаблони, які обробляє Jinja2 (шаблонізатор Flask).

index.html — головна сторінка сайту.

__init__.py — Означає, що папка app — це Python-пакет. Також тут часто ініціалізується Flask-додаток.

forms.py — Форми для вводу користувача (з використанням Flask-WTF або WTForms).

routes.py — тут описані маршрути (шляхи), які обробляє сервер: що показувати при переході на ту чи іншу URL-адресу.

config.py — Файл конфігурацій. Тут зберігаються налаштування додатка (наприклад, API-ключі, шляхи, режим розробки).

main.py — Головний файл, з якого запускається веб-додаток.

weather.py — Ймовірно, модуль, де реалізована логіка роботи з API погоди: запити, обробка даних тощо.

.gitignore — Файл, у якому вказано, які файли й папки ігнорувати при додаванні в Git (наприклад, .venv, .pyc тощо).

ssh-keygen... — SSH-ключі, які генеруються для доступу до віддалених репозиторіїв (наприклад, GitHub). Ці файли не повинні потрапляти у публічний репозиторій.

Така структура:

розділяє логіку: фронтенд (static/templates) і бекенд (Python-логіка),

робить код легше підтримуваним і читабельним, дозволяє масштабувати додаток (наприклад, додати нові маршрути, шаблони або API), відповідає офіційним рекомендаціям Flask.

Тому я вибрав таку структуру, як вказано на рисунку 3.1.

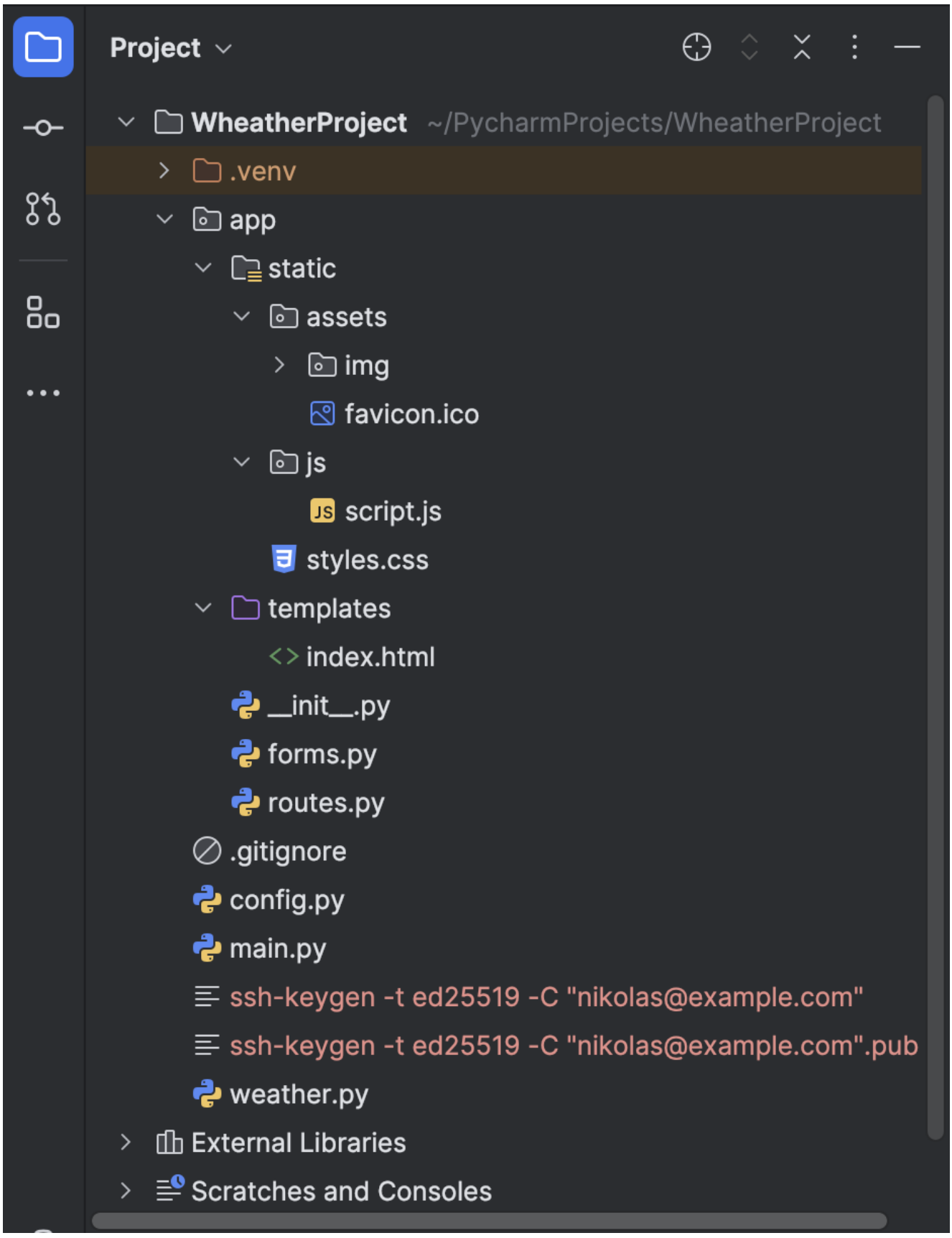


Рисунок 3.1 Структура (дерево)

3.2 Обробка HTTP-запиту

Спочатку ми створюємо файл «__init__.py» функція create_app(), яка створює та налаштовує Flask-додаток. Такий підхід називається «фабрика додатку» (application factory pattern). Зазвичай такий підхід використовується для масштабованих Flask-проєктів (Рисунок3.2).

```
1  from flask import Flask
2  from app.routes import bp
3
4  def create_app():  2 usages  nikolas
5      app = Flask(__name__)
6      app.secret_key = 'supersecretkey'
7      app.register_blueprint(bp)
8      return app
```

Рисунок 3.2 Струтура файлу «__init__.py»

Таблиця 3.1

from flask import Flask	Импортуємо клас Flask — головний клас вебдодатку.
from app.routes import bp	Импортуємо Blueprint bp з файлу routes.py. Blueprint — це спосіб організувати маршрути (routes) у великому додатку.
def create_app(): app = Flask(__name__)	Flask(__name__) — створює додаток. __name__ потрібен Flask для визначення шляху до ресурсів (шаблонів, static тощо).
app.secret_key = 'supersecretkey'	secret_key використовується для безпеки, наприклад, підпису cookies, форм тощо.

Продовеження таблиці 3.1

app.register_blueprint(bp)	Додає до додатку всі маршрути (routes), які описані в bp. Це дозволяє тримати код чистим та модульним.
return app	Повертаємо готовий додаток Flask, який буде запускатися з іншого файлу, в нашому випадку «main.py»

Наступним кроком ми створюємо файл запуску (точку входу). Створювати ми її будемо в файлі «main.py». Тепер кожен раз коли нам буде потрібно перевірити роботу нашого проєкта ми будемо запускати саме з цього файла. Запускається з зеленої кнопки (Рисунок 3.3).

Рисунок 3.3 Структура файлу «main.py» (точка входу)

Таблиця 3.2

Пояснення команд файла «main.py»

from app import create_app	Цей рядок імпортує функцію create_app() з модуля app
app = create_app()	Викликається функція create_app(), і створюється об'єкт додатку app.
if __name__ == '__main__': app.run(debug=True)	Це умовне твердження означає: «Якщо цей файл запускається напряму,

Продовження таблиці 3.2

	а не імпортується як модуль — то запустити Flask-сервер».

Таким чином: create_app() створює додаток, register_blueprint() додає маршрути, app.run() запускає сервер.

3.3 Отримання API_KEY

Наступним кроком створюємо файл «config.py»

```
1   API_KEY = "cc2[redacted]2"
2
```

Рисунок 3.4 Створення змінної «API_KEY»

Щоб отримати свій персональний API key нам потрбіно перейти за посиланням на сайт OpenWeatherMap https://openweathermap.org/ зареєструватися та увійти в свій аккаунт → натиснути на свій логін та обрати пункт My API keys (Рисунок 3.4)

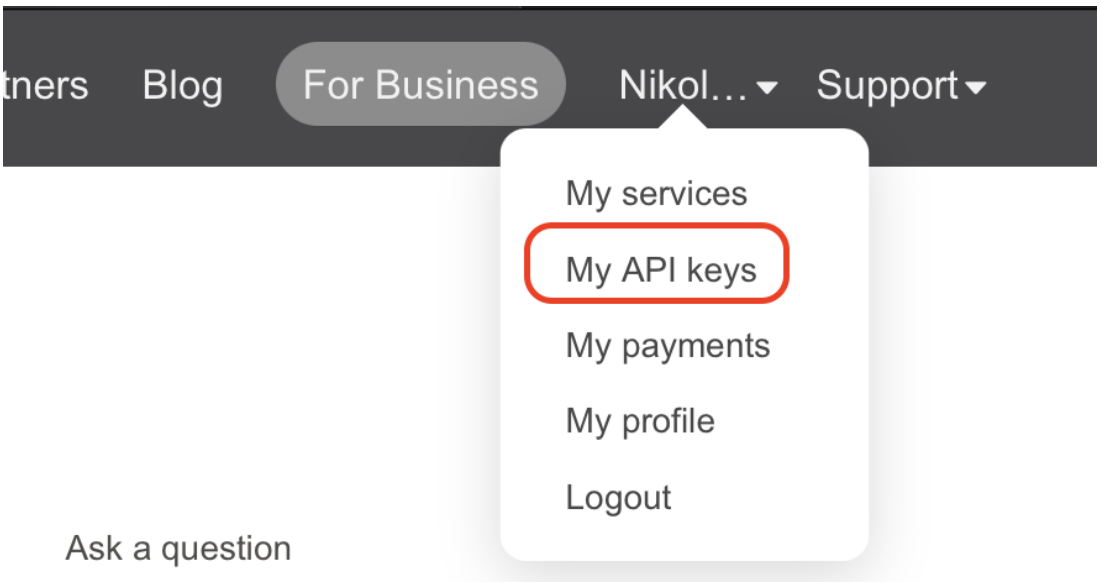


Рисунок 3.5 Меню з API key

Далі нас перекидає на іншу сторінку, де ми можемо отримати наш API key або створити його самостійно (Рисунок3.6).

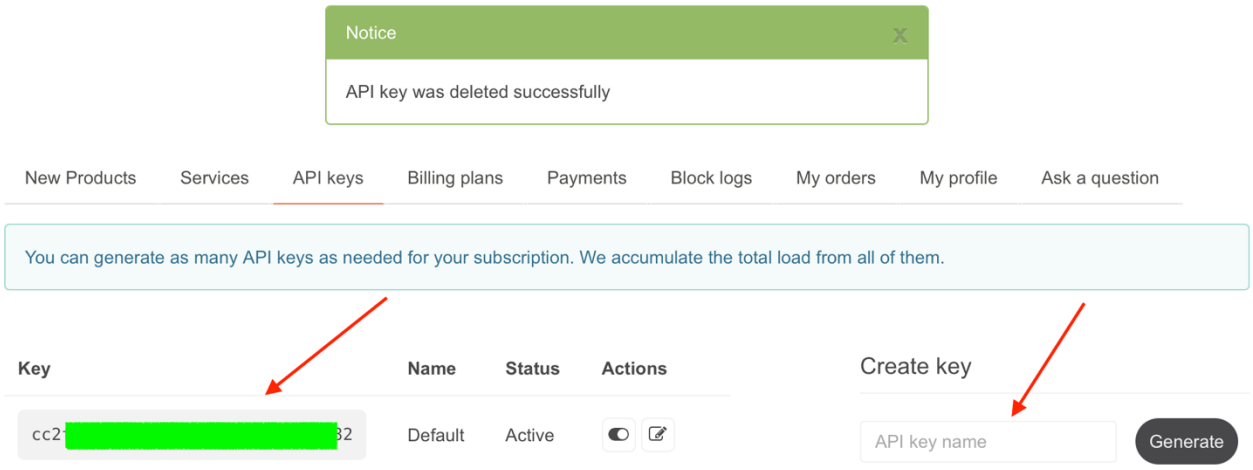


Рисунок 3.6 API key

Рекомендація: нікому не давайте свій ключ, тому що він є, як логін і пароль до вашого акаунта. Якщо хтось отримає його, то зможе використовувати ключ від вашого імені, отримати доступ до вашої інформації (якщо це дозволяє API), або зловживати ним (наприклад, надсилати фальшиві чи шкідливі запити).

Якщо у вас є платний тариф або обмежена кількість запитів:

- Хтось може швидко вичерпати ліміт запитів;
- Ви можете отримати несподівано великий рахунок за використання API (якщо включений білінг);
- Сервіс може тимчасово заблокувати ключ за перевищення ліміту.

Сервіси можуть заблокувати ваш ключ, якщо запідозрять підозрілі активності (багато запитів, доступ з різних IP, неправильне використання тощо). Це може порушити роботу вашої програми або сайту.

3.4 Створення запиту погоди

Наступним кроком створюємо файл «weather.py», де буде написана логіка нашого запиту з сайта OpenWeatherMap (Рисунок3.7)

```
1  import requests
2  import locale
3  from datetime import datetime
4  from collections import defaultdict
5  from config import API_KEY
```

Рисунок 3.7 Підключення імпортів в файлі «weather.py»

Пояснення:

- запити — бібліотека для HTTP-запросів.
- locale — дозволяє встановити мову і формат дати/часу.
- datetime — робота з датами.
- defaultdict — словник з автоматично створеним списком, якщо ключ не знайдено.
- API_KEY — ключ доступу до OpenWeather API (береться з файлу config.py).

```
7  try:
8      locale.setlocale(locale.LC_TIME, locale: 'en_US.UTF-8')
9  except locale.Error:
10     locale.setlocale(locale.LC_TIME, locale: 'C')
```

Рисунок 3.8 Метод try

Пояснення:

Налаштовує виведення днів неділі на англійському (наприклад, понеділок, вівторок).

Якщо в системі немає потрібної локалі, встановлюється дефолтна C.

```
13  def get_weather_by_city(city):  2 usages  nikolas *
14      url = f'https://api.openweathermap.org/data/2.5/forecast?q={city}&appid={API_KEY}&units=metric&lang=ua'
15      response = requests.get(url)
16      data = response.json()
```

Рисунок 3.9 Функція «get_weather_by_city(city)»

Пояснення:

- Відправляється GET-запит до OpenWeather API.
- q={city} — місто.
- appid={API_KEY} — ключ API.
- unit=metric — температура в °C.
- lang=ua — мова відповід— український.

```
18      if response.status_code != 200 or 'list' not in data:
19          return None
```

Рисунок 3.10 Перевірка помилки

Пояснення:

Якщо немає успішної відповіді (код 200) або відсутній список потрібних блоків, повертається None.

```
23      for item in data['list']:
24          date_str = item['dt_txt']
25          date = datetime.strptime(date_str, format: '%Y-%m-%d %H:%M:%S')
26          day_name = date.strftime('%A')
27          temp = item['main']['temp']
28          forecast_by_day[day_name].append(temp)
```

Рисунок 3.11 Угруповання по днях

Пояснення:

Прогноз містить дані на 3 години — ми збираємо їх у групі по дням неділі (понеділок, вівторок і т.д.). Всі температури зберігаються в списку по дню.

```
30      weekdays_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
31      today = datetime.today().strftime('%A')
32      start_index = weekdays_order.index(today)
33      ordered_days = weekdays_order[start_index:] + weekdays_order[:start_index]
```

Рисунок 3.12 Сортування днів

Пояснення:

Отримуємо поточний день тижня і далі створюємо список ordered_days, починаючи з сьогодення (щоб прогноз починався з поточного дня).

```
35      result = []
36      for day in ordered_days:
37          if day in forecast_by_day:
38              temps = forecast_by_day[day]
39              day_data = {
40                  'day': day[:3],
41                  'temp_max': round(max(temps)),
42                  'temp_min': round(min(temps))
43              }
44              result.append(day_data)
45          if len(result) == 5:
46              break
47      return result
```

Рисунок 3.13 Формування результату

Пояснення:

- Для кожного дня (в правильному порядку) знаходимо максимальну та мінімальну температуру.
- Формуємо словник і додаємо його до результату.
- Обмежуємося п'ятьма днями.

