Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)»

Виконав: студент II курсу
групи 2ПІ-22м спеціальності
121 - Інженерія програмного забезпечення
Пінчуков О. М.
(прізвище та ініціали)

Керівник: д.т.н, професор каф. ПЗ
Ліщинська Л. Б.
(прізвище та ініціали)
«08» грудня 2023 р.

Опонент: к.т.н, доцент каф. ЗІ
Дудатьєв А. В.
«08» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«08» грудня 2023 р.

Вінниця ВНТУ – 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
_____ Романюк О. Н.
« 19 » вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пінчукову Олександру Миколайовичу

1. Тема роботи - Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19).

Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., професор кафедри ПЗ, затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.

2. Строк подання студентом роботи _____5 грудня 2023р._____

3. Вихідні дані до роботи: операційна система macOS; середовище розробки - Visual Studio Code, формат зберігання конфігурації - JSON; мова програмування - JavaScript; фреймворк для розробки мобільних додатків React Native; симулятори мобільних пристроїв XCode та Android Studio; транспілятор Babel; інструмент статичного аналізу коду ESLint; CASE засіб Visual Paradigm; збірник модулів Webpack; розподілена система управління версіями - Git.

4. Зміст текстової частини: вступ; аналіз методів і засобів відстеження та інформування про глобальні пандемії, моделювання відстеження та інформування про глобальні пандемії та проектування програмної системи, розробка програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19), тестування програмного засобу, економічна частина, висновки, список використаних джерел, додатки;

5. Перелік ілюстративного матеріалу: UML діаграми архітектури додатку; графічний інтерфейс додатку; результати тестування додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ліщинська Л. Б., д.т.н., проф. кафедри ПЗ	19.09.23	05.12.23
5	Кавецький В. В., к.е.н., доцент кафедри ЕПВМ	11.11.23	24.11.23

7. Дата видачі завдання 19 вересня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів і засобів відстеження та інформування про глобальні пандемії	19.09.2023 - 25.09.2023	вик
2	Моделювання відстеження та інформування про глобальні пандемії та проектування програмної системи	26.09.2023 - 05.10.2023	вик
3	Розробка програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)	06.10.2023 - 02.11.2023	вик
4	Тестування програмного засобу	03.11.2023 - 10.11.2023	вик
5.	Економічна частина	11.11.2023 - 24.11.2023	вик

Студент _____ **Пінчуков О. М.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ліщинська Л.Б.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Пінчуков О. М. Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19) : магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 101 с.

На укр. мові. Бібліогр. : 51 назв ; рис. : 28; табл. 24.

У магістерській кваліфікаційній роботі проведено аналіз методів і засобів відстеження та інформування про глобальні пандемії на прикладі Covid-19.

Було здійснено аналіз галузі застосування, призначення і вимог до створення засобу відстеження та інформування про глобальні пандемії.

Подальшого розвитку отримали методи візуалізації та сегментації даних відстеження пандемії.

Виконаний варіантний аналіз та обґрунтування вибору способу реалізації програмного засобу.

Здійснено проектування програмного засобу, під час якого використано інструментальне програмне забезпечення CASE засіб Visual Paradigm, за допомогою якого було побудовано UML діаграми, а також створено таблиці. Розроблено кросплатформенний мобільний додаток для iOS та Android з використанням мови JavaScript, фреймворку React Native та середовища розробки Visual Studio Code.

Отримані в магістерській кваліфікаційній роботі результати можна використати для відстеження та інформування про глобальні пандемії.

Ключові слова: пандемія, Covid-19, цифрова медицина, мобільний додаток, відстеження та інформування.

ABSTRACT

Pinchukov O. M. Development of a method and software tool for tracking and informing about global pandemics (Covid-19). Vinnytsia: VNTU, 2023. 101 p.

In Ukrainian language. Bibliographer: 51 titles ; fig. : 28; tabl. 24.

In the master's qualification work, a detailed analysis of methods and means of tracking and informing about global pandemics was carried out using the example of Covid-19.

An analysis of the field of application, purpose and requirements for the creation of a tool for tracking and informing about global pandemics was carried out.

The methods of visualization and segmentation of pandemic tracking data received further development.

Variant analysis and justification of the choice of the method of implementation of the software tool was performed.

The design of the software tool was carried out, during which the Visual Paradigm CASE tool software was used, with the help of which UML diagrams were built, as well as tables were created. Cross-platform mobile application for iOS and Android was developed with usage of JavaScript programming language, React Native framework and Visual Studio Code development environment.

The results obtained in the master's qualification work can be used for tracking and informing about global pandemics.

Keywords: pandemic, Covid-19, digital medicine, mobile application, tracking and informing.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Пандемії, як показав приклад Covid-19, викликають серйозне занепокоєння через їх широкомасштабний і руйнівний вплив на здоров'я людей, соціальні структури та економіку. Вони перевантажують системи охорони здоров'я, що ускладнює отримання необхідної допомоги людям як у зв'язку з пандемією, так і при інших захворюваннях.

Існує багато невизначеностей щодо оптимальних шляхів запобігання поширенню захворювання та найбільш ефективних методів лікування.

Відстеження може допомогти нам зрозуміти як поширюється вірус і виявити місця з високим ризиком, а також прийняти потрібні заходи у відповідь. Це має велике значення для ефективності системи охорони здоров'я.

Крім того, результати відстеження пандемії є надзвичайно важливими для наукових досліджень. Це може допомогти науковцям краще зрозуміти поведінку самого вірусу і розробити ефективні методи лікування та профілактики.

Безумовно, пандемії створюють серйозні проблеми, але за допомогою точного відстеження та інформування про них ми можемо працювати над ефективними стратегіями їх подолання.

Проте наявні засоби відстеження та інформування про пандемії мають суттєво обмежений функціонал, а також дуже низьку аналітичну результативність.

В зв'язку з вищезазначеним, можна стверджувати, що актуальність теми "Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)" є надзвичайно високою. Відстеження та інформування грають ключову роль у глобальній боротьбі з пандемією, оскільки допомагають розробити стратегії для пом'якшення її наслідків на здоров'я, соціальний і економічний розвиток.

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 підвищення ефективності оперативного відстежування розвитку пандемій, таких як Covid-19.

Відповідно до поставленої мети основними задачами дослідження є:

- провести аналіз ролі, призначення, сфери застосування системи відстежування та інформування про пандемії;
- провести аналіз існуючих аналогів та їх недоліків;
- покращення методів відстеження та інформування про глобальні пандемії;
- здійснити архітектурне проектування системи;
- безпосередня розробка логіки програмної системи;
- здійснити розробку сучасного адаптивного графічного інтерфейсу додатку;
- проведення тестування програмного засобу для прояву невідповідностей із запланованою функціональністю.

Об’єкт дослідження - процеси відстеження поширюваності та інформування про глобальні пандемії.

Предмет дослідження - методи та засоби процесів відстеження та інформування про глобальні пандемії.

Методи дослідження. У процесі дослідження та для досягнення поставлених завдань використовувались такі методи дослідження, зокрема:

- метод ядрової оцінки густини розподілу був використаний для здійснення геоспросторового аналізу та візуалізації щільності подій захворювання;
- кластеризація методом k-середніх була використана для визначення і розподілу на підгрупи країн з схожими профілями пандемії;
- методи проектування логіки програмних систем були використані при архітектурному проектуванні програмної системи;
- методи розробки програмних систем були використані для розробки модулів програмної системи;

– методи тестування програмного забезпечення були використані для перевірки функціоналу програмної системи.

Ці методи дослідження дозволять провести аналіз, отримати відповіді на поставлені питання та зробити обґрунтовані рекомендації щодо проектування та розробки додатку відстеження захворюваності Covid-19.

Наукова новизна отриманих результатів.

– Подальшого розвитку отримали методи візуалізації даних відстеження пандемії, шляхом застосування ядрової оцінки густини розподілу (англ. Kernel density estimation), що надало можливість оцінити базовий розподіл (щільність) набору точок даних і дозволило здійснити геопросторовий аналіз та візуалізувати щільність подій захворювання у географічній області.

– Подальшого розвитку отримали методи сегментації даних відстеження пандемії, шляхом застосування кластеризації методом k-середніх (англ. k-means clustering), що надало можливість розділити набір даних на окремі непересічні підгрупи, де кожна точка даних належить до кластера з найближчим центроїдом і дозволило ефективно визначити країни зі схожими профілями пандемії.

Практична цінність отриманих результатів. Полягає у тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень запропоновано програмний засіб для відстеження та інформування про глобальні пандемії (Covid-19).

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. Автору належать такі результати: аналіз ролі програмних засобів для відстеження та інформування про пандемії та їх значення для системи охорони здоров’я [4]; аналіз можливостей застосування програмних засобів для відстеження та інформування про пандемії [8]; програмний засіб для відстеження та інформування про поширення пандемії Covid-19.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення й результати магістерської кваліфікаційної роботи представлені на

науково-практичній конференції «Електронні інформаційні ресурси: створення, використання, доступ-2023» (Вінниця, 2023).

Публікації. Основні результати досліджень опубліковано в 2 наукових працях представлених у матеріалах конференції «Електронні інформаційні ресурси: створення, використання, доступ-2023» (Вінниця, 2023).

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу; п’яти розділів; списку використаних джерел, що містить 51 найменувань, 28 ілюстрацій, 24 таблиці, 4 додатки.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ВІДСТЕЖЕННЯ ТА ІНФОРМУВАННЯ ПРО ГЛОБАЛЬНІ ПАНДЕМІЇ

1.1 Аналіз ролі та призначення програмного засобу

Пандемія Covid-19 проявила значну вразливість глобальної системи охорони здоров’я. Вона призвела до критичного перевантаження медичних систем багатьох країн світу. Використання сучасних можливостей програмного забезпечення може стати надзвичайно важливим інструментом, який допоможе покращити розуміння поширення захворювання. Це дозволить швидко реагувати та здійснювати необхідні заходи для пом’якшення широкомасштабних негативних наслідків.

На сучасному рівні цифрового розвитку суспільства, програмні засоби надають безпрецедентну можливість не тільки збирати великі обсяги даних але і швидко їх аналізувати. Вони можуть інтегрувати дані з різних джерел, таких як: медичні установи, лабораторні центри, лікарні тощо. Це у свою чергу дає можливість отримувати актуальне уявлення про розповсюдження захворювання в режимі реального часу.

Технології відіграють центральну роль у багатьох, якщо не у всіх аспектах пандемії Covid-19 [1]. Використовуючи сучасні технології цілком можливо створити програмну систему, яка постійно оновлюється та надає актуальну інформацію та зручну візуалізацію поширення захворювання.

Використання таких систем для відстеження має низку переваг:

– надання актуальної інформації в режимі реального часу, що дає можливість мати доступ до найсвіжіших даних;

– програмний аналіз поширення захворювання дозволяє оперативно реагувати та запобігати утворенню більш значних зон зараження; крім того, глибше вивчення отриманих даних може пояснити чому певні території стають "гарячими точками" захворювання та які фактори впливають на це;

– завдяки більш чіткій і точній інформації про швидкість поширення хвороби та загальну епідемічну ситуацію, посадові особи можуть приймати

ефективніші управлінські рішення.

Пандемія викликала значну потребу в точних і своєчасних епідеміологічних даних з різних тем, щоб зрозуміти вплив та спланувати адекватну відповідь [2]. Тому значення програмних засобів для відстеження та інформування про пандемії для системи охорони здоров'я також важко переоцінити. Серед вагомих переваг їх використання в цій сфері є:

– за допомогою точного відстеження розповсюдження вірусу, адміністративні заходи можна суттєво локалізувати; замість широкомасштабних локдаунів можна буде спрямовувати максимальні зусилля на зони з більшою швидкістю розповсюдження хвороби, що допоможе зменшити економічні наслідки;

– закладам охорони здоров'я буде корисно мати доступ до актуальних даних про поширення хвороби у районі їх відповідальності; це дозволить бути більш підготовленими та набагато ефективніше розподіляти ресурси;

– доступ до актуальних даних може бути цінним інструментом для попередження громадськості про поточну ситуації та наголошення про дотримання рекомендованих заходів безпеки.

Однак, використання програмних засобів у цій сфері потенційно може мати ряд серйозних викликів, зокрема:

– точність вихідних даних залежить від надійності та точності джерела інформації; дезінформація або недостатня точність можуть спотворити результат;

– щоб будь-який програмний інструмент був ефективним, необхідне широке сприйняття його спільнотою; тому важливим фактором є подолання скептицизму та опору громадськості;

– віруси постійно мутують та розвиваються, програмні засоби, створені для відстеження та інформування про пандемії, повинні бути здатні адаптуватися до нових штамів вірусу, змін у поширені або передачі вірусу; тому програмна система має мати таку ж гнучкість, як і біологічний виклик, що перед нею стоїть.

Крім того, швидкість ідентифікації або діагностики та оцінка клінічних даних є однією з основних перешкод для сучасних систем. Існує величезний простір для розробки цифрових платформ, у яких висока швидкість і комунікаційний бар'єр між обміном інформацією будуть більш ефективними та плідними [3].

Важливо додати, що оскільки пандемії не мають географічних кордонів, ефективність програмних засобів для відстеження та інформування значним чином також залежить від міжнародної співпраці. Щоб ці інструменти максимально розкрили свій потенціал, урядам різних країн вкрай необхідно співпрацювати у повній взаємодії [4].

1.2 Аналіз сфери застосування програмного засобу

Приклад Covid-19 показав, що спалах пандемій створює не лише серйозний виклик системі охорони здоров'я але й тягне за собою інші соціальні, економічні, гуманітарні негативні наслідки. Крім безсумнівної користі в галузі охорони здоров'я, ці програмні інструменти також можуть бути потенційно корисні в наукових дослідженнях, бізнесі, туризмі та інших галузях.

Безперечно, основне використання засобів відстеження та інформування про пандемії полягає в їх застосуванні у галузі охорони здоров'я. Моніторинг є ключовим компонентом для стримування відомого інфекційного агента, а також виявлення невідомого або неочікуваного спалаху хвороби [5]. Ці програмні інструменти дозволяють збирати дані у режимі реального часу та аналізувати поширення хвороби, що допомагає:

– прогнозувати спалахи захворювання шляхом аналізу історичних та поточних даних; таким чином можна передбачити потенційні спалахи та прийняти своєчасні заходи для запобігання поширенню захворювання;

– максимально ефективно визначати пріоритети та розподіляти медичні ресурси;

– оперативніше надавати громадськості інформацію про можливі загрози та профілактичні заходи.

Наукова спільнота також може отримати значну користь від цих інструментів. Аналіз отриманих даних відстеження дозволяє:

– доповнювати зусилля у розробці вакцин та оцінювати ефективність кампаній щодо їх впровадження шляхом виявлення закономірностей прояву та поширення захворювання;

– глибше зрозуміти поведінку вірусу та механізми його передачі шляхом вивчення закономірностей його поширення.

Після спалаху пандемії Covid-19, через порушення ланцюгів постачання виробництво на підприємствах було значно загальмовано. Однак попит зазнав ще більшого удару, так як споживчі витрати різко впали, а інвестиції зупинились [6]. У сучасній глобалізованій економіці для підприємств важливо мати засоби відстеження та інформування про пандемії, оскільки це надає значну кількість даних, які можуть вплинути на деякі аспекти бізнесу, зокрема:

– ефективніше планування діяльності підприємства на випадок посилення поширюваності захворювання; завдяки даним відстеження пандемії, бізнес може передбачити потенційно небезпечні “гарячі точки” та підготуватись до збоїв у ланцюгах постачання або проблем з залученням робочої сили;

– зміна інвестиційних рішень з урахуванням економічних наслідків пандемії;

– більш точна та адекватна оцінка ризикових факторів; компанії можуть проаналізувати розповсюдження заражень у певних регіонах та пов’язані з цим ризики, що дасть змогу розробити стратегію подальших дій;

– гнучкість кадрової політики, включаючи впровадження дистанційної роботи та закриття офісних приміщень.

Туристична галузь є однією з тих, що найбільше постраждали від пандемії. Ця галузь завжди відчуває найсильніші удари від різних захворювань, епідемій, сезонного грипу та пандемій [7]. У цій сфері інструменти для відстеження пандемій можуть допомогти:

– на основі отриманих даних авіакомпанії та туроператори можуть надавати туристам відповідні поради та рекомендації щодо подорожей;

– гарантувати безпеку планування подорожей для туристів, враховуючи епідеміологічну ситуацію у місцях призначення.

Також ці засоби є надзвичайно корисним ресурсом для широкої громадськості, оскільки вони дозволяють:

– підвищити особисту безпеку, допомагаючи людям уникати зон високого ризику та більш зважено приймати рішення стосовно заходів особистого захисту та участі у масових зібраннях або соціальних зустрічах;

– отримувати достовірну й актуальну інформацію, протидіючи цим поширенню дезінформації, яка може поширюватися під час надзвичайних ситуацій.

Таким чином ці програмні інструменти можуть надати широкі можливості для контролю за захворюванням та його поширенням у різних сферах. Вони забезпечують надання необхідної інформації для влади, бізнесу та громадськості для більш ефективного вирішення можливих кризових ситуацій [8].

1.3 Порівняльний аналіз аналогів

Порівняльний аналіз проводився серед веб-додатків та мобільних додатків. Аналіз маркетів мобільних додатків, таких як Google Play [9] та Apple Store [10], показав що на сьогодні відсутнє суто мобільне рішення у формі додатку на смартфон для відстеження та інформування про пандемії.

Натомість аналіз Web-додатків показав наявність декількох альтернатив. Були розглянуті найпопулярніші з них: web-додаток від Всесвітньої організації здоров’я [11], а також дві приватні ініціативи - covidvisualizer.com [12] та coronavirus.app [13].

Після аналізу аналогів, були визначені їх можливості та недоліки. В подальшому це було враховано при розробці власного програмного забезпечення.

Для детального порівняння була складена аналітична таблиця (таблиця 1.1).

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	covidvisualizer.com	coronavirus.app	WHO	Власна розробка
Інтерактивна мапа поширення Covid-19	+	+	+	+
Покращена аналітична складова додатку (KDE, K-means)	-	-	-	+
Фільтрація даних	-+	+	+	+
Новини про Covid-19	-	-	+	+
Мобільний версія (додаток)	-	-	-	+
Пуш-повідомлення (нагадування) на пристрій	-	-	-	+
Сума	1,5	2	3	6

З порівняльної таблиці випливає, що створення власної розробки вирішує проблеми наявних рішень, тому розробка програмного засобу є доцільною.

1.4 Постановка задач дослідження

Задачею дослідження є створення засобу для відстеження та інформування про глобальні пандемії, базуючись на прикладі Covid-19. Інструмент має за мету забезпечити можливість в реальному часі відстежувати та візуально

представляти дані, пов’язані з пандемією, користуючись надійними джерелами даних. Основні компоненти засобу повинні включати пошук, обробку, фільтрацію даних, зручний і доступний інтерфейс для графічного представлення даних (включно з картами і таблицями).

Функціональні вимоги:

– засіб має підключатися до надійного офіційного джерела даних;

– засіб повинен мати можливість очищати, форматувати та регулярно оновлювати дані, щоб забезпечити їх актуальність;

– засіб має представляти дані у зручній формі, маючи такі функції, як: карти, таблиці, а також параметри фільтрації;

– засіб повинен мати функції авторизації та реєстрації;

– засіб повинен мати функцію надання сповіщень користувачам;

Ці функціональні вимоги визначають конкретну поведінку системи, описуючи яким чином система має діяти у різних ситуаціях. Вони встановлюють чіткі очікування щодо того, як користувачі взаємодіють з системою і також слугують основою для перевірки правильності роботи системи під час тестування.

Нефункціональні вимоги:

– засіб повинен швидко завантажуватися та оновлюватися не споживаючи надмірних системних ресурсів;

– засіб має застосувати суворі заходи безпеки, щоб захистити дані та забезпечити конфіденційність користувачів;

– засіб має бути надійним, з мінімальними простоями або помилками; користувачі повинні бути впевнені, що інформація, яку вони бачать, є точною та актуальною;

– програма має бути простою у використанні, мати інтуїтивно зрозумілу навігацію, а також чітке та зрозуміле представлення інформації;

– засіб повинен мати адаптивний дизайн, тобто він повинен забезпечувати хорошу взаємодію з користувачем на різних пристроях і розмірах екрана;

– засіб повинен бути кросплатформенним, тобто працювати як на Android, так і на iOS без додаткової індивідуальної доробки;

Зазначені нефункціональні вимоги окреслюють такі властивості системи, як: безпека, надійність, продуктивність, зручність використання. Вони визначають очікувану поведінку системи та відіграють ключове значення для забезпечення високоякісного досвіду використання для користувачів програмного засобу.

1.5 Висновки

В першому розділі був здійснений поглиблений аналіз ролі та призначення програмних засобів для відстеження та інформування про пандемії. Також визначено переваги та окреслено ряд викликів, які потенційно можуть виникнути при їх застосуванні.

Вивчена сфера застосування цих програмних засобів. Були проаналізовані інші сфери, окрім охорони здоров’я, де вони можуть бути корисними.

Розглянуто існуючі аналоги та здійснено порівняльний аналіз.

Здійснено постановку задач дослідження та визначено вимоги до програмного засобу для відстеження та інформування про пандемії.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ВІДСТЕЖЕННЯ ТА ІНФОРМУВАННЯ ПРО ГЛОБАЛЬНІ ПАНДЕМІЇ ТА ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Покращення методів візуалізації даних відстеження пандемії

Візуалізація даних має велике значення для сучасного аналізу даних. Вона включає різні методи, які перетворюють складні набори даних на зрозумілі та детальні графічні зображення. Класичні методи візуалізації даних, такі як стовпчасті діаграми, лінійні графіки, кругові і точкові діаграми є достатньо базовими і звичними методами візуалізації даних. Кожен з них пропонує свої унікальні можливості для представлення та аналізу даних.

Наприклад, стовпчасті діаграми є важливим інструментом для порівняння даних у різних категоріях. Вони надають чіткий і простий спосіб візуалізації різниці у значеннях. Це чудовий спосіб показати різницю у кількості випадків Covid-19 або розподіл вакцинації у різних регіонах.

З іншого боку, лінійні графіки дуже добре відображають зміни, які відбуваються з часом. Цей метод особливо корисний для відстеження того, як змінюється ситуація з кількістю нових випадків захворювання, одужань і смертей Covid-19. Це дозволяє побачити динаміку пандемії.

Кругові діаграми є ще одним популярним інструментом для візуалізації даних. Вони добре показують структуру та співвідношення між частинами цілого. В контексті Covid-19 кругові діаграми можуть бути корисними для представлення наочної інформації про розподіл випадків серед різних демографічних груп, наприклад за віком або статтю.

Важливо, що хороша візуалізація даних включає характеристики читабельності, впізнаваності та зрозумілості [14].

Природа пандемій є дуже складною і має багато невизначених факторів, а дані постійно швидко змінюються. Це вимагає багатогранного підходу до представлення та аналізу даних. Хоча класичні методи візуалізації відіграють важливу роль, доцільним є застосовування більш прогресивних методів аналізу

та представлення складних масивів даних для більш ефективного відстеження пандемій.

Використання методу ядрової оцінки густини розподілу (англ. Kernel density estimation) для створення теплової карти розповсюдження захворювання може стати інноваційним інструментом для візуалізації складних наборів даних, що може бути особливо корисним в геопросторовому аналізі відстеження пандемій.

Ядрова оцінка густини розподілу (KDE), також відома як вікно Парзена (Parzen, 1962) є одним із найвідоміших підходів для оцінки ймовірності густини розподілу набору даних. KDE - це непараметричний оцінювач густини розподілу, який не потребує припущення, що базова функція густини належить до параметричного сімейства. KDE визначає форму густини розподілу з даних автоматично. Ця гнучкість, що виникає через його непараметричну природу, робить KDE дуже популярним підходом для даних отриманих із складного розподілу [15]. Метод ядрової оцінки густини розподілу представлений у формулі (2.1):

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-X_i}{h}\right) \tag{2.1}$$

де $\hat{f}(x)$ - оцінка густини розподілу в точці x; це значення, яке ми отримуємо після виконання всієї операції;

n - кількість точок даних у наборі даних які використовуються для оцінки;

K - ядерна функція, яка визначає форму "гладкості" оцінки; функція приймає відстань між оцінюваною точкою x та кожною точкою даних X_i, поділену на ширину смуги h;

h - ширина смуги, що визначає наскільки "гладкою" буде оцінка густини розподілу.

Застосування методу ядрової оцінки густини розподілу для візуалізації даних, зокрема в контексті даних про Covid-19, пропонує значне покращення в

підходах до відстеження та інформування про глобальні пандемії.

Традиційні карти відстеження пандемії зазвичай представляють дані в звичайних базових одиницях, як-от кількість випадків на конкретний регіон, як це реалізовано в розглянутих аналогах до власної розробки. Однак теплові карти які можуть бути створені з використанням KDE виходять за межі цього. Вони надають більш деталізований погляд на поширення пандемії, а також її наслідків, в залежності від вибраних критеріїв дослідження. Вони пропонують детальну оцінку щільності захворювання, яка є особливо ефективною для виявлення гарячих точок і розуміння просторового розподілу випадків. Завдяки цьому досягається детальне геопросторове уявлення про поширення захворювання. Проте, ця просторова чіткість стосується не лише поліпшення візуалізації даних, а й також покращення інтерпретації та доступності складної інформації для різноманітної аудиторії, включаючи наукове середовище.

Інновація використання теплових карт Covid-19 з використанням методу KDE полягає в їх здатності покращувати просторовий аналіз таким способом, якого не можуть забезпечити традиційні карти відстеження. Вони пропонують високий рівень деталізації та ясності, що має вирішальне значення для осіб, які приймають управлінські державні рішення та медичних працівників у розробці стратегії реагування системи охорони здоров’я. За допомогою використання градієнтів щільності даних, теплові карти KDE дозволяють точніше ідентифікувати області, які потребують більш цілеспрямованого втручання. Крім того, здатність оновлюватися новими даними в режимі реального часу робить їх невід’ємними інструментами у боротьбі з пандемією, яка має особливість швидко змінюватись.

У власній розробці метод ядрової оцінки густини розподілу був застосований у модулі "Карти" для створення теплової карти відстеження поширення пандемії. Як вже зазначалось, для аналізу можуть бути вибрані різні види даних, як-от дані про кількість смертей на регіон, нових випадків на регіон або, навпаки, кількість одужавших пацієнтів. Це повністю залежить від критеріїв та мети відстеження.

Наприклад, для наочності представлення реалізації візуалізації методу ядрової оцінки густини розподілу у власній розробці продемонстровано теплову карту, яка використовує дані щодо кількості випадків на мільйон осіб у різних штатах США (рис. 2.1).

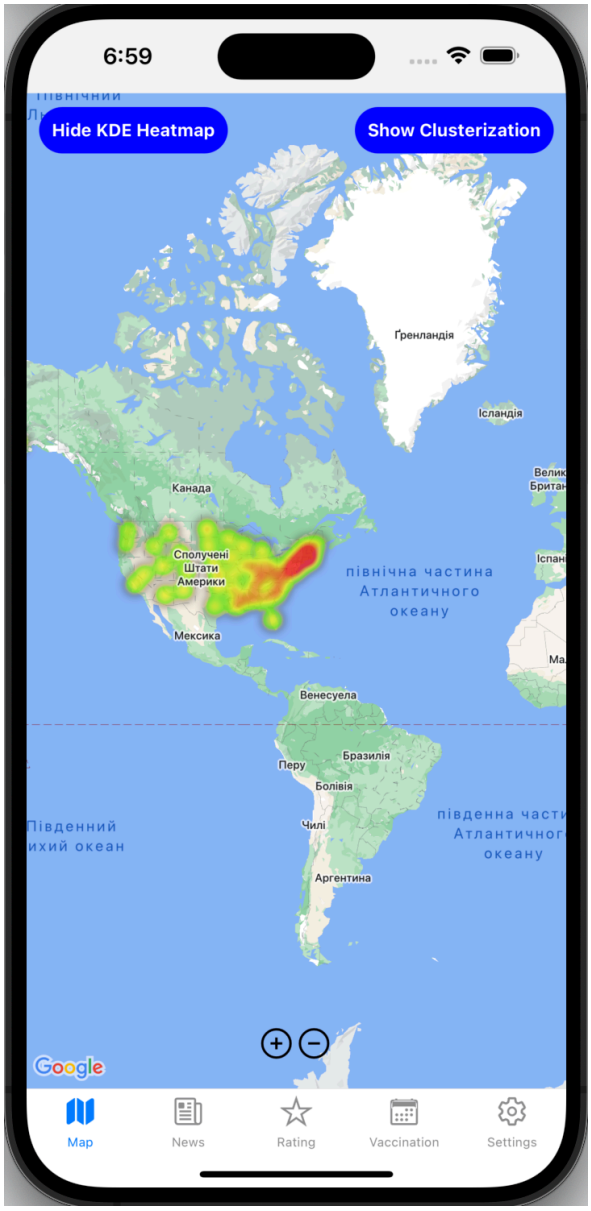

Рисунок 2.1 – Реалізація візуалізації методу ядрової оцінки густини розподілу

Як можна явно побачити з отриманої теплової карти з використанням методу KDE, кількість випадків різко збільшена у східній частині країни. Ця інформація є надзвичайно важливою для проведення подальшого аналізу причин такої поведінки вірусу. Можливо система охорони здоров’я у східних штатах має

певні відмінності у протоколах боротьби з пандемією, інше фінансування тощо. Тому застосований метод візуалізації відкриває нові можливості для відстеження та інформування про пандемії, а класичні методи, про які було згадано вище, дещо програють у якості зворотнього зв'язку для здійснення подальшого аналізу.

Підсумовуючи, варто зазначити, що застосування теплових карт з використанням методу KDE є значним прогресом у методах візуалізації даних відстеження пандемії. Використання цього методу візуалізації разом з іншими допомагає нам краще розуміти пандемію та сприяє ефективнішому аналізу великого масиву даних захворювання.

2.2 Покращення методів сегментації даних відстеження пандемії

У сфері сегментації даних існує широкий спектр простих методів, які часто використовуються для базового аналізу даних. Такі методи включають просту фільтрацію, категоризацію, бінарну класифікацію та інші. Вони корисні завдяки своїй простоті та легкості в застосуванні, особливо коли потрібно швидко оцінити наявні дані.

Одним із основних методів сегментації даних є фільтрація. Вона використовує заздалегідь визначені критерії для розподілу даних на групи, наприклад, такі як: кількість випадків Covid-19, кількість смертей або одужань. Цей метод дозволяє швидко розділити дані за такими сегментами. Переваги фільтрації полягають у її простоті та легкості застосування, не потребуючи складних статистичних знань.

Класифікація також є простим способом сегментації даних, що передбачає розподіл даних на групи залежно від конкретних характеристик. Наприклад, у випадку Covid-19 дані можна розподілити за віковими групами або географічними регіонами. Цей підхід забезпечує структуровану форму даних, що полегшує порівняння та аналіз різних категорій. Це особливо корисно для виявлення закономірностей або тенденцій у відповідних групах, наприклад для моніторингу поширення вірусу серед демографічних груп.

Бінарна класифікація являє собою розділення даних на дві групи на основі

двійкового критерію. У контексті Covid-19 це може означати поділ даних пацієнтів на «інфікованих» і «неінфікованих» на підставі результатів тестування. Цей метод особливо корисний для швидкої початкової оцінки даних, що буде основою для подальшого детального аналізу.

Безумовно, зазначені базові методи сегментації даних відіграють важливу роль у аналізі даних. Особлива перевага цих методів полягає в їх простоті використання та здатності швидко надавати чітку зрозумілу інформацію для аналізу. Проте вони мають свої обмеження, особливо коли мова йде про дослідження складних залежностей між даними. Цим методам бракує потужних аналітичних можливостей більш просунутих методів. Вони не беруть до уваги закономірності між різними змінними у даних, а лише сегментують дані на основі окремих, заздалегідь визначених критеріїв.

