Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка методу та програмних засобів автоматизованої системи підбору одягу»

Виконав: студент IV курсу
групи ЗПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Гуменюк Р. О.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Коваленко О. О.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Тарновський М. Г.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«_10_» __червня_____ 2024 р.

ВНТУ – 2024

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

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

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

Гуменюку Руслану Олексійовичу

1. Тема роботи – «Розробка методу та програмних засобів автоматизованої системи підбору одягу».

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 року №80.

2. Термін подання студентом роботи: 3 червня 2024 року.

3. Вихідні дані до роботи: середовище розробки – PyCharm, мова запитів – MySQL; мови розробки – Python; операційна система – Windows 7/8/10/11.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану засобів системи автоматизованого підбору одягу та постановка задач дослідження; розробка методу, моделі та алгоритмів роботи системи; розробка програмних модулів програми; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської дипломної роботи, тема; мета, об'єкт та предмет дослідження; задачі дослідження; наукова новизна; практична цінність отриманих результатів; апробація та публікації результатів роботи; фінальний слайд.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	Коваленко О. О., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання _____13 березня 2024 р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз розвитку засобів системи автоматичного підбору одягу	14.03.2024 – 31.03.2024	
2	Розробка методу, моделі та алгоритмів роботи вебсистеми	01.04.2024– 15.04.2024	
3	Розробка програмних модулів	16.04.2024 – 18.05.2024	
4	Тестування програми	19.05.2024 – 24.05.2024	
5	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.2024	

Студент _____ (підпис) **Гуменюк Р. О.** (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) **Коваленко О. О.** (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 65 сторінок формату А4, на яких є 26 рисунки, 3 таблиці, список використаних джерел містить 27 найменувань.

У бакалаврській дипломній роботі було розроблено програмні модулі для системи автоматизованого підбору одягу у вебзастосунку. Застосунок призначений для забезпечення користувачів автоматизованим підбором одягу відповідно до погодних умов за системою ECWCS gen 2.

Розроблені програмні модулі дозволяють клієнтам швидко та ефективно отримувати рекомендації щодо вибору одягу на основі погодних умов у їхньому регіоні. Використання вебзастосунку спрощує взаємодію з системою, забезпечуючи зручний інтерфейс для взаємодії.

Застосунок реалізовано з використанням мови програмування Python та середовища розробки PyCharm. Використання вебзастосунку дозволяє забезпечити доступ до системи з будь-якого пристрою з підтримкою месенджера Telegram.

У результаті виконання бакалаврської дипломної роботи було створено систему, що підвищує ефективність процесу підбору одягу, роблячи його зручнішим та доступнішим для користувачів.

Отримані в бакалаврській дипломній роботі результати можна використати для покращення процесу автоматизованого підбору одягу.

Ключові слова: підбір одягу, вебзастосунку, погодні умови, Python, ECWCS.

ABSTRACT

The bachelor’s thesis consists of 65 A4 pages, which have 26 figures, 3 tables, the list of sources used contains 27 items.

In the bachelor's thesis, software modules have been developed for an automated clothing selection system using a Telegram bot. The application is designed to provide users with automated clothing selection based on weather conditions according to the ECWCS gen 2 system.

The developed software modules allow customers to quickly and efficiently receive recommendations for clothing selection based on weather conditions in their region. The use of a Telegram bot simplifies interaction with the system, providing a convenient interface for communication.

The application is implemented using the Python programming language and the PyCharm development environment. Using the Telegram bot allows you to access the system from any device supporting the Telegram messenger.

As a result of completing the bachelor's thesis, a system was created that increases the efficiency of the clothing selection process, making it more convenient and accessible for users.

The results obtained in the bachelor’s thesis can be used to improve the process of automated clothing selection.

Keywords: clothing selection, Telegram bot, weather conditions, Python, ECWCS.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Розширена система одягу для холодного клімату (ECWCS) є комплексною платформою одягу, спеціально розробленою для захисту людей від екстремальних погодних умов і надання комфорту в умовах низьких температур, вітру та дощу. Ця система призначена не лише для військових, але й для широкого спектру користувачів, таких як рятувальники, любителі активного відпочинку, альпіністи, пошуково-рятувальні групи, туристи та інші. ECWCS складається з різноманітних шарів та компонентів, які працюють у поєднанні, щоб забезпечити оптимальний захист від холоду, вологи та вітру, а також для збереження тепла та комфорту під час виконання активних дій або перебування на відкритому повітрі [1].

У наш час, зі зростанням частоти та інтенсивності екстремальних погодних явищ, безпека людей стала найвищим пріоритетом. Необхідність надійного захисту та адекватної підготовки до небезпечних ситуацій стає все більш актуальною. Ефективне вирішення цієї проблеми передбачає вжиття заходів, спрямованих на забезпечення швидкого та зручного доступу до необхідного екіпірування та одягу, які відповідають погодним умовам.

Важливість розробки автоматизованих рішень для вибору відповідного одягу та екіпірування в контексті екстремальних погодних умов стає беззаперечною. Швидкість та точність у прийнятті рішень можуть вирішально вплинути на безпеку людей. Автоматизована система, що аналізує погодні умови та рекомендує відповідний одяг, спрощує процес вибору та забезпечує користувачів необхідними засобами для ефективного протистояння екстремальним погодним умовам [2].

На сьогоднішній день, незважаючи на наявність автоматизованих систем для вибору одягу відповідно до погодних умов, існують деякі недоліки, що варто врахувати. Перш за все, існуючі системи можуть не завжди надавати точні та адаптовані рекомендації для конкретного користувача. Це може бути

пов'язано з обмеженими даними або не достатньою точністю алгоритмів прогнозування. Крім того, деякі системи можуть бути складними у використанні або не оптимальними з точки зору інтерфейсу користувача.

Тому розробка методу і засобів автоматизованої системи підбору одягу є актуальною задачею на сьогоднішній день.

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

Мета та завдання дослідження. Метою роботи є покращення процесу вибору одягу за рахунок розробки і використання автоматизованої вебсистеми з інтеграцією ECWCS, що дозволяє підбирати одяг з урахуванням погодних умов.

Планується інтеграція засобів управління даними, які дозволять збирати та аналізувати інформацію про погоду та вимоги користувачів. Це допоможе забезпечити точність та надійність рекомендацій системи. Також у рамках роботи передбачається розробка модуля для взаємодії зі сторонніми додатками, такими як мобільні додатки або вебсервіси, що розширить можливості користувачів отримувати рекомендації щодо одягу у будь-який час.

Задачі дослідження:

– реалізувати систему, що використовуватиме сторонні API, які надаватимуть погодні дані та іншу необхідну інформацію;

– інтегрувати сторонні API для отримання актуальних погодних даних за отриманою геолокацією;

– розробити метод і модель роботи системи та алгоритм підбору одягу, який буде враховувати отримані погодні умови;

– забезпечити роботу зручного інтерфейсу користувача;

– провести тестування та оптимізацію роботи розробленої системи.

Об’єкт дослідження – процес розробки автоматизованої системи підбору одягу.

Предмет дослідження – методи та програмні засоби розробки автоматизованої системи підбору одягу.

Методи дослідження. У дослідженні було використано методи:

– методи аналізу джерел та існуючих рішень для вивчення аналогів та обґрунтування необхідності розробки власної системи;

– методи аналізу інформації для створення алгоритмів підбору одягу;

– методи інтеграції сторонніх API для отримання актуальних погодних даних на основі геолокації користувачів;

– методи розробки програмного забезпечення для програмної реалізації системи;

– методи розробки інтерфейсу користувача (UI/UX) для створення зручного та інтуїтивного інтерфейсу вебзастосунку;

– методи тестування та оптимізації системи для перевірки працездатності, продуктивності та стабільності роботи системи.

Новизна отриманих результатів:

1. Подальшого розвитку набув метод формування і відображення історії підбору одягу, який, на відміну від відомих, інтегрує ECWCS та враховує не лише поточні погодні умови, але й індивідуальні параметри користувача, такі як рівень активності та особисті вподобання, що дозволяє забезпечити більш точні та персоналізовані рекомендації щодо вибору одягу та підвищити комфорт й ефективність використання системи.

2. Подальшого розвитку набула модель інтеграції сторонніх API для отримання погодних даних, яка, на відміну від існуючих, забезпечує автоматичне оновлення даних у реальному часі та надає можливість отримання прогнозів погоди на кілька днів, що дозволяє користувачам планувати свій гардероб заздалегідь.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що розроблена у бакалаврській дипломній роботі автоматизована система підбору одягу може використовуватись у повсякденному житті для забезпечення користувачів

рекомендаціями щодо вибору одягу відповідно до поточних та прогнозованих погодних умов. Система дозволяє користувачам швидко та ефективно отримувати індивідуальні рекомендації, що підвищує комфорт та зручність у повсякденному житті. Використання вебзастосунку для взаємодії з системою забезпечує доступність та простоту використання, що робить систему привабливою для широкого кола користувачів.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У друкованій роботі [3], опублікованій у співавторстві, автору належать такі результати: розробка алгоритму роботи застосунку, алгоритму підбору одягу за системою ECWCS, а також алгоритму функціонування вебзастосунку.

Апробація матеріалів бакалаврської дипломної роботи. Матеріали бакалаврської дипломної роботи доповідалися на конференції: LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету, Вінниця, 2024.

Публікації. Результати дослідження були опубліковані в матеріалах конференції [3].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 25 найменувань, 5 додатків. Робота містить 34 ілюстрації та 3 таблиці.

1 АНАЛІЗ РОЗВИТКУ ЗАСОБІВ СИСТЕМИ АВТОМАТИЗОВАНОГО ПІДБОРУ ОДЯГУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану засобів системи підбору одягу

У сучасному світі зростає популярність та значущість технологій, які спрямовані на забезпечення комфорту та безпеки людей в різних ситуаціях, зокрема, під час вибору одягу в залежності від погодних умов. Програмні засоби, які автоматизують цей процес та надають користувачам рекомендації щодо вибору відповідного одягу, стають необхідними інструментами в повсякденному житті [4].

Однак, багато існуючих систем підбору одягу не завжди враховують індивідуальні потреби користувачів та не можуть надати оптимальний вибір відповідно до конкретних умов. Більшість з них оперують загальними алгоритмами, які не враховують особливості місця проживання, особисті уподобання та інші фактори, які можуть впливати на вибір одягу.

У зв'язку з цим, виникає потреба в розробці простої та ефективної системи підбору одягу, яка враховуватиме не лише погодні умови, але й індивідуальні характеристики користувача. Така система має бути легкою у використанні та забезпечувати точні та надійні рекомендації щодо вибору одягу.

Розробка такої системи відкриває можливості для створення інноваційних програмних рішень, які допоможуть користувачам з ефективністю та задоволенням вирішувати завдання підбору одягу у будь-яку погоду та ситуацію.

1.2 Порівняльний аналіз аналогів

З розвитком сучасних технологій змінюється спосіб підбору одягу: від смарт-текстилю до розумного підбору гардеробу. У цьому контексті стає очевидною необхідність створення засобів, які допомагатимуть зробити процес вибору одягу гнучким, зручним та ефективним. Розробка

автоматизованої системи підбору одягу через вебзастосунок на основі системи ECWCS виходить за межі традиційного підходу до вибору гардеробу.

Зараз споживачі вимагають більшого комфорту та індивідуального підходу до вибору одягу, що ставить перед модними брендами та ринком одягу нові виклики. Автоматизована система підбору одягу через вебзастосунку на базі системи ECWCS відкриває шлях до інноваційного підходу до цього процесу. Вона поєднує в собі передові технології з розумним аналізом погодних умов та індивідуальних вподобань користувачів, щоб надати їм найкращі рекомендації щодо вибору одягу. Такий підхід забезпечує не лише зручність та ефективність, але й персоналізований підхід до потреб кожного клієнта, що є ключовим в сучасній модній індустрії [5].

Порівняно з традиційними методами вибору одягу, автоматизована система підбору через вебзастосунку на основі системи ECWCS може забезпечити більш точний та індивідуалізований підбір, враховуючи особливості погодних умов, потреби користувача та рекомендації системи.

Розглянемо популярні ресурси як аналоги розроблюваної системи, такі як:

- Looksize;
- Rozetka;
- Shop the look.

Одним з прикладів використання розумного підбору гардеробу є Looksize (рис. 1.1). Looksize є інноваційною віртуальною примірочною для інтернет-магазинів одягу та взуття [6]. Програмне забезпечення було розроблене для того, щоб допомогти покупцям визначити правильний розмір одягу та взуття за допомогою простої кнопки. Враховуючи те, що в світі не існує єдиної системи розмірів, а кожен бренд має свою власну таблицю розмірів, Looksize ставить перед собою завдання знизити кількість повернень товарів та покращити досвід покупців.

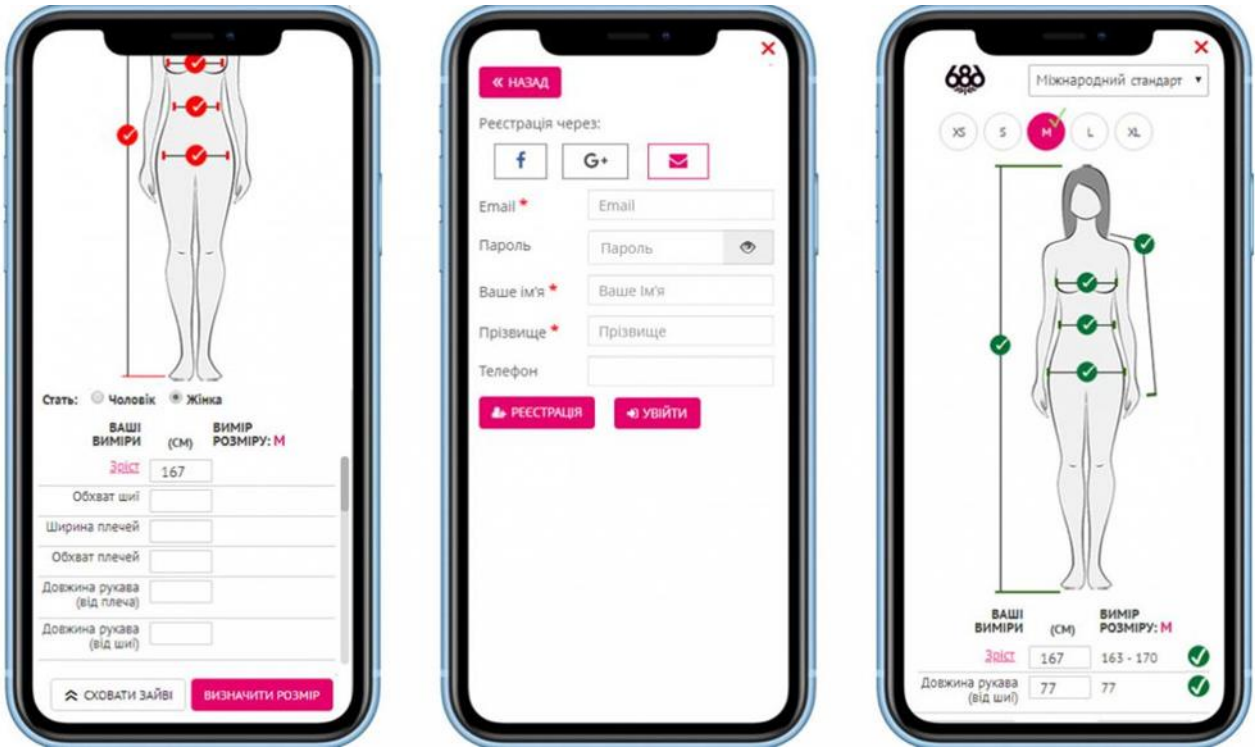

Рисунок 1.1 – Приклад роботи застосунку Looksize

Інший приклад - інтернет-магазин Rozetka [7], відомий своїм постійним прагненням до інновацій та покращень, впроваджує нову послугу – «онлайн-примірювальні» (рис. 1.2). Ця послуга, що доступна на їхньому вебсайті, спрямована на підбір розміру для одягу, взуття та аксесуарів. Зокрема, вона стала відповіддю на проблему різниці в розмірах між різними брендами, що часто викликає незручності у покупців.

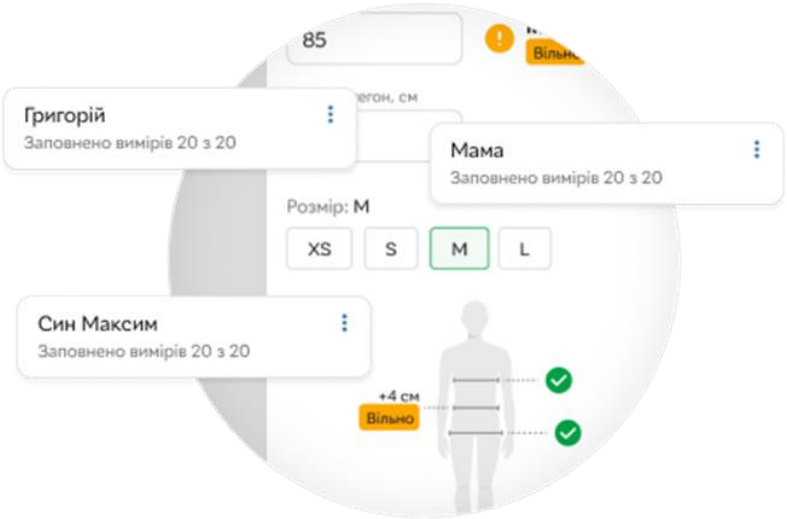

Рисунок 1.2 – Приклад роботи застосунку Rozetka

Shop the look – застосунок в екосистемі інтернет-магазину “Amazon”, дозволяє підібрати товари в магазині за фотографією [8]. Штучний інтелект обробляє фото, знаходить на фото одяг та пробує знайти схожі товари в інтернет-магазині (рис. 1.3).

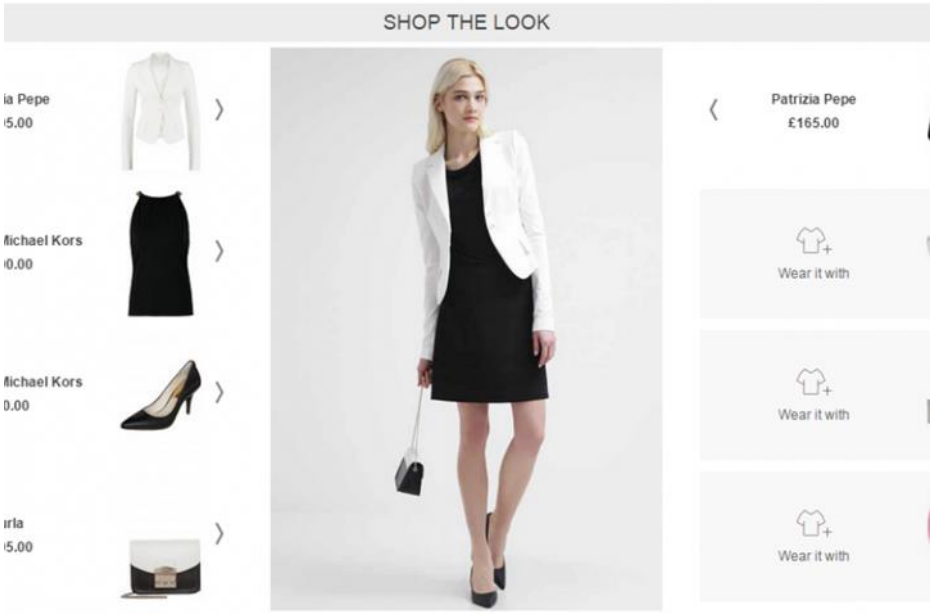

Рисунок 1.3 – Приклад роботи застосунку Shop the look

Для наочної демонстрації відмінностей розглянутих додатків було зведено їх переваги й недоліки у таблицю порівняння (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз аналогів

	Looksize	Rozetka	Shop the Look	Власна розробка
Підбір за системою ECWCS	0	0	0	1
Підбір одягу відповідно до погодних умов	0	0	0	1
Персоналізація даних	1	1	0	1
Автоматична система	0	0	1	1
Штучний інтелект	0	0	1	0
Сумарний коефіцієнт	1	1	2	4

Аналізуючи таблицю 1.1, відзначимо, що власна розробка має вищий сумарний коефіцієнт за розглянутими критеріями у порівнянні з аналогами Shop the look на 50%, Looksize та Rozetka на 80%.

Отже, порівняльний аналіз аналогів у сфері автоматизованого підбору одягу розкриває різноманітні підходи та можливості, що відкриваються перед розробниками програмної системи. Розуміння переваг та обмежень існуючих аналогів дозволяє налагодити оптимальні технічні рішення та функціонал, які відповідають вимогам та потребам користувачів.

