ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

StrikePlagiarism.com

Дата звіту 6/9/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics

Заголовок
2025_Б_ПІ_ПЗПІ-21-6_Замковий_А_Г_ скорочений

Автор / Науковий керівник / Експерт
Замковий Анатолій ГеннадійовичЄвген Кардаш

підрозділ
каф. ПІ

Обсяг знайдених подібностей

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

4.91% (4.91%) КП 1

4.03% (4.03%) КЦ

25	7229	54705
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

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

Заміна букв	ß	2
Інтервали	A→	0
Мікропробіли	0	0
Білі знаки	▯	0
Парафрази (SmartMarks)	a	30

Подібності за списком джерел

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

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://dev.to/lambkpl/setup-fastapi-and-sqlalchemy-mysql-48m9	26 0.36 %
2	https://openarchive.nure.ua/server/api/core/bitstreams/e84c9972-be3f-41bb-a499-69f315a08e4c/content	25 0.35 %
3	2024_Б_ШІ_ІТШІ_20_5_Падалкін_К_Д_записка [illegible] 12/15/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	20 0.28 %
4	https://www.backendmesh.com/strategy-design-pattern/	18 0.25 %

Рисунок А.1 – Результат перевірки на плагіат

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

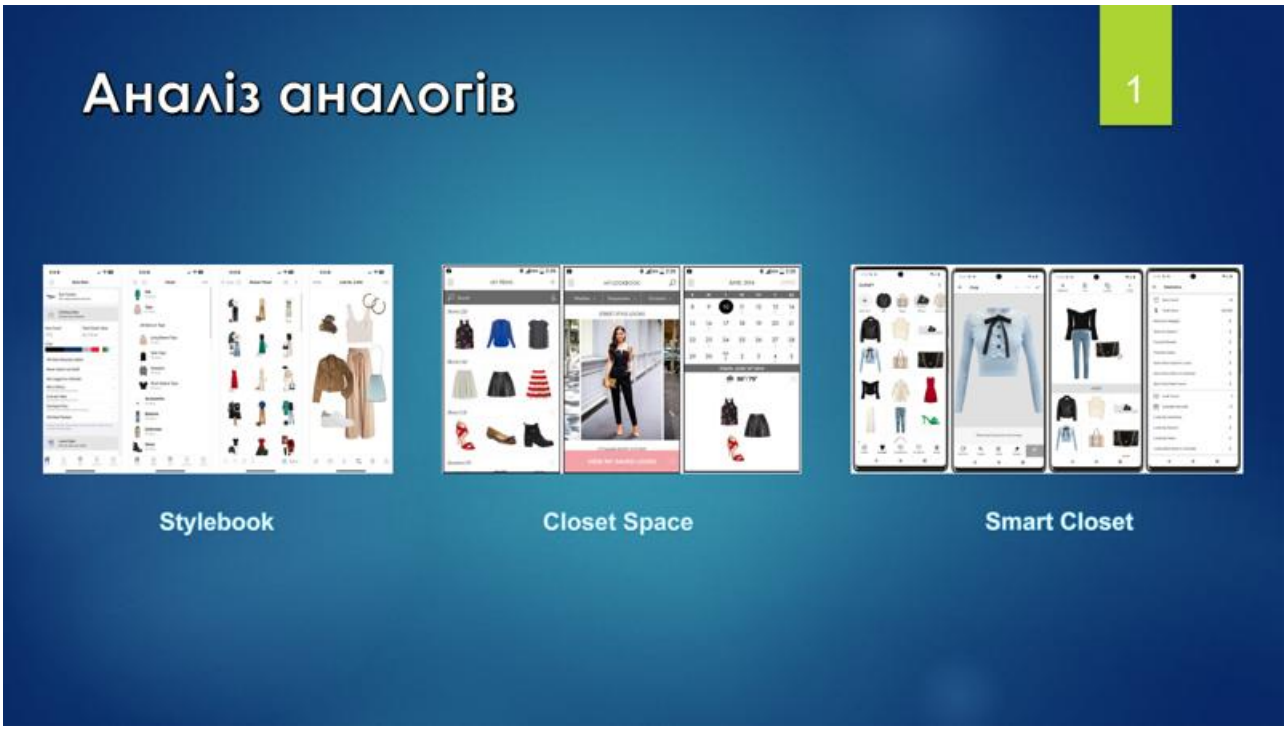

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

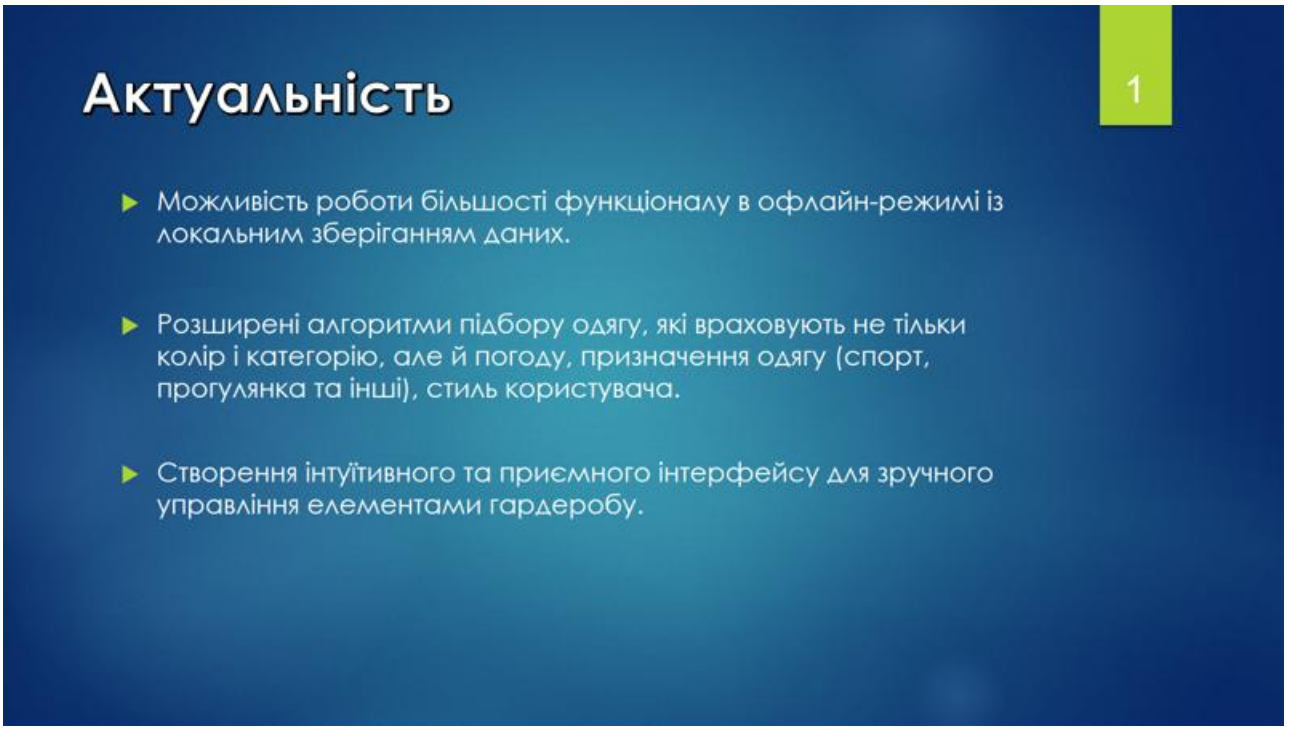

Рисунок Б.5 – Слайд 5

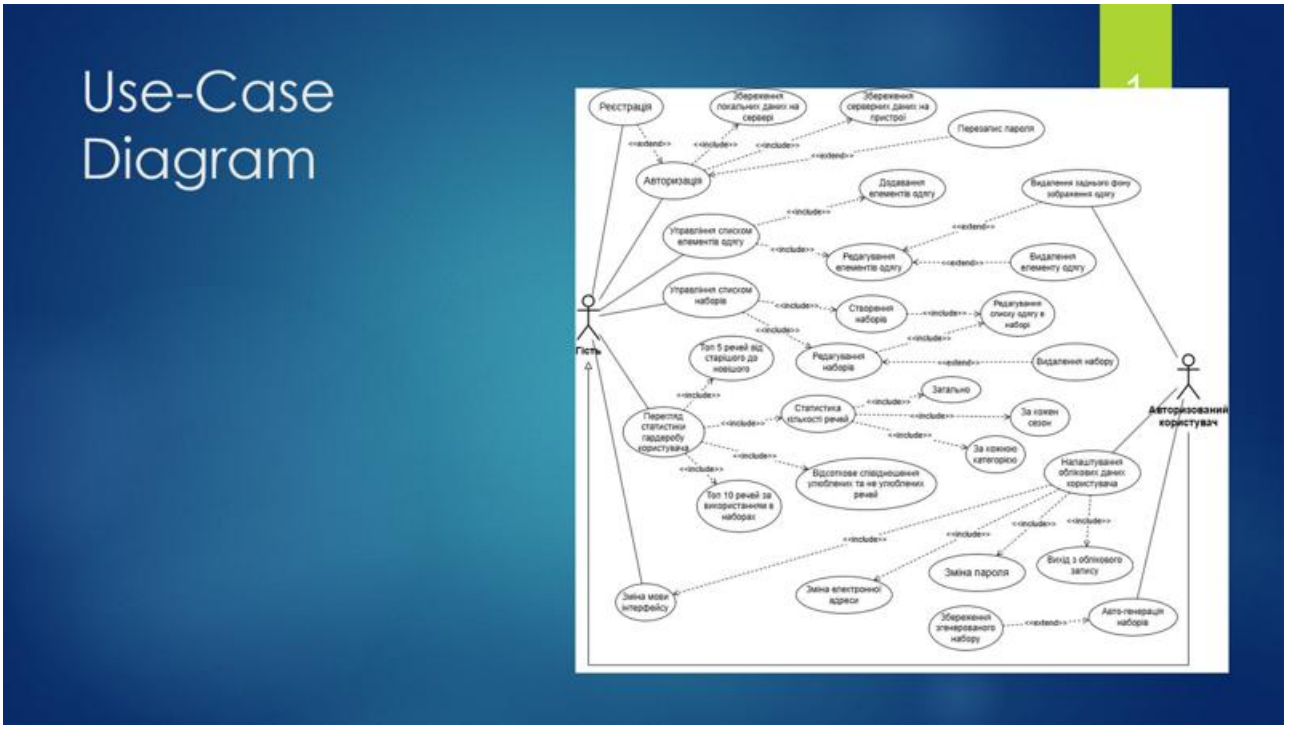

Рисунок Б.6 – Слайд 6

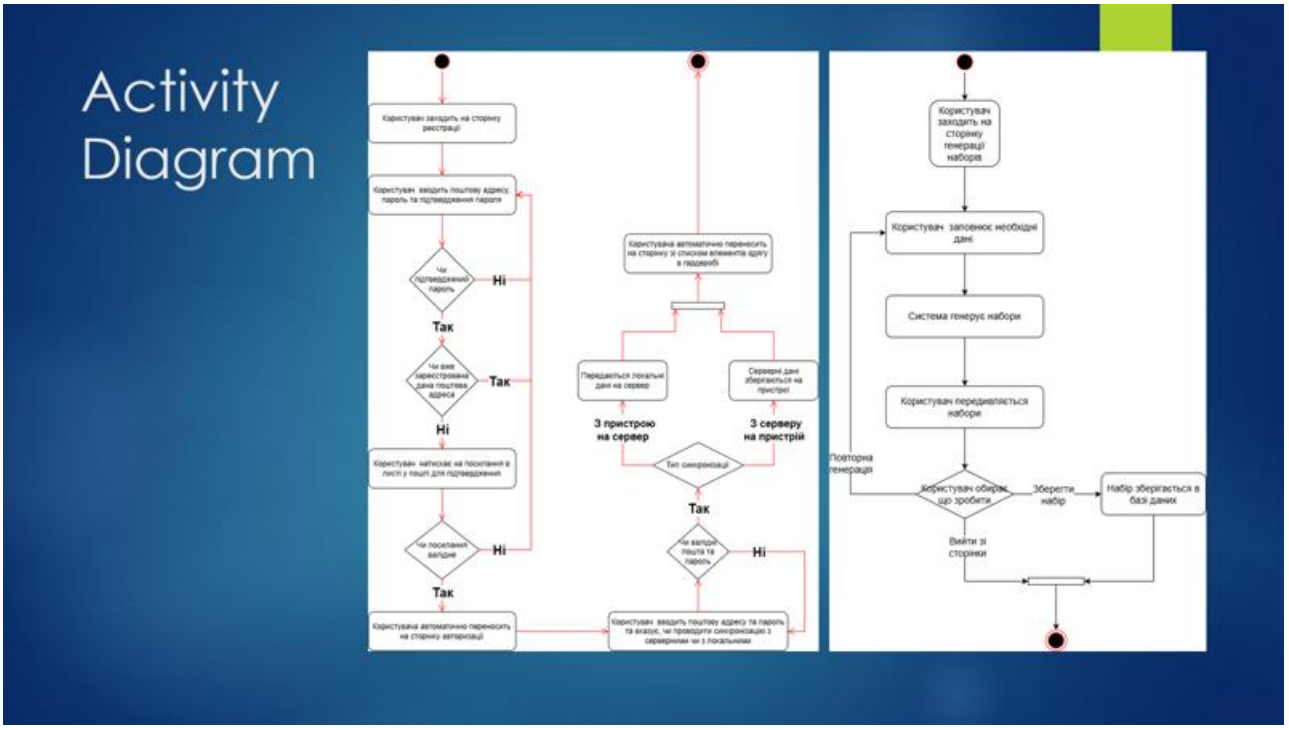

Рисунок Б.7 – Слайд 7

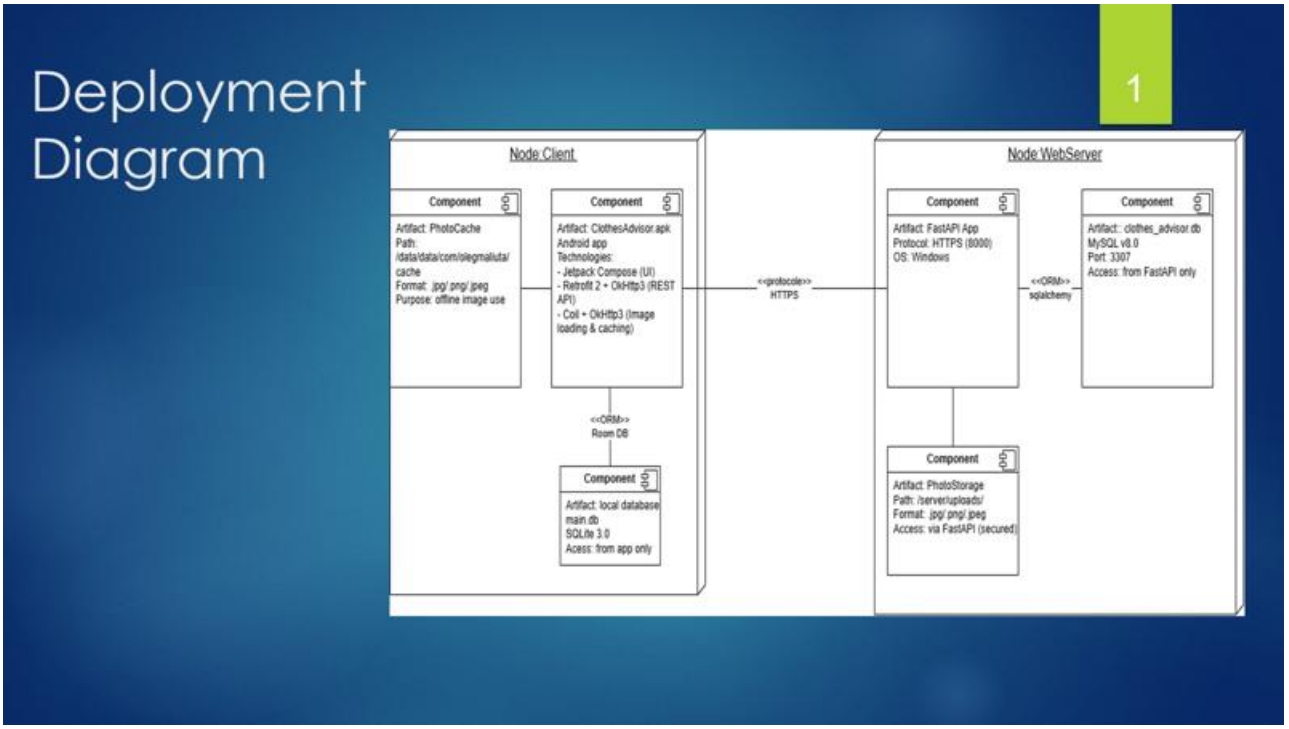

Рисунок Б.8 – Слайд 8

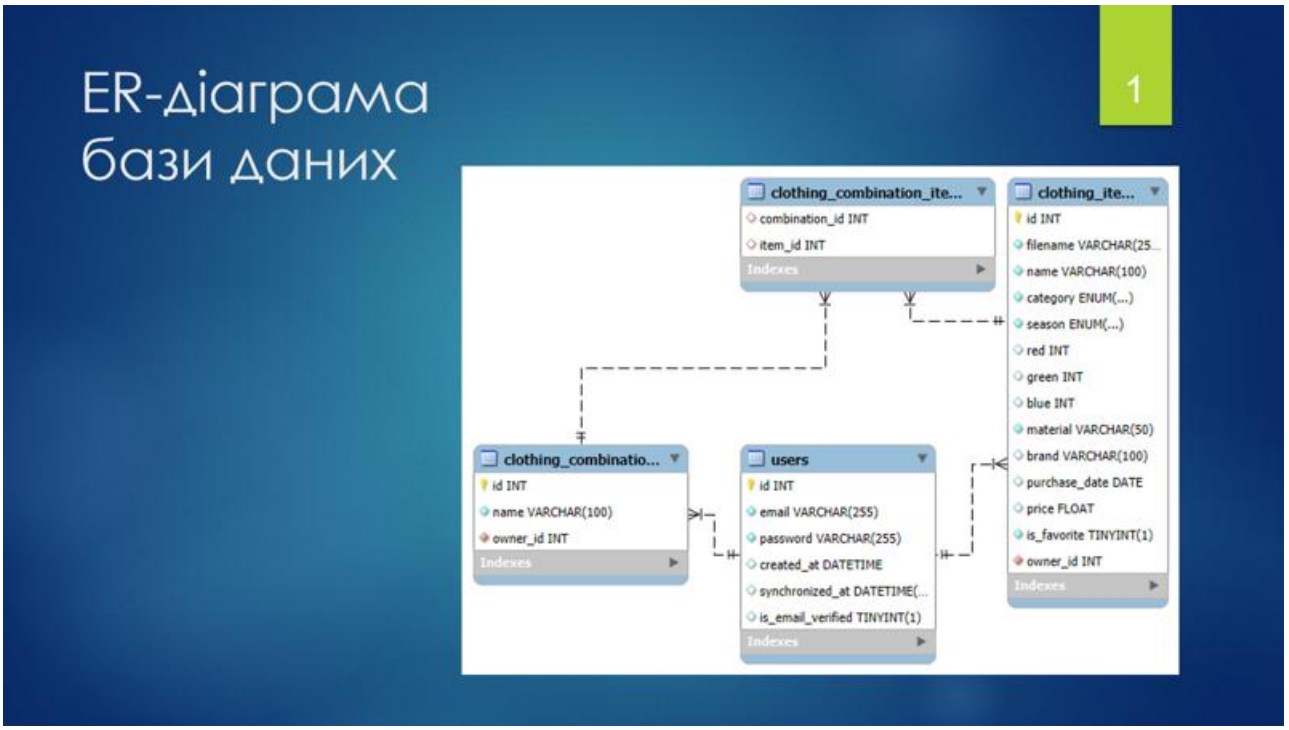

Рисунок Б.9 – Слайд 9

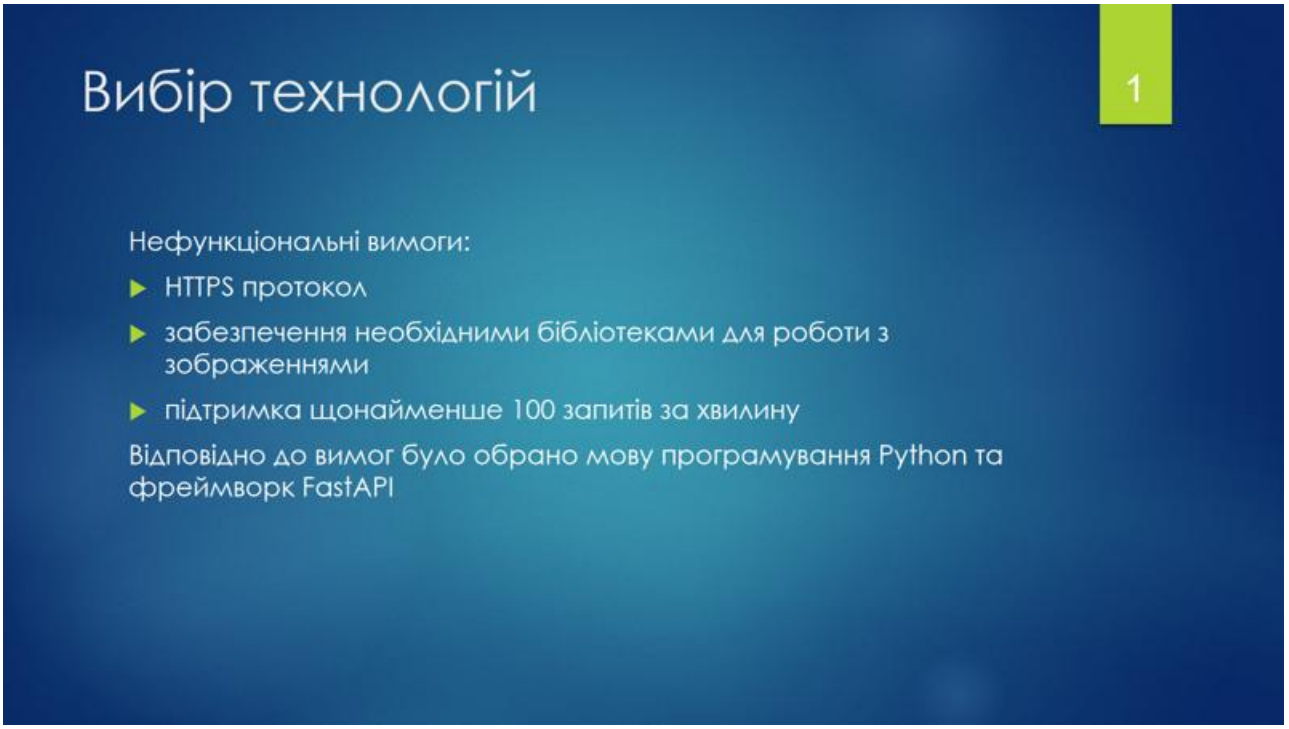

Рисунок Б.10 – Слайд 10

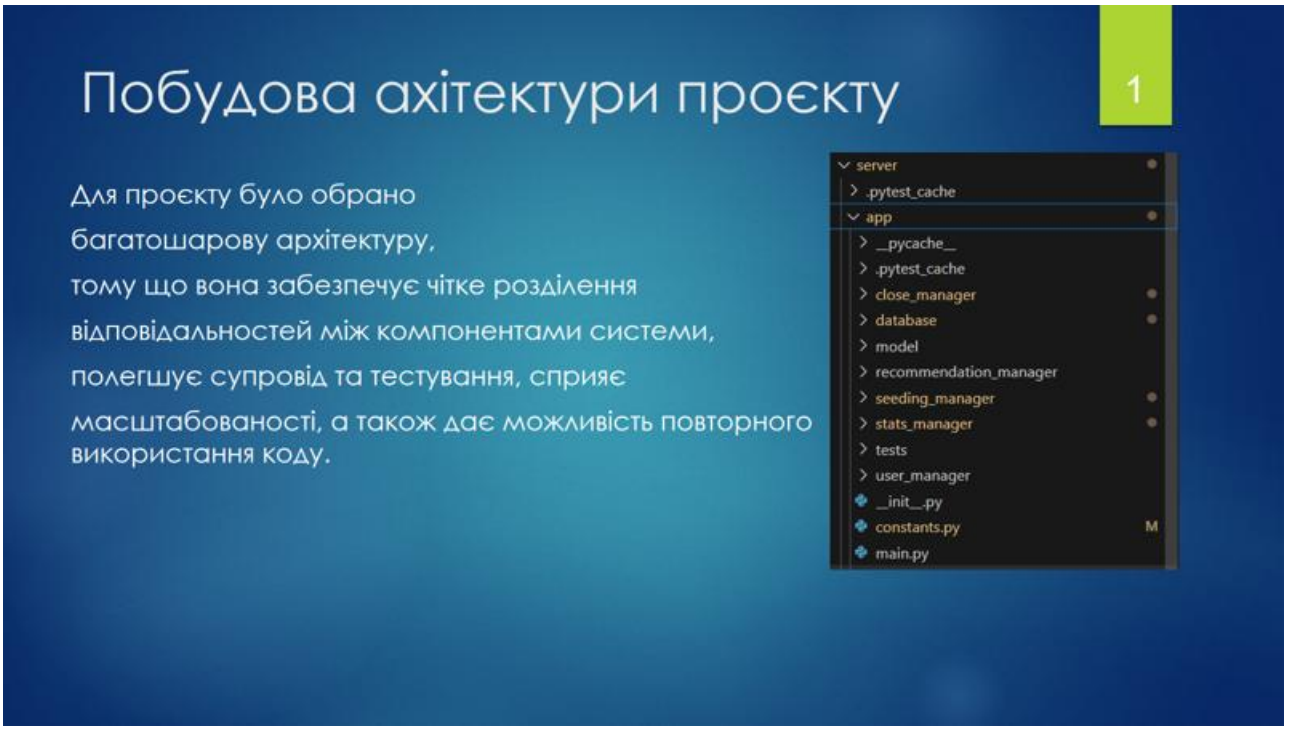

Рисунок Б.11 – Слайд 11

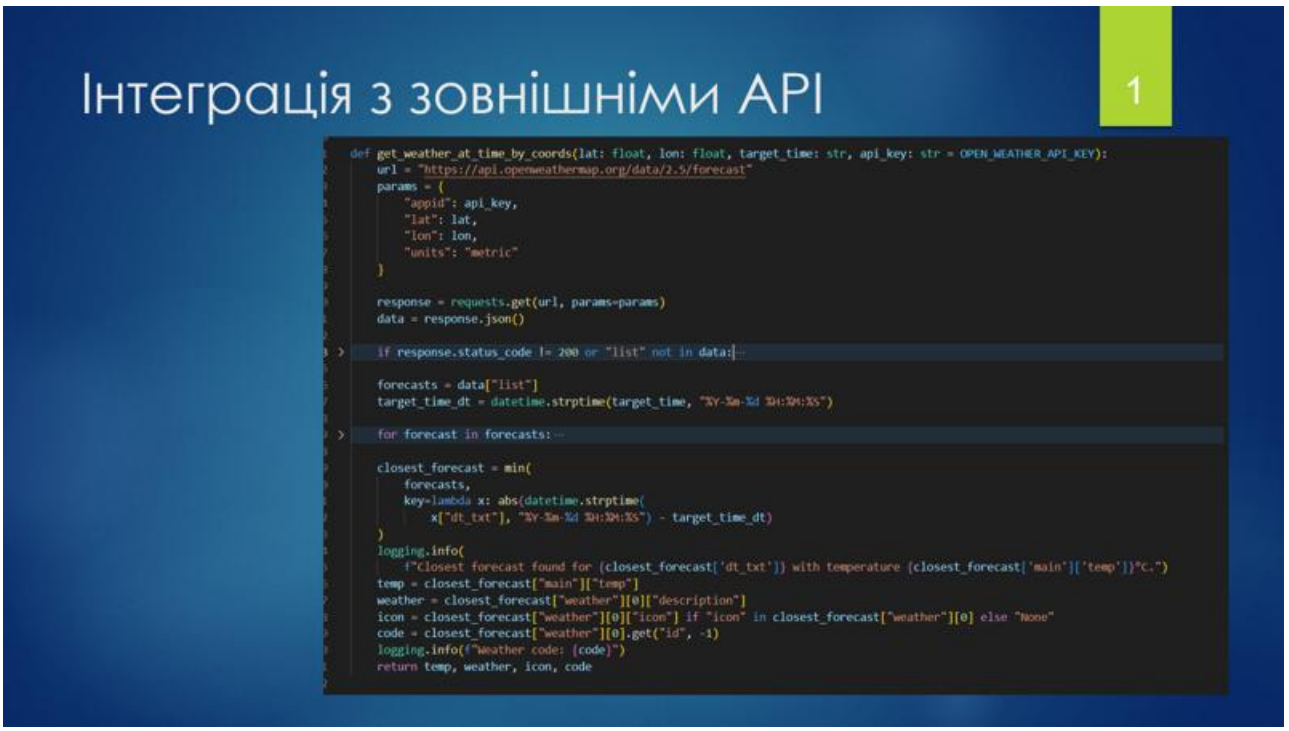

Рисунок Б.12 – Слайд 12

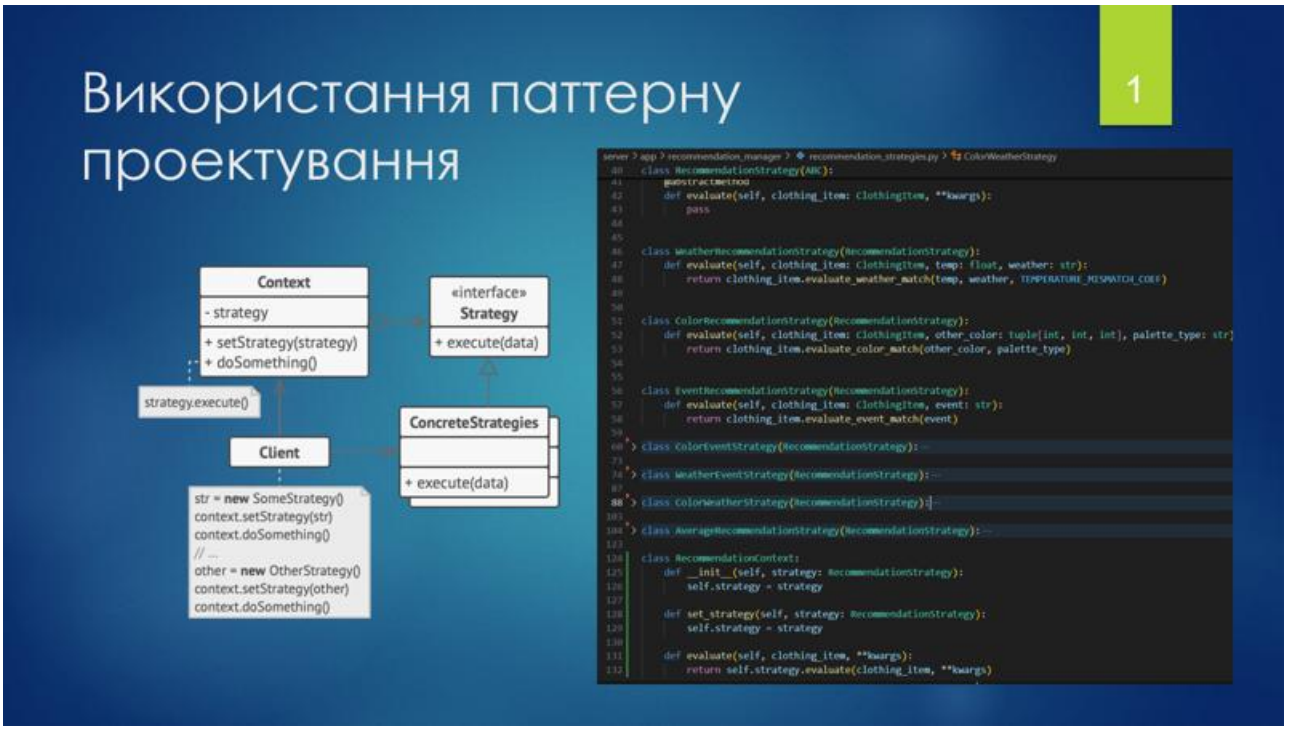

Рисунок Б.13 – Слайд 13

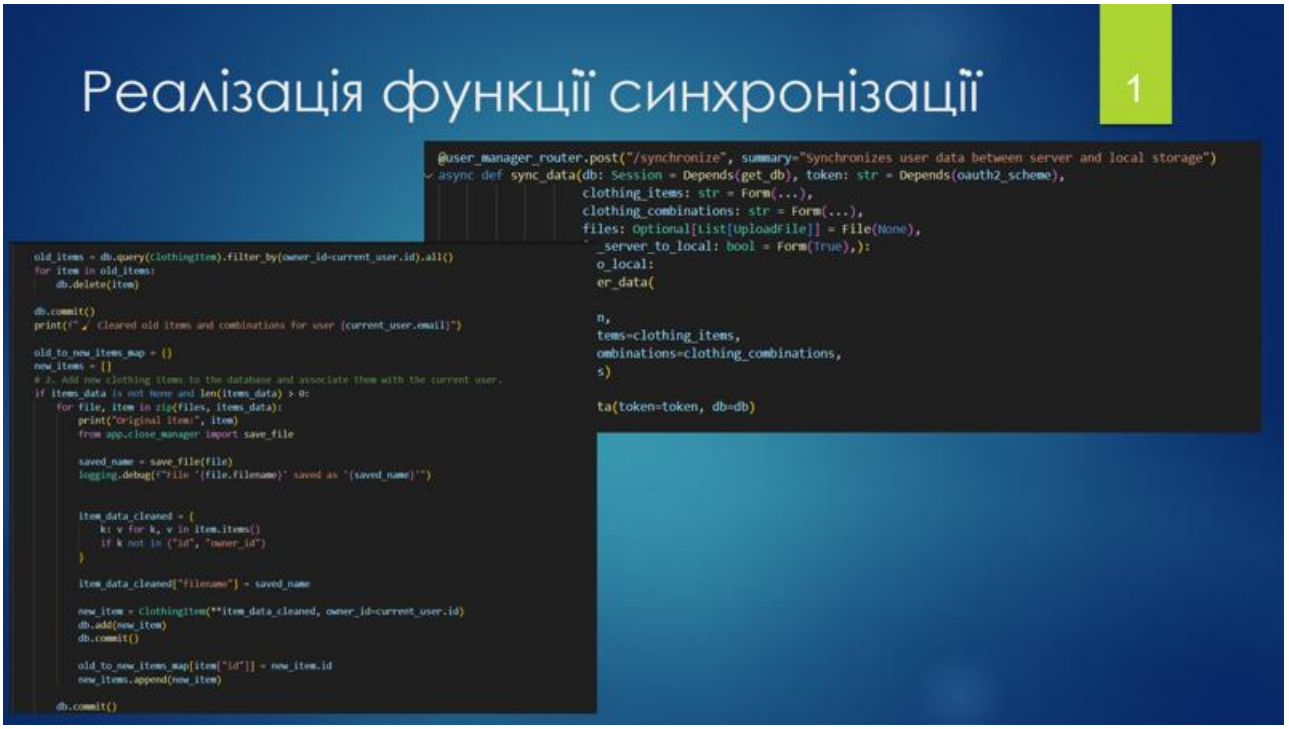

Рисунок Б.14 – Слайд 14

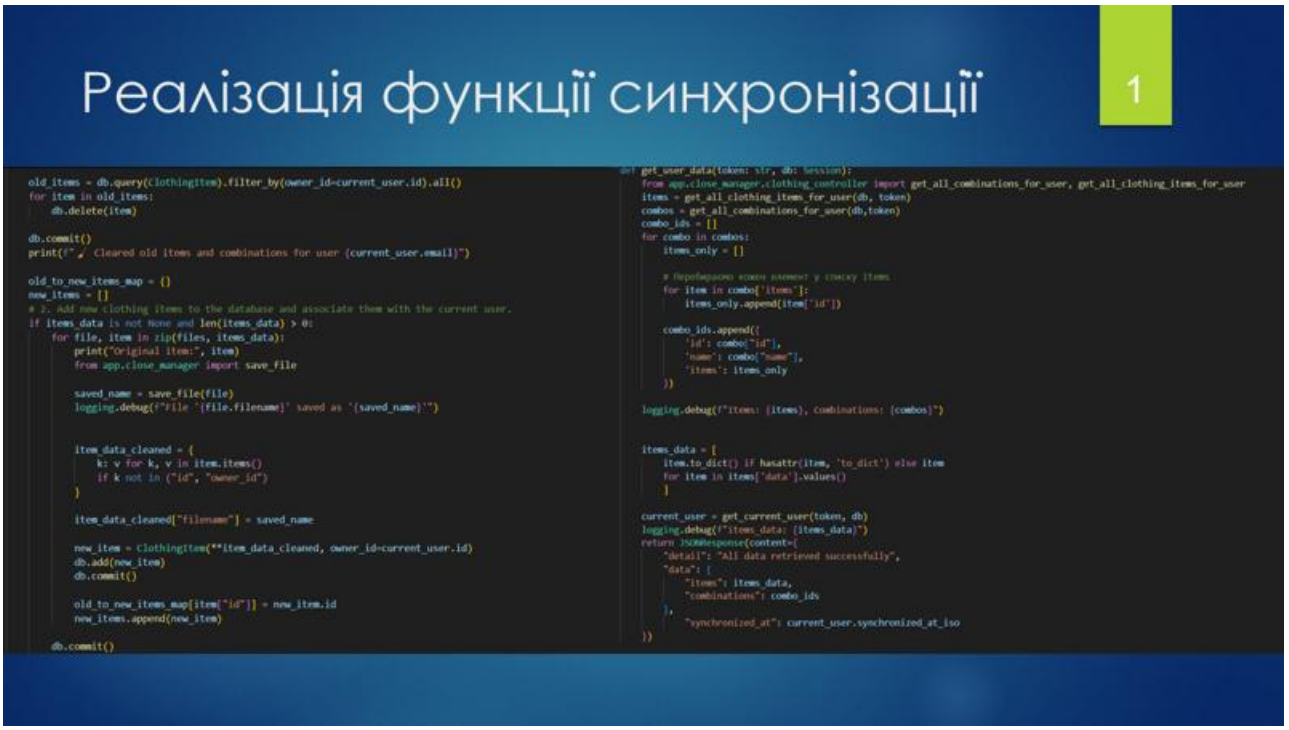

