МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

"_____" _____ 2024 р.

Зав. кафедри, к.т.н., доцент

_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Мобільний застосунок для створення медіа-контенту з забезпеченням приватності даних користувача

Виконав – студент 2 курсу, групи 21м-з-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Кулаков Павло Ігорович
Науковий керівник – д.т.н., професор
Кучерук Володимир Юрійович

Умань – 2024

Реферат

на кваліфікаційну роботу «Мобільний застосунок для створення медіа-контенту з забезпеченням приватності даних користувача»

студента факультету економіки та підприємництва
Уманського національного університету садівництва
Кулакова Павла Ігоровича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 3 розділів, містить 79 сторінок, 28 рисунків, 6 додатків, використано 50 літературних джерел.

Кваліфікаційна робота магістра присвячена розробці мобільного застосунку для створення медіа-контенту, особливістю якого є забезпечення приватності даних користувача. Розроблено діаграми варіантів використання, спроектовано логічну та фізичну структури мобільного додатку, розроблено алгоритм та модуль агрегування медіа-контенту, виконано розробку додатка для Android та iOS.

Наукова новизна отриманих результатів кваліфікаційної роботи полягає у тому, що розвинуто метод розпізнавання зображень для створення медіа-контенту, відмінністю якого є функціонування безпосередньо на мобільному пристрої, за рахунок чого досягається приватність даних, удосконалено модель мобільної системи, що дозволило скоротити об'єм пам’яті і робоче навантаження мережі при розгортанні на мобільних пристроях. Практичне значення одержаних результатів полягає в тому, що розроблені програмні засоби використовують сучасні технології, що забезпечує програмному продукту високу конкурентоспроможність. Для досягнення поставлених завдань використовуються сучасні методи створення мобільних додатків з інтерактивним інтерфейсом.

У першому розділі проведено обґрунтування доцільності розробки, досліджуються існуючі аналоги та формулюється завдання розробки. У другому розділі виконано розробку структури програмного продукту, алгоритм агрегування медіа-контенту, проектується структура сторінок та розробляється дизайн. Третій розділ містить процес розробки мобільного додатку.

Анотація

У кваліфікаційній роботі магістра розроблено мобільний застосунок для розпізнавання зображень з метою створення медіа-контенту, особливістю якого є забезпечення приватності даних користувача. В процесі виконання роботи проведено дослідження предметної області, здійснено порівняльний аналіз аналогів, що використовують методи та засоби розпізнавання зображень. На основі аналізу недоліків та переваг аналогів визначено завдання розробки, розроблено діаграми варіантів використання, спроектовано логічну та фізичну структури мобільного додатку, проведено аналіз технологій розробки, розроблено алгоритм та модуль агрегування медіа-контенту, виконано розробку для платформ Android та iOS. Для досягнення поставлених завдань використовуються сучасні методи створення мобільних додатків з інтерактивним інтерфейсом.

Ключові слова: розпізнавання зображень, медіа-контент, штучний інтелект, мобільний додаток, приватність даних.

Annotation

In the master's qualification work, a mobile application for image recognition was developed for the purpose of creating media content, the feature of which is to ensure the privacy of user data. In the course of the work, a study of the subject area was carried out, a comparative analysis of analogues using image recognition methods and tools was carried out. Based on the analysis of the disadvantages and advantages of analogues, the development tasks were defined, diagrams of use cases were developed, the logical and physical structure of the mobile application was designed, an analysis of development technologies was carried out, an algorithm and a module for aggregating media content were developed, and development was carried out for Android and iOS platforms. Modern methods of creating mobile applications with an interactive interface are used to achieve the set goals.

Keywords: image recognition, media content, artificial intelligence, mobile application, data privacy.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва
Кафедра _____Інформаційних технологій_____
Освітня програма _____Комп'ютерні науки_____
Спеціальність_____122 Комп'ютерні науки_____
(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ ___Ліщук Р.І._____
(підпис) (ініціали, прізвище)
"_____" _____ 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту_____*Кулакову Павлу Ігоровичу*_____
(прізвище, ім'я, по батькові)
1. Тема проекту_*Мобільний застосунок для створення медіа-контенту з забезпеченням приватності даних користувача*_____
керівник роботи_____*Ліщук Роман Ігорович, к.т.н., доцент кафедри інформаційних технологій*_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _"03" грудня 2024 року № 01-06/320_
2. Строк подання студентом роботи_____01.12.2023_____
3. Вихідні дані до роботи _*Розробити мобільний застосунок для розпізнавання зображень з метою створення медіа-контенту, забезпечити приватність даних користувача, виконати розробку мобільного додатку для платформ Android та iOS на мові програмування Dart у середовищі розробки Android Studio та Xcode.*_
4. Перелік питань, які потрібно розробити _*Аналітичний огляд предметної області, обґрунтування доцільності розробки, розробка структури програмного продукту, алгоритм агрегування медіа-контенту, структура сторінок та дизайн додатку, розробка мобільного додатку.*_____
5. Перелік графічного матеріалу _____28 рисунків, 6 додатків, 50 джерел_____
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи		
2.	Одержання індивідуального завдання		
3.	Складання календарного плану написання кваліфікаційної роботи		
4.	Підготовка до виконання кваліфікаційної роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи		
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи		
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків		
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень наукового керівника		
8.	Написання та оформлення роботи в остаточному варіанті		
9.	Перевірка роботи на академічний плагіат		
10.	Одержання відгуку наукового керівника		
11.	Попередній захист роботи на кафедрі		
12.	Затвердження роботи завідувачем кафедри		
13.	Одержання рецензії		
14.	Захист кваліфікаційної роботи		

Студент_____ **Кулаков П.І.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Кучерук В.Ю.**
(підпис) (прізвище та ініціали)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ……... 7
ВСТУП…… 8
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ…….. 11
1.1. Аналіз існуючих засобів агрегування медіа-контенту…….. 11
1.2. Методи та засоби реалізації програмного продукт……... 16
1.3. Постановка задачі……. 18
РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ МОБІЛЬНОГО ДОДАТКУ ТА АЛГОРИТМУ АГРЕГУВАННЯ МЕДІА-КОНТЕНТУ…….. 20
2.1. Розробка архітектури мобільного додатку……. 20
2.2. Проектування алгоритму агрегування медіа-контенту……. 26
2.3. Розробка структури екранів та дизайну мобільного додатку…….. 33
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……. 44
3.1. Аналіз засобів реалізації програмного продукту……..... 44
3.2. Програмна реалізація мобільного додатку …… 50
3.3. Програмна реалізація алгоритму агрегування медіа-контенту…… 57
ВИСНОВОК…… 61
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ…… 63
ДОДАТОК А Лістинг вихідного коду…….. 67
ДОДАТОК Б Лістинг файлу CoreApp..……..... 68
ДОДАТОК В Лістинг файлу CoreAppController..... 69
ДОДАТОК Г Лістинг файлу AutoCreateStoryManager 72
ДОДАТОК Д Лістинг файлу PhotoManager…..……..... 75
ДОДАТОК Е Лістинг файлу CreateStoryManual..... 77

Перелік умовних позначень, скорочень і термінів

ІТ	Інформаційні технології.
МП	Мікропроцесор.
БД	База даних.
ПЗ	Програмне забезпечення.
ПД	Програмний додаток.
МД	Мобільний додаток
ШІ	Штучний інтелект.

Вступ

Мобільні пристрої в житті сучасної людини займають особливе місце. Вони допомагають бути на зв’язку зі світом у будь-який час і активно використовуються в різних сферах нашого життя. Особливо з розвитком технологій мобільної фотографії вони почали замінювати собою необхідність використання професійних та аматорських камер, оскільки дають змогу робити якісні знімки без особливих умов та навиків.

На даний момент часу розробка мобільних додатків є одним із пріоритетних напрямків процесу інформатизації сучасного суспільства. Над зручністю і простотою створення мобільних додатків з інтерактивним інтерфейсом працює велика кількість працівників різних фахів. Мобільні додатки з кожним днем стають дедалі популярнішими, і часто замінюють собою використання WEB сайтів, а інколи мають унікальний або розширений функціонал у порівнянні з WEB додатками, оскільки розраховуються на більшу кількість активних користувачів.

Актуальність теми дослідження. У кваліфікаційній роботі магістра розроблено програмне забезпечення у вигляді мобільного додатку, що використовує методи та засоби розпізнавання зображень для створення медіа-контенту, основною задачею якого є сканування галереї мобільного пристрою та об’єднання знайдених медіа файлів у події за розробленим алгоритмом. Завдяки тому, що розпізнавання зображень виконується на мобільному пристрої, забезпечується повна приватність відео та фотоматеріалів користувача, та дозволяє виконувати операції з розпізнавання об’єктів швидко. Більшість звичайних людей використовують мобільні телефони як головний пристрій для здійснення відео та фото знімків, тому що він завжди під рукою та його можливостей в більшості достатньо для здійснення прийнятих фотографій. Таким чином з часом в галереї пристрою накопичується досить велика кількість знімків та відео матеріалів. З часом, переглядаючи здійснені знімки, можуть виникнути проблеми з їх відокремленням до певних життєвих подій. Для цього необхідний програмний засіб, який допоможе здійснити агрегування фото та відеоматеріалів та віднести їх

до певних життєвих подій, при цьому залишивши лише ті знімки, які мають певну цінність. Тому тематика даного дослідження щодо розробки методів і програмних засобів розпізнавання зображень для агрегування медіа контенту є актуальною.

Об’єкт дослідження кваліфікаційної роботи магістра - процеси розробки методів та програмних засобів розпізнавання зображень для агрегування медіа контенту.

Предмет дослідження кваліфікаційної роботи магістра - методи та засоби агрегування медіа контенту.

Мета дослідження кваліфікаційної роботи магістра - зменшення ресурсів для реалізації алгоритму агрегування зображень у події за рахунок застосування технологій трансферного навчання.

Задачі дослідження кваліфікаційної роботи магістра:

- ідентифікація проблеми агрегування медіа контенту;
- здійснення порівняльного аналізу аналогів;
- аналіз методів та засобів реалізації програмного додатку;
- розробка алгоритму агрегування медіа-контенту;
- розробка UML-діаграм для відображення основних функцій системи;
- розробка програмного додатку.

Для досягнення поставлених в роботі завдань використовуються сучасні методи створення мобільних додатків з інтерактивним інтерфейсом та баз даних.

Наукова новизна отриманих результатів:

- отримав розвиток метод розпізнавання зображень для агрегування медіа-контенту, відмінністю якого є використання відповідних технологій безпосередньо на мобільному пристрої, за рахунок чого досягається приватність даних користувача;

- отримала розвиток модель мобільної системи, яка на відміну від відомих моделей трансферного навчання дозволяє скоротити об'єм пам’яті і робоче навантаження мережі при розгортанні на мобільних пристроях.