Одержані в ході аналізу знання про існуючі рішення стають основою для вдосконалення та розробки нової автоматизованої системи підбору одягу з використанням вебзастосунку на основі системи ECWCS. Переваги та недоліки аналогів надають цінний вихідний матеріал для створення інноваційного продукту, спроможного задовольнити потреби сучасного ринку та підняти якість обслуговування користувачів на новий рівень.

1.3 Аналіз методів розв’язання задачі

Аналіз методів розв'язання поставленої задачі є ключовим етапом у розробці автоматизованої системи підбору одягу з використанням вебзастосунку на основі системи ECWCS. Для ефективного вирішення поставленої задачі потрібно ретельно вивчити і порівняти різні підходи та методи, що застосовуються у сфері автоматизованого підбору одягу. Такими можуть бути, наприклад:

1. Метод застосування алгоритмів математичних обчислень. За допомогою цих алгоритмів система може аналізувати велику кількість даних про користувачів, їхні вподобання, стиль життя, метеорологічні умови та інші фактори, що впливають на вибір одягу. На основі цього аналізу система може надавати рекомендації щодо оптимального одягу для конкретної ситуації.

2. Метод використання бази даних та алгоритмів аналізу даних для зберігання та обробки інформації про асортимент одягу, його характеристики та сумісність з погодними умовами. Цей підхід дозволить ефективно керувати

інформацією та швидко здійснювати підбір відповідного одягу для користувачів.

3. Метод інтеграції системи зовнішніх API для отримання актуальної інформації про погоду та прогнозування, що дозволить системі надавати користувачам точні та актуальні рекомендації щодо вибору одягу в залежності від погодних умов у їхньому регіоні.

4. Метод дослідження існуючих програмних продуктів та сервісів, які вже здійснюють автоматизований підбір одягу. Це дозволить виявити переваги та недоліки існуючих рішень і врахувати їх при розробці власної системи.

Таким чином, аналіз методів розв'язання поставленої задачі дозволить обрати оптимальний шлях для розробки автоматизованої системи підбору одягу з використанням вебзастосунку на основі системи ECWCS. Аналіз цих методів підтверджує важливість обрання комплексного підходу до розробки системи, який забезпечить якість та ефективність її функціонування.

1.4 Постановка задач для розробки методу і засобів автоматизованої системи підбору одягу

На основі аналізу переваг та недоліків існуючих систем підбору одягу та використання засобів доповненої реальності, для успішної розробки мобільної системи були визначені наступні завдання:

- реалізація вебзастосунку, який використовуватиме сторонні API для отримання геолокації за назвою міста, що дозволить користувачам зручно вказувати своє місцезнаходження без необхідності вводити координати;
- інтеграція іншого API для отримання актуальних погодних даних за отриманою геолокацією, що забезпечить можливість отримати достовірну та актуальну інформацію про погоду у конкретному регіон;
- розробка моделі методу та алгоритмів роботи системи;
- тестування та оптимізація роботи системи, що включатиме перевірку правильності реакції системи на різні погодні умови та

удосконалення алгоритму підбору одягу для кращої відповідності потребам користувачів.

1.5 Висновки

У першому розділі було роглянуто стан додатків з автоматизованою системою підбору одягу. Серед них варто відзначити такі додатки, як Looksize, Rozetka та Shop the Look. Вони являють собою нові рішення в галузі вибору одягу, демонструючи сучасні тенденції в цьому напрямку.

Аналіз показав, що багато існуючих систем підбору одягу не враховують індивідуальні потреби користувачів та не забезпечують оптимальний вибір відповідно до конкретних умов. Водночас, порівняно з традиційними методами вибору одягу, автоматизована система підбору через вебзастосунку на основі системи ECWCS може забезпечити більш точний та індивідуалізований підбір. На основі аналізу було сформовано перелік завдань, які включають розробку архітектури системи, створення вебзастосунку, інтеграцію з системою ECWCS, розробку алгоритмів підбору одягу та розробку інтерфейсу.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ

2.1 Аналіз вхідних даних системи

Для реалізації застосунку для вибору одягу в екстремальних температурних умовах будуть використані засоби програмування Python та інтегроване середовище розробки PyCharm. Python є потужною мовою програмування, яка надає широкі можливості для обробки даних та розробки алгоритмів. PyCharm, у свою чергу, забезпечує зручне середовище для написання, тестування та налагодження програмного коду на Python [9].

Розробка програмних модулів для автоматизованої системи підбору одягу включає кілька послідовних етапів. Першим кроком є збір всіх вхідних даних, детальний опис логіки роботи системи, а також створення таблиці даних, на основі якої буде здійснюватись вибір одягу. Ця таблиця включатиме різні параметри, такі як тип заходу, погодні умови, особисті вподобання користувача та інші фактори.

Вхідні дані:

– погодні умови: температура повітря, вологість, швидкість і напрямок вітру, опади, поточний стан погоди;

– геолокація: координати, місцезнаходження користувача.

Один з вирішальних модулів в рамках розробки системи - це модуль розпізнавання геолокації за назвою міста [10]. Завдяки цьому модулю система матиме можливість автоматично визначати місцезнаходження користувача з використанням інформації про назву міста, що введена користувачем. Це дозволить системі точно враховувати погодні умови в конкретній локації під час підбору одягу. Наприклад, якщо користувач знаходиться у місті з теплим кліматом, система може рекомендувати легку та повітропроникну одяг, тоді як у холодних регіонах буде врахована необхідність у теплому та утепленому одязі. Такий підхід забезпечить користувачам більш точні та індивідуалізовані рекомендації відповідно до місцевих погодних умов [11].

Крім того, ключовим аспектом системи буде її інтеграція з різними системами провайдерів даних, які постачають інформацію про погоду, модні тенденції та наявність товарів у магазинах. Цей модуль забезпечить систему актуальними даними, що дозволить надавати користувачам найточніші рекомендації щодо вибору одягу. Наприклад, він буде взаємодіяти з сервісами прогнозу погоди, щоб коректно враховувати погодні умови під час формування рекомендацій. Крім того, він буде отримувати дані про модні тенденції та асортимент товарів у магазинах, щоб забезпечити користувачів актуальними інформацією та створити зручні умови для покупки необхідного одягу. Такий підхід дозволить системі підбору одягу надавати високоякісний сервіс та забезпечувати задоволення користувачів її функціоналом [12].

Отже, розробка програмних модулів для створення автоматизованої системи підбору одягу на Python у середовищі розробки PyCharm є складним завданням, яке вимагає деяких підходів. Модулі повинні бути уважно спроектовані, ефективно реалізовані та ретельно протестовані, щоб гарантувати оптимальну функціональність та забезпечити задоволення користувачів результатами роботи системи. Незважаючи на ці виклики, з використанням відповідних інструментів та компетентного підходу можна створити програмні рішення, які значно полегшать процес підбору одягу та покращать досвід користувачів.

2.2 Розробка моделі системи

Для кращого усвідомлення принципів роботи автоматизованої системи підбору одягу за ECWCS (Extended Cold Weather Clothing System), було вирішено побудувати модель її функціонування з використанням UML діаграми діяльності. Ця діаграма дозволяє наглядно відображати послідовність дій та процесів, які відбуваються в системі під час підбору одягу для різних погодних умов.

На рисунку 2.1 представлена UML діаграма діяльності, яка ілюструє взаємодію різних компонентів системи підбору одягу за ECWCS [13]. Ця

модель допомагає зрозуміти процеси вибору одягу, включаючи врахування погодних умов, вимог до захисту та комфорту користувача. Розглядання такої діаграми сприяє кращому розумінню функціональності та логіки роботи системи.

UML (Unified Modeling Language) діаграми є стандартною мовою для візуалізації, проектування та документування програмних систем. Вони дозволяють розробникам інженерії програмного забезпечення відображати структуру та поведінку системи за допомогою графічних символів та правил. Ці діаграми допомагають зрозуміти та комунікувати різні аспекти системи між розробниками, командами та стейкхолдерами.

У UML існують різні види діаграм, такі як структурні (наприклад, діаграми класів, об'єктів, компонентів) та поведінкові (наприклад, діаграми активностей, послідовностей, станів). Кожен вид діаграми має свою специфіку та призначення, що дозволяє моделювати різні аспекти системи.

Використання UML діаграм допомагає виявити та вирішити проблеми ще на ранніх етапах розробки, сприяє покращенню комунікації між учасниками проекту та сприяє створенню документації, яка є важливою частиною будь-якого програмного проекту.

Ця модель дозволяє аналізувати та вдосконалювати функціональність системи підбору одягу за ECWCS, забезпечуючи оптимальний вибір одягу для різних сценаріїв використання. Використання UML діаграм діяльності допомагає зрозуміти взаємозв'язки між різними компонентами системи та процесами взаємодії між ними.

Згідно з цією моделлю, першим кроком для користувача буде обрати потрібний вид одягу за системою ECWCS. Потім він переходить до перегляду рекомендованого одягу. Перед використанням системи автоматизованого підбору одягу, система проводить перевірку статусу авторизації користувача. Якщо авторизація невдалий, система пропонує користувачу авторизуватися. Після цього користувач може розпочати використання системи, прокладаючи

маршрут з урахуванням поточної геолокації або переглядаючи рекомендований одяг у режимі доповненої реальності.

Завдяки UML можна краще зрозуміти архітектуру програми, її компоненти та взаємозв'язки між ними [14]. Це допомагає розробникам та іншим зацікавленим сторонам отримати чітке уявлення про те, як працює програма та які процеси в ній відбуваються.

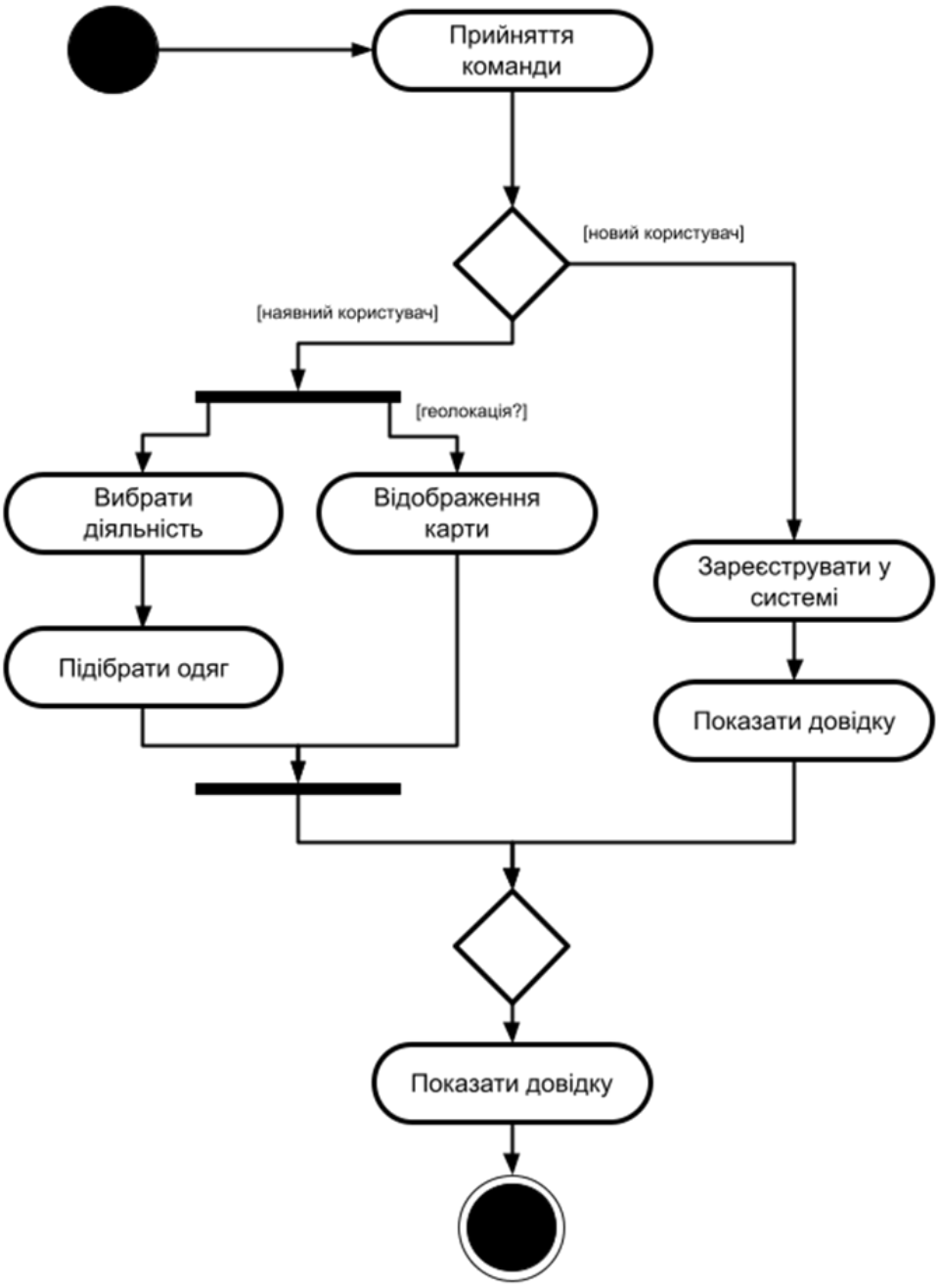

Рисунок 2.1 – Модель системи у вигляді діаграми діяльності

Діаграма діяльності системи показує загальну роботу програми на високому рівні абстракції для легшого розуміння робочих процесів.

2.3 Розробка методу формування і відображення історії підбору одягу

У будь-який момент використання системи до повного завершення підбору одягу користувач має можливість за бажанням або необхідністю сформувати файл-звіт, що дозволяє детально відслідковувати рекомендації та обрані комбінації одягу. Метод формування і відображення результатів дозволяє згрупувати дані користувача та отримати їх у порівнянні з рекомендаціями системи в зручному графічному вигляді.

Метод включає такий функціонал:

1. Після входу в особистий кабінет користувача доступний розділ "Історія підбору одягу".

2. Користувач натискає кнопку «Сформувати звіт» у секції профілю для ініціювання процесу формування звіту.

3. При натисканні кнопки запускається функція, яка відповідає за отримання даних з MySQL бази даних та їх підготовку для подальшого використання [15].

4. Метод getDataFromMySQL виконує з’єднання з базою даних MySQL.

5. За допомогою SQL-запитів до бази даних виконується запит для отримання user_id, який використовується як критерій для додаткових запитів до бази даних.

6. За протоколом HTTP з бази даних отримується user_id.

6.1. За критерієм user_id здійснюються запити до бази даних з метою отримання інформації, яка необхідна для заповнення решти полей звіту.

6.2. Відповідно до запитів, з бази даних отримуються дані у форматі JSON, які записуються у змінну user_data.

6.3. Метод getDataFromMySQL повертає об’єкт, що містить усю

інформацію, яку запитувала система, зокрема: ім’я користувача, список рекомендованого одягу та погодні умови.

7. Отримані дані передаються в метод formatDataForChart, який форматує дані для подальшого використання та повертає підготовлений масив formattedData.

7.1. Після отримання підготовленого масиву даних метод заповнює поля ідентифікації користувача: surname та name.

7.2. Після заповнення полей ідентифікації метод встановлює решту даних масиву у стан компонента data, який далі буде використано для графічного представлення даних.

7.2. Сформатовані дані передаються в компонент фреймворку ReactJS, який використовує бібліотеки react-charts та react-chartjs-2.

7.3. За допомогою бібліотек генеруються діаграми та гістограми, що відображають прогрес навчання користувача.

8. Після цього завершується формування звіту:

8.1. Отримується актуальна дата і встановлюється у поле звіту «Дата формування».

8.2. Встановлюється значення поля «report_number» типу int.

9. Користувачеві відображається сторінка із згенерованим звітом.

Описаний метод включає кроки авторизації користувача, отримання даних з MySQL, форматування цих даних для графічного представлення, завершення формування звіту і відображення результату користувачу. Алгоритм роботи цього методу наведено на рисунку 2.2.

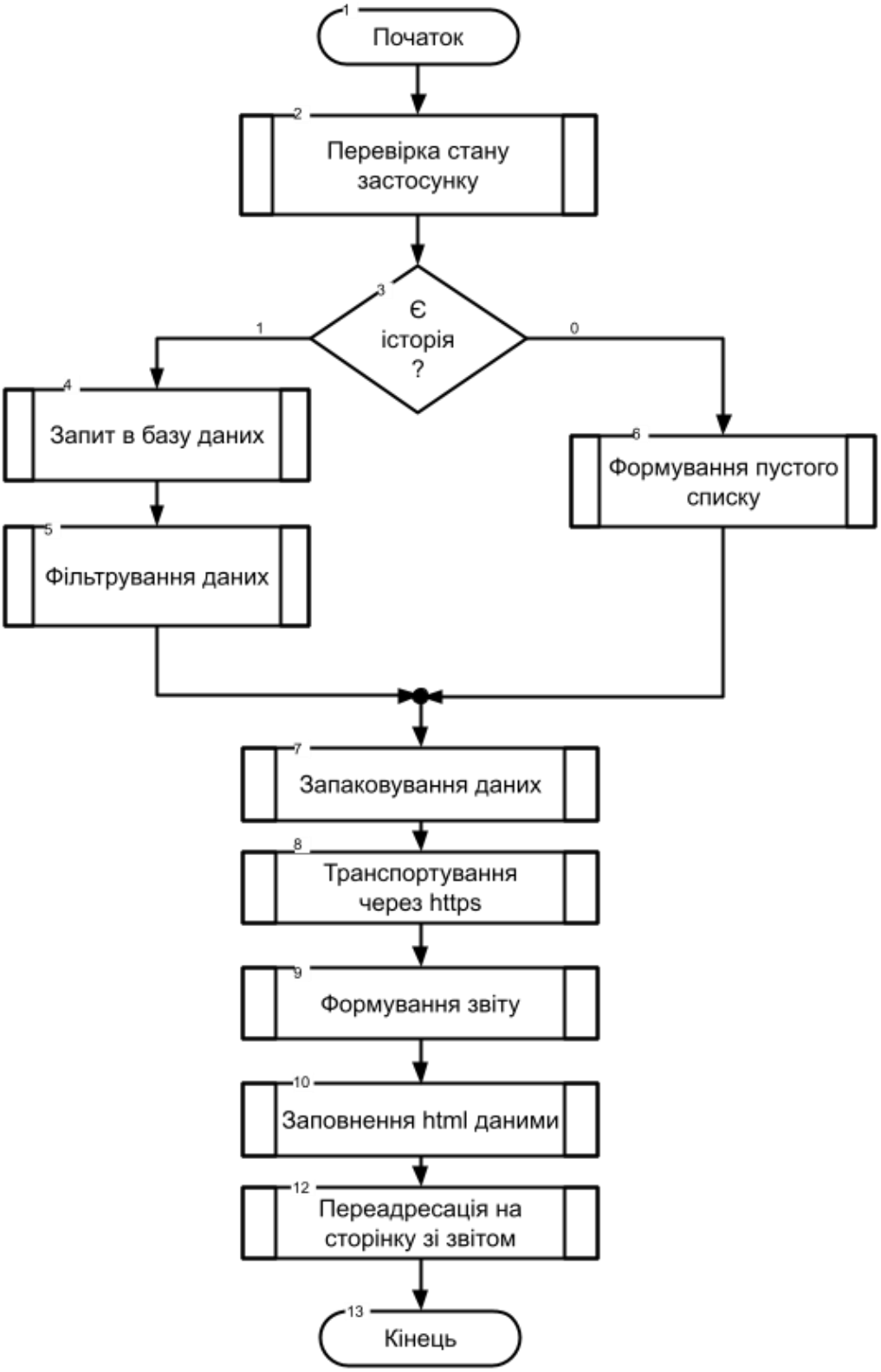

Рисунок 2.2 – Блок-схема алгоритму роботи методу формування і відображення історії підбору одягу

2.4 Розробка алгоритмів роботи системи

Для розробки програмного забезпечення автоматизованої системи підбору одягу, необхідно розробити загальний алгоритм, який визначатиме його функціональні можливості. Згідно цього алгоритму, спочатку користувач отримує можливість взаємодіяти з ботом шляхом виклику команди або взаємодії з візуальним інтерфейсом. Після цього користувач може здійснювати різні запити, такі як отримання рекомендацій щодо одягу для певного заходу або погодних умов. Після вибору опції, користувач отримує відповідну інформацію від бота. Крім того, бот може надавати додаткові можливості, такі як збереження улюблених комбінацій одягу або підписка на отримання оновлень про погоду. Цей підхід дозволить створити зручний та ефективний інструмент для підбору одягу через вебзастосунок.

