МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

(повна назва)

Кафедра програмної інженерії

(повна назва)

**КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЄКТ**

**Пояснювальна записка**

рівень вищої освіти перший (бакалаврський)

Веб-додаток для управління особистим гардеробом та створення образів “SmartStyle”

(тема)

Виконав:

здобувач 3 курсу, групи ПЗПІ-22-9

Артем ФІЛАТОВ

(Власне ім’я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення

(код і повна назва спеціальності)

Тип програми освітньо-професійна Освітня програма Програмна інженерія

(повна назва освітньої програми)

Керівник доц. кафедри ПІ Олександр ВЕЧУР

(посада, Власне ім’я, ПРІЗВИЩЕ)

|  |
| --- |
| Члени комісії (Власне ім’я, ПРІЗВИЩЕ, підпис) |
| Олександр ВЕЧУР |
| Наталія КРАВЕЦЬ |
| Андрій РАБОТЯГОВ |

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-професійна

Освітня програма Програмна Інженерія

Курс 3 Група ПЗПІ-22-9 Семестр 6

**ЗАВДАННЯ**

***на курсовий проект(роботу) студента***

здобувачеві Філатова Артема Дмитровича

1. Тема роботи Веб-додаток для управління особистим гардеробом та створення образів “SmartStyle”.

2. Термін здачі студентом закінченої роботи „ 20 ” червня 2025 р.

3. Вихідні дані до проєкту Методичні рекомендації до курсового проєктування, функціональні та нефункціональні вимоги до програмної системи, аналіз предметної області, що стосується інтерактивних систем управління гардеробом та алгоритмів підбору одягу з використанням штучного інтелекту.

4. Перелік питань, що потрібно опрацювати в роботі

Провести порівняльний аналіз існуючих рішень, спроєктувати та розробити повноцінний веб-додаток з нормальною формою бази даних, реалізувати ключовий функціонал клієнтської та серверної частин, провести тестування системи та сформулювати підсумкові висновки.

**КАЛЕНДАРНИЙ ПЛАН**



| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
| --- | --- | --- | --- |
| 1 | Аналіз предметної галузі | 15.03.25 – 27.03.25 | виконано |
| 2 | Розробка постановки задачі | 28.03.25 – 29.03.25 | виконано |
| 3 | Проектування ПЗ | 5.04.25 – 10.04.25 | виконано |
| 4 | Програмна реалізація | 13.04.25 – 15.05.25 | виконано |
| 5 | Аналіз результатів | 16.05.25 – 29.05.25 | виконано |
| 6 | Підготовка пояснювальної записки | 03.06.25 – 17.06.25 | виконано |
| 7 | Перевірка на наявність ознак академічного плагіату | 18.06.25 – 20.06.25 | виконано |
| 8 | Захист роботи | З 09.06.25 | виконано |

Дата видачі завдання “26” лютого 2025р.



Здобувач

(підпис)

Керівник роботи доц. кафедри ПІ Олександр ВЕЧУР

(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

**РЕФЕРАТ / ABSTRACT**

Пояснювальна записка містить: 69 с., 14 рис., 4 табл., 11 джерел.

УПРАВЛІННЯ ГАРДЕРОБОМ, ШТУЧНИЙ ІНТЕЛЕКТ, ВЕБ-ЗАСТОСУНОК, СИСТЕМА РЕКОМЕНДАЦІЙ, REACT, NODE.JS, EXPRESS, POSTGRESQL, GEMINI, OPENWEATHERMAP.

Предметом дослідження є програмне рішення для персоналізованого стайлінгу та оптимізації гардеробу "SmartStyle". Метою роботи є створення веб-системи, що оснащує користувачів функціоналом для систематизації власного одягу, візуального конструювання комбінацій та отримання індивідуальних порад, які генеруються логічним алгоритмом на основі метеорологічних даних та вказаних стильових переваг. Для реалізації серверної логіки було застосовано платформу Node.js у поєднанні з фреймворком Express, що забезпечує роботу REST API та зв'язок із базою даних PostgreSQL. Інтерфейс користувача побудовано за допомогою бібліотеки React з використанням компонентної бази Material-UI, що гарантує його динамічність та адаптивність. Механізм рекомендацій базується на алгоритмі, що аналізує відповідність призначених одягу тегів до актуальних погодних умов з сервісу OpenWeatherMap.Результатом роботи є готовий до використання веб-додаток, що надає користувачам змогу оптимізувати використання власного гардеробу, експериментувати зі створенням образів та одержувати обґрунтовані рекомендації, що допомагає подолати щоденні труднощі з вибором вбрання та заохочує до більш раціонального підходу до моди.

WARDROBE MANAGEMENT, ARTIFICIAL INTELLIGENCE, WEB APPLICATION, RECOMMENDATION SYSTEM, REACT, NODE.JS, EXPRESS, POSTGRESQL, GEMINI, OPENWEATHERMAP.

The subject of the research is a software solution for personalized styling and wardrobe optimization, "SmartStyle". The purpose of the work is to create a web-system that equips users with functionality for systematizing their own clothes, visually constructing combinations, and receiving individual advice generated by a logical algorithm based on meteorological data and specified style preferences. To implement the server-side logic, the Node.js platform was used in conjunction with the Express framework, which powers the REST API and communication with the PostgreSQL database. The user interface is built using the React library with the Material-UI component base, ensuring its dynamism and adaptability. The recommendation mechanism is based on an algorithm that analyzes the correspondence of tags assigned to clothes with current weather conditions from the OpenWeatherMap service. The result of the work is a ready-to-use web application that enables users to optimize the use of their own wardrobe, experiment with creating outfits, and receive well-founded recommendations, which helps overcome daily difficulties with choosing attire and encourages a more rational approach to fashion.

**ЗМІСТ**

[Вступ](#_jbgju3d5kpry)…………..…………..…………..…………..…………..…………..…………..………………………. [7](#_jbgju3d5kpry)

[1 Аналіз предметної галузі](#_z1f5fa5cr6eg)…………..…………..…………..…………..…………..………………… [9](#_z1f5fa5cr6eg)

[1.1 Аналіз предметної галузі…………..…………..…………..…………..………. 9](#_qxjyt5655026)

[1.2 Виявлення та вирішення проблем…………..…………..…………..………... 11](#_oldif6codz0b)

[1.3 Аналіз аналогів програмного забезпечення…………..…………..…………. 12](#_acot7lr482em)

[2 Постановка задачі](#_mnqhw3s4h745)……………………………………………………………………………………... [15](#_mnqhw3s4h745)

[3 Архітектура та проєктування програмного забезпечення](#_hfs34n28z3hn)…………..………………….. [17](#_hfs34n28z3hn)

[3.1 UML проєктування ПЗ…………..…………..…………..……………………. 17](#_rff199wtqyz0)

[3.2 Проєктування архітектури ПЗ…………..…………..…………..……………. 19](#_f6zo1oh47q53)

[3.3 Проєктування структури зберігання даних…………..…………..………….. 22](#_scuc5ymzmzdx)

[3.4 Приклади використаних алгоритмів та методів…………..………………… 26](#_awfr8p5jinqk)

[3.5 Проєктування UI/UX…………..…………..…………..…………..………….. 28](#_ji5p2asrjrgw)

[4 Опис прийнятих програмних рішень](#_a0cx736zaoex)…………..…………..…………..…………..………….. [33](#_a0cx736zaoex)

[5 Аналіз отриманих результатів](#_metf52xo67nx)…………..…………..…………..…………..…………..……….. [39](#_metf52xo67nx)

[5.1 Порівняння результатів із початковими вимогами…………..……………... 39](#_rw8rh5eq0etc)

[5.2 Оцінка якості роботи системи…………..…………..…………..……………. 39](#_u1d0ndytbmc)

[5.3 Аналіз ефективності прийнятих рішень…………..…………..……………... 40](#_mlu30zeazpfi)

[5.4 Обмеження та недоліки…………..…………..…………..…………..……….. 40](#_sdepiku8e66d)

[Висновки](#_bnb7eomv9dph)…………..…………..…………..…………..…………..…………..…………..…………..…… [41](#_bnb7eomv9dph)

Перелік джерел посилання…………..…………..…………..…………..…………… [42](#_ilqbv3rk0ufn)

Додаток А Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ……. [43](#_gnl4thlmiq74)

Додаток Б Специфікація ПЗ…………..…………..…………..…………..………….. [44](#_qfmbk545wfvp)

Додаток В Слайди презентації…………..…………..…………..…………..………. [56](#_mgbtpj47xdbb)

## 

## **ВСТУП**

У сучасному світі швидкої моди та надмірного споживання багато людей стикаються з парадоксом: маючи повну шафу одягу, вони щодня відчувають труднощі з вибором образу. Ця проблема, відома як "decision fatigue" (втома від прийняття рішень), посилюється мінливими погодними умовами, постійним соціальним тиском виглядати відповідно до обставин та необхідністю швидко приймати рішення в умовах обмеженого часу. Часто це призводить до ситуацій, коли людина, незважаючи на кількість доступного одягу, відчуває незадоволення своїм зовнішнім виглядом або вибором.

Традиційні методи організації гардеробу, такі як сортування за категоріями, списки чи прості мобільні додатки, часто є недостатньо ефективними. Вони не враховують динамічний контекст життя сучасної людини, який включає зміни погоди, різноманіття подій, особисті переваги, стильові вподобання, настрій та навіть графік дня. Тому виникає потреба в інтелектуальних системах, які здатні надати рекомендації, враховуючи всі ці фактори в режимі реального часу.

Розвиток веб-технологій, поява сучасних інструментів розробки та доступність API для отримання даних, зокрема погодних умов, відкривають нові можливості для ефективного вирішення цієї проблеми. Замість складних та не завжди зрозумілих для користувача AI-моделей, системи, що базуються на чітких правилах і прозорих алгоритмах, можуть генерувати логічно обґрунтовані, практичні та легко пояснювані рекомендації. Такий підхід дозволяє створити продукт, якому користувач може довіряти, оскільки він розуміє принципи його роботи.