Практичне значення одержаних результатів кваліфікаційної роботи магістра полягає в тому, що розроблена система методів та програмних засобів використовує низку сучасних технологій, що забезпечують створеному програмному продукту високу конкурентоспроможність на ринку.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих засобів агрегування медіа-контенту

У теперішній час мобільні телефони та інші мобільні пристрої мають надзвичайно велике росповсюдження, більшість людей у світі є власниками подібних пристроїв. Сьогодні у світі налічується більше 5 мільярдів користувачів мобільних телефонів, що становить 65% населення. До появи смартфонів зробити якісну фотографію було трудомістким процесом. Зараз, з розвитком технологій обробки зображень та самих камер, зробити якісну фотографію не є особливо складним процесом. Смартфон, який завжди знаходиться поруч, завжди є ефективною фотокамерою, оскільки він завжди готовий до використання, і допомагає зафіксувати важливі моменти у житті. Подорожі, дні народження або зустріч з друзями - смартфон завжди під рукою, і здійснивши декілька натискань він збереже момент життя, допоможе згадати його у будь-який момент, коли користувачу цього захочеться. В цьому і є найбільша перевага мобільних телефонів над професійними або аматорськими камерами.

Завдяки смартфонам, творче самовираження може бути доступним кожному, а не лише тим, хто може дозволити собі професійні інструменти та обладнання. Це одна з причин, чому використання мобільної фотографії є таким росповсюдженим [1]. Оскільки смартфони, планшетні комп’ютери та інші мобільні пристрої продовжують удосконалюватись і стають для багатьох творчими супутниками, все більше людей долучаються до демократизації творчості та розширюють межі того, що можливо за допомогою фотографії. З покращенням технічних характеристик смартфонів, в основному об’єму вбудованої пам’яті, збереження великої кількості зображень на ньому не є проблемою. Ця можливість є значною перевагою, оскільки дозволяє тримати всі сфотографовані моменти життя в одному місці, допомагаючи швидко знайти їх за необхідністю.

Однак зі збільшенням кількості фотографій на смартфоні збільшується і складність пошуку спогадів на ньому. Також, за час експлуатації, на смартфоні збирається багато непотрібних фотографій, наприклад, документів, фотографій

лічильників, записів тощо. Певні моменти в році викликають додаткові позитивны емоції - дні народження, подорожі та свята, тому люди користуються своїми смартфонами щоб переглянути старі фотографії. Позитивні емоції втрачаються, коли користувачу доводиться переглядати сотні дублікатів фотографій.

Для вирішення даної проблеми необхідне рішення, яке допоможе відокремити значущі відзняті знімки від незначущих у галереї, і агрегувати їх у спогади, які потім з легкістю можна буде знайти в одному місці.

Для розробки програмного продукту необхідно визначитись з доцільністю розробки за допомогою порівняльного аналізу існуючих аналогів на ринку. Це допоможе не тільки визначити актуальність розробки, а й допоможе підкреслити переваги та недоліки існуючих програмних рішень, що допоможе імплементувати існуючі переваги продуктів, і таким чином отримати продукт, який ще на етапі проектування буде мати меншу кількість недоліків ніж у аналогів.

Крім основного завдання, з агрегування медіа контенту, розроблюваний програмний продукт має працювати на основних операційних системах, для сучасних мобільних пристроїв, тому необхідно ознайомитись з аналогами на обох платформах.

Вирішенням даної проблеми займаються в основному розробники головних операційних систем для мобільних пристроїв. На сьогоднішній день існують різні рішення даної проблеми. Програмні продукти допомагають зберігати зображення у хмарних сховищах, а також групувати їх і створювати спогади. Для основних операційних систем на сьогоднішній день існують наступні рішення.

«Photos» для iOS [2] – це програма для керування та редагування фотографій, розроблена Apple. Даний програмний продукт є вбудованим додатком для смартфонів, які працюють на операційній системі iOS. За його допомогою можна як просто переглядати знімки, які збережені в смартфоні, так і переглядати згенеровані спогади, редагувати фото та відео контент. Додаток впорядковує фотографії за «моментом», як це було визначено за допомогою комбінації метаданих часу та місця, доданих до фотографії. Починаючи з версії 5.0 (випущеної

в 2019 році з macOS 10.15 Catalina), фотографії можна переглядати за роком, місяцем або днем. Згенеровані спогади за допомогою цього додатку наведено на рисунку 1.1.

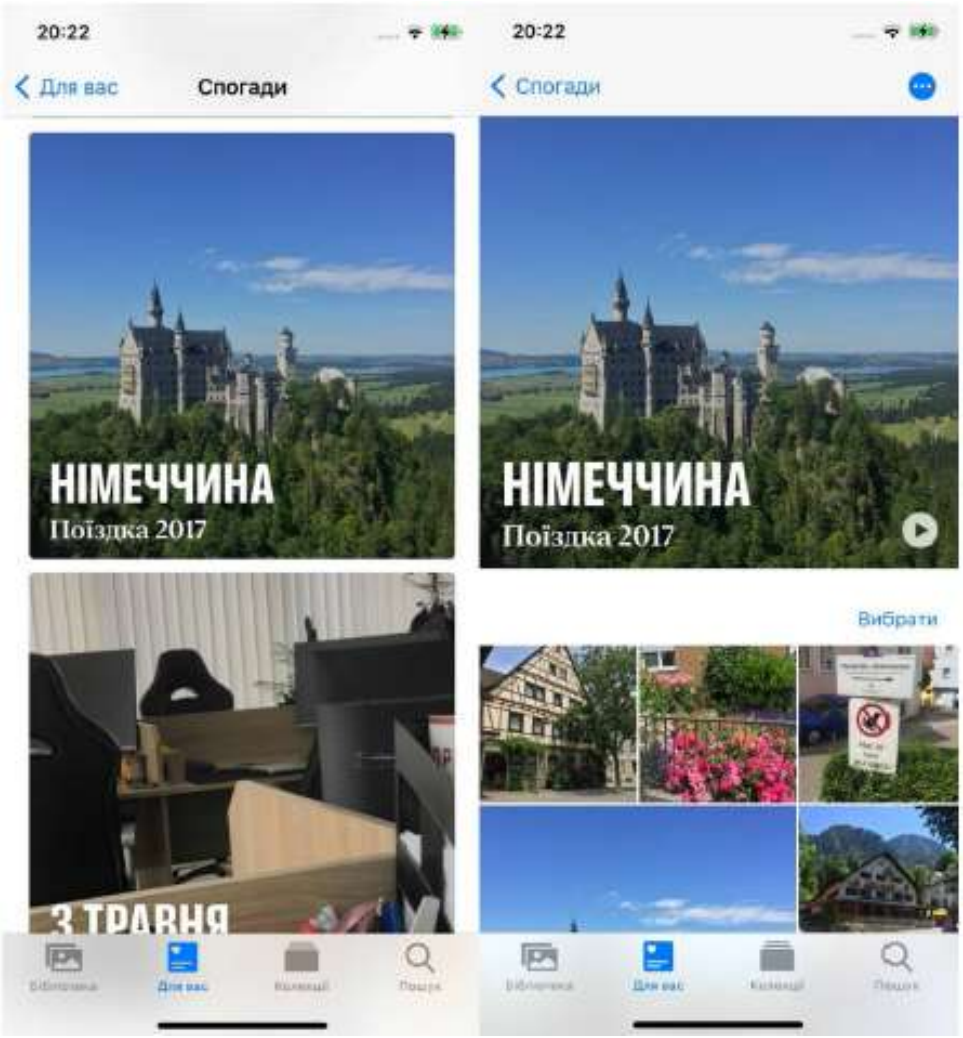


Рисунок 1.1 – Згенеровані спогади за допомогою додатку «Фотографії»

Google Photos [3] - доволі популярний додаток для перегляду знімків на смартфоні [4]. Він попередньо встановлюється на усі нові смартфони, що працюють під операційною системою Android, а також мають додаток для iOS. Даний додаток також має автоматичну генерацію спогадів, що і аналог на платформі iOS. Однак має більшу кількість функціоналу та категорій для

групування. Спогади, згенеровані за допомогою додатку «Google Photos» наведено на рисунку 1.2.

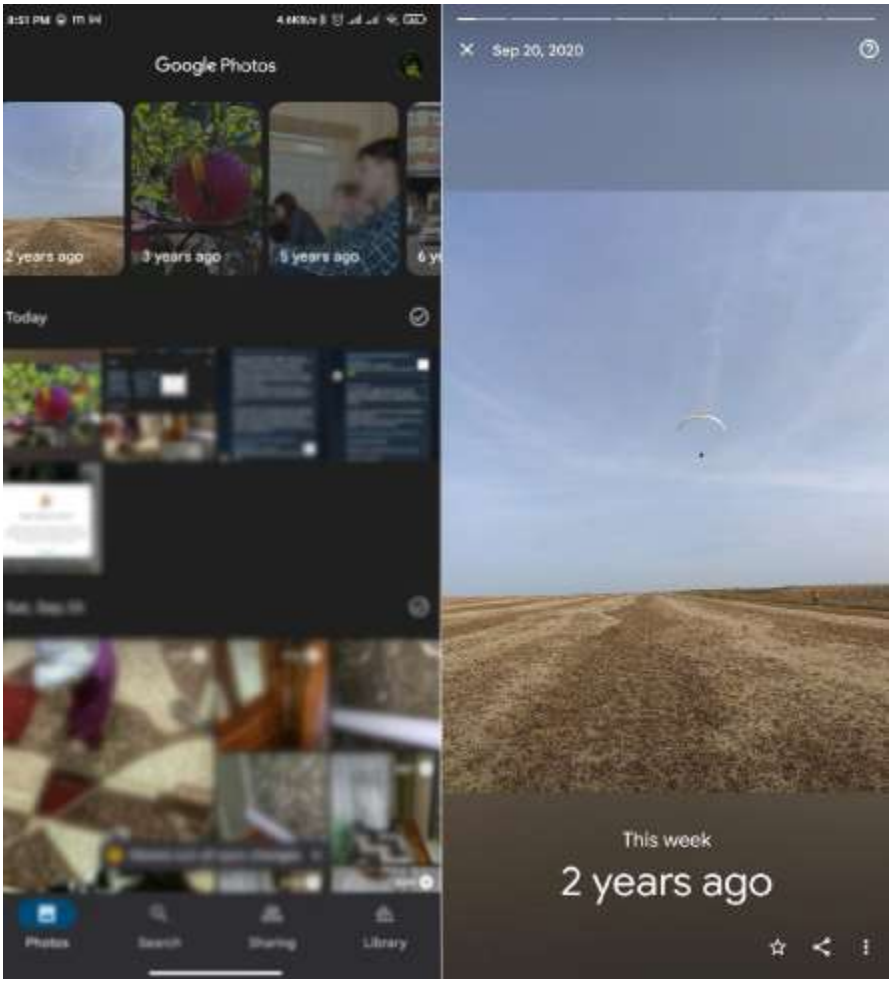


Рисунок 1.2 – Згенеровані спогади за допомогою додатку «Google Photos»

Обидва додатки мають вбудовану підтримку збереження медіа контенту на хмарних сховищах. Однак «Google Photo» для цього використовує «Google Drive» [5], а додаток «Photos» використовує «Apple iCloud» [6]. Головним недоліком, який присутній в обох рішеннях є відсутність приватності персональних даних, а саме, оскільки користувацькі медіа файли завантажуються на хмарні сховища, що належать корпораціям, вони не є максимально захищеними від взломів. Це означає, що корпорації Google та Apple мають доступ до Ваших персональних фотографій та відео без вашого відому. Досить частою практикою є так званий «злом

аккаунтів», коли персональні дані переходять до рук зловмисників, найчастіше подібні випадки відбуваються у зіркових персон, оскільки їх дані мають особливу цінність.