Алгоритм - це послідовність інструкцій або правил, за якими виконується обчислення або вирішується певна задача. Вони використовуються в різних областях, від програмування до математики та інженерії. Ефективні алгоритми є ключовим елементом розвитку програмного забезпечення та технологій [16].

Один з основних аспектів алгоритмів - це їх ефективність. Ефективні алгоритми забезпечують швидке та ефективне вирішення завдань при мінімальному використанні ресурсів. Розробка та вдосконалення алгоритмів становить важливу частину роботи багатьох інженерних та програмних проектів, оскільки вони дозволяють досягти більшої продуктивності та оптимізації робочих процесів.

Для кращого розуміння роботи модуля підбору одягу через вебзастосунок, було розроблено додатковий алгоритм (рис. 2.3). Цей алгоритм передбачає введення ідентифікаторів користувача та предмета одягу. Після цього, система завантажує список доступного одягу та рекомендацій з підбору. Користувач має можливість переглянути детальний опис предмета одягу, його фото та можливість переглянути у режимі додаткової реальності. Крім того, в

системі враховується географічне розташування користувача для надання актуальних рекомендацій.

Рисунок 2.3 – Загальний алгоритм роботи системи

Аналізуючи ці алгоритми, можна зрозуміти, що розроблений застосунок для автоматизованого підбору одягу дозволяє користувачам з легкістю знаходити потрібний одяг для різних ситуацій. Цей алгоритм повністю автоматизує процес підбору одягу, роблячи його зручним і ефективним для користувачів.

2.5 Висновки

У другому розділі було проведено детальний аналіз вхідних та вихідних даних програмного продукту. Досліджено різноманітні параметри, які впливають на функціонування системи та її ефективність. Крім того, ретельно розглянуто можливості оптимізації продуктивності мобільної платформи шляхом кешування вихідних даних, що дозволяє зменшити час доступу до необхідної інформації.

Далі, у цьому розділі було розроблено основний алгоритм роботи мобільної платформи, який визначає послідовність операцій та взаємодію компонентів системи для досягнення поставленої мети. Також розглянуто та розроблено алгоритм валідації особистих даних, який забезпечує безпеку та надійність використання програмного продукту.

Надалі, для кращого розуміння роботи програмного продукту, використано UML діаграми для моделювання його робочих процесів та структури. Це дозволяє чітко уявити взаємозв'язки між компонентами системи та спростити її розуміння та подальшу розробку.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ. РЕАЛІЗАЦІЯ СИСТЕМИ З ВИКОРИСТАННЯМ ЗАСОБІВ ПІДБОРУ ОДЯГУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробка програмних модулів для мобільної системи з автоматизованим підбором одягу вимагає уважного вибору технологій. Python стає одним із найбільш зручних і потужних інструментів для розробки таких додатків. Його простота, гнучкість та широкі можливості роблять його привабливим вибором для програмістів у цій галузі [17].

Перед початком аналізу розглянемо основні вимоги до нашого веб застосунку. Важливо забезпечити наявність всіх необхідних функцій та можливостей для користувача, здатність обробляти велику кількість запитів без значних затримок, можливість розширення системи при збільшенні навантаження, захист даних користувачів та безпеку комунікацій, інтуїтивно зрозумілий інтерфейс і позитивний досвід користувача, а також легкість у впровадженні нових функцій і можливостей.

При виборі технологій для розробки веб застосунку розглядаємо кілька основних аспектів: фронтенд, бекенд, бази даних, засоби для управління версіями та розгортання. Для фронтенду існує багато популярних бібліотек та фреймворків, таких як React, Angular і Vue.js. React пропонує гнучкість та високу продуктивність завдяки своїй компонентній архітектурі, що дозволяє створювати динамічні інтерфейси. Angular, у свою чергу, надає комплексний набір інструментів та двостороннє зв'язування даних, що зручно для створення великих масштабованих додатків. Vue.js поєднує простоту та потужність, що робить його чудовим вибором для швидкої розробки інтерактивних користувацьких інтерфейсів.

На боці бекенду популярні мови програмування включають JavaScript (з Node.js), Python (з Flask або Django), Ruby (з Ruby on Rails) та PHP (з Laravel). Node.js забезпечує високу продуктивність та асинхронність, що робить його

ідеальним для обробки великої кількості запитів в реальному часі. Python є популярним завдяки своїй простоті та наявності потужних фреймворків, таких як Flask і Django, які дозволяють швидко створювати прототипи та забезпечують високу читабельність коду. Ruby on Rails надає структуру, що значно спрощує процес розробки, але може бути менш продуктивним у порівнянні з іншими варіантами. Laravel пропонує елегантний синтаксис і широкий спектр інструментів для швидкої та ефективної розробки.

Python - це високорівнева, інтерпретована мова програмування загального призначення. Вона відома своєю простотою вивчення та читання, що робить її популярним вибором серед початківців і досвідчених програмістів. Python має чистий і простий синтаксис, що дозволяє розробникам швидко створювати програми і розвивати їх.

Однією з основних переваг Python є його універсальність - ця мова підходить для розробки різноманітних застосунків, від веброзробки та наукових обчислень до штучного інтелекту та аналізу даних. Python має велику кількість стандартних бібліотек та сторонніх модулів, що дозволяє розробникам ефективно використовувати готові рішення для своїх проектів.

Ще однією з переваг Python є активна та дружня спільнота. Це означає, що розробники можуть швидко отримати допомогу, поради та підтримку від інших членів спільноти. Python постійно розвивається і оновлюється, що дозволяє розробникам використовувати нові функції та можливості для покращення своїх програм.

Geopy - це бібліотека для Python, яка надає можливість використовувати географічні дані та функції. Вона дозволяє розробникам отримувати інформацію про місцезнаходження за назвою міста, адресою або географічними координатами, обчислювати відстані між різними точками, а також виконувати інші операції, пов'язані з геолокацією. Geopy базується на відкритих джерелах даних та API, таких як OpenStreetMap, GeoNames та інші, і надає зручний і простий інтерфейс для роботи з географічними даними у програмах на Python [18].

Для розробки нашого веб застосунку обираємо наступні технології: для фронтенду - React, оскільки він забезпечує високу продуктивність та гнучкість; для бекенду - Node.js з Express.js, що дозволяє створювати високопродуктивні та масштабовані серверні додатки; для бази даних - PostgreSQL, який пропонує баланс між простотою використання та потужними можливостями; для управління версіями коду - GitHub, завдяки своїй популярності та широким можливостям для співпраці.

Такий вибір технологій забезпечить ефективну розробку, підтримку та розширення нашого веб застосунку, відповідаючи всім поставленим вимогам.

3.2 Розробка архітектури системи

Розробка архітектури та логіки системи - це ключовий етап у процесі створення програмного продукту, спрямованого на підбір одягу за погодними умовами. На цьому етапі розробники визначають загальну структуру системи, включаючи взаємозв'язки між її складовими частинами, а також алгоритми та логіку роботи.

Під час розробки архітектури системи враховуються потреби користувачів і вимоги до функціональності. Детально аналізуються функції системи, вхідні та вихідні дані, а також можливі сценарії використання. На основі цього аналізу визначаються основні компоненти системи та їх взаємодія [19].

Логіка системи визначає порядок виконання операцій та обробки даних в межах програмного продукту. Це включає в себе розробку алгоритмів для визначення необхідного одягу на основі погодних умов, обробку вхідних даних про погоду та особисті уподобання користувача, а також виведення рекомендаційних повідомлень.

У цьому процесі важливо врахувати ефективність та надійність роботи системи, а також зручність користування нею для кінцевого користувача. Тому розробники активно взаємодіють з дизайнерами та фахівцями з інтерфейсу, щоб забезпечити оптимальний досвід користувача.

Застосунок для підбору одягу за погодними умовами використовує layered архітектуру, що дозволяє створити систему з різними рівнями абстракції та відокремленням функціональності (рис 3.1). Ця архітектура дозволяє розділити програму на незалежні компоненти, що спрощує розробку, тестування та збереження коду [20].

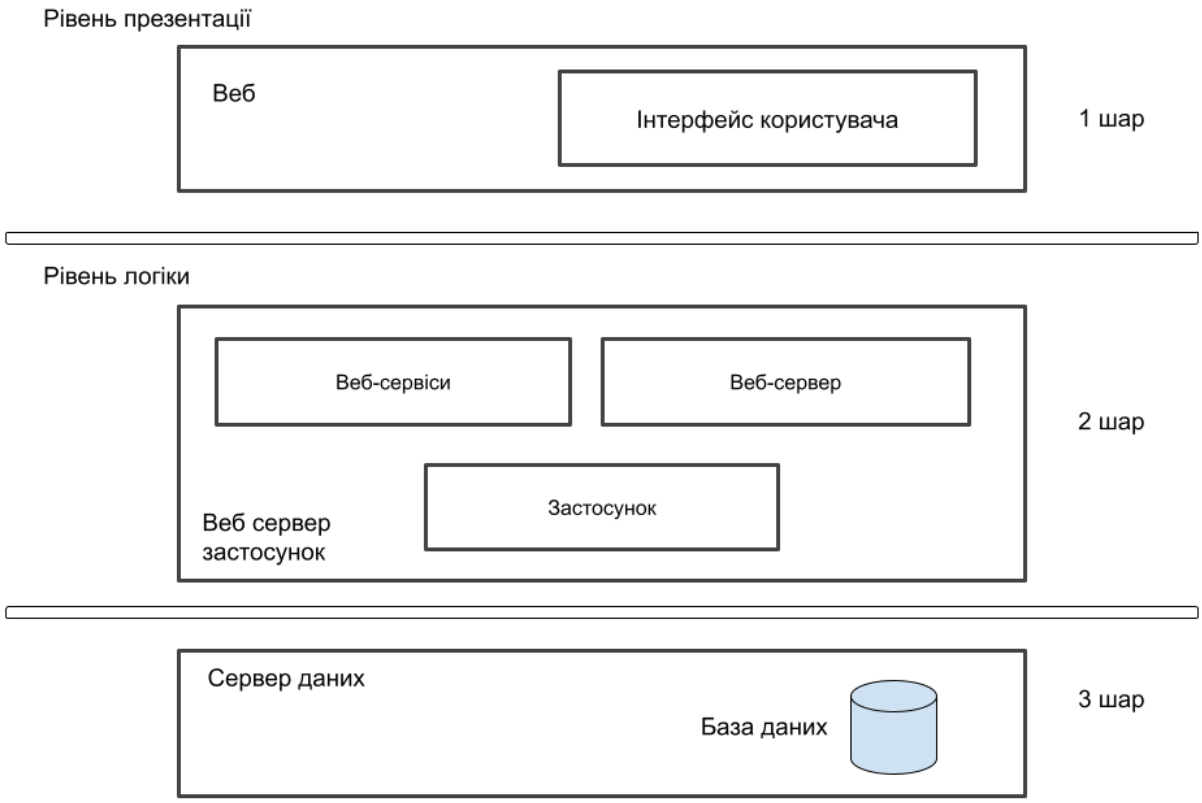

Рисунок 3.1 – Структурна діаграма рівнів

На першому рівні архітектури знаходиться інтерфейс користувача, який взаємодіє з користувачем та відображає результати підбору одягу. На рівні бізнес-логіки реалізовані алгоритми підбору одягу на основі погодних умов та інших факторів. На рівні доступу до даних знаходяться модулі для отримання інформації про погоду та інші вхідні дані, які використовуються для прийняття рішень щодо підбору одягу.

Ця структура дозволяє ефективно управляти складністю системи та полегшує її розвиток та підтримку. Крім того, вона сприяє високій

модульності та переносимості коду, що дозволяє з легкістю вносити зміни та розширювати функціональність застосунку у майбутньому (рис 3.2).

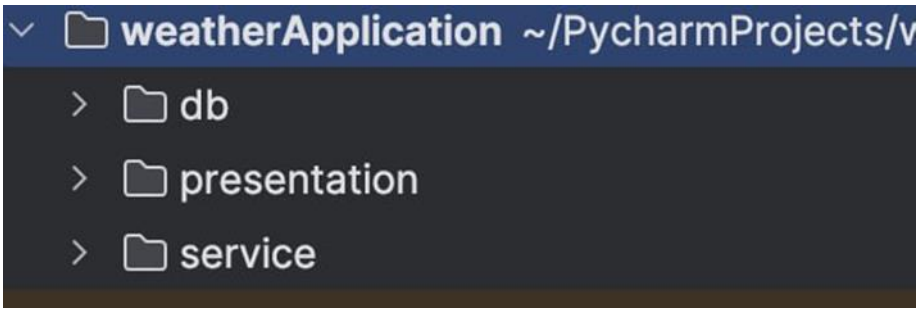

Рисунок 3.2 – Папки для класифікації файлів шаблону Layered

Використання шаблону layered архітектури має кілька переваг. По-перше, цей шаблон дозволяє розділити систему на логічні рівні, кожен з яких відповідає за конкретну частину функціональності [21]. Це сприяє покращенню розуміння системи та полегшує її розробку та підтримку. Крім того, кожен рівень може бути змінений або розширений незалежно від інших, що дозволяє легко вносити зміни в систему та додавати нову функціональність без значного впливу на інші частини програми. Через розділення системи на окремі компоненти, тести можна писати окремо для кожного рівня, що спрощує виявлення та виправлення помилок. Кожен рівень архітектури може бути реалізований як окремий модуль, що дозволяє їх використання у інших проектах або заміну на альтернативні реалізації без необхідності модифікації інших частин системи. Крім того, кожен рівень може бути розроблений та тестований окремо, що дозволяє різним командам працювати над різними частинами системи паралельно, що зменшує час розробки.

Однією з ключових переваг використання layered архітектури є полегшення розробки та підтримки програмного забезпечення. Завдяки чіткому розподілу функціональності на окремі рівні, розробники можуть працювати незалежно один від одного на різних частинах системи. Це сприяє паралельному розвитку коду та полегшує виявлення та виправлення помилок.

Крім того, layered архітектура підвищує модульність програми, оскільки кожен рівень може бути розглянутий як окремий модуль з власним функціоналом та інтерфейсами. Це дозволяє легко змінювати або замінювати окремі компоненти без впливу на інші частини системи (рис 3.3).

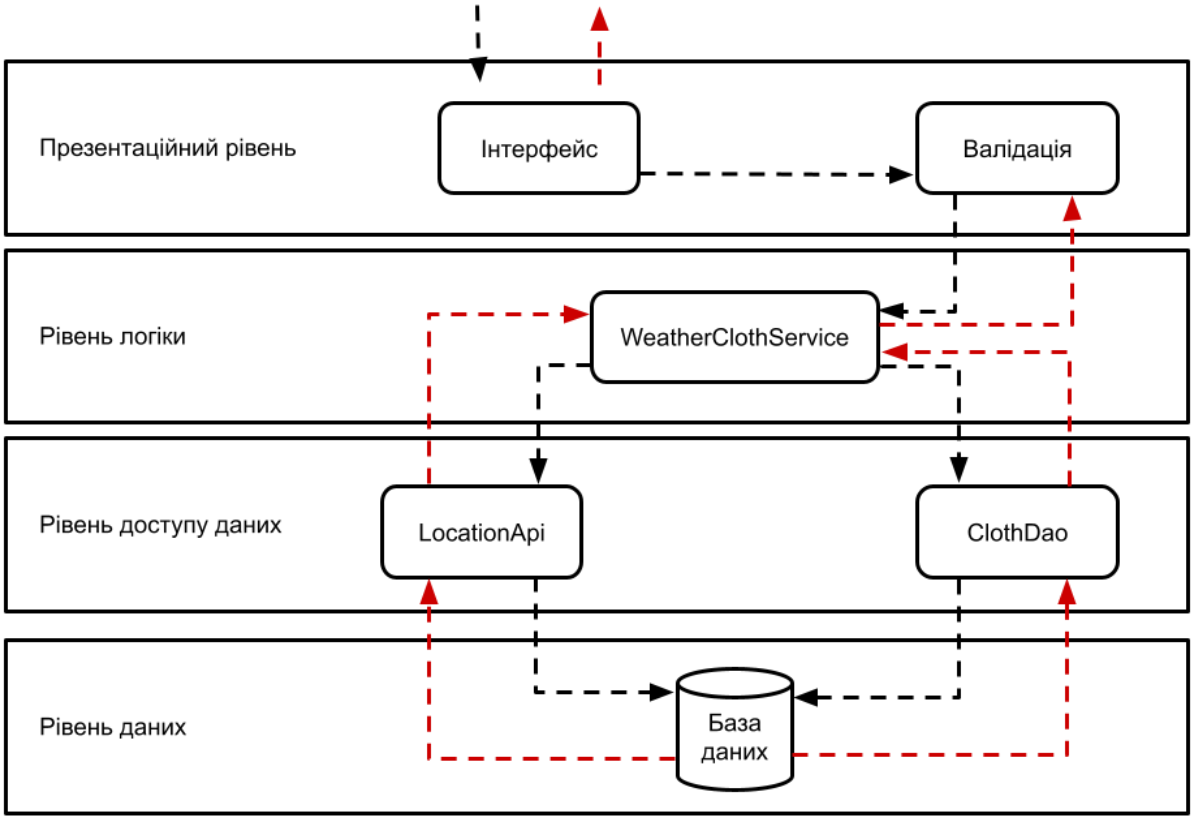

Рисунок 3.3 – Приклад комунікації сутностей

Щоб краще зрозуміти процес розгортання та взаємодію різних компонентів системи, корисно створити UML діаграму розгортання. UML діаграма розгортання візуально представляє архітектуру системи, показуючи, як компоненти програмного забезпечення розподілені по апаратним вузлам та як вони взаємодіють між собою.

На діаграмі розгортання можна побачити, які сервери використовуються, які програмні компоненти на них розгорнуті, а також мережеві з'єднання між цими компонентами (рис 3.4). Це допомагає краще

планувати та координувати процес розгортання, забезпечуючи, щоб всі компоненти системи працювали узгоджено та ефективно.

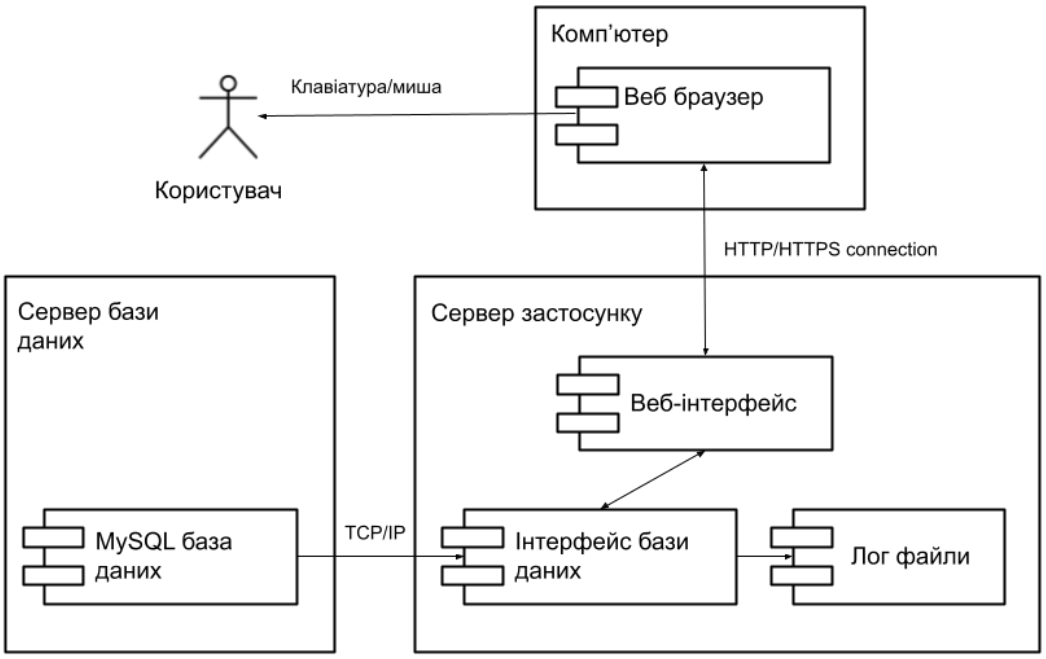

Рисунок 3.4 – Діаграма розгортання

Отже, використання шаблону layered архітектури має кілька переваг. По-перше, цей шаблон дозволяє розділити систему на логічні рівні, кожен з яких відповідає за конкретну частину функціональності. Це сприяє покращенню розуміння системи та полегшує її розробку та підтримку. Крім того, кожен рівень може бути змінений або розширений незалежно від інших, що дозволяє легко вносити зміни в систему та додавати нову функціональність без значного впливу на інші частини програми.