Це створює унікальну можливість для розробки практичних, зручних та адаптивних систем-помічників у сфері стилю, які не тільки полегшують щоденний вибір одягу, а й сприяють більш усвідомленому підходу до формування гардеробу. Такі системи допомагають зменшити кількість спонтанних покупок, оптимізувати використання наявного одягу та розвивати особистий стиль.

Актуальність даної роботи полягає у необхідності створення зручного, інтуїтивно зрозумілого та інтелектуального інструменту, який би допомагав користувачам не лише каталогізувати свій одяг, а й отримувати розумні, контекстно-залежні поради, адаптовані до їхнього способу життя. Розробка веб-додатку "SmartStyle" з інтегрованою системою рекомендацій на основі правил дозволяє вирішити проблему щоденного вибору одягу, сприяє більш усвідомленому використанню наявного гардеробу, підвищує задоволеність користувача своїм стилем та допомагає ефективніше комунікувати через зовнішній вигляд.

Метою курсового проєкту є проєктування та розробка інтерактивного веб-додатку "SmartStyle", що надає користувачам персоналізований досвід управління гардеробом через візуалізацію вмісту, зручне створення образів та отримання алгоритмічних рекомендацій, які враховують як об'єктивні (погода, температура), так і суб'єктивні (настрій, стильові вподобання) фактори.

## **1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ**

### 1.1 Аналіз предметної галузі

Предметною галуззю даного курсового проєкту є ринок цифрових технологій для персонального стилю, а саме сегмент програмних продуктів для управління особистим гардеробом. В умовах сучасної культури, де зовнішній вигляд є важливим елементом самовираження та соціальної комунікації, ефективне керування власним одягом перетворилося з рутинної справи на значущу частину повсякденного життя. Це стимулює стабільно високий попит на зручні, функціональні та інтелектуальні інструменти, що робить дану галузь однією з динамічних та конкурентних у сфері розробки програмного забезпечення для споживачів.

Глобальний ринок додатків для управління гардеробом демонструє стабільне зростання. Згідно з прогнозами аналітичних агенцій, таких як Allied Market Research, очікується, що обсяг ринку Fashion Tech продовжить збільшуватися, із середньорічним темпом росту близько 10-15%. Ключовими драйверами цього росту є поширення смартфонів, зростання інтересу до сталого розвитку (sustainability) та усвідомленого споживання, а також прагнення користувачів до персоналізації. Сучасні споживачі шукають не просто цифрові каталоги одягу, а повноцінні екосистеми, що пропонують персоналізований та інтерактивний досвід.

Сучасні платформи для управління гардеробом можна класифікувати за основними підходами, що лежать в їх основі. Перший підхід, каталогізація, є базовим. Додатки, такі як Stylebook, дозволяють користувачам фотографувати свій одяг, видаляти фон та організовувати його за категоріями. Основна мета — створити повний цифровий опис реальної шафи для кращого огляду та планування.

Другий підхід — соціальний та натхненний. Платформи, як Pinterest або Instagram, хоч і не є спеціалізованими інструментами для гардеробу, використовуються для пошуку ідей та збереження образів. Користувачі створюють дошки (boards) з луками, які їм подобаються, але цей підхід відірваний від їхнього реального гардеробу.

Третій підхід — рекомендаційний. Він реалізується через додатки, що намагаються автоматично генерувати образи з речей користувача. Часто такі системи використовують AI для аналізу зображень або прості алгоритми. Проте якість таких рекомендацій буває низькою, оскільки вони не завжди враховують важливі контекстуальні фактори.

Ключовим трендом, що об'єднує та вдосконалює ці підходи, є інтеграція з зовнішніми даними та використання прозорої логіки. Замість непрогнозованого AI, системи на основі правил використовують теги та атрибути, присвоєні одягу, для зіставлення їх із даними про погоду, місцезнаходження та тип події. Саме на перетині детальної каталогізації та логічно обґрунтованих, контекстуальних рекомендацій і знаходиться предметна область даного проєкту.

### 1.2 Виявлення та вирішення проблем

Попри значну кількість додатків на ринку, багато з них не вирішують фундаментальних труднощів, з якими стикаються користувачі при управлінні своїм гардеробом. Аналіз цих труднощів дозволяє сформулювати ключові проблеми, на які орієнтований проєкт "SmartStyle".

Першочерговою проблемою є когнітивне перевантаження. Сучасна людина щодня приймає сотні рішень, і вибір одягу, хоч і здається тривіальним, є додатковим ментальним навантаженням. Маючи доступ до великої кількості речей, користувач часто стикається з "паралічем аналізу", що призводить до використання кількох звичних комбінацій та ігнорування решти гардеробу. "SmartStyle" вирішує це, виступаючи в ролі інтелектуального фільтра, який на основі чітких критеріїв пропонує обмежену кількість готових, релевантних образів, тим самим звільняючи користувача від необхідності самостійного перебору варіантів.

Друга системна проблема — це "ефект сліпої зони" в гардеробі. Речі, що не знаходяться безпосередньо перед очима, з часом забуваються і випадають з активного використання, хоча вони можуть бути цілком актуальними. Це призводить до неефективного використання ресурсів та помилкового відчуття "нема чого вдягнути". Розроблюваний додаток, маючи повну цифрову інвентаризацію гардеробу, здатен повертати в активний обіг забуті елементи одягу, систематично включаючи їх до рекомендацій та нагадуючи користувачеві про їх існування.

Третя, і чи не найважливіша, проблема — це статичність більшості цифрових шаф. Вони існують у вакуумі, ігноруючи динамічний зовнішній світ. Рекомендація, яка не враховує поточну температуру за вікном чи ймовірність опадів, є непрактичною і не несе реальної цінності. "SmartStyle" вирішує цю проблему через глибоку інтеграцію з погодними сервісами. Система правил динамічно адаптує образи до поточних метеорологічних умов, гарантуючи, що запропонований одяг буде не лише стильним, але й комфортним.

### 1.3 Аналіз аналогів програмного забезпечення

Щоб визначити конкурентні переваги та унікальну ринкову позицію проєкту "SmartStyle", було виконано порівняльний огляд ключових гравців у сегменті додатків для управління гардеробом.

Stylebook можна охарактеризувати як потужний інструмент для скрупульозних користувачів, які готові докладати зусиль для детальної каталогізації. Його сильна сторона — аналітичні можливості: відстеження частоти носіння речей, планування гардеробу в календарі. Однак цей функціонал вимагає значної ручної праці та дисципліни від користувача, при цьому додаток не пропонує жодної автоматичної інтелектуальної допомоги у формуванні образів.

Cladwell, навпаки, пропагує ідеологію мінімалізму та "капсульного гардеробу", пропонуючи щоденні готові образи. Цей підхід ефективно бореться з втомою від вибору, але його недоліком є певна ригідність. Система часто пропонує одноманітні комбінації та не залишає простору для творчих експериментів, обмежуючи користувача рамками заданої концепції.

Whering намагається поєднати сучасний інтерфейс та технології штучного інтелекту для розпізнавання одягу. Його привабливий дизайн та безкоштовна модель залучають широку аудиторію. Проте, на практиці, його AI-рекомендації часто критикують за нелогічність та невідповідність запитам користувачів, що свідчить про недосконалість "чорної скриньки" в основі його алгоритмів.

Pinterest не є спеціалізованим додатком для гардеробу, але слугує невичерпним джерелом візуального натхнення. Це абстрактний інструмент для формування мудбордів та дослідження стилів. Його головний недолік полягає у відсутності будь-якого зв'язку з реальними речами користувача, що перетворює його на каталог ідей, а не на практичного помічника.

Підсумовуючи аналіз, можна зробити висновок про існування очевидної ніші для продукту, що поєднує персоналізовану каталогізацію з прозорим та надійним механізмом рекомендацій. На відміну від непрогнозованих AI та занадто жорстких капсульних систем, "SmartStyle" пропонує користувачеві контроль та розуміння процесу. Він отримує не "магічний" результат, а логічно обґрунтовані поради, що базуються на його власних даних та реальних зовнішніх умовах.

## **2 ПОСТАНОВКА ЗАДАЧІ**

Центральним завданням проєкту є розробка інтерактивної веб-платформи "SmartStyle", що виконує роль персонального асистента з управління гардеробом. Програмний продукт має надати користувачам інструментарій для цифрової інвентаризації одягу та отримання логічно обґрунтованих рекомендацій. Система повинна оперувати такими сутностями, як профіль користувача, елементи одягу, користувацькі теги та збережені комплекти (пресети)

Етапи реалізації:

* проєктування та розробка серверної частини (backend) та програмного інтерфейсу (API);
* створення клієнтської частини (frontend) з інтуїтивним та адаптивним дизайном;
* імплементація функціоналу віртуального гардеробу (додавання, тегування, видалення);
* розробка алгоритмічної системи рекомендацій на основі правил;
* інтеграція зі стороннім погодним сервісом OpenWeatherMap API;
* проведення тестування ключових сценаріїв використання.

Дані, що зберігаються:

* предмети одягу: назва, URL зображення, категорія, набір тегів;
* збережені набори: назва образу, перелік ідентифікаторів речей;
* ключові функціональні вимоги;
* забезпечення можливості реєстрації та автентифікації користувачів;
* генерація рекомендацій щодо образу на основі детермінованого алгоритму, що аналізує відповідність тегів поточним погодним умовам;
* надання актуальної інформації про погоду у вибраній локації;
* надання інструментів для візуального конструювання та збереження образів.

## **3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

### 3.1 UML проєктування ПЗ

На початковому етапі проєктування, з метою візуалізації функціональних вимог системи з точки зору кінцевого користувача, було сконструйовано діаграму варіантів використання (рис. 3.1). Дана модель призначена для опису фундаментальних сценаріїв взаємодії з додатком "SmartStyle" та окреслює межі взаємодії між актором (користувачем) і системою, що розробляється. Изображение выглядит как текст, рисунок, диаграмма, зарисовка

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.1 – Діаграма прецедентів веб-додатку

Діаграма прецедентів ілюструє наступні ключові взаємодії:

Користувач – це головний актор системи. Він є узагальненим представленням будь-якої особи, що зареєструвалася в додатку, та має намір використовувати його функціонал для управління своїм гардеробом.