Таким чином, одним з основних завдань розроблюваного програмного продукту є забезпечення максимальної приватності персональних даних з використанням різних технологій шифрування даних та обробка даних на самому пристрої.

На основі проведеного порівняльного аналізу існуючих рішень створено порівняльну таблицю для виявлення переваг та недоліків існуючих рішень. Порівняльну характеристику аналогів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Google Photos	Photos для iOS	Mem
Агрегація медіа-контенту	Так	Так	Так
Підтримка обох платформ	Так	Ні	Так
Створення спогадів власноруч	Ні	Ні	Так
Редагування автоматично створених спогадів	Ні	Ні	Так
Збереження спогадів у хмарному середовищі	Так	Так	Так
Додавання аудіо файлів до спогадів	Ні	Ні	Так
Додавання текстового опису до спогадів	Ні	Ні	Так
Відображення спогадів на карті	Так	Так	Так
Можливість генерації спогадів без з’єднання з мережею	Ні	Так	Так
Обробка медіа даних ресурсами пристрою	Ні	Ні	Так
Забезпечення приватності персональних даних через локальну обробку даних	Ні	Ні	Так
Зміна назви локації для одного або групи спогадів	Ні	Ні	Так

Таким чином, розроблюваний програмний продукт «Mem» є кращим за існуючі аналоги, що підтверджує актуальність даної розробки.

1.2 Методи та засоби реалізації програмного продукта

Для здійснення агрегування медіа контенту галереї мобільного пристрою необхідно відокремити зображення, що мають певну цінність від не потрібних фотографій для подальшої генерації спогадів. Для цього було вирішено використовувати методи та програмні засоби розпізнавання зображень, що допоможе знайти на сканованих зображеннях та відео об’єкти, які можна буде віднести до певних груп, на основі яких буде згенеровано спогади.

Розпізнавання зображень - це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних. При постановці задач розпізнавання намагаються користуватися математичною мовою.

Для оптичного розпізнавання образів можна застосувати метод перебору вигляду об'єкта під різними кутами, масштабами, зсувами й т. д. Для букв потрібно перебирати шрифт, властивості шрифту та інш. [7]. Другий підхід - знайти контур об'єкта й досліджувати його властивості (зв'язність, наявність кутів та інш.). Ще один підхід — використовувати штучні нейронні мережі. Цей метод вимагає або великої кількості прикладів задачі розпізнавання (із правильними відповідями), або спеціальної структури нейронної мережі, що враховує специфіку даної задачі.

Для реалізації даної роботи буде використано другий підхід. Він не потребує створення і навчання власної нейронної мережі, оскільки дозволяє використовувати вже готові приклади вирішення. Результат роботи штучних нейронних мереж для розпізнавання зображень наведено на рисунку 1.3.

Оскільки програмний продукт передбачено для мобільних пристроїв, то необхідно використати максимально оптимізоване вирішення проблеми. Серед таких готових рішень є Tensor Flow. TensorFlow — відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявляння та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди [8].

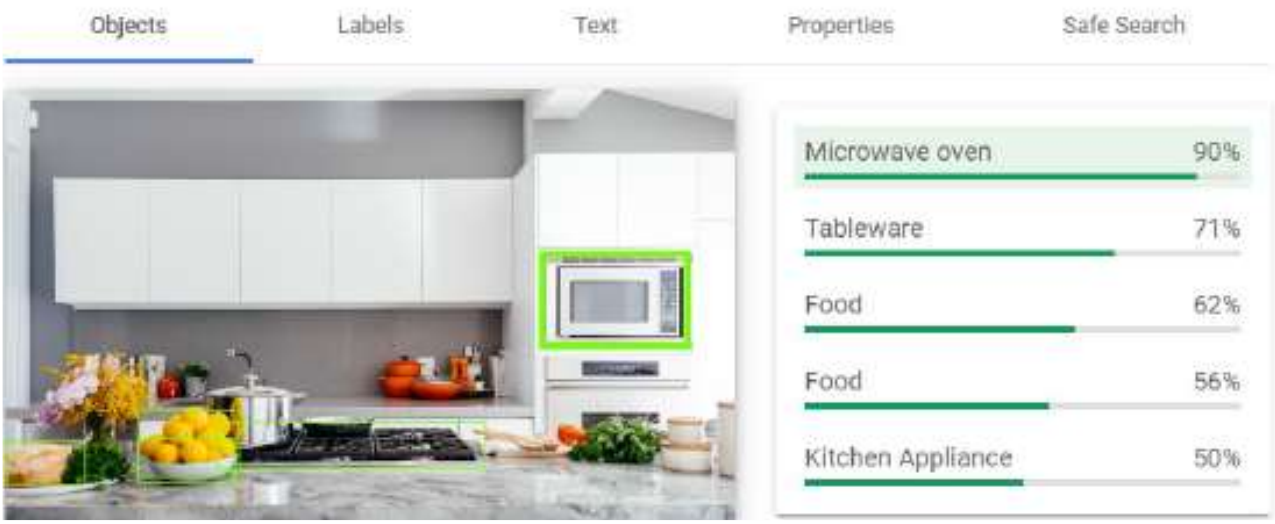


Рисунок 1.3 – Результат роботи штучних нейронних мереж
для розпізнавання зображень

Головною перевагою Tensor Flow є те, що дана бібліотека є відкритою, та вміє використовувати тензорні ядра процесора, що пришвидшує обробку зображень в рази [9]. Функціональний потік виконання для моделей Tensor Flow зображено на рисунку 1.4.

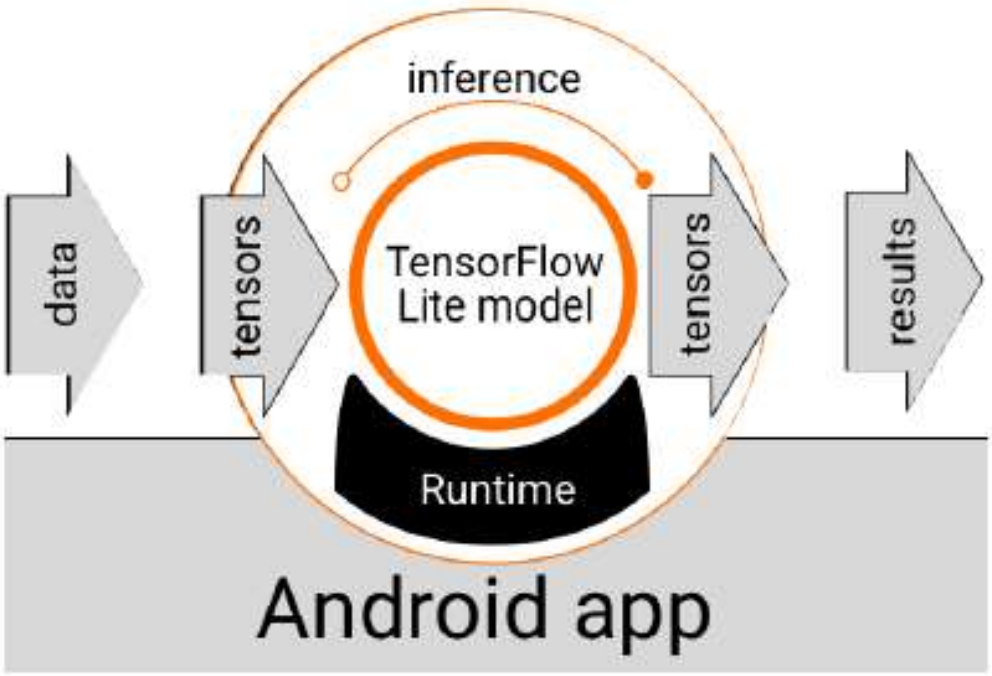


Рисунок 1.4 – Функціональний потік виконання для моделей TensorFlow

Таким чином, встановлено, що в сучасній сфері мобільних додатків існує потреба в реалізації програмного засобу для агрегування медіа-контенту, яку обумовлює низька ефективність захисту приватних даних в вже існуючих програмних рішеннях.

Методи і засоби для агрегування медіа контенту з використанням програмної бібліотеки Tensor Flow забезпечують низку переваг, зокрема можливість обробки зображень безпосередньо за допомогою ресурсів самого мобільного пристрою, з використанням переваг архітектурних рішень, що наявні в сучасних процесорах. Також це означає, що користувацькі дані, а саме медіа файли не будуть відправлятись на серверні засоби і цим забезпечується саме приватність даних.

За допомогою створеної порівняльної таблиці з такими програмними продуктами як Apple Photos та Google Photos визначено, що розроблюваний програмний ппродукт має переваги в забезпеченні приватності користувацьких даних, а також матиме додатковий функціонал, який допомагатиме редагувати автоматично агреговані медіа-файли.

1.3 Постановка задачі

Ціллю магістерської кваліфікаційної роботи є розробка методів і засобів розпізнавання зображень для створення медіа-контенту на основі технологій штучного інтелекту. Для її реалізації небхідно здійснити аналітичний огляд предметної області, ідентифікувати проблему агрегування медіа-контенту, провести порівняльний аналіз аналогів, провести аналіз методів та засобів реалізації програмного додатку, розробити алгоритм агрегування медіа-контенту, розробити UML-діаграми для відображення основних функцій програмного продукту, розробити програмний додаток.

Завдання розробки полягає у створенні програмного продукту, у якому отримає розвиток метод розпізнавання зображень для агрегування медіа-контенту, відмінністю якого є використання технологій штучного інтелекту безпосередньо на мобільному пристрої, за рахунок чого буде досягнуто приватність даних

користувача, а також отримає розвиток модель мобільної системи, яка на відміну від відомих моделей трансферного навчання дозволяє скоротити об'єм пам’яті і робоче навантаження мережі при розгортанні на мобільних пристроях.

Для реалізації завдань розробки необхідно у розробленому програмному продукті забезпечити стабільність його роботи, швидку відповідь на дії користувача, зручний графічний інтерфейс користувача, захист приватних даних, можливість агрегування медіа контенту у спогади, можливість редагування згенерованих спогадів, можливість створення спогадів власноруч, можливість створювати резервні копії спогадів у хмарних середовищах.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ МОБІЛЬНОГО ДОДАТКУ ТА АЛГОРИТМУ АГРЕГУВАННЯ МЕДІА-КОНТЕНТУ

2.1 Розробка архітектури мобільного додатку

Для чіткого визначення алгоритму дій користувача у системі необхідно спроектувати діаграму використанням [10]. Розглянемо варіанти використання для мобільного додатку, що відображає взаємодію користувача з застосунком.