3.3 Розробка модуля геолокації користувача

Розробка модуля геолокації користувача є важливим етапом у створенні системи підбору одягу, оскільки він дозволяє адаптувати рекомендації до конкретного місцезнаходження користувача. Під час розробки цього модуля

використовуються різноманітні технології та інструменти для точного визначення географічних координат користувача.

Перший крок у розробці модуля геолокації - це вибір відповідного програмного забезпечення та технологій для отримання географічних координат. Для цього можуть бути використані різноманітні інструменти, такі як GPS-системи, супутникові навігатори або сервіси, які базуються на вебтехнологіях.

Другим етапом є інтеграція обраного засобу геолокації з програмним додатком (рис 3.5). Це включає розробку програмного інтерфейсу (API), який дозволяє взаємодіяти з функціями отримання географічних координат та передавати їх у систему підбору одягу [22].

Для реалізації модуля геолокації з використанням GPS у веб застосунках можна використовувати різні підходи та інструменти. Одним з найбільш розповсюджених підходів є використання Geolocation API, який надається більшістю сучасних веб браузерів. Цей API дозволяє отримувати інформацію про поточне місцезнаходження користувача за допомогою GPS, Wi-Fi, мобільних мереж та інших джерел [23].

Geolocation API підтримує два основні методи: getCurrentPosition та watchPosition. Метод getCurrentPosition використовується для одноразового отримання поточного місцезнаходження користувача, тоді як watchPosition дозволяє отримувати оновлення про зміну місцезнаходження в режимі реального часу.

Рисунок 3.5 – Інтеграція засобу геолокації

Для нашого веб застосунку обираємо використання Geolocation API у поєднанні з Google Maps API. Geolocation API забезпечує простий та ефективний спосіб отримання даних про місцезнаходження користувача без необхідності використання додаткових бібліотек. Це дозволить нам швидко

отримувати та обробляти інформацію про місцезнаходження, забезпечуючи високу точність та швидкість оновлення даних.

Використання Google Maps API дозволить нам реалізувати візуалізацію місцезнаходження користувача на інтерактивних картах, надавати маршрути та інші геолокаційні сервіси. Google Maps API пропонує широкий набір функцій для роботи з картами, що значно розширить можливості нашого застосунку та покращить користувацький досвід.

Після успішної інтеграції модуля геолокації користувача необхідно провести його тестування. Важливо переконатися, що отримані дані про місцезнаходження коректно і точно передаються у систему та використовуються для надання рекомендацій щодо одягу.

Останнім етапом буде оптимізація роботи модуля геолокації, включаючи покращення швидкості та точності визначення географічних координат, а також забезпечення безпеки та конфіденційності даних користувачів. Через те, що користувачі застосунку можуть мати серйозні проблеми через компрометацію своєї геолокації застосунок повинен реалізовувати алгоритм, що буде захищати інтереси користувачів. В результаті мають бути приблизні координати.

Важливо забезпечити належне оброблення помилок у модулі геолокації, оскільки недостатня обробка може призвести до некоректної роботи застосунку та негативного впливу на користувацький досвід. Помилки можуть виникати з різних причин, таких як втрата зв'язку з GPS-модулем, неправильно вказані дозволи на доступ до місцезнаходження або непередбачені ситуації при спробі отримати географічні координати. Застосування відповідного обробника помилок дозволяє програмі виявити, зрозуміти та відповідно реагувати на виникаючі проблеми, що забезпечує стабільну роботу застосунку у різних умовах експлуатації.

Під час розробки модуля геолокації важливо перевірити правильність збору географічних координат, щоб точно визначати місцезнаходження

користувача. Для цього включаємо механізми перевірки та валідації отриманих даних, щоб упевнитися у їхній достовірності. Після отримання та обробки геолокаційних даних, надсилаємо цю інформацію користувачеві, забезпечуючи йому можливість використання актуального місцезнаходження у застосунку. Це дозволяє забезпечити користувача достовірною інформацією про його географічне положення та забезпечує ефективне функціонування всього застосунку.

Модуль геолокації є одним із ключових компонентів автоматизованої системи підбору одягу через вебзастосунку на основі системи ECWCS. Цей модуль дозволяє системі точно визначати місцезнаходження користувача і враховувати погодні умови в його регіоні. Завдяки інтеграції зі сторонніми сервісами і API, модуль геолокації отримує актуальну інформацію про погоду та кліматичні умови в реальному часі. Це дозволяє системі надавати користувачеві рекомендації щодо підбору відповідного одягу, оптимально враховуючи конкретні погодні умови в його регіоні перебування. Такий модуль додає значну функціональність та корисність системі, роблячи процес вибору одягу більш точним, зручним та ефективним для користувача.

Впровадження модуля геолокації з використанням GPS є важливим аспектом для багатьох веб застосунків [23]. Використання Geolocation API разом з Google Maps API забезпечить точність, швидкість та зручність використання, відповідаючи всім вимогам до модуля геолокації. Такий підхід дозволить створити сучасний та функціональний веб застосунок з потужними геолокаційними можливостями.

3.4 Розробка сервісу конвертації геолокації у погодні дані

Розробка сервісу конвертації геолокації і погодні дані є важливим компонентом в системі підбору одягу, оскільки він дозволяє отримувати актуальну інформацію про погодні умови в конкретному місці перебування користувача. На першому етапі розробки цього сервісу вирішується питання вибору джерела погодних даних та способу їх отримання.

API (інтерфейс програмування застосунків) є набором правил та протоколів, які дозволяють різним програмним компонентам взаємодіяти між собою. Використання API дозволяє розробникам створювати програми, які можуть використовувати функціонал інших систем або сервісів, не потрібно повторно відтворювати цю функціональність. Наприклад, вебсайти можуть використовувати API для отримання даних з соціальних мереж або онлайн-платформ для спільної роботи.

Однією з важливих переваг використання API є забезпечення відкритості та інтеграційної здатності програмних продуктів. Розробники можуть створювати власні API для своїх систем, дозволяючи іншим розробникам використовувати їх функціональність. Це сприяє розвитку екосистеми програмного забезпечення, дозволяючи різним системам та сервісам працювати разом і ефективно обмінюватися даними.

Після вибору джерела погодних даних необхідно розробити механізм конвертації географічних координат у відповідній місцевості для отримання інформації про погоду. Цей процес може включати в себе взаємодію з API погодних сервісів або використання власних алгоритмів для отримання прогнозу погоди.

Останнім етапом розробки є інтеграція розробленого сервісу конвертації геолокації і погодних даних з основною системою підбору одягу. Після успішної інтеграції необхідно провести тестування сервісу для переконання в його коректній роботі та надійності отриманих даних.

Інтеграція застосунку із провайдером погодних даних через weather API є ключовим етапом у забезпеченні користувачам актуальної та достовірної інформації про погоду. Під час цього процесу необхідно здійснити підключення до API, визначити необхідні параметри запиту та обробити отриману відповідь для відображення коректної інформації в застосунку.

Першим кроком є отримання ключа API від провайдера погодних даних та встановлення необхідних налаштувань для забезпечення безпечного та ефективного взаємодії з сервером (рис 3.6). Потім необхідно реалізувати

логіку взаємодії з API в програмному коді застосунку, включаючи формування запитів, обробку отриманих відповідей та відображення погодних даних у користувацькому інтерфейсі.

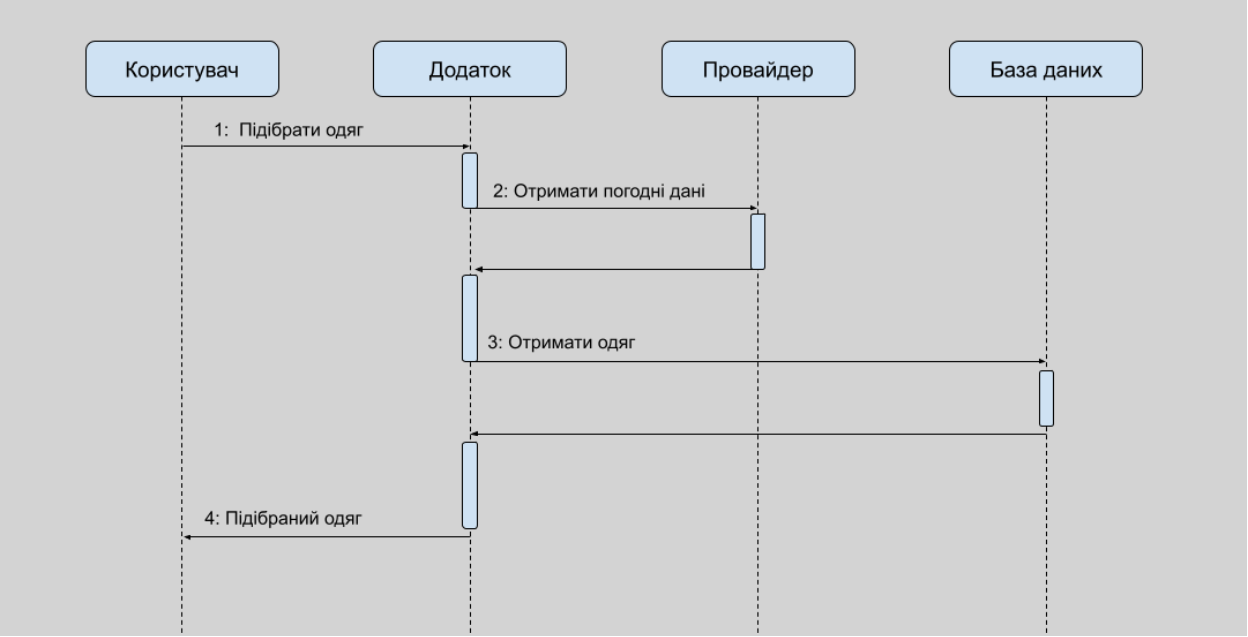

Рисунок 3.6 – Діаграма послідовності

Після отримання погодних даних від провайдера, наступним кроком є їх перетворення у внутрішній формат, зазвичай у форматі JSON (рис 3.7). Це необхідно для зручності подальшої обробки та відображення даних у користувацькому інтерфейсі. Конвертація у формат JSON дозволяє легко розбирати та використовувати ці дані в програмі, а також забезпечує їхню структурованість та доступність для подальшого аналізу [24].

Після цього оброблені дані можуть бути використані для відображення прогнозу погоди у застосунку та надання користувачу необхідної інформації про погодні умови. Після отримання JSON-даних, система зчитує їх, конвертує у зручний для обробки формат та виділяє необхідну інформацію, таку як температура, швидкість вітру, опади тощо. Ця інформація потім використовується для формування рекомендацій щодо підбору відповідного одягу для користувача відповідно до погодних умов у його регіоні.

Зчитування JSON-даних дозволяє системі отримувати та обробляти актуальну інформацію про погоду в реальному часі, що робить процес підбору одягу більш точним та ефективним для кінцевого користувача.

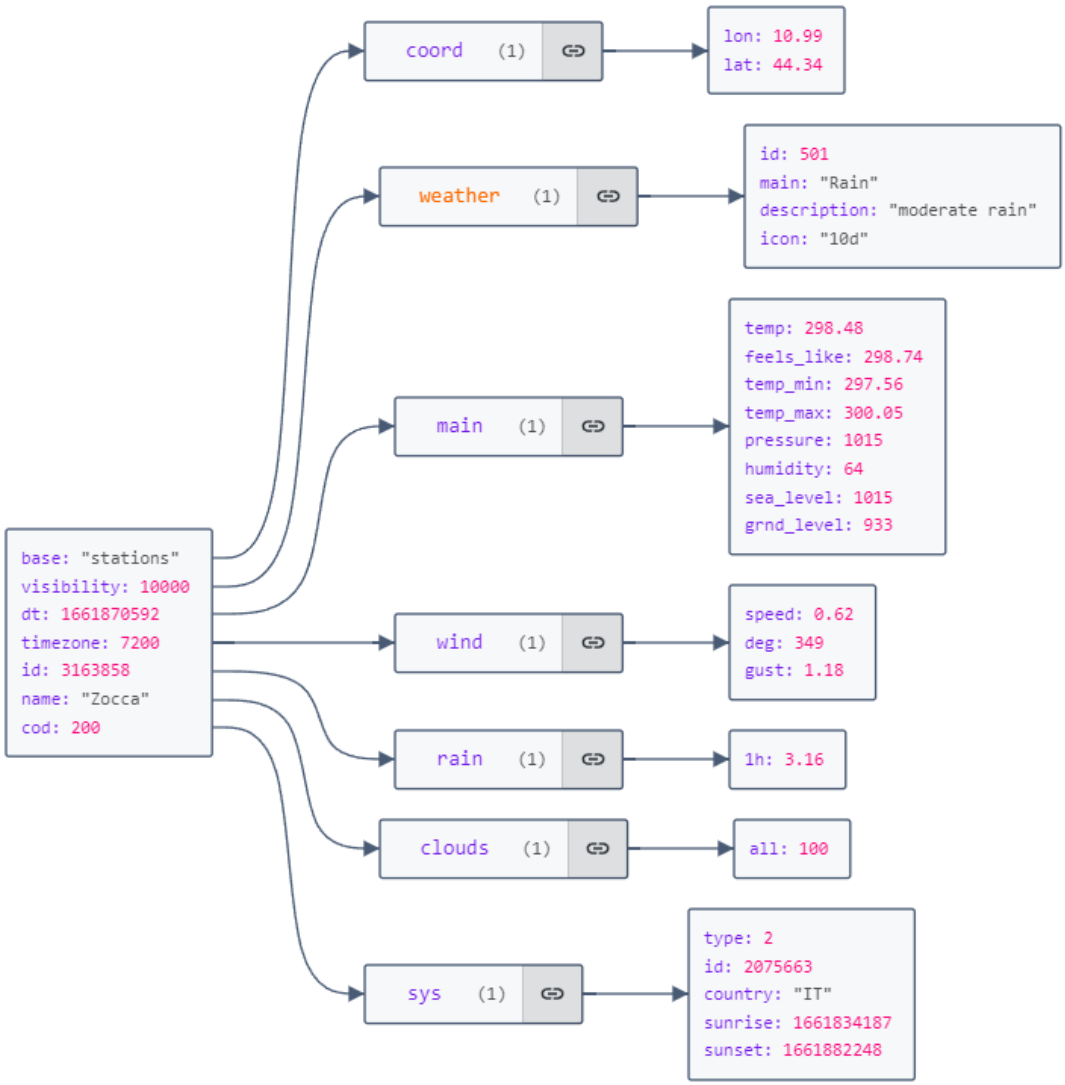

Рисунок 3.7 – Приклад погодних даних у форматі JSON

Після отримання даних про погоду внутрішній логіки програми використовуються для перетворення цих даних у відповідний набір речей за системою Extended Cold Weather Clothing System (ECWCS). Ця система базується на шаровому підході до одягу, де кожен шар має визначену функцію та використовується залежно від погодних умов. Логіка програми аналізує

отримані дані про погоду, такі як температура, вітер та опади, і відповідно до цих даних визначає необхідний комплект одягу. Наприклад, при низьких температурах та вітровому холоді програма може рекомендувати використання утеплених та вітрозахисних шарів одягу, в той час як при теплій та сухій погоді рекомендується використання більш легкого та протипотового одягу. Такий підхід дозволяє забезпечити користувачам оптимальний комфорт та захист у різних погодних умовах, відповідно до принципів ECWCS.

3.5 Розробка сервісу підбору одягу

Першим кроком у процесі розробки програмного модуля для підбору одягу за системою ECWCS є складання таблиці умов, яка відображає взаємозв'язок між різними погодними умовами та рекомендованим комплектом одягу. Ця таблиця включає в себе такі параметри, як температура, вітер, вологість, опади та інші метеорологічні показники, а також відповідний одяг для кожної з цих умов. Наприклад, для низьких температур та вітру може рекомендуватися використання утеплених шарів одягу, тоді як для високих температур і сонячної погоди - легких та дихаючих матеріалів. Складання цієї таблиці допомагає систематизувати інформацію та забезпечити зручний доступ до рекомендацій щодо підбору одягу в різних погодних умовах.

Після складання таблиці умов, наступним кроком є застосування правил до отриманих погодних даних [25]. Для цього модуль аналізує поточні метеорологічні умови, отримані з джерела погодних даних, такого як API метеорологічного сервісу. На основі цих даних модуль визначає, які саме умови відповідають поточному стану погоди, а потім використовує зазначені у таблиці умов правила для вибору відповідного комплекту одягу (рис 3.8).

Наприклад, якщо поточна температура низька, а вологість висока, модуль може визначити ці умови як "холодно та волого" і згідно таблиці умов рекомендувати одяг з утепленням та водовідштовхувальними властивостями. Такий підхід дозволяє системі автоматично визначати потрібний одяг для

конкретних погодних умов без необхідності вручну аналізувати всі аспекти погоди.

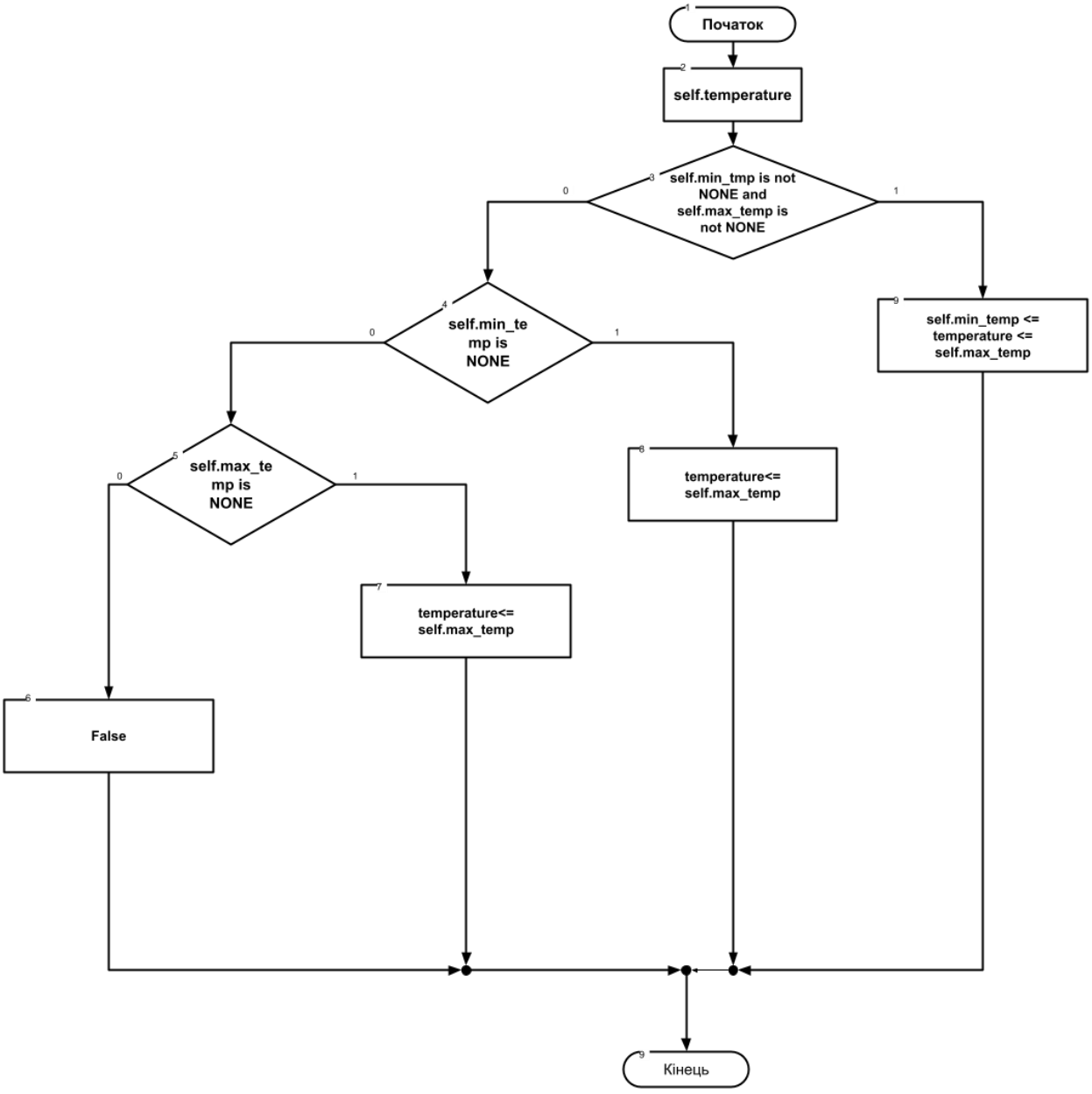

Рисунок 3.8 – Застосування правил одягу до погоди

Після визначення відповідного комплекту одягу для поточних погодних умов, модуль формує звіт для користувача. Цей звіт містить інформацію про рекомендований одяг, який найкраще підходить для зазначених погодних умов. У звіті можуть бути включені такі дані, як типи і шари одягу, які слід

носити, температурні рекомендації, захист від опадів та вітру, інструкції щодо зручного використання одягу та інші корисні поради.