3.5 Обробка головної сторінки сайту

Наступним кроком нам потрібно створити обробку головної сторінки сайта в фалйлі «routes.py». Як і любий новий файл починаємо з імпорту бібліотек та файлів (Рисунок3.14)

```
1   from flask import Blueprint, render_template, request
2   from weather import get_weather_by_city
3   from db import get_recommendation_from_db
```

Рисунок 3.14 Підключення файлів та бібліотек

Пояснення:

- Blueprint — дозволяє структурувати Flask-додаток на модулі.
- render_template — рендерить HTML-шаблон.
- request — дозволяє працювати з HTTP-запитом (наприклад, отримати дані з форми).
- get_weather_by_city — функція, яка отримує прогноз погоди для заданого міста.
- get_recommendation_from_db — функція, яка знаходить рекомендацію в базі даних на основі температури.

Далі ми створюємо маршрут головної сторінки. Ця, яка виконується, коли користувач відкриває головну сторінку ("/"), або надсилає форму (наприклад, ввів місто і натиснув кнопку).

```
5   bp = Blueprint( name: 'main', __name__)
6
7   @bp.route( rule: "/", methods=["GET", "POST"])   nikolas *
8   def index():
9       city = None
10      forecast = None
11      recommendation = None
```

Рисунок 3.15 Створення маршруту головної сторінки

Наступним кроком буде отримання назви міста з HTML-форми (з поля name="cityInput"), виклик функції, яка повертає список прогнозів погоди, а також створення перевірок, необхідних для відстеження переданої інформації — щоб переконатися, що всі етапи були пройдені та працюють коректно.

```
    if request.method == "POST":
        city = request.form.get("cityInput")
        print(f"City received: {city}")

        forecast = get_weather_by_city(city)
        if forecast:
            print(f"Forecast received: {forecast}")

            if forecast and len(forecast) > 0 and 'temp_max' in forecast[0]:
                recommendation = get_recommendation_from_db(forecast[0]['temp_max'])

                print(f"Recommendation generated!!!!: {recommendation}")
            else:
                print("Error: No 'temp_max' found in forecast.")
        else:
            print("Error: No forecast data received.")

    print(f"City: {city}, Forecast: {forecast}, Recommendation: {recommendation}")

    return render_template( template_name_or_list: "index.html", city=city, forecast=forecast, recommendation=recommendation)
```

Рисунок 3.16 POST-запит: прогноз погоди і рекомендації

Пояснення:

- Функція index() буде виконуватись при відкритті головної сторінки (/) або при надсиланні форми.
- Отримання назви міста з HTML-форми, з поля name="cityInput".
- get_weather_by_city(city) викликає функцію, яка повертає список прогнозів погоди.
- if forecast: — Перевірка, чи функція повернула валідні дані.
- (forecast[0]) є ключ 'temp_max'. — перевіряється, що список не порожній і що в першому елементі

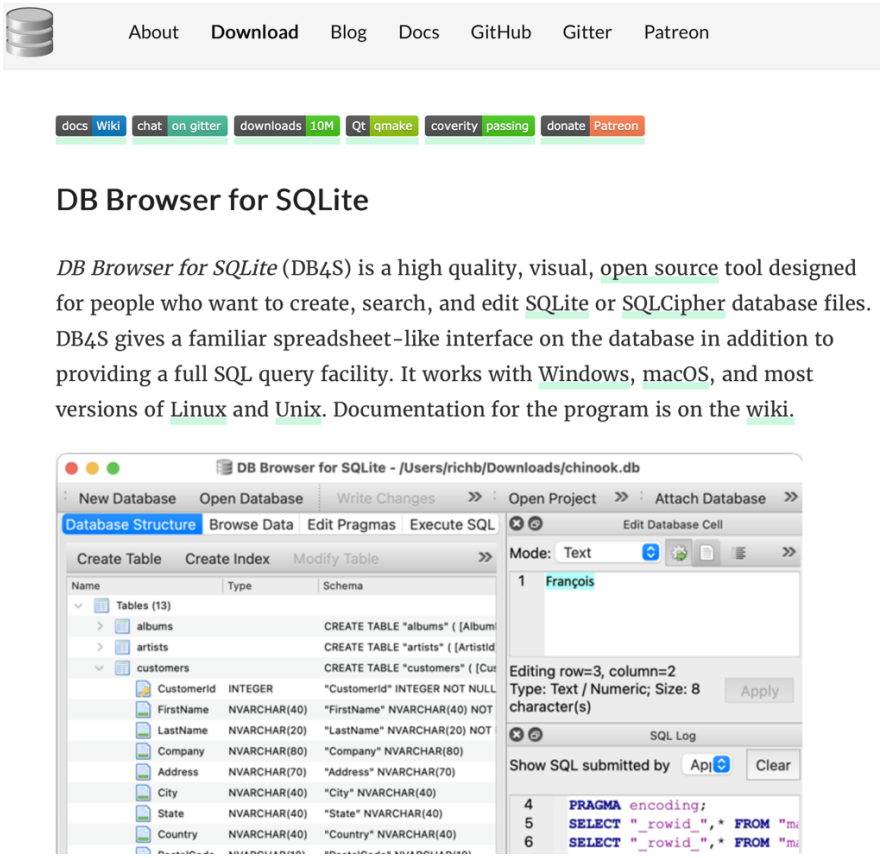
- Якщо все гаразд — викликається get_recommendation_from_db(temp_max), яка шукає рекомендацію в БД для цієї температури. Якщо щось пішло не так виводяться повідомлення в консоль для дебагу.
- Відображає HTML-сторінку index.html і передає в неї 3 змінні: city — введене місто, forecast — прогноз погоди (список температур), recommendation — отримана з БД рекомендація

3.6 Створення SQL бази

Зберігання наших рекомендацій буде здійснено за допомогою коду Python у файлі db.py.

Також створення таблиці для зберігання можна було б виконати вручну через додаток SQLite3, але я обрав перший спосіб для демонстрації своїх навичок.

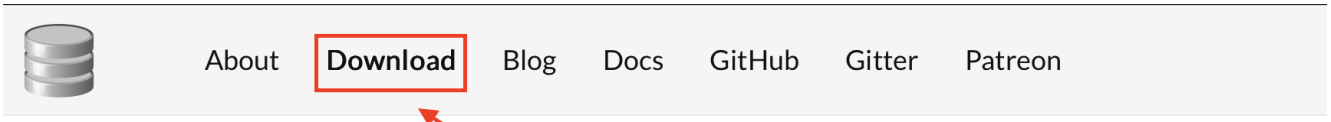
Створення SQL-бази є добрим рішенням для зберігання великої кількості інформації — у нашому випадку, рекомендацій. Це дозволить змінювати, додавати або видаляти рекомендації, вносити поправки, збирати інформацію від інших користувачів, проводити аналіз і вдосконалювати систему без потреби щоразу використовувати код Python. Але ще нам буде потрібно завантажити програму для керування базами даних https://sqlitebrowser.org/ DB Browser for SQLite



DB Browser for SQLite (DB4S) is a high quality, visual, open source tool designed for people who want to create, search, and edit SQLite or SQLCipher database files. DB4S gives a familiar spreadsheet-like interface on the database in addition to providing a full SQL query facility. It works with Windows, macOS, and most versions of Linux and Unix. Documentation for the program is on the wiki.

Рисунок 3.17 DB Browser for SQLite

Далі необхідно перейти до вкладки «Download» та обрати потрібну версію — або для Windows, або для macOS.



Downloads

(**Please** consider sponsoring us on Patreon 😀)

Windows

Our latest release (3.13.1) for Windows:

- DB Browser for SQLite - Standard installer for 32-bit Windows
- DB Browser for SQLite - .zip (no installer) for 32-bit Windows
- DB Browser for SQLite - Standard installer for 64-bit Windows
- DB Browser for SQLite - .zip (no installer) for 64-bit Windows

Free code signing provided by SignPath.io, certificate by SignPath Foundation.

Windows PortableApp

Our latest release (3.13.1) for Windows:

- DB Browser for SQLite - PortableApp

Note - If for any reason the standard Windows release does not work (e.g. gives an error), try a nightly build (below).

Рисунок 3.18 Версія DB Browser for SQLlite для Windows

macOS

Our latest release (3.13.1) for macOS:

- DB Browser for SQLite (Universal)

Версія DB Browser for SQLlite для macOS рисунок 3.19

Після завантаження програми відкриваємо її та можемо вільно працювати з нею.

```
1  import sqlite3
2
3  conn = sqlite3.connect('recommendations.db')
4  cursor = conn.cursor()
```

Рисунок 3.18 Імпорт та створення бази даних

Пояснення: Імпортується модуль sqlite3 для роботи з локальною базою даних. Створюється з'єднання з файлом бази recommendations.db. Створюється курсор для виконання SQL-запитів.

Далі, оскільки в нас ще немає таблиці, куди ми будемо записувати наші рекомендації, нам потрібно її створити.

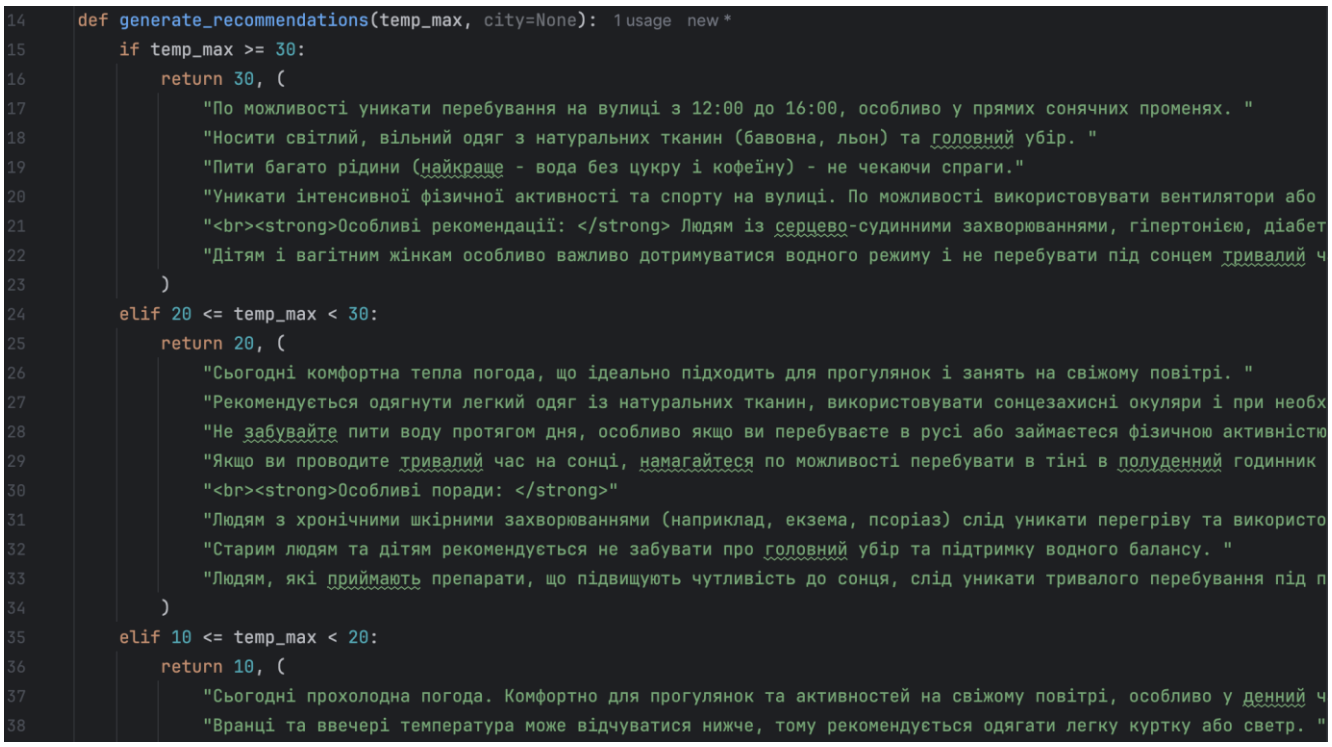
```
6  cursor.execute('''
7  CREATE TABLE IF NOT EXISTS recommendations (
8      id INTEGER PRIMARY KEY AUTOINCREMENT,
9      temp INTEGER,
10     recommendation TEXT
11 )
```

Рисунок 3.19 Створення таблиці

Пояснення:

Таблиця recommendations містить:

- id — унікальний ідентифікатор;
- temp — температура, для якої зберігається рекомендація;
- recommendation — текст рекомендації.



```
def generate_recommendations(temp_max, city=None):  1 usage  new *
    if temp_max >= 30:
        return 30, (
            "По можливості уникати перебування на вулиці з 12:00 до 16:00, особливо у прямих сонячних променях. "
            "Носити світлий, вільний одяг з натуральних тканин (бавовна, льон) та головний убір. "
            "Пити багато рідини (найкраще - вода без цукру і кофеїну) - не чекаючи спраги."
            "Уникати інтенсивної фізичної активності та спорту на вулиці. По можливості використовувати вентилятори або
            "<br><strong>Особливі рекомендації: </strong> Людям із серцево-судинними захворюваннями, гіпертонією, діабет
            "Дітям і вагітним жінкам особливо важливо дотримуватися водного режиму і не перебувати під сонцем тривалий ч
        )
    elif 20 <= temp_max < 30:
        return 20, (
            "Сьогодні комфортна тепла погода, що ідеально підходить для прогулянок і занять на свіжому повітрі. "
            "Рекомендується одягнути легкий одяг із натуральних тканин, використовувати сонцезахисні окуляри і при необх
            "Не забувайте пити воду протягом дня, особливо якщо ви перебуваєте в русі або займаєтеся фізичною активністю
            "Якщо ви проводите тривалий час на сонці, намагайтеся по можливості перебувати в тіні в полуденний годинник
            "<br><strong>Особливі поради: </strong>"
            "Людям з хронічними шкірними захворюваннями (наприклад, екзема, псоріаз) слід уникати перегріву та використо
            "Старим людям та дітям рекомендується не забувати про головний убір та підтримку водного балансу. "
            "Людям, які приймають препарати, що підвищують чутливість до сонця, слід уникати тривалого перебування під п
        )
    elif 10 <= temp_max < 20:
        return 10, (
            "Сьогодні прохолодна погода. Комфортно для прогулянок та активностей на свіжому повітрі, особливо у денний ч
            "Вранці та ввечері температура може відчуватися нижче, тому рекомендується одягати легку куртку або светр. "
```

Рисунок 3.20 Функція generate_recommendations

Рекомендації можна було б одразу записати в базу та надалі також змінювати їх безпосередньо там. Однак для демонстрації даних я вирішив прописати це в програмі.

Пояснення:

Ця функція повертає пару: температуру (ключ), текст рекомендації.

Вона використовує if...elif блоки, щоб повернути відповідні поради залежно від температури:

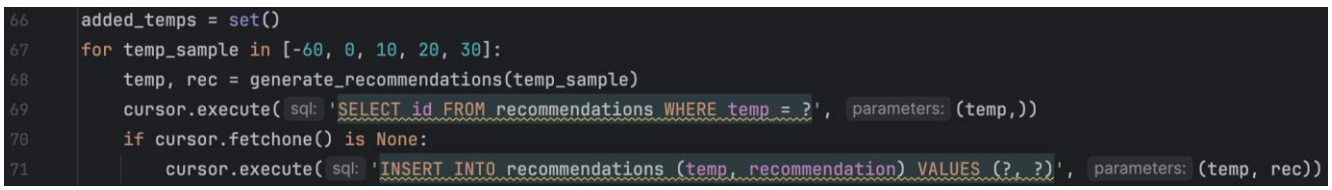
≥30 — дуже жарко,

20-29 — тепло,

10-19 — прохолодно,

0-9 — холодно,

<0 — дуже холодно.



```
added_temps = set()
for temp_sample in [-60, 0, 10, 20, 30]:
    temp, rec = generate_recommendations(temp_sample)
    cursor.execute( sql: 'SELECT id FROM recommendations WHERE temp = ?', parameters: (temp,))
    if cursor.fetchone() is None:
        cursor.execute( sql: 'INSERT INTO recommendations (temp, recommendation) VALUES (?, ?)', parameters: (temp, rec))
```

Рисунок 3.21 Заповнення бази даних згенерованими рекомендаціями

Пояснення:

- Створюється список прикладів температур.
- Для кожного з них генерується рекомендація.
- Якщо в БД ще немає такої температури, вона додається.

```
73      conn.commit()
74      conn.close()
```

Рисунок 3.22 Збереження та закриття бази

Пояснення:

- conn.commit() — зберігає всі зміни.
- conn.close() — закриває з'єднання з базою.

```
77  def get_recommendation_from_db(temp_max):  2 usages  new *
78      conn = sqlite3.connect("recommendations.db")
79      cursor = conn.cursor()
80
81
82      cursor.execute( sql: """
83          SELECT recommendation FROM recommendations
84          WHERE temp <= ?
85          ORDER BY temp DESC
86          LIMIT 1
87      """,  parameters: (temp_max,))
88
89      row = cursor.fetchone()
90      conn.close()
91
92      if row:
93          return row[0]
94      return "Рекомендація не знайдена для цієї температури."
95  print("The database has been successfully created and populated.")
```

Рисунок 3.23 Функція для отримання рекомендації з БД

Пояснення: ця функція підключається до бази даних, витягує найбільше значення температури, яке менше або дорівнює temp_max, і повертає відповідну рекомендацію. Якщо нічого не знайдено — повертає повідомлення про відсутність даних.