Рисунок Б.15 – Слайд 15

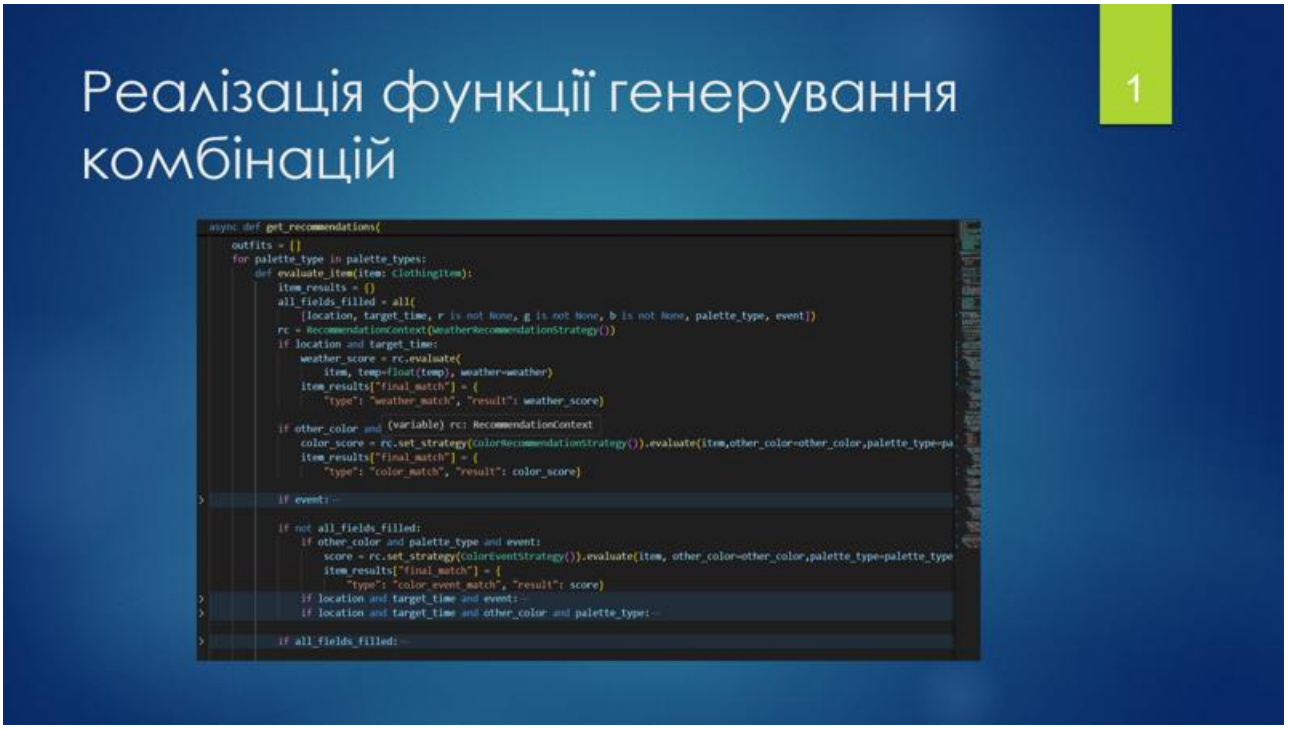

Рисунок Б.16 – Слайд 16

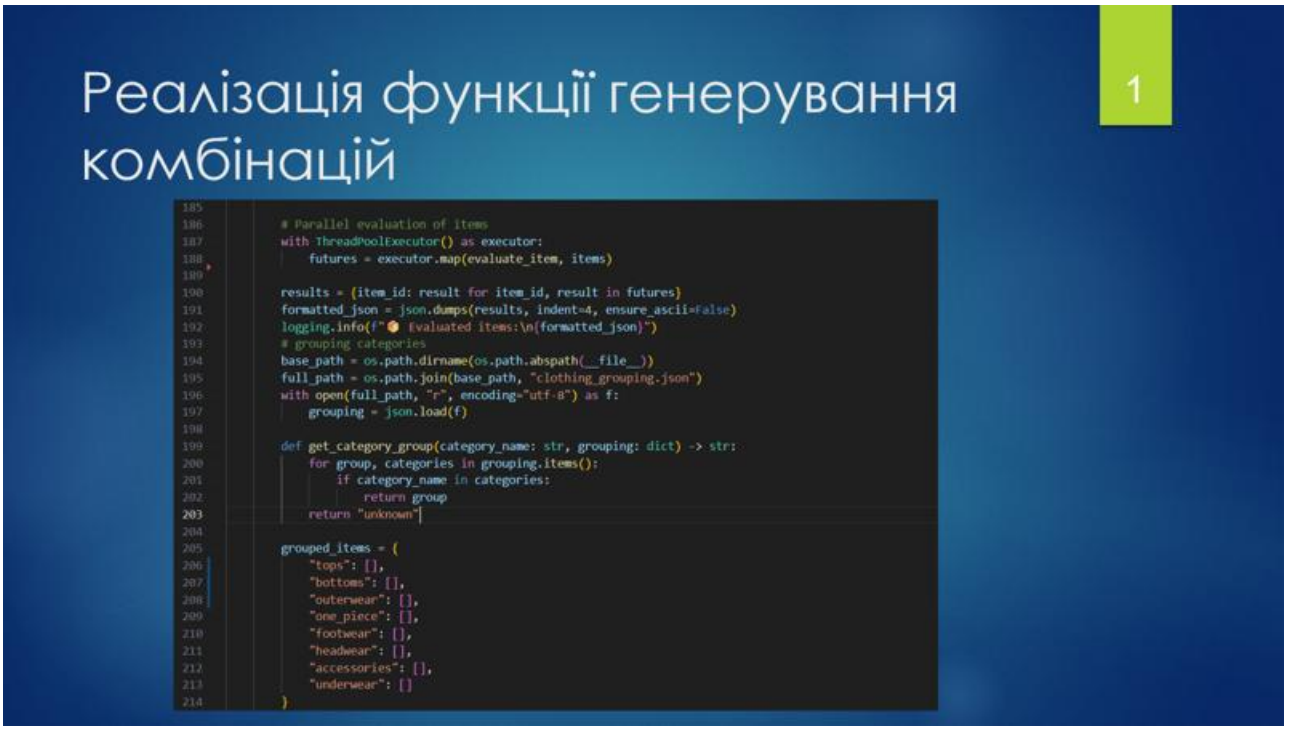

Рисунок Б.17 – Слайд 17

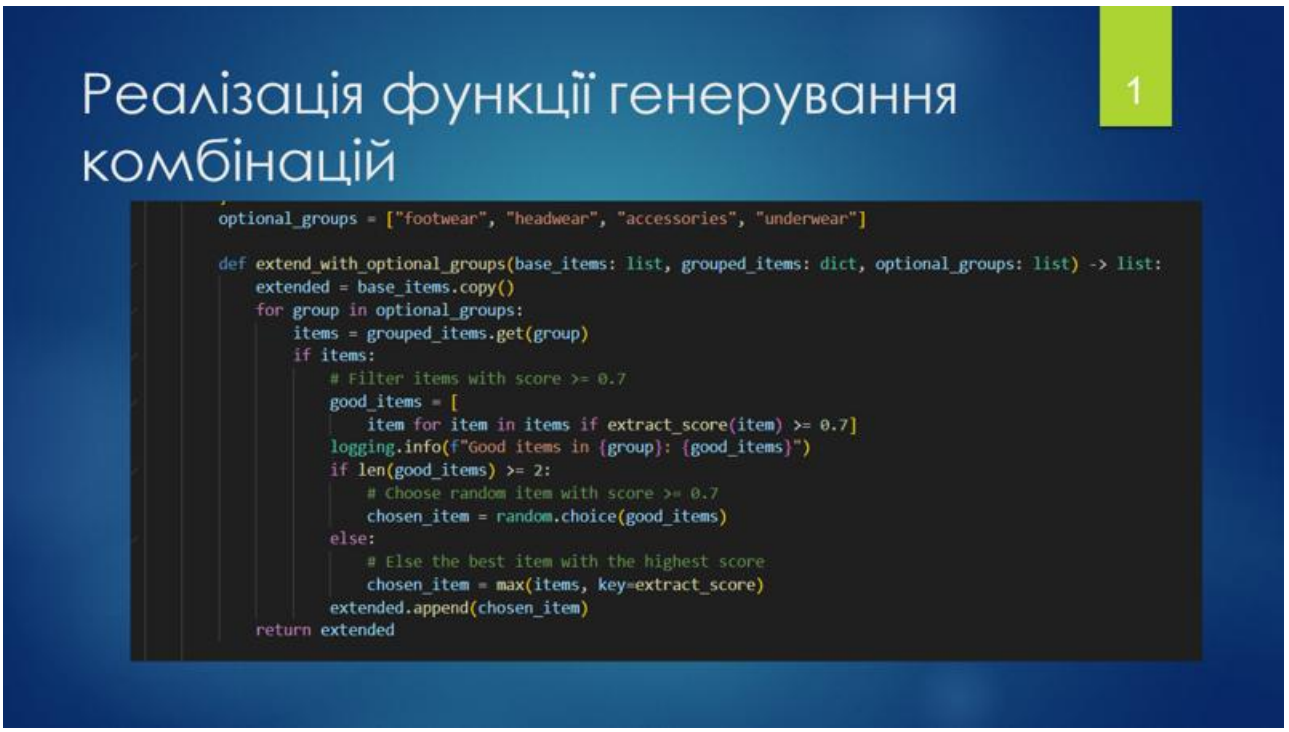

Рисунок Б.18 – Слайд 18

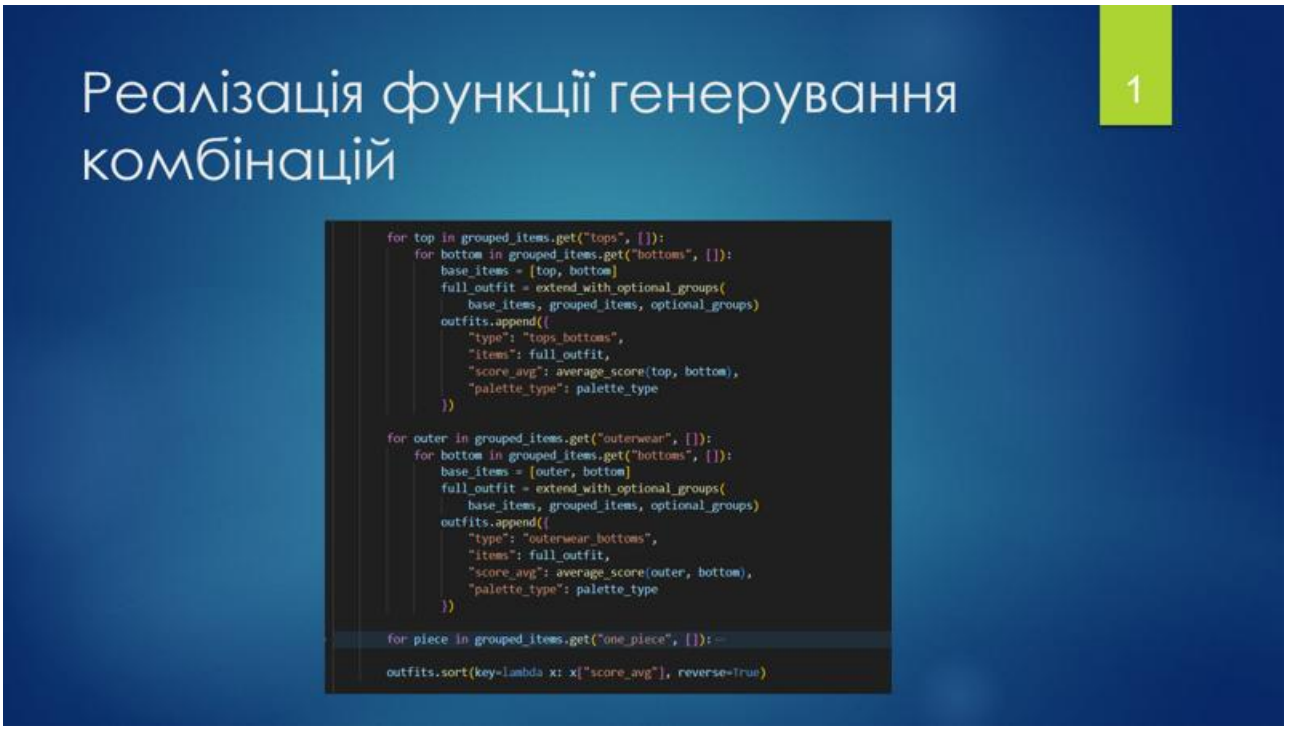

Рисунок Б.19 – Слайд 19

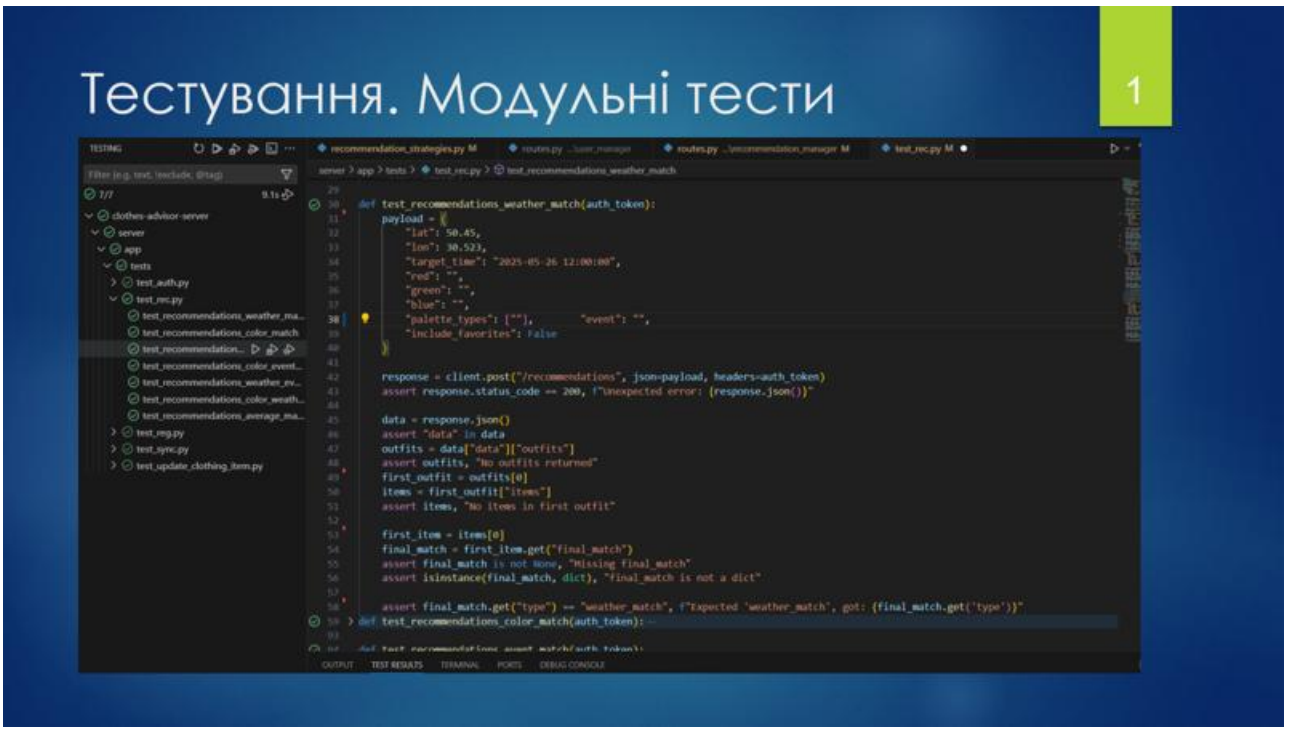

Рисунок Б.20 – Слайд 20

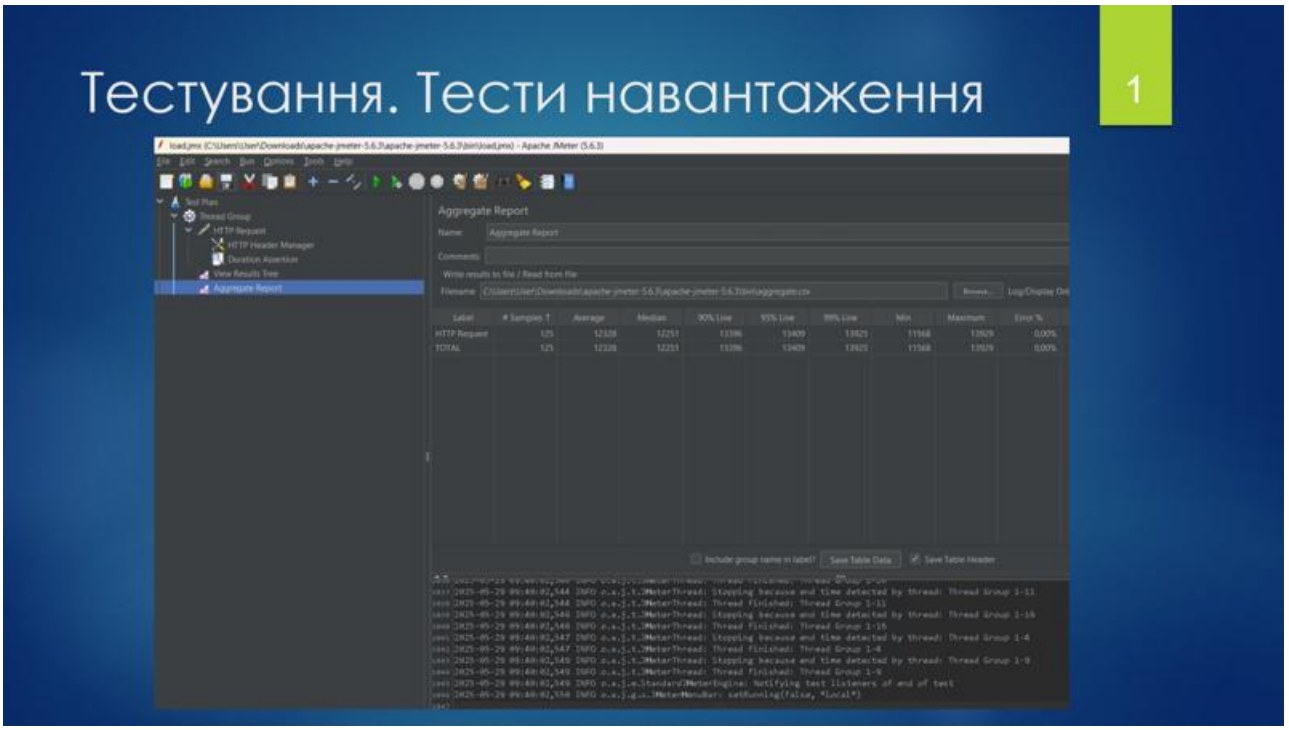

Рисунок Б.21 – Слайд 21

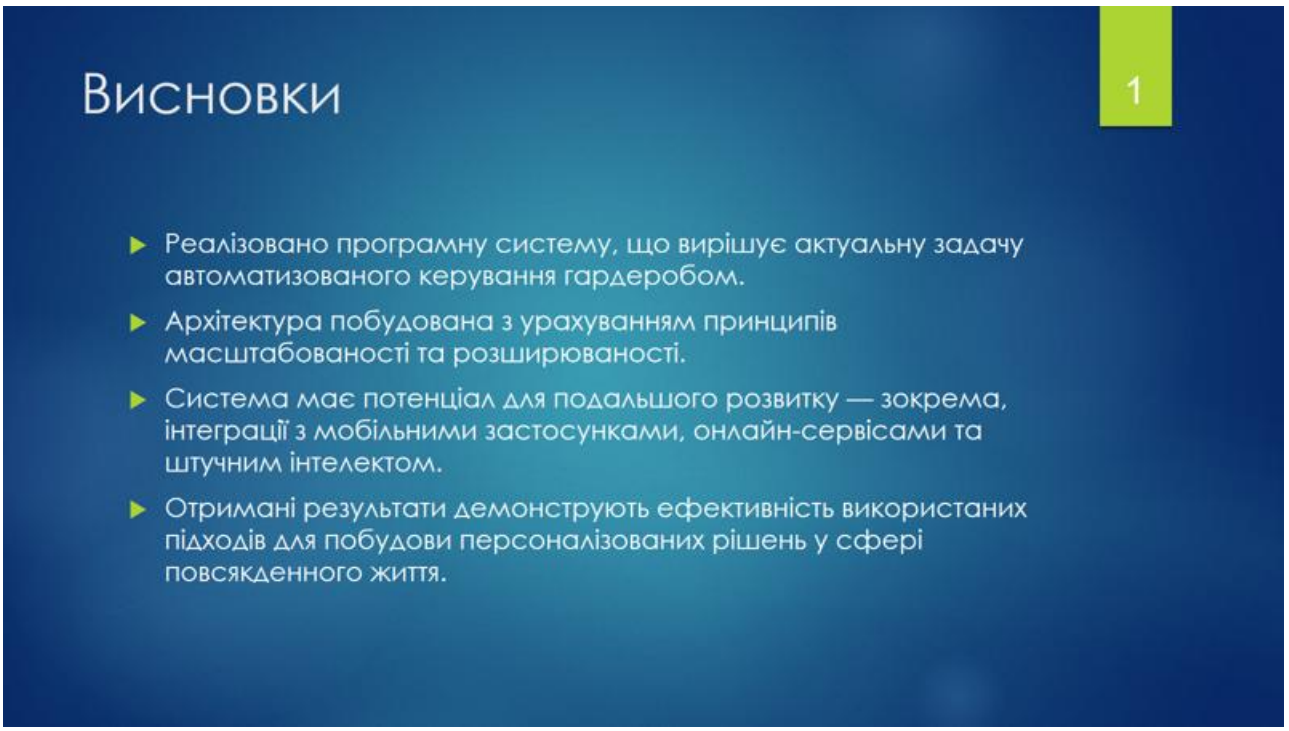

Рисунок Б.22 – Слайд 22

ДОДАТОК В

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр денної форми навчання
Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
Мобільного додатку для управління гардеробом

Студент гр. ПЗПІ-21-6 _____ Малюта О. В.

Студент гр. ПЗПІ-21-6 _____ Замковий А. Г.

Харків

2025

Специфікація

1 ВСТУП

1.1 Огляд продукту

Мобільний додаток для управління гардеробом призначений для платформи Android і надає користувачам інструменти для каталогізації одягу, створення наборів, отримання персоналізованих рекомендацій та аналізу статистики використання гардеробу. Додаток підтримує офлайн-режим та синхронізацію даних між пристроями.