Саме через складну природу пандемії і потребу в розумінні закономірностей у складних наборах даних, застосування більш просунутих методів, таких як кластеризація k-середніх, вбачається доцільним.

Метод k-середніх є, мабуть, найбільш широко використовуваним методом кластеризації і є особливо відомим серед методів кластеризації на основі поділу, які використовують центроїди для представлення кластера [16].

Цей метод розподіляє кожну точку даних у кластери (кількість кластерів повинна бути заздалегідь відома), що не пересікаються. Однією із основних і найбільш складних задач методу k-середніх є визначення початкових оптимальних центроїдів кластерів.

Метод кластеризації k-середніх має дві фази ітерації, а саме: фазу призначення або ініціалізації, яка включає ітераційний процес, де кожна точка даних призначається її найближчому центроїду за допомогою евклідової метрики; наступним є етап оновлення центроїда, де центроїди кластерів оновлюються відповідно до поділу, отриманого на попередньому етапі. Ітераційний процес зупиняється, коли немає точок даних які змінюють кластери або досягається максимальна кількість ітерацій [17].

Стандартна реалізація методу кластеризації K-середніх представлена у

формулі (2.2):

$$\sum_{i=1}^{N} d\left(x_i, m_j(x_i)\right)^2 \quad (2.2)$$

де d - функція відстані, яка використовується для вимірювання відстані між двома точками;

x_i - і-та точка даних;

$m_j(x_i)$ - центроїд кластера, до якого належить і-та точка даних.

Використання кластеризації за допомогою методу k-середніх у контексті відстеження та інформування про пандемії може дати значний прогрес у сегментації складних наборів даних та надати додаткові аналітичні можливості. Корисність цього методу для аналізу даних Covid-19 полягає у його здатності розділяти великі та складні набори даних на зрозумілі сегменти, що дозволяє виявити глибинні закономірності між ними. Такий аналіз може бути особливо важливим для системи охорони здоров'я.

Кластеризуючи географічні регіони на основі різних показників, таких як кількість випадків, смертей або одужань, метод k-середніх дозволяє органам охорони здоров'я точніше визначати місця, які потребують негайної уваги. Отже цей метод сегментації має велике значення не лише для ефективного розподілу ресурсів, але й для здійснення централізованих заходів. Наприклад, регіони, які знаходяться у кластерах віднесених до високого рівня зараження, але мають низькі медичні ресурси, можуть бути визначені більш пріоритетними для посилення медичних заходів.

У галузі вакцинації застосування кластеризації методом k-середніх може допомогти встановити області з низьким рівнем вакцинації та великою кількістю випадків. Ця інформація має важливе значення для розробки стратегії вакцинації, гарантуючи, що зусилля будуть спрямовані туди, де це найбільш необхідно.

У власній розробці кластеризація методом k-середніх була застосована у модулі "Карти" для створення сегментованої карти відстеження пандемії.

Наприклад, для наочності представлення методу у власній розробці продемонстровано сегментовану карту, яка використовує дані щодо кількості смертей на мільйон осіб. Було визначено 3 кластери та встановлено 3 центроїди відповідно до щільності випадків (25%, 50%, 75%). В результаті отримали 3 групи країн з низьким, середнім, та високим рівнем ризику (рис. 2.2).

Рисунок 2.2 – Реалізація сегментації за методом k-середніх

Підсумовуючи, варто зазначити, що здатність методу k-середніх групувати дані у значущі кластери на основі різних параметрів забезпечує глибше розуміння поширення та впливу пандемії. При поєднанні з іншими аналітичними методами кластеризація методом k-середніх стане важливим активом у наборі

інструментів для відстеження та інформування про пандемії.

2.3 Розробка моделі програмної системи

Моделювання у контексті програмних систем відноситься до високорівневої структури системи, яка описує як різні частини системи взаємодіють одна з одною. Це включає в себе організацію програмних компонентів, зв’язки між цими компонентами та шаблони, які керують цією організацією. Якісне моделювання гарантує, що система буде структурована таким чином, щоб її було легше розуміти, розробляти та тестувати.

У проектуванні та розробці програмного забезпечення на основі моделей, моделювання програмного забезпечення використовується як важлива частина процесу розробки програмного забезпечення. Моделі будуються та аналізуються перед імплементацією системи та використовуються щоб скеровувати наступні імплементації [18].

Моделювання дозволяє вирішити чотири різні завдання:

– візуалізувати систему в її поточному або бажаному для замовника стані;

– описати структуру або поведінку системи;

– отримати шаблон, що дозволяє сконструювати систему;

– документувати прийняті рішення, використовуючи отримані моделі [19].

По суті, моделювання служить схемою як для системи, так і для проекту, оскільки воно визначає робочі завдання, які має виконати команда проектувальників. Кожна конструкція унікальна для певної програмної системи для якої вона створена і відповідає конкретним вимогам та обмеженням цієї системи.

Уніфікована мова моделювання (UML) є однією з найпопулярніших мов моделювання програмного забезпечення, яка пропонує всебічний набір графічних нотацій для моделювання програмних систем з різних точок зору (логіки, поведінки, розгортання, інформації тощо) [20].

UML створювалася як мова моделювання загального призначення для застосування в таких «дискретних» галузях, як програмне забезпечення, апаратні засоби й цифрова логіка [21].

При моделюванні програмного засобу була застосована тришарова архітектура, де перший шар включає у себе користувацький інтерфейс; другий шар реалізує бізнес-логіку програми; третім шаром є сервер бази даних, доступ до якого відбувається з веб-сервера (рис. 2.3).

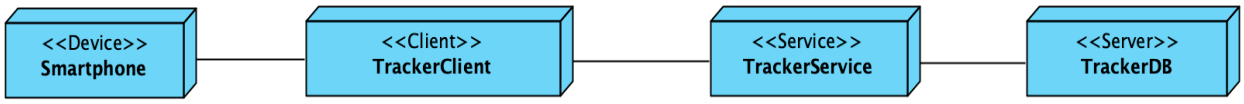

Рисунок 2.3 – Модель програмної системи

Тобто ми маємо рівень презентації, рівень бізнес-логіки та рівень даних.

TrackerClient (презентаційний рівень) - це рівень, з яким взаємодіють користувачі. Він містить компоненти що в цілому відповідають за дизайн інтерфейсу користувача. Наприклад, такі візуальні елементи, як: карти, таблиці, а також результати фільтрації.

TrackerService (рівень бізнес-логіки) - це ядро програми яке контролює функціональність програми. Він керує обробкою даних, обчисленням статистики та всією логікою, що стоїть за функціями програми, як-от отримання даних з API, обробка та агрегування даних, а також підготовка їх для презентації.

TrackerDB (рівень даних) - цей рівень відповідає за керування даними. Він включає компоненти, які взаємодіють із джерелами даних, сховищами даних і моделями даних, які визначають структуру даних.

Кожен шар має певну роль і відповідальність. Така концепція поділу завдань дозволяє легко створювати ефективні ролі і моделі відповідальності. Компоненти в межах певного шару обмежені за сферою дії, вони мають справу лише з логікою, яка відноситься до цього рівня [22].

2.4 Проектування програмної системи

Архітектура програмного забезпечення - це набір важливих проектних рішень про те, як організоване програмне забезпечення для сприяння бажаним ознакам якості та іншим властивостям [23]. Вона визначає структуру для задоволення всіх технічних і операційних потреб програмної системи.

Оскільки архітектура складається зі структур, а структури складаються з елементів і зв'язків, з цього випливає, що архітектура складається з елементів програмного забезпечення та того, як ці елементи пов'язані один з одним. Це означає, що архітектура спеціально оминає певну інформацію про елементи, яка не є корисною для міркувань про систему, зокрема, вона пропускає інформацію, яка не має наслідків за межами окремого елемента. Таким чином, архітектура є передусім абстракцією системи, яка вибирає певні деталі та оминає інші [24].

Без архітектурного мислення та раннього проектування не можна з упевненістю передбачити вартість проекту, його графік і якість. Архітектура визначає ключові підходи для загальної структури розподілу робіт, а також вибір інструментів, навичок і технологій, необхідних для реалізації системи [25].

Якщо архітектура стоїть на останньому місці, то розробка системи стане ще дорожчою, а зміни, зрештою, стануть практично неможливими для всієї системи або її частини [26].

Більш того, проектування передбачає набір важливих організаційних кроків пов'язаних з розробкою програмної системи, де кожен із цих кроків має суттєвий вплив на якість, зручність обслуговування, продуктивність і загальний успіх кінцевого продукту [27].

При роботі над невеликими проектами якісне проектування є корисним, а при роботі над великими - обов'язковим, тому що структура вдалого високорівневого проекту програмної системи може успішно охоплювати та включати в себе цілий ряд більш низькорівневих проектів [28].

Архітектура програмного забезпечення, якщо вона виконана належним чином надає низку переваг, що значно збільшує шанси на успішну роботу програмної системи [29].

Для проектування програмної системи були створені UML-діаграми. Вони використовуються для візуалізації, визначення, конструювання та документування компонентів системи.

UML діаграми полегшують визначення вимог і обсягів систем і програм, надаючи візуальні моделі [30]. Візуалізація - це головна мета UML, оскільки нотації та діаграми забезпечують стандартний галузевий механізм для графічного представлення вимог, процесів, проектувальницьких рішень та архітектури. Ці візуальні елементи створюються за допомогою CASE засобів для моделювання, які також дозволяють здійснювати командний розподіл роботи з моделювання [31].

CASE (Computer-Aided Software Engineering) засіб - це загальний термін, який використовується для позначення будь-якого виду автоматизованої підтримки для розробки програмного забезпечення. В більш вузькому значенні CASE засіб може означати будь-який інструмент, який використовується для автоматизації певної діяльності, пов’язаної з розробкою програмного забезпечення [32].

Ці інструменти призначені для допомоги у різних аспектах розробки програмного забезпечення: від аналізу вимог і проектування до кодування, тестування та обслуговування.

CASE засоби особливо корисні для організації та контролю розробки програмного забезпечення у великих комплексних проектах, що включають багато компонентів та людей. Вони дозволяють дизайнерам, розробникам, тестувальникам і менеджерам мати спільний погляд на проект. Це допомагає забезпечити дисциплінований і контрольований процес [33].

Зрештою, вибір правильного CASE засобу має важливе значення для максимізації продуктивності та забезпечення успішної розробки програмного забезпечення [34].

Існують різні CASE засоби, кожен з яких має свої сильні та слабкі сторони. Деякі з найвідоміших: Rational Rose, Visual Paradigm, Enterprise Architect і StarUML. Вибір конкретного інструменту може залежати від таких факторів, як:

потреби команди, бюджет або складність проекту.

Rational Rose і Visual Paradigm є двома найпопулярнішими CASE засобами, і кожен з них має свої сильні сторони.

Для порівняння цих двох інструментів була створена аналітична таблиця (таблиця 2.1).

Таблиця 2.1 – Таблиця порівняння Visual Paradigm та Rational Rose

Критерій	Rational Rose	Visual Paradigm
Функціональність	Високорівневе системне моделювання, підтримка прямого та зворотного проектування.	Моделювання UML, BPMN, ArchiMate, SysML, підтримка прямого та зворотного проектування, функції керування проектами.
Інтерфейс	Складний, професійний, може потребувати більше часу для освоєння.	Більш зручний та інтуїтивно зрозумілий, легший у опануванні. Дизайн інтерфейсу виглядає сучасніше.
Платформи	Підтримуються платформи Windows, Linux.	Підтримуються платформи Windows, macOS, Linux.
Підтримка мов програмування	Підтримує широкий спектр мов, включаючи мови C++, Java, Visual Basic, .NET.	Підтримує мови Java, C++, PHP, Python, .NET, Ruby та інші.
Інтеграції	Інтегрований з іншими продуктами IBM Rational.	Може інтегруватися з іншими інструментами розробки програмного забезпечення, такими як IDE, інструментами управління проектами тощо.

Продовження таблиці 2.1

Критерій	Rational Rose	Visual Paradigm
Оновлення	Підтримка продукту призупинена, але засіб все ще користується значною популярністю.	Функціонал продукту постійно оновлюється і розширюється.
Вартість	Дорогі корпоративні тарифи, немає можливості придбання у форматі підписки.	Більш доступний; кілька рівнів ціноутворення від 6 доларів США за користувача на місяць [35].

Після порівняльного аналізу було обрано Visual Paradigm, тому що він забезпечує більш зручний та інтуїтивно зрозумілий інтерфейс, у поєднанні з ширшим спектром стандартів проектування та більш доступною ціною, ніж Rational Rose, що робить його універсальним та економічно ефективним рішенням для багатьох проектів. Крім того, він підтримує кросплатформенні операції, що забезпечує більшу гнучкість.

Можливості, які пропонуються Visual Paradigm, є винятковими, адже програмне середовище дозволяє виконувати текстовий аналіз опису проблеми чи бізнес-кейсу, ідентифікуючи найуживаніші поняття та зв’язки між ними в автоматичному режимі [36].

UML діаграми включають діаграми прецедентів, діаграми класів, діаграми взаємодії, діаграми послідовності, діаграми станів, діаграми діяльності, діаграми компонентів та інші.

Діаграми прецедентів описують поведінку системи з точки зору користувача, визначаючи, як користувачі взаємодіють із системою. Вони відображають функціональні вимоги системи та ролі різних користувачів. Діаграми прецедентів використовуються для візуального представлення взаємодії між користувачами (акторами) і прецедентами.

Актори представляють ролі, які виконують користувачі або інші сутності,

коли вони взаємодіють із системою. Актори зазвичай зображуються як людські фігури, але вони також можуть представляти нелюдські сутності.

Прецеденти - це основні функції або сервіси, які надає система своїм акторам. Вони вказуються як овали на діаграмах і визначають конкретні сценарії з точки зору актора, описуючи, які дії або сервіси може надати система для кожного актора. В межах роботи було складено дві таблиці - таблиця акторів (таблиця 2.2) та таблиця прецедентів (таблиця 2.3).

Таблиця 2.2 – Огляд акторів

Актор	Короткий опис
Авторизований користувач	Користувач, який має доступ до всіх функцій додатку без обмеження.
Неавторизований користувач (гість)	Користувач, який має обмежений доступ до функцій додатку.

Таблиця 2.3 – Огляд прецедентів

Прецедент	Короткий опис
Авторизація	Зареєстрований користувач вводить email та password. Якщо відповідний користувач є в базі даних, то користувача успішно переводить у основне меню додатку.
Реєстрація	Незареєстрований користувач вводить email та password. Якщо email та password проходить валідацію, то користувача буде успішно зареєстровано, після чого його автоматично переведе у основне меню додатку.
Гостьова авторизація	Дозволяє неавторизованому користувачу не проходити процедуру стандартної авторизації і зайти в основне меню додатку але з обмеженим функціоналом. Йому буде доступний тільки модуль перегляду карт.

Продовження таблиці 2.3

Прецедент	Короткий опис
Модуль перегляду карт	Дозволяє переглядати інтерактивну світову карту Covid-19, а також користуватись додатковими опціями візуалізації даних про пандемію. Цей модуль доступний авторизованому і неавторизованому користувачу.
Модуль перегляду новин	Дозволяє переглядати останні новини пов’язані з Covid-19 від найбільш відомих видань новин. Цей модуль доступний тільки авторизованому користувачу.
Модуль перегляду рейтингу	Дозволяє переглядати інформацію щодо захворюваності Covid-19 у формі рейтингу країн з можливістю фільтрації за визначеними критеріями (всі випадки, випадки за сьогодні, смерті, одужання). Цей модуль доступний тільки авторизованому користувачу.
Модуль календаря вакцинації	Дозволяє вибрати використаний тип вакцини та дату вакцинації першою дозою вакцини, після чого додаток розрахує дату наступної дози та дату ревакцинації. Цей модуль доступний тільки авторизованому користувачу.
Модуль налаштувань додатку	Дозволяє змінити email та password користувача, а також вийти з додатку. Цей модуль доступний тільки авторизованому користувачу.

Після визначення акторів та прецедентів було складено діаграму прецедентів (рис. 2.4).

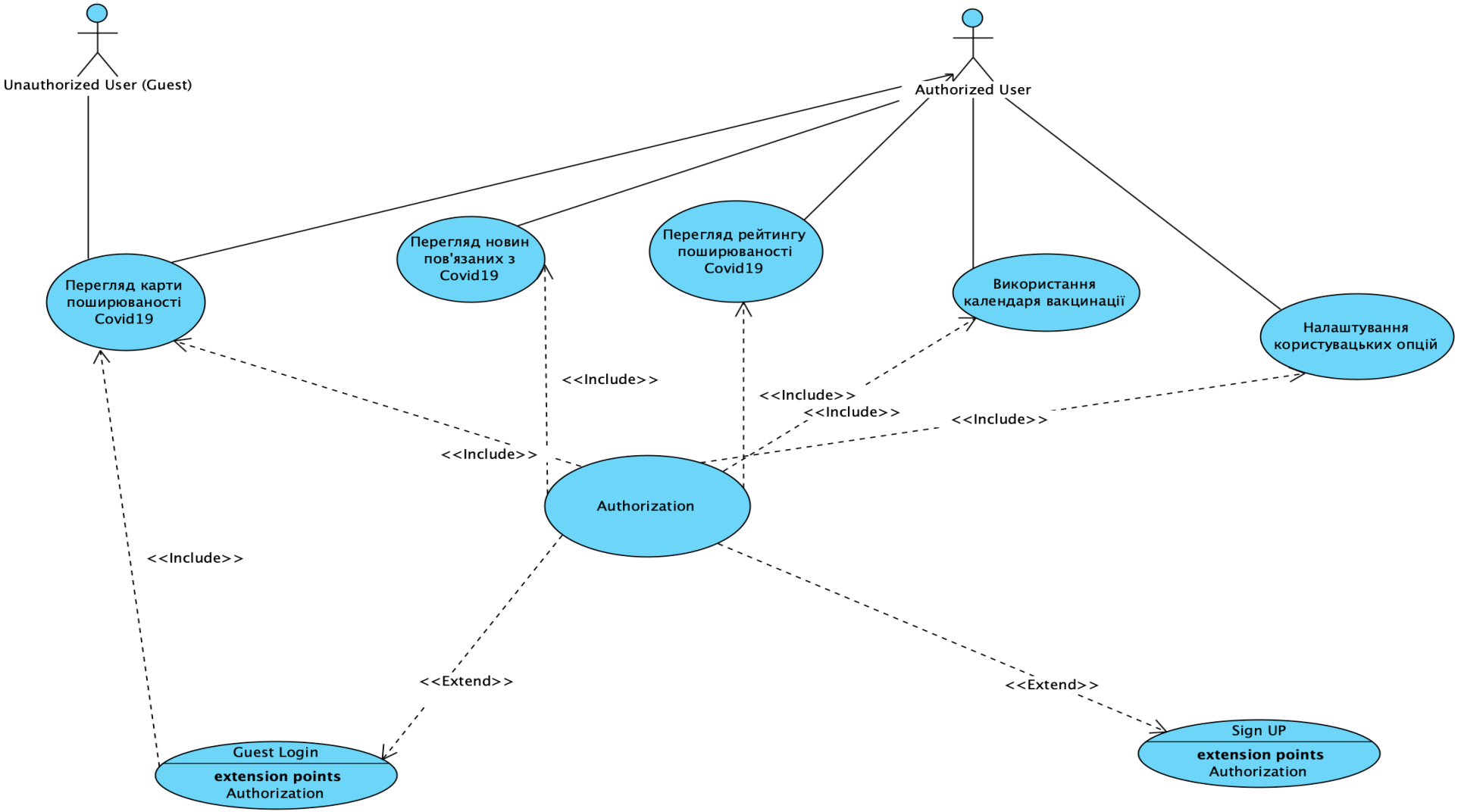

Рисунок 2.4 – Діаграма прецедентів

Діаграми класів представляють класи і зв’язки між ними. Кожен клас на діаграмі відображає назву класу, атрибути і операції класу.

Атрибути - це змінні, що містять інформацію про стан об’єкта класу.

Операції - це методи класу, які виконують певну дію у об’єкті.

Діаграми класів допомагають визначити необхідні класи та їх взаємодію, що сприяє більш організованому проектуванню програмної системи.

Більш того, вони дозволяють на ранніх етапах визначити потенційні проблеми проектування, зменшуючи ймовірність помилок під час розробки.

Сутності діаграми класів та їх атрибути і операції розписані у таблицях 2.4 - 2.10.

Таблиця 2.4 – Сутність Map

Параметр	Значення
Атрибути	- data : MapData - region : Region - clusterizedData: ClusterizedData - heatmapData: HeatmapData - showClusters: boolean - showHeatmap: boolean
Операції	+renderMarkers() +fetchMapData() +zoomIn() +zoomOut() +getClusterizedCountries () +getHeatMap ()

Таблиця 2.5 – Сутність News

Параметр	Значення
Атрибути	-news : News
Операції	+fetchNewsData()

Таблиця 2.6 – Сутність Rating

Параметр	Значення
Атрибути	-data : FilterData -filterName : string
Операції	+fetchRatingData() +filterData()

Таблиця 2.7 – Сутність Vaccination

Параметр	Значення
Атрибути	-vaccineType : string -firstDoseDate : number -secondDoseDate : number -revaccinationDate : number -showDatePicker : boolean -isUserAgreedForNotifications: boolean
Операції	+handleConfirm() +handleNotifications ()

Таблиця 2.8 – Сутність Login

Параметр	Значення
Атрибути	-email : string -password : string
Операції	+signInWithEmailAndPassword()

Таблиця 2.9 – Сутність SignUP

Параметр	Значення
Атрибути	-email : string -password : string
Операції	+createUserWithEmailAndPassword()

Таблиця 2.10 – Сутність Settings

Параметр	Значення
Атрибути	-oldEmail : string -oldPassword : string -newEmail : string -newPassword : string -showEmailInput : boolean -showPasswordInput : boolean
Операції	+updateEmailAndPassword() +logout()

Відносини успадкування та узагальнення представлені за допомогою суцільних стрілок із порожнистим трикутником де один клас успадковує інший. Відносини залежності між класами представлені пунктирними стрілками. Залежності вказують на те, що один клас певним чином покладається на інший, часто у формі викликів методів або передачі параметрів. Асоціації між класами зображуються сполучними лініями.

Були створені наступні групи класів:

– entities - класи які являють собою ключові абстракції: Login, SignUP, Map, News, Rating, Vaccination, Settings.

– control - керуючий клас AppNavigator який керує процесами роутингу.

– boundary - граничний клас AuthManager який керує процесами аутентифікації

Були встановлені відношення між классами:

– всі Entities мають віднощення асоціації з AppNavigator, тому що він керує роутингом між Entities.

– appNavigator має відношення асоціації з AuthManager, адже роутинг залежить від статусу аутентифікації.

– authManager мають відношення реалізації з Login, SignUP, Settings,

оскільки останні надають операції виражені в AuthManager.

У результаті отримано кінцеву діаграму класів із розподіленими класами за пакетами (рис. 2.5).

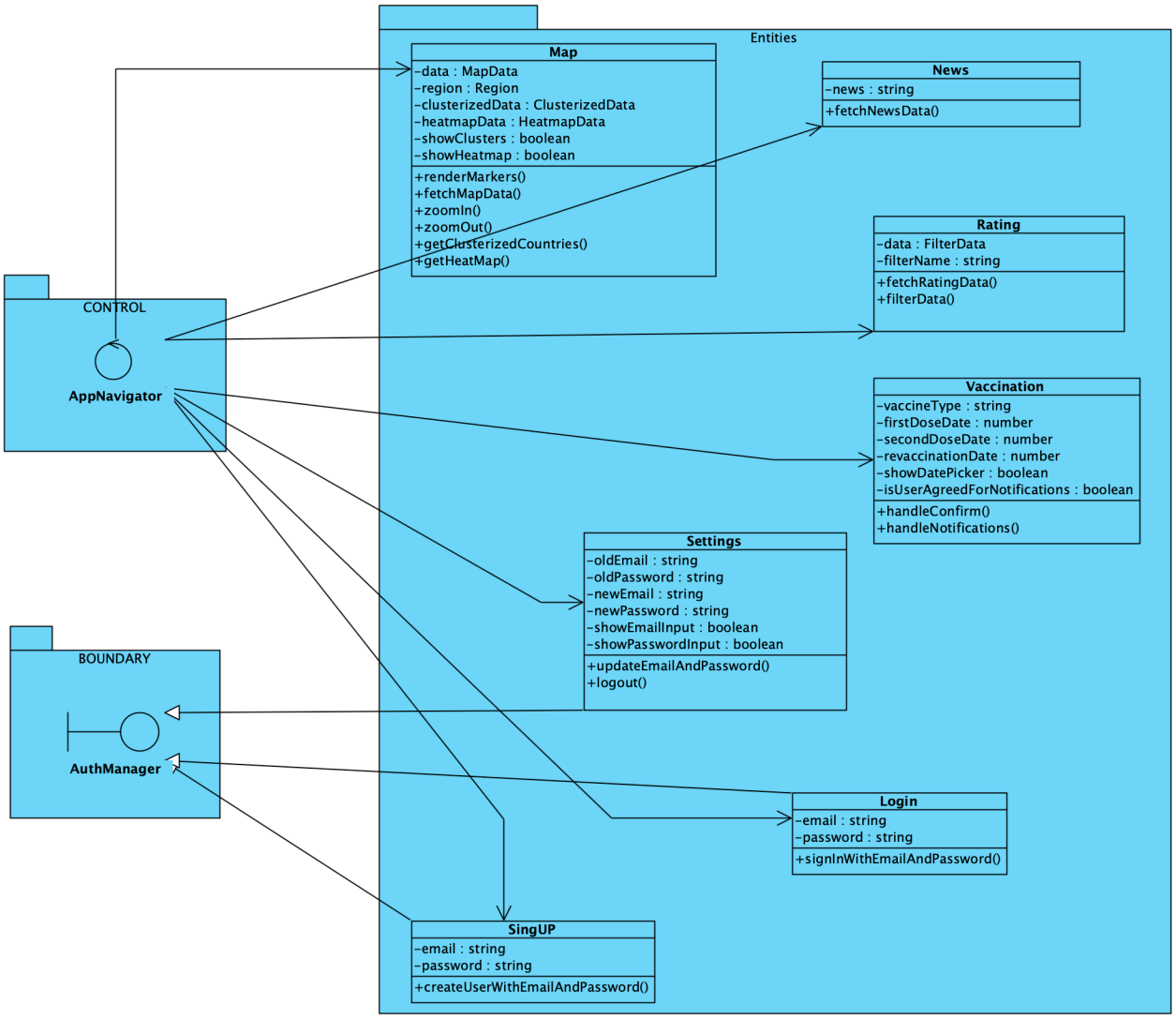

Рисунок 2.5 – Діаграма класів із розподіленими класами за пакетами

Діаграми взаємодії демонструють як об’єкти всередині системи взаємодіють між собою для виконання процесу. Ці діаграми поділяються на два основних типи: діаграми послідовності і діаграми комунікації.

Діаграми послідовностей зосереджені на хронологічному порядку взаємодії. Вони ідеально підходять для висвітлення точних послідовностей подій, де вертикальні лінії представляють кожен об’єкт, а горизонтальні стрілки показують послідовність повідомлень, якими обмінюються об’єкти.

Діаграми комунікації, на відміну від діаграм послідовностей,

зосереджуються на структурній організації, яка показує зв'язки об'єктів з повідомленнями, що пронумеровані для розуміння послідовності взаємодії. Тому вони особливо корисні коли зв'язок між об'єктами так само важливий, як і послідовність повідомлень.

Розглянемо створення діаграми послідовності для сценарію «Вхід зареєстрованого користувача» для прецеденту «Перегляд карти поширення Covid-19». Спочатку були розміщені об'єкти, які надсилають повідомлення, а потім об'єкти, які отримують їх. Ініціатором взаємодії виступає актор User. Далі було розміщено об'єкти Login, AuthManager, AppNavigator, Map (рис. 2.6).

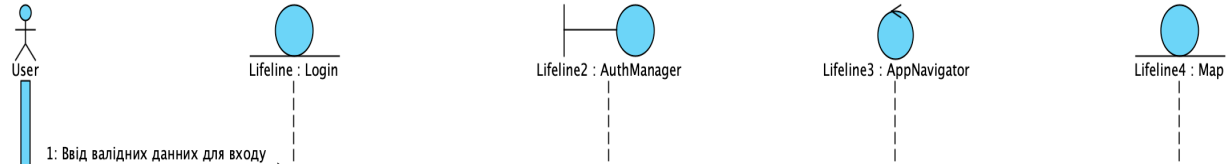

Рисунок 2.6 – Розташування на діаграмі об'єктів послідовності

Для визначення повідомлень якими будуть обмінюватись об'єкти була створена таблиця 2.11.

Таблиця 2.11 – Об'єкти та повідомлення, якими обмінюються об'єкти

№	Об'єкт - відправник повідомлення	Об'єкт - одержувач повідомлення	Назва повідомлення
1	Актор (User)	Login	Введення пошти/паролю
2	Login	AuthManager	Запит на валідацію введених даних
3	AuthManager	AuthManager	Валідація даних
4	AuthManager	AppNavigator	Користувач успішно авторизувався
5	AppNavigator	Map	Перехід на модуль Карт

На основі таблиці 2.11 було створено діаграму послідовності (рис. 2.7)

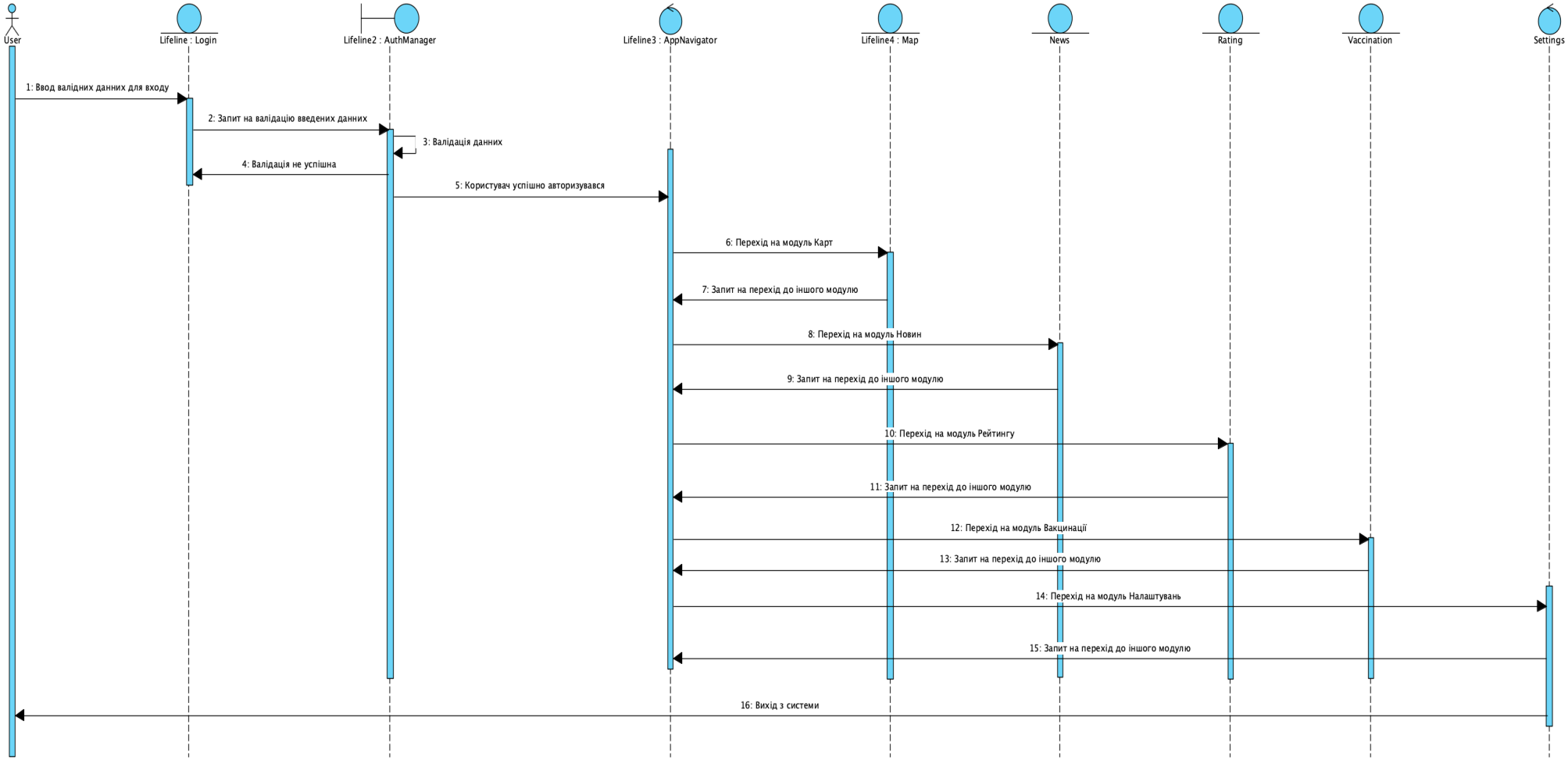

Рисунок 2.7 – Діаграма послідовності

Після створення діаграми послідовності було створено діаграму комунікації для аналогічного сценарію «Вхід зареєстрованого користувача» для прецеденту «Перегляд карти поширення Covid-19» (рис.2.8).