Реєстрація та Автентифікація – це базовий та початковий варіант використання. Він охоплює процеси створення нового облікового запису (реєстрація) та входу до системи за допомогою існуючих облікових даних (автентифікація). Важливо зазначити наявність відношення <<include>>, яке вказує, що доступ до всіх інших персоналізованих функцій є неможливим без попереднього успішного проходження автентифікації. Це гарантує безпеку та конфіденційність персональних даних.

Керування профілем – ця функція дозволяє користувачеві переглядати та редагувати власну інформацію, таку як відображуване ім'я. Це забезпечує персоналізацію досвіду користування додатком.

Керування гардеробом – одна з центральних функцій системи. Вона надає користувачеві повний інструментарій для створення цифрової копії своєї шафи: додавання нових предметів одягу шляхом завантаження їх зображень, перегляд списку наявних речей та видалення тих, що більше не актуальні.

Керування тегами – ця функція дозволяє створювати, переглядати та призначати одягу персоналізовані теги. Теги є основою для роботи рекомендаційної системи, оскільки вони дозволяють класифікувати одяг за такими критеріями, як "офісний стиль", "для прогулянки", "тепла погода", "дощовий день" тощо.

Візуальне створення образу – надає користувачеві інтерактивний інтерфейс ("манекен"), на якому можна візуально комбінувати різні предмети одягу зі свого гардеробу, щоб оцінити їх поєднання.

Збереження пресету – після створення вдалої комбінації одягу користувач може зберегти її як готовий набір (пресет) для швидкого доступу в майбутньому. Це звільняє від необхідності щоразу створювати образ заново.

Отримання AI-рекомендацій – ініціює процес генерації пропозицій щодо образу. Система, використовуючи інтеграцію з Google Gemini API, аналізує гардероб користувача, присвоєні теги та зовнішні дані (погоду), щоб запропонувати релевантні та обґрунтовані комбінації.

Перегляд даних про погоду – дозволяє користувачеві отримувати актуальну інформацію про погодні умови у вибраному місті. Ця функція є невід'ємною частиною процесу прийняття рішення щодо вибору одягу, роблячи його більш усвідомленим та практичним.

Для більш глибокого розуміння поведінки системи в динаміці, особливо в інтерактивних її частинах, доцільно використовувати діаграму станів. Вона моделює життєвий цикл об'єкта або компонента системи, показуючи, як він переходить з одного стану в інший у відповідь на певні події. Хоча в проєкті "SmartStyle" немає класичних "вправ з питаннями", ми можемо адаптувати цю ідею для моделювання процесу отримання рекомендації за системою правил, який є схожим за своєю інтерактивною природою.

На рисунку 3.2 представлена діаграма станів, яка моделює життєвий цикл процесу генерації Rules-Based-рекомендації для користувача.

Изображение выглядит как текст, диаграмма, План, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.2 – Діаграма станів для процесу отримання Rules Based-рекомендації

Діаграма станів (рис. 3.2) детально описує поведінку системи під час одного циклу генерації рекомендації:

* початковий стан – система перебуває у стані «Очікування запиту». Інтерфейс готовий до взаємодії, користувач може вказати критерії для образу (погода, подія, стиль);
* перехід до генерації – коли користувач натискає кнопку "Згенерувати образ", система переходить у стан «Генерація рекомендації». Цей стан є композитним і включає в себе два підстани:
* «запит до Gemini API»: бекенд формує та відправляє запит до зовнішнього сервісу Google Gemini API;
* «обробка відповіді»: система очікує та обробляє відповідь від API. На фронтенді в цей час відображається індикатор завантаження;
* перехід до результату – у разі успішного отримання відповіді від API, система переходить у стан «Відображення результату». На екрані з'являється запропонований образ та текстовий опис. Користувачеві стають доступні опції "Зберегти" цей образ або "Спробувати ще";
* перехід до помилки – якщо під час генерації виникає помилка (наприклад, недоступність API, некоректна відповідь, таймаут), система переходить у стан «Відображення помилки». Користувач бачить відповідне повідомлення та має можливість повернутися до попереднього етапу, натиснувши "Спробувати ще";
* повернення в очікування – з станів «Відображення результату» або «Відображення помилки» система повертається у початковий стан «Очікування запиту» після відповідної дії користувача, готуючись до нового циклу.

### 3.2 Проєктування архітектури ПЗ

Архітектура системи "SmartStyle" побудована на основі перевіреної часом трирівневої клієнт-серверної моделі. Вибір цього архітектурного підходу зумовлений його перевагами: чітке розмежування відповідальності (separation of concerns), що спрощує розробку та тестування; підвищена гнучкість, що дозволяє незалежно модифікувати та масштабувати клієнтську та серверну частини; та покращена безпека, оскільки вся бізнес-логіка та доступ до даних інкапсульовані на сервері.

Взаємодія між клієнтською та серверною частинами відбувається через чітко визначений програмний інтерфейс (API), реалізований за принципами REST (Representational State Transfer). Дані для обміну між компонентами серіалізуються у формат JSON (JavaScript Object Notation), що є галузевим стандартом для сучасних веб-додатків завдяки своїй легкості та простоті обробки як на клієнті, так і на сервері. Такий підхід забезпечує слабку зв'язність (loose coupling) між клієнтом і сервером, що дозволяє розвивати їх незалежно.

Це означає, що для системи "SmartStyle" можна легко створювати різні клієнтські застосунки (наприклад, веб-додаток, мобільний додаток для iOS та Android) без необхідності вносити зміни в серверну логіку. Усі вони будуть взаємодіяти з єдиним сервером через той самий API, що значно скорочує час та вартість розробки нових продуктів. На діаграмі компонентів, що наведена нижче, візуалізовано не лише самі компоненти, а й інтерфейси, через які вони взаємодіють.

Для детального опису структури програмного забезпечення використовується діаграма компонентів. Вона візуалізує основні програмні блоки, їх внутрішню структуру та залежності між ними. Діаграма компонентів для системи "SmartStyle" представлена на рисунку 3.3.

Изображение выглядит как текст, диаграмма, снимок экрана, План

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.3 – Діаграма компонентів системи

Як показано на рисунку 3.3, архітектура системи складається з наступних логічних компонентів:

* frontend (React App): Клієнтська частина, яка є точкою входу для користувача. Цей компонент відповідає за візуальний інтерфейс та взаємодію. Він, у свою чергу, складається з підкомпонентів для управління UI, станом додатку та комунікацією з сервером;
* backend (Node.js/Express): Серверна частина, що є ядром системи. Вона надає RESTful API, через яке фронтенд отримує доступ до даних та бізнес-логіки. Внутрішні компоненти бекенду (маршрутизація, контролери, сервіси, DAL) забезпечують структуровану обробку запитів;
* postgreSQL [7]: Компонент бази даних, що відповідає за персистентне зберігання всієї інформації;
* зовнішні сервіси (OpenWeatherMap API, Google Gemini API):
* незалежні компоненти, з якими взаємодіє бекенд для отримання додаткових даних.

Для ілюстрації фізичного розміщення цих компонентів на апаратних ресурсах використовується діаграма розгортання. Вона показує, як програмні артефакти розподіляються по вузлах системи. Діаграма розгортання для "SmartStyle" представлена на рисунку 3.4.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.4 – Діаграма розгортання системи

Діаграма розгортання (рис. 3.4) ілюструє фізичну архітектуру системи:

* клієнтський пристрій: Це може бути будь-який пристрій користувача (ПК, ноутбук, смартфон) з встановленим веб-браузером. На цьому вузлі розгортається артефакт Frontend (React App), який завантажується з сервера додатку;
* сервер додатку: Фізичний або віртуальний сервер, на якому виконується артефакт Backend (Node.js/Express). Цей вузол обробляє всю бізнес-логіку;
* сервер бази даних: Окремий сервер (або той же, що і сервер додатку, у випадку простого розгортання), на якому розгорнуто СУБД PostgreSQL. Взаємодія з сервером додатку відбувається через мережевий протокол;
* зовнішні сервіси (Хмара): Це логічний вузол, що представляє сервери сторонніх провайдерів (OpenWeatherMap, Google), які надають свої API через Інтернет.

Така архітектура дозволяє гнучко масштабувати систему: наприклад, при зростанні навантаження можна додати більше серверів додатку або винести сервер бази даних на більш потужне обладнання, не змінюючи при цьому логіку роботи компонентів.

### 3.3 Проєктування структури зберігання даних

Основою будь-якої інформаційної системи є надійна та добре продумана структура зберігання даних. Для проєкту "SmartStyle" було спроєктовано реляційну модель даних, яка оптимально підходить для СУБД PostgreSQL. Головними цілями при проєктуванні були забезпечення цілісності даних, мінімізація їх надлишковості (нормалізація) та гарантування високої продуктивності запитів.

Схема бази даних розроблена для ефективного зберігання інформації про користувачів, їхні гардероби, персоналізовані теги, збережені образи (пресети) та всі зв'язки між цими сутностями. Нижче наведено детальний опис структури ключових таблиць.

Таблиця 3.1 – Специфікація таблиці «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля | Тип даних | Обмеження |
| 1 | id | SERIAL | PRIMARY KEY, NOT NULL |
| 2 | username | VARCHAR(255) | UNIQUE, NOT NULL |
| 3 | email | VARCHAR(255) | UNIQUE, NOT NULL |
| 4 | password\_hash | VARCHAR(255) | NOT NULL |
| 5 | display\_name | VARCHAR(255) | NULL |
| 6 | created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |
| 7 | Назва поля | Тип даних | Обмеження |
| 8 | id | SERIAL | PRIMARY KEY, NOT NULL |

Таблиця users (табл. 3.1) є центральною в системі. Вона зберігає всю необхідну інформацію для автентифікації та ідентифікації користувачів.

Використання обмеження UNIQUE для полів username та email гарантує відсутність дублікатів, а зберігання паролів у вигляді хешів є обов'язковою практикою для забезпечення безпеки.