1.2 Мета

Метою розробки є створення зручного інструменту для організації гардеробу, який автоматизує процес підбору одягу, надає детальну статистику і більша частина якого може працювати без інтернет-з'єднання.

1.3 Межі

Додаток призначений для ОС Android (версія 7.0 та вище);

Такі функції як реєстрація, авторизація, синхронізація, генерація наборів, та деякі інші вимагають підключення до інтернету.

1.4 Посилання

1. Android — OpenStreetMap Wiki [Електронний ресурс] – URL: https://wiki.openstreetmap.org/wiki/Android (дата звернення: 15.04.2025)
2. Jetpack Compose UI App Development Toolkit - Android Developers [Електронний ресурс] – URL: https://developer.android.com/compose (дата звернення: 06.04.2025)
3. Coroutine Image Loader [Електронний ресурс] – URL: https://coil-kt.github.io/coil/ (дата звернення: 10.04.2025)

4. MVVM (Model View ViewModel) Architecture Pattern in Android | GeeksforGeeks [Електронний ресурс] – URL: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/ (дата звернення: 08.04.2025)
5. Python 3.12 documentation. Python documentation. URL: https://docs.python.org/3/ (date of access: 10.04.2025).

1.5 Означення та абревіатури

API – Application Programming Interface

DB – Data Base

HTTP – Hyper-Text Transfer Protocol

MVVM – Model-View-ViewModel

REST – Representational State Transfer

RGB – Red Green Blue

SDK – Software Development Kit

UI – User Interface

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Аналізуючи поточний ринок та існуючі аналоги, додаток має дані перспективи, що роблять його більш актуальним:

- Можливість роботи більшості функціоналу в офлайн-режимі із локальним зберіганням даних.
- Розширені алгоритми підбору одягу, які враховують не тільки колір і категорію, але й погоду, призначення одягу (спорт, прогулянка та інші), стиль користувача.
- Створення інтуїтивного та приємного інтерфейсу для зручного управління елементами гардеробу.