Після того, як ми написали код і запустили його, з’явиться повідомлення про те, що базу успішно створено. «The database has been successfully created and populated».

Для того щоб переконатися, що база створилася відкриємо її в нашому додатку. Для цього нам потрібно знайти файл з базою даних у тій самій папці, де й наш проєкт

Пошагові дії відкриття БД (рисунок 3.24 – рисунок 3.27)

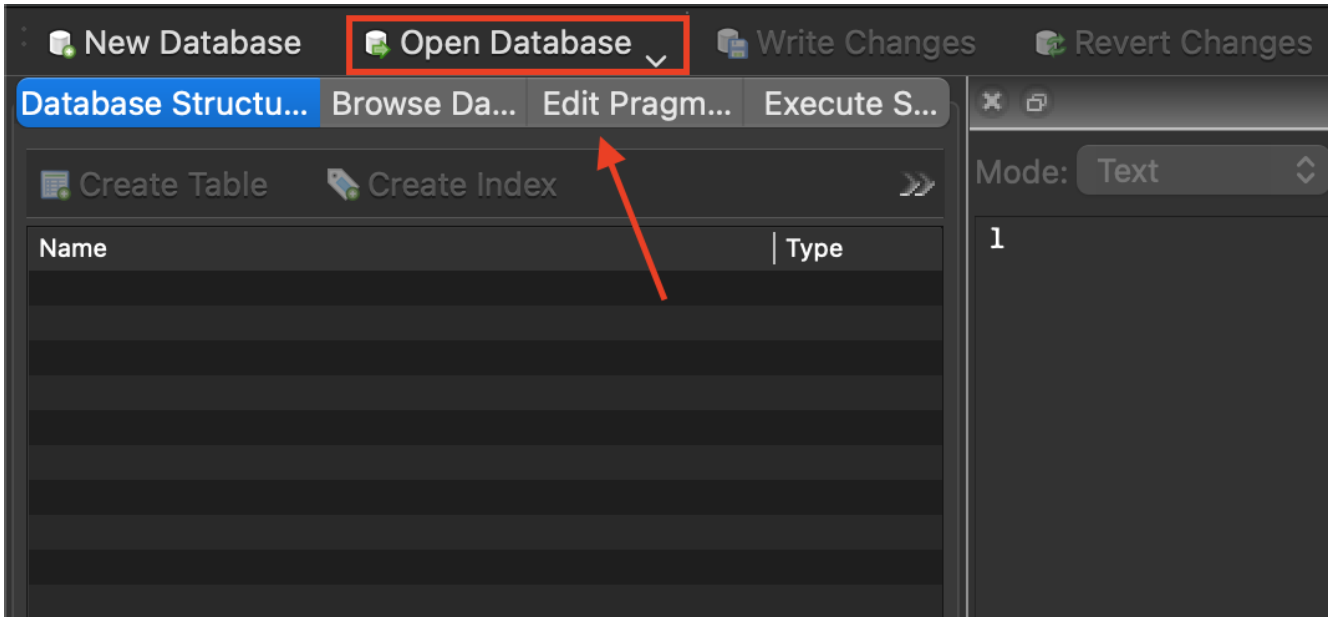


Рисунок 3.24 Відкриття БД частина (1)

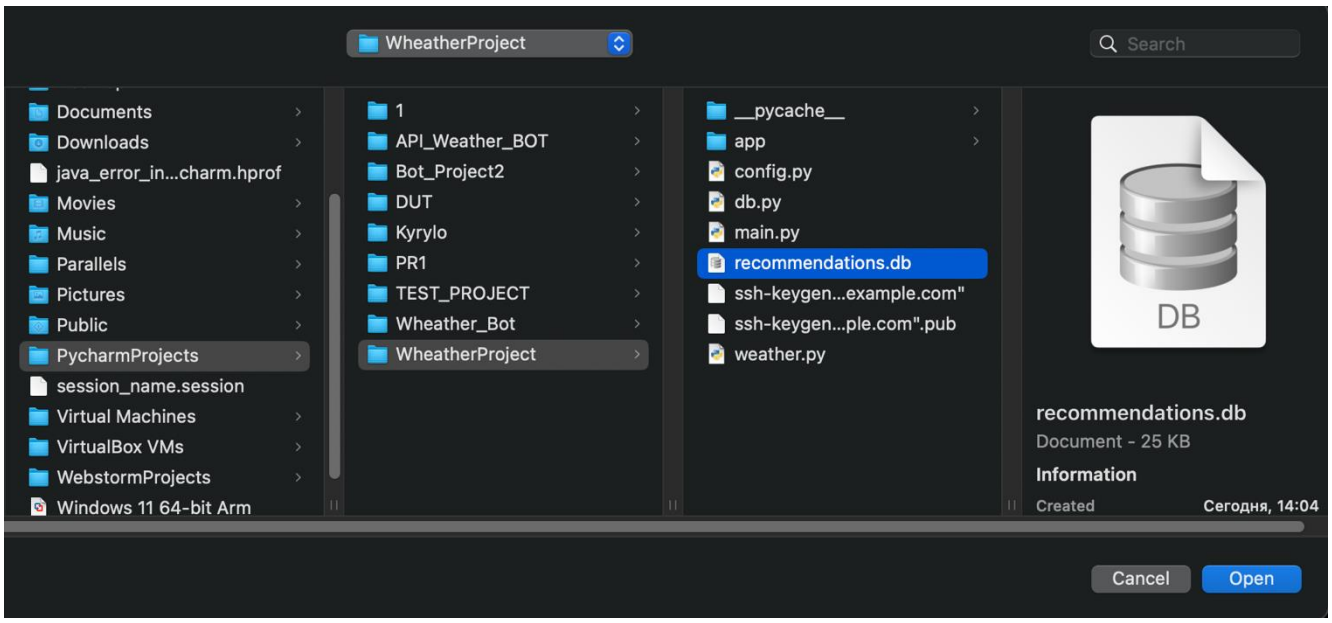


Рисунок 3.25 Відкриття БД частина (2)

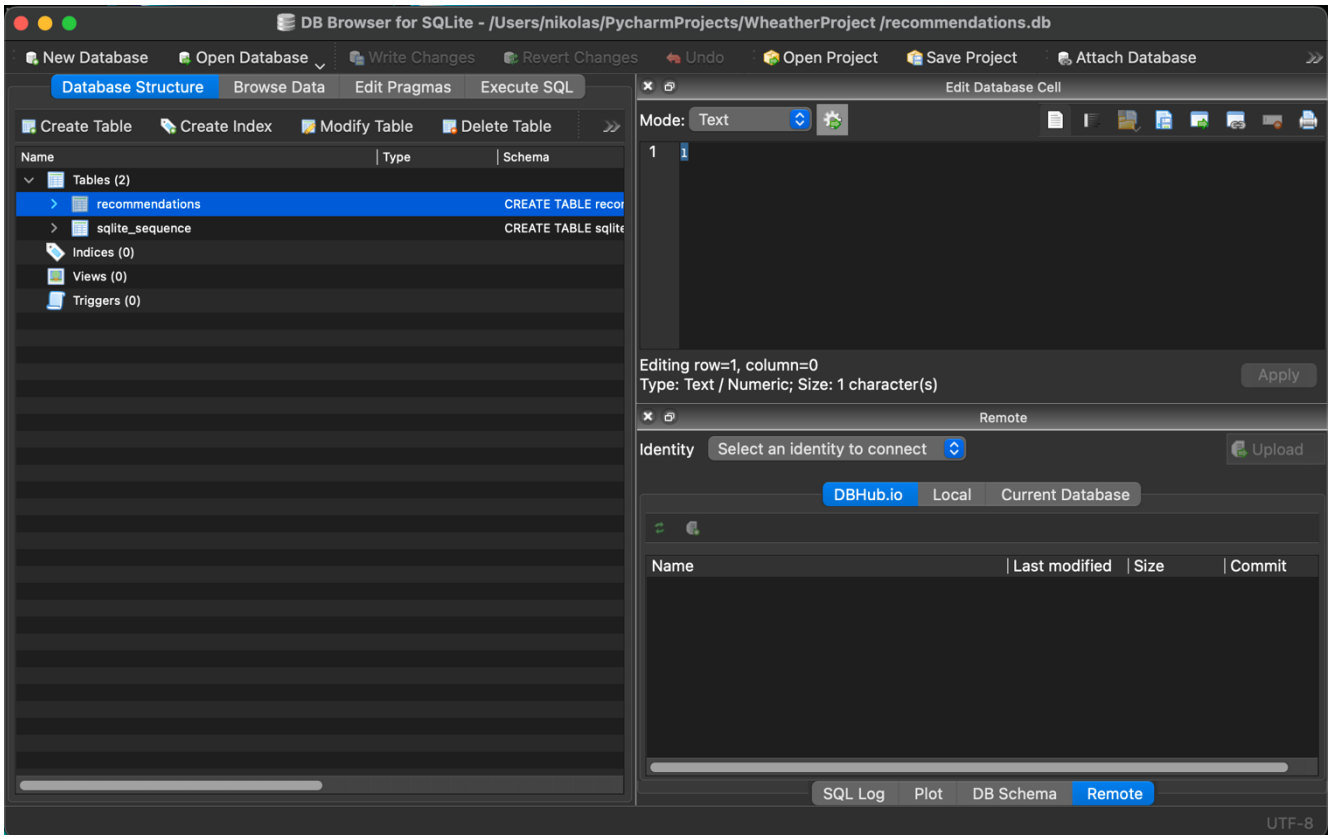


Рисунок 3.26 Відкриття БД частина (3)

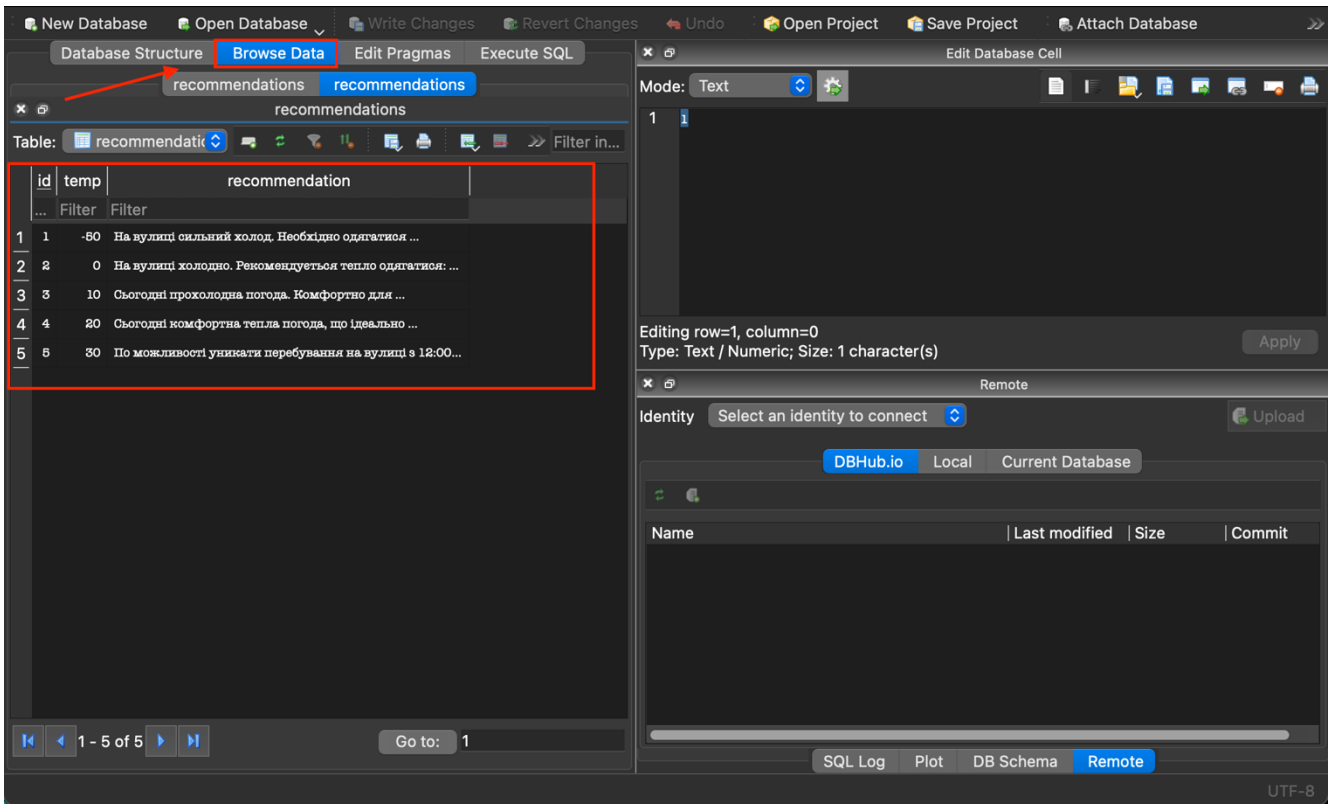


Рисунок 3.27 Відкриття БД частина (4)

Тепер ми бачимо, що наша БД створена з колонками id, temp, recommendation.

Зверніть увагу, що значення у колонці «temp» є дуже важливими, оскільки саме на їх основі відбувається вибір правильних рекомендацій для користувачів.

У подальшому ви зможете вдосконалювати рекомендації за бажанням, відштовхуючись від різних джерел інформації та змінювати їх прям в таблиці .

3.7 Візуалізація та стилізація веб-платформи.

Створення візуалізації та стилізації відбувається у файлах index.html, styles.css та script.js. Почнемо з HTML. Файл, у якому ми будемо писати, називається index.html. Зазвичай його називають саме так, але за бажанням можна обрати й іншу назву.

Основна структура

DOCTYPE та мова: Сторінка оголошена як HTML5 (<!DOCTYPE html>) з мовою за замовчуванням — англійська (lang="en").

Мета-теги: Кодування UTF-8 та адаптивна верстка для мобільних пристроїв.

Заголовок: Заголовок вкладки браузера — "WeatherByNikolas".

Фавікон: Іконка сайту підключена до папки assets.

Підключення стилів та шрифтів

Локальний CSS: Під'єднано файл styles.css із папки static.

Leaflet CSS: Підключено зовнішній CSS для картки Leaflet (карти OpenStreetMap).

FontAwesome: Підключено JS для іконок FontAwesome.

Google Fonts: Підключені шрифти Varela Round та Nunito з різними зображеннями.

Рисунок 3.28 Основа HTML-кода

Пояснення: цей код задає основу HTML-сторінки: вказує мову і кодування, робить сторінку адаптивною для мобільних пристроїв, задає заголовок і іконку сайту. Потім підключає зовнішні стилі — власні стилі, стилі для карти (Leaflet), шрифти Google і іконки Font Awesome. Далі починається тіло сторінки, де буде основний контент.

Наступний пункт це навігація (<nav>)

Навігаційна панель із фіксованим позиціонуванням зверху.

Логотип "Weather by Nikolas", який веде нагору сторінки.

Гамбургер кнопки для мобільних меню.

Посилання: About, Weather, Contact – плавний перехід до відповідних секцій на сторінці.

```
<!-- Navigation -->
<nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav">
    <div class="container">
        <a class="navbar-brand" href="#page-top">Weather by Nikolas</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                data-bs-target="#navbarResponsive" aria-controls="navbarResponsive"
                aria-expanded="false" aria-label="Toggle navigation">
            Menu <i class="fas fa-bars"></i>
        </button>
        <div class="collapse navbar-collapse" id="navbarResponsive">
            <ul class="navbar-nav ms-auto">
                <li class="nav-item"><a class="nav-link" href="#about">About</a></li>
                <li class="nav-item"><a class="nav-link" href="#projects">Weather</a></li>
                <li class="nav-item"><a class="nav-link" href="#signup">Contact</a></li>
            </ul>
        </div>
    </div>
</nav>
```

Рисунок 3.29 Навігація (<nav>)

Пояснення: цей блок створює верхнє меню сайту (навігацію), яке прилипає до верху сторінки. В ньому є назва сайту з посиланням і кнопка меню для мобільних пристроїв. Також є три пункти меню, які ведуть до різних частин сторінки: About, Weather і Contact.