Наприклад, звіт може містити рекомендації щодо носіння теплої куртки, шапки, рукавичок та взуття з водовідштовхувальними властивостями у випадку холодної та вологої погоди. Крім того, можуть бути вказані рекомендації щодо використання додаткових аксесуарів, наприклад, парасоля або захисного крему від сонця, в залежності від погодних умов.

Формування звіту допомагає користувачам краще розуміти, який одяг є найбільш відповідним для поточних погодних умов, та надає їм необхідну інформацію для комфортного перебування на вулиці.

Після формування звіту про рекомендований одяг на основі погодних умов, наш застосунок автоматично надсилає цей звіт користувачеві через обрану месенджерну платформу. Це може бути текстове повідомлення у Telegram, електронний лист або сповіщення у мобільному застосунку. Користувач отримує звіт з детальним описом рекомендованого одягу, що дозволяє йому готуватися до певних погодних умов заздалегідь. Це покращує користувацький досвід і забезпечує більш комфортне перебування під час виходу на вулицю.

3.6 Розробка інтерфейсу програмного застосунку

Вибір веб платформи для реалізації інтерфейсу програмного застосунку для підбору одягу виявився логічним та обґрунтованим кроком у процесі розробки. Веб застосунки є дедалі популярнішими серед користувачів, завдяки своїй доступності, зручності використання та широкому спектру функціональних можливостей. Враховуючи той факт, що вебзастосунку є платформою для комунікації, наявність ботів у цьому месенджері надає можливість надзвичайно зручного та простого взаємодії користувачів з програмним застосунком. Такий підхід також відкриває широкі перспективи у плані розширення та адаптації функціоналу за потребами користувачів, а також у впровадженні нововведень та оновлень у майбутньому.

При відкритті застосунку, користувача зустрічає екран із описом роботи програми (рис. 3.9), це дозволить користувачам відразу розуміти суть роботи застосунку.

Рисунок 3.9 – Початковий екран

Нижче користувач зможе побачити всі можливі типи одежі (рис 3.10). Це дозволить користувачам краще оцінити власний гардероб відповідно до вимог системи ECWCS. Кожен рівень одягу системи ECWCS забезпечує певний рівень захисту і комфорту, що дозволяє користувачам ефективно адаптувати свій одяг до змінних погодних умов. Завдяки наявності різних типів одягу для кожного шару, користувачі зможуть легко підібрати оптимальні комплекти для будь-якої ситуації, забезпечуючи необхідний баланс між теплоізоляцією, вологовідведенням та захистом від зовнішніх впливів. Це допоможе користувачам не тільки зберегти комфорт в

екстремальних умовах, але й ефективно використовувати свій гардероб для підтримання тепла і сухості в будь-яких погодних умовах.

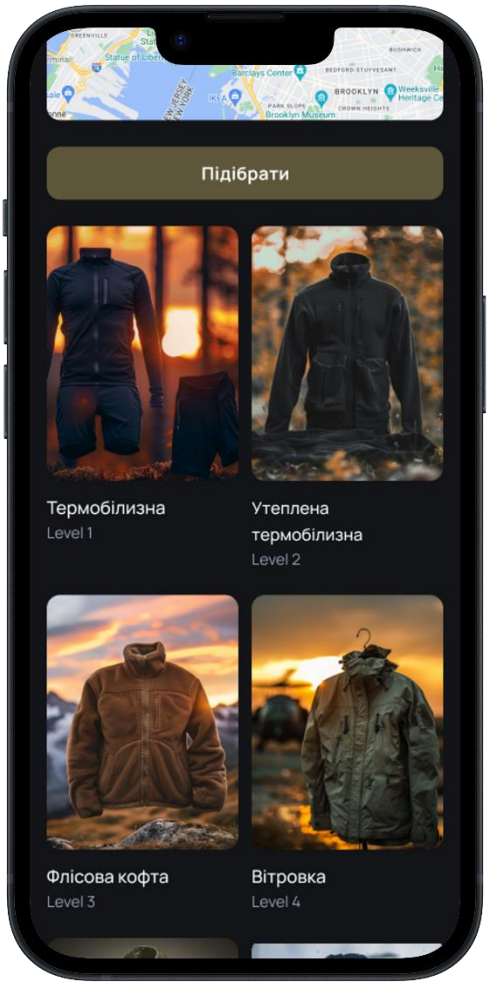

Рисунок 3.10 – Перелік одягу за системою ECWCS

Меню команд у вебзастосуноку представляє собою перелік доступних опцій або функцій, які користувач може викликати, використовуючи спеціальні команди або символи. Це зручний спосіб навігації та взаємодії з застосунком, оскільки дозволяє швидко отримувати доступ до різноманітних функцій безпосередньо з меню.

Також в меню команд можуть бути включені варіанти налаштувань або параметрів, які дозволяють користувачам налаштовувати різні аспекти роботи бота, такі як мова, часовий пояс або повідомлення про сповіщення.

У цілому, меню команд вебзастосуноку є потужним засобом для навігації та взаємодії з користувачем, який дозволяє забезпечити зручний та ефективний досвід використання (рис. 3.11).

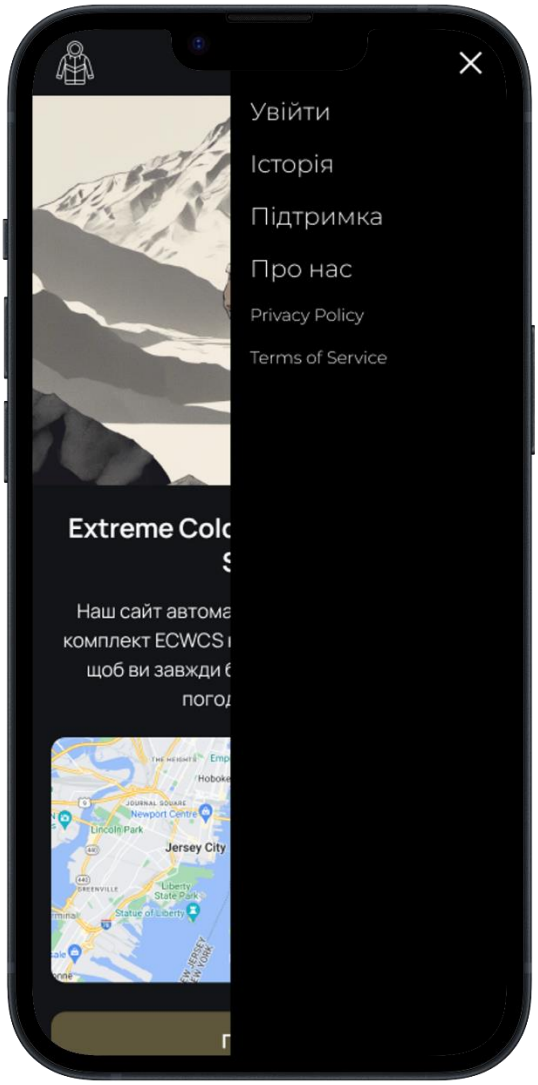

Рисунок 3.11 – Меню

Реєстрація в системі автоматизованого підбору одягу на основі системи ECWCS є важливим кроком, який дозволяє користувачам отримувати персоналізовані рекомендації (рис 3.12). Під час реєстрації користувачі вводять основні дані, такі як ім’я, електронну адресу та місцезнаходження. Це забезпечує можливість системи враховувати індивідуальні параметри та геолокацію користувача для точнішого визначення погодних умов у його

регіоні. Реєстраційний процес простий і швидкий, що робить його доступним для всіх користувачів.

Після завершення реєстрації користувач отримує доступ до особистого кабінету, де можна налаштувати додаткові параметри, такі як переваги в одязі, стиль життя та активність. Ці дані використовуються системою для покращення алгоритму підбору одягу, забезпечуючи більш точні та відповідні рекомендації. Користувач також може переглядати історію отриманих рекомендацій та коригувати свої налаштування в будь-який момент, що забезпечує гнучкість і адаптивність системи.

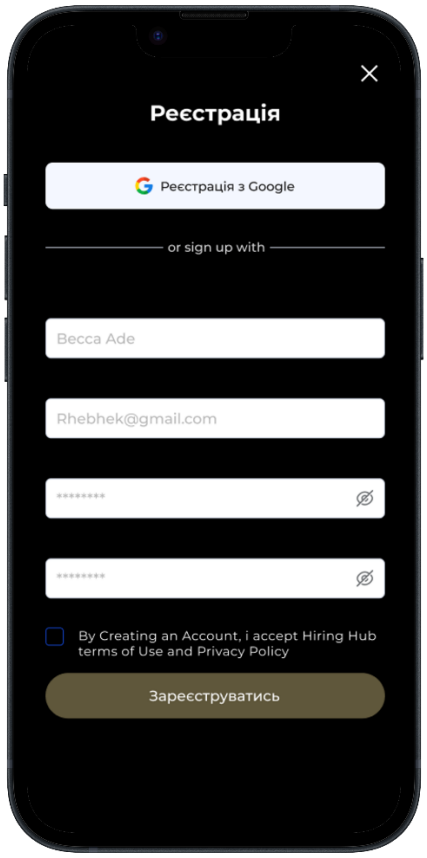

Рисунок 3.12 – Екран реєстрації

Авторизація в системі автоматизованого підбору одягу на основі системи ECWCS є важливим етапом, що забезпечує безпеку та персоналізований доступ користувачів до їхніх акаунтів. Після реєстрації користувачам надаються унікальні облікові дані, такі як ім’я користувача або

електронна адреса, а також пароль. Ці дані використовуються для входу в систему, гарантують захист особистої інформації та забезпечують доступ до індивідуальних налаштувань і рекомендацій (рис 3.13).

Процес авторизації простий і швидкий, що дозволяє користувачам легко входити до системи з будь-якого пристрою з підтримкою інтернету. Після введення облікових даних система перевіряє їх на відповідність, і в разі успіху користувач отримує доступ до свого особистого кабінету. Тут він може переглядати та редагувати свої налаштування, отримувати рекомендації щодо вибору одягу, враховуючи поточні погодні умови, та зберігати улюблені комплекти одягу.

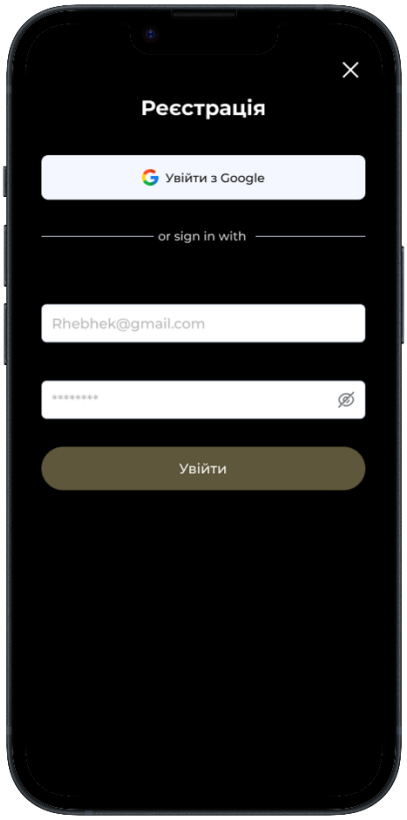

Рисунок 3.13 – Авторизація

Екран вибору геолокації є ключовим компонентом системи автоматизованого підбору одягу на основі системи ECWCS, оскільки він дозволяє користувачам точно визначити своє місцезнаходження для отримання актуальних погодних даних. Після авторизації або під час першого

налаштування, користувачеві пропонується обрати своє місцезнаходження за допомогою інтерактивної карти або автоматично, дозволивши доступ до даних геолокації пристрою.

Інтерактивна карта забезпечує зручний інтерфейс для вибору місця розташування. Користувач може збільшувати та зменшувати масштаб карти, переглядати різні райони та міста, і точно вказувати своє місцезнаходження, що особливо корисно в разі перебування в новому або маловідомому місці. Альтернативно, користувач може дозволити автоматичне визначення геолокації, що значно спрощує процес налаштування та забезпечує миттєве оновлення місцевих погодних умов.

Вибір геолокації є критичним для роботи системи, оскільки на основі цих даних алгоритми підбирають відповідний одяг, враховуючи поточні та прогнозовані погодні умови. Після вибору місцезнаходження система регулярно оновлює погодні дані, забезпечуючи користувача актуальними рекомендаціями. Таким чином, екран вибору геолокації не лише покращує користувацький досвід, але й гарантує, що рекомендації щодо одягу завжди будуть точними та релевантними (рис 3.14).

Крім основної функції вибору місцезнаходження, екран геолокації також надає користувачам можливість зберігати кілька місць для швидкого доступу до погодних даних у різних регіонах. Це особливо корисно для користувачів, які часто подорожують або мають кілька постійних місць перебування, таких як дім і робота. Збережені локації дозволяють миттєво перемикатися між різними місцями та отримувати актуальні рекомендації щодо одягу для кожного з них. Такий підхід забезпечує ще більшу гнучкість і зручність у використанні системи, роблячи її незамінним інструментом для планування щоденного гардеробу відповідно до змінних погодних умов.

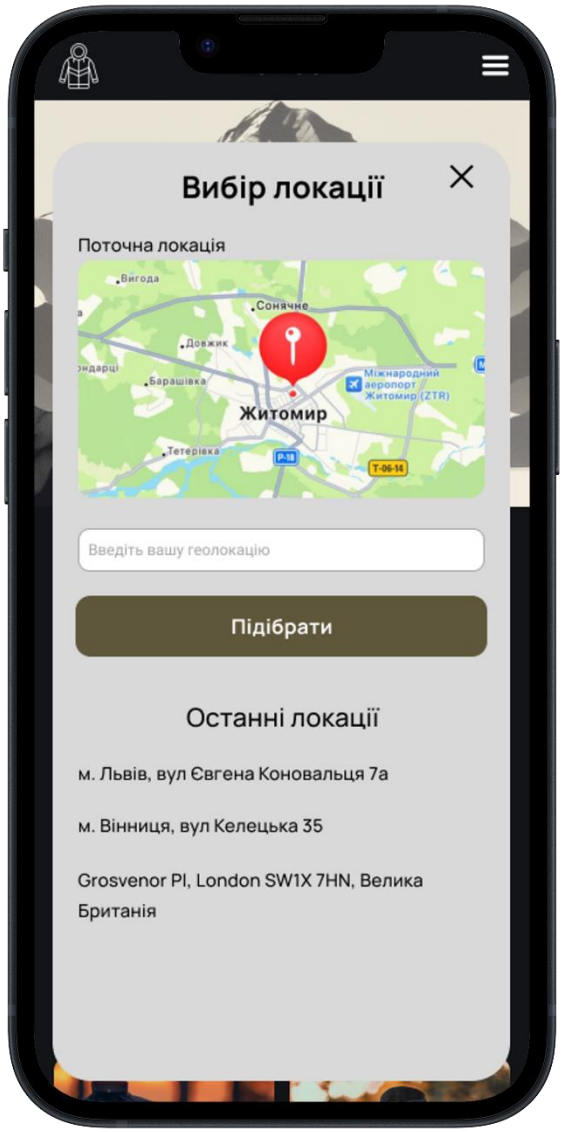

Рисунок 3.14 – Введення міста

Відображення результатів підбору одягу та погодних умов є важливими складовими інтерфейсу системи автоматизованого підбору одягу на основі системи ECWCS. Після того, як користувач обирає або система автоматично визначає його геолокацію, на екрані з’являються актуальні погодні умови для зазначеного місця. Ці дані включають температуру, вологість, швидкість та напрямок вітру, а також прогноз на найближчі дні, що дозволяє користувачам завчасно підготувати свій гардероб.

Результати підбору одягу відображаються у вигляді рекомендацій щодо конкретних предметів одягу з системи ECWCS, розбитих на відповідні рівні.

Наприклад, користувач може побачити, що для поточних погодних умов йому рекомендовано вдягнути термобілизну (перший рівень), утеплені штани і куртку (третій рівень), та водонепроникний верхній одяг (шостий рівень). Кожен рівень одягу супроводжується коротким описом і поясненням, чому саме ці предмети були обрані для поточних погодних умов (рис 3.15).

Для зручності користувачів, інтерфейс також передбачає можливість швидкого перегляду деталей кожного рекомендованого предмету одягу, включаючи його характеристики та поради щодо використання. Візуальні елементи, такі як іконки та зображення одягу, роблять інтерфейс більш зрозумілим і привабливим. Таким чином, користувач отримує чітке уявлення про те, як слід одягатися за поточних погодних умов, що сприяє комфорту та безпеці під час перебування на відкритому повітрі.

Рисунок 3.15 – Отримання кінцевої інформації

Пропонована одяг у системі базується на стандартах і специфікаціях системи ECWCS, що забезпечує високу якість та функціональність. Вона включає різні шари одягу, які допомагають користувачеві залишатися сухим та теплим навіть в екстремальних погодних умовах. Наприклад, в базовому комплекті входять термобілизна, яка забезпечує тепло та відводить вологу від шкіри, та водонепроникний верхній одяг, що захищає від дощу та вітру.

Крім того, пропонована одяг також враховує особливості різних активностей та середовищ. Наприклад, для активного відпочинку або роботи на відкритому повітрі, можуть бути використані дихаючі та еластичні матеріали, що забезпечують комфорт та свободу рухів. У той же час, для військових операцій чи рятувальних дій, може бути важливим захист від вологи та вітру, тому система також включає водонепроникні та утеплені верхній одяг. Такий різноманітний асортимент дозволяє користувачам обирати оптимальний одяг для різних умов та ситуацій (рис 3.16).

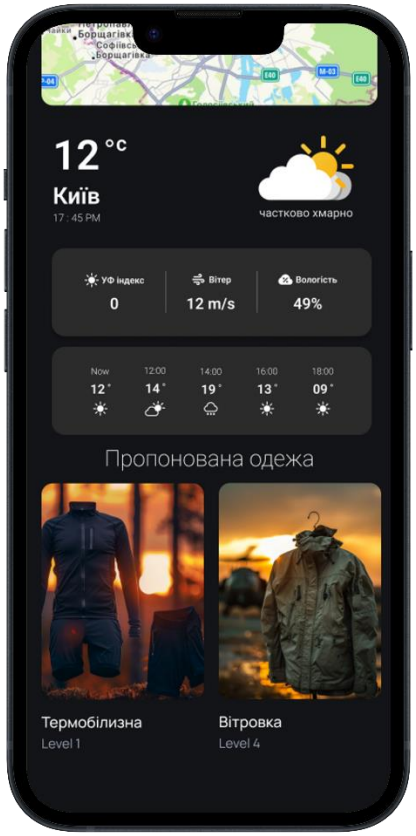

Рисунок 3.16 – Підібрана одежа

Розробка графічного інтерфейсу платформи була проведена у середовищі розробки Microsoft Visual Studio з використанням технології pyrogram.

3.7 Висновки

У результаті аналізу використаних технологій для розробки системи підбору одягу можна зробити кілька висновків. Перш за все, вибір мови програмування Python та використання різноманітних бібліотек дозволить нам забезпечити потрібний функціонал застосунку, враховуючи вимоги до швидкості роботи та гнучкості в розширенні. Використання бібліотек для обробки даних, роботи з геолокацією та інтеграції з погодними сервісами дозволить нам ефективно впроваджувати ключові функції застосунку.

Другим важливим висновком є необхідність вдосконалення механізмів обробки помилок та відлагодження програмного коду, особливо в контексті модуля геолокації та інтеграції з погодними сервісами. Це допоможе забезпечити стабільну та надійну роботу застосунку навіть у випадку виникнення непередбачених ситуацій.

Нарешті, важливою рисою вибраних технологій є їхня широка підтримка та активна спільнота розробників. Це дозволить нам отримати підтримку та вирішення можливих проблем швидко та ефективно, а також використовувати передові розробки та рішення для поліпшення функціоналу застосунку в майбутньому.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування модуля підбору одягу є невід'ємною частиною процесу розробки програмного продукту. Основною метою цього етапу є перевірка функціональності, безпеки та коректності роботи системи на мобільній платформі. Одним з ключових аспектів успішного тестування є забезпечення сумісності програми з різними пристроями та версіями операційних систем.