Рисунок 2.1 – Діаграма варіантів використання системи

На даній діаграмі можна побачити усі можливі варіанти використання програмного засобу користувачем, зокрема:

- створення резервної копії - додаток має можливість створення резервної копії згенерованих зображень та бази даних додатку на хмарні сервіси в зашифрованому вигляді;

- автоматичне агрегування медіа - створення спогадів автоматичним способом за допомогою алгоритмів мобільного додатку;

- ручне створення спогадів;

- додавання малюнків до спогадів - користувач має можливість створення малюнків у спогадах;

- додавання аудіо файлів до спогадів - користувач має можливість записувати та відтворювати голосові повідомлення та прикріплювати їх до спогадів;

- редагування спогадів - користувач має можливість редагувати як автоматично створені так і створені власноруч спогади, в тому числі додавати до них нові фото та відео матеріали, додавати опис, та змінювати шаблон відображення фотографій на головному екрані;

- перегляд спогадів - перегляд як створених вручну так і автоматично згенерованих спогадів.

Переваги моделі варіантів використання:

- визначає користувачів і межі системи;

- визначає системний інтерфейс;

- зручний для спілкування користувачів з розробниками;

- використовується для написання тестів;

- є основою для написання документації користувача;

- добре вписується в будь-які методи проектування (як об'єктно-орієнтовані, так і структурні).

Добре опрацьована архітектура потрібна всім додаткам і складним, і шаблонним. З її допомогою економиться час, зусилля та гроші. Архітектура мобільних додатків – сукупність рішень, як організувати програму. До неї входять: структурні елементи та інтерфейси, зв'язки між вибраними елементами, загальний стиль програми [11 - 15].

Якісна архітектура означає вигоду: простота та ефективність. Програму з такою архітектурою легше змінювати, тестувати та налагоджувати. Правильна архітектура мобільного додатку має забезпечувати наступні базові питання.

Ефективність: програма виконує поставлені завдання та виконує функції у будь-яких умовах. Система продуктивна, надійна та справляється з усіма навантаженнями.

Гнучкість: вибране рішення легко змінювати, і помилок стає менше. Можна змінити один елемент і це не стане фатальним для інших складових.

Розширюваність: у програму можна додавати скільки завгодно функцій, якщо потрібно.

Масштабованість: час на розробку та доповнення зменшується. Якісна архітектура дозволяє направити розробку у кілька паралельних потоків.

Тестування: додаток легко тестується, а відповідно, зменшується кількість помилок і збільшується його надійність.

Повторне використання: елементи та структуру можна використовувати в інших проектах.

Зрозумілість: код має бути зрозумілим якомога більшою кількістю людей. Над програмою працює багато людей. Хороша архітектура дозволяє новачкам швидко розібратися у проекті.

Для побудови програмного рішення використаємо патерн архітектури, а саме MVC (Model View Controller) [16]. Схема роботи даного патерну зображена на рисунку 2.2.

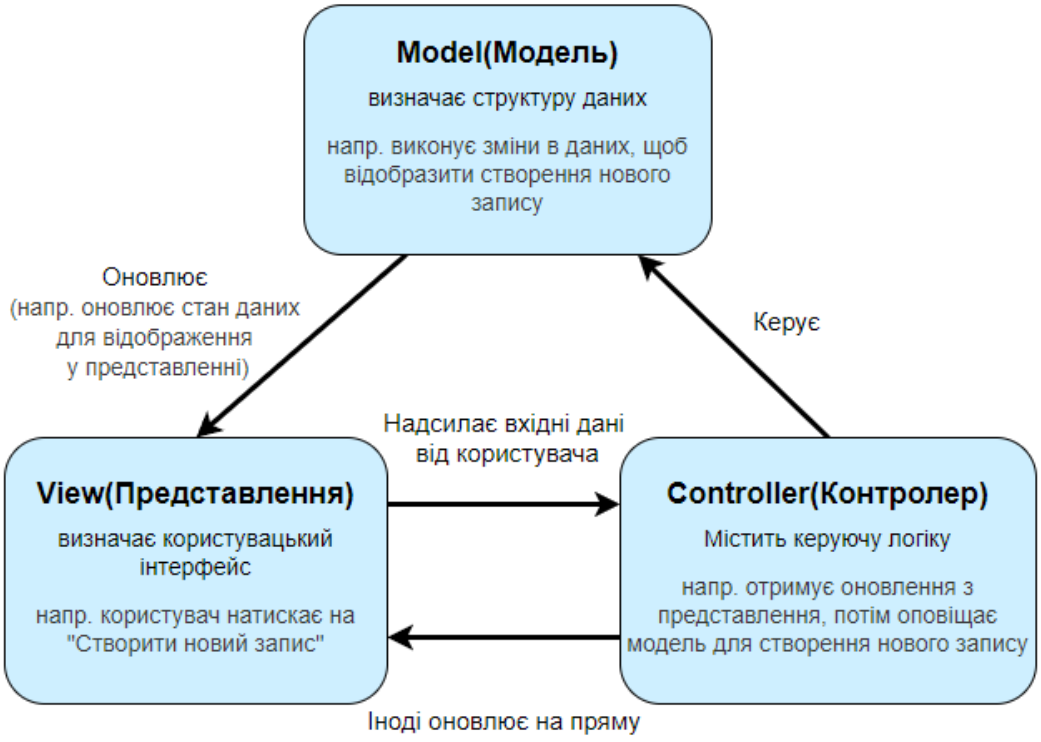


Рисунок 2.2 – Схема роботи Model-View-Controller.

Модель представляє єдине джерело істини, яке передає дані в реальному часі або запити, пов’язані з базою даних. Цей рівень може містити бізнес-логіку, перевірку коду тощо. Цей рівень взаємодіє з ViewModel для локальних даних або для реального часу. Дані надаються у відповідь на ViewModel. ViewModel є посередником між View і Model, який приймає всі події користувача та запитує їх у Model для даних. Коли модель має дані, вона повертається до ViewModel, а потім ViewModel сповіщає ці дані в View. ViewModel може використовуватися кількома представленнями, що означає, що одна ViewModel може надавати дані більш ніж одному View.

Представлення - це місце, де користувач взаємодіє з віджетами, які відображаються на екрані. Ці події користувача вимагають певних дій, які переходять до ViewModel, а решта ViewModel виконує цю роботу. Коли ViewModel має необхідні дані, він оновлює View.

Таким чином для розробки мобільного додатку буде використано архітектурний патерн Model View Controller, що допоможе розбити програму на модулі, і таким чином полегшаться розробка та тестування розроблюваного програмного забезпечення.

Для коректної роботи мобільного додатку необхідно розробити чітку і продуману структуру проєкту. Структура мобільного додатку - це внутрішній устрій застосунку, розташування у ньому сторінок, розділів, підрозділів, додаткових матеріалів. З позиції розробника структуру сайту умовно можна поділити на два рівні: логічний і фізичний [17 - 20].

На логічному рівні структура мобільного додатку є сукупністю сторінок, які об'єднані між собою єдиним дизайном, стилем і посиланнями. Логічне проектування передбачає організацію інформації у додатку, побудову його структури та навігацію по розділах.

На рисунку 2.3 зображено логічну структуру мобільного додатку з інтерактивним інтерфейсом для агрегування медіа контенту.

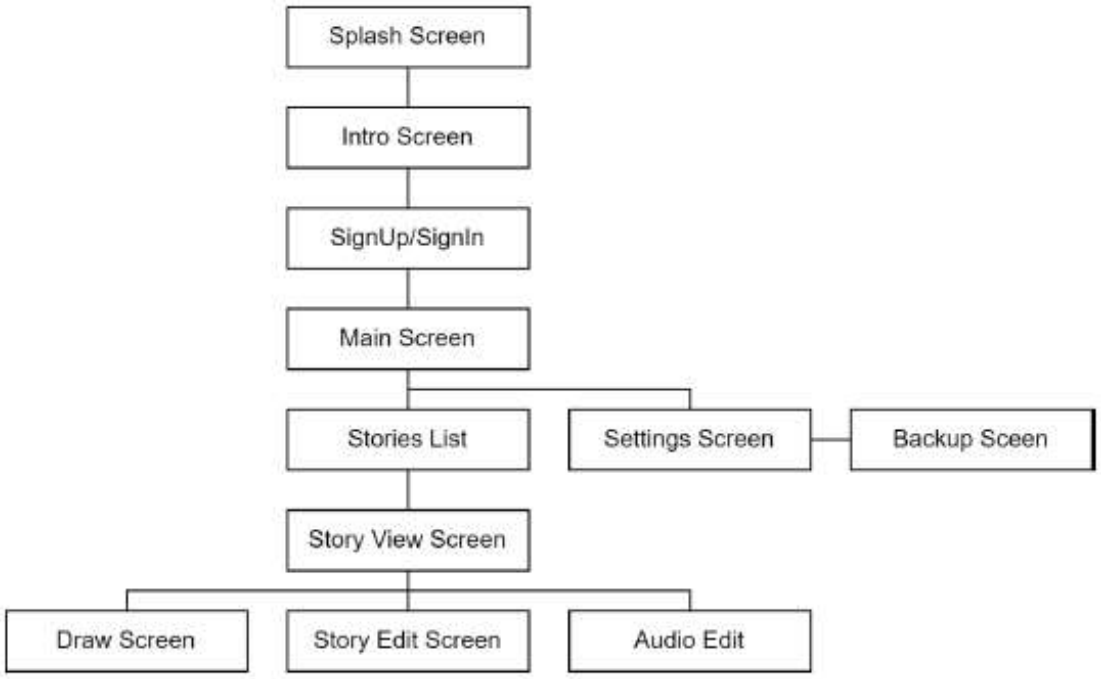


Рисунок 2.3 – Логічна структура мобільного додатку

При проектуванні сторінок мобільного додатку та засобів навігації краще дотримуватися чотирьох принципів, що засновані на сприйнятті інформації:

- використання позначень - застосування слів і термінів, які є сталими і зрозумілими для більшості користувачів додатку, слід розміщувати змістовні елементи, посилання і елементи навігації з їх позначеннями, а також уникати невиразних термінів на екрані;

- доцільність - розділи мобільного додатку повинні містити інформацію та елементи, які відповідають даному розділу чи фрагменту екрану, невідповідні елементи краще перенести в інше місце або взагалі прибрати;

- одноманітність - однакові елементи на екранах повинні мати один і тойже розмір і знаходитися в одному і тому ж місці;

- розділення на частини - користувачі мобільного додатку краще орієнтуються і швидше знаходять потрібні для них матеріали, коли вони візуально поділені на групи.