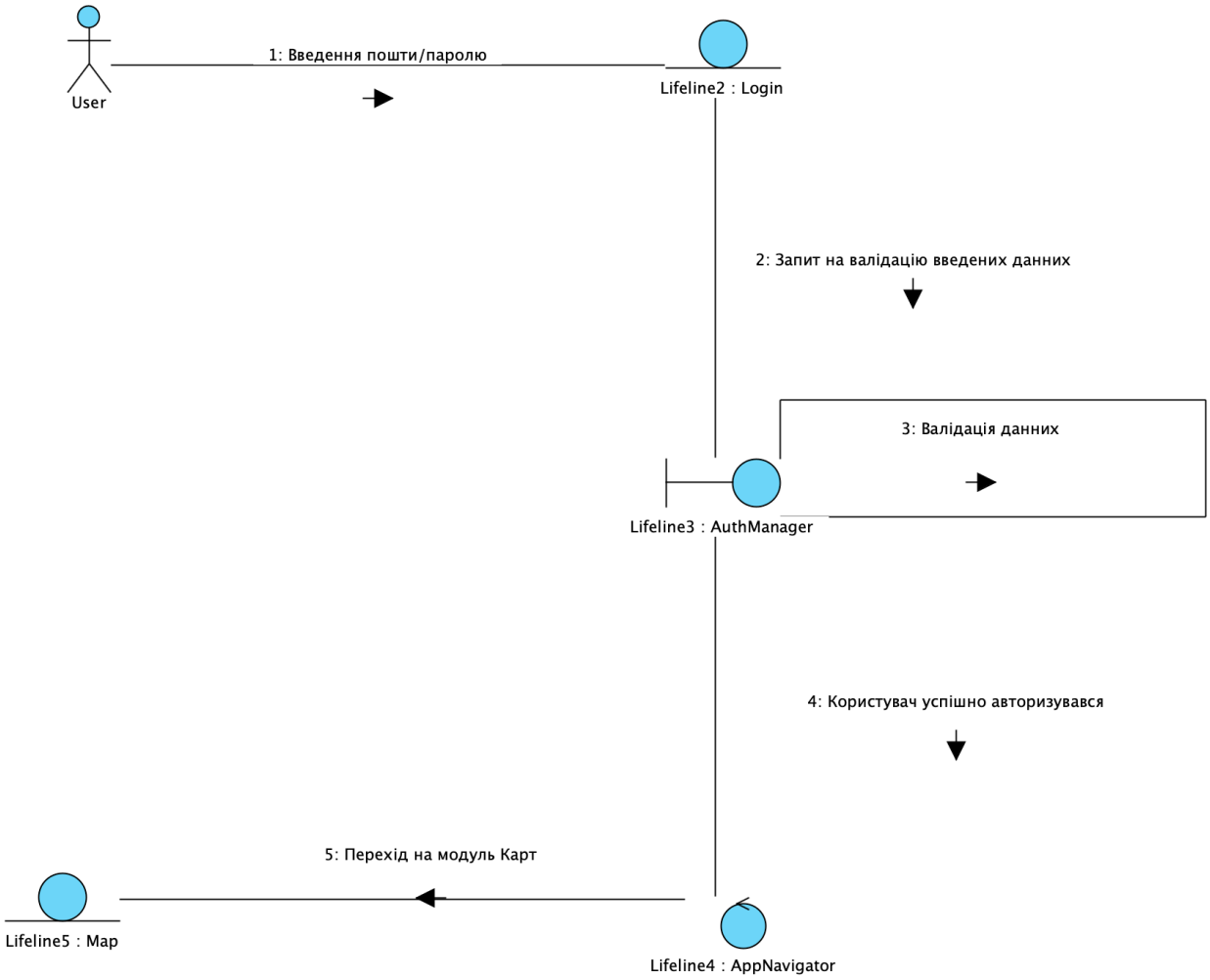

Рисунок 2.8 – Діаграма комунікації

Діаграми діяльності представляють робочі процеси, показуючи послідовність дій та умови їх виконання. Вони корисні для моделювання бізнес-логіки або складної алгоритмічної логіки.

Розглянемо діаграму діяльності для потоку подій прецеденту «Авторизація».

Спочатку користувач обирає одну з можливих дій: авторизація, гостьова авторизація чи реєстрація. В залежності від вибраної дій користувач вводить аутентифікаційні дані які валідуються, або одразу перенаправляється на

презентаційний модуль Карт (рис. 2.9).

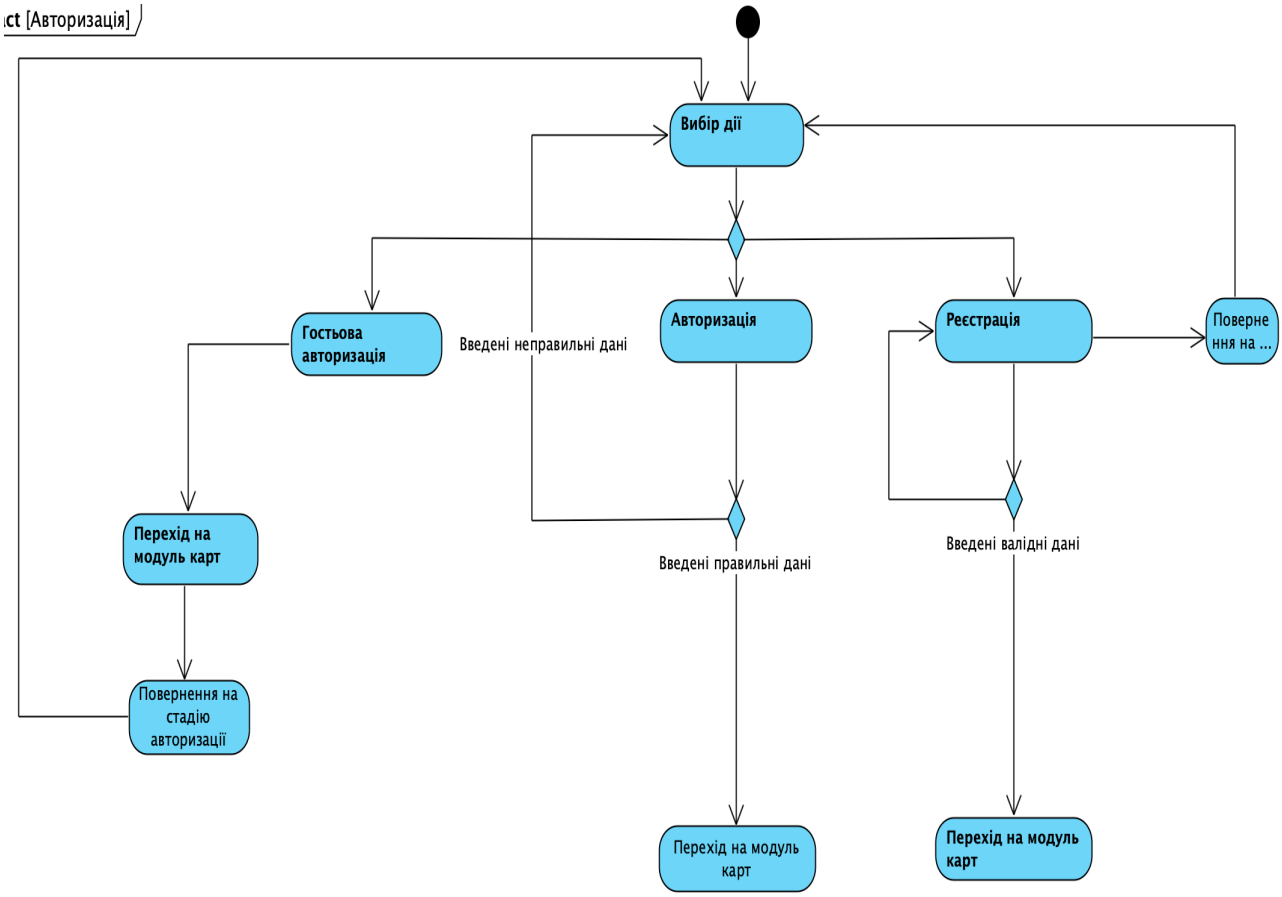

Рисунок 2.9 – Діаграма діяльності для потоку подій прецеденту «Авторизація»

Розглянемо діаграму діяльності для потоку подій прецеденту «Персональні налаштування додатку».

Спочатку користувач обирає одну з можливих дій: зміна пошти, зміна паролю або вихід з системи.

Коли користувач обирає опцію зміни пошти або зміни паролю, йому потрібно спочатку ввести старі дані, а вже потім нові бажані дані і підтвердити свій вибір. Якщо старі і нові дані проходять валідацію (пошта та пароль відповідають потрібному формату) - користувач отримує підтвердження про те що дані змінені. У разі якщо дані не проходять валідацію - користувач отримує попередження проте що він повинен ввести правильну інформацію і спробувати знову.

Коли користувач обирає опцію виходу з додатку, його автоматично

повертає на вікно авторизації (рис. 2.10).

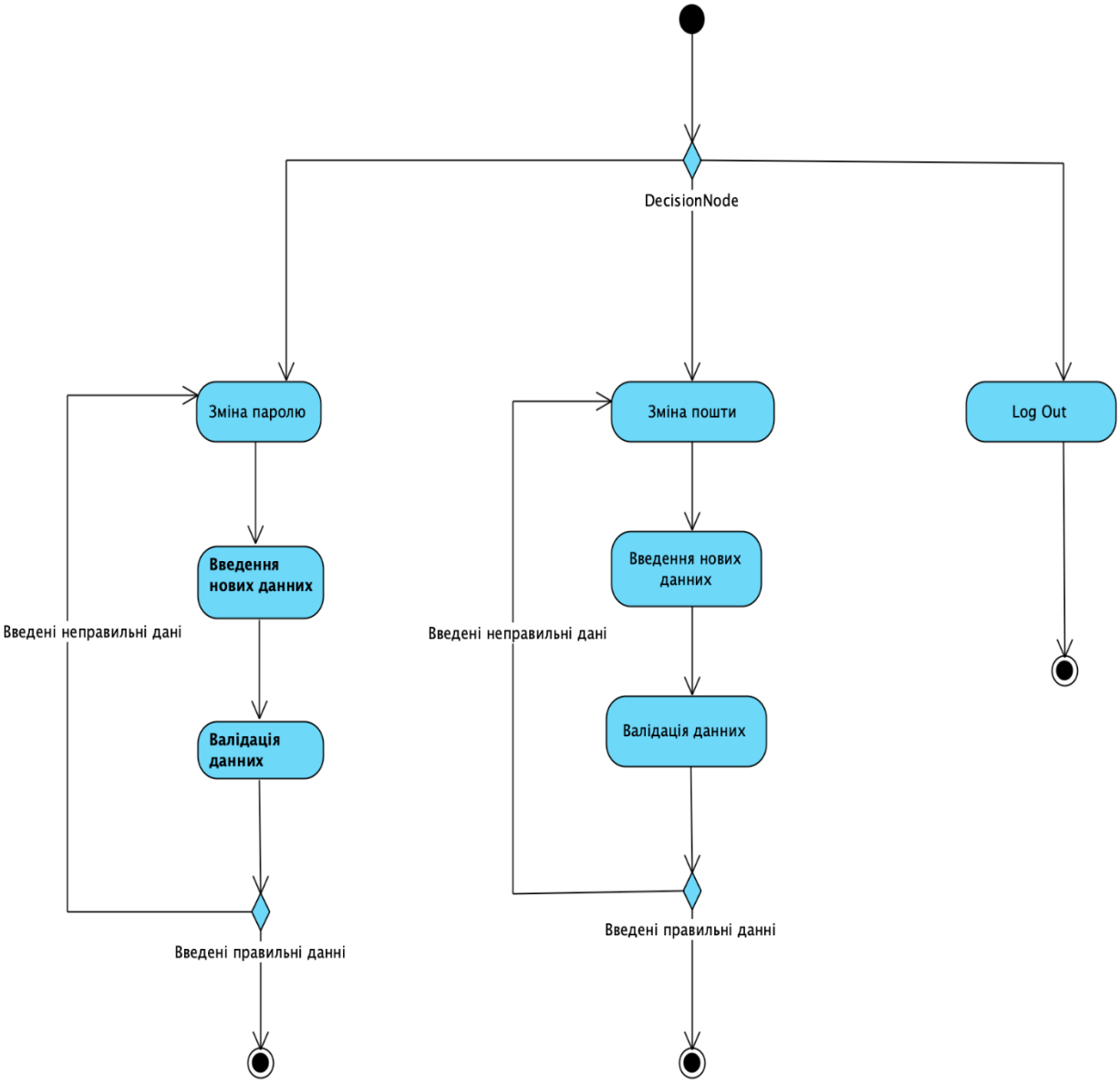

Рисунок 2.10 – Діаграма діяльності для потоку подій прецеденту «Персональні налаштування додатку»

Розглянемо діаграму діяльності для потоку подій прецеденту «Перегляд рейтингу поширення Covid-19».

Спочатку при переході на цей модуль користувач одразу бачить відфільтровані по країнам випадки Covid-19 за весь час збору інформації.

Також користувач може використати вікно вибору для фільтрації інформації за іншими трьома параметрами, крім вищезгаданого: фільтрація

випадки хвороби за сьогодні, загальні випадки смерті, випадками одужання (рис. 2.11).

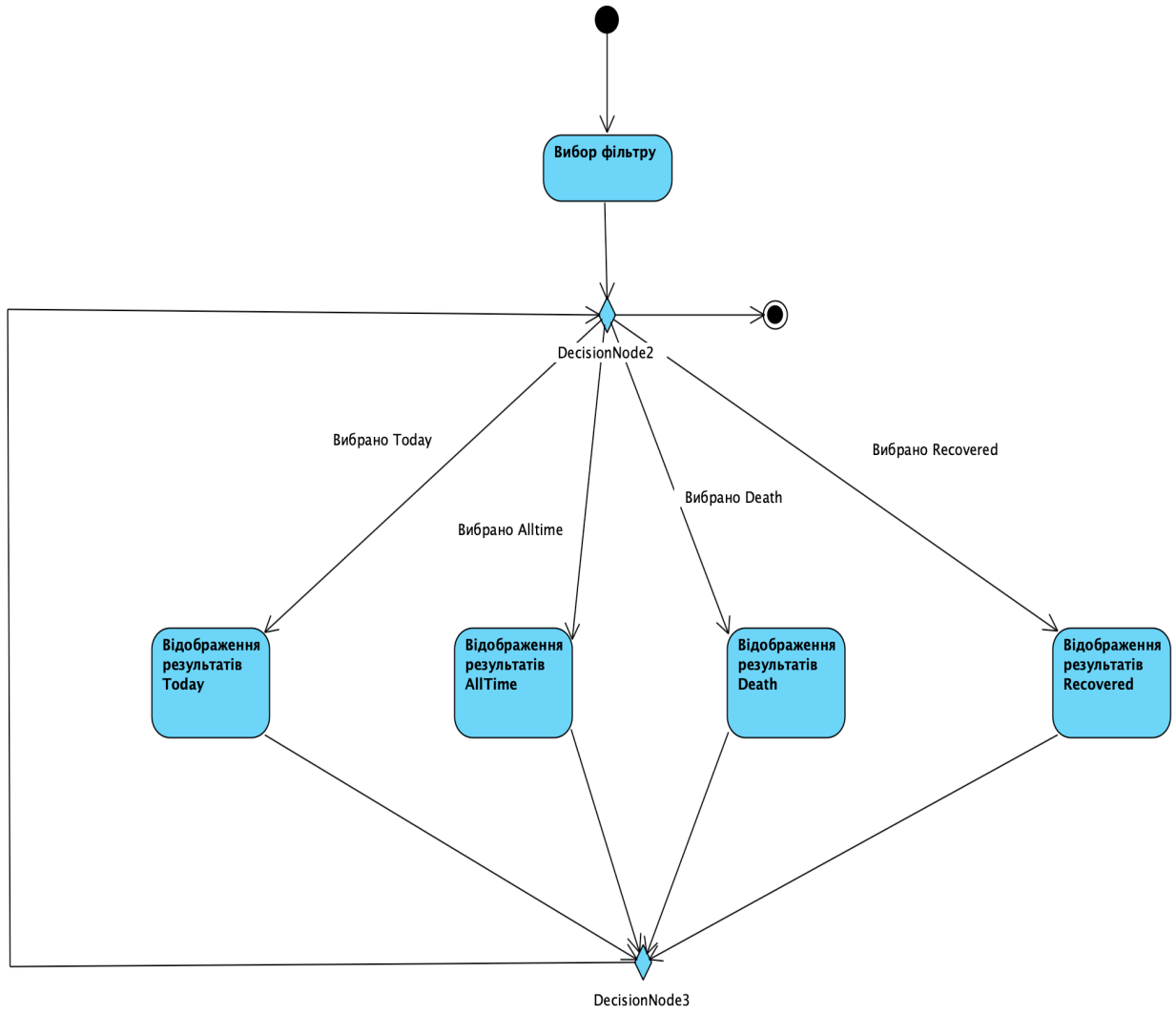

Рисунок 2.11 – Діаграма діяльності для потоку подій прецеденту «Перегляд рейтингу поширення Covid-19»

Розглянемо діаграму діяльності для потоку подій прецеденту «Використання календарю вакцинації».

Спочатку при переході на цей модуль користувач повинен вибрати один з видів вакцини, яка була використана при вакцинації першою дозою щоб запустити алгоритм розрахунку подальшого вакцинування.

Після вибору виду вакцини користувач в інтерактивному вікні може вибрати дату своєї першої вакцинації.

В результаті користувач отримує розраховану дату вакцинування

наступною дозою відповідно до стандартних рекомендацій Всесвітньої організації охорони здоров’я.

Також у користувача є опція підписатись на пуш нагадування про дату вакцинації у разі необхідності (рис. 2.12).

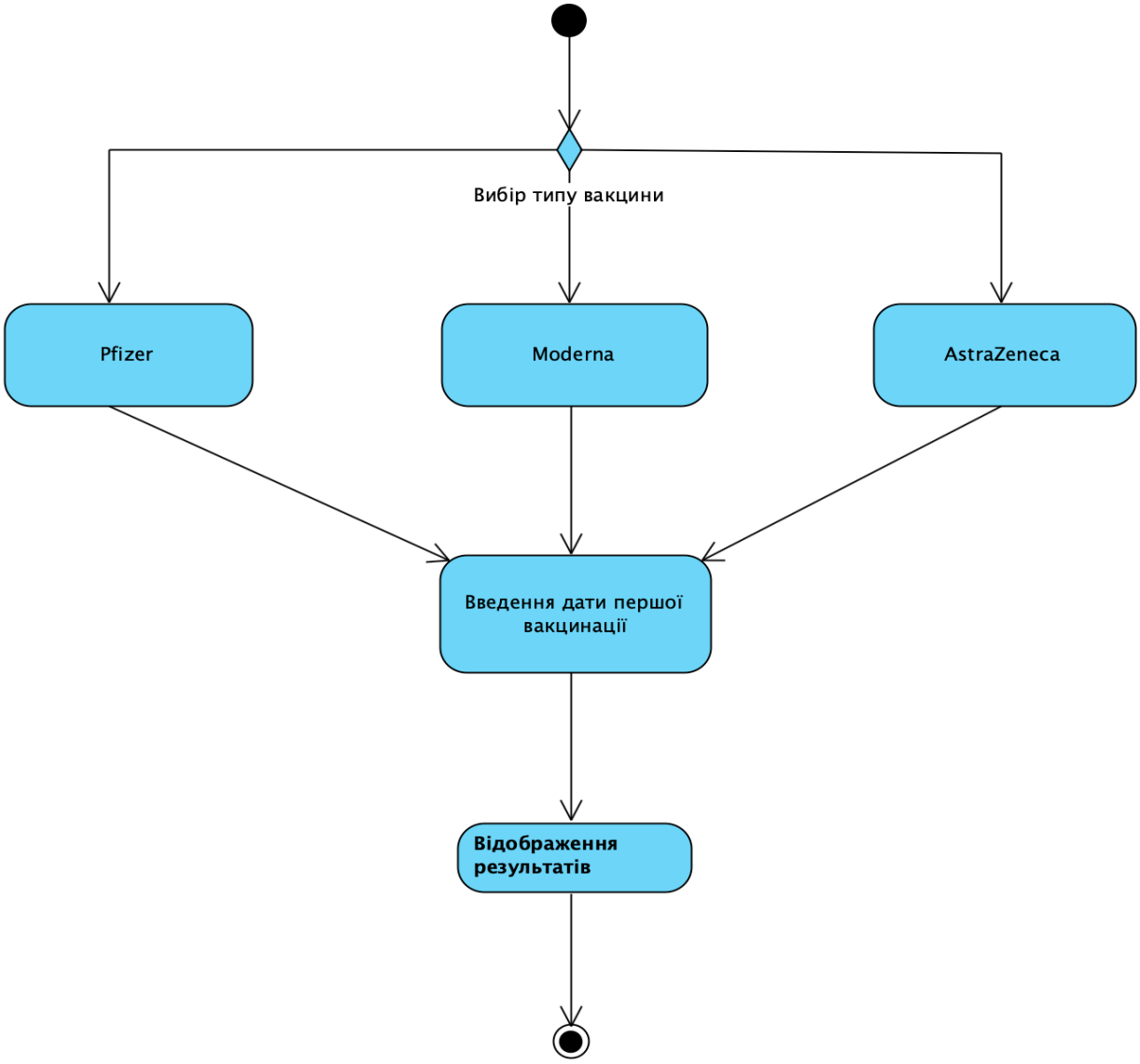

Рисунок 2.12 – Діаграма діяльності для потоку подій прецеденту «Використання календарю вакцинації»

Діаграма станів - це вид діаграм який використовується для представлення різних станів об’єкта чи системи, а також переходів між цими станами.

На цих діаграмах кожен об’єкт або система може перебувати у конкретному стані у певний момент часу, а переходи між станами відбуваються в результаті спрацювання певних дій. Діаграми станів описують поведінку

системи, демонструючи послідовність станів, через які проходить об’єкт у відповідь на події.

Головними елементами цього типу діаграм є стани, переходи та дії.

Стани є основним елементом діаграм станів. Вони позначають різні фази, в яких може існувати об’єкт.

Переходи зображують, як об’єкт переходить з одного стану в інший. Переходи зазвичай позначаються дією що його запускає.

Дії відносяться до станів чи переходів. Вони описують операції, що відбуваються під час входу, виходу або всередині стану.

Розглянемо діаграму станів для прецеденту “Використання календарю вакцинації”.

Для створення діаграми станів для прецеденту «Використання календарю вакцинації» потрібно визначити стани та їх описи (таблиця 2.12).

Таблиця 2.12 – Опис станів для діаграми станів для прецеденту «Використання календарю вакцинації»

Стан	Опис стану
ChooseVaccineType	Entry / SetVaccineList Do / NormalizeString Exit / SetVaccineString
Choose Date	Entry / GetVaccineString Do / CalculateDate Exit / SetDateString
Processing	Entry / GetNormalizedData Do / CalculateInfo Exit / sendCalculatedInfo
Completed	Entry / GetResults Do / CalculateInfo
Cancelled	Do / DiscardResults

На основі таблиці 2.4 будуємо діаграму станів (рис. 2.13).

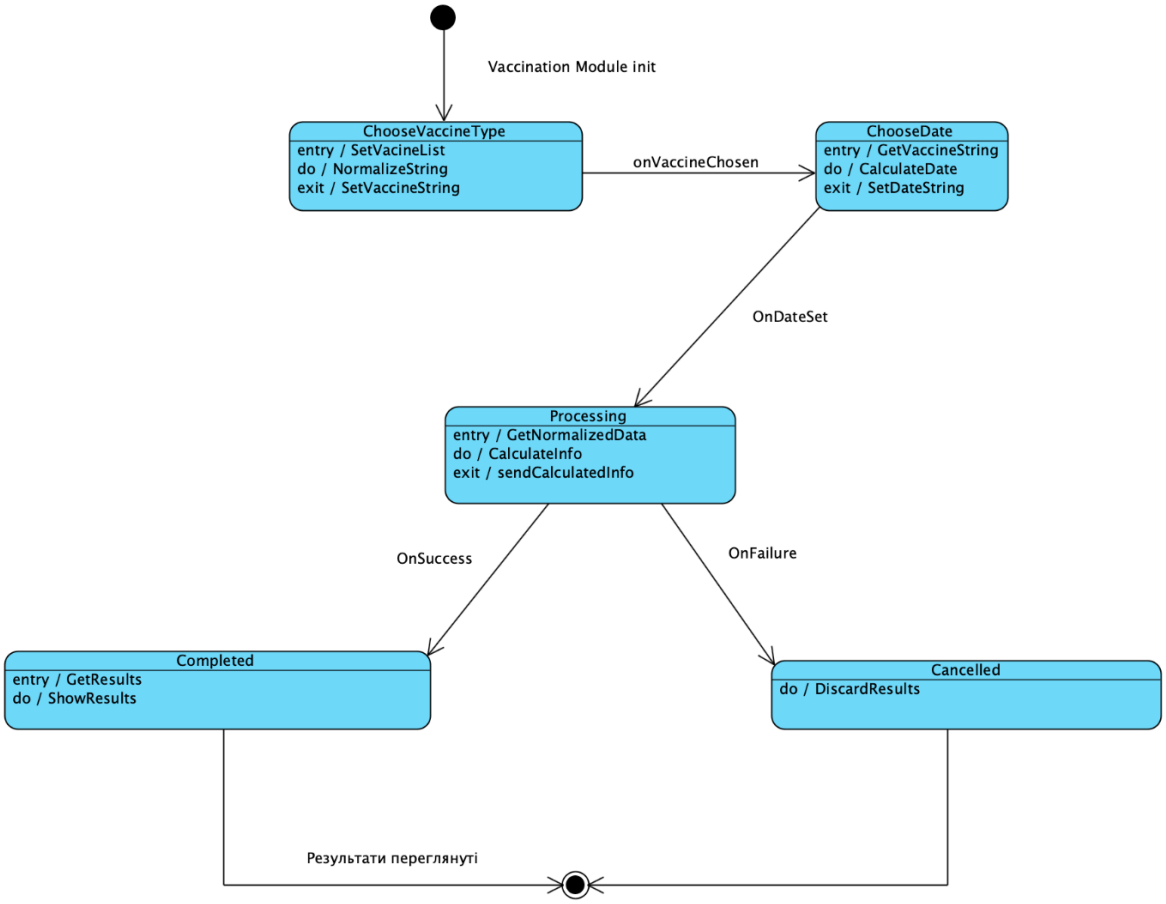

Рисунок 2.13 – Діаграма станів для прецеденту «Використання календарю вакцинації»

Розглянемо діаграму станів для прецеденту “ Перегляд карти поширення Covid-19”. Для створення діаграми станів для прецеденту «Перегляд карти поширення Covid-19» потрібно визначити стани та їх описи (таблиця 2.13).

Таблиця 2.13 – Опис станів для діаграми станів для прецеденту «Використання календарю вакцинації»

Стан	Опис стану
onMount	Entry / getClientData Do / normalizeClientData Exit / setClientData

Продовження таблиці 2.13

Стан	Опис стану
getMapData	Entry / useClientData Do / fetchMapData Exit / sendMapData
Processing	Entry / receiveMapData Do / proccessData Exit / sendProccessedData
Success	Entry / receiveProccessedData Do / showData
Failure	Entry / getFallbackData Do / showFallbackData

На основі таблиці 2.13 будуємо діаграму станів (рис. 2.14).

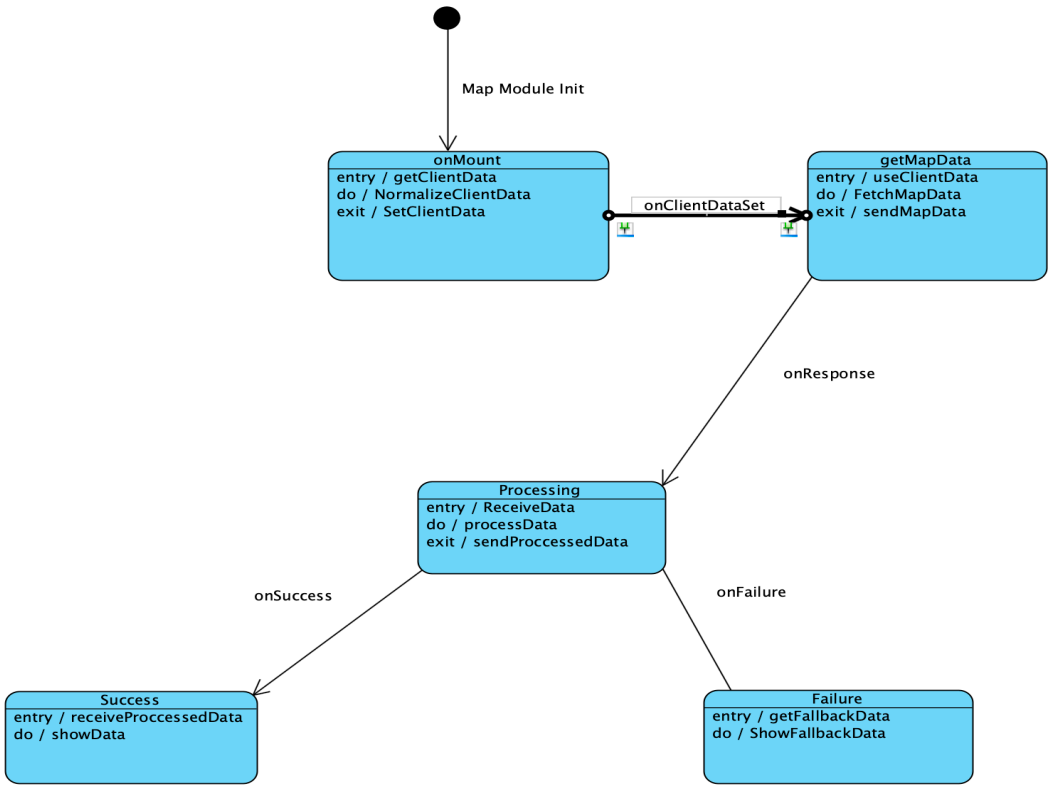

Рисунок 2.14 – Діаграма станів для прецеденту «Перегляд карти поширення Covid-19»

Діаграми компонентів візуалізують взаємозв’язок різних програмних

компонентів. Вони використовуються для організації та керування компонентами.