2.2 Функції продукту

Мобільний додаток для управління гардеробом має наступні функціональні можливості:

- авторизація та реєстрація користувачів:
    - вхід у систему у ролі гостя з локальним зберіганням даних;
    - реєстрація за електронною адресою для збереження даних у обліковому записі;
    - авторизація з можливістю синхронізації даних між пристроями та сервером, де користувач буде вибирати, за якими даними (локальними або серверними) буде відбуватись синхронізація;
    - перезапис пароля користувача.
- управління гардеробом:
    - додавання елементів одягу з фотографуванням та збереженням зображень;
    - редагування характеристик одягу: назва, категорія (головний убір, верхній одяг, взуття тощо), сезонність (зима, весна, літо,

осінь, всесезонний), основний колір (формат RGB), матеріал (бавовна, вовна, поліестер тощо), додаткові параметри: бренд, дата придбання, вартість (за бажанням);

- видалення елементів одягу;
- можливість видалення заднього фону зображень для одягу з гардеробу;
- можливість позначання улюблених елементів одягу для покращення результату генерації комбінацій одягу.

– генерація наборів (комбінацій) одягу:
  - автоматична генерація наборів на основі: стилю та кольорової гами, погодних умов та призначення (робота, відпочинок тощо), переваг користувача (кольорова палітра та інше);
  - ручне створення наборів (комбінацій) з можливістю вибору елементів гардеробу;
  - редагування збережених наборів;
  - видалення збережених наборів.
– статистика:
  - статистика кількості речей (загально, для кожного сезону, для кожної категорії);
  - статистика за віком речей (топ 5 речей від старішого до новішого);
  - статистика використання одягу в наборах (топ 10 речей);
  - відсоткове співвідношення улюблених та не улюблених елементів одягу.
– особисті налаштування:
  - зміна мови інтерфейсу, вибираючи між англійською та українською мовами;
  - зміна електронної пошти користувача;
  - зміна пароля користувача.

2.3 Характеристики користувачів

Цільова аудиторія додатка охоплює широке коло людей з різними потребами та звичками:

- студенти та молоді професіонали (18–30 років) – активні користувачі смартфонів, які цінують простоту та функціональність. Вони часто стикаються з нестачею часу по вранці та потребують швидкого підбору одягу для навчання, роботи або зустрічей з друзями. Для них важливі такі функції, як швидке створення комплектів, синхронізація з розкладом і рекомендації на основі погоди;
- офісні працівники (25–45 років) – користувачі, які дотримуються дрес-коду або хочуть виглядати презентабельно. Вони потребують інструменту для планування образів на тиждень, аналізу того, які речі найчастіше використовуються, а які взагалі лежать без діла. Також їм буде корисна функція створення "капсульних гардеробів" для роботи;
- модні ентузіасти (будь-якого віку) – люди, які слідкують за трендами і люблять експериментувати зі стилем. Для них важливі такі функції, як збереження ідей образів, можливість ділитися своїми комплектами в соціальних мережах та отримувати інспірацію на основі власного гардеробу;
- подорожуючі – користувачі, які часто змінюють кліматичні умови і потребують швидкого підбору одягу під конкретну подорож. Для них буде особливо корисна функція автоматичного підбору речей з урахуванням прогнозу погоди в пункті призначення;
- батьки сімейств – зайняті люди, які керують не лише своїм гардеробом, але й одягом дітей. Вони оцінять можливість створення окремих профілів для кожного члена сім'ї, нагадування про сезонну зміну одягу та поради щодо комплектів для різних подій.

2.4 Загальні обмеження

Програмна система має такі обмеження:

- підтримка додатка тільки на Android 7.0 та вище;
- обмеження на розмір зображень для елементів гардеробу (5 МБ);
- обмеження по кількості елементів одягу (100) та наборів (50) на одного користувача;
- необхідність підключення до інтернету для реєстрації, авторизації, синхронізації, генерації наборів, видалення фону зображення одягу, можливості зміни електронної пошти та пароля.

2.5 Припущення й залежності

Розробка та функціонування додатка "ClothesAdvisor" базуються на наступних ключових припущеннях і мають такі залежності.

Пристрій користувача має мінімальні апаратні вимоги:

- Android 7.0 (API 24) або новішої версії;
- камера для фотографування та додавання фотографій одягу.

Для повноцінної роботи окремих функцій необхідно:

- доступ до інтернету для синхронізації даних між пристроями;
- дозвіл на використання геолокації для автоматичного підбору наборів з урахуванням погоди;
- дозвіл на доступ до сховища для доступу до фотографій.

Система залежить від:

- сервісів Google Play для роботи на Android-пристроях;
- зовнішніх погодних API для функції рекомендацій;
- зовнішніх API [1], які відповідають за відображення інтерактивної мапи для вибору локації;
- безпечних HTTPS-з'єднань для роботи з серверною частиною.

Продуктивність додатка може залежати від:

- швидкості інтернет-з'єднання для синхронізації;
- апаратних можливостей пристрою;

– доступу до GPS для визначення локації.

Бізнес-припущення, в яких передбачається, що:

– користувач має базові навички роботи з мобільними додатками;
– додаток буде використовуватись персонально, а не на спільних пристроях.

Важливо враховувати, що зміна будь-якого з цих припущень може вимагати модифікації архітектури або функціоналу додатка.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс додатка побудований за допомогою бібліотеки Jetpack Comose [2] та повинен забезпечувати інтуїтивну взаємодію для користувачів з різним рівнем технічної підготовки. Головний екран містить чотири основні секції: список одягу, список наборів, генерація наборів та налаштування.

3.1.2 Апаратний інтерфейс

Додаток використовує стандартні апаратні можливості Android-пристроїв. Для роботи з фотографіями одягу необхідний доступ до галереї зображень. Функція автоматичної генерації наборів може використовувати локацію пристрою для подальшого використання в алгоритмі.

3.1.3 Програмний інтерфейс

API додатка реалізовано з використанням архітектурного стилю REST. Більшість запитів містить заголовки авторизації з JWT-токеном. Для роботи з зображеннями використовується бібліотека Coil [3], що дозволяє отримувати та відображати на екрані завантажені зображення з сервера або локального прострою. Документація API доступна у Swagger UI за захищеним HTTPS-з'єднанням.

3.1.4 Комунікаційний протокол

Основним протоколом обміну даними є HTTPS. Для передачі даних використовуються такі формати, як JSON, Multipart та FormURLEncoded з UTF-8 кодуванням.

3.1.5 Обмеження пам’яті

Фотографії зображень для завантаження у використанні для елементів одягу повинні мати розмір не більше 5 МБ. Один авторизований користувач або гість може мати максимум 100 елементів одягу та 50 наборів.

3.1.6 Операції

Система забезпечує атомарність операцій з базою даних через механізм транзакцій. Критичні операції (видалення наборів або одягу, дозвіл на геолокацію або фотографії) супроводжуються підтвердженням користувача.

3.1.7 Функції продукту

Ядро функціоналу включає три основні групи можливостей: управління гардеробом (додавання, редагування, категорізація, фільтрація), генерація наборів (ручний підбір, автоматичні рекомендації) та аналітика (статистика використання, рекомендації з оновлення гардеробу). Більшість функцій доступна у двох режимах - онлайн та офлайн.

3.1.8 Припущення й залежності

Система передбачає наявність стабільного інтернет-з’єднання для отримання актуальної інформації про погоду з API OpenWeather та роботи з сервером.

Передбачається, що зовнішній погодний сервіс надає точні, актуальні та стандартизовані погодні описи (наприклад, "clear sky", "light rain" тощо), які система інтерпретує згідно з внутрішньою таблицею відповідностей.

Користувач підтримує гардероб у системі в актуальному стані: додає, редагує або видаляє одяг відповідно до реального вмісту.

Система розробляється для роботи на сучасних пристроях Android , тому покладається на наявність підтримки сучасних веб-технологій або Android API.

Користувач приймає рекомендації одягу як допоміжні — остаточний вибір залежить від його вподобань.

Система залежить від сторонніх сервісів (погодне API, бібліотек тощо).

3.2 Властивості програмного продукту

Програмний продукт відрізняється гнучкістю налаштувань інтерфейсу, включаючи можливість вибору між компактним і розширеним відображенням елементів. Архітектура мобільного додатку MVVM [4] забезпечує чітке розділення логіки та відображення. Специфічною особливістю є алгоритм автоматичної генерації наборів, який враховує такі фактори, як погода, сезон, основний колір, кольорові палітри, призначення, та врахування улюблених речей.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна функціонувати без збоїв протягом тривалого періоду часу, навіть при багаторазовому зверненні до погодного API або частій зміні користувацьких даних.

Застосовуються механізми обробки помилок для запобігання критичних збоїв при втраті з'єднання з мережею або при недоступності сторонніх сервісів.

У випадку непередбаченого завершення роботи або збою, система має забезпечити збереження введених даних (наприклад, оновлення гардеробу чи обрані налаштування).

При повторному запуску програми користувач продовжує з останнього стабільного стану.

Програма перевіряє правильність введених користувачем даних (наприклад текстові назви одягу, числові параметри температури), щоб уникнути некоректної обробки або краху.

3.3.2 Доступність

Офлайн-режим забезпечує доступ до всіх локально збережених даних без обмежень. При відсутності інтернет-з'єднання недоступні операції позначаються повідомленнями або повною візуальною відсутністю. Синхронізація відбувається у фоновому режимі при відновленні зв'язку.

3.3.3 Безпека

Система використовує захищений режим HTTPS для передачі даних до сервера та відправки відповіді. Для безпечного зберігання паролів користувачів використовується хешування алгоритмом bcrypt. Аутентифікація користувачів реалізована за допомогою токенів JWT. Після успішного входу система видає унікальний токен, який використовується для авторизації у всіх наступних запитах. Це забезпечує безпечний та контрольований доступ до персональних даних користувача без потреби повторного введення пароля. Токени мають обмежений термін дії та можуть бути відкликані у випадку втрати доступу або підозрілої активності.

3.3.4 Супроводжуваність

Кодова база мобільного додатку супроводжується модульними та автоматизованими і мануальними UI-тестами.

Усі REST API-запити мають документацію у форматі OpenAPI (Swagger), що спрощує підтримку, налагодження та інтеграцію з іншими клієнтами.

Сервер підтримує централізоване логування подій (зокрема помилок, авторизацій, запитів)

Основні компоненти серверної логіки покриті автоматизованими тестами та тестами навантаження, що дозволяє виявляти помилки до розгортання.

3.3.5 Переносимість

Система керування гардеробом повинна мати високий рівень переносимості, щоб забезпечити можливість її використання на різних платформах. Вимоги до переносимості включають:

Платформи: Система повинна працювати на операційних системах Android (версія 7.0 і вище). У майбутньому можлива адаптація для iOS та веб-платформи.

Мобільні пристрої: Інтерфейс користувача повинен коректно відображатися на екранах різних розмірів (від 4" до 7.5").

Переносимість даних: Повинна бути реалізована можливість експорту та імпорту даних гардеробу між клієнтом та сервером для резервного копіювання та перенесення між пристроями.

3.3.6 Продуктивність

Система повинна забезпечувати належну продуктивність для комфортного використання, зокрема:

Час запуску: Час запуску застосунку не повинен перевищувати 3 секунд на середньостатистичному пристрої (з 4 ГБ оперативної пам’яті).

Час відповіді: Час реакції на дії користувача (перегляд, редагування, додавання одягу) не повинен перевищувати 1 с, час відповіді від серверу не повинен перевищувати 3 с.

Масштабованість: Система повинна підтримувати зберігання 100 елементів одягу без суттєвого падіння продуктивності.

Оперативне споживання пам’яті: Додаток має раціонально використовувати ресурси, не перевищуючи 150 МБ оперативної пам’яті під час стандартної роботи.

3.4 Вимоги баз даних

Система повинна використовувати вбудовану базу даних для зберігання інформації про гардероб на локальному пристрої та на виділеному сервері. Основні вимоги до бази даних:

Має бути SQL сситемою з підтримкою ORM.

Операції CRUD: Повна підтримка операцій створення, читання, оновлення та видалення елементів.

Цілісність: Має бути реалізована підтримка зв’язків між сутностями (наприклад, багато до багатьох та один до багатьох).

Резервне копіювання: Повинна бути можливість експортувати базу даних у вигляді файлу, який може бути збережений.

3.5 Інші вимоги

Система повинна підтримувати інтернаціоналізацію з можливістю додавання нових мов без змін коду.