На фізичному рівні структура додатку є набором файлів різного типу. Для правильної роботи скомпільованого проекту, для уникнення помилок з

завантаженням вбудованих файлів, а також для полегшення роботи розробникам необхідно притримуватись таких правил для створення назв директорій та файлів:

- призначати назви папок, назви і розширення всіх файлів сайту з використанням символів лише латинського алфавіту і лише в рядковому регістрі, що гарантує універсальність і працездатність на різних платформах;

- призначати назви файлів логічними назвами, використовуючи латинський алфавіт, що буде корисно як під час створення сайту, так і коли потрібно буде внести зміни в проект, а також для інших розробників, які працюють над сайтом.

Продумана і зручна файлова структура допомагає розробнику оптимізувати свою роботу, а також буде зрозумілою для інших фахівців, що працюють над проєктом. На основі описаних правил було розроблено фізичну структуру програмного засобу для агрегування медіа-контенту у вигляді мобільного додатку з інтерактивним інтерфейсом, яку наведено на рисунку 2.4.



Рисунок 2.4 – Фізична структура мобільного додатку

Проектний код складається з основних компонентів, таких як:

- resources - використовується для збереження ресурсів додатку, таких як констант, посилань на зображення, файлів локалізації;

- router - містить файли конфігурацій всіх екранів додатку, а також переходів між ними;

- src - головна директорія файлів проекту, що містить базові класи екранів, роботи з репозиторіями, локальними даними, базою даних, містить класи логіки проекту, а також реалізацію всіх можливих екранів.

2.2 Проектування алгоритму агрегування медіа-контенту

Для роботи з зображеннями, на платформі pub.dev є необхідна бібліотека Photo Manager [21]. Даний плагін надає змогу працювати з нативними методами обох платформ, для отримання списку усіх медіа ресурсів пристрою. Тому для роботи алгоритму буде використано дану бібліотеку.

За допомогою вище згаданої бібліотеки ми отримуємо список усіх медіа файлів, що є на пристрої. Далі, в параметрах медіа файлів ми маємо дату їх створення. Використовуємо цю дату для подальшого агрегування. Отримавши список дат медіа файлів, ми можемо поділити їх на умовні спогади. Тобто ділимо список медіа файлів на невеликі списки. За проміжок у цьому списку відповідає часовий інтервал 2 години. Тобто якщо між створеними фотографіями інтервал більше ніж 2 години, то ми вважаємо, що ці фотографії належать до різних спогадів. Якщо проміжок між ними менше то ці фотографії попадають до одного списку. Таким чином ми отримали список спогадів, кожен з яких має декілька медіа-файлів, які необхідно обробити далі. Список проміжків зображено на рисунку 2.5.

Далі нам необхідно створити спогад з кожного проміжку, якщо це можливо. Для цього ми використовуємо алгоритм розпізнавання зображень TensorFlow. За допомогою вже навчаних моделей на кожному зображенні, яке відноситься до проміжку ми намагаємось знайти об’єкти. Список отриманих об’єктів з медіа файлу зображено на рисунку 2.6.

```
2021-09-15 10:41:50.579478 Range 1 = 2021-09-14 18:35:56.000 - 2021-09-14 18:35:56.000
2021-09-15 10:41:50.579483 Range 2 = 2021-09-13 20:47:32.000 - 2021-09-13 20:47:21.000
2021-09-15 10:41:50.579489 Range 3 = 2021-09-13 16:25:57.000 - 2021-09-13 16:25:57.000
2021-09-15 10:41:50.579494 Range 4 = 2021-09-13 07:49:55.000 - 2021-09-13 07:49:55.000
2021-09-15 10:41:50.579500 Range 5 = 2021-09-12 16:17:48.000 - 2021-09-12 16:13:36.000
2021-09-15 10:41:50.579505 Range 6 = 2021-09-12 08:13:47.000 - 2021-09-12 08:13:47.000
2021-09-15 10:41:50.579510 Range 7 = 2021-09-09 18:24:55.000 - 2021-09-09 18:24:55.000
2021-09-15 10:41:50.579515 Range 8 = 2021-09-06 15:37:53.000 - 2021-09-06 15:37:53.000
2021-09-15 10:41:50.579521 Range 9 = 2021-09-04 14:12:14.000 - 2021-09-04 10:45:09.000
2021-09-15 10:41:50.579526 Range 10 = 2021-09-03 09:02:46.000 - 2021-09-03 09:02:46.000
2021-09-15 10:41:50.579531 Range 11 = 2021-09-02 09:58:03.000 - 2021-09-02 08:36:50.000
2021-09-15 10:41:50.579536 Range 12 = 2021-09-01 17:05:40.000 - 2021-09-01 17:05:01.000
2021-09-15 10:41:50.579542 Range 13 = 2021-08-31 15:44:02.000 - 2021-08-31 15:44:02.000
2021-09-15 10:41:50.579547 Range 14 = 2021-08-29 21:40:41.000 - 2021-08-29 21:22:16.000
2021-09-15 10:41:50.579552 Range 15 = 2021-08-29 09:41:52.000 - 2021-08-29 09:41:52.000
2021-09-15 10:41:50.579557 Range 16 = 2021-08-29 00:25:24.000 - 2021-08-29 00:25:24.000
2021-09-15 10:41:50.579562 Range 17 = 2021-08-21 18:07:08.000 - 2021-08-21 18:07:08.000
2021-09-15 10:41:50.579566 Range 18 = 2021-08-15 13:55:51.000 - 2021-08-15 13:55:51.000
```

Рисунок 2.5 – Список проміжків зі спогадами

```
-> started detection
-> -> got image = /var/mobile/Media/PhotoData/CPLAssets/group111/B3B086FD-1C2A-4F34-BA33-88A0B8A479C.HEIC
-> -> -> image saved with milliseconds = 9
-> -> -> Tflite.detectObjectOnImage finished with milliseconds = 178 2021-09-15 10:42:12.877486 and returned 5 detected objects
-> -> -> -> added detected object = DetectedObject{name: laptop, confidence: 0.6796875}
-> -> -> -> added detected object = DetectedObject{name: person, confidence: 0.64453125}
-> -> -> -> added detected object = DetectedObject{name: potted plant, confidence: 0.5859375}
-> -> -> -> added detected object = DetectedObject{name: dining table, confidence: 0.5234375}
-> -> -> -> added detected object = DetectedObject{name: tv, confidence: 0.5}
```

Рисунок 2.6 – Список знайдених об’єктів

У списку знайдених об’єктів ми бачимо назву об’єкту та вірогідність того, що цей об’єкт є на цьому зображені. Таким чином ми можемо зрозуміти чи є на даному медіа файлі об’єкти і чи варто включати даний медіа файл у спогад. Якщо медіа файл взагалі не має жодного об’єкту, то можна вважати що він непотрібний. Далі використовуємо таблицю порогових значень для кожного об’єкта. Використавши цю таблицю ми бачимо які об’єкти і які мінімальні порогові значення для них є. Тобто, якщо на медіа файлі знайдено об’єкт «Apple» з вірогідністю 0.7, а порогове значення у таблиці це 0.6, то ми можемо вважати, що даний медіа файл можна використати для спогаду. Якщо б вірогідність була меншою, то даний файл було б відкинуто як не потрібний.

Відфільтрувавши таким чином медіа файли ми отримаємо зображення, з яких можна далі генерувати спогад. Для головного зображення спогаду буде

використано медіа файл, у якого найбільша вірогідність знайденого об’єкту. Далі необхідно виконати створення назви даного спогаду. Для цього ми використовуємо медіа файл, який ми обрали як головний і переходимо до його обробки. Серед мета даних медіа файлу також можна знайти дані геолокації, де було створено медіа файл. Якщо такі дані знайдено, то використовуючи сервіси Google знаходимо назву місця за координатами.

Далі використовуємо такі дані, як час створення медіа файлу, геолокація, об’єкти, знайдені на зображенні. За допомогою часу, у який було створено медіа файл визначаємо першу частину назви, за таблицею 2.1.

Таблиця 2.1 – Таблиця визначення частини дня

Час	Назва
01:30 – 04:00	Early morning
04:00 – 06:00	Morning
06:00 – 10:30	Late morning
10:30 – 12:00	Early afternoon
12:00 – 13:00	Afternoon
13:00 – 15:00	Late afternoon
15:00 – 16:00	Early evening
16:00 – 18:30	Evening
18:30 – 21:30	Late evening
21:30 – 23:00	Night
23:00 – 01:30	Late night

За допомогою знайдених об'єктів на медіа файлі можна визначити категорію спогаду, яка піде до назви. За допомогою таблиці 2.2 визначаємо назву категорії та додаємо її до назви спогаду.

Таблиця 2.2 – Таблиця категорій визначених об’єктів

Назва категорії	Список об’єктів
Travel	airplane, backpack, suitcase
Nature	bear, bird, giraffe, sheep, zebra
Sport	baseball bat, baseball glove, frisbee, skateboard, sports ball, tennis racket, skis, snowboard
Meal	apple, banana, bottle, bowl, broccoli, cake, carrot, dining table, donut, fork, hot dog, knife, orange, pizza, sandwich, spoon, wine glass
Indoors	bed, couch
Outdoor	bench, bicycle, boat, bus, car, cow, elephant, fire hydrant, handbag, horse, kite, motorcycle, parking meter, stop sign, traffic light, train, truck
Home	book, chair, clock, hair drier, microwave, potted plant, refrigerator, scissors, teddy bear, toilet, toothbrush, vase
Pets	cat, dog
Gadgets	cell phone, keyboard, laptop, mouse, remote
Kitchen	oven, sink
People	person
Clothes	tie

За допомогою цих даних можемо згенерувати назву спогаду за таким алгоритмом: [Частина дня] + [Категорія] + [Назва локації] + [Місяць, рік]. Блок схему алгоритму зображено на рисунку 2.7.