Діаграма компонентів показує тришарову архітектуру додатку, де є три основні пакети: PresentationLogic, BussinessLogic, DBLogic, а також четвертий пакет який є самим додатком - MainComponent (рис. 2.15).

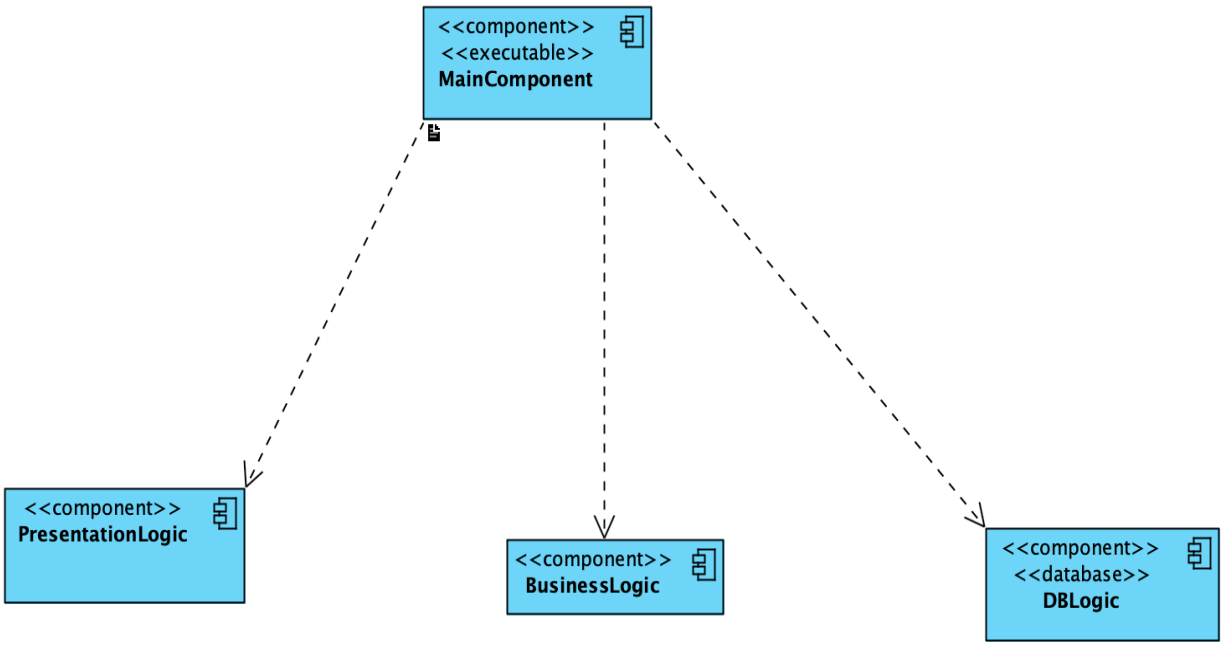

Рисунок 2.15 – Діаграма компонентів

2.5 Висновки

В другому розділі запропоновані покращення методів візуалізації та сегментації даних відстеження пандемії за допомогою методів ядрової оцінки густини розподілу та кластеризації методом k-середніх, що вирішують існуючі проблеми у відстеженні та інформуванні про пандемії. Ці підходи не тільки забезпечують комплексну візуалізацію даних, але й надають можливості аналітичних досліджень.

В результаті проектування програмної системи вимоги до програмного засобу були перетворені у повний, змістовний проект системи, який може служити схемою для подальшої розробки. Кінцевим результатом проектування

стала комплексна специфікація проекту, яка включає такі елементи, як: таблиці прецедентів та акторів, UML діаграми, деталізовані атрибути та операції класів тощо. Це буде гарантувати, що програмний засіб створено структурованим, організованим та ефективним способом, що відповідає визначеним вимогам і цілям дослідження. З вагомих переваг виконаного проектування слід зазначити: покращене розуміння системи, полегшена розробка та тестування, зменшені витрати та час на розробку, ефективніше вирішення потенційних проблем під час розробки.

Підсумовуючи, слід зазначити, що спроектований програмний виходить за рамки традиційного звітування даних, пропонуючи також інструмент аналізу. Він не лише надає дані, але й інтерпретує їх за допомогою передових методів, забезпечуючи глибше розуміння пандемії.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ВІДСТЕЖЕННЯ ТА ІНФОРМУВАННЯ ПРО ГЛОБАЛЬНІ ПАНДЕМІЇ (COVID-19)

3.1 Варіантний аналіз та обґрунтування вибору способу реалізації програмного засобу

Оскільки одною з основних вимог було створення кросплатформенного додатку, який можливо було б використовувати як на Android, так і на iOS без додаткової індивідуальної доробки, вибір проводився між Javascript, Dart і Kotlin.

Ці мови володіють зазначеними можливостями використовуючи сучасні фреймворки.

JavaScript - це прототипно-орієнтована, мультипарадигменна мова з динамічною типізацією, яка підтримує об’єктно-орієнтований, імперативний та декларативний (наприклад, функціональне програмування) стилі програмування [37].

React Native - технологія, яка поєднує найкращі частини нативної розробки з React - кращою у своєму класі бібліотекою JavaScript для створення інтерфейсів користувача [38].

Dart - це оптимізована для клієнта мова програмування швидких програм на будь-якій платформі. Її метою є запропонувати найпродуктивнішу мову програмування для мультиплатформенної розробки в поєднанні з гнучкою платформою виконання для фреймворків [39].

Flutter - це портативний набір інструментів інтерфейсу користувача від Google для створення нативно скомпільованих програм для мобільних пристроїв, вебу та десктопу з єдиної кодової бази. Flutter використовується розробниками та організаціями по всьому світу і є безкоштовним з відкритим кодом [40].

Kotlin - мова програмування, яка є стислою, безпечною, сумісною із Java і іншими мовами та надає багато способів повторного використання коду між різними платформами для продуктивного програмування [41].

Kotlin Multiplatform - технологія розроблена для спрощення розробки кросплатформенних проектів. Це скорочує час, витрачений на написання та

підтримку того самого коду для різних платформ, зберігаючи при цьому гнучкість і переваги нативного програмування [42].

Для визначення мови програмування та відповідних фреймворків, що будуть застосовані для реалізації додатку була створена аналітична таблиця (таблиця 3.1).

Таблиця 3.1 – Порівняння JS, Dart, Kotlin

Характеристика	JS з React Native	Dart з Flutter	Kotlin Multiplatform
Якість UI/UX	Оскільки в React Native використовуються нативні компоненти, продуктивність інтерфейсу користувача близька до продуктивності нативної програми.	Flutter використовує високоякісні віджети, які забезпечують гнучку та послідовну роботу на різних платформах.	Kotlin Multiplatform дозволяє створювати інтерфейс користувача за допомогою нативних технологій, забезпечуючи високий рівень нативного вигляду інтерфейсу.
Сторонні бібліотеки і інструменти	React Native має широкий спектр сторонніх бібліотек завдяки своїй зрілій екосистемі та підтримці Facebook.	Незважаючи на те, що Flutter швидко розвивається, він не має такої ж широкої підтримки сторонніх розробників, як React Native. Але він підтримується Google і має широкий вибір бібліотек.	Мультиплатформа Kotlin є відносно новою, тому її екосистема не є такою розгалуженою, як інші, але вона може використовувати існуючі бібліотеки Kotlin і власні бібліотеки для Android та iOS.

Продовження таблиці 3.1

Характеристика	JS with React Native	Dart with Flutter	Kotlin Multiplatform
Інтеграція з існуючим кодом	React Native дозволяє інтегруватися з існуючим кодом програми, що може бути корисним для поступової міграції на іншу платформу. Це особливо актуально при необхідності мігрувати додаток у веб-версію на React.	Flutter також підтримує цю функцію, але вона може бути дещо складнішою через меншу поширеність Dart.	Kotlin Multiplatform можна легко інтегрувати в існуючі проекти Android, а з додатковими зусиллями також в існуючі проекти iOS.
Інструменти розробки	React Native може використовуватися з будь-яким текстовим редактором або IDE, що підтримує JavaScript.	Dart разом з Flutter має хороший рівень підтримки в різних IDE.	Kotlin Multiplatform добре інтегрується з IntelliJ IDEA та Android Studio.
Особливості мови	JavaScript - це динамічна мова зі слабкою типізацією з великою екосистемою та широким використанням.	Dart - це статично типізована мова, яка може ефективно виявляти помилки під час компіляції.	Kotlin є статично типізованою мовою, яка повністю сумісна з Java. Вона має багато сучасних функцій і може використовувати всю екосистему Java.
Hot Reload	+	+	-

За результатами аналізу було прийнято рішення використовувати мову програмування JavaScript разом з фреймворком React Native.

Переважна більшість веб-сайтів використовує JavaScript, а всі сучасні веб-браузери на комп'ютерах, планшетах і телефонах містять інтерпретатори JavaScript, що робить JavaScript найпоширенішою мовою програмування в історії. За останнє десятиліття Node.js уможливив програмування на JavaScript за межами веб-браузерів, і вражаючий успіх Node означає, що тепер JavaScript також є найбільш використовуваною мовою програмування серед розробників програмного забезпечення [43].

Екосистема JavaScript є динамічною і постійно розвивається разом із впровадженням нових інструментів і технологій, спрямованих на вдосконалення розробки мобільних додатків. JavaScript полегшує доступ до власних функцій мобільних пристроїв, таких як камера, GPS і акселерометр через плагіни та API, надані фреймворками. Це дає розробникам змогу застосовувати власні функціональні можливості пристроїв у своїх мобільних програмах.

React Native заснований на базі React, бібліотеці JavaScript від Facebook для створення користувальницьких інтерфейсів, але замість того, щоб націлюватися на браузер, він націлений на мобільні платформи. Іншими словами, це дозволяє веб-розробникам створювати мобільні програми, які виглядають і відчуваються справді нативними, але все ще використовуючи знайому мову JavaScript [44].

Оскільки браузер виконує JavaScript через віртуальну машину JavaScript, таку як V8, SpiderMonkey та інші, React Native також містить віртуальну машину JavaScript. Там виконується JS-код, здійснюються виклики API, обробляються події дотику та багато інших процесів [45].

Також, додатковим фактором вибору стало те, що використовуючи React Native можливо повторно використати значну частину кодової бази між веб-додатками та мобільними додатками, що потенційно може заощадити багато часу та зусиль на розробку, у разі необхідності створення веб-платформи. Більш того, гнучкість React Native дозволяє писати деякі компоненти у рідному нативному коді, якщо необхідно оптимізувати певні аспекти програми.

3.2 Розробка базових модулів програмної реалізації

За допомогою JavaScript та фреймворку React Native був розроблений засіб для відстеження інформування про глобальні пандемії (Covid-19). Вибір цих інструментів надав можливість ефективно використовувати одну кодову базу для розробки додатків на обох платформах (OS та Android), що заощадило час та ресурси.

React Native використовує бібліотеку React для створення інтерфейсів користувача, що дозволяє створювати компоненти інтерфейсу, які можна повторно використовувати. Це забезпечує однаковий зовнішній вигляд всього додатку. Для керування станом програми та обміном даними використовувалися вбудовані можливості стейт-менеджменту React Native.

Під час розробки програмного засобу Webpack використовувався як інструмент для збирання модулів у програмі. Він допомагає керувати залежностями, а також перетворювати та упаковувати активи проекту. Цей інструмент дозволяє виконувати такі завдання, як транспіляція нових можливостей JavaScript для сумісності зі старими пристроями за допомогою Babel.

За своєю суттю Webpack - це збірник статичних модулів для сучасних програм JavaScript. Коли webpack обробляє програму, він внутрішньо створює графік залежностей з однієї або кількох точок входу, а потім об’єднує кожен модуль потрібний проекту в один або кілька пакетів, які є статичними активами [46].

Для підтримки якості коду та дотримання стандартів кодування використовувався статичний аналізатор коду ESLint.

ESLint - це інструмент-лінтер JavaScript. Він допомагає знайти та виправити проблеми у коді написаному на мові JavaScript. Проблеми можуть бути будь-якими: від потенційних помилок середи виконання до недотримання найкращих практик коду і проблем зі стилями [47].

Додаток також використовує Firebase для серверних служб, таких як авторизації та реєстрації користувачів. Firebase від Google дозволяє

підтримувати та масштабувати бекенд програмного засобу без необхідності створювати і керувати сервером [48].

Для отримання даних про Covid-19 у реальному часі програма використовує API, які надають актуальну та надійну інформацію про пандемію. Ці API інтегровані в програму для отримання даних про поточний стан пандемії, включаючи кількість випадків, одужань та смертей.

В ході розробки програмного засобу варто виокремити наступні створені модулі:

- модуль "Авторизація";
- модуль "Карти";
- модуль "Новини";
- модуль "Рейтинг";
- модуль "Вакцинація";
- модуль "Налаштування";

3.3 Розробка користувацького інтерфейсу програмної реалізації

Для створення інтерфейсу програмного забезпечення було використано вбудовані можливості React Native, який має універсальну та надійну систему для створення користувацького інтерфейсу (UI).

Основним аспектом стилів React Native є фокус на flexbox - це модель макету, яка дозволяє розробникам гнучко та динамічно структурувати та розташовувати елементи інтерфейсу користувача. Гнучкість цього підходу є особливо важливою для адаптації до екранів різних розмірів та орієнтацій.

Стилі в React Native реалізовані з використанням об’єктів JavaScript, що значно відрізняється від традиційного CSS. Це дозволяє гнучко змінювати стилі і дає можливість розробникам застосовувати умови для стилей та адаптувати інтерфейс користувача до різних станів пристрою та взаємодії з користувачем. Властивості стилів значно схожі з класичним CSS, але застосовуються з використанням формату camelCase. Наприклад, backgroundColor замість background-color.

Важливо відзначити, що React Native пропонує широкий спектр готових компонентів, які бездоганно працюють на різних платформах. Вони не залежать від конкретної платформи і перетворюються безпосередньо на нативні елементи інтерфейсу користувача для iOS та Android. Це дає можливість розробникам створювати інтерфейси, які не лише виглядають так само, як у нативних програмах, але також зберігають перевагу продуктивності нативних компонентів інтерфейсу користувача.

Користувацький інтерфейс модулю «Авторизація - Вхід у систему» (рис. 3.1):

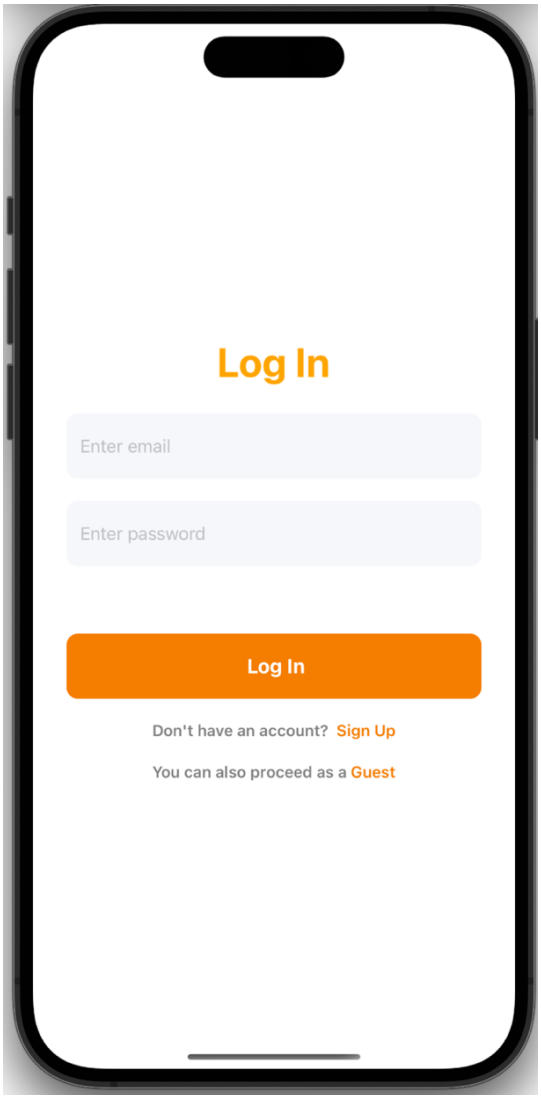

Рисунок 3.1 – Користувацький інтерфейс модулю «Авторизація - Вхід у систему»

В цьому модулі для входу користувача доступно дві опції: вхід

авторизованого користувача і гостьовий вхід. Авторизований користувач - це попередньо зареєстрований користувач, який після входу у свій обліковий запис буде мати змогу користуватись усіма функціями додатку без обмежень. Користувач також може зареєструватись у системі у разі відсутності облікового запису.

Користувацький інтерфейс модулю «Авторизація - Реєстрація» (рис. 3.2):

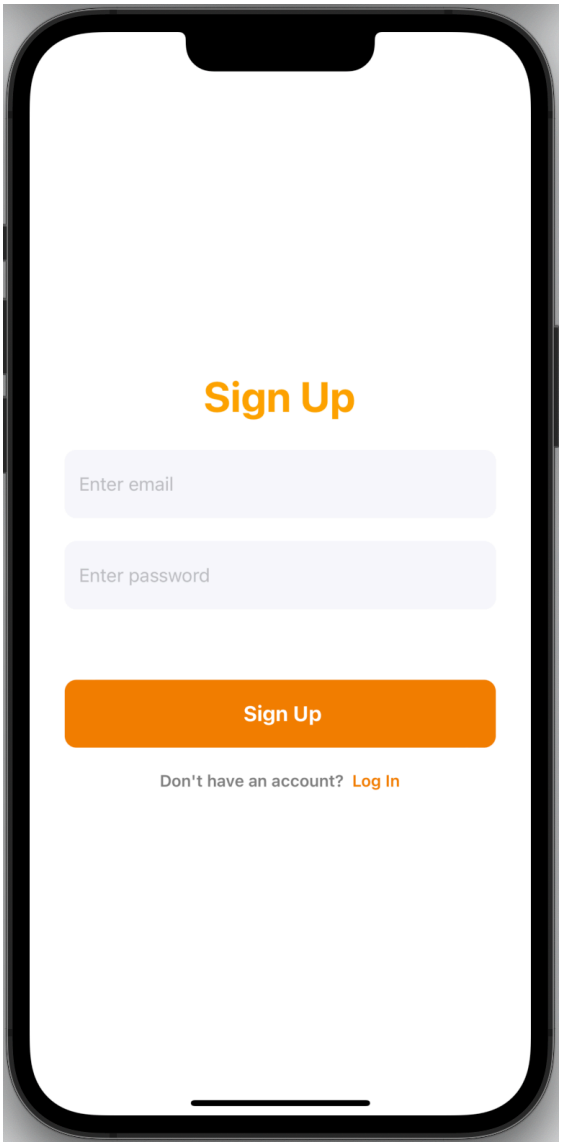

Рисунок 3.2 – Користувацький інтерфейс модулю «Авторизація - Реєстрація»

Користувач може зареєструватись в системі за допомогою вводу поштової адреси і паролю. При здійсненні процесу реєстрації також імплементована

валідація даних користувача. Це означає, що користувач не може ввести будь-яку текстову інформацію у відповідні поля. Валідація не пропустить довільний набір символів замість поштової адреси і не дасть користувачу зареєструватись з слабким паролем.

Користувацький інтерфейс модулю «Гостьовий вхід - Карти» (рис. 3.3):

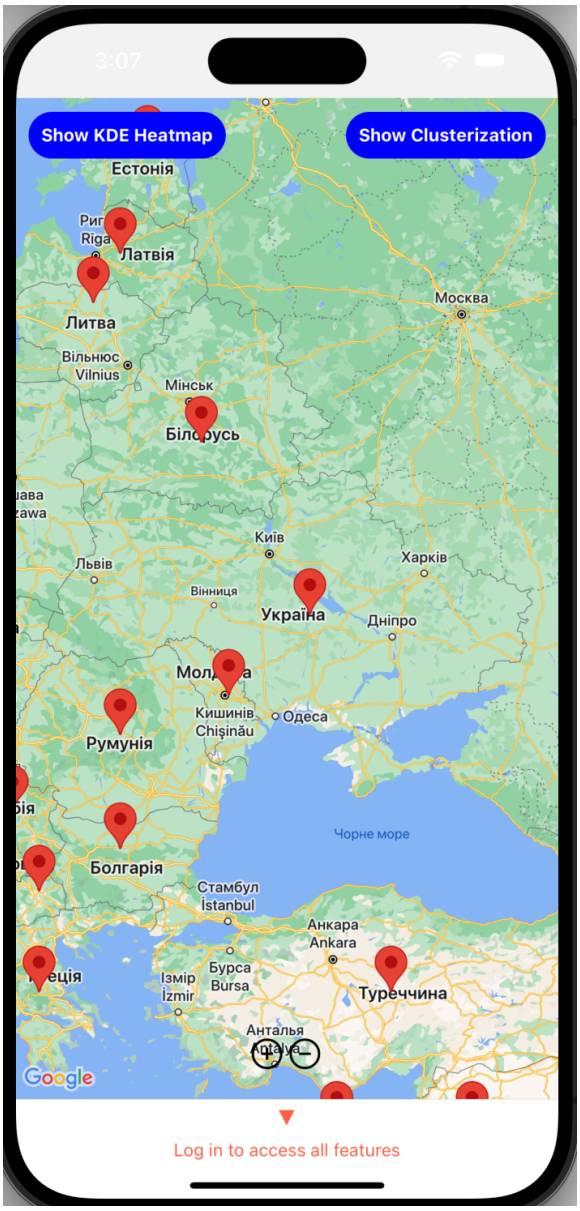

Рисунок 3.3 – Користувацький інтерфейс модулю «Гостьовий вхід - Карти»

Користувач який скористується гостьовим входом не повинен мати обліковий запис, але йому буде наданий доступ до обмеженого функціоналу

додатку. Неавторизованому користувачу буде доступний тільки модуль «Карти».

Користувацький інтерфейс модулю «Карти» (рис. 3.4):

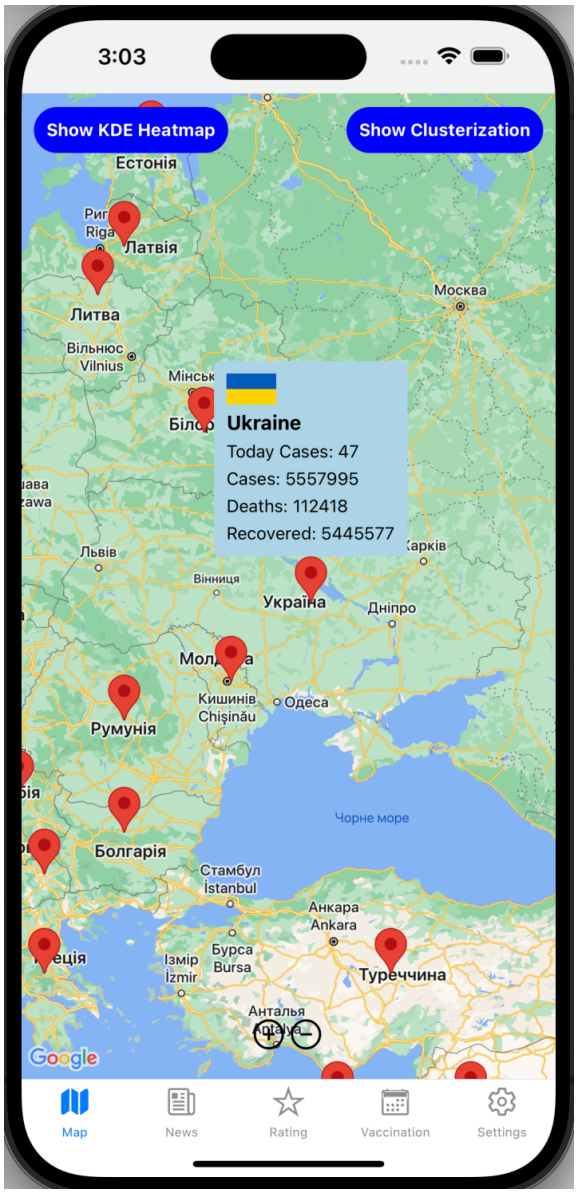

Рисунок 3.4 – Користувацький інтерфейс модулю «Карти»

Цей модуль забезпечує візуальне представлення глобального поширення Covid-19. Він використовує геолокаційні дані, щоб нанести на карту світу підтверджені випадки хвороби, а також одужання та смерті. Користувачі можуть збільшувати та зменшувати масштаб, щоб переглянути певні регіони, і натискати окремі маркери, щоб переглянути детальнішу інформацію про пандемію в цій області. Ця функція надає користувачам швидкий та інтуїтивно зрозумілий

спосіб зрозуміти географічне поширення пандемії.

Надалі користувач може вільно пересуватись між основними презентаційними модулями програми. Для цього було розроблено навігаційне меню яке розташоване на нижньому полі вікна додатку щоб забезпечити найкращий користувацький досвід.

Користувацький інтерфейс модулю «Новини» (рис. 3.5):

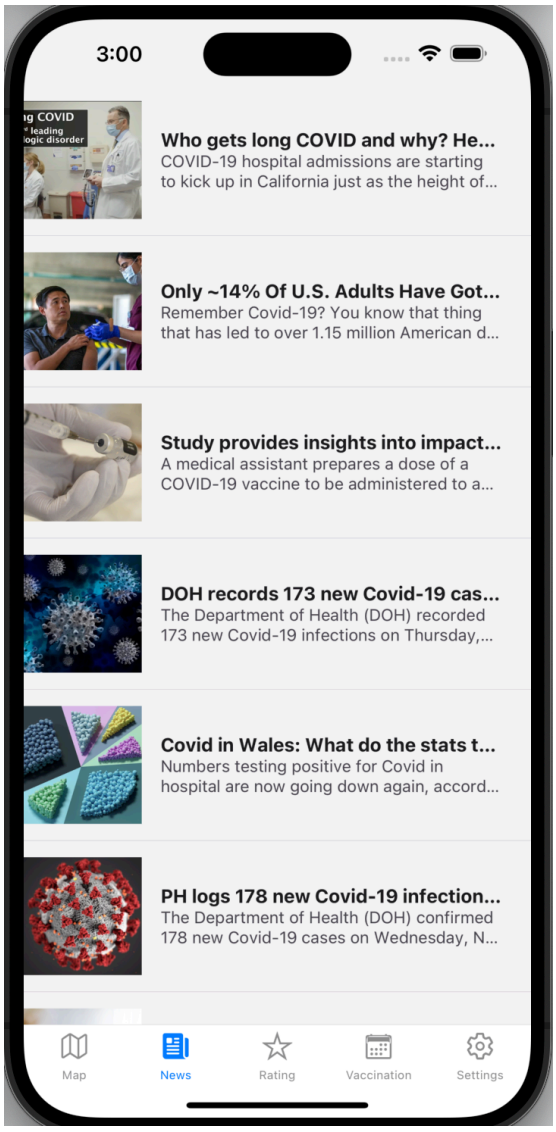

Рисунок 3.5 – Користувацький інтерфейс модулю «Новини»

Модуль «Новини» об’єднує останні статті новин, пов’язані з Covid-19, із різних джерел. Користувачі можуть переглядати ці статті, щоб бути в курсі останніх подій у зв’язку з пандемією. Ця функція забезпечує користувачам

легкий доступ до надійної та актуальної інформації про Covid-19.

Користувацький інтерфейс модулю «Рейтинг» (рис. 3.6):

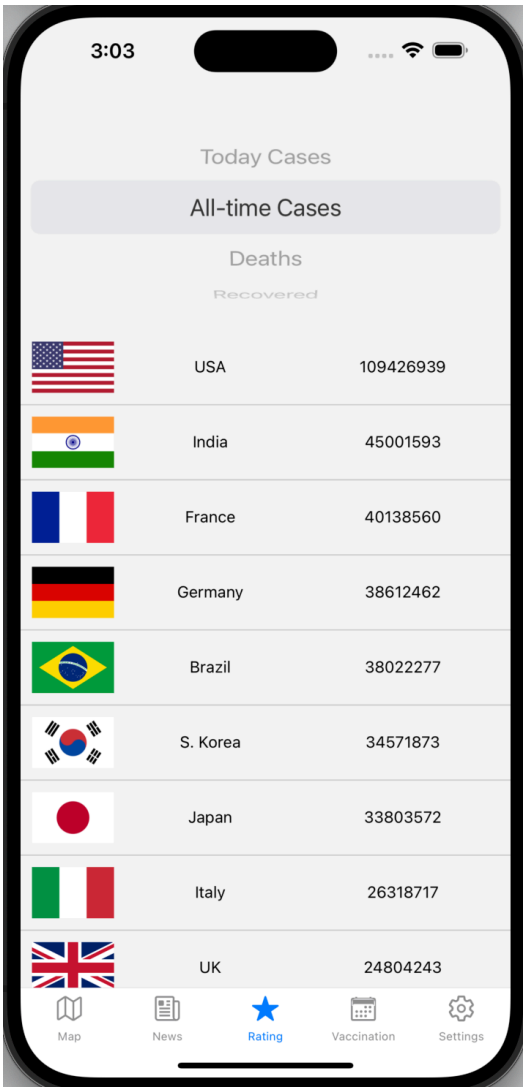

Рисунок 3.6 – Користувацький інтерфейс модулю «Рейтинг»

Модуль «Рейтинг» - це інструмент порівняння у додатку для відстеження Covid-19, який дозволяє побачити, як різні країни справляються з пандемією. Він представляє список країн, відсортований за кількістю зареєстрованих випадків Covid-19, смертей або одужань, що дає змогу зрозуміти глобальний вплив пандемії.

Угорі присутнє спадне меню, яке дозволяє вибрати категорію, яку потрібно порівняти: «Сьогоднішні випадки», «Випадки за весь час», «Смерті» або «Одужавші». Після вибору категорії список країн оновлюється автоматично,

показуючи прапор кожної країни, назву та кількість випадків, смертей або одужань, відповідано до вибраної опції фільтрації. Країни відсортовані в порядку спадання, тому країна з найбільшим числом для вибраної категорії завжди відображатиметься вгорі.

Користувацький інтерфейс модулю «Вакцинація» (рис. 3.7):

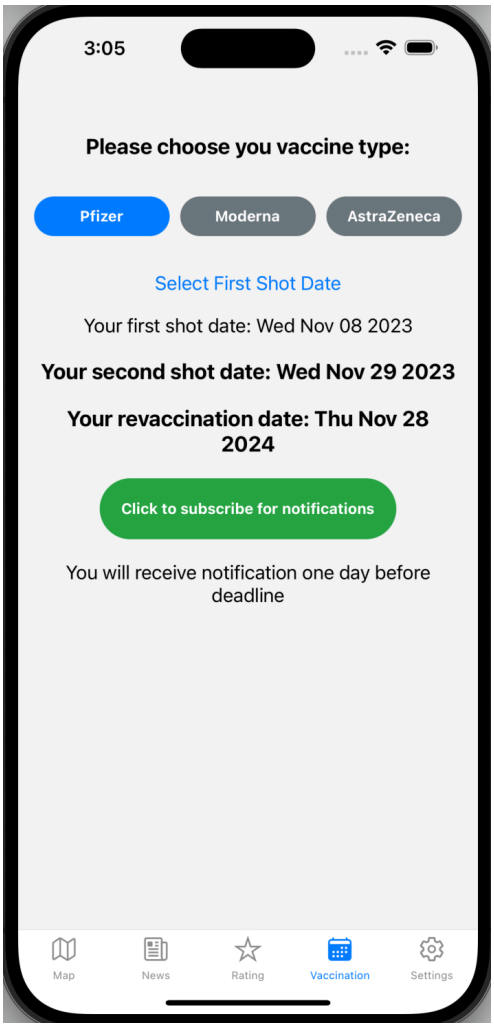

Рисунок 3.7 – Користувацький інтерфейс модулю «Вакцинація»

Модуль «Вакцинація» - це персоналізований інструмент, який допомагає користувачам керувати графіком вакцинації проти Covid-19. Цей модуль забезпечує структурований спосіб відстеження дат вакцинації та надсилає нагадування про майбутні події.