Таблиця 3.2 – Специфікація таблиці “Clothing Items”

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Обмеження | Опис |
| id | SERIAL | PRIMARY KEY, NOT NULL | Унікальний ідентифікатор предмета одягу |
| user\_id | INTEGER | NOT NULL, FOREIGN KEY (users.id) | Зовнішній ключ, що посилається на власника одягу |
| name | VARCHAR(255) | NOT NULL | Назва предмета одягу (напр., "Синя сорочка") |
| image\_url | VARCHAR(255) | NOT NULL | URL-адреса для доступу до зображення одягу |
| category | VARCHAR(50) | NOT NULL | Категорія одягу (напр., "взуття", "верхній одяг", "низ") |
| added\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Час та дата додавання предмета до гардеробу |

Таблиця clothing\_items (табл. 3.2) призначена для зберігання каталогу одягу кожного користувача. Поле user\_id встановлює чіткий зв'язок "один-до-багатьох" з таблицею users, гарантуючи, що кожен предмет належить конкретному користувачеві.

Таблиця 3.3 – Специфікація таблиці “Tags”

| Назва поля | Тип даних | Обмеження | Опис |
| --- | --- | --- | --- |
| id | SERIAL | PRIMARY KEY, NOT NULL | Унікальний ідентифікатор тега |
| user\_id | INTEGER | NOT NULL, FOREIGN KEY (users.id) | Зовнішній ключ, що посилається на творця тега |
| name | VARCHAR(50) | NOT NULL | Назва тега (напр., "офіс", "літо", "дощова погода") |

Таблиця tags (табл. 3.3) дозволяє користувачам створювати власну систему класифікації одягу. Це є основою для гнучкої фільтрації та роботи рекомендаційної системи. Кожен тег також прив'язаний до користувача, що забезпечує персоналізацію.

Для реалізації зв'язків "багато-до-багатьох" між предметами одягу, тегами та збереженими образами використовуються додаткові асоціативні таблиці (clothing\_item\_tags, outfit\_sets, outfit\_set\_items), які не деталізовані тут для стислості, але є невід'ємною частиною загальної схеми.

Для візуалізації спроєктованої структури та взаємозв'язків між таблицями наведено схему бази даних (рис. 3.5).

Изображение выглядит как текст, диаграмма, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.5 – Схема реляційної бази даних

Запропонована структура бази даних є логічно завершеною та повністю відповідає поставленим завданням. Нормалізований підхід до проєктування, де кожна сутність винесена в окрему таблицю, запобігає дублюванню даних та забезпечує їх цілісність на рівні СУБД за допомогою зовнішніх ключів. Використання асоціативних таблиць clothing\_item\_tags та outfit\_set\_items є ефективним рішенням для реалізації складних зв'язків "багато-до-багатьох", що є критично важливим для функціоналу персоналізації образів та роботи системи рекомендацій.

Таким чином, розроблена схема створює надійний фундамент для зберігання інформації, що дозволяє системі ефективно оперувати даними користувачів, їхніми гардеробами та вподобаннями, а також забезпечує можливості для подальшого розширення функціоналу без необхідності кардинальної перебудови архітектури даних.

### 3.4 Приклади використаних алгоритмів та методів

Для реалізації ключових функціональних можливостей системи "SmartStyle" було розроблено та застосовано низку алгоритмів. Особливу увагу було приділено логіці взаємодії з інтелектуальними сервісами, оскільки вона є однією з центральних у додатку. Для візуалізації цих процесів використовуються діаграми діяльності, які детально показують послідовність дій та умови переходів.

Одним з найважливіших сценаріїв є генерація персоналізованої рекомендації щодо образу за допомогою AI та Rules Based System. Цей процес є складним і включає перевірку даних, взаємодію із зовнішнім API та обробку різних результатів. Діаграма діяльності на рисунку 3.6 наочно демонструє покроковий алгоритм цієї взаємодії.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.6 – Діаграма діяльності для генерації рекомендацій на основі правил

Як детально ілюструє діаграма діяльності на рисунку 3.6, алгоритм генерації рекомендації складається з таких послідовних кроків та умовних розгалужень:

* ініціація процесу: Алгоритм активується, коли користувач натискає кнопку "Згенерувати образ" в інтерфейсі. На цьому етапі система збирає всі вказані користувачем критерії: дані про погоду, обраний тип події та бажаний стиль;
* валідація даних: Перш ніж звертатися до зовнішнього сервісу, система виконує внутрішню перевірку. Ключовою умовою є наявність у гардеробі користувача достатньої кількості речей для формування образу. Цей крок є критично важливим для запобігання безглуздих запитів до API та для забезпечення якісного результату;
* сценарій недостатньої кількості одягу: Якщо перевірка показує, що в гардеробі замало речей (наприклад, менше 5-ти предметів у різних категоріях), потік виконання переривається. Система виводить користувачеві інформативне повідомлення з проханням доповнити свій віртуальний гардероб. Це покращує користувацький досвід, оскільки пояснює причину неможливості виконати дію;
* сценарій успішної валідації: Якщо одягу достатньо, система переходить до наступного кроку – формування промпту. Це структурований текстовий запит до AI, який містить всю необхідну контекстну інформацію: опис погодних умов, мету (наприклад, "ділова зустріч"), стильові побажання та перелік доступних предметів одягу з їхніми тегами.

Взаємодія з API. Процес взаємодії з API починається з надсилання сформованого промпту до сервісу Google Gemini, після чого система переходить у режим очікування відповіді. Подальша логіка залежить від отриманого результату. У разі успішної відповіді, система парсить отримані дані у форматі JSON, виокремлює з них рекомендовані предмети одягу та текстовий опис, а потім відображає готовий образ в інтерфейсі відповідно до визначених правил. На цьому етапі сценарій вважається успішно завершеним. Якщо ж API повертає помилку або не відповідає протягом встановленого часу, система інформує користувача про тимчасову недоступність сервісу та пропонує повторити спробу пізніше.

Іншим важливим алгоритмом є процес додавання нового предмету одягу до гардеробу, оскільки він включає взаємодію з файловою системою (або хмарним сховищем) та оновлення бази даних. Цей процес показаний на рисунку 3.7.

Изображение выглядит как текст, диаграмма, чек, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.7 – Діаграма діяльності для додавання одягу до гардеробу

Цей алгоритм (рис. 3.7) забезпечує надійне та послідовне додавання нових елементів до персонального каталогу користувача. Він починається з ініціативи користувача, проходить етапи валідації, фізичного завантаження файлу і, нарешті, завершується оновленням записів у базі даних, що гарантує цілісність інформації.

### 3.5 Проєктування UI/UX

Проєктування користувацького інтерфейсу (UI) та досвіду взаємодії (UX) є ключовим аспектом створення успішного продукту, оскільки саме від них залежить, наскільки зручним, зрозумілим та приємним буде використання додатку "SmartStyle". Головна мета полягала у створенні чистого, інтуїтивно зрозумілого та естетично привабливого дизайну, який би не перевантажував користувача і дозволяв швидко виконувати основні завдання.

Колірна палітра та візуальний стиль. Основою візуального стилю була обрана темна тема з акцентами на глибоких фіолетових та пурпурних відтінках, як це показано на прикладі головної сторінки (рис. 3.8). Такий вибір зумовлений кількома причинами. По-перше, з точки зору естетики та фокусу, темна палітра створює відчуття елегантності та сучасності, дозволяючи яскравим зображенням одягу бути в центрі уваги. По-друге, такий інтерфейс забезпечує комфорт для очей, знижуючи навантаження при використанні додатку в умовах слабкого освітлення. Нарешті, психологія кольору також відіграє роль: фіолетовий колір асоціюється з креативністю та індивідуальністю, що добре корелює з тематикою моди.

Структура та навігація. Інтерфейс спроєктовано за модульним принципом з чітко визначеними функціональними зонами. Основна навігація реалізована через верхню панель (хедер), яка залишається видимою на всіх сторінках і надає швидкий доступ до ключових розділів: «Home» (Головна), «Wardrobe» (Гардероб), «Clothes Fitter» (Примірка образів), «Timeline» (Планувальник) та «About» (Про нас). У правій частині хедеру традиційно розташовані елементи керування профілем, кнопка виходу з системи та перемикач мови. Такий підхід є звичним для користувачів, що знижує поріг входження.

Локалізація. Важливою частиною UX є повна локалізація інтерфейсу. Система підтримує дві мови: українську та англійську. Користувач може легко перемикатися між ними за допомогою випадаючого меню в хедері. Всі елементи інтерфейсу, включаючи навігаційні посилання, заголовки, кнопки та системні повідомлення, перекладені, що робить додаток доступним для ширшої аудиторії.

Адаптивність. Оскільки користувачі можуть взаємодіяти з додатком з різних пристроїв, велику увагу було приділено адаптивному дизайну. Всі елементи інтерфейсу та сітки автоматично перебудовуються залежно від ширини екрана, забезпечуючи оптимальний досвід. Для реалізації адаптивності активно використовувались можливості фреймворку Material-UI та CSS Flexbox/Grid.

Ключові екрани та компоненти:

* сторінка входу та реєстрації: Мінімалістичний дизайн з чіткими полями вводу та помітними кнопками заклику до дії, щоб максимально спростити процес створення облікового запису;
* головна панель (Dashboard): Перший екран, який бачить користувач після входу (рис. 3.8). Він містить віджети з поточною погодою, швидким доступом до генерації образу та, можливо, останніми збереженими пресетами;
* сторінка "Мій гардероб": Реалізована у вигляді сітки з картками, кожна з яких представляє окремий предмет одягу. Картка містить зображення, назву та ключові теги. На сторінці є зручні фільтри за категоріями та тегами;
* інтерфейс створення образу ("Clothes Fitter"): Центральним елементом є візуальний "манекен" зі слотами для різних типів одягу (головний убір, верх, низ, взуття). Поряд розташована панель з усіма доступними речами з гардеробу, які можна перетягувати (drag-and-drop) у відповідні слоти;
* модальні вікна: Для таких дій, як додавання нового одягу, редагування профілю або перегляду деталей, використовуються модальні вікна. Це дозволяє користувачеві виконувати дію, не покидаючи поточного контексту сторінки.