Заголовок сторінки (<header>)

Центрований блок із заголовком «Weather» та підзаголовком — слоган.

Кнопка Get Started, яка веде до секції About.

```
<!-- Masthead -->
<header class="masthead">
    <div class="container d-flex h-100 align-items-center justify-content-center">
        <div class="text-center">
            <h1 class="mx-auto my-0 text-uppercase">Weather</h1>
            <h2 class="text-white-50 mx-auto mt-2 mb-5">Premium Weather for Free — by Nikolas.
                    Actually Want to See.</h2>
            <a class="btn btn-primary" href="#about">Get Started</a>
        </div>
    </div>
</header>
```

Рисунок 3.30 Заголовок сторінки (<header>)

Пояснення: цей блок — головний банер сайту (хедер), який показується одразу після відкриття сторінки. Він містить велику назву "Weather", підзаголовок із описом сервісу і кнопку "Get Started", що веде до блоку About. Усе вирівняно по центру екрану.

Секція About

Текст про проєкт пояснює, що це навчальний проєкт і не для публічного використання.

Зображення прев'ю (MacBook).

```
<!-- About -->
<section class="about-section text-center" id="about">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-lg-8">
                <h2 class="text-white mb-4">Built with Nikolas</h2>
                <p class="text-white-50">Developed as part of a university coursework to demonstrate individual
                    potential.
                    This version is not final, and is not intended for public or production use. Duplication or
                    distribution is prohibited.</p>
            </div>
        </div>
        <img class="img-fluid" src="../static/assets/img/MacBook-Photoroom.png" alt="Preview"/>
    </div>
</section>
```

Рисунок 3.31 Секція About

Пояснення: цей блок розповідає про проєкт, що він створений мною як частина курсової роботи, не є фінальною версією і не призначений для публічного

використання. Під текстом розміщено зображення (макбук із сайтом). Усе оформлено в центрі і на темному фоні.

Секція Weather Forecast

Якщо шаблон передається змінна forecast, відображається 5-денний прогноз для міста city.

Прогноз виводиться у вигляді карток із днем тижня, іконкою погоди та макс/хв температурою.

Є форма пошуку із введенням міста та кнопкою «Search».

Цей блок дозволяє користувачу ввести назву міста, щоб отримати погоду. Є заголовок, іконка з лупою, поле для введення міста та кнопка "Search", яка надсилає запит методом POST на головну сторінку (/).

Блок картки Leaflet ( ) для відображення геолокації.

Блок із рекомендаціями (recommendationBox), які передаються через змінну recommendation (без екранування, з фільтром safe).

```
<!-- Weather Forecast -->
<section class="projects-section bg-light" id="projects">
    <div class="forecast-section" id="forecast">
        {% if forecast %}
        <h2>5-Day Weather Forecast for {{ city }}</h2>
        <div class="forecast-cards">
            {% for day in forecast %}
            <div class="forecast-card">
                <div>{{ day.day }}</div>
                <img src="http://openweathermap.org/img/wn/{{ day.icon }}@2x.png" alt="Weather Icon"/>
                <div>{{ day.temp_max }}°C / {{ day.temp_min }}°C</div>
            </div>
            {% endfor %}
        </div>
        {% endif %}
```

Рисунок 3.32 Секція Weather Forecast

```
<!-- City search -->
<div class="col-md-10 col-lg-8 mx-auto text-center">
    <i class="fas fa-search-location fa-2x mb-2" style="color: #6c9b9b;"></i>
    <h2 class="mb-5">Enter a city to get the weather</h2>
    <form class="form-signup" id="cityForm" method="POST" action="/">
        <div class="row input-group-newsletter">
            <div class="col">
                <input class="form-control" name="cityInput" type="text" placeholder="Enter city name..." r
            </div>
            <div class="col-auto">
                <button class="btn btn-primary" type="submit">Search</button>
            </div>
        </div>
    </form>
</div>
```

Рисунок 3.33 Секція City search

Блок Map та Recommendation

У цьому блоці: спочатку відображається карта з допомогою div з id map, розміром 400px у висоту та на всю ширину.

Нижче — блок рекомендацій, де в прямокутному блоці recommendation-box виводяться поради, які динамічно вставляються з серверу через змінну {{ recommendation|safe }} (наприклад, поради щодо погоди в місті).

```
    <!-- Map -->
    <div class="col-md-10 col-lg-8 mx-auto text-center mt-4">
        <div id="map" style="height: 400px; width: 100%;"></div>
    </div>

    <!-- Recommendation -->
    <div class="col-md-10 col-lg-8 mx-auto text-center mt-4">
        <div class="recommendation-box" id="recommendationBox">
            <p>{{ recommendation|safe }}</p>
        </div>
    </div>
</section>
```

Рисунок 3.34 Блок з картою та рекомендаціями

Секція Contact From

У цьому блоці: створено секцію підписки — користувач може ввести свою електронну адресу, щоб отримувати оновлення.

Є поле для введення email і кнопка "Notify Me!" для надсилання форми.

Додані повідомлення про успішну або невдалу відправку, які з'являються після взаємодії з формою.

```
<!-- Contact Form -->
<section class="signup-section" id="signup">
    <div class="container px-4 px-lg-5">
        <div class="row gx-4 gx-lg-5">
            <div class="col-md-10 col-lg-8 mx-auto text-center">
                <i class="far fa-paper-plane fa-2x mb-2 text-white"></i>
                <h2 class="text-white mb-5">Subscribe to receive updates!</h2>
                <form class="form-signup" id="contactForm">
                    <!-- Email address input-->
                    <div class="row input-group-newsletter">
                        <div class="col">
                            <input class="form-control" id="emailAddress" type="email"
                                   placeholder="Enter email address..." aria-label="Enter email address..."
                                   required>
                        </div>
                        <div class="col-auto">
                            <button class="btn btn-primary" id="submitButton" type="submit">Notify Me!</button>
                        </div>
                    </div>
```

Рисунок 3.35 Секція Contact From частина (1)

```
                        <div class="col-auto">
                            <button class="btn btn-primary" id="submitButton" type="submit">Notify Me!</button>
                        </div>
                    </div>
                    <!-- Feedback messages -->
                    <div class="invalid-feedback mt-2">Please enter a valid email.</div>
                    <div class="d-none" id="submitSuccessMessage">
                        <div class="text-center mb-3 mt-2 text-white">
                            <div class="fw-bolder">Form submission successful!</div>
                        </div>
                    </div>
                    <div class="d-none" id="submitErrorMessage">
                        <div class="text-center text-danger mb-3 mt-2">Error sending message!</div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>
```

Рисунок 3.36 Секція Contact From частина (2)

Блок Contact info

Цей блок показує твої контактні дані у трьох картках: адресу, email і номер телефону. Під ними — іконки соцмереж з посиланнями на твої профілі (Instagram, Facebook, GitHub). Усе оформлено стильно, на темному фоні.

```
<!-- Contact Info -->
<section class="contact-section bg-black">
    <div class="container">
        <div class="row">
            <div class="col-md-4 mb-3">
                <div class="card py-4 h-100 text-center">
                    <i class="fas fa-map-marked-alt text-primary mb-2"></i>
                    <h4 class="m-0">Address</h4>
                    <hr class="my-4 mx-auto"/>
                    <div class="small text-black-50">Office 3702 Project Development</div>
                </div>
            </div>
            <div class="col-md-4 mb-3">
                <div class="card py-4 h-100 text-center">
                    <i class="fas fa-envelope text-primary mb-2"></i>
                    <h4 class="m-0">Email</h4>
                    <hr class="my-4 mx-auto"/>
                    <div class="small text-black-50"><a href="#">nikolay.siroshtanov@gmail.com</a></div>
                </div>
            </div>
```

Рисунок 3.37 Блок Contact info частина (1)

```
            <div class="col-md-4 mb-3">
                <div class="card py-4 h-100 text-center">
                    <i class="fas fa-mobile-alt text-primary mb-2"></i>
                    <h4 class="m-0">Phone</h4>
                    <hr class="my-4 mx-auto"/>
                    <div class="small text-black-50">+38 (095) 770 79 67</div>
                </div>
            </div>
        </div>
        <div class="social d-flex justify-content-center mt-4">
            <a class="mx-2" href="https://www.instagram.com/amaterasu_60hz/"><i class="fab fa-instagram"></i></a>
            <a class="mx-2" href="https://www.facebook.com/profile.php?id=61568963828380"><i class="fab fa-facebook-f"
            <a class="mx-2" href="https://github.com/Nikolas7070"><i class="fab fa-github"></i></a>
        </div>
    </div>
</section>
```

Рисунок 3.38 Блок Contact info частина (2)

Останій завершальний блок включає:

Підвал сайту (footer) з копірайтом "WeatherByNikolas 2025" на чорному фоні;

Підключення Leaflet.js — для роботи інтерактивної карти;

Підключення твого скрипта script.js із папки static/js, де, ймовірно, міститься логіка карти, геолокації чи обробки форми.

```
<!-- Footer -->
<footer class="footer bg-black text-center text-white-50 small">
    <div class="container">Copyright &copy; WeatherByNikolas 2025</div>
</footer>

<!-- Scripts -->
<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
<script src="{{ url_for('static', filename='js/script.js') }}"></script>
</body>
</html>
```

Рисунок 3.39 Footer та Scripts

3.8 Динамічна взаємодія з картою та оновлення контенту

Нам залишилося лише створити файл «script.js» для динамічної взаємодії контенту з картою, щоб покращити нашу стилізацію.

Наступний фрагмент коду відповідає за відображення карти за допомогою бібліотеки Leaflet.js на основі координат широти (lat) та довготи (lon). Також він додає маркер з підписом міста.

```
let map;
let marker;

function initMap(lat, lon, cityName : string = '') : void { Show usages  nikolas
    if (!lat || !lon || isNaN(lat) || isNaN(lon)) {
        console.error('Invalid coordinates');
        return;
    }

    if (!map) {
        map = L.map('map').setView([lat, lon], 10);
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            attribution: '&copy; OpenStreetMap contributors'
        }).addTo(map);
    } else {
        map.setView([lat, lon], 10);
    }

    if (marker) {
        map.removeLayer(marker);
    }

    marker = L.marker([lat, lon]).addTo(map)
        .bindPopup(cityName || 'You are here')
        .openPopup();
}
```

Рисунок 3.40 Навігаційна чатина

Пояснення:

- map — змінна, в якій зберігається об'єкт карти Leaflet.
- marker — змінна для зберігання маркера на карті (точки розташування).
- функції initMap(lat, lon, cityName = ''): — Перевіряє, чи координати дійсні.
- Далі якщо карта ще не була створена (!map), вона ініціалізується з переданими координатами (lat, lon) та масштабом 10.
- Підключається шар з тайлами OpenStreetMap.
- Якщо карта вже існує — вона просто переміщується (setView) до нових координат.
- if (marker) — Якщо на карті вже є маркер — він видаляється, щоб уникнути дублювання.

Ця асинхронна функція виконує геокодування міста — тобто, на основі назви міста визначає його географічні координати (широту й довготу), використовуючи публічний API Nominatim від OpenStreetMap.

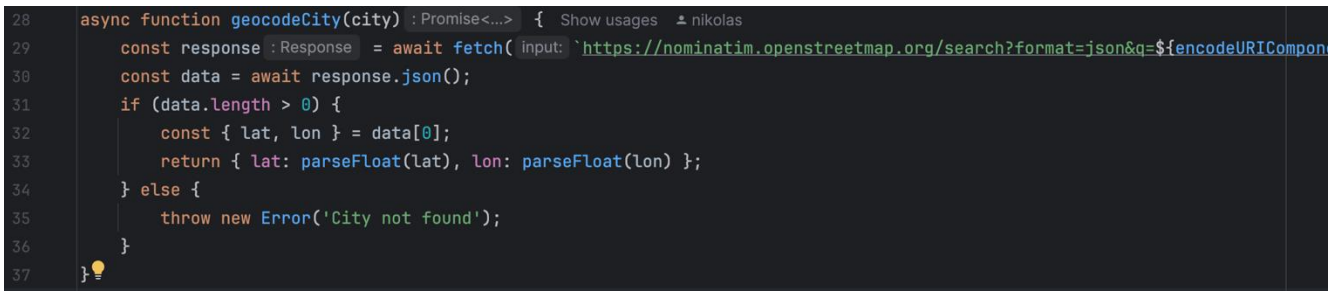


Рисунок 3.41 Асинхронна функція

Після повного завантаження структури HTML-документу (DOM) відбувається ініціалізація ключових елементів веб-платформи. За допомогою події DOMContentLoaded виконується JavaScript-код, який відповідає за обробку взаємодії користувача з формою введення міста, секцією з прогнозом погоди, блоком з рекомендаціями, а також динамічне оновлення карти. Спочатку здійснюється перевірка наявності необхідних елементів DOM (форма, блок прогнозу, блок рекомендацій). У випадку їх відсутності виводиться відповідне повідомлення в консоль, і подальше виконання припиняється. Однією з додаткових функцій є navbarShrink, яка автоматично змінює вигляд навігаційної панелі залежно від положення прокрутки: при прокручуванні сторінки додається клас navbar-shrink, що дозволяє візуально зменшити панель, а при поверненні вгору — цей клас видаляється. Таким чином, забезпечується адаптивна та зручна для користувача взаємодія з інтерфейсом. У цьому ж блоці також визначається поведінка під час завантаження сторінки: якщо браузер підтримує геолокацію, відбувається автоматичне визначення місцезнаходження користувача та відображення карти з відповідним маркером; у разі відмови або помилки – карта за замовчуванням фокусується на місті Київ. Загалом, цей фрагмент коду забезпечує початкову підготовку сторінки до роботи з користувачем, її динамічне оновлення без перезавантаження, а також створює зручний та інтерактивний досвід взаємодії з веб-платформою.

```
window.addEventListener( type: 'DOMContentLoaded', listener: function () : void {  nikolas
    const form : HTMLElement  = document.getElementById( elementId: 'cityForm');
    const forecastSection : HTMLElement  = document.getElementById( elementId: 'forecast');
    const recommendationBox : HTMLElement  = document.getElementById( elementId: 'recommendationBox');

    if (!form || !forecastSection || !recommendationBox) {
        console.error('Form, forecast section, or recommendation box not found');
        return;
    }

    // Navbar shrink function
    var navbarShrink = function () : void  {  Show usages  nikolas
        const navbarCollapsible : Element  = document.body.querySelector( selectors: '#mainNav');
        if (!navbarCollapsible) {
            return;
        }
        if (window.scrollY === 0) {
            navbarCollapsible.classList.remove( tokens: 'navbar-shrink');
        } else {
            navbarCollapsible.classList.add('navbar-shrink');
        }
    };
```

Рисунок 3.42 Обробка подій після завантаження сторінки

Наступний фрагмент коду виконує такі функції:

Виклик функції navbarShrink() при завантаженні сторінки — це забезпечує правильний початковий стан навігаційної панелі залежно від позиції прокрутки (якщо сторінка вже прокручена, панель буде зменшена).

Додавання слухача події scroll до документа — щоразу, коли користувач прокручує сторінку, викликається функція navbarShrink(), яка оновлює вигляд навбару (додає або видаляє клас navbar-shrink).

Перевірка підтримки геолокації браузером:

Якщо геолокація доступна, спробує отримати поточні координати користувача через navigator.geolocation.getCurrentPosition.

Якщо координати успішно отримані, викликається функція initMap для ініціалізації карти з цими координатами.

Якщо геолокація недоступна або користувач відмовляється, виконується fallback — ініціалізація карти з координатами Києва (50.45, 30.52) і підписом "Kyiv".

Якщо геолокація не підтримується взагалі, також ініціалізується карта з координатами Києва.

Обробка події надсилання форми submit:

Відміняється стандартна поведінка форми (e.preventDefault()), щоб уникнути перезавантаження сторінки.

Збирається значення введеного міста з форми.