Користувач може вибрати тип вакцини і дату першої вакцинації, після чого програма автоматично розраховує дату другої дози відповідно до рекомендацій

ВОЗ на основі типу вакцини, яку було обрано. Після встановлення дати другої дози програма також автоматично призначає дату ревакцинації після другої дози, дотримуючись загальних вказівок ВОЗ щодо ревакцинації. Більше того, модуль вакцинації піклується про нагадування для користувача. Він планує push-сповіщення на девайс, щоб сповістити користувача за день до другої дози та дати ревакцинаціїї.

Користувацький інтерфейс модулю «Налаштування» (рис. 3.8):

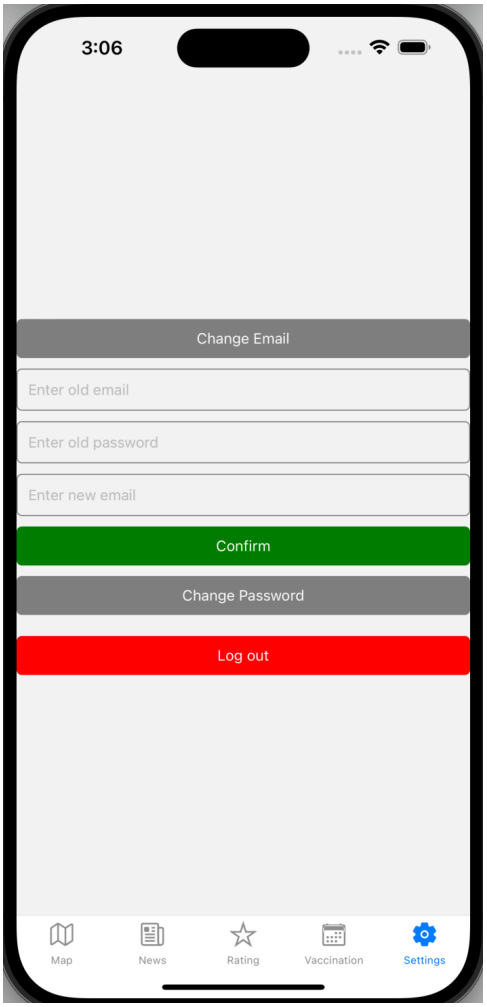

Рисунок 3.8 – Користувацький інтерфейс модулю «Налаштування»

Модуль «Налаштування» є центром для персоналізації та захисту облікового запису в програмі. Користувач може змінити свої пароль та пошту. Для зміни даних користувача необхідно ввести актуальні дати аутентифікації, а також нові бажані дані. Якщо вся інформація правильна, з’являється

повідомлення про успішне виконання. Якщо щось піде не так, програма нагадає про це повідомленням про помилку, щоб допомогти вирішити проблему. Також реалізована можливість виходу з системи натисканням кнопки «Вийти» .

3.4 Висновки

В третьому розділі було здійснено варіантний аналіз шляхів реалізації програмного засобу, за результатами якого вирішено здійснювати розробку за допомогою мови програмування JavaScript. Для розробки додатку був вибраний фреймворк React Native, який є найбільш стабільним, надійним та розповсюдженим варіантом розробки кросплатформенних мобільних додатків на сьогодні.

При розробці програмного засобу також використовувались додаткові програмні інструменти, такі як: збірник модулів Webpack, статичний аналізатор коду ESLint та Firebase для серверних служб.

У результаті був розроблений додаток який включає у себе наступні основні модулі:

– модуль "Авторизація", що відповідає за вхід зареєстрованого користувача, реєстрацію користувача та гостьовий вхід;

– модуль "Карти", в якому користувач може переглядати дані про розповсюдження захворювання на інтерактивній карті світу;

– модуль "Новини", за допомогою якого користувач може прямо у додатку отримати доступ до новин пов’язаних з Covid-19;

– модуль "Рейтинг", в якому користувач може переглянути відфільтровані дані захворювання у вигляді рейтингу країн;

– модуль "Вакцинація", в якому користувач може внести дані вакцинування і отримати пуш нагадування про наступну дату вакцинації;

– модуль "Налаштування", в якому користувач може змінити пошту або пароль, а також вийти з додатку.

РОЗДІЛ 4. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є невід'ємною частиною розробки, яка гарантує високу якість та надійність продукту для кінцевих користувачів.

Існують настпуні методи тестування програмного забезпечення:

– мануальне тестування - це метод тестування, що вимагає участі людини, яка виконує тести без використання автоматизованих інструментів. Основною його ідеєю є детальна ручна перевірка програмного забезпечення з метою виявлення будь-яких невідповідностей із запланованою функціональністю програмного засобу.

– автоматизоване тестування - це метод тестування при якому використовуються спеціальні інструменти для автоматичного проведення тестів. Воно допомагає покращити ефективність процесу тестування, а також підтримує постійну інтеграцію та розгортання системи.

Тестування програмного забезпечення поділяється на кілька типів, кожен з яких відноситься до окремого аспекту якості програмного забезпечення.:

– функціональне тестування: відповідає за процес перевірки програмного забезпечення, щоб впевнитися, що воно відповідає вимогам і специфікаціям, а також що кожна функція працює правильно;

– нефункціональне тестування: відрізняється від функціонального тестування тим, що оцінює такі характеристики, як: продуктивність, зручність використання, надійність і безпека; тобто при цьому типі тестування перевіряються операційні характеристики програмного забезпечення;

– юніт тестування: включає перевірку відокремлених частин програми, зазвичай окремих функцій або методів, щоб переконатися, що кожен блок працює належним чином;

– інтеграційне тестування: перевіряє взаємозв'язки між компонентами або системами, що забезпечує їх злагоджену роботу;

– системне тестування: передбачає комплексний підхід, який оцінює

повний і інтегрований програмний продукт для того щоб переконатися, що він відповідає всім визначеним вимогам;

– приймальне тестування: проводиться для того, щоб переконатися, чи задовольняє програмна система бізнес-критеріям, а також чи готова вона до релізу; це часто включає бета-тестування - коли потенційні користувачі тестують програмне забезпечення в реальних умовах.

Метою тестування програмного забезпечення є виявлення будь-яких помилок, які можуть негативно вплинути на його функціональність, продуктивність або зручність використання. Воно гарантує, що програмне забезпечення відповідає технічним і бізнес-вимогам.

Основним завданням тестування програмного забезпечення є забезпечення високої якості продукту. Це означає, що воно повинне бути надійним, продуктивним, безпечним і відповідати очікуваним стандартам.

Тестування програмного забезпечення є критично важливим етапом для перевірки цілісності програмного продукту. Воно гарантує не тільки правильне функціонування програмного засобу, але й забезпечує гарний користувацький досвід і підвищує готовність до випуску продукту на ринок.

4.2 Тестування розробленого програмного засобу

Для того щоб провести тестування програмного засобу, необхідно детальніше описати функціонал кожного модуля.

Для визначення подій, які відбуваються у кожному модулі була створена аналітична таблиця (таблиця 4.1).

Таблиця 4.1 – Потоки подій

Модуль	Події
Авторизація - Вхід	Система відображає вікно, що містить меню входу, переходу на модуль реєстрації та гостьовий вхід. 1. if Користувач вибирає опцію входу 1.1. Користувач вводить пароль|пошту

Продовження таблиці 4.1

Модуль	Події
Авторизація - Вхід	1.1.1 if Якщо користувач проходить валідацію 1.1.1.1 Користувач успішно авторизується end if 1.1.2 if Якщо користувач не проходить валідацію 1.1.2.1 Користувач бачить помилку end if end if 2. if Користувач вибирає опцію реєстрації 2.1. Користувач переходить до модулю реєстрації end if 3. if Користувач вибирає гостьовий вхід 3.1 Користувач переходить до модулю Карт end if
Гостьовий вхід	Додаткова опція входу в додаток без авторизації 1. if Користувач вибирає опцію гостьового входу на етапі авторизації 1.1. Користувач переходить до модулю Карт end if
Реєстрація	Система відображає вікно, що містить поле пошти і паролю та підтвердження реєстрації. 1. Користувач вводить пароль|пошту 1.1. if Якщо користувач проходить валідацію 1.1.1. Користувач успішно авторизується та переходить в основний модуль додатку end if 1.2. if Якщо користувач не проходить валідацію 1.2.1. Користувач бачить помилку end if

Продовження таблиці 4.1

Модуль	Події
Карти	Система відображає вікно, що містить карту світу з маркером на кожній країні 1. if Користувач виділяє маркер будь-якої країни 1.1. Система відображає інформацію про країну end if 2. if Користувач натискає на іконку приближення 2.1. Масштаб карти збільшується end if 3. if Користувач натискає на іконку віддалення 3.1. Масштаб карти зменшується end if 4. if Користувач натискає на кнопку "Show KDE Heatmap" 4.1. Карти отримують додаткову візуалізацію виконану за допомогою методу ядрової оцінку густини розподілу end if 5. if Користувач натискає на кнопку "Show Clusterization" 5.1. Карти отримують додаткову візуалізацію виконану за допомогою кластеризації методом k-середніх end if
Новини	Система відображає вікно, що містить новини. 1. if Користувач натискає на будь-яку новину 1.1. відкривається браузер, який переводить користувача на відповідну новину end if

Продовження таблиці 4.1

Модуль	Події
Рейтинг	Система відображає вікно, що містить віджет вибору опцій фільтрації та список країн в рейтинговому порядку. 1. if Користувач змінює опцію фільтрації 1.1. Список країн змінюється відповідно до вибраної опції end if
Вакцинація	Система відображає вікно, що містить інтерактивне меню вакцинації (вибір типу вакцини і дату першої вакцинації). 1. if Користувач вибирає і натискає на вакцину якою він вакцинувався та вибирає дату першої дози 1.1. Система відображає інформацію про дату наступної вакцинації end if 2. if Користувач натискає на кнопку "Click to subscribe for notifications" 2.1. Система автоматично налаштує пуш-повідомлення на телефон з нагадуванням про дату наступну вакцинацію для користувача end if
Налаштування	Система відображає вікно, що містить меню зміни паролю та пошти, а також можливості виходу з системи. 1. if Користувач вибирає опцію зміни паролю 1.1. Користувач вводить поточний пароль|пошту та новий пароль 1.1.1 if Якщо користувач проходить валідацію

Продовження таблиці 4.1

Модуль	Події
Налаштування	1.1.1.1 Пароль успішно змінюється end if 1.1.2 if Якщо користувач не проходить валідацію 1.1.2.1 Користувач бачить помилку end if end if 2. if Користувач вибирає опцію зміни паролю входу 2.1. Користувач вводить поточний пароль|пошту та нову пошту 2.1.1. if Якщо користувач проходить валідацію 2.1.1.1. Пошта успішно змінюється end if 2.1.2. if Якщо користувач не проходить валідацію 2.1.2.1 Користувач бачить помилку end if end if 3. if Користувач вибирає гостьовий вхід 3.1. Користувач виходить з системи і повертається у модуль Авторизації end if

Також для здійснення тестування програмного засобу були розроблені юніт-тести для усіх основних модулів додатку. Для тестування використовувалась інструменти Jest та React Native Testing Library.

Jest - це фреймворк тестування JavaScript, призначений для перевірки правильності будь-якої кодової бази JavaScript. Він дозволяє писати тести за допомогою доступного, знайомого та багатофункціонального API, що швидко дає результати [49].

React Native Testing Library - це бібліотека тестування для React Native, натхненна React Testing Library [50]. Ця бібліотека надає багато корисних функцій для пошуку елементів, виконання асинхронних операцій та ін.

Юніт-тестування дуже добре співвідноситься з розробкою основного функціоналу програмного забезпечення. Коли впроваджується новий функціонал, зазвичай спочатку розробляються окремі його блоки, які з часом працюватимуть разом, щоб забезпечити фінальний результат. Розробляючи кожен блок набагато легше переконатися, що він працює належним чином. Ретельне та ефективне тестування невеликих блоків набагато легше, ніж тестування великої частини функціоналу.

Також здійснення юніт-тестування має інші переваги, зокрема:

– юніт-тести швидкі; виконання юніт-тесту зазвичай займає лише декілька мілісекунд, що дозволяє перевірити величезні частини системи за невеликий проміжок часу;

– юніт-тестами легко керувати; юніт-тестування перевіряє програмне забезпечення, надаючи певні параметри для методу, а потім порівнює повернене значення цього методу з очікуваним результатом; ці вхідні значення та очікуване значення результату легко адаптувати або змінити у тесті;

– юніт-тести легко розробляти; вони не вимагають складного налаштування або складних зусиль [51];

Для модулю "Карти" були написані наступні тест кейси:

– перевірка коректного рендерингу маркерів у країн з інформацією щодо захворювання;

– перевірка правильної роботи приближення екрану;

– перевірка правильної роботи віддалення екрану.

Детальний код юніт-тестів цього модулю знаходиться в Додатку В, файл MapScreen.test.js.

Всі тести були успішно пройдені, про що говорять результати запуску тестів (рис. 4.1).

```
PASS tests/MapScreen.test.js
  MapScreen
    ✓ loads and renders markers after fetching data (167 ms)
    ✓ handles zoom in (60 ms)
    ✓ handles zoom out (60 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        1.016 s
```

Рисунок 4.1 – Результати тестів для модуля "Карти"

Для модулю "Рейтинг" були написані наступні тест кейси:

– перевірка коректного відображення вікна очікування у разі завантаження даних з сервера;

– перевірка правильного рендеру даних після їх отримання клієнтом з серверу;

– перевірка правильної фільтрації даних рейтингу відповідно до вибраної опції фільтру.

Детальний код юніт-тестів цього модулю знаходиться в Додатку В, файл RatingScreen.test.js.

Всі тести були успішно пройдені, про що говорять результати запуску тестів (рис. 4.2).

```
PASS tests/RatingScreen.test.js
  RatingScreen
    ✓ renders loading indicator initially (229 ms)
    ✓ renders data after fetching (53 ms)
    ✓ sorts data according to the selected filter (10 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.797 s, estimated 1 s
```

Рисунок 4.2 – Результати тестів для модуля "Рейтинг"

Для модулю "Новини" були написані наступні тест кейси:

– перевірка коректного відображення вікна очікування у разі завантаження даних з сервера;

– перевірка коректного відображення новин після отримання даних з серверу;

– перевірка успішного відкриття вікна новин після вибору конкретної статті.

Детальний код юніт-тестів цього модулю знаходиться в Додатку В, файл NewsScreen.test.js.

Всі тести були успішно пройдені, про що говорять результати запуску тестів (рис. 4.3).

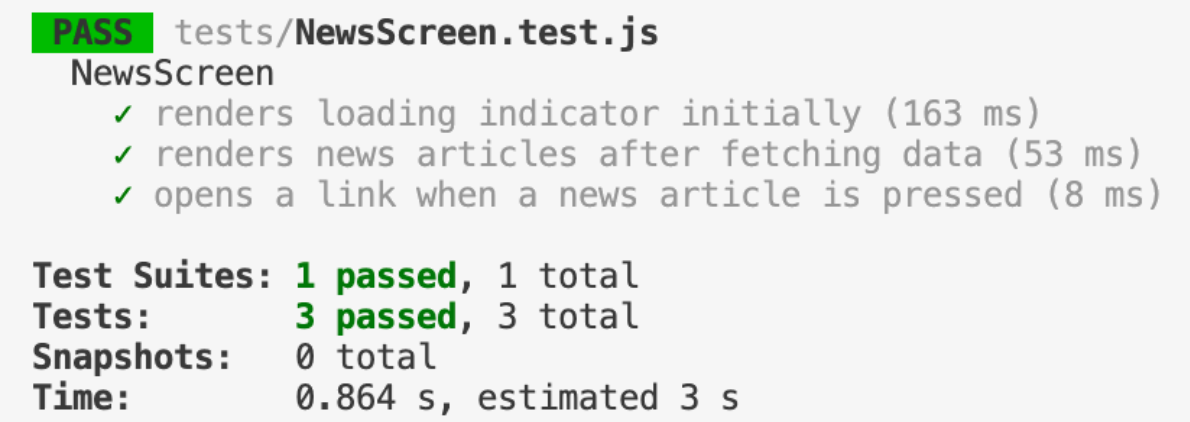

Рисунок 4.3 – Результати тестів для модуля "Новини"

Для модулю "Вакцинація" були написані наступні тест кейси:

– перевірка коректного відображення кнопки вибору дати після вибору типу вакцини;

– перевірка успішного відображення інтерактивного вікна вводу дати;

– перевірка коректного відображення опції підписки на пуш повідомлення якщо користувач правильно ввів усі дані.

Детальний код юніт-тестів цього модулю знаходиться в Додатку В, файл VaccinationScreen.test.js.

Всі тести були успішно пройдені, про що говорять результати запуску тестів (рис. 4.4).

Рисунок 4.4 – Результати тестів для модуля "Вакцинація"

Для модулю "Налаштування" були написані наступні тест кейси:

– перевірка коректного відображення полів вводу нової пошти у разі натискання на кнопку зміни пошти;

– перевірка коректного відображення полів вводу нового паролю у разі натискання на кнопку зміни паролю;

– перевірка успішної зміни пошти у разі вводу користувачем валідних даних для зміни пошти;

– перевірка успішної зміни паролю у разі вводу користувачем валідних даних для зміни паролю;

Детальний код юніт-тестів цього модулю знаходиться в Додатку В, файл SettingsScreen.test.js.

Всі тести були успішно пройдені, про що говорять результати запуску тестів (рис. 4.5).

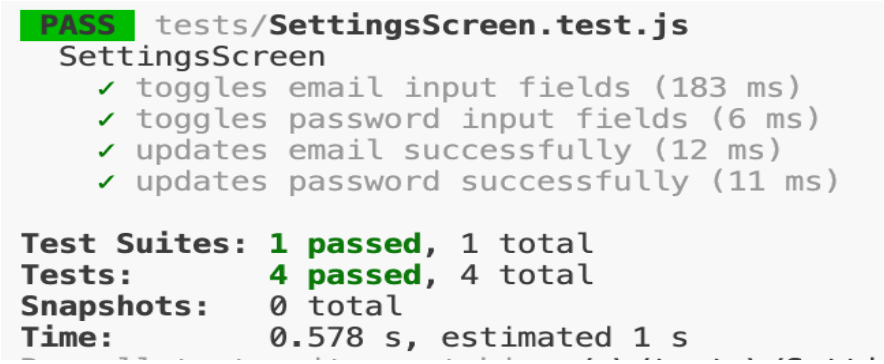

Рисунок 4.5 – Результати тестів для модуля " Налаштування "

4.3 Висновки

В четвертому розділі були проаналізовані методи тестування програмного забезпечення, а також мета та завдання його здійснення.

Для здійснення тестування програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19) були написані юніт-тести для кожного з основних модулів додатку.

Створенні тест-кейси покрили більшість функціоналу програмного засобу та були успішно пройдені. Це однозначно має позитивний вплив на зручність використання, надійність і безпеку програмного засобу, завдяки зменшенню вірогідності прояву невідповідностей із запланованою функціональністю програмного засобу.

РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Для проведення комерційного та технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету к.т.н., доцента Майданюка В. П., к.т.н., доцента Ракитянську Г. Б., к.т.н., доцента Рейду О.М з кафедри програмного забезпечення.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу проведено за допомогою таблиці 5.1, застосовуючи п’ятибальну шкалу оцінювання за 12-ма критеріями оцінки.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
#	1	2	3	4	5
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 5.1

#	1	2	3	4	5
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуренті в немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

#	1	2	3	4	5
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

В таблиці 5.2 наведено результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Майданюк В. П.	2. Ракитянська Г. Б.	3. Рейда О. М.
	Бали, виставлені експертами:		
1	3	4	2
2	2	3	3
3	3	2	2
4	3	4	4
5	3	2	3
6	2	3	2
7	4	3	3
8	4	3	4
9	2	3	3
10	3	4	3
11	4	3	3
12	1	1	2
Сума балів	$СБ_1$=34	$СБ_2$=35	$СБ_3$=34
Середньо арифмети чна сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{34+35+34}{3} = 34.3$		

В таблиці 5.3 наведено шкалу оцінки науково-технічного рівня та комерційного потенціалу розробки.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41...48	Високий

Продовження таблиці 5.3

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
31...40	Вищий середнього
21...30	Середній
11...20	Нижчий середнього
0...10	Низький

За результатами розрахунків, наведених в таблиці 5.2 та шкалою оцінки наведеної в таблиці 5.3 можна зробити висновок щодо рівня комерційного потенціалу розробки. Середньоарифметична сума балів, виставлених експертами склала 34.3, що відповідає рівню «вище середнього».

Такий рівень комерційного потенціалу досягнуто за рахунок значного розширення функціональних можливостей нової науково-технічної розробки порівняно з аналогічними розробками, існуючими в цей час на ринку; суттєве покращення соціальних показників розвитку суспільства, адже програмний засіб відстеження та інформування про пандемії допомагає розробити стратегії для пом’якшення її наслідків для здоров’я, соціального та економічного розвитку.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- витрати на сировину, матеріали та комплектуючі;
- витрати на програмне забезпечення;
- амортизаційні відрахування;
- витрати на паливо та енергію для науково-виробничих цілей;
- витрати на службові відрядження;

– витрати на роботи, які виконують сторонні підприємства, установи і організації;
– інші витрати;
– накладні (загальновиробничі) витрати;

5.2.1 Витрати на оплату праці

До цієї статті належать витрати на виплату основної та додаткової заробітної плати працівникам безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Витрати на основну заробітну плату розраховують відповідно до посадових окладів працівників розраховуються за формулою (5.1):

$$З_0 = \sum_{i=1}^{K} \frac{M_{ni} * t_i}{T_p} \quad (5.1)$$

де k – кількість посад працівників, залучених до процесу дослідження і розробки;

M_{ni} – місячний посадовий оклад конкретного працівника (інженера, дослідника, науковця тощо), грн.;

$Т_р$– число робочих днів в місяці; $Т_р = 22$;

t – кількість робочих днів роботи працівника.

Для проектування і розробки програмної системи для відстеження і інформування про пандемії було залучено наступних працівників:

– Solution Architect;
– Software Engineer;
– Quality Assurance Engineer.

Посадові оклади, число днів роботи та витрати на компенсацію наведено в таблиці 5.4.

Таблиця 5.4 - Компенсація спеціаліста в дослідницькій установі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Кількість днів роботи	Витрати на заробітну плату, грн.
Solution Architect	70000	3182	15	47730
Software Engineer	50000	2273	30	68190
Quality Assurance Engineer	20000	909	7	6363
Всього				122283

Додаткова заробітна плата розраховується як 10...12% від суми основної заробітної плати працівників за формулою (5.2).

Встановлено додаткову заробітну плату в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.2)$$

$$З_д = 0{,}1 * 122283 = 12228 \text{ грн}$$

5.2.2 Відрахування на соціальні заходи

До цієї статті належать нарахування на заробітну плату працівників, які брали участь у виконанні роботи, розраховується за формулою (5.3):

$$З_н = (З_о + З_р + З_д) * \frac{Н_{ЗП}}{100} \text{ (грн)} \quad (5.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$Н_{ЗП}$ – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Основна ставка єдиного внеску на загальнообов'язкове державне соціальне страхування на 2023 рік – 22 %, тоді:

$$З_н = (122283 + 12228) * 0.22 = 39130 \text{ грн}$$

5.2.3 Витрати на сировину, матеріали та комплектуючі

Витрати на комплектуючі, які використовують при дослідженні та розробці нового технічного рішення, розраховуються, згідно з їхньою номенклатурою, за формулою (5.4):

$$K_в = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j \quad (5.4)$$

де $Н_j$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_j$ – коефіцієнт транспортних витрат (1,1…1,15).

Закладений коефіцієнт транспортних витрат – 1,1.

Інформацію про використані матеріали та комплектуючі наведено у таблиці 5.5.

Таблиця 5.5 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір офісний Home & Office A4 500 аркушів	225	1	225
Картридж для принтера Canon PG-46 PIXMA Ink Efficiency Black	659	1	659
Всього			884
З врахуванням коефіцієнта транспортування			972

5.2.4 Витрати на програмне забезпечення

До цієї статті належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення (програм, алгоритмів, баз даних), необхідних для проведення досліджень, а також витрати на їх проектування та розробку.

Балансову вартість програмного забезпечення розраховують за формулою (5.5):

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot С_{пргі} \cdot К_і, \quad (5.5)$$

де $Ц_{іпрг}$ – ціна придбання/використання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_і$ = 1,10…1,12).

k – кількість найменувань програмних засобів.

Закладений коефіцієнт налагодження програмних засобів – 1,10.

Отримані результати наведено в таблиці 5.6.

Таблиця 5.6 – Витрати на використання програмних

Найменування устаткування	Час використання, місяців	Ціна за місяць, грн	Вартість, грн
Підписка Visual Paradigm Standard	1	1405	1405
Google Firebase	1	9367	9367
Всього			10772
З урахуванням коефіцієнта налагодження			11849 грн

5.2.5 Витрати на амортизаційні відрахування

До цієї статті включаються амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проектування та розробки програмної системи.

Амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою (5.6):

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (5.6)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Амортизаційні відрахування наведено в таблиці 5.7.

Таблиця 5.7 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук (2)	90 000*2	5	2	6000
Офісне приміщення	2 000 000	30	2	11111
Всього				17111 грн

5.2.6 Паливо та енергія для науково-виробничих цілей

До цієї статті належать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень.

Витрати на силову електроенергію ($В_е$) розраховують за формулою (5.7):

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і} \quad (5.7)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ – коефіцієнт корисної дії обладнання, $\eta_і < 1$.

У 2023 році Вартість електроенергії становить 7.6 грн за кВт.

Найменування обладнання і його потужність наведено в таблиці 5.8.

Таблиця 5.8 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год
Ноутбук (2)	0.2*2	352
Кондиціонер	0.8	352
Освітлення	0.3	352

Сумарні витрати на електроенергію становлять:

$$В_е = \frac{(0.4 + 0.8 + 0.3) \cdot 352 \cdot 7.6 \cdot 0.5}{0.7} = 2866 \text{ грн}$$

5.2.7 Витрати на службові відрядження

До цієї статті належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуються як 20...25% від суми основної заробітної плати дослідників та робітників за формулою (5.8).

Встановлено витрати на службові відрядження в розмірі 20% від основної заробітної плати.

$$В_{св} = \left(З_о + З_р\right) \cdot \frac{Н_{св}}{100\%} \quad (5.8)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = 122283 * 0.2 = 24457 \text{ грн}$$

5.2.8 Витрати на роботи, які виконують сторонні підприємства, установи і організації

До цієї статті належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30...45% від суми основної заробітної плати дослідників та робітників за формулою (5.9).

Встановлено витрати на роботи, які виконують сторонні підприємства, установи і організації в розмірі 20% від основної заробітної плати.

$$В_{сп} = \left(З_о + З_р\right) \cdot \frac{Н_{сп}}{100} \quad (5.9)$$

де $Н_{пв}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = 122283 * 0.30 = 36685 \text{ грн}$$

5.2.9 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у попередніх статтях витрат і можуть бути віднесені безпосередньо на собівартість розробки за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати робітників за формулою (5.10).

Встановлено інші витрати в розмірі 20% від основної заробітної плати.

$$І_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100} \quad (5.10)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = 122283 * 0.5 = 61142 \text{ грн}$$

5.2.10 Накладні (загальновиробничі) витрати

До цієї статті належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100...150% від суми основної заробітної плати дослідників та

робітників за формулою (5.11).

Встановлено інші витрати в розмірі 100% від основної заробітної плати.

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100}, \quad (5.11)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = 122283 * 1 = 122283$$

5.2.11 Загальні витрати

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою (5.12):

$$В = З_о + З_{дод} + З_н + К_в + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (5.12)$$

$$В = 122283 + 12228 + 39130 + 972 + 11849 + 17111 + 2866 + 24457 + 36685 + 61142 + 122283 = 451006$$

Загальні витрати ЗВ на завершення науково-дослідної (науково- технічної) роботи та оформлення її результатів розраховуються за формулою (5.13):

$$ЗВ = \frac{В_{заг}}{\eta} \quad (5.13)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Оскільки, робота знаходиться на стадії дослідного зразка, то коефіцієнт η = 0.5.

$$ЗВ = \frac{451006}{0.5} = 902012 грн.$$

5.3 Розрахунок економічної ефективності науково-технічної розробки та її можливої комерціалізації потенційним інвестором

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою (5.14):

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \quad (5.14)$$

де $\pm\Delta Ц_0$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_0$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_0 = Ц_б \pm \Delta Ц_0$;

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (0,2...0,5);

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор.

Впровадження результатів науково-технічної розробки дозволить набагато ефективніше і якісніше відстежувати та отримувати інформацію про пандемії. Майбутній економічний ефект буде формуватися на основі збільшення кількості споживачів продукту, в аналізовані періоди часу; зміна вартості пристрою (зростання) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Вихідні дані для розрахунків:

– (N) попит на аналогічні розробки до впровадження результатів нової науково-технічної розробки становить в середньому 20000 підписників у місяць;

– ($Ц_б$) вартість підписки існуючої (базової) науково-технічної розробки у році до впровадження результатів - 500 грн;

– ($\pm\Delta Ц_0$) вартість послуги від впровадження результатів науково-технічної розробки буде зростати кожен рік на 50 грн на протязі перших трьох років;

– (ΔN) кількість підписників кожен рік буде зростати на 10-20% від середнього показника аналогічних розробок на протязі перших трьох років;

– (λ) коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість = 0,8333;

– (ρ) коефіцієнт, який враховує рентабельність інноваційного продукту = 0,25;

– (ϑ) ставка податку на прибуток, який має сплачувати потенційний інвестор = 18%.

Отримаємо:

$$\Delta П_1 = (50 \cdot 20000 + (500 + 50) \cdot 3000) \cdot 0.833 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right)$$

$$= (1\,000\,000 + 1\,650\,000) \cdot 0.170765 = 452\,527 \text{ грн}$$

$$\Delta П_2 = (100 \cdot 20000 + (500 + 100) \cdot 6500) \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right)$$
$$= (2\,000\,000 + 3\,900\,000) \cdot 0.170765 = 1\,007\,514 \text{ грн}$$
$$\Delta П_3 = (150 \cdot 20000 + (500 + 150) \cdot 10000) \cdot 0.833 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right)$$
$$= (3\,000\,000 + 6\,500\,000) \cdot 0.170765 = 1\,622\,268 \text{ грн}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки за формулою (5.15):

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.15)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки. За основу береться 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15. В Україні прогнозований рівень на 2024 рік складає 10,8%;

t – період часу (в роках) від моменту початку впровадження розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{452\,527}{(1 + 0.108)^1} + \frac{1\,007\,514}{(1 + 0.108)^2} + \frac{1\,622\,268}{(1 + 0.108)^3} =$$
$$408418 + 820718 + 1192669 = 2\,421\,805 \text{ грн}$$

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки за формулою (5.16):

$$PV = k_{\text{інв}} \cdot \text{ЗВ} \quad (5.16)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{інв}}$ = 2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 451006 = 902\,012 \text{ грн}$$

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації розробки розраховуємо за формулою (5.17):

$$E_{\text{абс}} = \text{ПП} - PV \quad (5.17)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

$$E_{\text{абс}} = 2\,421\,805 - 902\,012 = 1\,519\,793 \text{ грн}$$

Величина $E_{\text{абс}}$ має велике додатне значення. Це свідчить про потенційну

зацікавленість інвесторів у впровадженні та комерціалізації цієї науково-технічної розробки.