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.8 – Інтерфейс головної сторінки

Варто зазначити, що представлені у цьому розділі рисунки відображають переважно ключові сторінки, в той час як повний функціонал веб-додатку був реалізований в повному обсязі, як це було описано в попередніх розділах.

Проєктування інтерфейсу було зосереджено на ключових користувацьких сценаріях: швидке додавання речей до віртуального гардеробу, легкий пошук та фільтрація, а також інтуїтивне створення образів. Кожен екран розроблявся з урахуванням логіки користувача, щоб мінімізувати кількість необхідних дій для досягнення мети.

Особливу увагу було приділено узгодженості візуального стилю. Всі елементи керування, від кнопок до випадаючих списків, мають однаковий вигляд та поведінку на всіх сторінках додатку. Це створює відчуття передбачуваності та надійності системи. Кольорова гама та типографіка були підібрані таким чином, щоб забезпечити високу читабельність і не відволікати від головного — контенту користувача.

Загалом, дизайн "SmartStyle" спрямований на створення цілісного, функціонального та емоційно привабливого продукту, який допомагає користувачеві вирішувати його щоденні завдання з мінімальними зусиллями.

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.9 – Інтерфейс сторінки для завантаження одягу

З точки зору проєктування інтерфейсу, найбільше уваги вимагала сторінка візуального конструктора образів. Завданням було організувати велику кількість функціональних елементів у єдиний, зрозумілий простір. Для цього було застосовано багатопанельне компонування. Головна область відведена під інтерактивну модель-аватар, куди користувач може перетягувати елементи одягу.

Изображение выглядит как текст, снимок экрана, фиолетовый, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.10 – Інтерфейс сторінки для генерації та підбору стильних образів

Бічна панель слугує галереєю всього доступного гардеробу, оснащеною системою швидкого пошуку та фільтрації.

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.11 – Інтерфейс обрання часового відрізку для обрання специфічних умов

Окрема секція інструментів надає доступ до функцій, пов'язаних з поточним образом, таких як запит до AI-помічника або збереження фінальної комбінації одягу.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.12 – Інтерфейс сторінки “Про нас”

Таке розділення робочого простору дозволяє уникнути візуального безладу та забезпечує логічний потік дій: від вибору окремих речей до створення та фіксації цілісного вигляду.

## **4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ**

При розробці веб-додатку "SmartStyle" було прийнято низку ключових архітектурних та технологічних рішень, спрямованих на створення сучасного, гнучкого та масштабованого продукту. Вибір стеку технологій базувався на таких критеріях, як продуктивність, зручність розробки, наявність великої спільноти та екосистеми інструментів.

Технологічний стек:

* frontend: Для клієнтської частини було обрано бібліотеку React, яка дозволяє створювати динамічні та інтерактивні користувацькі інтерфейси за допомогою компонентного підходу. Для навігації використовується React Router, для керування станом — вбудовані хуки React (useState, useContext), а для візуального оформлення — бібліотека компонентів Material-UI [9] (MUI) , що забезпечує консистентний та адаптивний дизайн;
* backend: Серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js. Такий вибір дозволяє побудувати швидкий та легкий RESTful API. Асинхронна природа Node.js [5] ідеально підходить для обробки великої кількості одночасних запитів та взаємодії зі сторонніми сервісами;
* база даних: В якості системи управління базами даних було обрано PostgreSQL — потужну об'єктно-реляційну СУБД, відому своєю надійністю, розширюваністю та підтримкою складних запитів, що є важливим для реалізації логіки додатку;
* зовнішні API: Для розширення функціональності були інтегровані OpenWeatherMap API (для отримання даних про погоду) [10] та Google Gemini API (для генерації інтелектуальних текстових рекомендацій) [8].

Нижче детально розглянуто реалізацію чотирьох ключових функцій, що демонструють застосування обраних технологій та підходів на практиці.

4.1 Генерація образу за допомогою Rules Based System та AI

Ця функція є однією з центральних у "SmartStyle" і дозволяє користувачеві отримати рекомендації щодо одягу, адаптовані до погодних умов, а також до запланованої активності та місця призначення. Реалізація цієї функціональності передбачає складну взаємодію між фронтендом, бекендом та зовнішніми API.

Процес відбувається наступним чином:

* збір вхідних даних: Фронтенд надсилає запит на бекенд, що містить інформацію про місто, часовий проміжок та мету (наприклад, "робота", "прогулянка в парку");
* отримання погодних даних: Бекенд, використовуючи спеціальний модуль, звертається до OpenWeatherMap API для отримання детального погодинного прогнозу[10];
* формування промпту для AI: На основі отриманих погодних даних та мети користувача, бекенд формує розгорнутий текстовий промпт, використовуючи попередньо визначені шаблони;
* запит до Google Gemini API: Сформований промпт надсилається до Google Gemini API[8]. AI обробляє промпт і генерує текстову рекомендацію щодо образу;
* надання рекомендації: Отримана відповідь від AI повертається на фронтенд, де вона відображається користувачеві.

На рисунку 4.1 показано приклад реалізованого інтерфейсу та виконання функції генерації образу на сторінці примірки образів, де користувач ініціював генерацію комбінації одягу.

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.1 – Інтерфейс сторінка з згенерованою комбінацією одягу

Приклад коду демонструє логіку на стороні бекенду, яка об'єднує дані про погоду та запити до AI.

Контролер timelineController.js (Backend):

import { getWeather } from './weatherApi.js';

import { getAIResponse } from './geminiApi.js';

import { prompts } from './prompts.js';

export async function getTimelineSuggestion(req, res) {

try {

const { city, startTime, endTime, goal, timeline } = req.body;

if (!city || !goal || (!timeline && (!startTime || !endTime))) {

return res.status(400).json({ error: 'city, goal, and either timeline or startTime/endTime are required' });

}

// Fetch weather forecast for the city

const weatherData = await getWeather(city);

let selectedHours = [];

let timeString = '';

if (timeline && Array.isArray(timeline) && timeline.length > 0) {

// Тут можемо бачити використання обраного часу та фільтрації., ['16:00', '18:00', ...])

selectedHours = weatherData.hourly.filter(hour => timeline.includes(hour.time));

timeString = timeline.join(', ');

} else {

// Fallback: use startTime/endTime logic

selectedHours = weatherData.hourly.filter(hour => {

const hourNum = parseInt(hour.time.split(':')[0], 10);

const startNum = parseInt(startTime.split(':')[0], 10);

const endNum = parseInt(endTime.split(':')[0], 10);

if (endNum < startNum) {

return hourNum >= startNum || hourNum <= endNum;

}

return hourNum >= startNum && hourNum <= endNum;

});

timeString = `${startTime} to ${endTime}`;

}

// Підсумовуємо погодні параметри щоб зробити правильний промпт з вже готових шаблонів.

const temps = selectedHours.map(h => h.temperature);

const avgTemp = temps.length ? Math.round(temps.reduce((a, b) => a + b, 0) / temps.length) : weatherData.current.temperature;

const conditions = [...new Set(selectedHours.map(h => h.description))].join(', ');

const weatherSummary = `${avgTemp}°C, ${conditions}`;

const promptTemplate = prompts[0][1].prompt;

const prompt = `${promptTemplate}\nDestination: ${goal}\nCity: ${city}\nTime: ${timeString}\nWeather: ${weatherSummary}`;

const aiResult = await getAIResponse(prompt);

res.json({ suggestion: aiResult, prompt });

} catch (error) {

res.status(500).json({ error: error.message || 'Failed to get timeline suggestion' });

}

}

4.2 Інтерактивне компонування образу на манекені

Ця функція дозволяє користувачеві інтерактивно створювати образи, розміщуючи предмети одягу на віртуальному манекені. Манекен поділений на "слоти" (головний убір, верхній одяг, основний верх, низ, взуття). Логіка керування станом манекена інкапсульована у кастомному React-хуці.

Хук useMannequinSlots.js (Frontend):

export function useMannequinSlots() {

const [headwear, setHeadwear] = useState(null);

const [outerLayer, setOuterLayer] = useState(null);

const [mainTop, setMainTop] = useState(null);

const [bottoms, setBottoms] = useState(null);

const [footwear, setFootwear] = useState(null);

const resetAllSlots = () => {

setHeadwear(null);

setOuterLayer(null);

setMainTop(null);

setBottoms(null);

setFootwear(null);

};

const setItemInSlot = (item) => {

// Встановлюємо елемент у відповідний слот за категорією

switch (item.category) {

case 'cap':

case 'headwear':

setHeadwear(item);

break;

case 'jacket':

case 'outerwear':

setOuterLayer(item);

break;

case 't-shirt':

case 'shirt':

case 'blouse':

case 'sweater':

case 'hoodie':

case 'dress': // Сукня може займати слот основного верху, якщо низ окремо не вибирається

setMainTop(item);

break;

case 'pants':

case 'jeans':

case 'shorts':

case 'skirt':

case 'trousers':

setBottoms(item);

break;

case 'shoes':

case 'sneakers':

case 'boots':

case 'sandals':

case 'heels':

case 'flats':

case 'loafers':

case 'slippers':

setFootwear(item);

break;

default:

console.warn('Unknown category for item:', item.category);

}

};

// ... existing code ...

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.2 – Інтерфейс манекену для збірки комбінацій

4.3 Завантаження одягу та його тегування

Ця функція дозволяє користувачам додавати нові предмети одягу до свого віртуального гардеробу, завантажуючи їх зображення та присвоюючи їм відповідні теги. Теги є ключовими для подальших рекомендацій та організації гардеробу.

Процес включає завантаження файлу на фронтенді, відкриття модального вікна для введення даних (назва, категорія, теги) та подальшу відправку цих даних на бекенд для збереження в базу даних. На стороні бекенду контролер обробляє запит і додає новий тег, забезпечуючи його унікальність для кожного користувача.