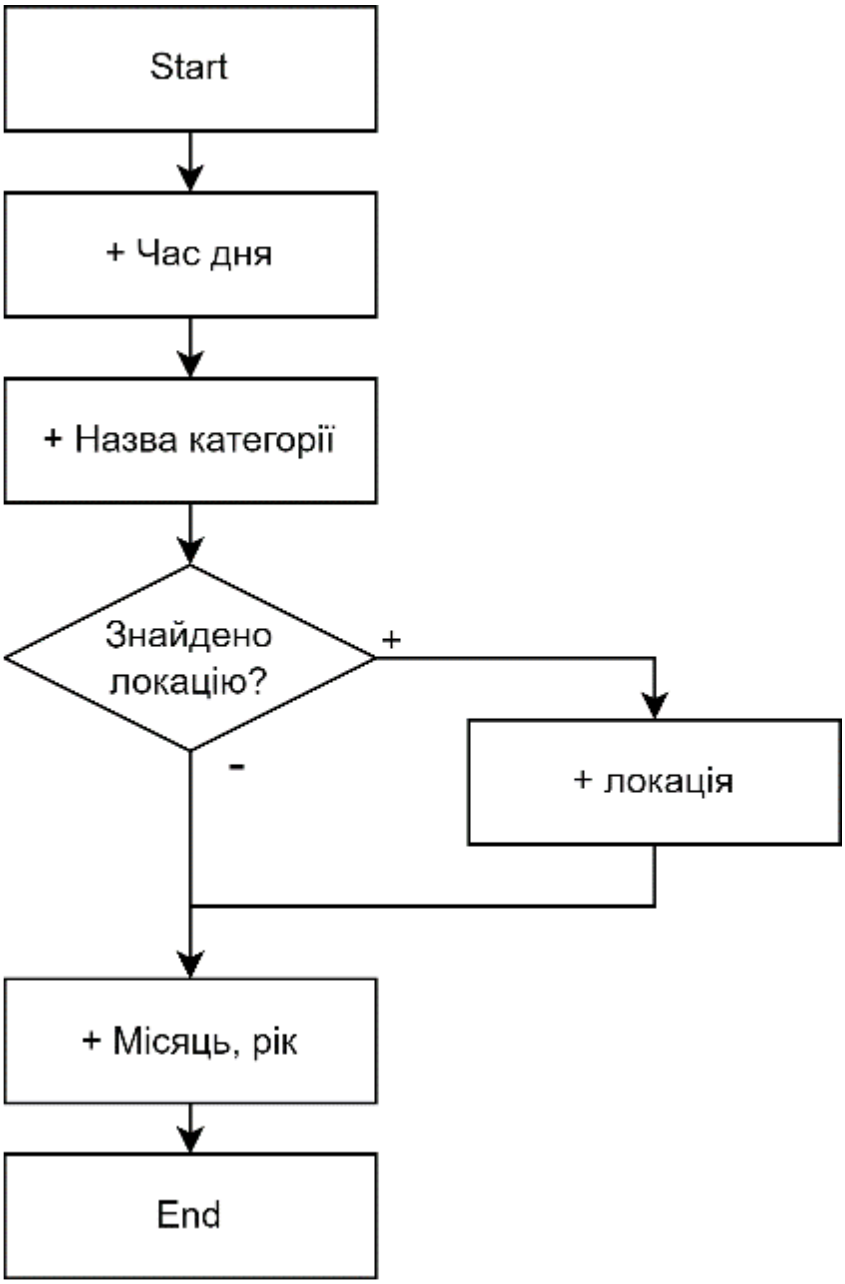


Рисунок 2.7 – Схема роботи Model-View-Controller.

Модель представляє єдине джерело істини, яке передає дані отримання в реальному часі або запити, пов’язані з базою даних. Цей рівень може містити бізнес-логіку, перевірку коду тощо. Цей рівень взаємодіє з ViewModel для локальних даних або для реального часу. Дані надаються у відповідь на ViewModel. ViewModel є посередником між View і Model, який приймає всі події користувача

та запитує їх у Model для даних. Коли модель має дані, вона повертається до ViewModel, а потім ViewModel сповіщає ці дані в View.

ViewModel може використовуватися кількома представленнями, це означає, що одна ViewModel може надавати дані більш ніж одному View. Представлення - це місце, де користувач взаємодіє з віджетами, які відображаються на екрані. Ці події користувача вимагають певних дій, які переходять до ViewModel, а решта ViewModel виконує цю роботу. Коли ViewModel має необхідні дані, він оновлює View.

Таким чином для розробки мобільного додатку буде використано архітектурний патерн Model View Controller, що допоможе розбити програму на модулі, і таким чином полегшиться розробка та тестування розроблюваного програмного забезпечення.

Для збереження згенерованих спогадів буде використовуватись база даних, яку в разі необхідності можна зберегти разом з медіа файлами на хмарному сховищі таким чином отримавши резервну копію згенерованих спогадів. Для збереження даних локально буде використано Sqflite [22 - 26], оскільки вона має реалізацію для платформ iOS та Android.

SQLite - це система баз даних, написана мовою програмування C. Це не окрема програма; скоріше це бібліотека, яку розробники програмного забезпечення вбудовують у свої програми. Таким чином, він належить до сімейства вбудованих баз даних.

Дана СУБД є досить легкою і не потребує великої продуктивності, що досить важливо коли розробка ведеться для мобільних пристроїв. Проте вона також має свої недоліки, зокрема вона не має всіх можливостей, які має звичайний SQL, відрізняється за своїм синтаксисом та обмежена в типах підтримуваних даних. Проте всі ці недоліки не є критичними для розроблюваного програмного продукту.

Для збереження даних історії було створено таблицю Story. Вона містить наступні поля:

- id;

- date_Start;

- date_General;

- date_End;

- main_Image;

- canvas_File;

- audio_File;

- data.

Поле data пов’язано зв’язком один до одного з таблицею StoryData, що містить наступні поля:

-id;

-story_id;

-medias.

Для збереження даних про таги було створено таблицю Tag, вона має наступні поля:

- id;

- name.

Вона має зв’язок з таблицею Story через суміжну таблицю Story_Tag, оскільки інакше матиме зв’язок багато до багатьох.

Для збереження даних про локації було створено таблицю Location, яка пов'язана з таблицею Story зв’язком один до одного та містить наступні поля:

- id;

- latitude;

- longitude;

- name;

- story_id.

Таким чином обравши СУБД за допомогою якої буде реалізовано збереження даних приступимо до створення схеми бази даних.

В результаті отримаємо схему бази даних, яка наведена на рисунку 2.8.

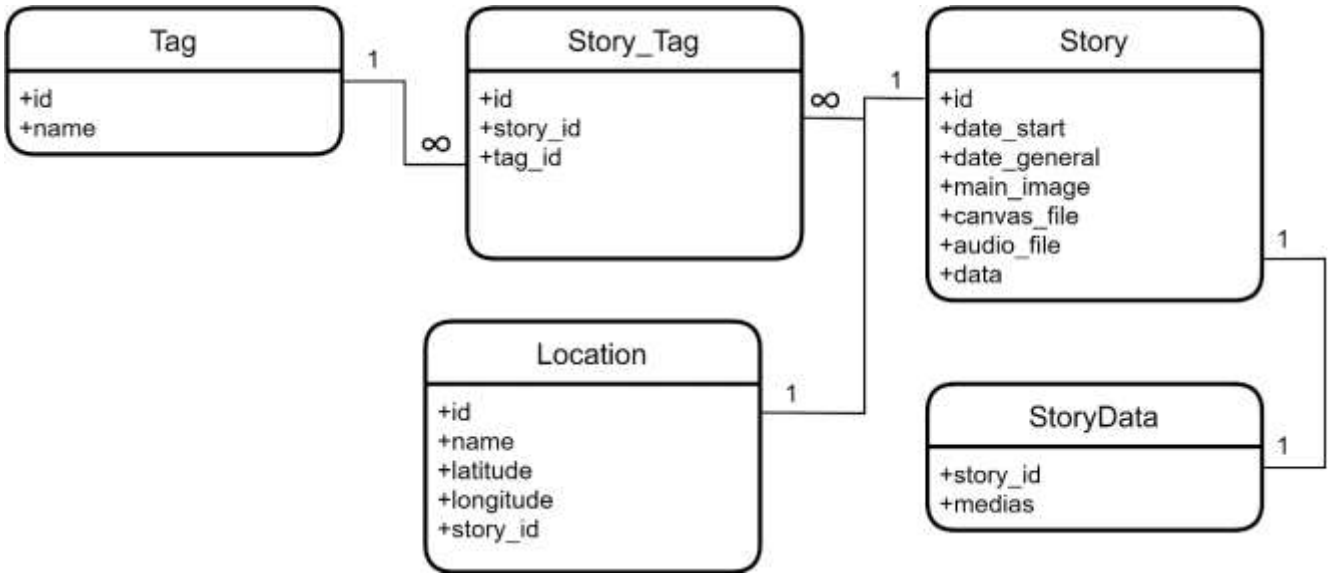


Рисунок 2.8 – Розроблена схема бази даних

2.3 Розробка структури екранів та дизайну мобільного додатку

Структура додатку - екрани, розділи, навігація (посилання); розташування цих елементів і їх зв'язок між собою. Для того, щоб зробити структуру додатку наочною, можна використовувати нескладні геометричні фігури та написи [27]. Розміщуючи елементи на робочій області і стрілочками з'єднавши їх, можна отримати схему як відвідувач повинен переходити по додатку.

При розробці структури додатку важливо знати той факт, що екрани додатку можуть бути різного ступеня вкладеності. Різниця рівня визначається ступенем важливості, як розділ і підрозділ.

Найпростіша структура додатку - лінійна. Екрани йдуть один за одним, і користувач повинен переглядати їх як слайд-шоу. У лінійній структурі не існує поділу контенту на рівні. Всі екрани на таких додатках рівноправні, і їх повинен побачити кожен відвідувач. Незважаючи на простоту реалізації лінійної структури, недоліків у неї набагато більше, ніж переваг. Реалізація лінійної структури не є складною. Найпростіший варіант додатку - набір екранів, кожен з яких має посилання на наступну та попередню сторінку. Але при цьому на кожній сторінки обов'язково має бути заголовок і посилання на першу сторінку. Інакше відвідувачі,

що потрапили в середину сайту, наприклад, з пошукової системи, нічого не зрозуміють. Орім того, дуже корисним є представлення загального числа екранів і певне виділення номеру того з них, на якому користувач перебуває в поточний момент.

Лінійна структура додатку з альтернативами і варіантами. Основою цієї структури є просте лінійне розміщення екранів. Однак у додатках, побудованих за цим принципом, відвідувачі можуть проявити деяку ініціативу, полегшивши для себе пошук потрібної інформації. Під альтернативами в даному випадку розуміється вибір між двома гілками. Найчастіше подібна структура використовується для збору інформації про відвідувача. Лінійна структура з альтернативами і варіантами зручна в тому плані, що з одного боку вона дозволяє інженеру контролювати діяльність відвідувачів, направивши їх в певне русло. А з іншого боку проявити деяку ініціативу, яка дозволить їм, по-перше, відчути свободу, а по-друге, полегшити доступ до потрібної саме їм інформації.

Лінійна структура сайту з відгалуженнями. Це теж контрольована структура, яка нагадує дорогу з тупиковими стежками, які час від часу відгалужуються від неї. Тобто відвідувач послідовно переходить з однієї сторінки на іншу. Якщо інформація, розміщена на якийсь із них його зацікавила, і він хоче дізнатися подробиці, то може перейти на відгалуження, а потім повернутися назад на основну «доріжку». Головною перевагою даної структури є те, що до неї легко перейти зі звичайного лінійного розміщення екранів. Таке часто буває, коли створений одного разу сайт перестає задовольняти збільшеним вимогам, а глобальна переробка з тих чи інших причин неможлива. В цьому випадку інженер може швидко і без всяких проблем розширити проект.