Для остаточного прийняття рішення про впровадження розробки та виведення її на ринок необхідно розрахувати внутрішню економічну дохідність $Е_в$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою (5.18):

$$E_{\text{в}} = T_{\text{ж}}\sqrt{1 + \frac{E_{\text{абс}}}{PV}} - 1 \qquad (5.18)$$

де $Е_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{1\,519\,793}{902\,012}} - 1 = \sqrt[3]{2.68} - 1 = 1.39 - 1 = 0.39 = 39\%$$

Далі визначають бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою (5.19):

$$\tau_{min} = d + f \quad (5.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; В 2023 році в Україні d= (0,13…0,18);
f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0.15 + 0.08 = 0.23$$

Так як $E_в > \tau_{min}$ то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_{ок}$ (DPP, Discounted Payback Period), які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки за формулою (5.20).

$$T_{ок} = \frac{1}{E_в} \quad (5.20)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0.39} = 2.56 \text{ роки} = 30.7 \text{міс} = 2 \text{ роки та } 7 \text{ міс}$$

Так як $T_{ок} \leq 3$ років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

5.4 Висновки

В п’ятому розділі був проведений комерційний і технологічний аудит

програмного засобу для відстеження та інформування про пандемії. За результатами розрахунків був отриманий результат "вище середнього".

Було проведено розрахунок витрат на здійснення науково-дослідної роботи за необхідними статтями, після якого здійснено розрахунок економічної ефективності науково-технічної розробки та її можливої комерціалізації потенційним інвестором.

В ході розрахунку економічної ефективності виявлено, що абсолютний економічний ефект має велике додатне значення, а також короткий період окупності інвестицій, що свідчить про комерційну привабливість науково-технічної розробки для інвесторів.

ВИСНОВКИ

У магістерській кваліфікаційній роботі були розроблені методи та програмний засіб для відстеження та інформування про пандемії на прикладі Covid-19. Подальшого розвитку отримали методи візуалізації та сегментації даних відстеження пандемії за допомогою застосування методу ядрової оцінки густини розподілу та кластеризації методом k-середніх.

Розглянуто актуальний стан питання, аналіз ролі, призначення та сфери програмного засобу, а також здійснено порівняльний аналіз власної розробки з існуючими аналогами та визначено, що подальше дослідження і розробка є доцільними. Здійснено постановку задач дослідження та виконано аналіз вимог до програмного засобу.

Здійснено проектування програмної системи під час якого використано інструментальне програмне забезпечення CASE засіб Visual Paradigm, за допомогою якого було побудовано UML діаграми, а також створено таблиці. В результаті було отримано комплексну специфікацію проекту.

Було виконано варіантний аналіз та обґрунтування вибору способу реалізації програмного засобу. При розробці додатку обрано мову програмування JavaScript, середовище розробки Visual Studio Code, а також фреймворк React Native для реалізації поставлених задач. В процесі розробки було описано програмну реалізацію основних модулів та користувацького інтерфейсу додатку.

Проведено аналіз методів тестування програмного забезпечення, їх мети та завдань. Були створені юніт-тести для кожного з основних модулів додатку, які покрили більшість функціоналу програмного засобу та були успішно пройдені.

Здійснено економічну оцінку програмного засобу, яка показала, що розробка є комерційно привабливою.

Основні положення й результати магістерської кваліфікаційної роботи представлені на науково-практичній конференції і опубліковано в наукових працях

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ågerfalk P. J., Conboy K., Myers, M. D. Information systems in the age of pandemics: Covid-19 and beyond / European Journal of Information Systems, 29(3), 203-207, 2020. [Електронний ресурс]. URL: https://doi.org/10.1080/0960085X.2020.1771968

2. Negro-Calduch E., Azzopardi-Muscat N., Nitzan D., Pebody R., Jorgensen P., Novillo-Ortiz D. Health Information Systems in the Covid-19 Pandemic: A Short Survey of Experiences and Lessons Learned From the European Region / Front Public Health, 9:676838, 2021. [Електронний ресурс]. URL: https://doi.org/10.3389/fpubh.2021.676838

3. Namdev More, Deepak Ranglani, Shubham Kharche, Mounika Choppad andi, Sumanta Ghosh, Sumedh Vaidya, Govinda Kapusetti. Current challenges in identification of clinical characteristics and detection of Covid-19: A comprehensive review / Measurement Sensors, 16(10223), 2021. [Електронний ресурс]. URL: https://doi.org/10.1016/j.measen.2021.100052

4. Пінчуков О. М., Ліщинська Л. Б. Роль програмних засобів для відстеження та інформування про пандемії та їх значення для системи охорони здоров'я / Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р., с. 199 - 201. - Вінниця: ВНТУ.

5. Prakash Narain J, Sodani PR, Kant L. Covid-19 Pandemic. Lessons for the Health Systems / Journal of Health Management, 23(1), 74-84, 2021. [Електронний ресурс]. URL: https://doi.org/10.1177/0972063421994982

6. Meyer, Brent H. and Prescott, Brian C. and Sheng, Xuguang Simon. The Impact of the Covid-19 Pandemic on Business Expectations / FRB Atlanta Working Paper No, 2020-17, 2020. [Електронний ресурс]. URL: https://dx.doi.org/10.2139/ssrn.3829894

7. Abbas Jaffar, Riaqa Mubeen, Paul Terhemba Iorember, Saqlain Raza, Gulnara Mamirkulova. Exploring the impact of Covid-19 on tourism: transformational

potential and implications for a sustainable recovery of the travel and leisure industry / Current Research in Behavioral Sciences, 2(2021), 2021. [Електронний ресурс]. URL: https://doi.org/10.1016/j.crbeha.2021.100033

8. Пінчуков О. М., Ліщинська Л. Б. Аналіз можливостей застосування програмних засобів для відстеження та інформування про пандемії / Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р., с. 197 - 199. - Вінниця: ВНТУ.

9. Google Play Market. [Електронний ресурс]. URL: https://play.google.com/store/apps

10. Apple App Store. [Електронний ресурс]. URL: https://www.apple.com/app-store/

11. WHO Covid-19 Map. [Електронний ресурс]. URL: https://covid19.who.int/

12. Covidvisualizer App. [Електронний ресурс]. URL: https://www.covidvisualizer.com/

13. Coronavirus App. [Електронний ресурс]. URL: https://coronavirus.app/map

14. Li Q. Overview of Data Visualization / Embodying Data, 20:17-47, 2020 Jun. [Електронний ресурс]. URL: https://dx.doi.org/10.1007/978-981-15-5069-0_2

15. Chen Yen-Chi. A Tutorial on Kernel Density Estimation and Recent Advances / Biostatistics & Epidemiology, 1:1, 161-187, 2017. [Електронний ресурс]. URL: https://dx.doi.org/10.1080/24709360.2017.1396742

16. V. Estivill-Castro Why so many clustering algorithms: a position paper. ACM SIGKDD Explorations Newsletter, 2002, 4(1), 65-75.

17. N. Slonim, E. Aharoni and K. Crammer, “Hartigan’s K-Means Versus Lloyd’s K-Means-Is It Time for a Change? Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013, pp. 1677-1684.

18. Gomaa H. Software Modeling and Design : Uml Use Cases Patterns and Software Architectures / Cambridge: Cambridge University Press, 2011 - 550p.

19. Петрик М.Р. Моделювання програмного забезпечення : науково-методичний посібник / М.Р. Петрик, О.Ю. Петрик - Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. - 200 с.

20. Mert Ozkaya, Ferhat Erata. Understanding Practitioners' Challenges on Software Modeling: A Survey / Journal of Computer Languages, Volume 58, 2020.

21. Проектування та моделювання програмного забезпечення сучасних інформаційних систем / Г. В. Табунщик, Т.І. Каплієнко, О.А. Петрова - Запоріжжя : Дике Поле, 2016. - 250 с.

22. Richards M, Ford N. Fundamentals of Software Architecture : An Engineering Approach. First ed. / O'Reilly Media, 2020 - 422p.

23. Keeling M. Design It! : From Programmer to Software Architect / Raleigh North Carolina: Pragmatic Bookshelf, 2017 - 408p.

24. Bass L, Clements P, Kazman R. Software Architecture in Practice. Fourth ed. / Boston: Addison-Wesley, 2021 - 464p.

25. Cervantes H, Kazman R. Designing Software Architectures : A Practical Approach / Boston: Addison-Wesley; 2016 - 320p.

26. Martin RC. Clean Architecture : A Craftsman's Guide to Software Structure and Design / Harlow England: Prentice Hall, 2017 - 432p.

27. Jaiswal Manishaben. Software Architecture and Software Design / International Research Journal of Engineering and Technology, 2395-0072, Volume: 06 Issue: 11, s. no -303 , pp. 2452-2454 , Nov 2019. [Електронний ресурс]. URL: https://dx.doi.org/10.2139/ssrn.3772387

28. Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення. Частина 1. Життєвий цикл програмного забезпечення. Київ : КПІ ім. Ігоря Сікорського, 2022. - 270 с.

29. Ingeno J. Software Architect's Handbook : Become a Successful Software Architect by Implementing Effective Architecture Concepts / Birmingham: Packt Publishing, 2018. - 594p.

30. Koç H, Erdoğan AM, Barjakly Y, Peker S. UML Diagrams in Software Engineering Research: A Systematic Literature Review / Proceedings, 74(1):13, 2021.

[Електронний ресурс]. URL: https://doi.org/10.3390/proceedings2021074013

31. Unhelkar B. Software Engineering with UML (1st ed.) / Auerbach Publications, 2017 - 426p.

32. Mall R. Fundamentals of Software Engineering. Fifth ed. / PHI Learning Private Limited, 2018 - 590p.

33. Gerhard Greeff, Ranjan Ghoshal. Practical E-Manufacturing and Supply Chain Management / Newnes, 2004, - 461p.

34. Azaredo S, Pires FP. Assessing the Advantages of CASE Tools in Software Engineering.Int J Adv Innovat Thoughts Ideas, 12(4):227, 2023. [Електронний ресурс]. URL: https://www.omicsonline.org/open-access/assessing-the-advantages-of-case-tools-in-software-engineering-126472.html

35. Visual Paradigm. [Електронний ресурс]. URL: https://www.visual-paradigm.com/shop/vp.jsp

36. Г. В. Луценко. Програмне середовище Visual Paradigm у навчанні технологій проєктної діяльності студентів-інженерів / Інформаційні технології і засоби навчання, 83(3), с. 208-225, 2021. [Електронний ресурс]. URL: https://dx.doi.org/10.33407/itlt.v83i3.3400

37. Javascript Documentation. [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript

38. React Native Documentation. [Електронний ресурс]. URL: https://reactnative.dev/

39. Dart Documentation. [Електронний ресурс]. URL: https://dart.dev/overview

40. Flutter Documentation. [Електронний ресурс]. URL: https://docs.flutter.dev/resources/faq

41. Kotlin Documentation. [Електронний ресурс]. URL: https://kotlinlang.org/docs

42. Kotlin Multiplatform. [Електронний ресурс]. URL: https://kotlinlang.org/docs/multiplatform.html

43. Flanagan D. Javascript : The Definitive Guide. Seventh ed. / O'Reilly

Media, 2020 - 704p.

44. Eisenman B. Learning React Native: Building Native Mobile Apps with JavaScript. Second ed. / O'Reilly Media, 2017 - 240p.

45. Boduch A, Derks R, Sakhniuk M. React and React Native : Build Cross-Platform Javascript Applications with Native Power for the Web Desktop and Mobile. Fourth ed. / Birmingham UK: Packt Publishing, 2022 - 606p.

46. Webpack Documentation. [Електронний ресурс]. URL: https://webpack.js.org/concepts/

47. ESLint Documentation. [Електронний ресурс]. URL: https://eslint.org/docs/latest/

48. Firebase Documentation. [Електронний ресурс]. URL: https://firebase.google.com/docs/build

49. Jest Documentation. [Електронний ресурс]. URL: https://jestjs.io/

50. React Native Testing Library Documentation. [Електронний ресурс]. URL: https://testing-library.com/docs/react-native-testing-library/intro/

51. Aniche Maurício. Effective Software Testing : A Developer's Guide. First ed. / Manning Publications, 2022 - 328p.

ДОДАТКИ

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. Романюк О. Н.

"19" вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ д-р техн. наук, проф. кафедри ПЗ Ліщинська Л.Б.

"19" вересня 2023 р.

Виконав:

_____ студент гр.2ПІ-22м Пінчуков О. М.

"19" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)».

Галузь застосування - цифрова медицина.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності оперативного відстежування розвитку пандемій, таких як Covid-19. Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19) може підвищити обізнаність громадськості про поточний стан пандемії та заохочувати захисну поведінку, таку як соціальне дистанціювання, носіння масок та вакцинація. Водночас, це допомагає зрозуміти, як вірус впливає на різні країни та регіони, проливаючи світло на відмінності та інформуючи зусилля міжнародної співпраці.

Призначення роботи – розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19).

3. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. WHO. [Електронний ресурс]. URL: https://covid19.who.int/
2. Richards M, Ford N. Fundamentals of Software Architecture : An Engineering Approach. First ed. / O’Reilly Media, 2020 - 422p.
3. Flanagan D. Javascript : The Definitive Guide. Seventh ed. / O’Reilly Media,

2020 - 704p.

4. Eisenman B. Learning React Native: Building Native Mobile Apps with JavaScript. Second ed. / O'Reilly Media, 2017 - 240p.
5. Aniche Maurício. Effective Software Testing : A Developer's Guide. First ed. / Manning Publications, 2022 - 328p.

4. Технічні вимоги

Операційна система macOS; середовище розробки – Visual Studio Code, формат зберігання конфігурації – JSON; мова програмування – JavaScript; фреймворк для розробки мобільних додатків React Native; симулятори мобільних пристроїв XCode та Android Studio; транспілятор Babel; інструмент статичного аналізу коду ESLint; CASE засіб Visual Paradigm; збірник модулів Webpack; розподілена система управління версіями – Git.

5. Конструктивні вимоги.

Програмна система повинна бути безпечна, надійна, продуктивна і зручніста у використанні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

–

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз методів і засобів відстеження та інформування про глобальні пандемії	19.09.2023 - 25.09.2023
2	Моделювання відстеження та інформування про глобальні пандемії та проектування програмної системи	26.09.2023 - 05.10.2023
3	Розробка програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)	06.10.2023 - 02.11.2023
4	Тестування програмного засобу	03.11.2023 - 10.11.2023
5	Економічна частина	11.11.2023 - 24.11.2023

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б. Протокол перевірки навчальної (кваліфікаційної) роботи

Назва роботи: "Розробка методу та програмного засобу для відстеження та інформування про глобальні пандемії (Covid-19)"

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 22м

Науковий керівник: д.т.н, професор каф. ПЗ, Ліщинська Л. Б.

Unicheck	
Оригінальність	**95,5 %**
Схожість	4,5 %

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.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Пінчуков О. М.

Керівник роботи _____ Ліщинська Л. Б.

Додаток В. Лістинг коду

Лістинг файлу App.js

```
import React, { Component, createContext } from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createStackNavigator } from "@react-navigation/stack";
import { onAuthStateChanged } from "firebase/auth";
import { auth } from "./utils/firebaseConfig";
import AuthorizedApp from "./components/AuthorizedApp";
import UnauthorizedApp from "./components/UnauthorizedApp";
import LoginScreen from "./screens/LoginScreen";
import SignupScreen from "./screens/SignupScreen";

const Stack = createStackNavigator();
const AuthenticatedUserContext = createContext({});

class AuthStack extends Component {
  render() {
    return (
      <Stack.Navigator
        initialRouteName="Login"
        screenOptions={{ headerShown: false }}
      >
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="SignUp" component={SignupScreen} />
        <Stack.Screen name="UnauthorizedApp" component={UnauthorizedApp} />
      </Stack.Navigator>
    );
  }
}

class RootNavigator extends Component {
  constructor(props) {
    super(props);
    this.state = {
      user: null,
      setUser: (user) => this.setState({ user: user }),
    };
  }

  componentDidMount() {
    this.unsubscribeAuth = onAuthStateChanged(
      auth,
      async (authenticatedUser) => {
```

```
      authenticatedUser
        ? this.state.setUser(authenticatedUser)
        : this.state.setUser(null);
    }
   );
  }

  componentWillUnmount() {
   this.unsubscribeAuth();
  }

  render() {
   return (
    <NavigationContainer>
     {this.state.user ? <AuthorizedApp /> : <AuthStack />}
    </NavigationContainer>
   );
  }
}

class App extends Component {
  AuthenticatedUserProvider = ({ children }) => (
   <AuthenticatedUserContext.Provider value={this.state}>
    {children}
   </AuthenticatedUserContext.Provider>
  );

  render() {
   return (
    <this.AuthenticatedUserProvider>
     <RootNavigator />
    </this.AuthenticatedUserProvider>
   );
  }
}

export default App;
```

Лістинг файлу LoginScreen.js

```
import React, { Component } from "react";
import {
  StyleSheet,
  Text,
  View,
  TextInput,
  SafeAreaView,
```

```
  TouchableOpacity,
  StatusBar,
  Alert,
} from "react-native";
import { signInWithEmailAndPassword } from "firebase/auth";
import { auth } from "../utils/firebaseConfig";

class Login extends Component {
  constructor(props) {
    super(props);

    this.state = {
      email: "",
      password: "",
    };
  }

  onHandleLogin = () => {
    const { email, password } = this.state;
    if (email !== "" && password !== "") {
      signInWithEmailAndPassword(auth, email, password)
        .then(() => console.log("Login success"))
        .catch((err) => Alert.alert("Login error", err.message));
    }
  };

  render() {
    const { navigation } = this.props;
    const { email, password } = this.state;

    return (
      <View style={styles.container}>
        <View style={styles.whiteSheet} />
        <SafeAreaView style={styles.form}>
          <Text style={styles.title}>Log In</Text>
          <TextInput
            style={styles.input}
            placeholder="Enter email"
            autoCapitalize="none"
            keyboardType="email-address"
            textContentType="emailAddress"
            autoFocus={true}
            value={email}
            onChangeText={(text) => this.setState({ email: text })}
          />
          <TextInput
```

```
  style={styles.input}
  placeholder="Enter password"
  autoCapitalize="none"
  autoCorrect={false}
  secureTextEntry={true}
  textContentType="password"
  value={password}
  onChangeText={(text) => this.setState({ password: text })}
/>
<TouchableOpacity style={styles.button} onPress={this.onHandleLogin}>
  <Text style={{ fontWeight: "bold", color: "#fff", fontSize: 18 }}>
    {" "}
    Log In
  </Text>
</TouchableOpacity>
<View
  style={{
    marginTop: 20,
    flexDirection: "row",
    alignItems: "center",
    alignSelf: "center",
  }}
>
  <Text style={{ color: "gray", fontWeight: "600", fontSize: 14 }}>
    Don't have an account?{" "}
  </Text>
  <TouchableOpacity onPress={() => navigation.navigate("SignUp")}>
    <Text
      style={{ color: "#f57c00", fontWeight: "600", fontSize: 14 }}
    >
      {" "}
      Sign Up
    </Text>
  </TouchableOpacity>
</View>
<View
  style={{
    marginTop: 20,
    flexDirection: "row",
    alignItems: "center",
    alignSelf: "center",
  }}
>
  <Text style={{ color: "gray", fontWeight: "600", fontSize: 14 }}>
    You can also proceed as a
  </Text>
```

```
          <TouchableOpacity
            onPress={() => navigation.navigate("UnauthorizedApp")}
          >
            <Text
              style={{ color: "#f57c00", fontWeight: "600", fontSize: 14 }}
            >
              {" "}
              Guest
            </Text>
          </TouchableOpacity>
        </View>
      </SafeAreaView>
      <StatusBar barStyle="light-content" />
    </View>
  );
 }
}

export default Login;

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#fff",
 },
 title: {
  fontSize: 36,
  fontWeight: "bold",
  color: "orange",
  alignSelf: "center",
  paddingBottom: 24,
 },
 input: {
  backgroundColor: "#F6F7FB",
  height: 58,
  marginBottom: 20,
  fontSize: 16,
  borderRadius: 10,
  padding: 12,
 },
 backImage: {
  width: "100%",
  height: 340,
  position: "absolute",
  top: 0,
  resizeMode: "cover",
```

```
  },
  whiteSheet: {
    width: "100%",
    height: "75%",
    position: "absolute",
    bottom: 0,
    backgroundColor: "#fff",
    borderTopLeftRadius: 60,
  },
  form: {
    flex: 1,
    justifyContent: "center",
    marginHorizontal: 30,
  },
  button: {
    backgroundColor: "#f57c00",
    height: 58,
    borderRadius: 10,
    justifyContent: "center",
    alignItems: "center",
    marginTop: 40,
  },
});
```

Лістинг файлу SignupScreen.js

```
import React from "react";
import {
  StyleSheet,
  Text,
  View,
  TextInput,
  SafeAreaView,
  TouchableOpacity,
  StatusBar,
  Alert,
} from "react-native";
import { createUserWithEmailAndPassword } from "firebase/auth";
import { auth } from "../utils/firebaseConfig";

class Signup extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      email: "",
      password: "",
    };
```

```
}

onHandleSignup = () => {
  if (this.state.email !== "" && this.state.password !== "") {
    createUserWithEmailAndPassword(
      auth,
      this.state.email,
      this.state.password
    )
      .then(() => console.log("Signup success"))
      .catch((err) => Alert.alert("Login error", err.message));
  }
};

render() {
  return (
    <View style={styles.container}>
      <View style={styles.whiteSheet} />
      <SafeAreaView style={styles.form}>
        <Text style={styles.title}>Sign Up</Text>
        <TextInput
          style={styles.input}
          placeholder="Enter email"
          autoCapitalize="none"
          keyboardType="email-address"
          textContentType="emailAddress"
          autoFocus={true}
          value={this.state.email}
          onChangeText={(text) => this.setState({ email: text })}
        />
        <TextInput
          style={styles.input}
          placeholder="Enter password"
          autoCapitalize="none"
          autoCorrect={false}
          secureTextEntry={true}
          textContentType="password"
          value={this.state.password}
          onChangeText={(text) => this.setState({ password: text })}
        />
        <TouchableOpacity style={styles.button} onPress={this.onHandleSignup}>
          <Text style={{ fontWeight: "bold", color: "#fff", fontSize: 18 }}>
            {" "}
            Sign Up
          </Text>
        </TouchableOpacity>
```

```
        <View
          style={{
            marginTop: 20,
            flexDirection: "row",
            alignItems: "center",
            alignSelf: "center",
          }}
        >
          <Text style={{ color: "gray", fontWeight: "600", fontSize: 14 }}>
            Don't have an account?{" "}
          </Text>
          <TouchableOpacity
            onPress={() => this.props.navigation.navigate("Login")}
          >
            <Text
              style={{ color: "#f57c00", fontWeight: "600", fontSize: 14 }}
            >
              {" "}
              Log In
            </Text>
          </TouchableOpacity>
        </View>
      </SafeAreaView>
      <StatusBar barStyle="light-content" />
    </View>
  );
 }
}

export default Signup;

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#fff",
 },
 title: {
  fontSize: 36,
  fontWeight: "bold",
  color: "orange",
  alignSelf: "center",
  paddingBottom: 24,
 },
 input: {
  backgroundColor: "#F6F7FB",
  height: 58,
```

```
    marginBottom: 20,
    fontSize: 16,
    borderRadius: 10,
    padding: 12,
  },
  backImage: {
    width: "100%",
    height: 340,
    position: "absolute",
    top: 0,
    resizeMode: "cover",
  },
  whiteSheet: {
    width: "100%",
    height: "75%",
    position: "absolute",
    bottom: 0,
    backgroundColor: "#fff",
    borderTopLeftRadius: 60,
  },
  form: {
    flex: 1,
    justifyContent: "center",
    marginHorizontal: 30,
  },
  button: {
    backgroundColor: "#f57c00",
    height: 58,
    borderRadius: 10,
    justifyContent: "center",
    alignItems: "center",
    marginTop: 40,
  },
});
```

Лістинг файлу MapScreen.js

```
import React from "react";
import MapView, {
  Marker,
  Callout,
  Geojson,
  Heatmap,
  PROVIDER_GOOGLE,
} from "react-native-maps";
import axios from "axios";
```

```
const { kmeans } = require("ml-kmeans");
import * as jsonData from "../assets/geoJson.json";
import usaStates from "../assets/usaStates";
import {
  SafeAreaView,
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
  ActivityIndicator,
} from "react-native";
import Icon from "react-native-vector-icons/Ionicons";
import CountryFlag from "react-native-country-flag";

const options = {
  method: "GET",
  url: "https://disease.sh/v3/Covid-19/countries",
};
const optionsUSA = {
  method: "GET",
  url: "https://disease.sh/v3/Covid-19/states",
};

class MapScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      data: [],
      dataUSA: [],
      region: {
        latitude: 48.3794,
        longitude: 31.1656,
        latitudeDelta: 20.0922,
        longitudeDelta: 20.0421,
      },
      isLoading: true,
      showClusters: false,
      showHeatmap: false,
      clusterizedData: null,
      heatmapData: null,
    };
  }

  componentDidMount() {
    this.fetchData();
    this.fetchDataUSA();
```

```
}

calculatePercentile = (data, percentile) => {
  data.sort((a, b) => a - b);
  const index = (percentile / 100) * (data.length - 1);
  if (Math.floor(index) === index) {
    return data[index];
  }
  const lower = data[Math.floor(index)];
  const upper = data[Math.ceil(index)];
  return lower + (upper - lower) * (index - Math.floor(index));
};

getClusterizedCountries = () => {
  const data = this.state.data;
  if (data.length) {
    let clusterColors = {
      0: "#00FF00",
      1: "#FFFF00",
      2: "#FF0000",
    };
    const modifiedJson = { ...jsonData };
    modifiedJson.features.forEach((feature) => {
      let countryIso = feature.properties.iso_a2;
      let cluster = data.find(
        (i) => i.countryInfo?.iso2 === countryIso
      )?.cluster;
      if (cluster !== undefined) {
        feature.properties.fill = clusterColors[cluster];
        feature.properties.cluster = cluster;
      }
    });
    this.setState({ clusterizedData: modifiedJson });
  }
};
getHeatMap = () => {
  const data = this.state.dataUSA;
  if (data.length) {
    const copiedUSData = [...usaStates];
    const filtered = copiedUSData.map((item1) => {
      const item2 = data.find((item2) => item2.state === item1.state);
      if (item2) {
        return { ...item1, ...item2 };
      }
    });
    const heatmapData = filtered.map((i) => ({
```

```
      latitude: i.latitude,
      longitude: i.longitude,
      weight: i.casesPerOneMillion,
    }));

    this.setState({
      heatmapData,
    });
  }
};

fetchData = async () => {
  try {
    const response = await axios.request(options);
    const deathsPerOneMillion = response.data.map((i) => [
      i.deathsPerOneMillion,
    ]);
    const flatedArr = deathsPerOneMillion.flat(Infinity);
    const lowCentroid = this.calculatePercentile(flatedArr, 25);
    const mediumCentroid = this.calculatePercentile(flatedArr, 50);
    const highCentroid = this.calculatePercentile(flatedArr, 75);
    let initialCentroids = [[lowCentroid], [mediumCentroid], [highCentroid]];
    const clusters = kmeans(deathsPerOneMillion, 3, {
      initialization: initialCentroids,
    });

    for (let i = 0; i < response.data.length; i++) {
      response.data[i].cluster = clusters.clusters[i];
    }
    const data = response.data.map((i) => ({
      country: i.country,
      countryInfo: i.countryInfo,
      cases: i.cases,
      deaths: i.deaths,
      recovered: i.recovered,
      todayCases: i.todayCases,
      cluster: i.cluster,
      casesPerOneMillion: i.casesPerOneMillion,
    }));
    this.setState({ data, isLoading: false }, () => {
      this.getClusterizedCountries();
    });
  } catch (error) {
    console.error(error);
    this.setState({ isLoading: false });
  }
```

```
};
fetchDataUSA = async () => {
  try {
    const response = await axios.request(optionsUSA);
    const dataUSA = response.data.map((i) => ({
      state: i.state,
      casesPerOneMillion: i.casesPerOneMillion,
    }));
    this.setState({ dataUSA, isLoading: false }, () => {
      this.getHeatMap();
    });
  } catch (error) {
    console.error(error);
    this.setState({ isLoading: false });
  }
};

renderMarkers = () => {
  return this.state.data.map(
    (
      { country, countryInfo, cases, deaths, recovered, todayCases },
      index
    ) => {
      return (
        <Marker
          testID={`marker-${index}`}
          key={index}
          coordinate={{
            latitude: countryInfo.lat,
            longitude: countryInfo.long,
          }}
        >
          <Callout tooltip style={styles.tooltip}>
            <View>
              {countryInfo.iso2 ? (
                <CountryFlag isoCode={countryInfo?.iso2} size={25} />
              ) : (
                <CountryFlag isoCode="de" size={25} />
              )}
              <Text style={styles.title}>{country}</Text>
              <Text style={styles.text}>Today Cases: {todayCases}</Text>
              <Text style={styles.text}>Cases: {cases}</Text>
              <Text style={styles.text}>Deaths: {deaths}</Text>
              <Text style={styles.text}>Recovered: {recovered}</Text>
            </View>
          </Callout>
```

```
        </Marker>
      );
    }
  );
};

zoomIn = () => {
  this.setState((prevState) => ({
    region: {
      ...prevState.region,
      latitudeDelta: prevState.region.latitudeDelta / 2,
      longitudeDelta: prevState.region.longitudeDelta / 2,
    },
  }));
};

zoomOut = () => {
  this.setState((prevState) => ({
    region: {
      ...prevState.region,
      latitudeDelta: prevState.region.latitudeDelta * 2,
      longitudeDelta: prevState.region.longitudeDelta * 2,
    },
  }));
};

render() {
  if (this.state.isLoading) {
    return (
      <SafeAreaView style={{ flex: 1, justifyContent: "center" }}>
        <ActivityIndicator
          testID="activity-indicator"
          size="large"
          color="#0000ff"
        />
      </SafeAreaView>
    );
  }
  return (
    <SafeAreaView style={{ flex: 1 }}>
      <MapView
        testID="map-view"
        style={{ flex: 1 }}
        region={this.state.region}
        provider={PROVIDER_GOOGLE}
      >
```

```
  {this.state.showClusters && !this.state.showHeatmap ? (
    <Geojson geojson={this.state.clusterizedData} />
  ) : !this.state.showClusters && this.state.showHeatmap ? (
    <Heatmap points={this.state.heatmapData} radius={60} />
  ) : (
    this.renderMarkers()
  )}
</MapView>
<View style={styles.buttons}>
  <TouchableOpacity testID="zoom-in-button" onPress={this.zoomIn}>
    <Icon name="add-circle-outline" size={30} color="#000" />
  </TouchableOpacity>
  <TouchableOpacity testID="zoom-out-button" onPress={this.zoomOut}>
    <Icon name="remove-circle-outline" size={30} color="#000" />
  </TouchableOpacity>
</View>
{this.state.clusterizedData ? (
  <View style={styles.clusterButton}>
    <TouchableOpacity
      onPress={() =>
        this.setState({
          showClusters: !this.state.showClusters,
          showHeatmap: false,
        })
      }
    >
      <Text style={styles.clusterText}>
        {this.state.showClusters ? "Hide" : "Show"} Clusterization
      </Text>
    </TouchableOpacity>
  </View>
) : (
  false
)}
{this.state.heatmapData ? (
  <View style={styles.heatmapButton}>
    <TouchableOpacity
      onPress={() =>
        this.setState({
          showHeatmap: !this.state.showHeatmap,
          showClusters: false,
          region: {
            latitude: 37.0902,
            longitude: -95.7129,
            latitudeDelta: 20,
            longitudeDelta: 20,
```

```
              },
            })
          }
        >
          <Text style={styles.heatmapText}>
            {this.state.showHeatmap ? "Hide" : "Show"} KDE Heatmap
          </Text>
        </TouchableOpacity>
      </View>
    ) : (
      false
    )}
  </SafeAreaView>
 );
 }
}

const styles = StyleSheet.create({
 buttons: {
  flexDirection: "row",
  justifyContent: "center",
  position: "absolute",
  bottom: 20,
  left: 0,
  right: 0,
 },
 tooltip: {
  padding: 10,
  borderRadius: 5,
  backgroundColor: "lightblue",
 },
 title: {
  fontWeight: "bold",
  fontSize: 16,
  marginTop: 5,
 },
 text: {
  marginTop: 5,
  fontSize: 14,
 },
 clusterButton: {
  flexDirection: "row",
  justifyContent: "center",
  position: "absolute",
  top: 70,
  right: 10,
```

```
    backgroundColor: "blue",
    padding: 10,
    borderRadius: "50%",
  },
  clusterText: {
    color: "white",
    fontWeight: "bold",
  },
  heatmapButton: {
    flexDirection: "row",
    justifyContent: "center",
    position: "absolute",
    top: 70,
    left: 10,
    backgroundColor: "blue",
    padding: 10,
    borderRadius: "50%",
  },
  heatmapText: {
    color: "white",
    fontWeight: "bold",
  },
});

export default MapScreen;
```