Для проведення тестування потрібно встановити тестову версію застосунку на мобільний пристрій, який використовуватиметься під час тестування. Після цього проводяться різноманітні функціональні тести для перевірки коректності роботи програми, включаючи тести на введення та виведення даних, а також на реакцію системи на різні вхідні сценарії.

Окрему увагу приділяється тестуванню безпеки програмного продукту. Це включає тести на проникнення, перевірку на збереження конфіденційної інформації та виявлення потенційних вразливостей програми. Для цього використовуються спеціальні інструменти, що дозволяють моделювати атаки на систему та виявляти її слабкі місця.

Після успішного проходження всіх тестів оцінюється продуктивність системи, включаючи швидкість роботи, використання ресурсів пристрою та загальний життєвий цикл програми. Результати тестування фіксуються у відповідній документації та використовуються для подальшого вдосконалення програми та забезпечення її якості на мобільній платформі.

Тестування у середовищі PyCharm є важливою складовою процесу розробки програмного забезпечення на мові Python. PyCharm надає розширені можливості для виконання тестів та відлагодження коду, що допомагає розробникам покращити якість своїх програм.

Один з основних інструментів для тестування у PyCharm - це модуль pytest, який дозволяє писати та запускати тести з легкістю. Розробники можуть

створювати тести для перевірки різних аспектів свого коду, таких як функціональність, коректність та ефективність.

Крім того, PyCharm має вбудовані засоби для відлагодження коду, які допомагають виявляти та виправляти помилки. Розробники можуть використовувати режим відлагодження для крокового виконання свого коду, а також використовувати точки зупинки для аналізу стану програми у конкретній точці виконання.

Загальний процес тестування та дебагінгу у PyCharm включає написання тестів, запуск їх для перевірки роботи коду, виявлення та виправлення помилок за допомогою відлагодження, а також використання додаткових інструментів для аналізу покриття коду тестами та виявлення потенційних проблем. Стрес-тестування є важливим етапом верифікації роботи автоматизованої системи підбору одягу. Простий стрес-тест полягає в навантаженні системи великою кількістю одночасних запитів для оцінки її стабільності та продуктивності під високим навантаженням.

Для проведення стрес-тесту використовується інструмент, такий як Apache JMeter або Locust, який дозволяє створити сценарій, що імітує велику кількість користувачів, які одночасно взаємодіють із системою. Під час тесту система отримує тисячі запитів до сервера, що моделює реальні умови експлуатації у пікові періоди.

Мета стрес-тесту полягає у виявленні точок відмови системи, визначенні її граничної продуктивності та оцінці здатності обробляти великий обсяг запитів без збоїв. Це дозволяє визначити, наскільки система здатна витримувати підвищені навантаження та чи необхідні додаткові оптимізації для забезпечення її стабільної роботи.

Генерація запитів є ключовим етапом у процесі стрес-тестування автоматизованої системи підбору одягу. Цей процес передбачає створення великої кількості запитів до системи, щоб оцінити її продуктивність і стійкість під високим навантаженням (рис 4.1).

Для генерації запитів зазвичай використовуються спеціалізовані інструменти, такі як Apache JMeter або Locust. Ці інструменти дозволяють налаштувати сценарії тестування, які імітують дії користувачів, що взаємодіють із системою. Наприклад, можна налаштувати сценарій, де кожен користувач здійснює запити для входу в систему, вибору геолокації, отримання рекомендацій щодо одягу та перегляду детальної інформації про одяг.

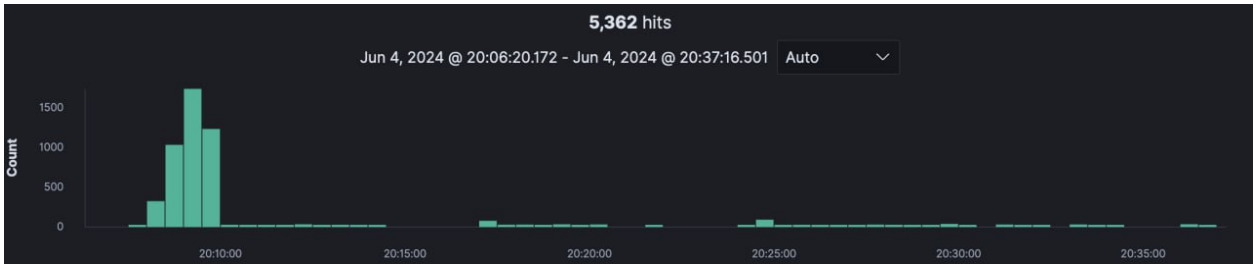

Рисунок 4.1 – Графік запитів на сервер

У процесі стрес-тестування важливим аспектом є моніторинг використання ресурсів системи, таких як процесор, оперативна пам'ять, мережевий трафік та диск. Відстеження цих параметрів дозволяє визначити, які компоненти системи є вузькими місцями та потребують оптимізації. Використання інструментів моніторингу, таких як Grafana або Prometheus, дозволяє в режимі реального часу отримувати інформацію про стан системи під навантаженням і приймати своєчасні заходи для покращення її продуктивності (рис. 4.2).

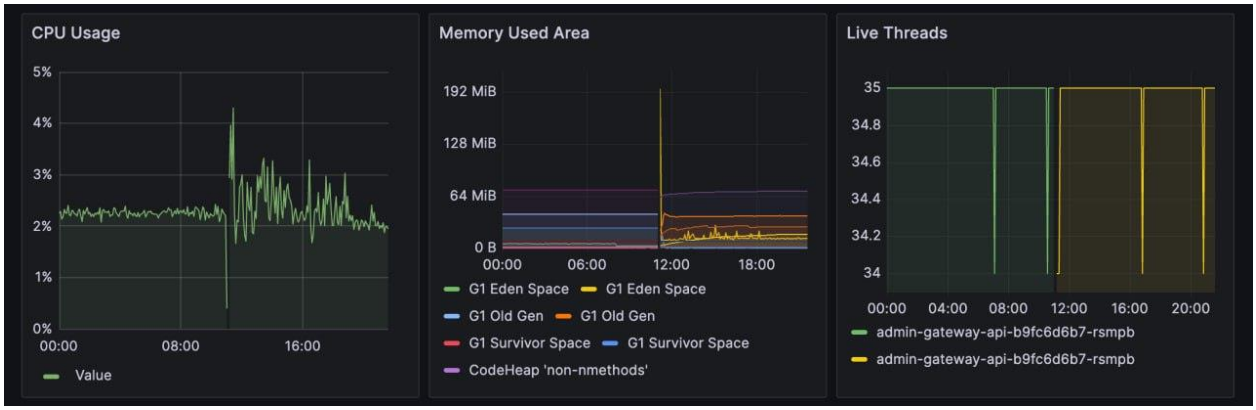

Рисунок 4.2 – Графік використання ресурсів системи

Використання Docker для контейнеризації компонентів системи також сприяє спрощенню процесу тестування та розгортання. Docker забезпечує ізоляцію середовища та дозволяє легко масштабувати систему, що є важливим при проведенні стрес-тестів і аналізі використання ресурсів. Завдяки контейнеризації можна швидко налаштувати тестове середовище та відтворити умови реальної експлуатації системи.

У рамках курсового проекту необхідно врахувати сценарії обробки некоректного введення користувачем місцезнаходження. Це важливо для забезпечення коректної роботи системи та зручного користувацького досвіду. При некоректному введенні місцезнаходження система повинна здатися на відповідне повідомлення, яке підкаже користувачеві про причину помилки та надасть вказівки щодо подальших дій. Такий підхід допоможе уникнути нерозумінь та забезпечить зручну взаємодію з програмою.

Крім того, важливо враховувати різноманітні ситуації, які можуть виникнути під час введення користувачем місцезнаходження. Наприклад, можливі помилки в написанні назв міст або використання невідомих або неіснуючих адрес. Враховуючи ці сценарії, система повинна надавати зрозумілі та добре структуровані повідомлення про помилки, що допоможе користувачам швидко виправити неправильно введені дані та продовжити використання програми без перешкод (рис 4.3).

Такий підхід до обробки некоректного введення дозволить створити користувацький інтерфейс, який буде інтуїтивно зрозумілим і зручним для використання. Крім того, це сприятиме покращенню загального враження від використання програми та забезпечить задоволення користувачів від їхнього досвіду.

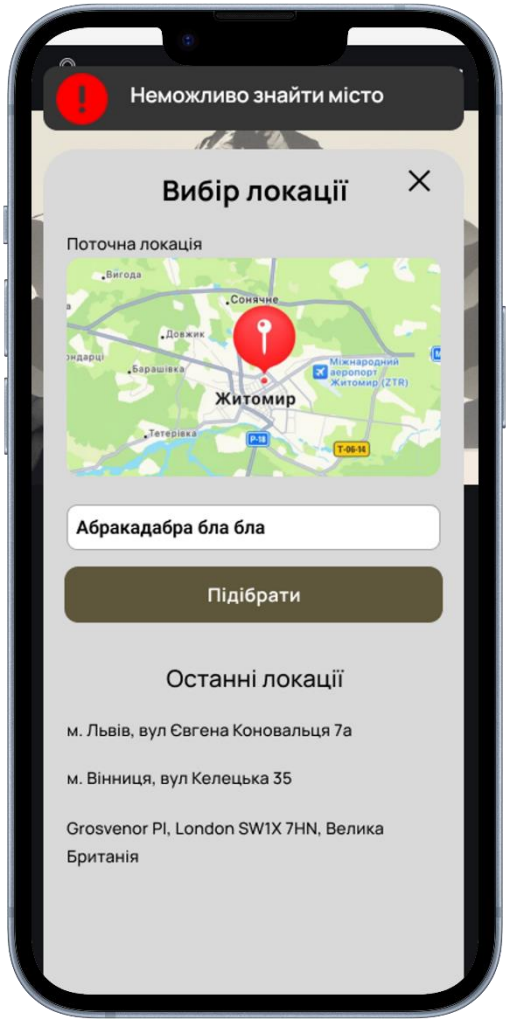

Рисунок 4.3 – Некоректна назва міста

Фільтрація незрозумілих вводів від користувача є важливою складовою процесу розробки програмного забезпечення. Користувачі можуть намагатися ввести різноманітні дані, які не відповідають очікуваному формату або не зрозумілі для системи. У таких випадках важливо вміти відрізняти коректні дані від некоректних та правильно їх обробляти.

Фільтрація незрозумілих вводів дозволяє підтримувати стабільну та безпечну роботу програми. Неправильні дані можуть спричинити виникнення помилок або некоректну роботу системи, що може вплинути на користувацький досвід та негативно вплинути на репутацію програми. Фільтрація дозволяє зменшити ймовірність виникнення таких проблем та забезпечити стабільну роботу програмного забезпечення в будь-яких умовах.

Крім того, фільтрація незрозумілих вводів допомагає забезпечити безпеку системи. Некоректні дані можуть бути використані для зловживання або атак на програму, що може призвести до порушення конфіденційності даних або втрати функціональності програми. Фільтрування таких вводів дозволяє уникнути потенційних загроз безпеці та зберегти цілісність системи.

Під час процесу розробки програмного забезпечення, використання фільтрації незрозумілих вводів є ключовим етапом в забезпеченні якості продукту. Цей підхід дозволяє розробникам виявити потенційні помилки та некоректність даних ще на ранніх етапах розробки, що значно зменшує час і витрати на виправлення проблем пізніше у процесі. Застосування фільтрації сприяє стабільності та надійності програмного забезпечення, а також дозволяє забезпечити високу якість фінального продукту.

Крім того, фільтрація незрозумілих вводів може позитивно вплинути на вартість розробки та підтримки програмного забезпечення. Заощаджені кошти, які були б витрачені на виправлення помилок та реагування на неправильні вводи після випуску продукту, можна перерозподілити на інші аспекти розробки, такі як вдосконалення функціональності або розширення можливостей програми. Такий підхід допомагає забезпечити оптимальне використання ресурсів та зниження загальних витрат на життєвий цикл продукту.

Продовження коректної роботи програми незважаючи на збої є важливою характеристикою надійного програмного забезпечення. Збої можуть виникати з різних причин, таких як неправильні дані введені користувачем, помилки в коді програми або непередбачені обставини. Однак, навіть у разі збоїв програма повинна вести себе контрольовано та продовжувати свою роботу в межах можливостей (рис 4.4).

Рисунок 4.4 – Некоректна назва міста

Забезпечення продовження коректної роботи програми можна здійснити за допомогою механізмів обробки помилок та винятків. Відповідні блоки коду можуть виявляти та обробляти помилки, що допомагає запобігти виникненню критичних ситуацій і забезпечує безперебійну роботу програми.

Крім того, розумна обробка даних може допомогти уникнути виникнення збоїв та забезпечити стабільну роботу програми. Використання валідації даних та перевірок перед їх використанням може запобігти некоректним операціям та вибуху помилок.

Також важливо мати механізми резервного копіювання та відновлення даних, що дозволяє відновлювати роботу програми після непередбачуваних збоїв або випадків втрати даних.

Усі ці підходи допомагають забезпечити надійну та безперебійну роботу програми навіть у разі виникнення збоїв, що робить програмне забезпечення більш стійким та надійним.

У процесі тестування було перевірено функціональність, безпеку та коректність роботи програмного продукту. Результати тестування свідчать про високу ефективність програми в умовах реального використання. Функціональні тести показали правильну реакцію програми на різноманітні вхідні дані та відповідність її бізнес логіці. Тести на безпеку підтвердили відсутність вразливостей і надійність заходів безпеки.

Додатково було проведено тести на продуктивність, що підтвердили швидкість та ефективність роботи програми, а також її стійкість до великої кількості запитів та навантаження. Усі результати тестування документувалися та використовувалися для подальшого вдосконалення програмного продукту.

Загалом, тестування підтвердило високу якість та надійність програмного забезпечення, що дозволить користувачам впевнено використовувати його у різних умовах та забезпечить стабільну та безперебійну роботу системи.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Крім того, інструкція включає розділ часто задаваних питань (FAQ) та поради з вирішення поширених проблем. Це дозволяє користувачам самостійно знаходити відповіді на питання, що виникають під час роботи з системою, та зменшує навантаження на службу підтримки.

Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Частота процесора	1 ГГц
Об’єм оперативної пам’яті	1 ГБ
Місце на жорсткому диску	1 ГБ
Операційна система	Windows 7/8/10/11

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	2 ГГц
Об’єм оперативної пам’яті	4 ГБ
Місце на жорсткому диску	2 ГБ
Операційна система	Windows 7/8/10/11

Для встановлення автоматизованої системи підбору одягу потрібно використовувати Docker – це платформа, яка дозволяє швидко та ефективно розробляти, тестувати та розгортати додатки. Docker упаковує програмне забезпечення в контейнери, які містять усі необхідні компоненти для роботи програми, включаючи бібліотеки, системні інструменти, код та середовище виконання [26]. Використання Docker дозволяє користувачам швидко розгортати та масштабувати систему підбору одягу в будь-якому середовищі, забезпечуючи стабільну роботу програми. Це також зменшує необхідність вручного налаштування середовища, оскільки користувачам достатньо виконати кілька команд для запуску контейнерів на онлайн-платформі.

Після входу в систему користувач має доступ до головної сторінки, де він може обрати свою геолокацію. На основі вибраної геолокації система підбору одягу надасть список рекомендованих елементів гардеробу, відповідно до поточних погодних умов. Користувач може ознайомитися з детальним описом рекомендованого одягу, включаючи інформацію про

матеріали, з яких він виготовлений, та поради щодо використання. Після перегляду рекомендацій користувач може зберегти свій вибір та переглянути історію попередніх рекомендацій для аналізу та порівняння.

4.3 Висновки

Під час тестування системи проведено ретельний аналіз функціональності, безпеки та коректності роботи програми на десктопній платформі. Виявлені та виправлені помилки дозволили забезпечити надійну та стабільну роботу застосунку.

Тестування також дозволило переконатися у сумісності програми з різними версіями операційної системи та виявити та виправити можливі проблеми в цьому напрямку.

Проведені тести на безпеку програми підтвердили її стійкість до атак та вразливостей, що дозволяє забезпечити конфіденційність та безпеку користувачів.

У результаті успішного проходження тестів на продуктивність було підтверджено, що програма працює швидко та ефективно, не надмірно використовуючи ресурси пристрою.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмні засоби реалізації системи автоматизованого підбору одягу з метою поліпшення процесу вибору одягу для користувачів в різних погодних умовах. Проведено аналіз стану проблеми на сьогоднішній день. Розглянуто існуючі аналоги, такі як: Looksize, Rozetka та Shop the Look. Визначено їхні переваги та недоліки в порівнянні з розробленою автоматизованою системою підбору одягу на основі системи ECWCS та встановлено, що дана система переважає у функціональності. Для розробки було використано середовище програмної розробки PyCharm 2022.

Базуючись на результатах аналізу та порівняння було встановлено основні задачі бакалаврської дипломної роботи та обрано відповідні методи розробки. Під час аналізу технологій розробки було обґрунтовано вибір мови програмування Python та технологію розробки графічного інтерфейсу pyrogram.

Відповідно до поставлених задач було:

– реалізовано систему, що використовує сторонні API, які надають погодні дані та іншу необхідну інформацію;

– інтегровано сторонні API для отримання актуальних погодних даних за отриманою геолокацією;

– розроблено метод і модель роботи системи та алгоритм підбору одягу, який буде враховувати отримані погодні умови;

– забезпечено роботу зручного інтерфейсу користувача;

– проведено тестування та оптимізацію роботи розробленої системи.

Було розроблено схеми загального алгоритму роботи системи та модуля тестування. Також було розроблено модель системи з використанням UML діаграм.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

Бакалаврську дипломну роботу було оформлено відповідно до методичних вказівок [27].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Американський комплекс одягу. URL: https://profarmor.com.ua/overviews/1057009-amerikanskiy-vsepogodnyy-kompleks-odezhdy-armii-ssha-extended-cold-weather-clothing-system-generation-iii-ecwcs-gen-iii.html.

2. Як вибрати одяг по погоді. URL: https://marieclaire.ua/uk/fashion/yak-vibrati-odyag-po-pogodi-praktichni-poradi.

3. Гуменюк Р.О. Розробка автоматизованої системи підбору одягу з використанням телеграм бота на основі системи ECWCS / Гуменюк Р.О., В. В. Войтко // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024.. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20403/16963

4. С. О. Полковниченко Оцінка сучасного стану розвитку ринку одягу в Україні / С. О. Полковниченко // Національний університет «Чернігівська політехніка», Чернігів, 2021. URL: http://www.economy.nayka.com.ua/pdf/6_2021/87.pdf.

5. ТОП-5 додатків, які допоможуть тримати гардероб під контролем. URL: https://maximum.fm/mobilnij-stilist-top-5-dodatkiv-yaki-dopomozhut-trimati-garderob-pid-kontrolem_n152435.

6. Looksize. URL: https://www.looksize.com/ua/

7. Rozetka. URL: https://rozetka.com.ua/ua/

8. Shop the look. URL: https://www.amazon.com/shopthelook/

9. Python Documentation. URL: https://docs.python.org/3/

10. Geo-Python Documentation. URL: https://geo-python-site.readthedocs.io/en/latest/

11. OpenWeather Map. URL: https://pytba.readthedocs.io/en/latest/index.html

12. pyTelegramBotAPI’s documentation. URL: https://openweathermap.org/weathermap?basemap=map&cities=true&layer=temperature&lat=30&lon=-20&zoom=5

13. Дізнайтеся все про діаграму активності UML. URL: https://www.mindonmap.com/uk/blog/uml-activity-diagram/

14. Майлс Р., Гамільтон К. Вивчення UML 2.0: прагматичне введення в UML / Расс Майлс, Кім Гамільтон. – Київ: КупиЧитай, 2021. – 289с.

15. Дізнайтеся все про діаграму активності UML. URL: https://www.mindonmap.com/uk/blog/uml-activity-diagram/

16. Бхардвадж А. Грокаємо алгоритми: ілюстрований посібник для програмістів і допитливих / Адітья Бхардвадж. – Київ: Балка-Бук, 2023. – 400с.

17. Ramalho L. Fluent Python, 2nd Edition / Luciano Ramalho. – Sebastopol: O'Reilly Media, 2022. – 1014c.

18. Welcome to GeoPy’s documentation. URL: https://geopy.readthedocs.io/en/stable/

19. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Martin Kleppmann. – 1st Edition: O'Reilly Media, 2017. – 616c.