Якщо поле міста порожнє або містить лише пробіли — користувачу виводиться повідомлення з проханням ввести назву міста і подальша обробка припиняється.

```
navbarShrink();

document.addEventListener( type: 'scroll', navbarShrink);

if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition( successCallback: async position : GeolocationPosition => {
        const { latitude : number , longitude : number } = position.coords;
        initMap(latitude, longitude);
    }, errorCallback: () : void => {
        // fallback: Киев
        initMap( lat: 50.45, lon: 30.52, cityName: "Kyiv");
    });
} else {
    initMap( lat: 50.45, lon: 30.52, cityName: "Kyiv");
}

form.addEventListener( type: 'submit', listener: async function (e : SubmitEvent ) : Promise<void> {
    e.preventDefault();

    const formData : FormData = new FormData(form);
    const city : FormDataEntryValue = formData.get('cityInput');

    if (!city || city.trim() === '') {
        alert('Введіть назву міста.');
        return;
    }
```

Рисунок 3.43 Обробка подій на сторінці та взаємодія з геолокацією

Цей JavaScript-фрагмент реалізує логіку обробки події надсилання форми пошуку міста. Після введення назви міста та натискання кнопки:

Надсилається POST-запит на сервер з назвою міста у тілі запиту.

Отримується HTML-відповідь із новими даними про прогноз погоди та рекомендації.

За допомогою DOMParser витягується новий вміст елементів forecast і recommendationBox, які динамічно оновлюються на сторінці без перезавантаження.

Потім викликається функція geocodeCity(city), яка визначає координати введеного міста через API Nominatim.

Визначені координати використовуються для переміщення карти до нового міста та відображення відповідного маркера.

Форма очищається для нового введення.

```
try {
    const response : Response = await fetch( input: '/', init: {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
        },
        body: new URLSearchParams( init: { cityInput: city })
    });

    if (!response.ok) throw new Error('Ошибка при загрузке данных');

    const html : string = await response.text();
    const parser : DOMParser = new DOMParser();
    const doc : Document = parser.parseFromString(html, type: 'text/html');

    const newForecast : HTMLElement = doc.getElementById( elementId: 'forecast');
    if (newForecast && forecastSection) {
        forecastSection.innerHTML = newForecast.innerHTML;
    }

    const newRecommendation : HTMLElement = doc.getElementById( elementId: 'recommendationBox');
    if (newRecommendation && recommendationBox) {
        recommendationBox.innerHTML = newRecommendation.innerHTML;
    }

    const { lat : number , lon : number } = await geocodeCity(city);
    initMap(lat, lon, city);
    form.reset();
```

Рисунок 3.44 Обробка форми пошуку та оновлення мапи й прогнозу погоди

Остання частина відповідає за виявлення й обробку ситуацій, коли запит до сервера не вдається або користувач ввів некоректне місто. Вона забезпечує інформування користувача про проблему та допомагає уникнути критичних збоїв у роботі веб-платформи.

```
118             } catch (error) {
119                 console.error('Ошибка:', error);
120                 alert('Місто не знайдено або сталася помилка при завантаженні даних.');
121             }
122         });
123     });
```

Рисунок 3.45 Обробка помилок при завантаженні даних

Останнім кроком є створення CSS-файлу. Оскільки мій CSS-файл містить понад 13 тисяч рядків, він займає багато місця. Тому я виклав свій проєкт на свій GitHub https://github.com/Nikolas7070 , щоб ви могли детально ознайомитися з CSS-файлом і з цілим проєктом.

3.9 Розміщення веб-платформи на сервері

Останнім кроком буде публікація веб-платформи на хостингу. Оскільки багато потужних хостингів є платними, я обрав безкоштовний хостинг, який підходить для демонстрації моїх навичок та для публікації веб-платформи у вільному доступі. Для цього я обрав сайт https://www.pythonanywhere.com.

PythonAnywhere — це хмарна платформа, яка дозволяє запускати, розгортати та розробляти Python-додатки прямо з веб-браузера. Вона особливо зручна для веб-проєктів на Flask або Django. Не потребує встановлення серверного ПЗ на ПК.

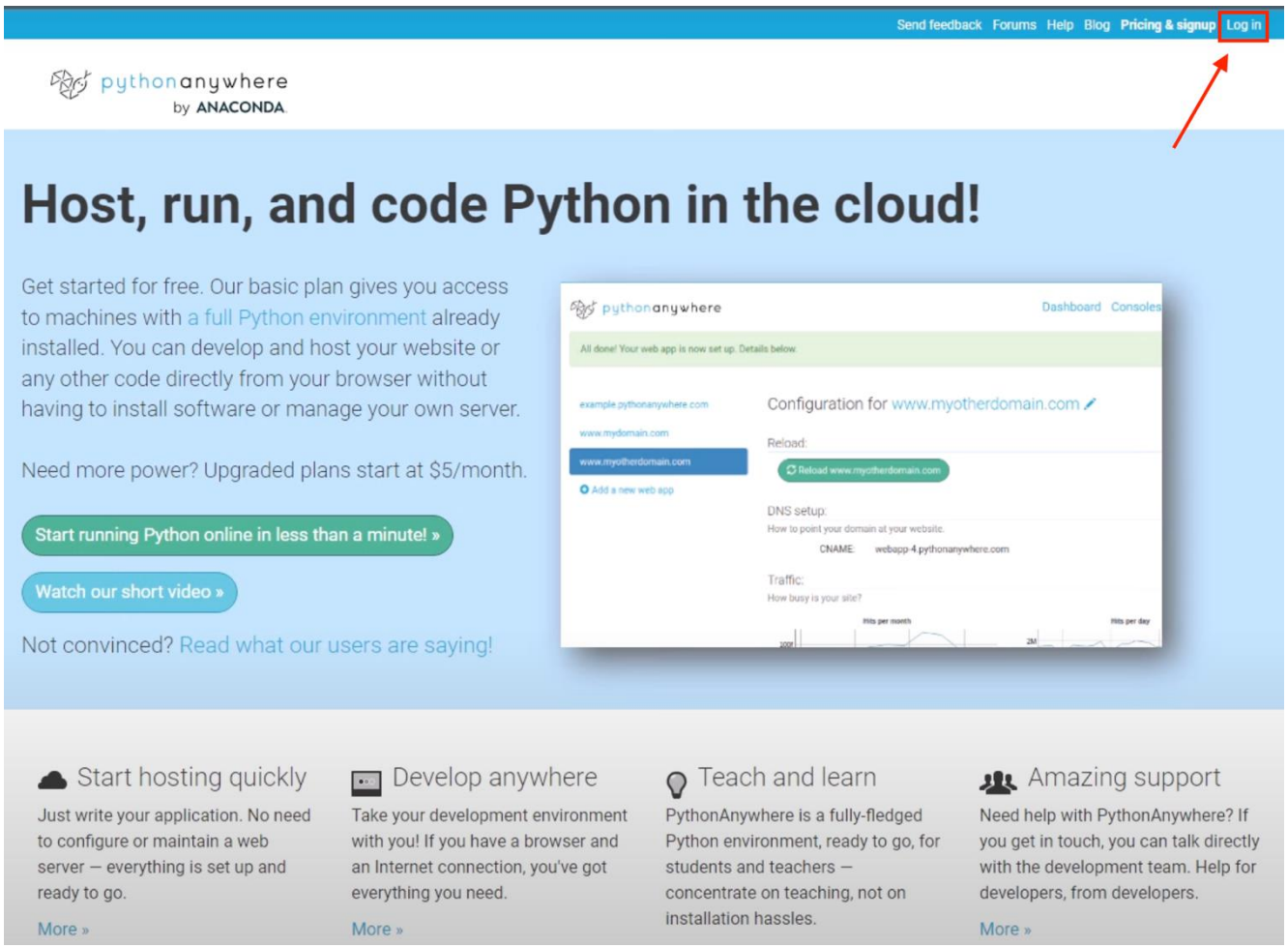


Рисунок 3.46 Реєстрація на PythonAnywhere

Далі нам треба перейти на вкладку WEB та натиснути «Add a new web app» (Рисунок 3.47 – 3.48)

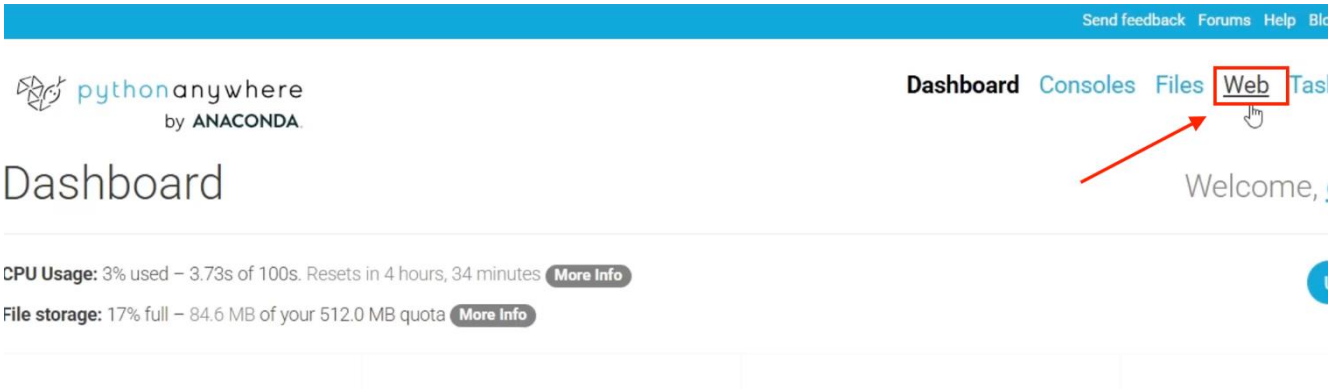


Рисунок 3.47 Вкладка web

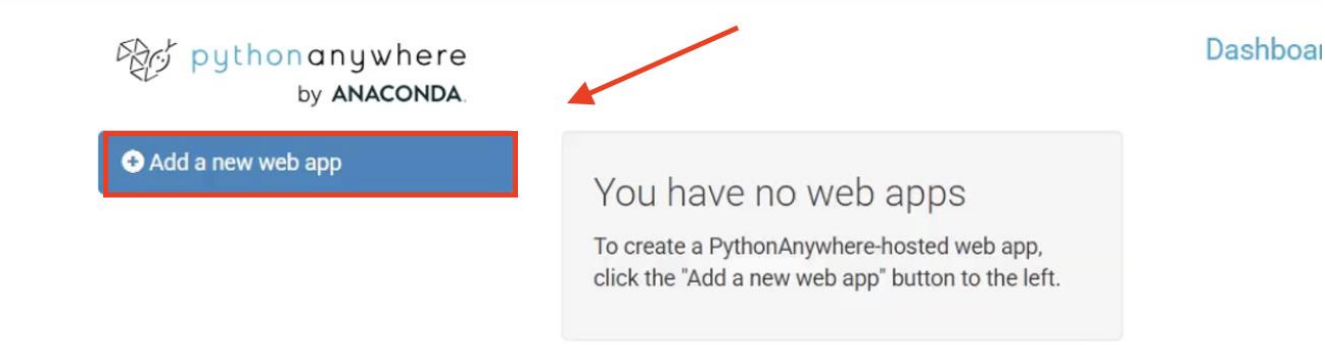


Рисунок 3.48 Кнопка створення нового проєкту

Далі нам потрібно обрати фреймворк, з яким ми працюватимемо. У нашому випадку — це Flask. (Рисунок 3.49)

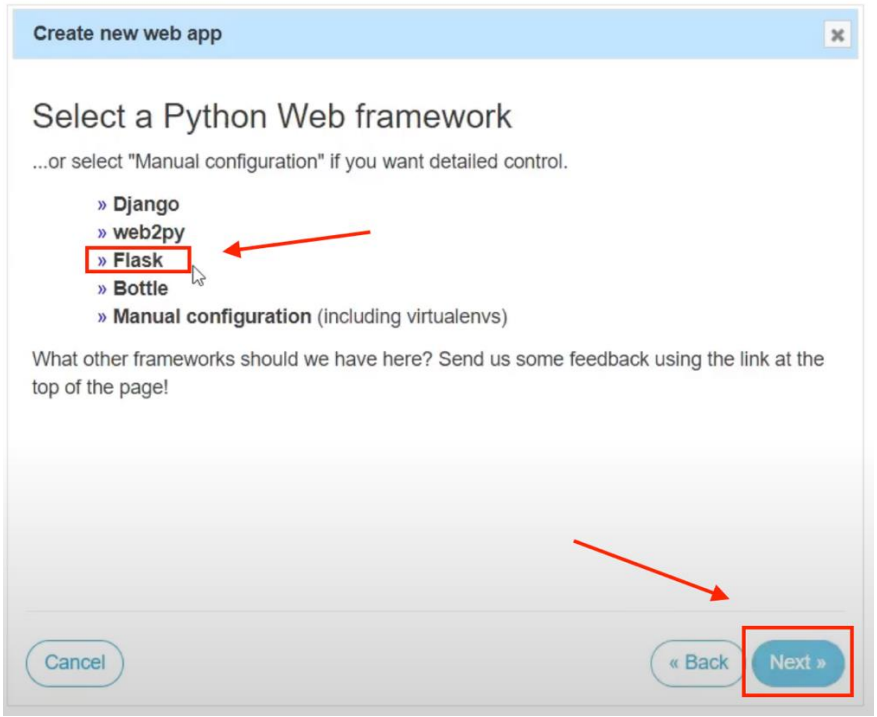


Рисунок 3.49 Обирання фрейм ворка для хостінгу

Наступним кроком обрати версію Python та Flask нашому випадку — це Python 3.10. та Flask 2.1.2

Рисунок 3.50 Обирання версії Python для хостінгу

Після реєстрації ви зможете побачити власне посилання на ваш сайт.

Зверніть увагу на пункт Reload! Усі зміни, які ви внесли в налаштування, обов'язково потрібно підтвердити, натиснувши «Reload», щоб вони набули чинності.

Також зверніть увагу на кнопку «Run until 3 months from today» — її потрібно натискати кожні 3 місяці, щоб подовжувати термін дії хостингу (Рисунок 3.51) .

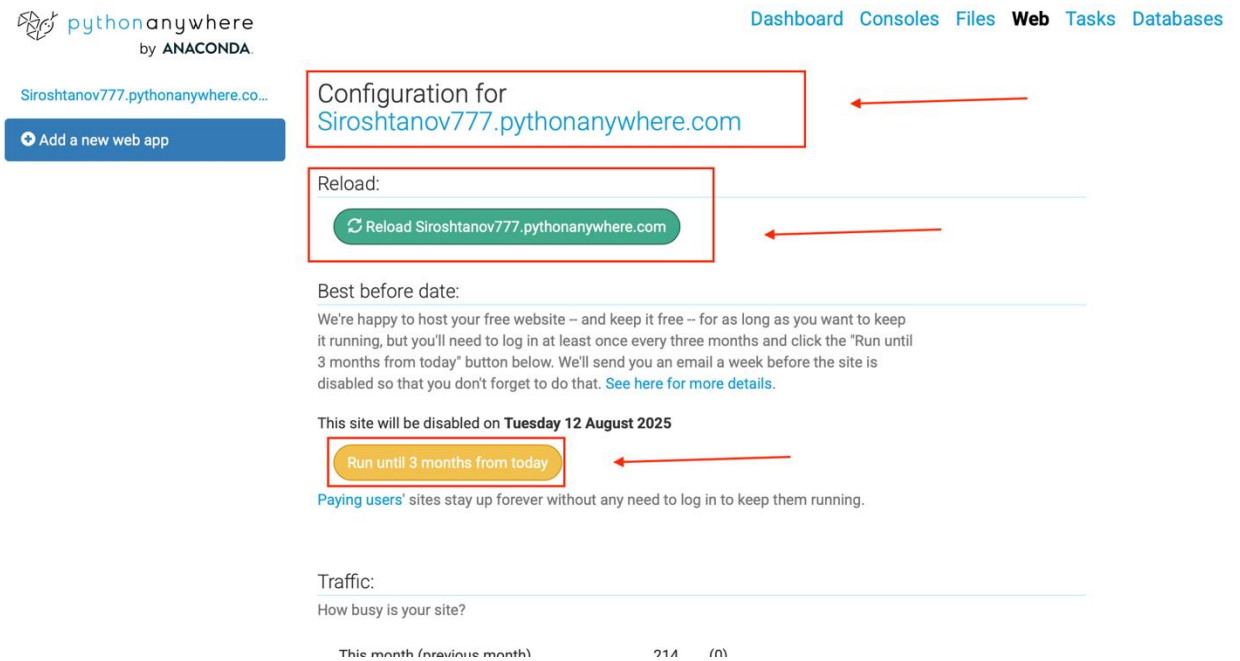


Рисунок 3.51 Головна сторінка хостінгу

Наступним кроком є завантаження нашого проєкту. Для цього потрібно перейти до папки з проєктом і заархівувати її. Після цього натискаємо «Upload a file» та обираємо створений архів (Рисунок 3.52)

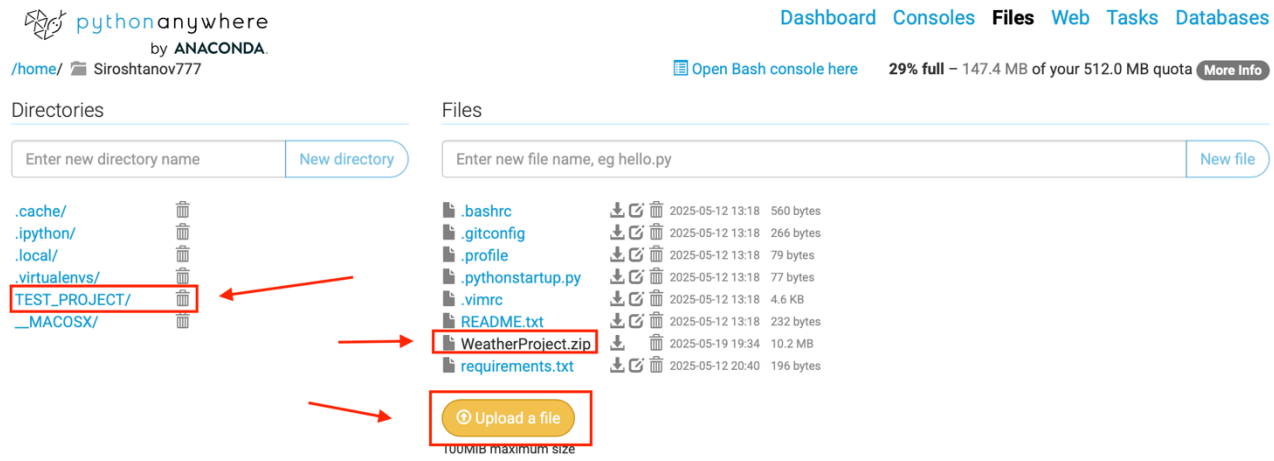


Рисунок 3.52 Завантаження проєкту на хостінг

Після того, як ми завантажили архів з нашим проєктом, спочатку папка з файлами проєкту не з’явиться автоматично.