ДОДАТОК Г

Схема бази даних

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `created_at` datetime DEFAULT NULL,
  `synchronized_at` datetime(6) DEFAULT NULL,
  `is_email_verified` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  KEY `ix_users_id` (`id`)
)
CREATE TABLE `clothing_items` (
  `id` int NOT NULL AUTO_INCREMENT,
  `filename` varchar(255) NOT NULL,
  `name` varchar(100) NOT NULL,
  `category`
enum('tshirt','pants','jacket','dress','skirt','shorts','hoodie','sweate
r','coat','blouse','shoes','accessories','boots','sneakers','sandals','h
at','scarf','gloves','socks','underwear','swimwear','belt','bag','watch',
'jeans','leggings','tank_top','overalls','beanie') NOT NULL,
  `season` enum('winter','spring','summer','autumn') NOT NULL,
  `red` int DEFAULT NULL,
```

```
    `green` int DEFAULT NULL,
    `blue` int DEFAULT NULL,
    `material` varchar(50) NOT NULL,
    `brand` varchar(100) DEFAULT NULL,
    `purchase_date` date DEFAULT NULL,
    `price` float DEFAULT NULL,
    `is_favorite` tinyint(1) NOT NULL,
    `owner_id` int NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `filename` (`filename`),
    KEY `owner_id` (`owner_id`),
    KEY `ix_clothing_items_id` (`id`),
    CONSTRAINT `clothing_items_ibfk_1` FOREIGN KEY (`owner_id`)
REFERENCES `users` (`id`)
  )
  CREATE TABLE `clothing_combinations` (
    `id` int NOT NULL AUTO_INCREMENT,
    `name` varchar(100) NOT NULL,
    `owner_id` int NOT NULL,
    PRIMARY KEY (`id`),
    KEY `owner_id` (`owner_id`),
    KEY `ix_clothing_combinations_id` (`id`),
```

```
        CONSTRAINT `clothing_combinations_ibfk_1` FOREIGN KEY (`owner_id`)
REFERENCES `users` (`id`)

      )

       CREATE TABLE `clothing_combination_items` (

        `combination_id` int DEFAULT NULL,

        `item_id` int DEFAULT NULL,

        KEY `combination_id` (`combination_id`),

        KEY `item_id` (`item_id`),

        CONSTRAINT   `clothing_combination_items_ibfk_1`   FOREIGN   KEY
(`combination_id`) REFERENCES `clothing_combinations` (`id`),

        CONSTRAINT   `clothing_combination_items_ibfk_2`   FOREIGN   KEY
(`item_id`) REFERENCES `clothing_items` (`id`)

      )
```

ДОДАТОК Д

Код seeding.py

```
from datetime import date, datetime, timezone

import os

import shutil


import requests

from app.database import get_db

from app.model.user import User

from sqlalchemy.orm import Session

from sqlalchemy.exc import IntegrityError


from app.user_manager.user_controller import hash_password

from app.model import *


def seed_users(db: Session):

    users_data = [

        {

            "email": "test@gmail.com",

            "password": "pass",

            "is_email_verified": True

        },
```

```
            {
                "email": "bob@example.com",
                "password": "pass",
                "is_email_verified": False
            },
            {
                "email": "charlie@example.com",
                "password": "pass",
                "is_email_verified": True
            }
        ]

        for data in users_data:
            existing_user = db.query(User).filter_by(email=data["email"]).first()
            if existing_user:
                print(f"    User    {data['email']}    already    exists.
Skipping.")
                continue

            user = User(
                email=data["email"],
                password=hash_password(data["password"]),
```

```
            created_at=datetime.now(timezone.utc),
            synchronized_at=datetime.now(timezone.utc),
            is_email_verified=data["is_email_verified"]
        )

        db.add(user)
        try:
            db.commit()
            print(f"✅ User {data['email']} seeded successfully.")
        except IntegrityError:
            db.rollback()
            print(f"❌ Failed to seed user {data['email']} (possible duplicate).")


def seed_clothing_items(db: Session):
    items_data = [
        {
            "filename": "1.jpg",
            "name": "Зимові штани",
            "category": CategoryEnum.pants,
            "season": SeasonEnum.winter,
            "red": 0, "green": 0, "blue": 0,
            "material": "Поліестер",
```

```
        "brand": "North Face",
        "purchase_date": date(2023, 12, 1),
        "price": 1200.50,
        "is_favorite": True,
        "owner_id": 1
    },
    {
        "filename": "2.jpg",
        "name": "Літній піджак",
        "category": CategoryEnum.jacket,
        "season": SeasonEnum.summer,
        "red": 113, "green": 175, "blue": 222,
        "material": "Бавовна",
        "brand": "H&M",
        "purchase_date": date(2022, 7, 15),
        "price": 299.99,
        "is_favorite": False,
        "owner_id": 1
    },
    {
        "filename": "3.jpg",
        "name": "Осіннє плаття",
```

```
        "category": CategoryEnum.dress,
        "season": SeasonEnum.autumn,
        "red": 255, "green": 255, "blue": 255,
        "material": "Шерсть",
        "brand": "Zara",
        "purchase_date": date(2023, 10, 5),
        "price": 450.00,
        "is_favorite": True,
        "owner_id": 2
    },
    {
        "filename": "4.jpg",
        "name": "Футболка",
        "category": CategoryEnum.tshirt,
        "season": SeasonEnum.spring,
        "red": 255, "green": 255, "blue": 255,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": False,
        "owner_id": 3
```

```
},
{
    "filename": "5.jpg",
    "name": "Шорти",
    "category": CategoryEnum.shorts,
    "season": SeasonEnum.spring,
    "red": 0, "green": 0, "blue": 0,
    "material": "Бавовна",
    "brand": "Uniqlo",
    "purchase_date": date(2023, 4, 10),
    "price": 199.00,
    "is_favorite": False,
    "owner_id": 1
},
{
    "filename": "6.jpg",
    "name": "Жовте худі",
    "category": CategoryEnum.hoodie,
    "season": SeasonEnum.autumn,
    "red": 252, "green": 186, "blue": 3,
    "material": "Бавовна",
    "brand": "Uniqlo",
```

```
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": True,
        "owner_id": 1
    },
    {
        "filename": "7.jpg",
        "name": "Джинси",
        "category": CategoryEnum.jeans,
        "season": SeasonEnum.winter,
        "red": 53, "green": 23, "blue": 135,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": True,
        "owner_id": 1
    },
    {
        "filename": "8.jpg",
        "name": "Червона майка",
        "category": CategoryEnum.tank_top,
```

```
        "season": SeasonEnum.summer,
        "red": 255, "green": 0, "blue": 0,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": True,
        "owner_id": 1
    },
    {
        "filename": "9.jpg",
        "name": "Кросівки",
        "category": CategoryEnum.sneakers,
        "season": SeasonEnum.summer,
        "red": 22, "green": 255, "blue": 75,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": False,
        "owner_id": 1
    },
```

```
{
    "filename": "10.jpg",
    "name": "Шарф",
    "category": CategoryEnum.scarf,
    "season": SeasonEnum.winter,
    "red": 62, "green": 57, "blue": 82,
    "material": "Бавовна",
    "brand": "Uniqlo",
    "purchase_date": date(2023, 4, 10),
    "price": 199.00,
    "is_favorite": False,
    "owner_id": 1
},
{
    "filename": "11.jpg",
    "name": "Apple Watch",
    "category": CategoryEnum.watch,
    "season": SeasonEnum.summer,
    "red": 245, "green": 122, "blue": 7,
    "material": "Бавовна",
    "brand": "Uniqlo",
    "purchase_date": date(2023, 4, 10),
```

```
        "price": 199.00,
        "is_favorite": True,
        "owner_id": 1
    },
    {
        "filename": "12.jpg",
        "name": "Сандалі",
        "category": CategoryEnum.sandals,
        "season": SeasonEnum.summer,
        "red": 66, "green": 36, "blue": 7,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": False,
        "owner_id": 1
    },
    {
        "filename": "13.jpg",
        "name": "Кепка",
        "category": CategoryEnum.hat,
        "season": SeasonEnum.summer,
```

```
        "red": 140, "green": 97, "blue": 140,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": False,
        "owner_id": 1
    },
    {
        "filename": "14.jpg",
        "name": "Браслет",
        "category": CategoryEnum.accessories,
        "season": SeasonEnum.summer,
        "red": 0, "green": 0, "blue": 255,
        "material": "Бавовна",
        "brand": "Uniqlo",
        "purchase_date": date(2023, 4, 10),
        "price": 199.00,
        "is_favorite": True,
        "owner_id": 1
    }
]
```

```
        image_urls = [

"https://fahrenheit.ua/files/products/4x7a3262.1000x1000.jpg",

            "https://lingerie.ua/files/product/0/19020/19020.jpg",

"https://preview.free3d.com/img/2014/08/2162617793575388491/fulrgt7a.jpg
",

"https://tornado.kiev.ua/image/cache/catalog/image/cache/new/41210_2-
1000x1200.webp",

"https://gard.com.ua/image/cache/catalog/image/cache/catalog/shop/produc
ts/cf8cb7c8-2f1d-11f0-80d8-ba4fdc50ab5f-450x600.webp",

            "https://football-world.com.ua/product-
images/default/Jako/38648-64a5320c81ade.webp",

"https://thenormalbrand.com/cdn/shop/files/FLATLAY_GB_804d9c90-577f-
4d7a-981d-4fd547514e0a.jpg?v=1741813113&width=1445",

"https://prostomayki.com.ua/calc/images/Mayki/mayka_red.jpg",

"https://images.prom.ua/3638886825_w600_h600_3638886825.jpg",

            "https://business-
style.com.ua/image/cache/catalog/product/9132_main-1164x1340.jpg",

"https://iplanet.one/cdn/shop/files/Apple_Watch_Ultra_2_LTE_49mm_Titaniu
m_Orange_Ocean_Band_PDP_Image_Avail_Position-1__en-IN_9ca47923-c612-
46ac-a932-fbce9b5c705c.jpg?v=1698878148&width=1445",
```

```
            "https://site-obuvi.com.ua/images/products/bosonozhki_s667-
12_na_site-obuvi.com__2_.jpg",

            "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSSG9CAt7HBLoSoWCcs29-
JraQQqxdooapRXQ&s",

            "https://shop.energetix.tv/media/image/21/52/e0/3191-
26_WEB_DBF55E0E0E5372B410889F2CB7F86CD3_768x768@2x.jpg"

        ]



        uploads_dir = os.path.join(os.getcwd(), "uploads")

        os.makedirs(uploads_dir, exist_ok=True)



        for idx, item_data in enumerate(items_data):

            image_url = image_urls[idx]

            dest_file = os.path.join(uploads_dir, item_data["filename"])



            # Завантаження зображення

            if not os.path.exists(dest_file):

                try:

                    headers = {

                        "User-Agent": "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 "

                                      "(KHTML, like Gecko)
Chrome/119.0.0.0 Safari/537.36"

                    }
```

```
                    response = requests.get(image_url, headers=headers)
                    response.raise_for_status()

                    with open(dest_file, "wb") as f:
                        f.write(response.content)

                    print(f"          Downloaded          image          as
{item_data['filename']} to uploads/")
                except Exception as e:
                    print(f"✗    Failed    to    download    image    from
{image_url}: {e}")

            # Перевірка, чи одяг уже існує
            existing                                                    =
db.query(ClothingItem).filter_by(filename=item_data["filename"]).first()
            if existing:
                print(f"  ClothingItem  {item_data['filename']}  already
exists. Skipping.")
                continue

            # Створення нового одягу
            clothing_item = ClothingItem(**item_data)
            db.add(clothing_item)
            print(f"✓ Added ClothingItem: {item_data['name']}")
```

```
    db.commit()
    print("✅ Clothing items seeded.")



def seed_clothing_combinations(db: Session):
    combinations_data = [
        {
            "name": "Зимова прогулянка",
            "owner_id": 1,
            "item_ids": [1, 2]
        },
        {
            "name": "Офісний образ",
            "owner_id": 2,
            "item_ids": [3, 4]
        },
        {
            "name": "Весняна прогулянка",
            "owner_id": 3,
            "item_ids": [5]
        }
```

```
        ]


        for combo_data in combinations_data:
            existing = db.query(ClothingCombination).filter_by(name=combo_data["name"]).first()
            if existing:
                print(f"  Combination '{combo_data['name']}' already exists. Skipping.")
                continue

            combination = ClothingCombination(
                name=combo_data["name"],
                owner_id=combo_data["owner_id"]
            )

            # Додаємо речі по item_ids
            for item_id in combo_data["item_ids"]:
                item = db.query(ClothingItem).filter_by(id=item_id).first()
                if item:
                    combination.items.append(item)

            db.add(combination)
```

```
    db.commit()
    print("✅ Clothing combinations seeded by ID.")


def seed():
    db = next(get_db())
    try:
        seed_users(db)
        seed_clothing_items(db)
        seed_clothing_combinations(db)
        print("✅ All data seeded successfully.")
    except Exception as e:
        print(f"❌ Error seeding data: {e}")
        db.rollback()
    finally:
        db.close()


if __name__ == "__main__":
    seed()
```

ДОДАТОК Е

Код реєстрації та авторизації