Деревоподібна структура додатку - універсальний спосіб розміщення екранів. Вона підходить для створення практично будь-яких типів додатків [28 - 30]. Її принцип полягає в тому, що користувач при заході на головну сторінку виявляється перед вибором, куди йти далі. Після переходу в потрібний розділ, він підбирає необхідний підрозділ і т.п. У деревовидної структури дуже багато переваг,

але так само є і недоліки. Йдеться про те, що в структурі дерева дуже складно дотримуватися балансу між глибиною і шириною. Якщо «дерево» додатку буде рости тільки вглиб, то користувачам, щоб дійти до якоїсь інформації, доведеться завантажити і обробити багато екранів, що буде дратувати користувачів. А якщо створити дуже широку деревоподібну структуру, то відвідувачі будуть змушені щоразу витрачати дуже багато часу для вибору потрібної їм гілки.

Таким чином, при використанні деревовидної структури додатку необхідно постійно стежити за її розростанням і дотримуватися золотої середини.

Гратчаста структура вже на порядок складніше усіх розглянутих вище. У ній усі екрани також розміщуються в різних гілках. Але у користувача є можливість переміщатися по ним не тільки вертикально (вгору - вниз), а й горизонтально (тобто між гілками на різних рівнях). Використовується решітка в основному тільки в каталогах. При цьому переміщення між гілками на глибинних рівнях здійснюється за допомогою відсилань на рубрики в інших розділах. Використання гратчастої структури в інших проектах недоцільне. По-перше, вона відносно складна в реалізації. По-друге, звертатися з «гратами» потрібно з дуже великою обережністю, так як структура додатку може вийти дуже заплутаною, і відвідувачі будуть змушені довго блукати в пошуках потрібної їм інформації.

Розробка структури інтерфейсу є одним з основних етапів ескізних проектів, які робляться перед виготовленням оригінал-макету мобільного додатку, і використовується при розробці дизайну мобільних додатків та сайтів, щоб показати структуру мобільного додатку, налаштовуються схеми шляху користувача, найбільш важливі елементи інтерфейсу користувача, їх положення і взаємозв'язок між сторінками сайту [31]. Каркаси сторінок сайту відображають в чорно-білому кольорі - найбільш важливі елементи інтерфейсу, такі як заголовок і нижній колонтитул сайту, форма контактів, навігація і т.п.

Каркасне моделювання - це процес, який може значно скоротити час, необхідний для проектування і розробки, усуваючи потенційні візуальні

відволікаючі фактори і фокусуючи увагу розробників проекту на базовій функціональності і стратегічних факторах маркетингу.

Існує кілька ключових причин, що визначають, чому перед створенням сайту бажано (обов'язково) потрібно спочатку зробити його прототип:

- це дозволить отримати чітку картину того, яка саме інформація буде необхідна на кожній сторінці сайту до розробки його дизайну;

- це дозволить раціонально витрачати час і зосереджуватися саме на тому, для чого призначена кожна сторінка. Ретельне планування має першорядне значення;

- це дозволить розробнику підстрахувати себе, в разі необізнаних клієнтів, які схильні змінювати свою думку на стадії розробки проекту. Якщо сайт належним чином планується на етапі каркасного моделювання, то функціональність окремих сторінок не повинна сильно змінитися;

- це дозволить встановити точку відновлення. Коли клієнт підписує каркасну модель, це означає, що він погоджується з тим, що буде перебувати на цій сторінці;

- обізнаність клієнта також грає тут важливу роль, і він повинен знати, що зміна вже підписаних елементів на каркасній моделі може з дуже великою ймовірністю збільшити бюджет;

- це дозволить отримати чітке уявлення про те, як будуть реагувати відвідувачі на сайт без колірної схеми або елементів дизайну;

- це дозволить видалити зайві елементи, які можуть виявитися непотрібними для майбутнього сайту;

- прототип досить легко створити, дозволяючи плавно і ефективно здійснювати процес планування;

- цей процес знижує ймовірність збільшення обсягу роботи з розробки дизайну;

- це надає дизайнеру чітке уявлення про те, що потрібно зробити;

- це дозволяє інтенсивно залучити клієнта в процес планування на ранньому етапі розробки сайту і дозволяє активно погоджувати весь процес планування між обома сторонами.

Користувацький інтерфейс застосунку має бути зручним для використання. Перш за все, інтерфейс має бути простим. Найкращі інтерфейси практично непомітні для користувача. Вони уникають непотрібних елементів і чітко викладають необхідну інформацію у повідомленнях застосунку.

В інтерфейсі має бути узгодженість елементів. Використовуючи загально прийняті, типові елементи інтерфейсу, користувачі відчуватимуть себе комфортніше та зможуть виконати бажані операції швидше, адже вони вже будуть знайомі з різними елементами інтерфейсу та приблизно знатимуть чого очікувати. Також важливо створювати шаблони в мові, макеті та дизайні різних функціональних елементів застосунку, щоб підвищити їх ефективність [32 - 35].

Макет різних сторінок інтерфейсу має бути цілеспрямованим, в кожного елемента має бути чітке призначення і причина його місцезнаходження. Елементи інтерфейсу які часто використовуються користувачем, слід розташувати знизу, адже при використанні мобільних пристроїв, користувачі найчастіше виконують дії саме великим пальцем руки. Ретельне розміщення елементів також може допомогти привернути увагу до найважливіших частин інформації, а також спростити її сприйняття.

Інтерфейс застосунку має швидко відповідати на дії користувача та завжди показувати що саме відбувається в даний момент. Велика кількість прихованих процесів та операцій зменшують зрозумілість та ускладнюють сприйняття інформації в застосунку. Усе залежить від того на скільки розробник розуміє своїх користувачів, включаючи розуміння їхніх цілей, навичок, уподобань та тенденцій, що неможливо ідеально зробити одразу. Цілком розумно в ході експлуатації застосунку користувачами отримувати від них відгуки які сприятимуть позитивним змінам інтерфейсу в майбутньому.

Опираючись на фактори, описані вище приступаємо до роботи над інтерфейсом програмного продукту. Для розробки макету дизайну додатку буде використано програмне рішення Figma [36 - 38]. Figma - векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи,

що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський додаток на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач.

Було розроблено стартові екрани, що наведені на рисунку 2.9, де зображено логотип додатку на стартовому вікні.

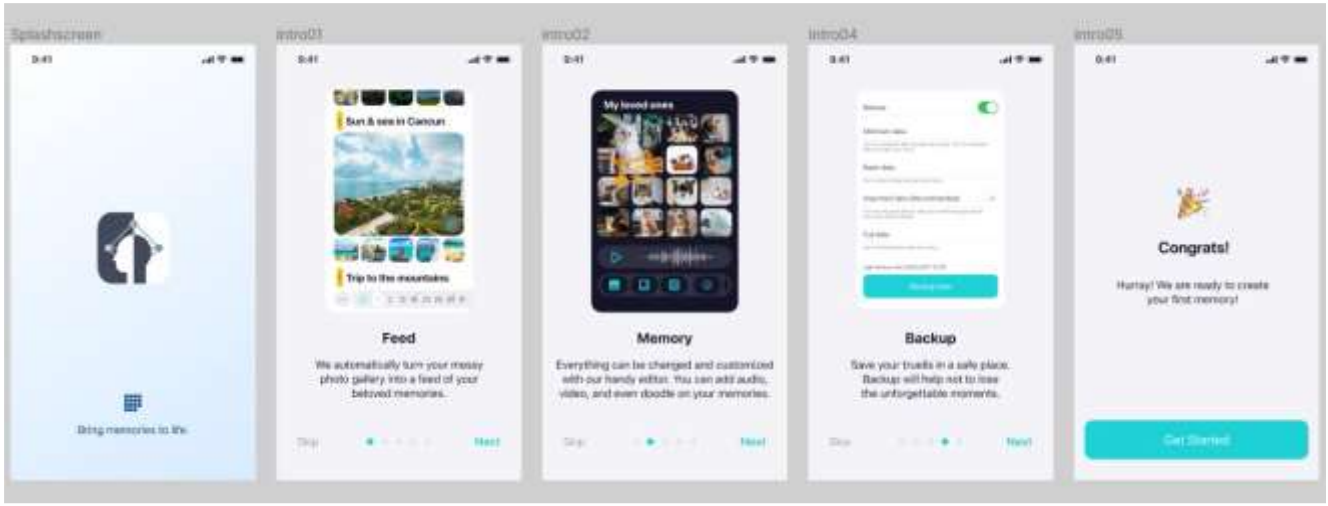


Рисунок 2.9 – Дизайн стартових екранів

Екран привітання нового користувача, що реалізований у вигляді слайдера з зображеннями, на яких користувач може побачити можливості застосунку. Розроблений дизайн інтерфейсу головного екрану наведений на рисунку 2.10. Наголовному вікні користувач матиме змогу переглядати вже агреговані медіа файли, а також запускати нове сканування галереї. З цього екрану користувач може перейти на всі інші екрани додатку, такі як перегляд історій, налаштування, резервне копіювання та створення спогадів власноруч. Також на цьому екрані присутній календар та карта. За допомогою календаря користувач зможе переглянути в які дні він має згенеровані спогади та легко перейти до потрібної дати. На екрані карти зображені усі спогади, у яких було знайдено локацію у метаданих медіа-файлів, натиснувши на зображення спогаду на карті користувач перейде на екран перегляду даного спогаду. Також на головному екрані користувач може додати тег до спогаду. Таким чином він зможе здійснювати пошук не лише по назві спогадів, а й фільтрувати спогади по тегах, наприклад, здійснивши

фільтрування, по тегу «Сім’я» користувач отримає список усіх спогадів, що прив’язані до даного тегу.