У моєму випадку папка проєкту має назву «TEST_PROJECT».

Щоб розархівувати проєкт, потрібно перейти на вкладку «Consoles» та створити консоль натиснувши кнопку «Bash», за допомогою якої ми зможемо розпакувати архів (Рисунок 3.53).

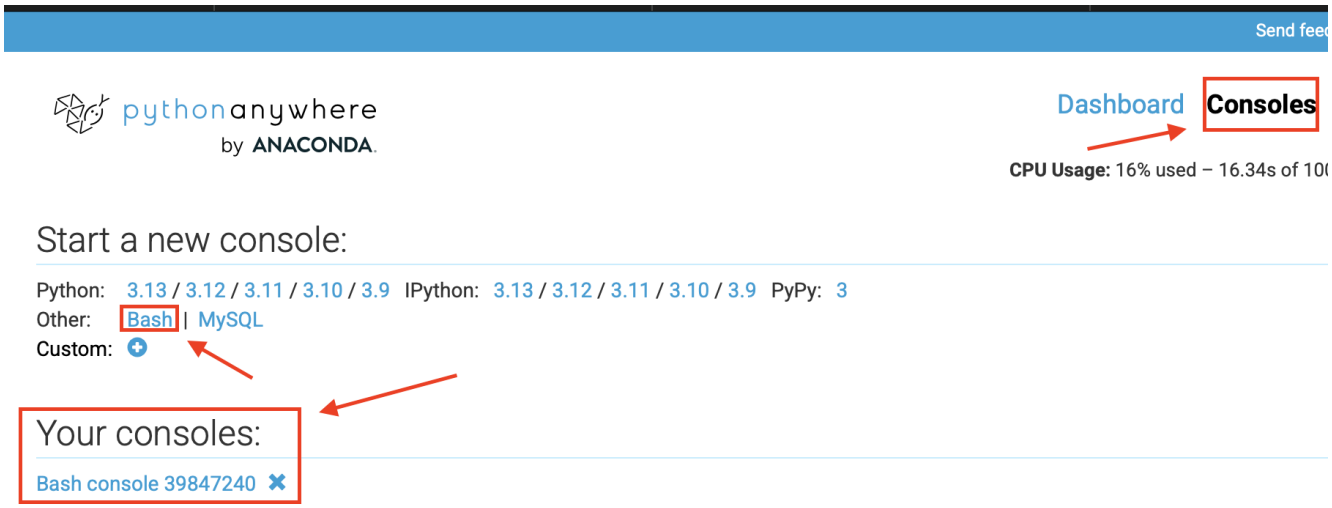


Рисунок 3.53 Створення консолі

Після цього нам потрібно прописати «ls» для того щоб точно побачити чи проєкт загрузився а в же потім написати «unzip НАЗВА_ВАШОГО_ПРОЄКТУ»

p

Рисунок 3.54 Розархівування проєкту

Зверніть увагу, що якщо ваш проєкт використовує додаткові бібліотеки, їх потрібно встановити окремо.

Для цього найкраще мати у проєкті файл з назвою «requirements.txt», в якому перелічені всі необхідні залежності. Для цього використовуйте команду

pip install -r requirements.txt

Коли ми вже розархівували проєкт, все, що нам залишилося — це правильно вказати шлях до нього.

Для цього перейдіть у вкладку «Web», прогорніть донизу й у пункті «Code» вкажіть шлях до вашого проєкту.

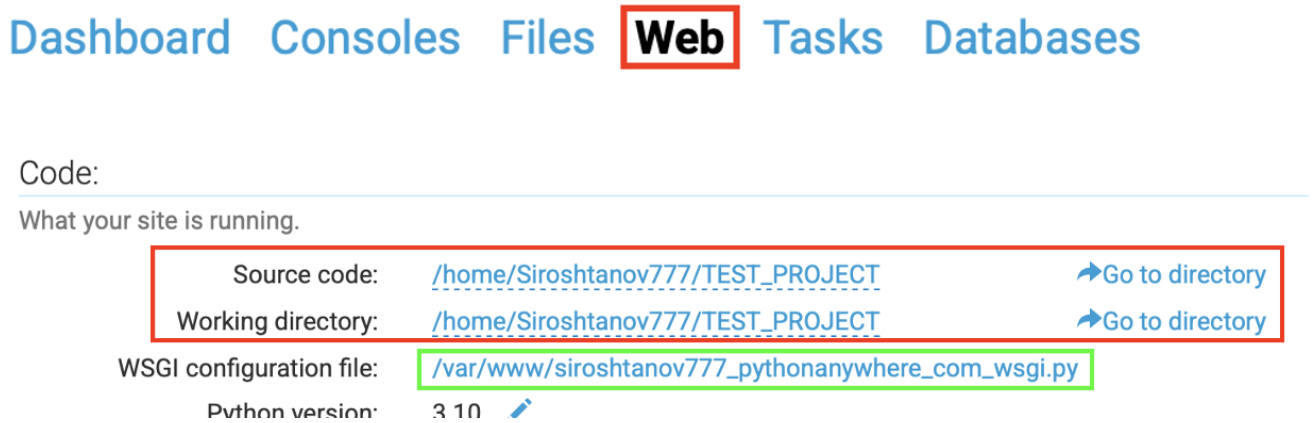


Рисунок 3.55 Налаштування шляху до проєкта

Він має бути налаштований правильно — це залежить від структури вашого сайту та того, як ви організували файли «__init__.py» і «routes.py.» У моєму випадку файл wsgi.py виглядає так (див. Рисунок 3.56).

```
1  import sys
2  import os
3
4  project_home = '/home/Siroshtanov777/TEST_PROJECT'
5  if project_home not in sys.path:
6      sys.path.append(project_home)
7
8  from app import create_app
9
10 application = create_app()
```

Рисунок 3.56 Файл wsgi.py

Пояснення:

WSGI (Web Server Gateway Interface) – це стандартний інтерфейс між веб-сервером та Python-додатком (наприклад, Flask, Django та ін.).

Файл wsgi.py містить точку входу для WSGI-сервера, щоб він знав, як запустити вашу програму.

4 ДЕМОНСТРАЦІЯ ГОТОВОЇ ВЕБ-ПЛАТФОРМИ

Для того щоб переглянути наш проєкт, переходимо за посиланням, яке надав хостинг: https://siroshtanov777.pythonanywhere.com (рисунок 4.1).

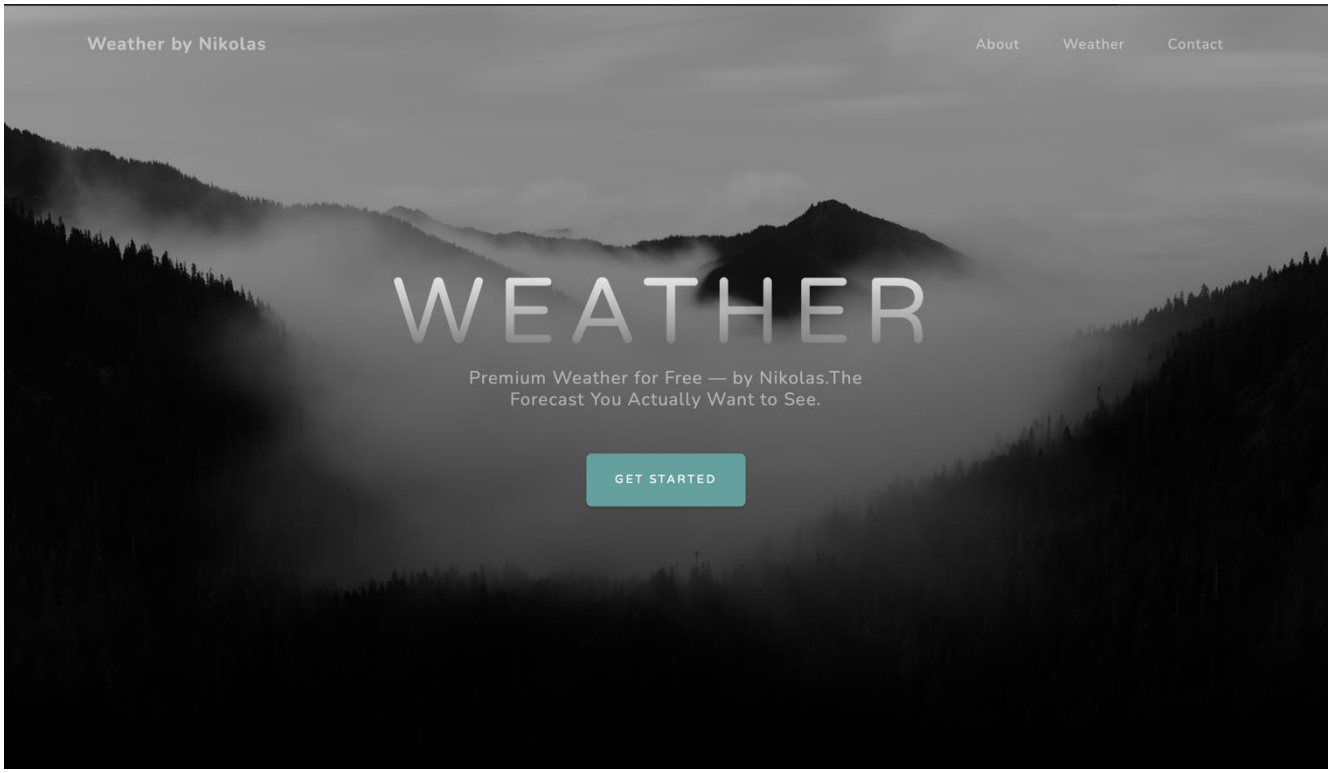


Рисунок 4.1 Головна сторінка

Нас зустрічає головна сторінка нашого погодного вебсайту, де розміщені такі вкладки, як About, Weather, Contact, а також кнопка "GET STARTED".

Далі в вкладці «About» написано: веб-платформу створено мною як частину університетської курсової роботи для демонстрації індивідуальних навичок і творчого потенціалу. Ця версія не є остаточною та не призначена для публічного або комерційного використання. Копіювання та розповсюдження заборонені.

Хочу підкреслити, що дана платформа повністю готова для демонстрації моїх навичок у межах дипломної роботи, однак не є остаточною версією сайту для загального користування (Рисунок4.2).

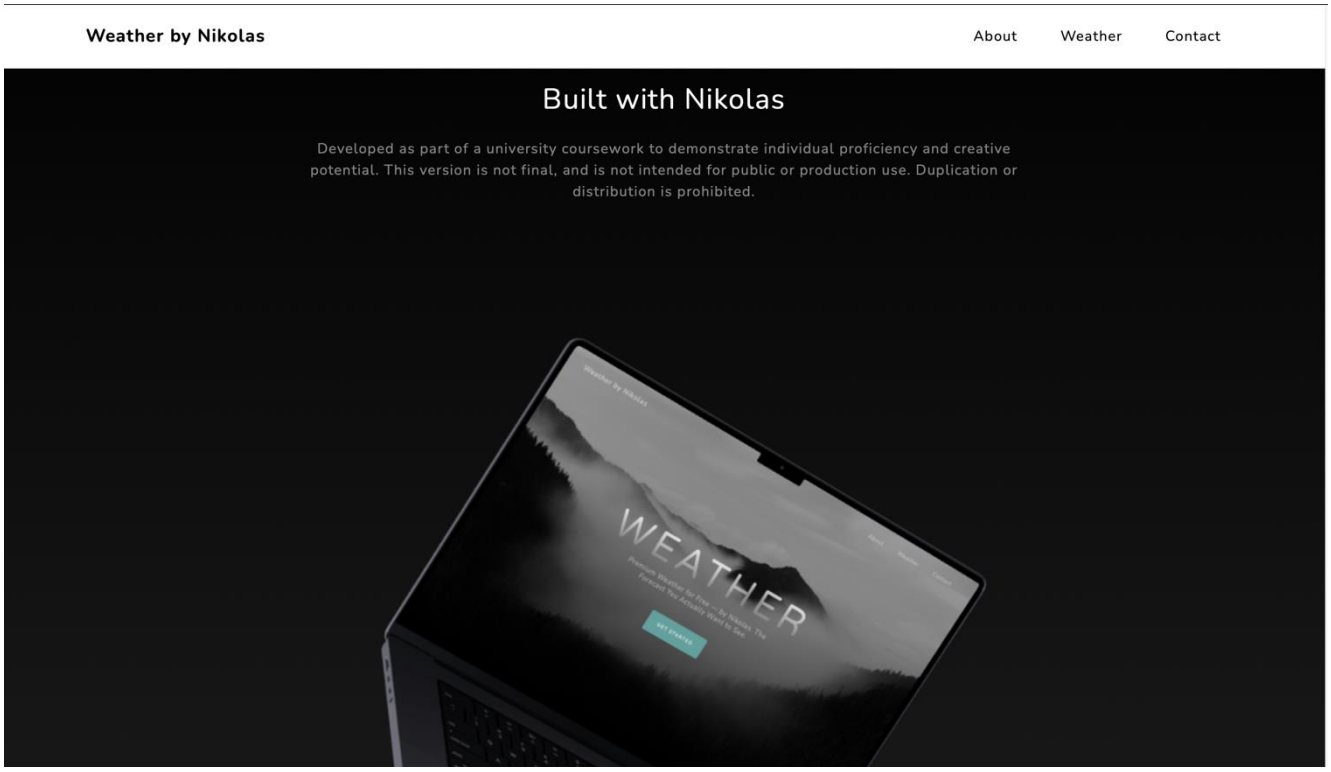


Рисунок 4.2 Вкладка «About»

Наступна вкладка, і найголовніша — це Weather, де буде відображено поточну погоду та відповідні рекомендації.