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.3 – Інтерфейс сторінки Гардеробу

Обрані програмні рішення та технологічний стек дозволили створити функціональний та сучасний веб-додаток. Використання React[4] на фронтенді забезпечило високу інтерактивність та швидкість роботи інтерфейсу, а Node.js на бекенді – ефективну обробку запитів та легку інтеграцію зі сторонніми сервісами. Розподіл логіки на кастомні хуки (наприклад, для керування манекеном) та чітка структура контролерів на бекенді сприяли чистоті коду та його подальшій підтримці. Інтеграція з потужними API, такими як Google Gemini, дозволила реалізувати ключову інтелектуальну функцію продукту, що виділяє його серед аналогів. Таким чином, прийняті рішення є збалансованими та повністю відповідають поставленим завданням проєкту.

Під час розробки було подолано низку технічних викликів, що дозволило отримати цінний практичний досвід. Одним із ключових завдань на бекенді була організація стабільної та швидкої взаємодії з API Google Gemini, що вимагало ретельного опрацювання асинхронних запитів, механізмів обробки помилок та тайм-аутів. На фронтенді особливу увагу було приділено оптимізації керування станом додатку, особливо в таких інтерактивних компонентах, як віртуальна примірочна. Ефективне використання кастомних хуків React дозволило ізолювати складну логіку та зробити компоненти більш перевикористовуваними й легкими для тестування.

Запроєктована архітектура та обраний технологічний стек створюють міцний фундамент для подальшого розвитку та масштабування проєкту "SmartStyle". У майбутньому функціонал може бути розширений за рахунок впровадження машинного навчання для аналізу особистих вподобань користувача та надання ще більш персоналізованих рекомендацій. Також перспективним напрямком є розробка нативних мобільних додатків для iOS та Android для покращення користувацького досвіду та розширення аудиторії, а також інтеграція з платформами електронної комерції, що дозволить користувачам купувати вподобані речі безпосередньо через додаток.

.

## **5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ**

### 5.1 Порівняння результатів із початковими вимогами

Основною метою проєкту було створення інтерактивного веб-додатку для управління гардеробом з функцією інтелектуальних рекомендацій. Порівнюючи фінальний продукт з початковими вимогами, можна констатувати, що всі ключові функціональні вимоги були успішно реалізовані. Система дозволяє користувачам реєструватися, керувати своїм профілем, додавати, переглядати та видаляти предмети одягу. Реалізовано центральну функцію генерації рекомендацій щодо образу на основі погодних умов та критеріїв користувача через інтеграцію з Google Gemini API.

Також були виконані нефункціональні вимоги. Інтерфейс розроблено з урахуванням принципів адаптивності, що забезпечує коректне відображення на різних пристроях. Реалізована повна локалізація інтерфейсу на українську та англійську мови, що відповідає вимозі щодо багатомовності. Архітектура додатку, побудована за клієнт-серверною моделлю, забезпечує необхідну гнучкість та масштабованість. Таким чином, розроблений продукт повністю відповідає специфікації вимог, визначеній на початковому етапі проєктування.

### 5.2 Оцінка якості роботи системи

Якість роботи системи "SmartStyle" можна оцінити за кількома критеріями. З точки зору продуктивності, додаток демонструє швидкий відгук інтерфейсу завдяки використанню React та архітектури SPA. Завантаження сторінок та оновлення компонентів відбувається практично миттєво. Час відповіді API для більшості операцій (отримання даних гардеробу, оновлення профілю) є мінімальним. Єдиним винятком є функція генерації AI-рекомендацій, час виконання якої залежить від зовнішнього сервісу Google Gemini, що є очікуваною поведінкою.

Надійність системи забезпечується чітким розмежуванням логіки на бекенді та обробкою можливих помилок. Система коректно обробляє невалідні вхідні дані та помилки, що надходять від зовнішніх API, інформуючи про це користувача. Зручність використання (usability) є високою завдяки інтуїтивно зрозумілій навігації, мінімалістичному дизайну та логічній організації функціоналу. Користувачеві не потрібно докладати значних зусиль для освоєння основних сценаріїв роботи з додатком.

### 5.3 Аналіз ефективності прийнятих рішень

Вибір технологічного стеку виявився повністю виправданим. Використання React дозволило створити сучасний та динамічний фронтенд, а компонентний підхід значно спростив розробку та подальшу підтримку коду. Інтеграція з бібліотекою Material-UI прискорила процес створення візуально привабливого та адаптивного інтерфейсу. На стороні бекенду Node.js[5] та Express.js[6] показали себе як ефективне рішення для побудови легкого та швидкого RESTful API, здатного легко інтегруватися зі сторонніми сервісами.

Рішення про використання PostgreSQL як бази даних було стратегічно правильним, оскільки ця СУБД забезпечує надійність, цілісність даних та можливості для майбутнього розширення. Найбільш ефективним рішенням з точки зору функціональності стала інтеграція з Google Gemini API. Це дозволило реалізувати потужну інтелектуальну функцію без необхідності розробки та навчання власної складної AI-моделі, що значно скоротило час та ресурси, витрачені на розробку.

### 5.4 Обмеження та недоліки

Не дивлячись на успішну реалізацію основних вимог, проєкт має певні обмеження та потенційні напрямки для вдосконалення. На даному етапі збереження пресетів реалізовано на стороні клієнта з використанням **localStorage**. Це робить функцію швидкою та простою, але має недолік: збережені образи не синхронізуються між різними пристроями чи браузерами одного користувача. Для повноцінної синхронізації необхідно реалізувати збереження пресетів у базі даних на стороні сервера.

Іншим обмеженням є залежність від зовнішніх API. Будь-які перебої в роботі OpenWeatherMap[10] або Google Gemini[8], а також зміни в їхніх тарифних планах чи умовах використання, можуть безпосередньо вплинути на функціональність "SmartStyle". Також, хоча система надає текстові рекомендації, вона не виконує автоматичне розпізнавання одягу на завантажених зображеннях, що вимагає від користувача ручного введення даних та тегів.

Крім того, на поточному етапі розробки програмна система розгорнута та функціонує в локальному тестовому середовищі (localhost). Це забезпечує зручність для розробки та налагодження, проте для ширшого доступу користувачів та експлуатації в реальних умовах необхідне розгортання на публічно доступній серверній інфраструктурі, що є одним із наступних кроків у розвитку проєкту.

Для забезпечення належної якості та функціональності веб-додатку "SmartStyle" було проведено базове ручне функціональне тестування. Це включало перевірку всіх ключових функцій, таких як реєстрація (якщо є) та авторизація користувачів, генерація образів на основі введених даних, робота з погодним API та API для генерації рекомендацій, збереження та відображення пресетів, а також коректність роботи інтерфейсу користувача та навігації. Основна увага приділялася відповідності реалізованого функціоналу заявленим вимогам та коректності обробки типових сценаріїв використання. Тестування проводилося в різних веб-браузерах (наприклад, Chrome, Firefox) для забезпечення базової кросбраузерності.

Окрім ручного тестування, для подальшого розвитку проєкту та забезпечення вищої надійності рекомендується впровадження автоматизованих тестів. Це може включати:

Модульне тестування (Unit Testing): Написання тестів для окремих компонентів React на фронтенді (з використанням бібліотек типу Jest та React Testing Library) та для окремих модулів/функцій на бекенді (наприклад, з Jest або Mocha/Chai). Це дозволить швидко виявляти помилки на ранніх стадіях розробки після внесення змін до коду.

Інтеграційне тестування (Integration Testing): Перевірка взаємодії між різними частинами системи, наприклад, між фронтендом та бекендом (перевірка коректності запитів до API та обробки відповідей), а також взаємодії бекенду з базою даних та зовнішніми API.

Тестування користувацького інтерфейсу (UI Testing / E2E Testing): Автоматизація сценаріїв взаємодії користувача з інтерфейсом (з використанням інструментів як Cypress або Puppeteer). Це дозволить перевіряти коректність роботи всього додатку "від початку до кінця".

Тестування API (API Testing): Написання тестів для перевірки всіх ендпоінтів бекенд API (з використанням Postman, Newman або бібліотек для автоматизованого тестування API).

Впровадження цих видів тестування значно підвищить стабільність додатку, полегшить рефакторинг та підтримку коду в довгостроковій перспективі.

## **ВИСНОВКИ**

У ході виконання курсового проєкту було успішно спроєктовано та розроблено веб-додаток "SmartStyle", призначений для управління персональним гардеробом та отримання інтелектуальних рекомендацій щодо створення образів. Було проведено детальний аналіз предметної галузі, що дозволило виявити ключові потреби користувачів та визначити конкурентні переваги майбутнього продукту.

На основі аналізу було сформульовано чіткі функціональні та нефункціональні вимоги, які стали основою для подальшого проєктування. Розроблена архітектура системи, що базується на трирівневій клієнт-серверній моделі, забезпечила надійне розмежування відповідальності між компонентами та створила гнучкий фундамент для подальшого розвитку. Було спроєктовано нормалізовану схему бази даних PostgreSQL, яка гарантує цілісність та ефективне зберігання даних.

Ключовим досягненням проєкту є успішна реалізація всіх запланованих функцій, включаючи реєстрацію користувачів, керування гардеробом, інтерактивне створення образів на віртуальному манекені та, що найважливіше, генерацію персоналізованих рекомендацій за допомогою інтеграції з Google Gemini API. Розроблений додаток має сучасний, інтуїтивно зрозумілий та повністю локалізований інтерфейс.

Таким чином, мета курсового проєкту була досягнута в повному обсязі. Створений програмний продукт є готовим до використання прототипом, який демонструє високу якість реалізації та має значний потенціал для подальшого розвитку, наприклад, шляхом додавання серверної синхронізації пресетів та впровадження функцій машинного зору для автоматичного тегування одягу.