20. Сесіл Р.М. Чиста архітектура / Сесіл Р.М., – Харків: Фабула, 2019. – 359 с.

21. Chapter 1. Layered Architecture. URL: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

22. Що таке API?. URL: https://dev.ua/news/chto-takoe-api-prostym-yazykom

23. Ел-Равас А. GPS / Аҳмад Ел-Равас. – Кембридж, Массачусетс: MIT Press, 2019. – 248с.

24. Вступ у JSON. URL: https://www.json.org/json-uk.html

25. DESIGN PATTERNS. URL: https://refactoring.guru/design-patterns

26. Poulton N. Docker Deep Dive / Nigel Poulton. – Kindle Edition: Amazon.com, 2023. – 300с.

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

ДОДАТКИ

67

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"__12__" ___березня_____2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу та програмних засобів автоматизованої системи підбору одягу»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. Коваленко О. О.
«_12_» __березня__ 2024 р.

Виконав:
_____студент гр. ЗПІ-20б Гуменюк Р. О.
«_12_» __березня__ 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу та програмних засобів автоматизованої системи підбору одягу».

Галузь застосування – електронна комерція.

2. Підстава для розробки.

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

3. Мета та призначення розробки.

Метою розробки автоматизованої системи підбору одягу за системою ECWCS є поліпшення процесу вибору одягу для користувачів в різних погодних умовах та забезпечення надійності рекомендацій системи. За рахунок розробки алгоритмів аналізу погодних умов та визначення відповідного одягу, а також створення інтерфейсу для взаємодії з користувачем.

Призначення роботи – розробка автоматизованої системи підбору одягу для забезпечення користувачів надійними та оптимальними рекомендаціями щодо одягу в різних погодних умовах.

4. Вихідні дані для проведення НДР

1. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Martin Kleppmann. – 1st Edition: O'Reilly Media, 2017. – 616с.

2. Сесіл Р.М. Чиста архітектура / Сесіл Р.М., – Харків: Фабула, 2019. – 359 с.

3. Бхардвадж А. Грокаємо алгоритми: ілюстрований посібник для програмістів і допитливих / Адітья Бхардвадж. – Київ: Балка-Бук, 2023. – 400с.

4. Ramalho L. Fluent Python, 2nd Edition / Luciano Ramalho. – Sebastopol: O'Reilly Media, 2022. – 1014с.

5. Технічні вимоги

Серверний комп’ютер з 64-розрядним (x64) процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 4 ГБ, розмір жорсткого диску 2 ГБ. Операційна система Windows 7/8/10/11.

З клієнтського боку: ОС Windows, Android, IOS та будь-який браузер.

6. Конструктивні вимоги.

Система має відповідати функціональним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

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

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку навчальних систем та постановка задач дослідження	14.03.2024 – 31.03.2024
2	Розробка методу, моделі та алгоритмів роботи вебсистеми	01.04.2024– 15.04.2024
3	Розробка модулів вебсервісу	16.04.2024 – 18.05.2024
4	Тестування програми	19.05.2024 – 24.05.2024
5	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.2024

10. Порядок контролю та прийняття

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

71

Додаток Б – ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу і засобів автоматизованої системи підбору одягу»

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Коваленко О.О.

Оригінальність	97,2%
Схожість	2,8%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek

Автор роботи _____ Гуменюк Р. О.

Керівник роботи _____ Коваленко О.О.

Додаток В – Лістинг програми

```
//  main.py
//
//  Created by Гуменюк Руслан on 26.02.2024.
//
import logging
import os
import telebot
import traceback
from telebot.util import quick_markup
from dotenv import load_dotenv


from exceptions import FetchingWeatherError
from models import ActivityType
from services import LocationService, WeatherService, ResponseService


load_dotenv()


BOT_TOKEN = os.environ.get('TELEGRAM_BOT_API_ECWCS')
POLLING_TIMEOUT = None
bot = telebot.TeleBot(BOT_TOKEN)


FORMAT = '%(asctime)s - %(name)s - %(levelname)s - %(message)s'


logger = logging.getLogger(__name__)
logging.basicConfig(format=FORMAT)
logging.getLogger().setLevel(logging.INFO)


location_service = LocationService()
response_service = ResponseService(logger)




@bot.message_handler(commands=['start'])
def send_welcome(message):
   '''
   returns a welcome message when the '/start' command is sent by the user
   '''
   bot.send_message(message.chat.id, "Вітаємо в чаті!")




@bot.message_handler(commands=['weather'])
def send_weather(message):
   markup = quick_markup({
       'Активна': {'callback_data': ActivityType.ACTIVE.value},
```

```
        'Статична': {'callback_data': ActivityType.STATIC.value}
    }, row_width=2)
    bot.send_message(message.chat.id, "Який ваш тип дяльності:", reply_markup=markup,
parse_mode='Markdown')



@bot.callback_query_handler(func=lambda call: True)
def callback_query(call):
    if call.data == ActivityType.ACTIVE.value or call.data ==
ActivityType.STATIC.value:
        display_text = "Введіть місто:"
        sent_message = bot.edit_message_text(chat_id=call.message.chat.id,
                                             message_id=call.message.message_id,
                                             text=display_text)
        bot.register_next_step_handler(sent_message, weather_handler, call.data)



def weather_handler(message, activity_type):
    location = message.text


    try:
        location_data = location_service.get_location_data(location)
        latitude = location_data.latitude
        longitude = location_data.longitude


        response = response_service.get_weather_response_message(latitude, longitude,
location_data, activity_type)


        bot.send_location(message.chat.id, latitude, longitude, protect_content=True)
        bot.send_message(message.chat.id, response, parse_mode='Markdown')
    except AttributeError:
        bot.send_message(message.chat.id, "Не можемо знайти таке місто. Спробуйте ще
раз.")
    except FetchingWeatherError as e:
        logging.error(
            f"Error fetching weather data, chat id = {message.chat.id}, exception:
{e}\n{traceback.format_exc()}")
        bot.send_message(message.chat.id, "Помилка при отриманні погоди. Спробуйте ще
раз пізніше.")
    except Exception as e:
        logging.error(f"Error, chat id = {message.chat.id}, exception:
{e}\n{traceback.format_exc()}")
        bot.send_message(message.chat.id, "Помилка. Спробуйте ще раз пізніше.")



@bot.message_handler(commands=['help'])
def send_help(message):
```

```
   bot.send_message(message.chat.id, 'Це ECWCS бот. Ви можете використовувати такі
команди:\n'
                                     '/start - щоб розпочати роботу з ботом\n'
                                     '/weather - щоб отримати погоду для певного
місця\n'
                                     '/help - щоб отримати довідку')



@bot.message_handler(func=lambda msg: True)
def echo_all(message):
   bot.reply_to(message, "Не зрозуміло, що ти хочеш від мене. Спробуй ще раз.")



# @bot.message_handler(content_types=[
#     "new_chat_members"
# ])
# def chat_join_request(message):
#     bot.send_message(message.chat.id, "Вітаємо в чаті!")
#     bot.send_message(message.chat.id,
"https://www.youtube.com/watch?v=NO2kaIecfIE&ab_channel=Reptiloid")


bot.infinity_polling()


from enum import Enum



class LocationData:
   def __init__(self, location_data):
       self.latitude = location_data.latitude
       self.longitude = location_data.longitude
       self.address = location_data.address



class ActivityType(Enum):
   ACTIVE = 'active'
   STATIC = 'static'


   @staticmethod
   def from_str(label):
       if label in 'active':
           return ActivityType.ACTIVE
       elif label in 'static':
           return ActivityType.STATIC
       else:
           raise NotImplementedError
```

```
class ClothingLevel:
   def __init__(self, level, name, description=None):
       self.level = level
       self.name = name
       self.description = description


   def __repr__(self):
       return f"{self.level}: {self.name}"




class ClothingRuleWeatherCondition:
   def __init__(self, min_temp, max_temp, wet):
       self.min_temp = min_temp
       self.max_temp = max_temp
       self.wet = wet


   def is_weather_condition_satisfied(self, temperature):
       if self.min_temp is not None and self.max_temp is not None:
           return self.min_temp <= temperature <= self.max_temp


       if self.min_temp is None:
           return temperature <= self.max_temp
       if self.max_temp is None:
           return temperature >= self.min_temp


       return False


   def __repr__(self):
       return f"Temperature: {self.min_temp}-{self.max_temp}, Wet: {self.wet}"




class ClothingRule:
   def __init__(self, activity_type, weather_condition, clothing_levels):
       self.activity_type = activity_type
       self.weather_condition = weather_condition
       self.clothing_levels = clothing_levels


   def __repr__(self):
       return f"Activity: {self.activity_type}, Weather: {self.weather_condition},
Clothing: {self.clothing_levels}"


   def is_suitable(self, activity_type, temperature, wet):
       return (self.activity_type == activity_type
```

```
                and
self.weather_condition.is_weather_condition_satisfied(temperature))


    def is_weather_condition_satisfied(self, temperature):
        return self.weather_condition.is_weather_condition_satisfied(temperature)



from enum import Enum


from geopy import Nominatim


from models import LocationData, ClothingLevel, ClothingRule, ActivityType,
ClothingRuleWeatherCondition
from weather_api import WeatherApi


WEATHER_TOKEN = "64d8e77fe734abb85e9e8efd9fc430bb"




class LocationService:
    def get_location_data(self, location) -> LocationData:
        # Create a geocoder instance
        geolocator = Nominatim(user_agent="my_app")


        location_data = geolocator.geocode(location)


        return LocationData(location_data)




class Weather:
    def __init__(self, description, visibility, temperature, feels_like, humidity,
wind_speed, wind_direction):
        self.description = description
        self.visibility = visibility
        self.temperature = temperature
        self.feels_like = feels_like
        self.humidity = humidity
        self.wind_speed = wind_speed
        self.wind_direction = wind_direction


    def __str__(self):
        return (f"*Погода:* {self.description}\n"
                f"*Видимість:* {self.visibility} м\n"
                f"*Температура:* {self.temperature}°C\n"
                f"*Відчувається як:* {self.feels_like}°C\n"
                f"*Вологість:* {self.humidity}%\n"
```

```
                f"*Швидкість вітру:* {self.wind_speed} м/с\n"
                f"*Напрям вітру:* {self.wind_direction}°")



class WeatherService:


   def __init__(self, api_key, logger):
       self.weather_api = WeatherApi(api_key, logger)
       self.logger = logger


   def get_weather_message(self, latitude, longitude):
       weather = self.weather_api.get_weather(latitude, longitude)


       description = weather['weather'][0]['description']
       visibility = weather['visibility']
       temperature = weather['main']['temp']
       feels_like = weather['main']['feels_like']
       humidity = weather['main']['humidity']
       wind_speed = weather['wind']['speed']
       wind_direction = weather['wind']['deg']


       return Weather(description, visibility, temperature, feels_like, humidity,
wind_speed, wind_direction)




def __map_activity_text__(activity_type):
   if activity_type == 'active':
       return 'Активна'
   elif activity_type == 'static':
       return 'Статична'
   else:
       return 'Невідомо'




class ResponseService:
   def __init__(self, logger):
       self.logger = logger
       self.weather_service = WeatherService(WEATHER_TOKEN, logger)


   def get_weather_response_message(self, latitude, longitude, location_data,
activity_type):
       activity_type_text = __map_activity_text__(activity_type)
       weather = self.weather_service.get_weather_message(latitude, longitude)


       clothing_service = ClothingService()
```

```
       cloth =
clothing_service.get_cloth_by_weather(ActivityType.from_str(activity_type),
weather.feels_like, False)
       cloth_text = "Не вдалося підібрати одяг для такої погоди."
       if cloth is not None:
           cloth_text = "\n".join([f"{c.level}: {c.name}\n{c.description}\n" for c in
cloth])


       return (f"*Погода для міста:* {location_data.address}\n"
               f"*Діяльність:* {activity_type_text}"
               f"\n\n"
               f"{weather}"
               f"\n\n"
               f"*Одяг для такої погоди:* \n"
               f"{cloth_text}")



level_1 = ClothingLevel(1,
                        "Термобілизна",
                        "Термобілизна з вологостійких матеріалів, призначена для"
                        " відведення вологи від тіла та збереження тепла."
                        )
level_2 = ClothingLevel(2, "Утеплена термобілизна",
                        "Цей рівень включає в себе середньої товщини утеплювачі, "
                        "які можна надягати поверх тонкого нижнього шару."
                        " Це може бути флісовий светр, куртка або брюки.")
level_3 = ClothingLevel(3, "Флісова кофта",
                        "Третій рівень - дихаюча флісова куртка, яка швидко сохне."
                        " Це чудовий ізолятор, створюючи повітряні кишені для
збереження тепла "
                        "тіла. Неймовірно універсальний, цей фліс використовується як
"
                        "основний зовнішній шар у прохолодну погоду при температурі "
                        "вище нуля або використовується разом із п’ятим, шостим і "
                        "сьомим шарами, коли температура стає значно нижче нуля.")
level_4 = ClothingLevel(4, "Вітровка",
                        "Четвертий рівень – вітрова куртка, створена для штормових
умов."
                        " Його основне призначення — не захист від температури, а
захист від вітру"
                        " та дощу в погану погоду. Його нейлонова/спандексна тканина
має "
                        "водостійке покриття, щоб солдати були сухими.")
level_5 = ClothingLevel(5, "Softshell куртка та штани",
                        "Куртка та штани п’ятого рівня забезпечують надійний захист
від холодної погоди,"
                        " залишаючись комфортно легкими. Вони також менш важкі та
громіздкі,"
                        " ніж подібний одяг до них. Створений для носіння в широкому
діапазоні "
                        "температур, п’ятий рівень чудово підходить з обмеженою
кількістю"
                        " базових шарів, коли погода вище нуля, і носіння з більшістю
(або всіма)"
```

```
                        " попередніми шарами в сильний холод.")
level_6 = ClothingLevel(6, "Зовнішня оболонка куртки та штанів",
                        "Штани та куртка шостого рівня розроблені для незамерзаючих "
                        "температур під час сидячого життя. Їхня двошарова тканина
GORE-TEX "
                        "із ущільненими швами забезпечує елітний захист від негоди. "
                        "Щоб запобігти перегріванню, цей зовнішній шар не слід носити"
                        " під час напруженої діяльності.")
level_7 = ClothingLevel(7, "Утеплені куртка та штани",
                        "Парка та штани сьомого рівня призначені для
найекстремальніших, "
                        "схожих на арктичні умови. Розрахований на температуру до -
60°F, "
                        "цей одяг допомагає солдатам зігріватися та бути в безпеці,
коли "
                        "температура значно нижче нуля. Сьомий рівень носить разом з
іншими рівнями, "
                        "щоб забезпечити максимальну ізоляцію та захист від небезпечно
низьких температур. "
                        "Щоб запобігти перегріванню, цей одяг не слід носити під час
напруженої діяльності.")


clothing_rules = [
   ClothingRule(ActivityType.STATIC, ClothingRuleWeatherCondition(7, None, True),
[level_1, level_4, level_6]),
   ClothingRule(ActivityType.STATIC, ClothingRuleWeatherCondition(-1, 7, True),
[level_1, level_3, level_5, level_6]),
   ClothingRule(ActivityType.STATIC, ClothingRuleWeatherCondition(-13, -1, False),
               [level_2, level_3, level_5, level_7]),
   ClothingRule(ActivityType.STATIC, ClothingRuleWeatherCondition(None, -13, False),
[level_2, level_7]),


   ClothingRule(ActivityType.ACTIVE, ClothingRuleWeatherCondition(7, None, True),
[level_1, level_4]),
   ClothingRule(ActivityType.ACTIVE, ClothingRuleWeatherCondition(-1, 7, True),
[level_1, level_5]),
   ClothingRule(ActivityType.ACTIVE, ClothingRuleWeatherCondition(-13, -1, False),
               [level_1, level_2, level_5]),
   ClothingRule(ActivityType.ACTIVE, ClothingRuleWeatherCondition(None, -13, False),
[level_2, level_5, level_7])
]




class ClothingService:


   def get_cloth_by_weather(self, activity_type, temperature, wet):
       for rule in clothing_rules:
           if rule.is_suitable(activity_type, temperature, wet):
               return rule.clothing_levels
       return None

import requests
```

```
from exceptions import FetchingWeatherError


WEATHER_URL =
("https://api.openweathermap.org/data/2.5/weather?lat={}&lon={}&exclude=minutely,hour
ly&appid={"
              "}&units=metric&lang=ua")




class WeatherApi:
   def __init__(self, api_key, logger):
       self.api_key = api_key
       self.logger = logger


   def get_weather(self, latitude, longitude):
       url = WEATHER_URL.format(latitude, longitude, self.api_key)
       self.logger.info(f"Fetching weather data from {url}")
       response = requests.get(url)
       # print(response.json())
       if response.status_code != 200:
           raise FetchingWeatherError("Error fetching weather data: message = {}",
response.json())
       return response.json()

class FetchingWeatherError(Exception):
   pass

version: '3.3'
services:
 db:
   platform: linux/x86_64
   image: mysql:5.7
   restart: always
   environment:
     MYSQL_DATABASE: 'db'
     MYSQL_USER: 'user'
     MYSQL_PASSWORD: 'password'
     MYSQL_ROOT_PASSWORD: 'password'
   ports:
     - '3306:3306'
   expose:
     - '3306'
   volumes:
     - my-db:/var/lib/mysql


volumes:
 my-db:
```

index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Автоматизована система підбору одягу ECWCS</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f5f5f5;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #4CAF50;
            color: white;
            padding: 20px;
            text-align: center;
        }
        main {
            padding: 20px;
        }
        .container {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        .form-container, .history-container, .recommendations-container {
            background-color: white;
            padding: 20px;
            margin: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 80%;
        }
        .form-container form {
            display: flex;
            flex-direction: column;
        }
        .form-container label {
            margin: 10px 0 5px;
        }
        .form-container input, .form-container select {
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 4px;
            width: 100%;
        }
        .form-container button {
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
```

```
            margin-top: 20px;
        }
        .form-container button:hover {
            background-color: #45a049;
        }
        .history-container table, .recommendations-container table {
            width: 100%;
            border-collapse: collapse;
        }
        .history-container th, .history-container td, .recommendations-container th,
.recommendations-container td {
            border: 1px solid #ddd;
            padding: 10px;
            text-align: left;
        }
        .history-container th, .recommendations-container th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <header>
        <h1>Автоматизована система підбору одягу ECWCS</h1>
    </header>
    <main>
        <div class="container">
            <div class="form-container">
                <h2>Введіть дані для підбору одягу</h2>
                <form>
                    <label for="location">Геолокація:</label>
                    <input type="text" id="location" name="location"
placeholder="Введіть ваше місцезнаходження">

                    <label for="activity">Тип активності:</label>
                    <select id="activity" name="activity">
                        <option value="low">Низька</option>
                        <option value="medium">Середня</option>
                        <option value="high">Висока</option>
                    </select>

                    <label for="temperature">Температура (°C):</label>
                    <input type="number" id="temperature" name="temperature"
placeholder="Введіть температуру повітря">

                    <button type="submit">Підібрати одяг</button>
                </form>
            </div>

            <div class="recommendations-container">
                <h2>Рекомендований одяг</h2>
                <table>
                    <thead>
                        <tr>
                            <th>Рівень</th>
                            <th>Елемент одягу</th>
                            <th>Опис</th>
                        </tr>
```

```
                    </thead>
                    <tbody>
                        <tr>
                            <td>Базовий шар</td>
                            <td>Термобілизна</td>
                            <td>Відводить вологу від тіла, забезпечуючи сухість і
комфорт.</td>
                        </tr>
                        <tr>
                            <td>Теплоізоляційний шар</td>
                            <td>Флісова куртка</td>
                            <td>Забезпечує додаткове тепло і комфорт в холодних
умовах.</td>
                        </tr>
                        <tr>
                            <td>Зовнішній шар</td>
                            <td>Водонепроникна куртка</td>
                            <td>Захищає від дощу і вітру, зберігаючи теплоізоляційні
властивості внутрішніх шарів.</td>
                        </tr>
                    </tbody>
                </table>
            </div>