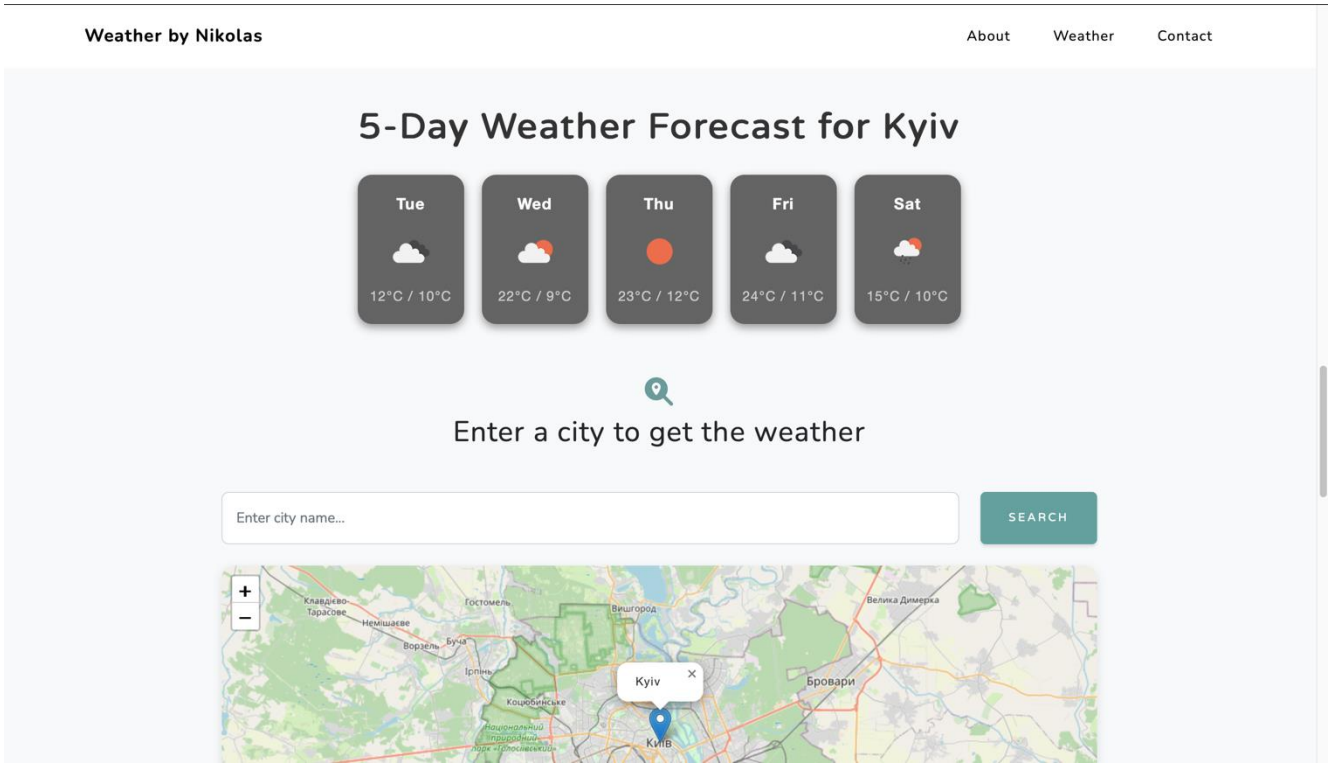


Рисунок 4.3 Вкладка «Weather» частина (1)

Вкладка Weather показує прогноз погоди на 5 днів, карту з розташуванням обраного міста, а також рекомендації щодо того, як краще одягнутися та чого уникати.

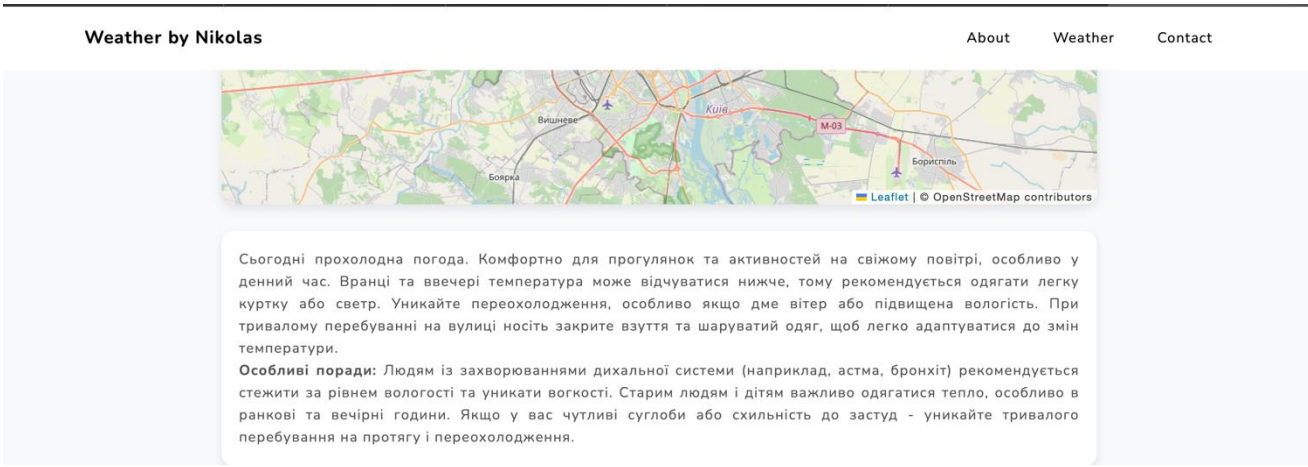


Рисунок 4.4 Владка «Weather» частина (2)

У розділі «Особливі поради» наведено спеціалізовані рекомендації, які призначені насамперед для людей із хронічними захворюваннями. Ці поради допомагають уникати потенційних ризиків, пов’язаних із погодними умовами, і підтримувати здоров’я у складних кліматичних ситуаціях. Таким чином, цей розділ створено для того, щоб покращити самопочуття і безпеку вразливих категорій користувачів.

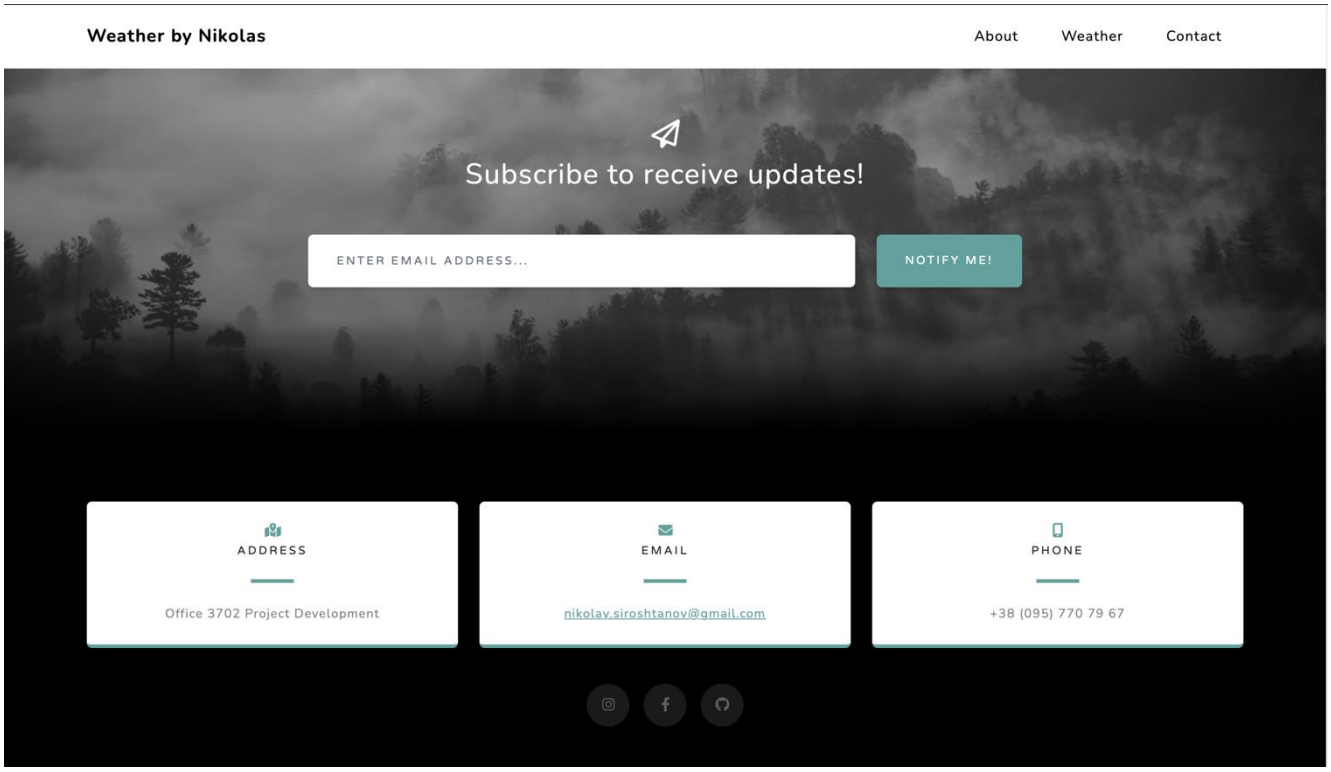


Рисунок 4.5 Соцмережі та контакти

У нижній частині сторінки користувачі можуть залишити свій email, щоб отримувати сповіщення про всі оновлення сайту. Також я додав свої персональні соцмережі — Facebook, Instagram та GitHub — щоб люди, як із хронічними захворюваннями, так і звичайні користувачі, могли надсилати свої поради щодо рекомендацій. Це допоможе зібрати більше корисної інформації про те, як люди почуваються за різної погоди, якими є їхні відчуття та досвід. Завдяки цьому я зможу аналізувати зібрані дані, удосконалювати рекомендації і робити платформу ще кориснішою для всіх користувачів.

ВИСНОВОК

У ході виконання дипломної роботи мною було розроблена модель веб-платформи, яка дозволяє поєднати метеорологічні дані з персоналізованими рекомендаціями щодо охорони здоров’я. Актуальність даної теми зумовлена погіршенням кліматичних умов, зростанням кількості метеозалежних людей, алергіків та осіб із хронічними захворюваннями, для яких погодні зміни можуть становити суттєвий ризик.

На відміну від більшості популярних погодних сервісів, які обмежуються стандартним набором даниху розробленій мною платформі додано блок із персоналізованими рекомендаціями, яких зазвичай не зустрінеш в аналогічних сервісах. Ці рекомендації враховують не лише погодні умови, а й можливий вплив кліматичних факторів на самопочуття людини, зокрема для метеозалежних осіб, алергіків та людей із хронічними захворюваннями. Такий підхід дозволяє користувачу не просто переглядати прогноз, а й отримувати практичні поради щодо адаптації до погодних змін — наприклад, обмежити перебування на вулиці в години найбільшої спеки, уникати фізичного навантаження при підвищеній вологості чи вдягатися відповідно до коливань температури.

У межах проєкту було розроблено повноцінну веб-платформу з використанням Python і Flask, що здійснює запити до OpenWeather API для отримання актуальної погоди. Система аналізує ці дані, визначає ключові параметри, а далі — звертається до локальної бази рекомендацій, сформованих відповідно до температурного режиму. У результаті користувач отримує не лише прогноз, а й відповідні поради щодо поведінки в тій чи іншій ситуації. Додатково було реалізовано інтуїтивно зрозумілий інтерфейс, що дозволяє легко взаємодіяти з платформою навіть користувачам без технічної підготовки.

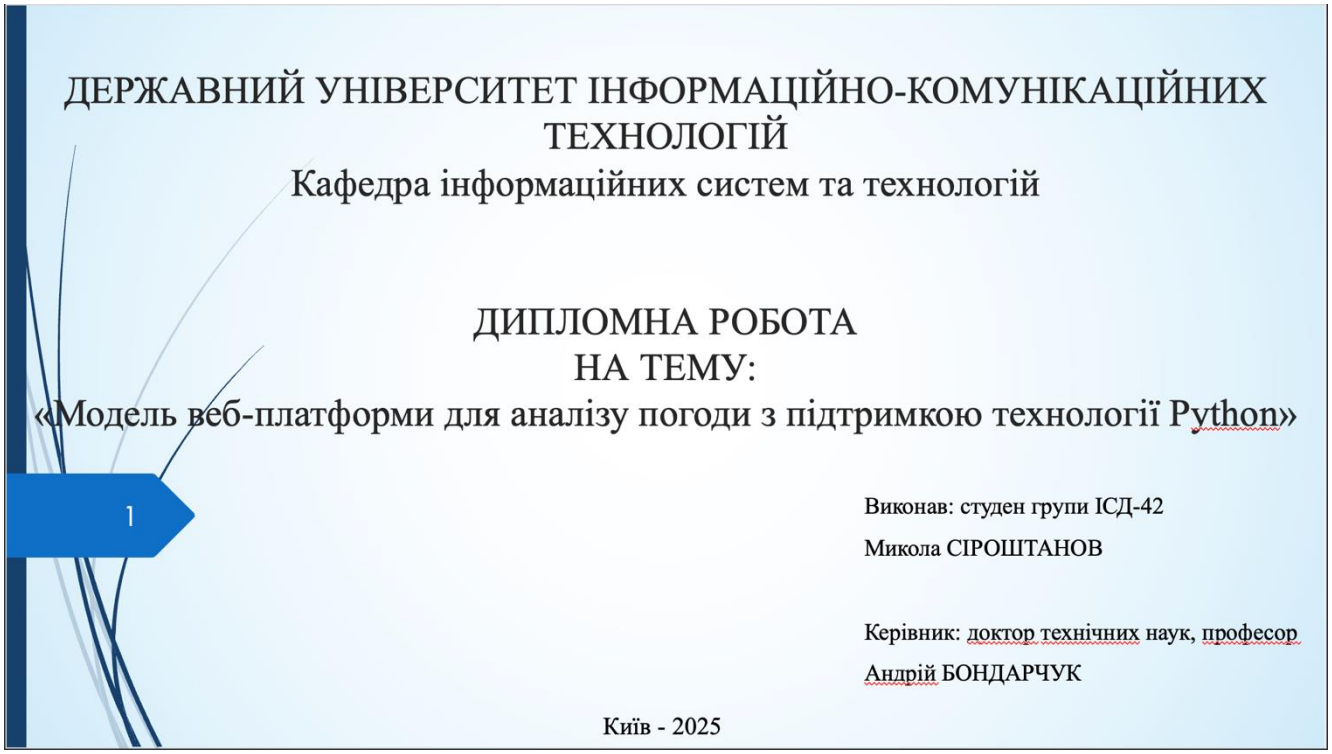
Таким чином, створена платформа поєднує елементи метеорології, медицини та веб-технологій і є прикладом персоналізованого цифрового сервісу, орієнтованого на покращення якості життя.