Лістинг файлу NewsScreen.js

```
import React from "react";
import {
  FlatList,
  Linking,
  Image,
  TouchableOpacity,
  SafeAreaView,
  ActivityIndicator,
} from "react-native";
import { List, Divider } from "react-native-paper";
import axios from "axios";

const options = {
  method: "GET",
  url: "https://Covid-19-news.p.rapidapi.com/v1/covid",
  params: {
    q: "covid",
    lang: "en",
    media: "True",
```

```
  },
  headers: {
    "X-RapidAPI-Key": "06b9cc1c5fmsh9da49ba1957eb08p17b9b0jsn1e3a76f22f0b",
    "X-RapidAPI-Host": "Covid-19-news.p.rapidapi.com",
  },
};

class NewsScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      news: [],
      isLoading: true,
    };
  }

  componentDidMount() {
    this.fetchData();
  }

  fetchData = async () => {
    try {
      const response = await axios.request(options);
      const data = response.data.articles.map((i) => {
        return {
          link: i.link,
          media: i.media,
          title: i.title,
          summary: i.summary,
        };
      });
      this.setState({ news: data, isLoading: false });
    } catch (error) {
      console.error(error);
      this.setState({ isLoading: false });
    }
  };

  renderItem = ({ item, index }) => (
    <TouchableOpacity onPress={() => Linking.openURL(item.link)}>
      <List.Item
        testID="news-item"
        key={index}
        title={item.title}
        titleStyle={{ fontWeight: "bold" }}
        description={item.summary}
```

```
        left={(props) => (
          <Image
            {...props}
            source={{ uri: item.media }}
            style={{ width: 100, height: 100 }}
          />
        )}
      />
    </TouchableOpacity>
  );

  render() {
    if (this.state.isLoading) {
      return (
        <SafeAreaView style={{ flex: 1, justifyContent: "center" }}>
          <ActivityIndicator
            testID="activity-indicator"
            size="large"
            color="#0000ff"
          />
        </SafeAreaView>
      );
    }

    return (
      <SafeAreaView style={{ flex: 1 }}>
        <FlatList
          data={this.state.news}
          renderItem={this.renderItem}
          keyExtractor={(item) => item.link}
          ItemSeparatorComponent={Divider}
        />
      </SafeAreaView>
    );
  }
}

export default NewsScreen;
```

Лістинг файлу RatingScreen.js

```
import React from "react";
import { Picker } from "@react-native-picker/picker";
import {
  SafeAreaView,
  View,
  Text,
```

```
  FlatList,
  Image,
  StyleSheet,
  ActivityIndicator,
} from "react-native";
import axios from "axios";
import CountryFlag from "react-native-country-flag";

const options = {
  method: "GET",
  url: "https://disease.sh/v3/Covid-19/countries",
};

class RatingScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      data: [],
      filter: "cases",
      isLoading: true,
    };
  }

  componentDidMount() {
    this.fetchData();
  }

  fetchData = async () => {
    try {
      const response = await axios.request(options);
      const data = response.data.map((i) => {
        return {
          country: i.country,
          iso2: i.countryInfo.iso2,
          cases: i.cases,
          deaths: i.deaths,
          recovered: i.recovered,
          todayCases: i.todayCases,
        };
      });

      this.setState({ data, isLoading: false });
    } catch (error) {
      this.setState({ isLoading: false });
    }
  };
```

```
filterData = (filter) => {
  if (this.state.data.length) {
    const filteredData = [...this.state.data];
    const filteredArr = filteredData.map((i) => {
      const randomNumber = Math.floor(Math.random() * 200) + 1;
      const refTodaysCases = i.todayCases
        ? i.todayCases
        : i.todayCases + randomNumber;
      i.todayCases = refTodaysCases;
      return i;
    });

    filteredArr.sort((a, b) => b[filter] - a[filter]);
    return filteredArr;
  }
};

renderItem = ({ item }) => (
  <View style={styles.itemContainer}>
    {item.iso2 ? (
      <CountryFlag isoCode={item?.iso2} size={45} />
    ) : (
      <CountryFlag isoCode="de" size={25} />
    )}
    <Text testID="country-name" style={styles.countryName}>
      {item.country}
    </Text>
    <Text style={styles.casesNumber}>{item[this.state.filter]}</Text>
  </View>
);

render() {
  if (this.state.isLoading) {
    return (
      <SafeAreaView style={{ flex: 1, justifyContent: "center" }}>
        <ActivityIndicator
          testID="activity-indicator"
          size="large"
          color="#0000ff"
        />
      </SafeAreaView>
    );
  }
  return (
    <SafeAreaView style={{ flex: 1 }}>
```

```
        <Picker
          testID="picker"
          selectedValue={this.state.filter}
          onValueChange={(itemValue) => this.setState({ filter: itemValue })}
        >
          <Picker.Item label="Today Cases" value="todayCases" />
          <Picker.Item label="All-time Cases" value="cases" />
          <Picker.Item label="Deaths" value="deaths" />
          <Picker.Item label="Recovered" value="recovered" />
        </Picker>
        <FlatList
          data={this.filterData(this.state.filter)}
          renderItem={this.renderItem}
          keyExtractor={(item) => item.country}
        />
      </SafeAreaView>
    );
  }
}

export default RatingScreen;

const styles = StyleSheet.create({
  itemContainer: {
    flexDirection: "row",
    justifyContent: "space-around",
    alignItems: "center",
    padding: 10,
    borderBottomWidth: 1,
    borderBottomColor: "#ccc",
  },
  countryName: {
    flex: 1,
    textAlign: "center",
  },
  casesNumber: {
    flex: 1,
    textAlign: "center",
  },
});
```

Лістинг файлу SettingsScreen.js

```
import React from "react";
import {
  SafeAreaView,
  StyleSheet,
```

```
  Text,
  View,
  TextInput,
  TouchableOpacity,
  Alert,
} from "react-native";
import {
  reauthenticateWithCredential,
  EmailAuthProvider,
  updateEmail,
  updatePassword,
  signOut,
} from "firebase/auth";
import { auth } from "../utils/firebaseConfig";

class SettingsScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      oldEmail: "",
      oldPassword: "",
      newEmail: "",
      newPassword: "",
      showEmailInput: false,
      showPasswordInput: false,
    };
  }

  updateEmailAndPassword = () => {
    const user = auth.currentUser;
    const credential = EmailAuthProvider.credential(
      this.state.oldEmail,
      this.state.oldPassword
    );

    reauthenticateWithCredential(user, credential)
      .then(() => {
        if (this.state.newEmail !== "") {
          updateEmail(user, this.state.newEmail)
            .then(() => {
              Alert.alert("Success", "Email updated");
              this.setState({
                newEmail: "",
                oldEmail: "",
                oldPassword: "",
                showEmailInput: false,
```

```
          });
        })
        .catch((error) => {
          Alert.alert("Failed to update email", error.message);
        });
    }

    if (this.state.newPassword !== "") {
      updatePassword(user, this.state.newPassword)
        .then(() => {
          Alert.alert("Success", "Password updated");
          this.setState({
            newPassword: "",
            oldEmail: "",
            oldPassword: "",
            showPasswordInput: false,
          });
        })
        .catch((error) => {
          Alert.alert("Failed to update password", error.message);
        });
    }
  })
  .catch((error) => {
    console.log(error);
    Alert.alert("Authentication failed", "Please check your credentials");
  });
};

logout = () => {
  signOut(auth)
    .then(() => {
      Alert.alert("Logged out", "You have been logged out successfully");
    })
    .catch((error) => {
      console.log(error);
    });
};

render() {
  return (
    <SafeAreaView style={styles.container}>
      <TouchableOpacity
        style={styles.button}
        onPress={() =>
          this.setState({ showEmailInput: !this.state.showEmailInput })
```

```
  }
>
  <Text style={styles.buttonText}>Change Email</Text>
</TouchableOpacity>
{this.state.showEmailInput && (
  <View>
    <TextInput
      style={styles.input}
      placeholder="Enter old email"
      keyboardType="email-address"
      value={this.state.oldEmail}
      onChangeText={(text) => this.setState({ oldEmail: text })}
    />
    <TextInput
      style={styles.input}
      placeholder="Enter old password"
      secureTextEntry={true}
      value={this.state.oldPassword}
      onChangeText={(text) => this.setState({ oldPassword: text })}
    />
    <TextInput
      style={styles.input}
      placeholder="Enter new email"
      keyboardType="email-address"
      value={this.state.newEmail}
      onChangeText={(text) => this.setState({ newEmail: text })}
    />
    <TouchableOpacity
      style={styles.confirmButton}
      onPress={this.updateEmailAndPassword}
    >
      <Text style={styles.confirmButtonText}>Confirm</Text>
    </TouchableOpacity>
  </View>
)}
<TouchableOpacity
  style={styles.button}
  onPress={() =>
    this.setState({ showPasswordInput: !this.state.showPasswordInput })
  }
>
  <Text style={styles.buttonText}>Change Password</Text>
</TouchableOpacity>
{this.state.showPasswordInput && (
  <View>
    <TextInput
```

```
            style={styles.input}
            placeholder="Enter old email"
            keyboardType="email-address"
            value={this.state.oldEmail}
            onChangeText={(text) => this.setState({ oldEmail: text })}
          />
          <TextInput
            style={styles.input}
            placeholder="Enter old password"
            secureTextEntry={true}
            value={this.state.oldPassword}
            onChangeText={(text) => this.setState({ oldPassword: text })}
          />
          <TextInput
            style={styles.input}
            placeholder="Enter new password"
            secureTextEntry={true}
            value={this.state.newPassword}
            onChangeText={(text) => this.setState({ newPassword: text })}
          />
          <TouchableOpacity
            style={styles.confirmButton}
            onPress={this.updateEmailAndPassword}
          >
            <Text style={styles.confirmButtonText}>Confirm</Text>
          </TouchableOpacity>
        </View>
      )}
      <TouchableOpacity style={styles.logoutButton} onPress={this.logout}>
        <Text style={styles.logoutButtonText}>Log out</Text>
      </TouchableOpacity>
    </SafeAreaView>
  );
 }
}

const styles = StyleSheet.create({
 container: {
  flex: 1,
  justifyContent: "center",
  padding: 40,
 },
 input: {
  height: 40,
  borderColor: "gray",
  borderWidth: 1,
```

```
    marginBottom: 10,
    paddingLeft: 10,
    borderRadius: 5,
  },
  button: {
    backgroundColor: "grey",
    padding: 10,
    marginBottom: 10,
    borderRadius: 5,
  },
  buttonText: {
    color: "white",
    textAlign: "center",
  },
  confirmButton: {
    backgroundColor: "green",
    padding: 10,
    marginBottom: 10,
    borderRadius: 5,
  },
  confirmButtonText: {
    color: "white",
    textAlign: "center",
  },
  logoutButton: {
    backgroundColor: "red",
    padding: 10,
    marginTop: 10,
    borderRadius: 5,
  },
  logoutButtonText: {
    color: "white",
    textAlign: "center",
  },
});

export default SettingsScreen;
```

Лістинг файлу VaccinationScreen.js

```
import React, { useEffect } from "react";
import {
  SafeAreaView,
  View,
  Text,
  TouchableOpacity,
  StyleSheet,
```

```
  Button,
} from "react-native";
import DateTimePicker from "@react-native-community/datetimepicker";
import * as Notifications from "expo-notifications";

class VaccinationScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      vaccineType: null,
      firstDoseDate: null,
      secondDoseDate: null,
      revaccinationDate: null,
      showDatePicker: false,
      isUserAgreedForNotifications: false,
    };
  }

  handleConfirm = (event, selectedDate) => {
    const currentDate = selectedDate || this.state.firstDoseDate;
    this.setState({ firstDoseDate: currentDate, showDatePicker: false });

    if (this.state.vaccineType) {
      let secondDoseDate;
      switch (this.state.vaccineType) {
        case "Pfizer":
          secondDoseDate = new Date(
            currentDate.getTime() + 21 * 24 * 60 * 60 * 1000
          );
          break;
        case "Moderna":
          secondDoseDate = new Date(
            currentDate.getTime() + 28 * 24 * 60 * 60 * 1000
          );
          break;
        case "AstraZeneca":
          secondDoseDate = new Date(
            currentDate.getTime() + 12 * 7 * 24 * 60 * 60 * 1000
          );
          break;
      }
      this.setState({ secondDoseDate: secondDoseDate });
      let revaccinationDate = new Date(
        secondDoseDate.getTime() + 365 * 24 * 60 * 60 * 1000
      );
      this.setState({ revaccinationDate: revaccinationDate });
```

```
  }
};

handleNotifications = () => {
  const { secondDoseDate, revaccinationDate } = this.state;
  this.schedulePushNotification(
    secondDoseDate,
    `It's time for your second dose of ${this.state.vaccineType} vaccine!`
  );
  this.schedulePushNotification(
    revaccinationDate,
    `It's time for your revaccination with ${this.state.vaccineType} vaccine!`
  );
};

componentDidUpdate(prevProps, prevState) {
  if (prevState.vaccineType !== this.state.vaccineType) {
    this.setState({
      secondDoseDate: null,
      revaccinationDate: null,
      showDatePicker: false,
      isUserAgreedForNotifications: false,
      firstDoseDate: null,
    });
  }
}

schedulePushNotification = async (date, message) => {
  await Notifications.scheduleNotificationAsync({
    content: {
      title: "Vaccine Reminder",
      body: message,
    },
    trigger: new Date(date.getTime() - 24 * 60 * 60 * 1000),
  });
};

render() {
  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.content}>
        <Text style={styles.headline}>Please choose you vaccine type:</Text>
        <View style={styles.buttonContainer}>
          <TouchableOpacity
            style={[
              styles.button,
```

```
      {
        backgroundColor:
          this.state.vaccineType === "Pfizer" ? "#007BFF" : "#6c757d",
      },
    ]}
    onPress={() => this.setState({ vaccineType: "Pfizer" })}
  >
    <Text style={styles.buttonText}>Pfizer</Text>
  </TouchableOpacity>
  <TouchableOpacity
    style={[
      styles.button,
      {
        backgroundColor:
          this.state.vaccineType === "Moderna"
            ? "#28a745"
            : "#6c757d",
      },
    ]}
    onPress={() => this.setState({ vaccineType: "Moderna" })}
  >
    <Text style={styles.buttonText}>Moderna</Text>
  </TouchableOpacity>
  <TouchableOpacity
    style={[
      styles.button,
      {
        backgroundColor:
          this.state.vaccineType === "AstraZeneca"
            ? "#ffc107"
            : "#6c757d",
      },
    ]}
    onPress={() => this.setState({ vaccineType: "AstraZeneca" })}
  >
    <Text style={styles.buttonText}>AstraZeneca</Text>
  </TouchableOpacity>
</View>
{this.state.vaccineType && (
  <Button
    title="Select First Shot Date"
    onPress={() => this.setState({ showDatePicker: true })}
  />
)}
{this.state.showDatePicker && (
  <DateTimePicker
```

```
          style={styles.datePicker}
          testID="dateTimePicker"
          value={this.state.firstDoseDate || new Date()}
          mode={"date"}
          display="default"
          onChange={this.handleConfirm}
        />
      )}

      {this.state.firstDoseDate && (
        <Text style={styles.dateText}>
          Your first shot date: {this.state.firstDoseDate.toDateString()}
        </Text>
      )}
      {this.state.secondDoseDate && (
        <Text style={styles.dateTextBold}>
          Your second shot date: {this.state.secondDoseDate.toDateString()}
        </Text>
      )}
      {this.state.revaccinationDate && (
        <Text style={styles.dateTextBold}>
          Your revaccination date:{" "}
          {this.state.revaccinationDate.toDateString()}
        </Text>
      )}
      {this.state.secondDoseDate && this.state.revaccinationDate && (
        <TouchableOpacity
          style={styles.agreeButton}
          onPress={() =>
            this.setState({ isUserAgreedForNotifications: true })
          }
        >
          <Text style={styles.buttonText}>
            Click to subscribe for notifications
          </Text>
        </TouchableOpacity>
      )}
      {this.state.isUserAgreedForNotifications && (
        <Text style={styles.dateText}>
          You will receive notification one day before deadline
        </Text>
      )}
    </View>
  </SafeAreaView>
 );
}
```

```
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
  content: {
    display: "flex",
    alignItems: "center",
    paddingTop: 40,
    paddingHorizontal: 10,
    height: "100%",
  },
  title: {
    fontSize: 24,
    fontWeight: "bold",
    marginBottom: 20,
  },
  buttonContainer: {
    flexDirection: "row",
    justifyContent: "space-around",
    width: "100%",
    marginTop: 20,
    marginBottom: 20,
  },
  button: {
    flex: 1,
    alignItems: "center",
    padding: 10,
    borderRadius: 50,
    margin: 5,
  },
  agreeButton: {
    alignItems: "center",
    padding: 20,
    borderRadius: 50,
    margin: 10,
    backgroundColor: "#28a745",
  },
  buttonText: {
    color: "white",
    fontWeight: "bold",
  },
  datePicker: {
    margin: 20,
    fontWeight: "bold",
```

```
    fontSize: 20,
  },
  dateText: {
    margin: 10,
    fontSize: 18,
    textAlign: "center",
  },
  dateTextBold: {
    margin: 10,
    fontSize: 20,
    textAlign: "center",
    fontWeight: "bold",
  },
  headline: {
    margin: 10,
    fontWeight: "bold",
    fontSize: 20,
  },
});

export default VaccinationScreen;
```

Лістинг файлу MapScreen.test..js

```
import React from "react";
import { render, fireEvent, waitFor, act } from "@testing-library/react-native";
import MapScreen from "../screens/MapScreen";
import axios from "axios";

jest.mock("axios");

const mockMapData = [
  {
    country: "Country1",
    countryInfo: { _id: 1, lat: 10, long: 10, iso2: "CT1" },
    cases: 1000,
    deaths: 50,
    recovered: 500,
    todayCases: 10,
  },
  {
    country: "Country2",
    countryInfo: { _id: 2, lat: 20, long: 20, iso2: "CT2" },
    cases: 2000,
    deaths: 100,
    recovered: 1000,
    todayCases: 20,
```

```
  },
];

describe("MapScreen", () => {
  beforeEach(() => {
    axios.request.mockResolvedValue({ data: mockMapData });
  });
  afterEach(() => {
    jest.clearAllMocks();
  });

  it("loads and renders markers after fetching data", async () => {
    const { getByTestId, queryByTestId, getAllByTestId } = render(
      <MapScreen />
    );
    await waitFor(() =>
      expect(queryByTestId("activity-indicator")).toBeFalsy()
    );
    expect(getAllByTestId(/^marker-/).length).toBe(mockMapData.length);
  });

  it("handles zoom in", async () => {
    const { getByTestId, queryByTestId } = render(<MapScreen />);
    await waitFor(() =>
      expect(queryByTestId("activity-indicator")).toBeFalsy()
    );
    act(() => {
      fireEvent.press(getByTestId("zoom-in-button"));
    });
    await waitFor(() => {
      const mapView = getByTestId("map-view");
      expect(mapView.props.region.latitudeDelta).toBeLessThan(20.0922);
      expect(mapView.props.region.longitudeDelta).toBeLessThan(20.0421);
    });
  });

  it("handles zoom out", async () => {
    const { getByTestId, queryByTestId } = render(<MapScreen />);
    await waitFor(() =>
      expect(queryByTestId("activity-indicator")).toBeFalsy()
    );
    act(() => {
      fireEvent.press(getByTestId("zoom-out-button"));
    });
    await waitFor(() => {
      const mapView = getByTestId("map-view");
```

```
      expect(mapView.props.region.latitudeDelta).toBeGreaterThan(20.0922);
      expect(mapView.props.region.longitudeDelta).toBeGreaterThan(20.0421);
    });
  });
});
```

Лістинг файлу RatingScreen.test..js

```
import React from "react";
import { render, waitFor, fireEvent, act } from "@testing-library/react-native";
import RatingScreen from "../screens/RatingScreen";
import axios from "axios";

jest.mock("axios");

const mockRatingData = [
  {
    country: "Country1",
    countryInfo: { iso2: "CT1" },
    cases: 1200,
    deaths: 10,
    recovered: 1100,
    todayCases: 12,
  },
  {
    country: "Country2",
    countryInfo: { iso2: "CT2" },
    cases: 1500,
    deaths: 20,
    recovered: 1400,
    todayCases: 15,
  },
];

describe("RatingScreen", () => {
  beforeEach(() => {
    axios.request.mockResolvedValue({ data: mockRatingData });
  });
  afterEach(() => {
    jest.clearAllMocks();
  });

  it("renders loading indicator initially", () => {
    const { getByTestId } = render(<RatingScreen />);
    expect(getByTestId("activity-indicator")).toBeTruthy();
  });
```

```
  it("renders data after fetching", async () => {
    const { queryByTestId, getByText } = render(<RatingScreen />);
    await waitFor(() =>
      expect(queryByTestId("activity-indicator")).toBeFalsy()
    );
    expect(getByText("Country1")).toBeTruthy();
    expect(getByText("Country2")).toBeTruthy();
  });

  it("sorts data according to the selected filter", async () => {
    const { getByTestId, getAllByTestId, rerender } = render(<RatingScreen />);
    await waitFor(() => expect(axios.request).toHaveBeenCalled());
    act(() => {
      fireEvent(getByTestId("picker"), "onValueChange", "deaths");
    });
    rerender(<RatingScreen />);
    await waitFor(() => {
      const countryTexts = getAllByTestId("country-name").map(
        (t) => t.props.children
      );
      const expectedOrder = mockRatingData
        .sort((a, b) => b.deaths - a.deaths)
        .map((item) => item.country);
      expect(countryTexts).toEqual(expectedOrder);
    });
  });
});
```

Лістинг файлу NewsScreen.test..js

```
import React from "react";
import { render, waitFor, fireEvent } from "@testing-library/react-native";
import NewsScreen from "../screens/NewsScreen";
import axios from "axios";
import { Linking } from "react-native";

jest.mock("axios");
jest.mock("react-native/Libraries/Linking/Linking", () => ({
  openURL: jest.fn(),
}));

const mockNewsData = {
  articles: [
    {
      link: "http://linktoarticle1.com",
      media: "http://linktoimage1.com/image1.jpg",
      title: "Article Title 1",
```

```
      summary: "Summary of article 1",
    },
    {
      link: "http://linktoarticle2.com",
      media: "http://linktoimage2.com/image2.jpg",
      title: "Article Title 2",
      summary: "Summary of article 2",
    },
  ],
};

describe("NewsScreen", () => {
  beforeEach(() => {
    axios.request.mockResolvedValue({ data: mockNewsData });
  });
  afterEach(() => {
    jest.clearAllMocks();
  });

  it("renders loading indicator initially", () => {
    const { getByTestId } = render(<NewsScreen />);
    expect(getByTestId("activity-indicator")).toBeTruthy();
  });

  it("renders news articles after fetching data", async () => {
    const { queryByTestId, getAllByTestId } = render(<NewsScreen />);
    await waitFor(() =>
      expect(queryByTestId("activity-indicator")).toBeFalsy()
    );
    const newsItems = getAllByTestId("news-item");
    expect(newsItems.length).toBe(mockNewsData.articles.length);
  });

  it("opens a link when a news article is pressed", async () => {
    const { getByText } = render(<NewsScreen />);
    await waitFor(() => expect(axios.request).toHaveBeenCalled());
    const firstArticleTitle = mockNewsData.articles[0].title;
    fireEvent.press(getByText(firstArticleTitle));
    expect(Linking.openURL).toHaveBeenCalledWith(mockNewsData.articles[0].link);
  });
});
```

Лістинг файлу VaccinationScreen.test..js

```
import React from "react";
import { render, fireEvent, waitFor, act } from "@testing-library/react-native";
import VaccinationScreen from "../screens/VaccinationScreen";
```

```
jest.mock("expo-notifications");

describe("VaccinationScreen", () => {
  it('shows "Select First Shot Date" button after selecting a vaccine type', () => {
    const { getByText, queryByText } = render(<VaccinationScreen />);
    fireEvent.press(getByText("Pfizer"));
    const selectDateButton = queryByText("Select First Shot Date");
    expect(selectDateButton).not.toBeNull();
  });

  it('shows the date picker after pressing "Select First Shot Date"', () => {
    const { getByText, getByTestId } = render(<VaccinationScreen />);
    fireEvent.press(getByText("Pfizer"));
    fireEvent.press(getByText("Select First Shot Date"));
    const datePicker = getByTestId("dateTimePicker");
    expect(datePicker).toBeTruthy();
  });

  it("shows the notification subscription button after selecting a date", async () => {
    const { getByText, getByTestId, queryByText } = render(
      <VaccinationScreen />
    );
    fireEvent.press(getByText("Pfizer"));
    fireEvent.press(getByText("Select First Shot Date"));
    const mockDate = new Date(2021, 1, 1);
    act(() => {
      fireEvent(getByTestId("dateTimePicker"), "onChange", {
        nativeEvent: { timestamp: mockDate },
      });
    });
    await waitFor(() => {
      const notificationButton = queryByText(
        "Click to subscribe for notifications"
      );
      expect(notificationButton).not.toBeNull();
    });
  });
});
```

Лістинг файлу SettingsScreen.test..js

```
import React from "react";
import { render, fireEvent, act } from "@testing-library/react-native";
import { Alert } from "react-native";
import SettingsScreen from "../screens/SettingsScreen";
```

```
jest.mock("firebase/auth", () => {
  return {
    getAuth: jest.fn(() => {
      return {
        currentUser: {
          uid: "testUid",
        },
      };
    }),
    EmailAuthProvider: {
      credential: jest.fn(),
    },
    reauthenticateWithCredential: jest.fn(() => Promise.resolve()),
    updateEmail: jest.fn(() => Promise.resolve()),
    updatePassword: jest.fn(() => Promise.resolve()),
    signOut: jest.fn(() => Promise.resolve()),
  };
});
jest.mock("react-native/Libraries/Alert/Alert", () => ({
  alert: jest.fn(),
}));

jest.mock("react-native/Libraries/Alert/Alert", () => ({
  alert: jest.fn(),
}));

describe("SettingsScreen", () => {
  const updateTextInput = (getByPlaceholderText, text) => {
    fireEvent.changeText(getByPlaceholderText, text);
  };

  it("toggles email input fields", () => {
    const { getByText, getByPlaceholderText } = render(<SettingsScreen />);
    const changeEmailButton = getByText("Change Email");
    fireEvent.press(changeEmailButton);
    expect(getByPlaceholderText("Enter old email")).toBeTruthy();
    expect(getByPlaceholderText("Enter new email")).toBeTruthy();
  });

  it("toggles password input fields", () => {
    const { getByText, getByPlaceholderText } = render(<SettingsScreen />);
    const changePasswordButton = getByText("Change Password");
    fireEvent.press(changePasswordButton);
    expect(getByPlaceholderText("Enter old password")).toBeTruthy();
    expect(getByPlaceholderText("Enter new password")).toBeTruthy();
  });
```

```
  it("updates email successfully", async () => {
    const { getByText, getByPlaceholderText } = render(<SettingsScreen />);
    fireEvent.press(getByText("Change Email"));
    updateTextInput(getByPlaceholderText("Enter old email"), "old@example.com");
    updateTextInput(getByPlaceholderText("Enter new email"), "new@example.com");
    updateTextInput(getByPlaceholderText("Enter old password"), "oldPassword");

    await act(async () => {
      fireEvent.press(getByText("Confirm"));
    });

    expect(Alert.alert).toHaveBeenCalledWith("Success", "Email updated");
  });

  it("updates password successfully", async () => {
    const { getByText, getByPlaceholderText } = render(<SettingsScreen />);
    fireEvent.press(getByText("Change Password"));
    updateTextInput(getByPlaceholderText("Enter old email"), "old@example.com");
    updateTextInput(getByPlaceholderText("Enter old password"), "oldPassword");
    updateTextInput(getByPlaceholderText("Enter new password"), "newPassword");
    await act(async () => {
      fireEvent.press(getByText("Confirm"));
    });
    expect(Alert.alert).toHaveBeenCalledWith("Success", "Password updated");
  });
});
```