            <div class="history-container">
                <h2>Історія підбору одягу</h2>
                <table>
                    <thead>
                        <tr>
                            <th>Дата</th>
                            <th>Місцезнаходження</th>
                            <th>Температура</th>
                            <th>Активність</th>
                            <th>Рекомендації</th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr>
                            <td>2023-06-01</td>
                            <td>Київ</td>
                            <td>15°C</td>
                            <td>Середня</td>
                            <td>Термобілизна, Флісова куртка, Водонепроникна
куртка</td>
                        </tr>
                        <tr>
                            <td>2023-06-02</td>
                            <td>Львів</td>
                            <td>10°C</td>
                            <td>Низька</td>
                            <td>Термобілизна, Утеплені штани, Вітрозахисна
куртка</td>
                        </tr>
                        <tr>
                            <td>2023-06-03</td>
                            <td>Одеса</td>
                            <td>20°C</td>
```

```
                            <td>Висока</td>
                            <td>Легка футболка, Шорти, Кепка</td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </main>
</body>
</html>
```

app.js

```
document.addEventListener('DOMContentLoaded', () => {
    const form = document.querySelector('form');
    const recommendationsContainer = document.querySelector('.recommendations-
container tbody');
    const historyContainer = document.querySelector('.history-container tbody');

    form.addEventListener('submit', (event) => {
        event.preventDefault();
        const location = document.querySelector('#location').value;
        const activity = document.querySelector('#activity').value;
        const temperature = document.querySelector('#temperature').value;

        const recommendations = getRecommendations(temperature, activity);
        displayRecommendations(recommendations);
        saveHistory(location, temperature, activity, recommendations);
        displayHistory();
    });

    function getRecommendations(temperature, activity) {
        // Проста логіка для рекомендацій на основі температури та активності
        let recommendations = [];

        if (temperature <= 10) {
            recommendations.push({
                level: 'Базовий шар',
                item: 'Термобілизна',
                description: 'Відводить вологу від тіла, забезпечуючи сухість і
комфорт.'
            });
            recommendations.push({
                level: 'Теплоізоляційний шар',
                item: 'Флісова куртка',
                description: 'Забезпечує додаткове тепло і комфорт в холодних
умовах.'
            });
            recommendations.push({
                level: 'Зовнішній шар',
                item: 'Водонепроникна куртка',
                description: 'Захищає від дощу і вітру, зберігаючи теплоізоляційні
властивості внутрішніх шарів.'
            });
        } else if (temperature <= 20) {
            recommendations.push({
                level: 'Базовий шар',
```

```
            item: 'Легка футболка',
            description: 'Забезпечує комфорт у помірних умовах.'
        });
        recommendations.push({
            level: 'Зовнішній шар',
            item: 'Вітрозахисна куртка',
            description: 'Захищає від вітру, зберігаючи тепло.'
        });
    } else {
        recommendations.push({
            level: 'Базовий шар',
            item: 'Легка футболка',
            description: 'Забезпечує комфорт у теплих умовах.'
        });
        recommendations.push({
            level: 'Штани/шорти',
            item: 'Шорти',
            description: 'Забезпечують свободу рухів у теплих умовах.'
        });
        recommendations.push({
            level: 'Головний убір',
            item: 'Кепка',
            description: 'Захищає від сонця.'
        });
    }

    return recommendations;
}

function displayRecommendations(recommendations) {
    recommendationsContainer.innerHTML = '';
    recommendations.forEach(rec => {
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${rec.level}</td>
            <td>${rec.item}</td>
            <td>${rec.description}</td>
        `;
        recommendationsContainer.appendChild(row);
    });
}

function saveHistory(location, temperature, activity, recommendations) {
    const history = JSON.parse(localStorage.getItem('history')) || [];
    const newEntry = {
        date: new Date().toISOString().split('T')[0],
        location,
        temperature,
        activity,
        recommendations: recommendations.map(rec => rec.item).join(', ')
    };
    history.push(newEntry);
    localStorage.setItem('history', JSON.stringify(history));
}

function displayHistory() {
    const history = JSON.parse(localStorage.getItem('history')) || [];
```

```
        historyContainer.innerHTML = '';
        history.forEach(entry => {
            const row = document.createElement('tr');
            row.innerHTML = `
                <td>${entry.date}</td>
                <td>${entry.location}</td>
                <td>${entry.temperature}°C</td>
                <td>${entry.activity}</td>
                <td>${entry.recommendations}</td>
            `;
            historyContainer.appendChild(row);
        });
    }

    displayHistory();
});
```

Додаток Г – Ілюстративний матеріал

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНИХ ЗАСОБІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ПІДБОРУ ОДЯГУ

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Слайд «Актуальність розробки»

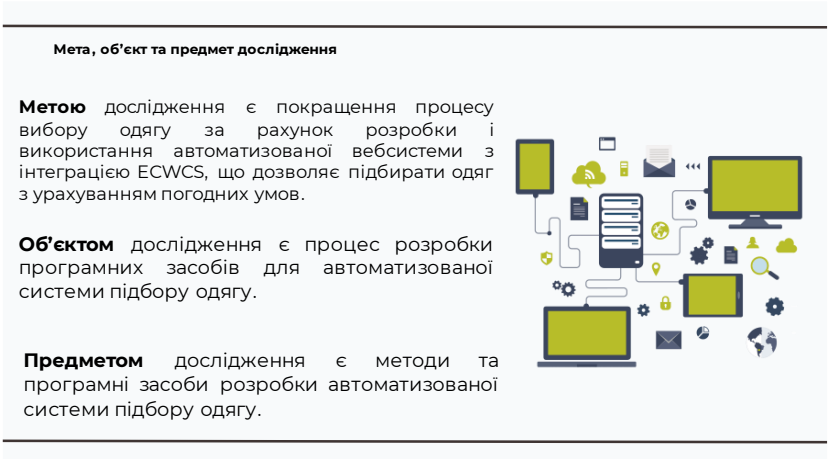

Рисунок Г.3 – Слайд «Мета, предмет та об'єкт дослідження»

Рисунок Г.4 – Слайд «Постановка задачі»

Рисунок Г.5 – Слайд «Новизна»

Рисунок Г.6 – Слайд «Практична цінність отриманих результатів»

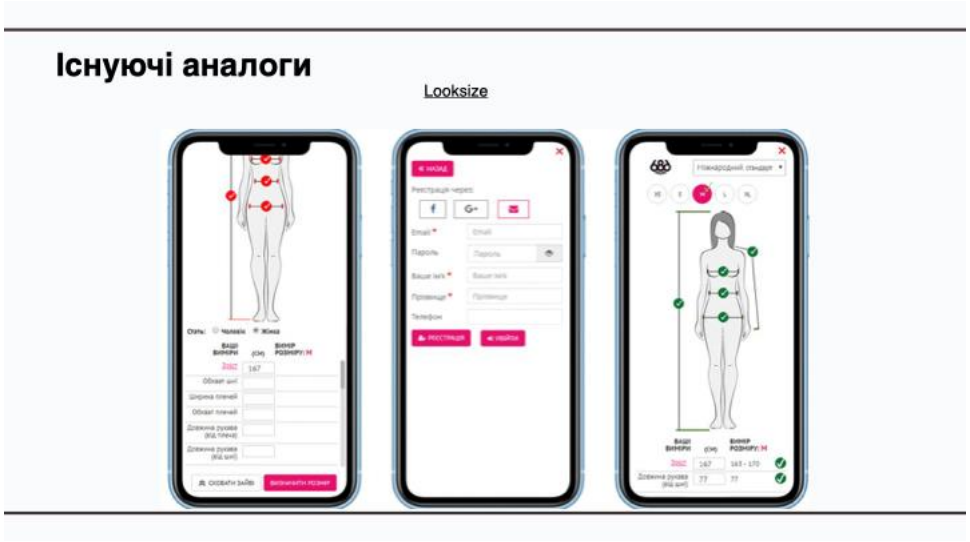

Рисунок Г.7 – Слайд «Існуючі аналоги»

Рисунок Г.8 – Слайд «Перший аналог»

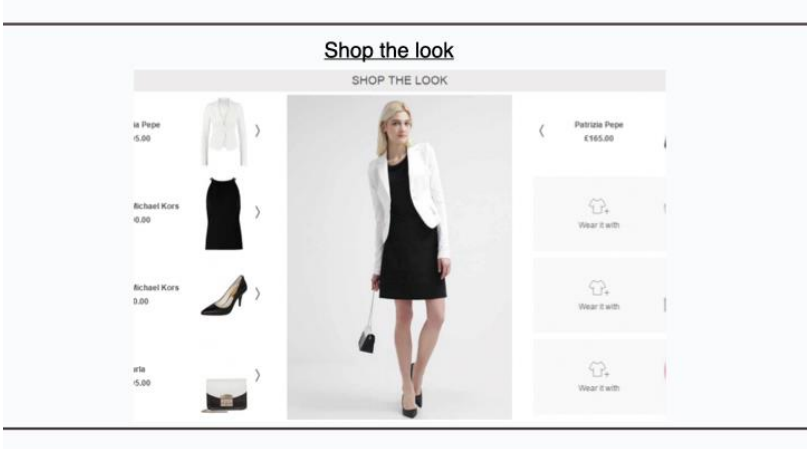

Рисунок Г.9 – Слайд «Другий аналог»

Аналіз аналогів

Критерії	Looksize	Rozetka	Shop the Look	Власна розробка
Підбір за системою ECWCS	0	0	0	1
Підбір одягу відповідно до погодних умов	0	0	0	1
Персоналізація даних	1	1	0	1
Автоматична система	0	0	1	1
Штучний інтелект	0	0	1	0
Сумарний коефіцієнт	**1**	**1**	**2**	**4**

Рисунок Г.10 – Слайд «Третій аналог»

Метод формування і відображення історії підбору одягу (ч. 1)

1. Після входу в особистий кабінет користувача доступний розділ "Історія підбору одягу".

2. Користувач натискає кнопку «Сформувати звіт» у секції профілю для ініціювання процесу формування звіту.

3. При натисканні кнопки запускається функція, яка відповідає за отримання даних з MySQL бази даних та їх підготовку для подальшого використання.

4. Метод getDataFromMySQL виконує з'єднання з базою даних MySQL.

5. За допомогою SQL-запитів до бази даних виконується запит для отримання user_id, який використовується як критерій для додаткових запитів до бази даних.

6. За протоколом HTTP з бази даних отримується user_id.

6.1. За критерієм user_id здійснюються запити до бази даних з метою отримання інформації, яка необхідна для заповнення решти полей звіту.

6.2. Відповідно до запитів, з бази даних отримуються дані у форматі JSON, які записуються у змінну user_data.

6.3. Метод getDataFromMySQL повертає об'єкт, що містить усю інформацію, яку запитувала система, зокрема: ім'я користувача, список рекомендованого одягу та погодні умови.

Рисунок Г.11 – Слайд «Метод формування і відображення історії підбору одягу(ч. 1)»

Метод формування і відображення історії підбору одягу (ч. 2)

7. Отримані дані передаються в метод formatDataForChart, який форматує дані для подальшого використання та повертає підготовлений масив formattedData.

7.1. Після отримання підготовленого масиву даних метод заповнює поля ідентифікації користувача: surname та name.

7.2. Після заповнення полей ідентифікації метод встановлює решту даних масиву у стан компонента data, який далі буде використано для графічного представлення даних.

7.2. Сформатовані дані передаються в компонент фреймворку ReactJS, який використовує бібліотеки react-charts та react-chartjs-2.

7.3. За допомогою бібліотек генеруються діаграми та гістограми, що відображають прогрес навчання користувача.

8. Після цього завершується формування звіту:

8.1. Отримується актуальна дата і встановлюється у поле звіту «Дата формування».

8.2. Встановлюється значення поля «report_number» типу int.

9. Користувачеві відображається сторінка із згенерованим звітом.

Рисунок Г.12 – Слайд «Метод формування і відображення історії підбору одягу(ч. 1)»

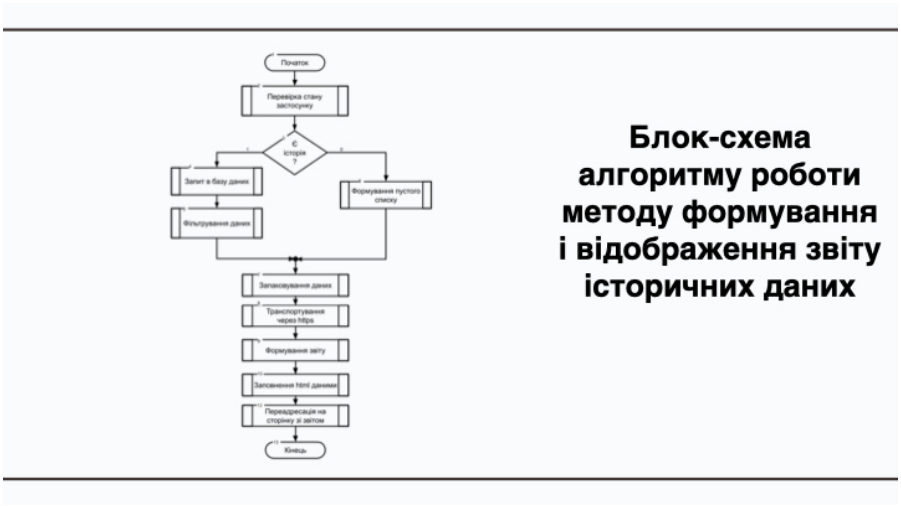

Рисунок Г.13 – Слайд « Блок-схема алгоритму роботи методу формування і відображення звіту історичних даних»

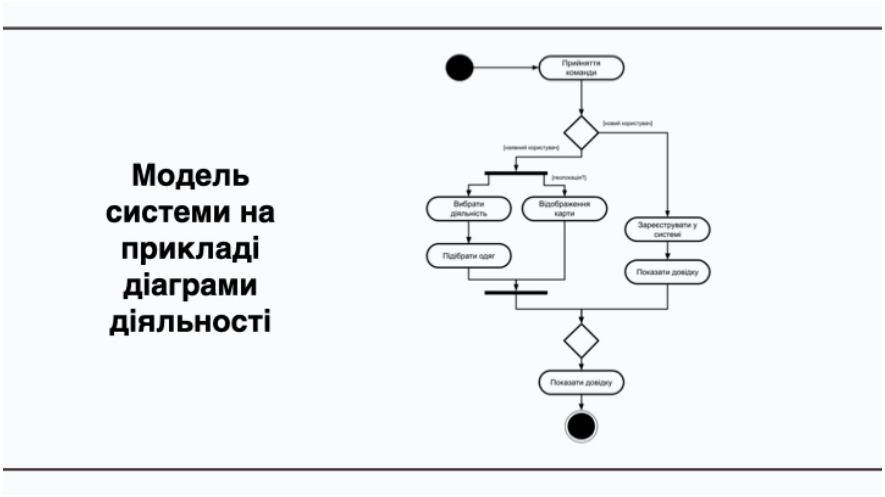

Рисунок Г.14 – Слайд « Модель системи на прикладі діаграми діяльності»

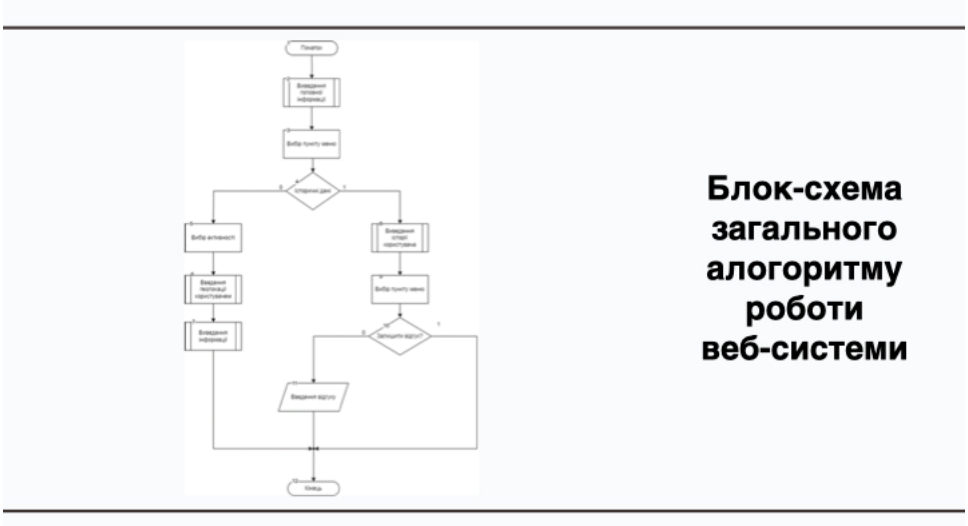

Рисунок Г.15 – Слайд « Блок-схема загального алогоритму роботи веб-системи»

Рисунок Г.16 – Слайд « Діаграми послідовності модуля керування обліковими записами»

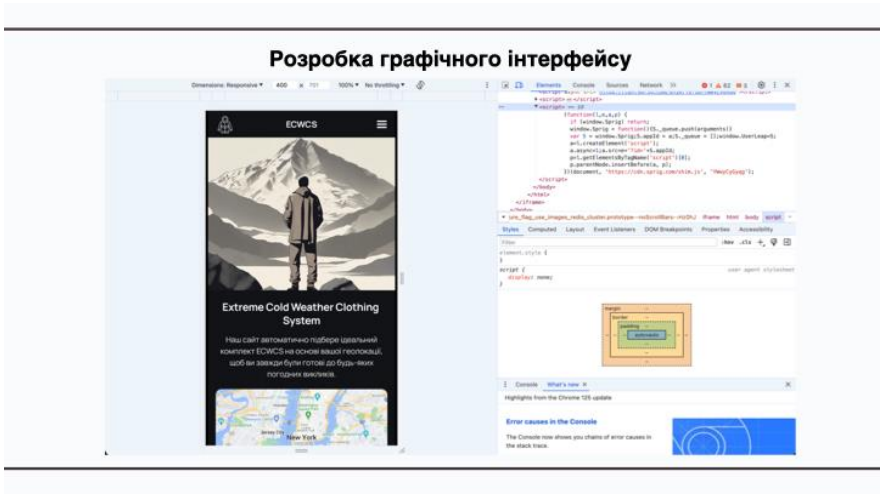

Рисунок Г.17 – Слайд « Розробка графічного інтерфейсу»

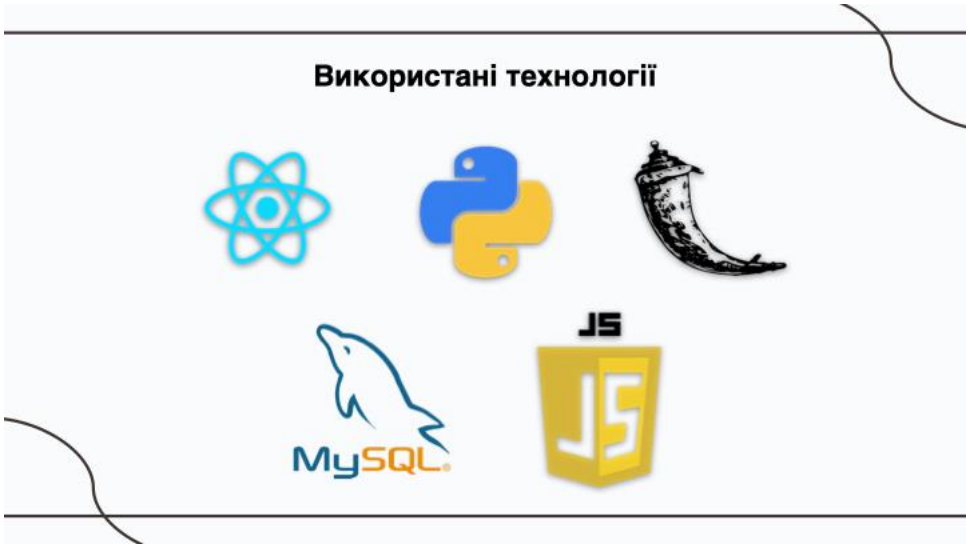

Рисунок Г.18 – Слайд « Використані технології»

Апробація та публікація результатів роботи

Матеріали бакалаврської дипломної роботи доповідалися на:

LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету, Вінниця, 2024.

За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференції.

Рисунок Г.19 – Слайд « Апробація та публікація результатів роботи»

Висновки

- розроблено програмні засоби реалізації системи автоматизованого підбору одягу з метою поліпшення процесу вибору одягу для користувачів в різних погодних умовах;
- реалізовано систему, що використовує сторонні API, які надають погодні дані та іншу необхідну інформацію;
- інтегровано сторонні API для отримання актуальних погодних даних за отриманою геолокацією;
- розроблено метод і модель роботи системи та алгоритм підбору одягу, який буде враховувати отримані погодні умови;
- забезпечено роботу зручного інтерфейсу користувача;
- проведено тестування та оптимізацію роботи розробленої системи.

Рисунок Г.20 – Слайд «Висновки»

Дякую за увагу!

Рисунок Г.21 – Слайд «Дякую за увагу!»