## **ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Гаверіо, Р. Веб-дизайн з React. Створення сучасних користувацьких інтерфейсів / пер. з англ. – Харків: Вид-во "Ранок", 2023. – 416 с.
2. Фрімен, А. Pro Express.js: Посібник із розробки на Node.js / пер. з англ. – Київ: Видавнича група BHV, 2021. – 528 с.
3. Моммані, А. PostgreSQL для початківців. Практичний посібник / пер. з англ. – Львів: Видавництво Старого Лева, 2022. – 320 с.
4. React – Бібліотека JavaScript для створення користувацьких інтерфейсів. Meta. URL: [https://uk.react.dev/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fuk.react.dev%2F) (дата звернення: 10.06.2025).
5. Node.js — відкрите кросплатформне середовище виконання JavaScript. Фонд OpenJS. URL: [https://nodejs.org/uk/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fnodejs.org%2Fuk%2F) (дата звернення: 10.06.2025).
6. Express – Швидкий, не обтяжений зайвими правилами, мінімалістичний веб-фреймворк для Node.js. ExpressJS. URL: [https://expressjs.com/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fexpressjs.com%2F) (дата звернення: 10.06.2025).
7. PostgreSQL: Найбільш прогресивна реляційна база даних з відкритим кодом у світі. Глобальна розробницька група PostgreSQL. URL: [https://www.postgresql.org/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.postgresql.org%2F) (дата звернення: 10.06.2025).
8. Штучний інтелект Google для розробників. Google. URL: [https://ai.google.dev/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fai.google.dev%2F) (дата звернення: 10.06.2025).
9. MUI: Бібліотека компонентів React, яку ви завжди хотіли. MUI. URL: [https://mui.com/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fmui.com%2F) (дата звернення: 10.06.2025).
10. API Погоди. OpenWeatherMap. URL: [https://openweathermap.org/api](https://www.google.com/url?sa=E&q=https%3A%2F%2Fopenweathermap.org%2Fapi) (дата звернення: 10.06.2025).
11. Посилання завантаженого проєкту на GitHub: https://github.com/ArtemFNP/2025\_B\_KKP\_PZPI-22-9\_Filatov\_A\_D (дата звернення: 20.06.2025).

## **ДОДАТОК А**

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

## **ДОДАТОК Б**

Специфікація ПЗ

**CS330 Software Engineering**

**Software Requirements Specification (SRS) Template**

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

Software System for Styling and Smart Use of Clothes

SmartStyle Software Engineer

Artem Filatov

**Software Requirements Specification**

**Document**

**Version: 1.0** **Date: 20.06.2025**

**Зміст**

**1. ВСТУП**

1.1 Огляд продукту

1.2 Мета

1.3 Межі

1.4 Посилання

1.5 Означення та абревіатури

**2. ЗАГАЛЬНИЙ ОПИС**

2.1 Перспективи продукту

2.2 Функції продукту

2.3 Характеристики користувачів

2.4 Загальні обмеження

2.5 Припущення й залежності

**3. КОНКРЕТНІ ВИМОГИ**

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

3.1.2 Апаратний інтерфейс

3.1.3 Програмний інтерфейс

3.1.4 Комунікаційний протокол

3.1.5 Обмеження пам’яті

3.1.6 Операції

3.1.7 Функції продукту

3.1.8 Припущення й залежності

3.2 Властивості програмного продукту

3.3 Атрибути програмного продукту

3.3.1 Надійність

3.3.2 Доступність

3.3.3 Безпека

3.3.4 Супроводжуваність

3.3.5 Переносимість

3.3.6 Продуктивність

3.4 Вимоги бази даних

3.5 Інші вимоги

1 ВСТУП

1.1 Огляд продукту

"SmartStyle" – це інноваційний веб-додаток, розроблений для допомоги користувачам в управлінні їхнім гардеробом, отриманні персоналізованих рекомендацій щодо одягу та створенні образів на основі погодних умов та особистих уподобань. Продукт поєднує функції віртуальної шафи, погодного дашборду та рекомендацій на базі штучного інтелекту, щоб спростити щоденний вибір одягу та оптимізувати використання наявного гардеробу.

1.2 Мета

Основною метою проєкту "SmartStyle" є надання користувачам інтуїтивно зрозумілого та функціонального інструменту для:

* ефективного каталогізування власного одягу;
* отримання розумних рекомендацій щодо образів, адаптованих до поточної та прогнозованої погоди;
* збереження та керування улюбленими наборами одягу (пресетами);
* планування гардеробу для конкретних подій або періодів часу;

1.3 Межі

Проєкт "SmartStyle" зосереджений на розробці веб-додатку з наступними ключовими функціями:

* реєстрація та автентифікація користувачів;
* керування персональним профілем користувача;
* додавання, перегляд та видалення предметів одягу у віртуальному гардеробі;
* система тегування одягу (категорії, випадки, локації, активності), прив'язана до користувача;
* відображення поточної погоди та погодинного прогнозу для вибраних міст;
* генерація рекомендацій щодо образів за допомогою AI, враховуючи погоду, місце призначення, тип активності та стиль;
* можливість збереження створених образів як пресетів;
* багатомовна підтримка інтерфейсу (українська, англійська);
* поза межами проєкту залишаються;
* інтеграція з електронною комерцією або онлайн-магазинами;
* функції соціальних мереж (наприклад, обмін образами з іншими користувачами);
* розширені можливості редагування зображень одягу всередині програми;
* розробка нативних мобільних додатків (фокус на адаптивному веб-дизайні);

1.4 Посилання

* кодова база проєкту "SmartStyle" (локальний репозиторій);
* документація до зовнішніх програмних інтерфейсів: OpenWeatherMap API, Google Gemini API;
* стандарти розробки веб-додатків W3C.

1.5 Означення та абревіатури

* ui (User Interface): Користувацький інтерфейс.
* ux (User Experience): Досвід користувача.
* api (Application Programming Interface): Програмний інтерфейс додатків.
* jwt (JSON Web Token): Стандарт для створення токенів доступу, що використовуються для автентифікації.
* ai (Artificial Intelligence): Штучний інтелект.
* frontend: Клієнтська частина веб-додатку, що працює в браузері користувача.
* backend: Серверна частина веб-додатку, що обробляє логіку та взаємодіє з базою даних.
* db (Database): База даних.
* crud: Акронім для Create, Read, Update, Delete (створення, читання, оновлення, видалення) – основні операції з даними.
* mui (Material-UI): Бібліотека компонентів React, що реалізує Material Design.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

"SmartStyle" – це незалежний веб-додаток, який функціонує за моделлю клієнт-сервер. Фронтенд-частина, розроблена на React, взаємодіє з бекенд-частиною (Node.js з фреймворком Express.js) через RESTful API. Бекенд відповідає за бізнес-логіку, керування базою даних PostgreSQL [3] та інтеграцію із зовнішніми API для отримання даних про погоду та генерації рекомендацій. Додаток розроблено як самодостатнє рішення для персонального стилю, не інтегроване в більші системи.

2.2 Функції продукту

* автентифікація користувачів: Реєстрація, вхід та вихід із системи;
* керування профілем: Перегляд та редагування імені користувача;
* керування гардеробом: Додавання, перегляд та видалення предметів одягу;
* створення образів: Візуальний підбір одягу на інтерактивному манекені;
* збереження пресетів: Можливість зберігати комбінації одягу;
* погодні дані: Відображення поточної погоди та прогнозу;
* ai-рекомендації: Генерація пропозицій щодо образу на основі погоди та критеріїв користувача;
* тегування одягу: Створення та керування користувацькими тегами;
* локалізація: Підтримка англійської та української мов.

2.3 Характеристики користувачів

* зареєстровані користувачі: Основна цільова аудиторія. Користувачі, які пройшли процес реєстрації та мають доступ до повного функціоналу програми. Передбачається, що вони мають базові навички роботи з веб-додатками;
* незареєстровані користувачі: Відвідувачі сайту, які можуть ознайомитися з продуктом через загальні сторінки (наприклад, "Про нас"), але не мають доступу до персоналізованих функцій.

2.4 Загальні обмеження

* залежність від зовнішніх API: Функціонал рекомендацій та погодних даних повністю залежить від доступності та коректної роботи API OpenWeatherMap та Google Gemini;
* локальне розгортання: На етапі розробки бекенд розгортається локально, що обмежує доступність для зовнішніх користувачів;
* веб-середовище: Додаток призначений для роботи у веб-браузерах і не оптимізований як нативний мобільний додаток;
* зберігання зображень: Наразі керування зображеннями відбувається через URL-адреси. Для масштабованого використання потрібне більш надійне рішення для зберігання файлів (наприклад, хмарне сховище).

2.5 Припущення й залежності

* користувачі повинні мати стабільне інтернет-з'єднання;
* правильно налаштовані API-ключі для зовнішніх сервісів повинні бути доступні в оточенні бекенду;
* база даних PostgreSQL повинна бути встановлена, налаштована та доступна для підключення;
* бекенд-сервер та фронтенд-додаток повинні бути розгорнуті в сумісних середовищах.

3. КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

* дизайн: Сучасний, чистий та адаптивний дизайн, що забезпечує коректне відображення на різних пристроях. Використання темної палітри кольорів з фіолетовими акцентами;
* навігація: Інтуїтивно зрозуміла навігація через верхню навігаційну панель та футер;
* елементи управління: Використання компонентів Material-UI для стандартних елементів для забезпечення консистентності;
* завантаження зображень: Зручний інтерфейс для завантаження зображень одягу (drag-and-drop або вибір файлу).

3.1.2 Апаратний інтерфейс

Вимоги до апаратного забезпечення системи є стандартними для сучасних веб-додатків та не передбачають використання спеціалізованого обладнання. З боку клієнта, для повноцінної роботи з додатком "SmartStyle" користувачеві потрібен будь-який пристрій (персональний комп'ютер, ноутбук, планшет або смартфон) з встановленим сучасним веб-браузером та стабільним доступом до мережі Інтернет. Це забезпечує максимальну доступність продукту для широкого кола користувачів.

З боку сервера, для розгортання та функціонування бекенд-частини та бази даних потрібен сервер, фізичний або віртуальний. Конкретні вимоги до обчислювальних ресурсів, таких як потужність центрального процесора (CPU) та обсяг оперативної пам'яті (RAM), не є фіксованими та залежать від очікуваного навантаження, зокрема кількості одночасних користувачів. Для етапу розробки та тестування достатньо ресурсів стандартної локальної машини розробника.

3.1.3 Програмний інтерфейс

Програмна архітектура "SmartStyle" побудована на взаємодії кількох ключових програмних компонентів та технологій.