Додаток Г. Ілюстративна частина

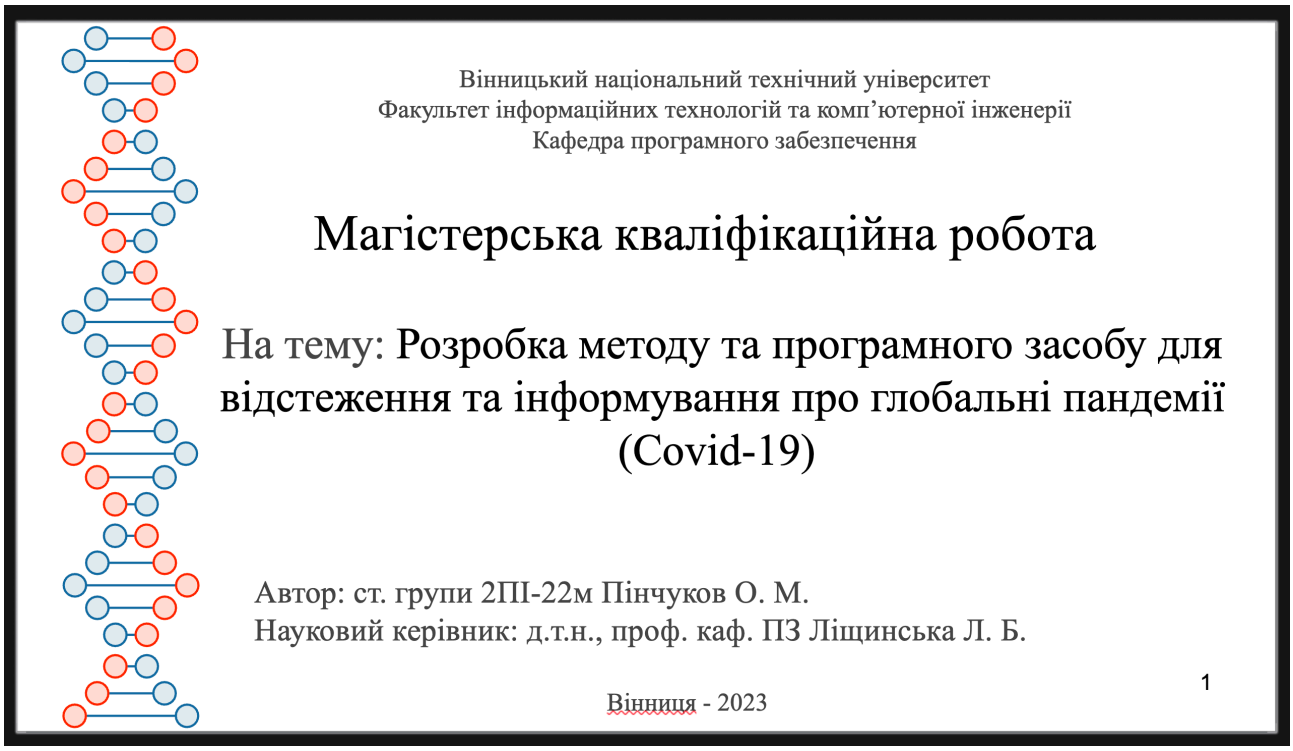

Рисунок Г.1 – Перший слайд

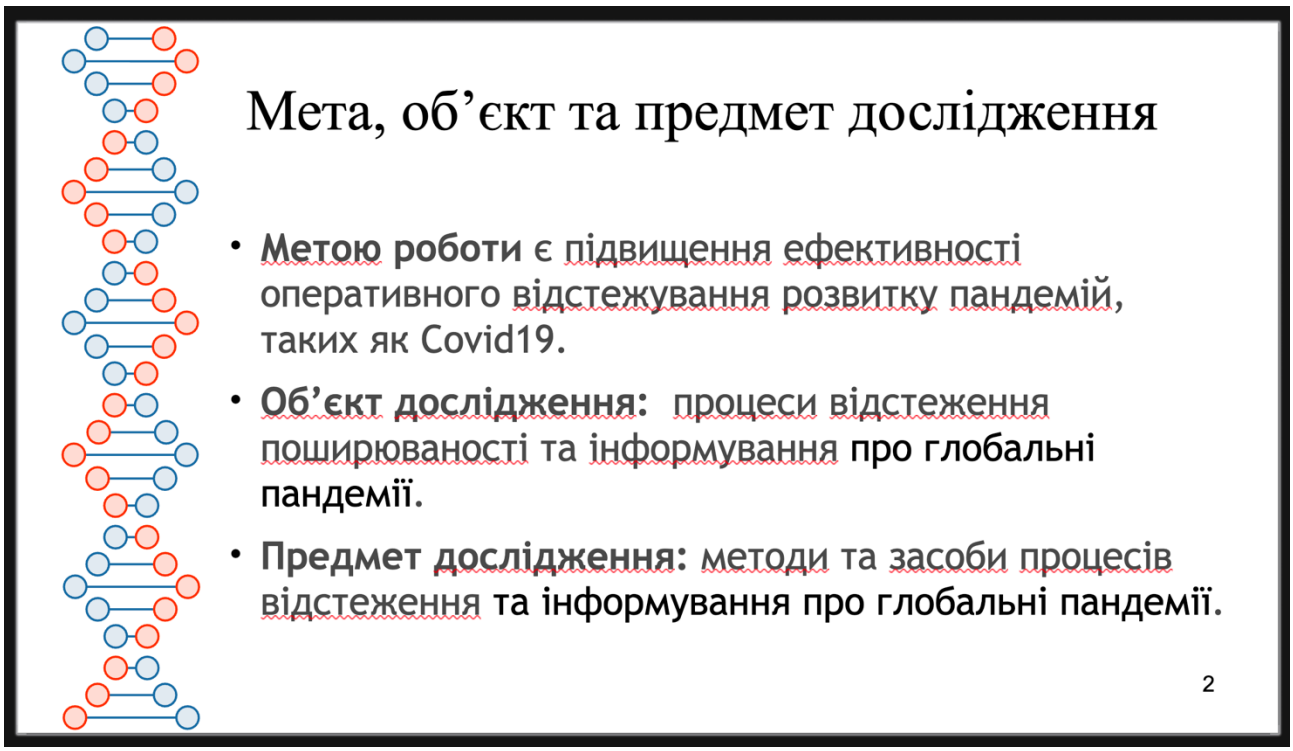

Рисунок Г.2 – Мета, об'єкт та предмет дослідження

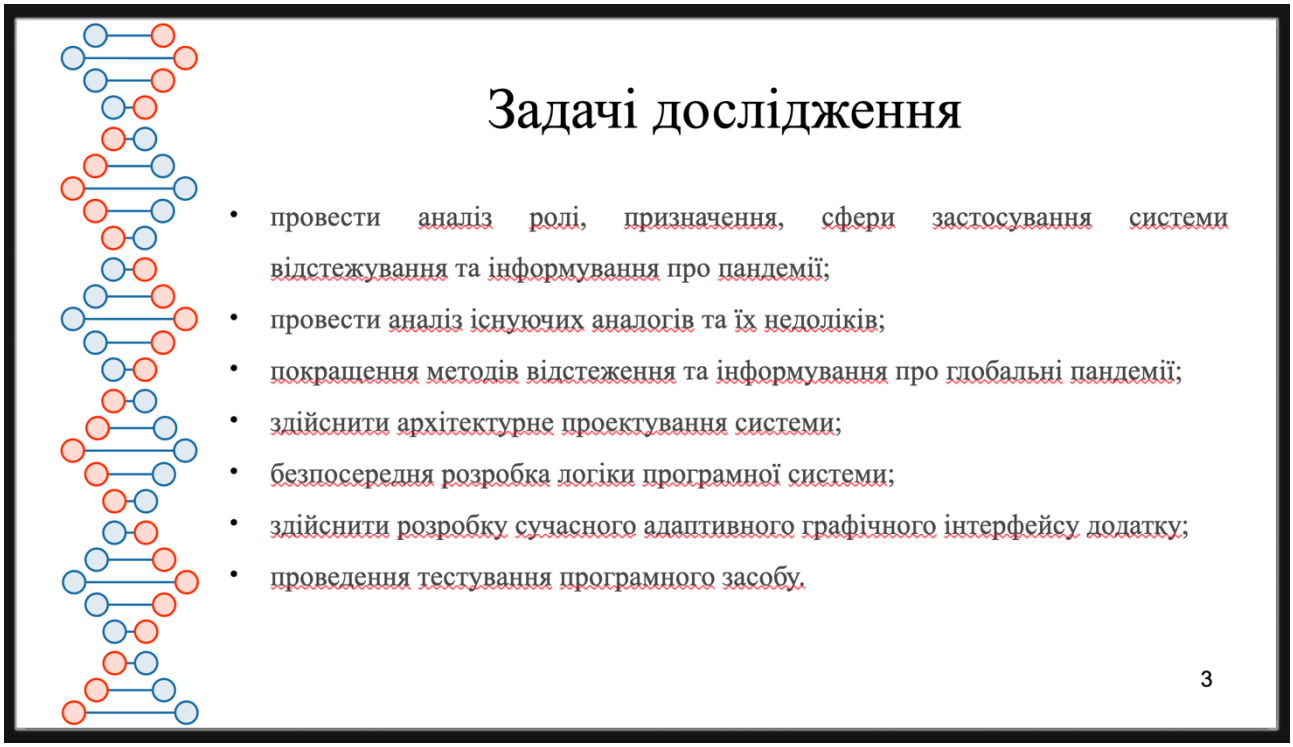

Рисунок Г.3 – Задачі дослідження

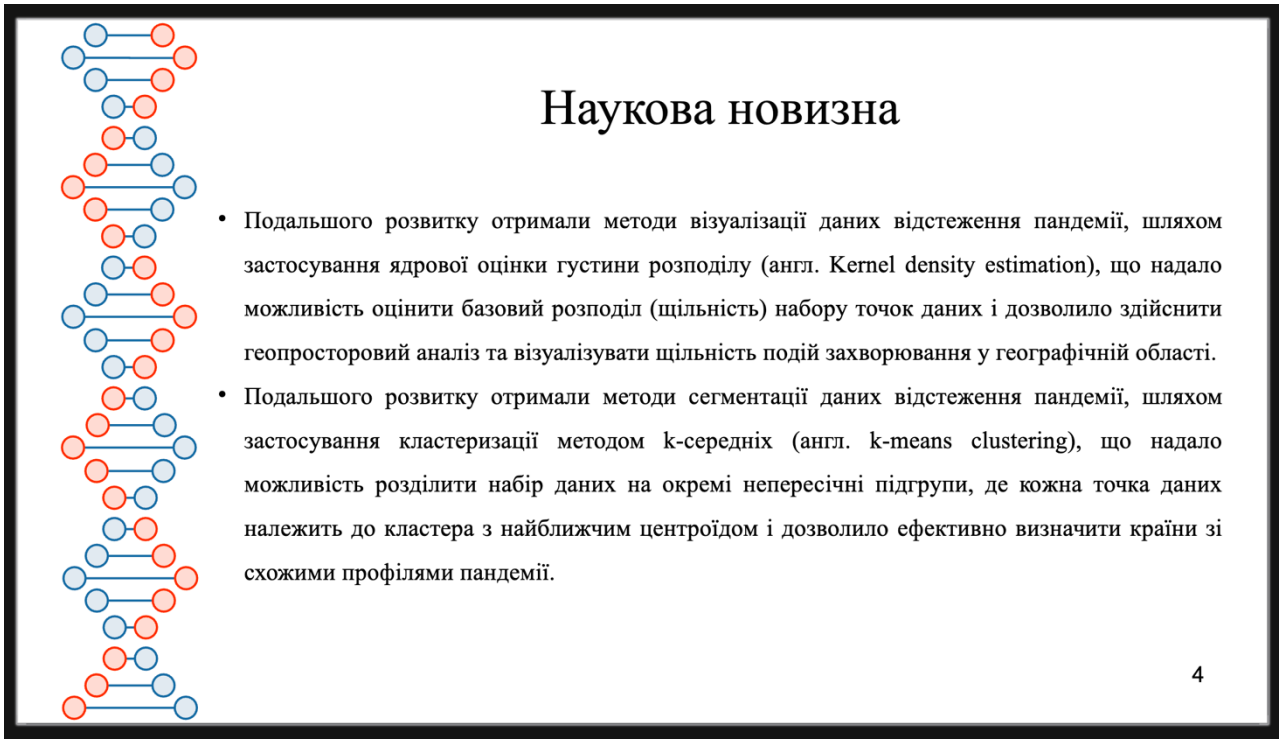

Рисунок Г.4 – Наукова новизна

Порівняльний аналіз аналогів

Критерій	covidvisualizer.com	Coronavirus.app	WHO	Власна розробка
Інтерактивна мапа поширення Covid-19	+	+	+	+
Покращені аналітична складова додатку (KDE, K-means)	-	-	-	+
Фільтрація даних	-+	+	+	+
Новини про Covid-19	-	-	+	+
Мобільна версія (додаток)	-	-	-	+
Пуш-повідомлення (нагадування) на пристрій	-	-	-	+
Сума	1.5	2	3	6

Рисунок Г.5 – Порівняльний аналіз аналогів

Покращення методів візуалізації даних шляхом застосування ядрової оцінки густини розподілу (KDE)

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-X_i}{h}\right)$$

- Дозволив здійнити геопросторовий аналіз та візуалізувати щільність подій у географічній області.
- За основу аналізу були взяті дані щодо кількості випадків на мільйон осіб.
- Ці дані були застосовані для відображення теплової карти випадків.

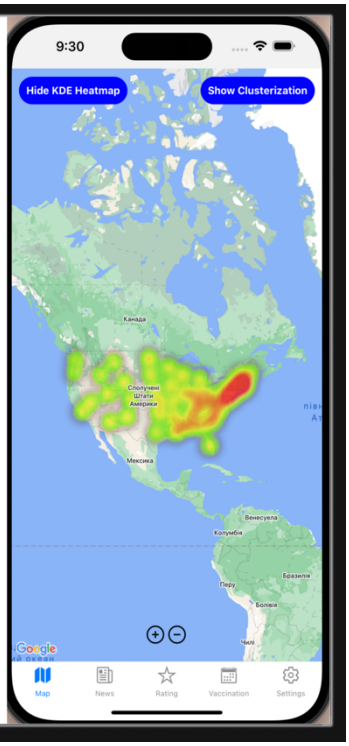

Рисунок Г.6 – Покращення методів візуалізації даних

Рисунок Г.7 – Покращення методів сегментації даних

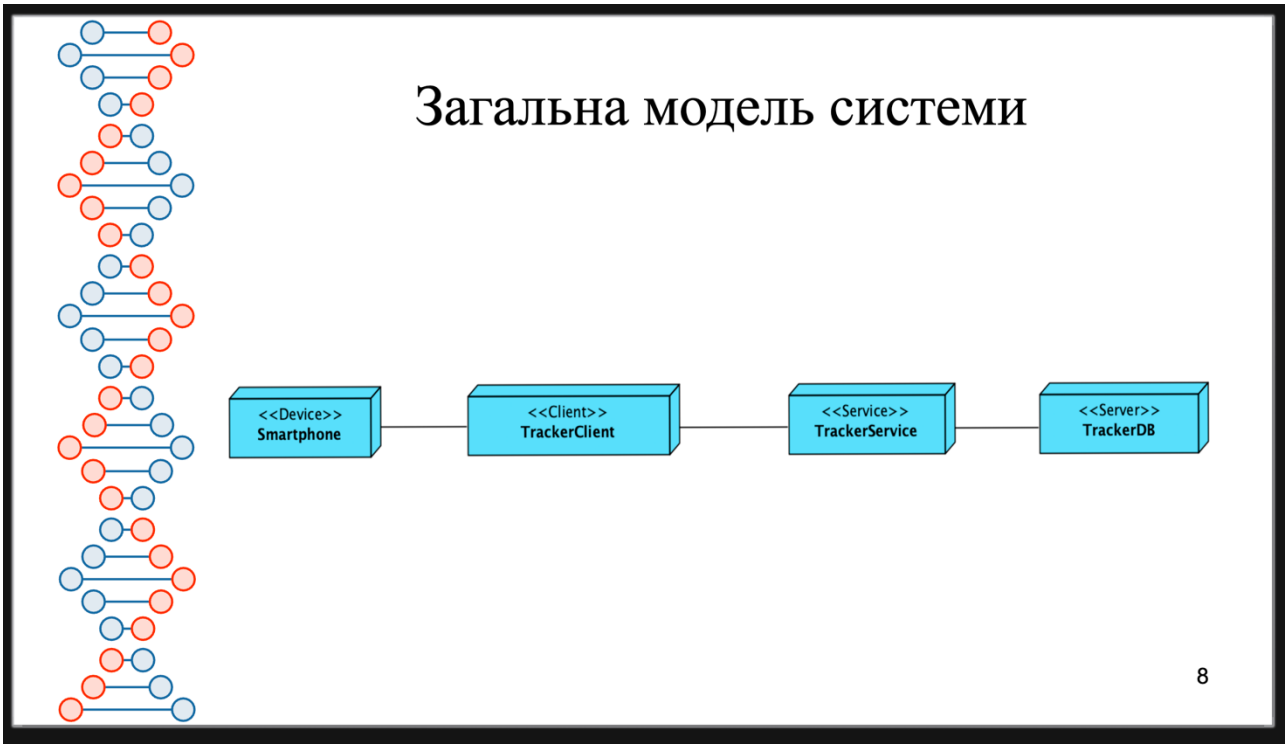

Рисунок Г.8 – Загальна модель системи

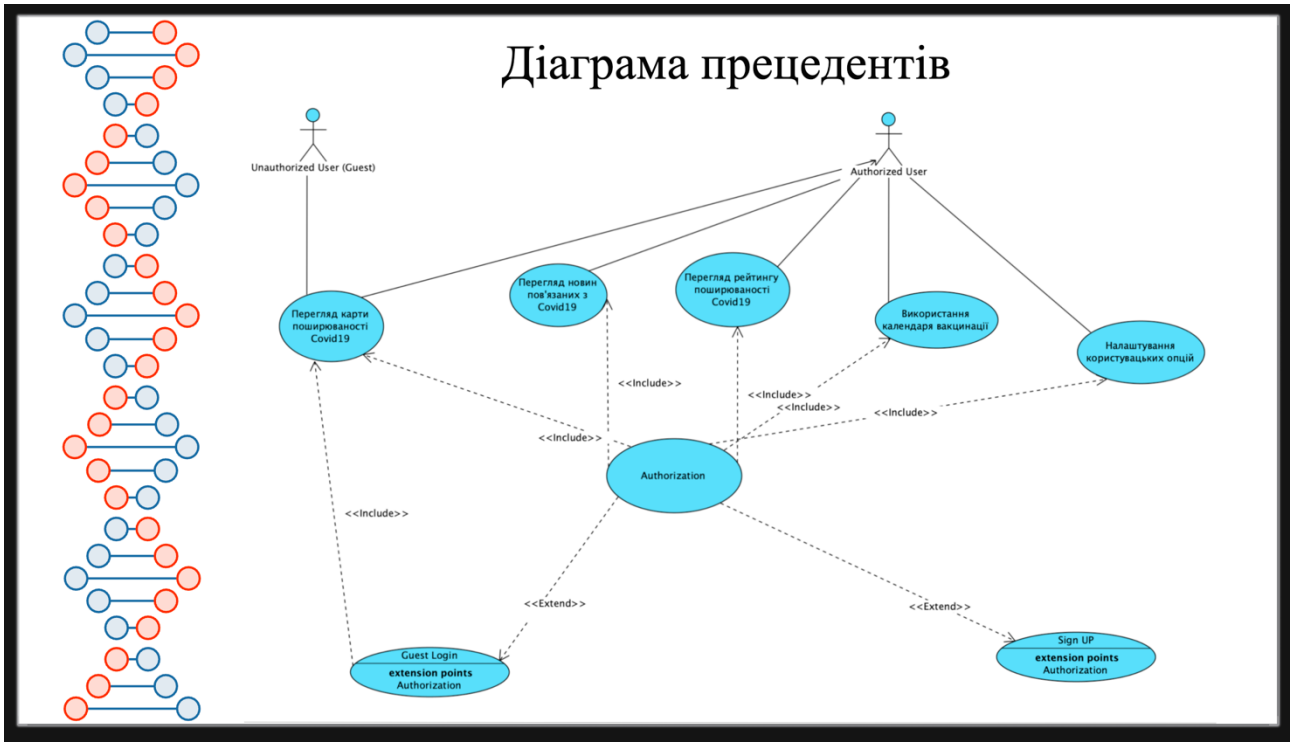

Рисунок Г.9 – Діаграма прецедентів

Рисунок Г.10 – Модуль "Авторизація"

Рисунок Г.11 – Модуль "Карти"

Рисунок Г.12 – Модуль "Рейтинг"

Рисунок Г.13 – Модуль "Новини"

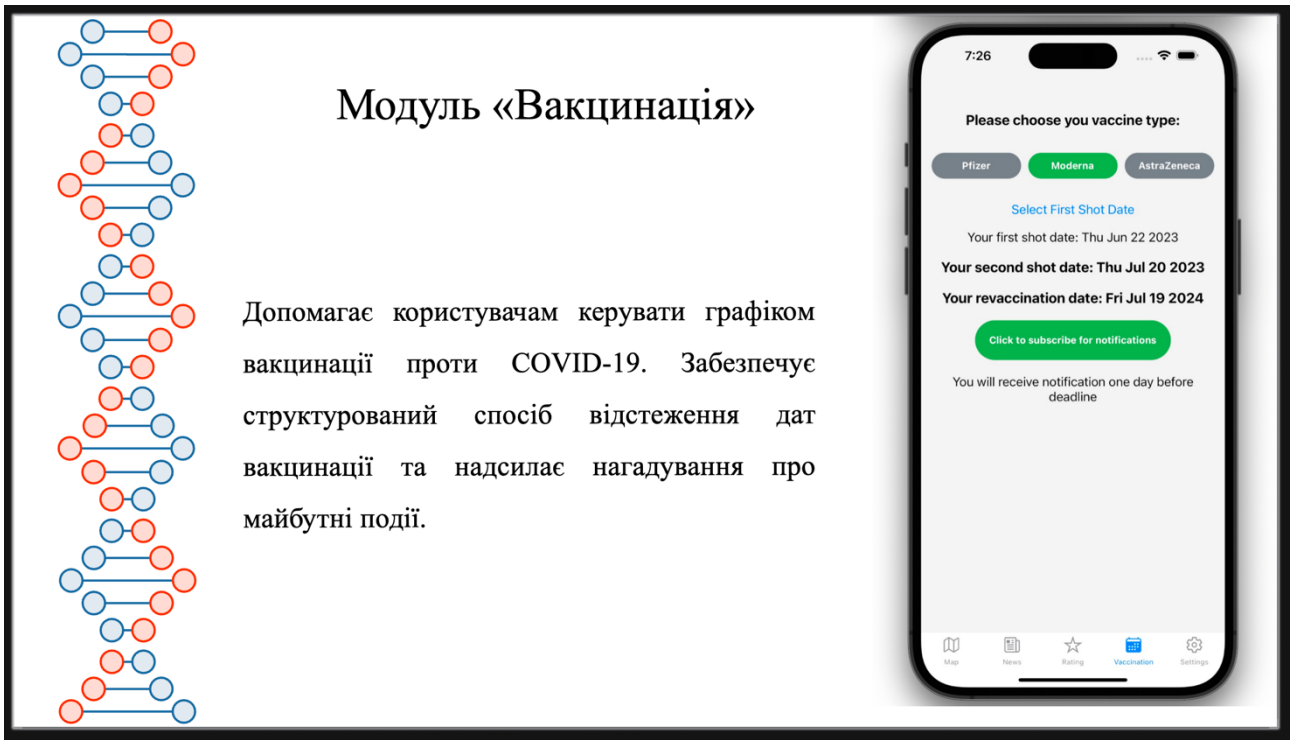

Рисунок Г.14 – Модуль "Вакцинація"

Рисунок Г.15 – Модуль "Налаштування"

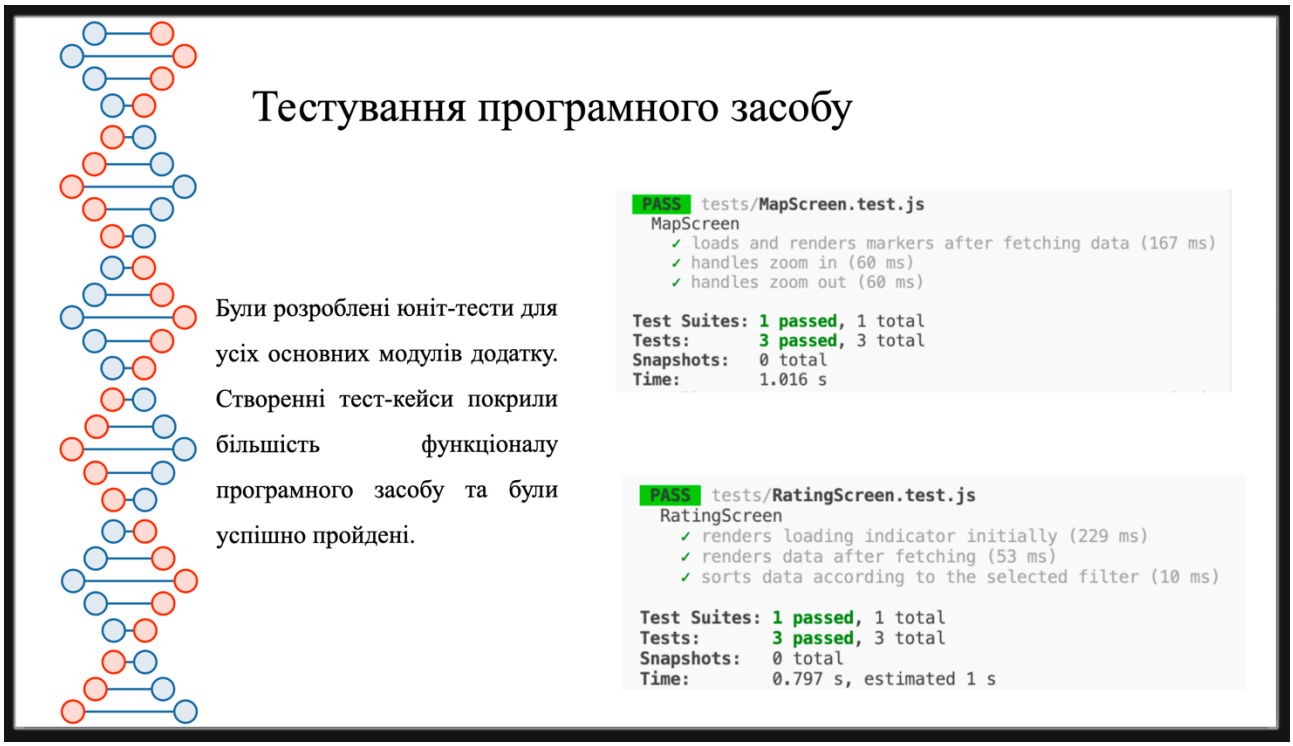

Рисунок Г.16 – Тестування програмного засобу

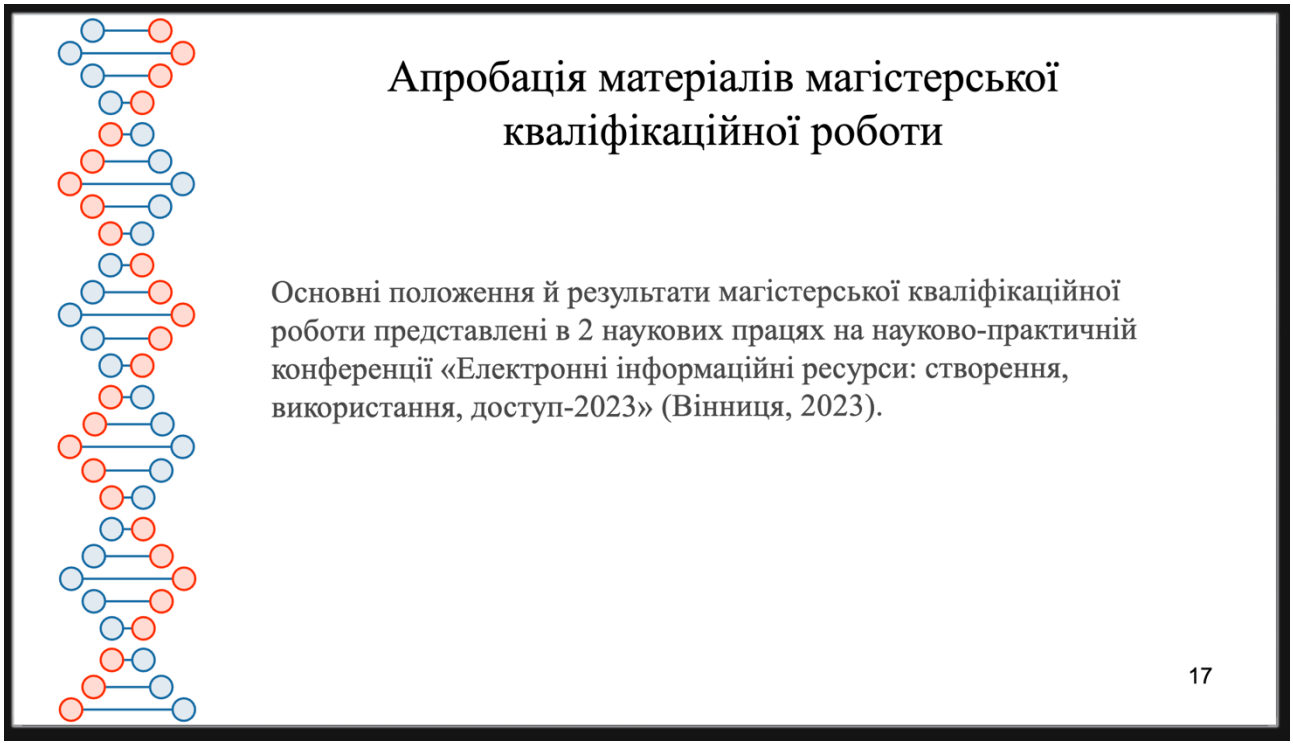

Рисунок Г.17 – Апробація матеріалів

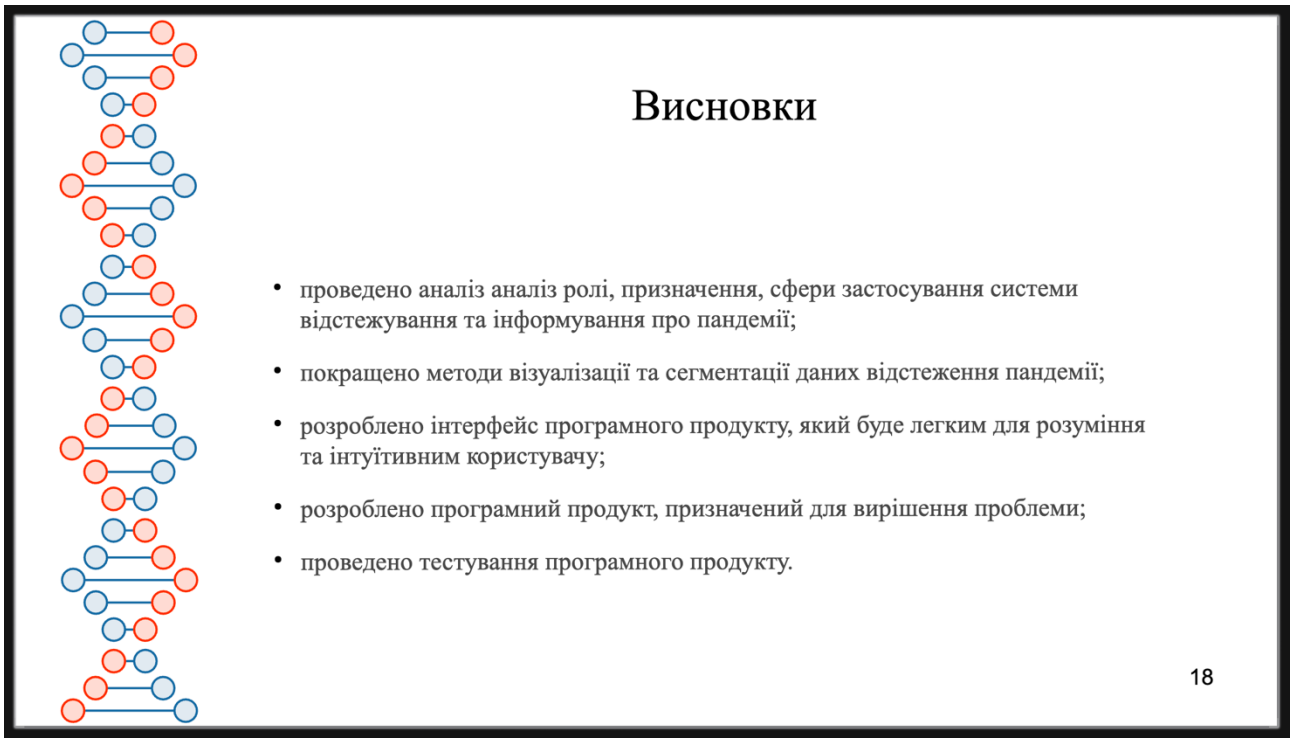

Рисунок Г.18 – Висновки