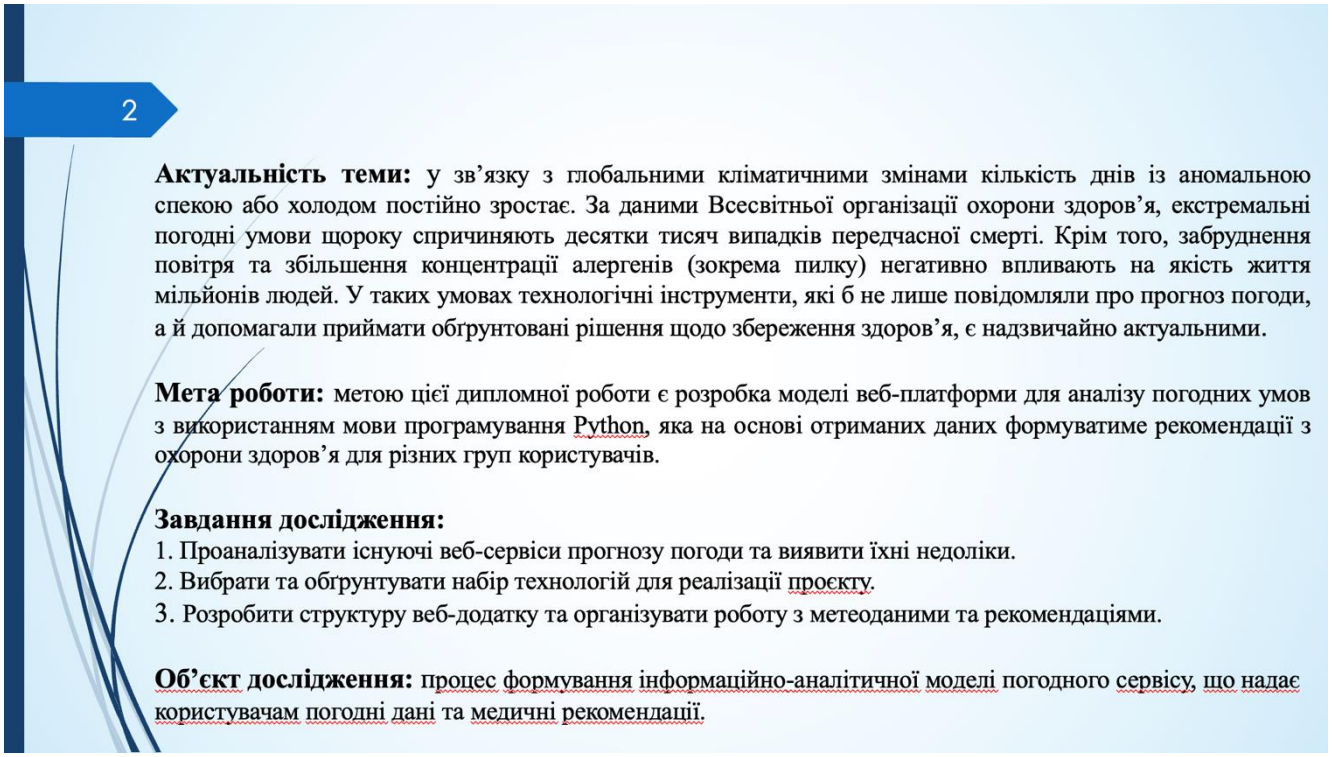
ПЕРЕЛІК ПОСИЛАНЬ

1. OpenWeather. Weather API. – https://openweathermap.org/api
2. World Health Organization. Climate change and health. – https://www.who.int/news-room/fact-sheets/detail/climate-change-and-health
3. National Oceanic and Atmospheric Administration (NOAA). – https://www.noaa.gov/
4. European Environment Agency. Air quality and health. – https://www.eea.europa.eu/themes/air/health-impacts-of-air-pollution
5. AccuWeather API. – https://developer.accuweather.com/
6. Weather.com (The Weather Channel). – https://weather.com/
7. Leaflet.js – JavaScript library for interactive maps. – https://leafletjs.com/
8. Flask documentation (Python web framework). – https://flask.palletsprojects.com/
9. Python official website. – https://www.python.org/
10. SQLite documentation. – https://sqlite.org/docs.html
11. GitHub – Hosting for version control and collaboration. – https://github.com/
12. Pandas – Python data analysis library. – https://pandas.pydata.org/
13. What is Django Web Framework? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-django-web-framework/ (date of access: 30.05.2024).
14. Scikit-learn – Machine learning in Python. – https://scikit-learn.org/

15. IPCC – Intergovernmental Panel on Climate Change. – https://www.ipcc.ch/
16. European Centre for Medium-Range Weather Forecasts (ECMWF). – https://www.ecmwf.int/
17. IBM Weather Company API – https://www.ibm.com/weather
18. Centers for Disease Control and Prevention (CDC). Climate Effects on Health. – https://www.cdc.gov/climateandhealth/
19. Meteoblue Weather API – https://content.meteoblue.com/en/business-solutions/weather-apis
20. AQICN (World Air Quality Index Project) – https://aqicn.org/api/
21. Google Fonts – https://fonts.google.com/
22. Font Awesome – Icon toolkit. – https://fontawesome.com/
23. Geonames API (for location data). – https://www.geonames.org/export/
24. European Respiratory Society – Air pollution and lung health. – https://www.ersnet.org/publications/factsheets/air-pollution/
25. PythonAnyWhere – (https://help.pythonanywhere.com)

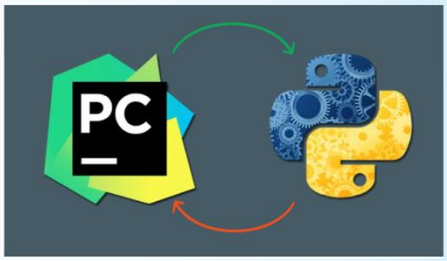
ПРЕЗИНТАЦІЯ





3

PyCharm — це інтегроване середовище розробки (IDE), спеціально створене для програмування на мові Python. Воно надає інструменти для написання, тестування, відлагодження й запуску коду.



Переваги PyCharm:

- Автодоповнення коду
- Вбудований налагоджувач (debugger)
- Підтримка фреймворків (Flask, Django)
- Інтеграція з Git
- Робота з базами даних

Python — це високорівнева, інтерпретована мова програмування загального призначення. Вона відома своєю простотою, читабельністю коду та широким застосуванням в веб-розробці.
Також вона є ідеальним вибором для створення сучасних та масштабованих веб-додатків, що підходить для моєї курсової роботи.

4

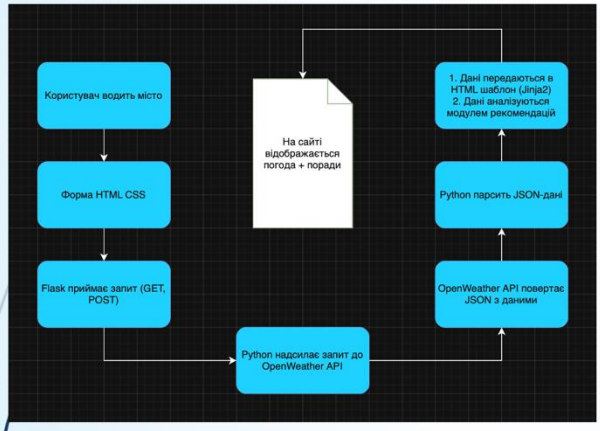


Схема роботи веб-платформи

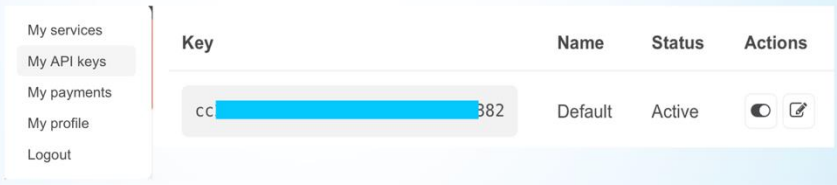
Інструменти для створення веб-платформи та їх призначення:

1. Python – відповідає за серверну частину (логіку).
2. Flask – є головним фреймворком проєкта.
3. HTML & CSS – відповідає за стилізацію та дизайн веб-платформи.
4. JavaScript – використаний для динамічних елементів на веб-платформі.
5. SqlLite – забезпечую зберігання рекомендацій в базі даних та додаємможливість їх розширювати

5

Використання технології API

OpenWeather API — це зовнішній програмний інтерфейс (Application Programming Interface), який надає доступ до різноманітних метеорологічних даних у режимі реального часу. Даний сервіс дозволяє отримувати не лише поточну погоду, а й прогноз на декілька днів.



Приклад API ключа для використання запиту погоди

6

Технології фронтенду: HTML, CSS, JavaScript

- **HTML (HyperText Markup Language)**
- HTML відповідає за **структуру сторінки**.
  Він задає, які елементи будуть на сайті: заголовки, параграфи, кнопки, форми, блоки з прогнозом тощо.
- **CSS (Cascading Style Sheets)**
- CSS — це **мова стилів**, яка дозволяє змінювати вигляд елементів: кольори, шрифти, розміри, відступи, анімації.
  Завдяки CSS мій сайт виглядає сучасно й адаптивно.
- **JavaScript**
- JavaScript додає **інтерактивність** — наприклад, завантаження карти, обробка введення міста, а також динамічне оновлення блоків із погодою та рекомендаціями без перезавантаження сторінки.

```
<div class="col-md-10 col-lg-8 mx-auto text-center mt-4">
    <div class="recommendation-box" id="recommendationBox">
        <p>{{ recommendation|safe }}</p>
    </div>
</div>
```

Приклад коду HTML

```
.recommendation-box h3 {
    font-size: 22px;
    color: #2c3e50;
    margin-bottom: 15px;
}
```

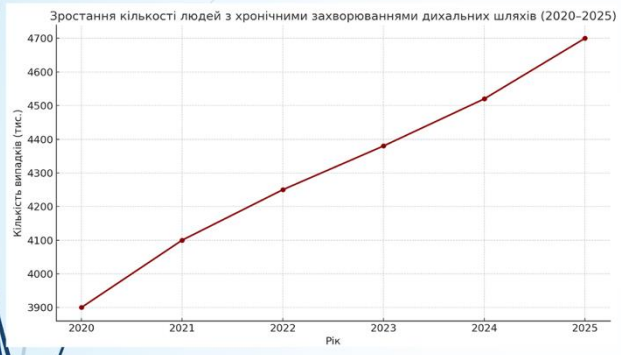
Приклад коду CSS

```
const { lat : number , lon : number } = await geocodeCity(city);
initMap(lat, lon, city);
form.reset();
```

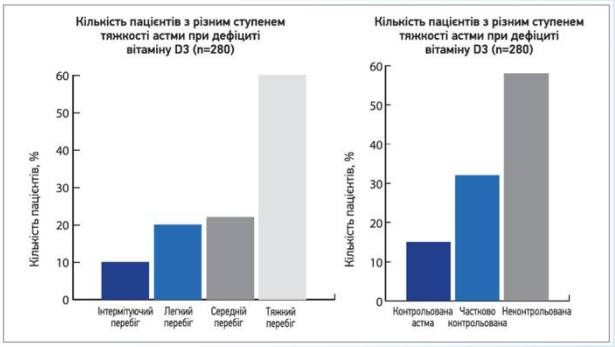
Приклад коду JavaScript

7

Статистка людей з хронічними захворюваннями



Графік людей з хронічними захворюваннями за останні 5 років



Статистиа порівняльного дослідження K. Stephanie

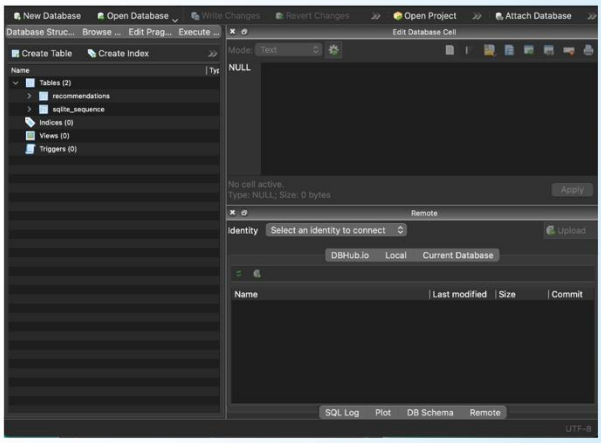
8

База даних SQLlite

SQLite — це легка, вбудована система керування базами даних (СКБД), яка не потребує окремого сервера.

Переваги SQLite:

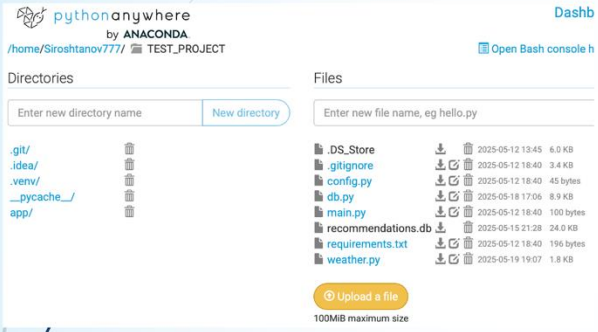
- Без серверу - не потрібно встановлювати або запускати окремий сервер (як MySQL/PostgreSQL).
- Швидкість - дуже швидка для невеликих або локальних проєктів.
- Простота використання - підключення в Python через sqlite3 — одна команда.
- Легка вага - уся база є одним файлом, якій легко копіюється або переноситься.
- І деально для навчання - Чудово підходить для невеликих застосунків, тестів і прототипів.



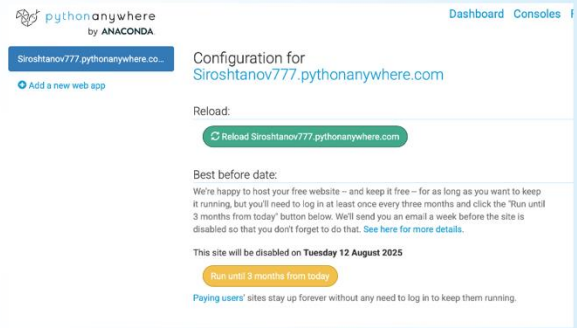
Вигляд SQLlite бази

9

Хостін Pythonanywhere



Приклад структури на хостінгу



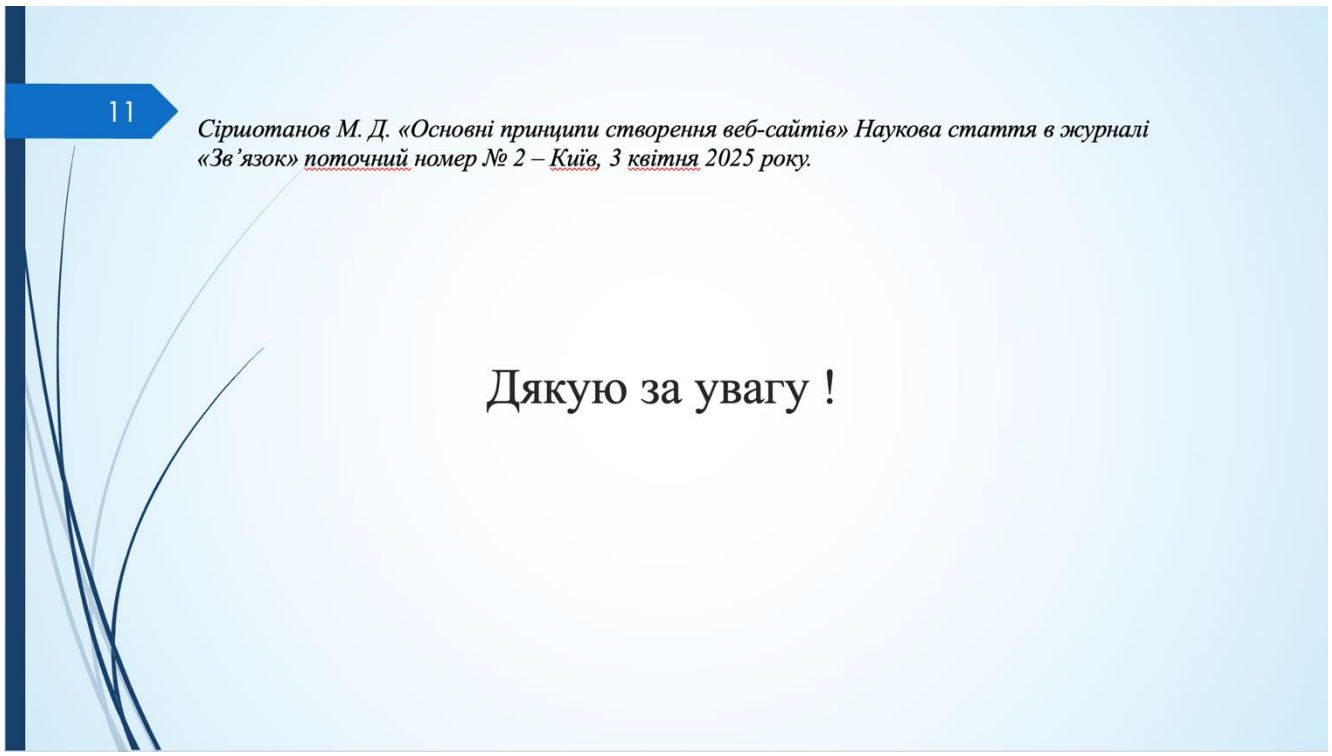
Посилання на веб-платворму через pythonanywhere

10

ВИСНОВКИ

У ході виконання дипломної роботи мною було розроблена модель веб-платформи, яка дозволяє поєднати метеорологічні дані з персоналізованими рекомендаціями щодо охорони здоров'я. Актуальність даної теми зумовлена погіршенням кліматичних умов, зростанням кількості метеозалежних людей, алергіків та осіб із хронічними захворюваннями, для яких погодні зміни можуть становити суттєвий ризик.

На відміну від більшості популярних погодних сервісів, які обмежуються стандартним набором даниху розробленій мною платформі додано блок із персоналізованими рекомендаціями, яких зазвичай не зустрінеш в аналогічних сервісах. Ці рекомендації враховують не лише погодні умови, а й можливий вплив кліматичних факторів на самопочуття людини, зокрема для метеозалежних осіб, алергіків та людей із хронічними захворюваннями. Такий підхід дозволяє користувачу не просто переглядати прогноз, а й отримувати практичні поради щодо адаптації до погодних змін — наприклад, обмежити перебування на вулиці в години найбільшої спеки, уникати фізичного навантаження при підвищеній вологості чи вдягатися відповідно до коливань температури.


11
Сіршотанов М. Д. «Основні принципи створення веб-сайтів» Наукова стаття в журналі «Зв'язок» поточний номер № 2 – Київ, 3 квітня 2025 року.
Дякую за увагу !