Клієнтська частина (Frontend) реалізована на мові JavaScript (стандарт ES6+) з використанням бібліотеки React.js, яка є основою для побудови користувацького інтерфейсу. Навігація між сторінками додатку забезпечується бібліотекою React Router DOM. Взаємодія з серверним API відбувається за допомогою бібліотеки Axios, що дозволяє надсилати асинхронні HTTP-запити. Для інтернаціоналізації інтерфейсу використовується бібліотека i18next. Візуальне оформлення та компоненти інтерфейсу реалізовані за допомогою Material-UI (MUI).

Серверна частина (Backend) побудована на платформі Node.js з використанням фреймворку Express.js для організації RESTful API. Для взаємодії з базою даних PostgreSQL використовується клієнт pg. Безпека паролів забезпечується за допомогою бібліотеки bcrypt для їх хешування, а автентифікація на основі токенів реалізована за допомогою jsonwebtoken (JWT). Для керування змінними середовища використовується пакет dotenv.

В якості системи управління базами даних виступає PostgreSQL [7]. Також система інтегрується із зовнішніми програмними інтерфейсами: OpenWeatherMap API для отримання метеорологічних даних та Google Gemini API для генерації інтелектуальних рекомендацій.

3.1.4 Комунікаційний протокол

Взаємодія між клієнтською та серверною частинами системи "SmartStyle" здійснюється за допомогою протоколу HTTP/HTTPS. В основі комунікації лежить архітектурний стиль REST (Representational State Transfer), що забезпечує безстатусний (stateless) обмін даними.

Для передачі даних між клієнтом та сервером використовується уніфікований формат JSON (JavaScript Object Notation), який є легким, читабельним для людини та легко обробляється на стороні JavaScript. Автентифікація користувача для доступу до захищених ресурсів API реалізована за допомогою Bearer токенів на основі стандарту JWT. Після успішного входу користувач отримує токен, який він повинен додавати до заголовка Authorization кожного наступного запиту до захищених кінцевих точок.

3.1.5 Обмеження пам’яті

Хоча система не має жорстких лімітів на використання пам'яті, її архітектура спроєктована з урахуванням принципів ефективності. На клієнтській стороні додаток оптимізовано для раціонального використання пам'яті браузера, щоб уникнути "витоків" та забезпечити плавну роботу інтерфейсу навіть на менш потужних пристроях. Це досягається за рахунок ефективного керування станом та життєвим циклом компонентів React.

На серверній стороні було прийнято важливе архітектурне рішення для мінімізації навантаження на базу даних та використання пам'яті. Замість зберігання бінарних даних зображень одягу безпосередньо в базі даних (у вигляді BLOB-об'єктів), система зберігає лише URL-адреси, що вказують на місцезнаходження файлів. Самі файли зображень передбачається зберігати у файловій системі сервера або в спеціалізованому хмарному сховищі, що є значно більш продуктивним та масштабованим підходом.

3.1.6 Операції

Програмний інтерфейс додатку (API) надає набір кінцевих точок (endpoints) для виконання всіх необхідних операцій. Ці операції логічно згруповані за функціональними модулями:

* автентифікація (/api/auth): Ця група операцій відповідає за керування життєвим циклом користувача, включаючи реєстрацію нового акаунту (POST /register), вхід у систему (POST /login), оновлення даних профілю (PUT /users/:userId) та отримання інформації про поточного автентифікованого користувача (GET /me);
* керування гардеробом (/api/clothing, /api/tags): Набір CRUD-операцій для взаємодії з особистим гардеробом. Включає отримання списку одягу, додавання нових предметів, їх видалення, а також створення та отримання користувацьких тегів;
* інтелектуальні функції (/api/ai, /api/weather): Операції, що надають доступ до "розумних" можливостей додатку. Це, зокрема, отримання AI-рекомендацій для часової шкали (POST /ai/timeline-suggestion) та запит актуальних погодних даних (GET /weather).

3.1.7 Функції продукту

Ключовий функціонал продукту визначається набором можливостей, доступних кінцевому користувачеві. Система забезпечує повний цикл автентифікації та авторизації, дозволяючи користувачам безпечно створювати облікові записи та отримувати доступ до персонального простору. В рамках керування профілем користувач може змінювати своє відображуване ім'я.

Центральною функцією є керування гардеробом. Користувачі можуть завантажувати зображення предметів одягу, класифікувати їх за категоріями та призначати їм набір персоналізованих тегів. На основі цього цифрового гардеробу реалізована функція підбору образів, де користувач може візуально комбінувати речі на інтерактивному манекені та зберігати вдалі комбінації як "пресети" для майбутнього використання. Найбільш інноваційною є функція AI-рекомендацій, яка аналізує погоду, критерії користувача та його гардероб для генерації пропозицій щодо образу.

3.1.8 Припущення й залежності

Коректна робота системи "SmartStyle" базується на кількох ключових припущеннях та залежностях. Передбачається, що для успішної роботи криптографічних функцій автентифікації на сервері має бути налаштована змінна середовища JWT\_SECRET. Якість та точність даних про погоду та AI-рекомендацій безпосередньо залежать від стабільності та доступності зовнішніх API (OpenWeatherMap та Google Gemini), а також від якості даних, що надаються цими сервісами.

Також робиться припущення, що користувачі завантажуватимуть коректні файли зображень у підтримуваних форматах (PNG, JPG, JPEG, SVG) і що для виконання операцій, які вимагають ідентифікації користувача (наприклад, редагування профілю), його user.id буде коректно отриманий з контексту автентифікації на стороні клієнта.

3.2 Властивості програмного продукту

Система повинна надавати повний набір функцій для реалізації свого основного призначення. Користувач повинен мати можливість пройти повний цикл від реєстрації та входу до активного використання всіх можливостей додатку. Система має забезпечувати сесійну стійкість після входу, дозволяючи користувачеві залишатися в системі при перезавантаженні сторінки.

Ключовою функціональною вимогою є надання повного набору CRUD-операцій (створення, читання, оновлення, видалення) для управління особистими предметами одягу. Система повинна реалізовувати механізм генерації рекомендацій щодо образів, який враховує погодні умови та критерії, задані користувачем. Також система зобов'язана відображати актуальні погодні дані та надавати інструменти для створення та збереження персоналізованих наборів одягу (пресетів). Нарешті, інтерфейс має підтримувати повну багатомовність з можливістю перемикання між українською та англійською мовами.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система спроєктована з фокусом на високу надійність. Це досягається за рахунок ретельної обробки помилок як на клієнтській, так і на серверній стороні, що запобігає несподіваним збоям та надає користувачеві зрозумілі повідомлення про проблеми. Цілісність даних у базі даних PostgreSQL забезпечується використанням обмежень (constraints) та механізмів транзакцій, що унеможливлює появу неузгоджених даних.

3.3.2 Доступність

Додаток розробляється з метою забезпечення цілодобової доступності (24/7), за винятком коротких періодів планового технічного обслуговування. Зручність використання є пріоритетом: інтерфейс є інтуїтивно зрозумілим, що дозволяє користувачам з різним рівнем технічної підготовки легко освоїти функціонал системи.

3.3.3 Безпека

Безпека є одним з найважливіших атрибутів системи. Захист даних користувачів реалізовано шляхом зберігання паролів у вигляді криптографічних хешів за допомогою алгоритму bcrypt. Автентифікація всіх запитів до захищених ресурсів API відбувається за допомогою JWT токенів з обмеженим терміном дії. Авторизація гарантує, що користувачі мають доступ лише до власних персональних даних. Також система має базові механізми захисту від поширених веб-уразливостей, таких як SQL-ін'єкції та XSS.

3.3.4 Супроводжуваність

Для забезпечення легкості підтримки та подальшого розвитку, структура коду є модульною та добре організованою. Код відповідає загальноприйнятим стандартам кодування, а критичні та складні його секції належним чином прокоментовані. Усі зовнішні залежності чітко визначені у файлах package.json, що спрощує розгортання та налаштування середовища розробки.

3.3.5 Переносимість

Система спроєктована з урахуванням переносимості. Фронтенд-додаток забезпечує крос-браузерну сумісність і коректно функціонує в останніх версіях основних веб-браузерів (Chrome, Firefox, Safari, Edge). Крос-платформенність серверної частини гарантується використанням Node.js та PostgreSQL, які можуть бути розгорнуті на різних операційних системах (Windows, macOS, Linux).

3.3.6 Продуктивність

Продуктивність системи є важливим нефункціональним атрибутом. Час відповіді API для більшості стандартних операцій не повинен перевищувати 500 мс. Час завантаження основних сторінок додатку оптимізовано для забезпечення швидкого доступу. Розуміючи, що продуктивність AI-операцій залежить від зовнішнього сервісу, в інтерфейсі передбачено візуальні індикатори завантаження, щоб інформувати користувача про процес очікування.

3.4 Вимоги бази даних

Для зберігання даних використовується система управління базами даних PostgreSQL. Схема бази даних спроєктована за реляційною моделлю і включає набір таблиць для зберігання ключових сутностей: users (користувачі), clothing\_items (предмети одягу), tags (теги), outfit\_sets (збережені образи). Для реалізації зв'язків "багато-до-багатьох" (наприклад, між одягом та тегами) використовуються спеціальні асоціативні таблиці. Для оптимізації продуктивності запитів передбачено використання індексів на полях, які часто використовуються для пошуку та зв'язування, зокрема на зовнішніх ключах.

3.5 Інші вимоги

Окрім вищезазначених, до системи висуваються й інші вимоги. Зручність використання має бути максимальною, а процес взаємодії – логічним та простим. Код повинен супроводжуватися належною внутрішньою документацією у вигляді коментарів. Система обробки помилок повинна надавати користувачеві не лише технічну інформацію, а й зрозумілі підказки щодо вирішення проблеми, що покращує загальний досвід користування продуктом.

## **ДОДАТОК В**

Слайди презентації

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.1

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.2

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.3

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.4

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.5

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.6

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.7

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.8

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.9

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.10

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.11

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.12

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.13