```
    # Create a new user
    async def create_user(db: Session, email: str, password: str,
locale: str):
        try:
            # email verification
            email_regex = r"(^[a-z0-9]+[\._]?[a-z0-
9]+[@]\w+[.]\w{2,3}$)"
            if not re.match(email_regex, email):
                return JSONResponse(
                    status_code=400,
                    content={
                        "detail": "Invalid email format",
                        "data": None
                    }
                )
            user = db.query(User).filter(User.email == email).first()
            if user:
                return JSONResponse(
                    status_code=400,
                    content={
                        "detail": "Email is already taken",
                        "data": None
                    }
                )

            hashed_password = hash_password(password)
            user = User(
                email=email,
                password=hashed_password,
                is_email_verified=False,
            )

            db.add(user)
            db.commit()
            db.refresh(user)
```

```
            # Token access creation
            token = create_access_token(
                {"sub": email}, expires_delta=timedelta(hours=24))

            # Sending email verification link
            await send_verification_link(email, token, locale)
            return JSONResponse(
                status_code=201,
                content={
                    "detail": "Register successful. Please verify your
email",
                    "data": {
                        "access_token": token,
                        "token_type": "bearer"
                    }
                }
            )

        except SQLAlchemyError as e:
            db.rollback()
            return JSONResponse(
                status_code=500,
                content={
                    "detail": "Database error",
                    "data": None
                }
            )


    #  Authenticate user and generate JWT token
    def authenticate_user(db: Session, email: str, password: str):
        user = db.query(User).filter(User.email == email).first()
        logging.debug(f"Retrieved user: {user}")

        # Check if user exists in DB
        if not user or not verify_password(password, user.password):

            if(not user):
```

```
                logging.debug(f"No user found with email: {email}")
            elif not verify_password(password, user.password):
                logging.debug(f"Authentication failed for user:
{email}")
            return JSONResponse(
                status_code=404,
                content={
                    "detail": "User not found or incorrect password",
                    "data": None
                }
            )

        if user.is_email_verified == False:
            return JSONResponse(
                status_code=403,
                content={
                    "detail": "Email not verified",
                    "data": None
                }
            )
        # Generate token for the user
        access_token = create_access_token(
            data={"sub": str(user.email)},
            expires_delta=timedelta(minutes=30)
        )

        return {
            "detail": "Authentication successful",
            "data": {
                "access_token": access_token,
                "token_type": "bearer"
            }
        }
```

ДОДАТОК Ж

Код обробки повідомленнь електронною поштою

```
    async def send_verification_code(email: str, user_id: int, locale: Optional[str] = "en"):
        """Send email with verification code."""

        # Generate a random verification code
        code = generate_verification_code()
        verification_codes[user_id] = code  # Store the code in the dictionary

        # Create the verification email content
        if locale == "en":
            subject = "Your verification code"
            body = f"Your verification code is: {code}"
        else:
            subject = "Ваш код підтвердження"
            body = f"Ваш код підтвердження: {code}"


        message = MessageSchema(
            subject=subject,
            recipients=[email],
            body=body,
            subtype="html",
        )

        # Send the email
        fm = FastMail(conf)
        await fm.send_message(message)

    async def send_verification_link(email: str, token: str, locale: Optional[str] = "en"):
        """
        Send email with verification link.
        :param email: email adress
        :param token: token for verification
```

```
        :param locale: language of the email content (default is 'en')
        """
        verification_url = f"{SERVER_URL}/verify_email?token={token}"

        if locale == "en":
            subject = "Email verification"
            body = f"Click the link to verify your email: <a href='{verification_url}'>{verification_url}</a>"
        else:
            subject = "Підтвердження електронної пошти"
            body = f"Перейдіть за посиланням, щоб підтвердити email: <a href='{verification_url}'>{verification_url}</a>"


        message = MessageSchema(
            subject=subject,
            recipients=[email],
            body=body,
            subtype="html",
        )

        fm = FastMail(conf)
        await fm.send_message(message)
    async def verify_code(user_id: int, code: str, db: Session) -> bool:
        logging.debug(f"Current list of verification codes: {verification_codes}")
        """Verifies the confirmation code for the given user_id and updates the email verification status in the database."""
        logging.debug(f"Verifying code {code} for user_id {user_id}")
        stored_code = verification_codes.get(user_id)
        logging.debug(f"Stored code {stored_code} for user_id {user_id}")

        if stored_code == code:
            logging.debug(f"Code confirmed for user_id {user_id}")
            del verification_codes[user_id]  # Remove the code after successful verification

            user = db.query(User).filter(User.id == user_id).first()
```

```
            if user:
                user.is_email_verified = True
                db.add(user)
                db.commit()
                db.refresh(user)
                logging.debug(f"User {user_id}'s email has been
verified in the database")
                return True

        logging.debug(f"Invalid code or user {user_id} not found")
        return False
```

ДОДАТОК З

Формування рекомендацій на основі паттерну Strategy

```
class RecommendationStrategy(ABC):
    @abstractmethod
    def evaluate(self, clothing_item: ClothingItem, **kwargs):
        pass


class WeatherRecommendationStrategy(RecommendationStrategy):
    def evaluate(self, clothing_item: ClothingItem, temp: float,
weather: str):
        return clothing_item.evaluate_weather_match(temp, weather,
TEMPERATURE_MISMATCH_COEF)


class ColorRecommendationStrategy(RecommendationStrategy):
    def evaluate(self, clothing_item: ClothingItem, other_color:
tuple[int, int, int], palette_type: str):
        return      clothing_item.evaluate_color_match(other_color,
palette_type)


class EventRecommendationStrategy(RecommendationStrategy):
    def evaluate(self, clothing_item: ClothingItem, event: str):
        return clothing_item.evaluate_event_match(event)


# --- Combined strategies for different scenarios ---


class ColorEventStrategy(RecommendationStrategy):
    def __init__(self):
        self.color_strategy = ColorRecommendationStrategy(
        )
        self.event_strategy = EventRecommendationStrategy()
```

```
        def evaluate(self, clothing_item, other_color, palette_type,
event):
            score_color = extract_score(self.color_strategy.evaluate(
                clothing_item, other_color, palette_type))
            score_event = extract_score(
                self.event_strategy.evaluate(clothing_item, event))
            if score_color is None or score_event is None:
                return 0
            return (score_color + score_event) / 2


    class WeatherEventStrategy(RecommendationStrategy):
        def __init__(self):
            self.weather_strategy = WeatherRecommendationStrategy(
            )
            self.event_strategy = EventRecommendationStrategy()

        def evaluate(self, clothing_item, temp, weather, event):
            score_weather                                          =
extract_score(self.weather_strategy.evaluate(
                clothing_item, temp, weather))
            score_event = extract_score(
                self.event_strategy.evaluate(clothing_item, event))
            if score_weather is None or score_event is None:
                return 0
            return (score_weather + score_event) / 2


    class ColorWeatherStrategy(RecommendationStrategy):
        def __init__(self):
            self.color_strategy = ColorRecommendationStrategy(
            )
            self.weather_strategy = WeatherRecommendationStrategy(
            )

        def evaluate(self, clothing_item, other_color, palette_type,
temp, weather):
            score_color = extract_score(self.color_strategy.evaluate(
                clothing_item, other_color, palette_type))
```

```
            score_weather =
extract_score(self.weather_strategy.evaluate(
                clothing_item, temp, weather))
            if score_color is None or score_weather is None:
                return 0
            return (score_weather + score_color) / 2


    class AverageRecommendationStrategy(RecommendationStrategy):
        def __init__(self):
            self.weather_strategy = WeatherRecommendationStrategy()
            self.color_strategy = ColorRecommendationStrategy()
            self.event_strategy = EventRecommendationStrategy()

        def evaluate(self, clothing_item, temp, weather, other_color,
palette_type, event):
            score_weather =
extract_score(self.weather_strategy.evaluate(
                clothing_item, temp, weather))
            score_color = extract_score(self.color_strategy.evaluate(
                clothing_item, other_color, palette_type))
            score_event = extract_score(
                self.event_strategy.evaluate(clothing_item, event))

            scores = [score for score in (
                score_weather, score_color, score_event) if score is
not None]

            if not scores:
                return 0
            return sum(scores) / len(scores)
```

ДОДАТОК И

Таблиці Тест-кейсів

Таблиця И.1 – Тест-кейс №1 (таблиця виконана самостійно)

Назва тесту:	*Test_app*	
Функція:	**Старт серверу**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Кроки тесту:		
Перевірити запуск сервера та доступ до корневого елементу "/"	Доступ до сервера надано	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 11.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.2 – Тест-кейс №2 (таблиця виконана самостійно)

Назва тесту:	*test_auth*	
Функція:	**Вхід користувача в систему**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу	База даних доступна	Пройдено

даних		
Кроки тесту:		
Створи запит до серверу з { "email": "test@gmail.com", "password": "pass" } з url = "/login_with_email"	Перевірити, що користувач успішно увійшов в систему	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 12.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.3 – Тест-кейс №3(таблиця виконана самостійно)

Назва тесту:	test_reg	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Кроки тесту:		
Створи запит на реєстрацію до серверу email, не зареєстрованим в базі даних.	Перевірити, що сервер приймає запит	Пройдено
Перевірити успішність запиту	Код відповіді рівний 200	Пройдено
Перевірити наявність користувача з заданим email	Користувача знайдено	Пройдено
Постумова:		

Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 13.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.4 – Тест-кейс №4(таблиця виконана самостійно)

Назва тесту:	test_sync	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на синхронізацію з тестовими даними	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що кількість даних користувча рівна кількості тестових даних в запиті	Кількість даних співпадає	Пройдено
Перевірити, що серед даних вказані саме ті тестові дані, які були передані	Дані перевірені	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 14.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.5 – Тест-кейс №5(таблиця виконана самостійно)

Назва тесту:	test_update_clothing_item	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на оновлення даних елементу одягу	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що дані оновилися	Дані оновлені	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 15.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.6 – Тест-кейс №6(таблиця виконана самостійно)

Назва тесту:	test_recommendations_weather_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено

Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі погоди	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації створені	Рекомендації створені	Пройдено
Перевірити, що для оцінки рекомендацій було використано weather_match	Було використано weather_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 16.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.7 – Тест-кейс №7(таблиця виконана самостійно)

Назва тесту:	test_recommendations_event_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі події	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації	Рекомендації	Пройдено

створені	створені	
Перевірити, що для оцінки рекомендацій було використано event_match	Було використано event_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 17.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**
Примітка:		

Таблиця И.8 – Тест-кейс №8(таблиця виконана самостійно)

Назва тесту:	test_recommendations_color_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі бажаної кольорової гами	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації створені	Рекомендації створені	Пройдено
Перевірити, що для оцінки рекомендацій було використано color_match	Було використано color_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути	Пройдено

	успішно пройдено	
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 17.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**

Таблиця И.9 – Тест-кейс №9(таблиця виконана самостійно)

Назва тесту:	test_recommendations_color_event_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі бажаної кольорової гами та події	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації створені	Рекомендації створені	Пройдено
Перевірити, що для оцінки рекомендацій було використано color_event_match	Було використано color_event_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 19.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**

Таблиця И.10 – Тест-кейс №10(таблиця виконана самостійно)

Назва тесту:	test_recommendations_weather_event_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі погоди та події	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації створені	Рекомендації створені	Пройдено
Перевірити, що для оцінки рекомендацій було використано weather_event_match	Було використано weather_event_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 19.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**

Таблиця И.11 – Тест-кейс №11(таблиця виконана самостійно)

Назва тесту:	test_recommendations_color_weather_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено

Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі погоди та бажаної кольорої гами	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації створені	Рекомендації створені	Пройдено
Перевірити, що для оцінки рекомендацій було використано color_weather_match	Було використано color_weather_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 21.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**

Таблиця И.12 – Тест-кейс №12 (таблиця виконана самостійно)

Назва тесту:	test_recommendations_average_match	
Функція:	**Реєстрація користувача**	
Дія	Очікуваний результат	Результат тесту: - пройдено (P); - провалено (F); -заблоковано(B).
Передумова:		
Створити тестову базу даних	База даних доступна	Пройдено
Викон запит на логін та отримати токен	Токен отримано	Пройдено
Кроки тесту:		
Створи запит на отримання рекомандації на основі погоди, бажаної кольорої гами та події	Перевірити, що сервер приймає запит	Пройдено
Перевірити, що рекомендації	Рекомендації	Пройдено

створені	створені	
Перевірити, що для оцінки рекомендацій було використано average_match	Було використано average_match	Пройдено
Постумова:		
Unit test пройдено	Unit test має бути успішно пройдено	Пройдено
Результати тесту		
Тестувальник: Замковий А.Г.	**Дата прогона теста:** 21.05.2025	**Результат теста (P/F/B): ПРОЙДЕНО (P)**