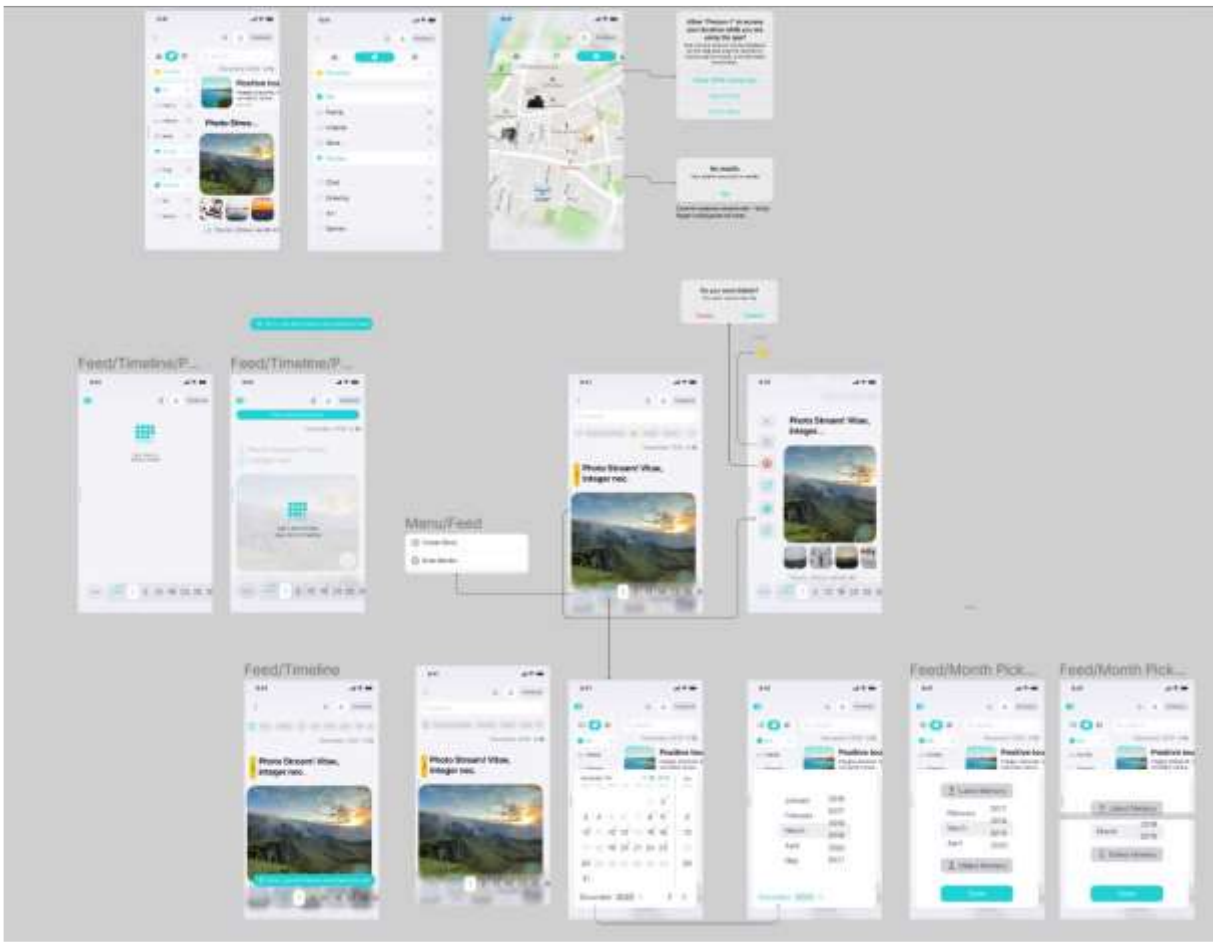


Рисунок 2.10 – Дизайн інтерфейсу головного екрану

Екрани редагування спогадів, створення малюнків та створення аудіо-запису наведені на рисунку 2.11. На екрані редагування спогадів користувач матиме змогу здійснити наступний функціонал:

- зміна медіа контенту місцями;
- додавання назви спогаду;
- додавання опису до спогаду;
- додавання нових фото та відео матеріалів до спогаду;
- перехід на екран додавання аудіо фалу;
- перехід на екран додавання малюнку;
- зміна головного шаблону;
- зміна шаблонів комірок головного шаблону.



Рисунок 2.11 – Дизайн екрану редагування спогадів

На екрані створення малюнку користувач матиме змогу власноруч створити малюнок, який буде відображено поверх головного шаблону спогаду. Для малювання користувач матиме змогу змінювати колір курсору, а також стирати ділянки малюнку за допомогою ластику.

На екрані додавання аудіо файлу користувач матиме змогу записати аудіо повідомлення за допомогою мікрофону, або завантажити аудіо файл з мобільного пристрою. Цей аудіо файл буде прикріплений до спогаду, та його можна буде відтворити на екрані редагування або на екрані перегляду спогаду. Екран перегляду спогадів наведено на рисунку 2.12.

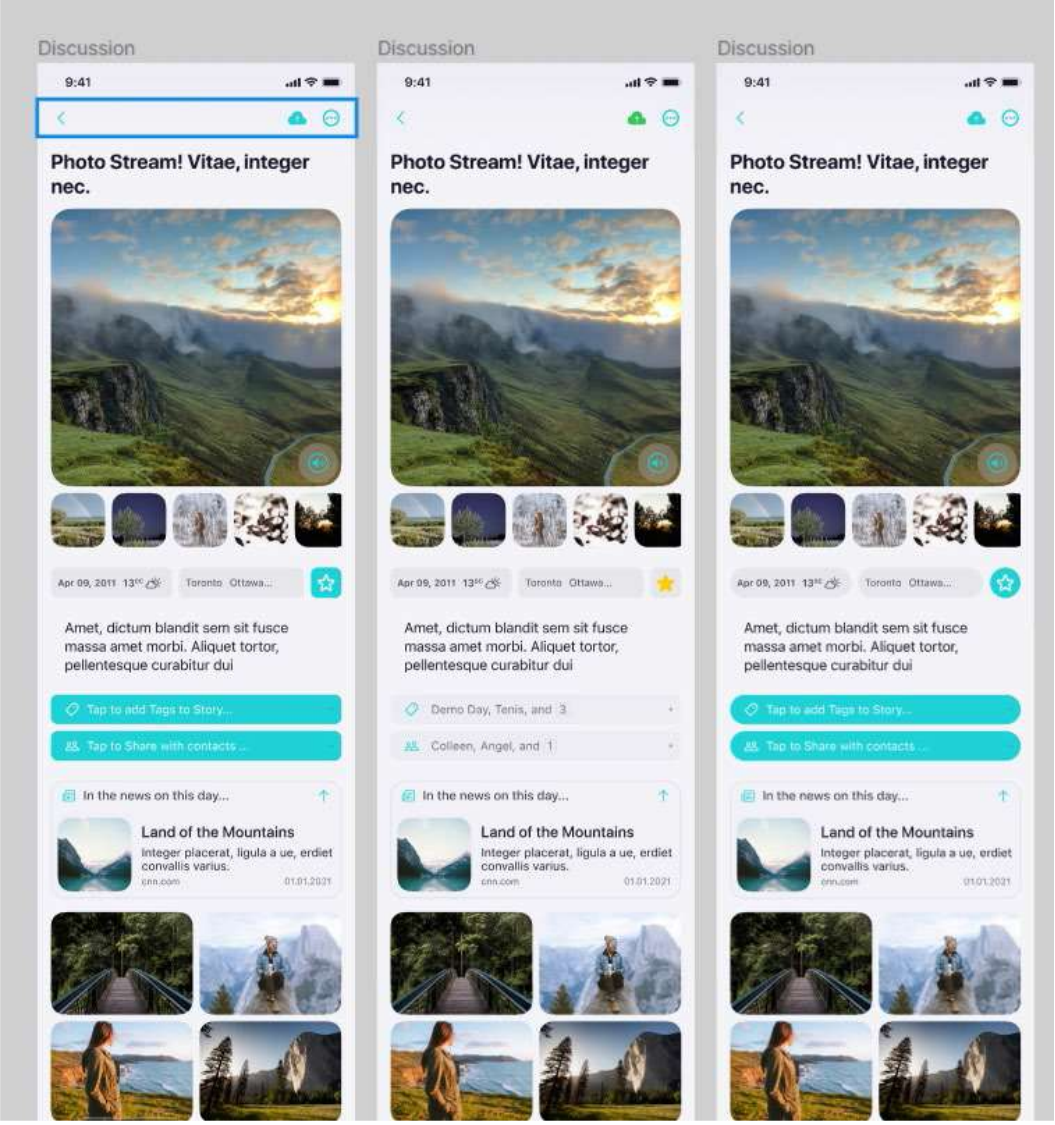


Рисунок 2.12 – Дизайн екрану повного перегляду спогаду

На даному екрані користувач може переглядати спогади згенеровані додатком, або створені власноруч. На даному екрані присутній функціонал зміни дати історії та зміни локації. Користувач має можливість перегляду фото та відео матеріалів у повноекранному режимі за допомогою екрану слайдеру, а також з даного екрану користувач може додати тег до спогаду або перейти на екран редагування даного спогаду.

Дизайн вікна, призначеного для створення тегів, зображено на рисунку 2.13. Дизайн екрану верифікації електронної пошти, номера телефону та введення пін-коду для входу в мобільний додаток наведений на рисунку 2.14.

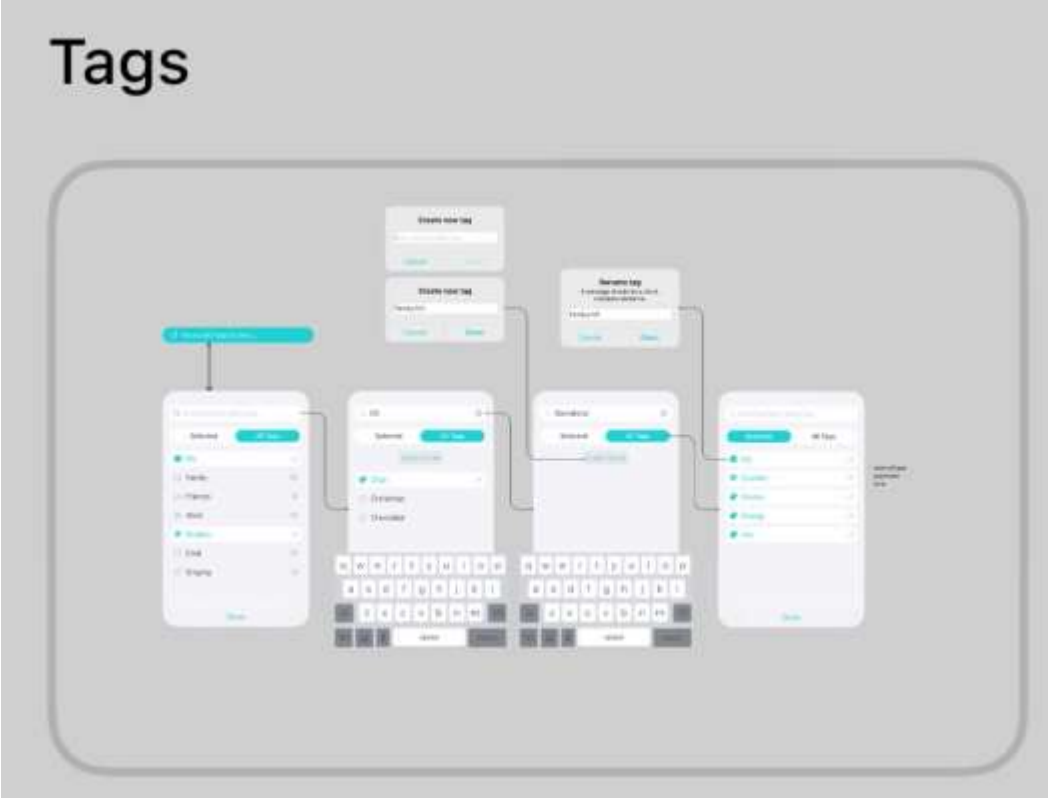


Рисунок 2.13 – Дизайн вікна створення тагів

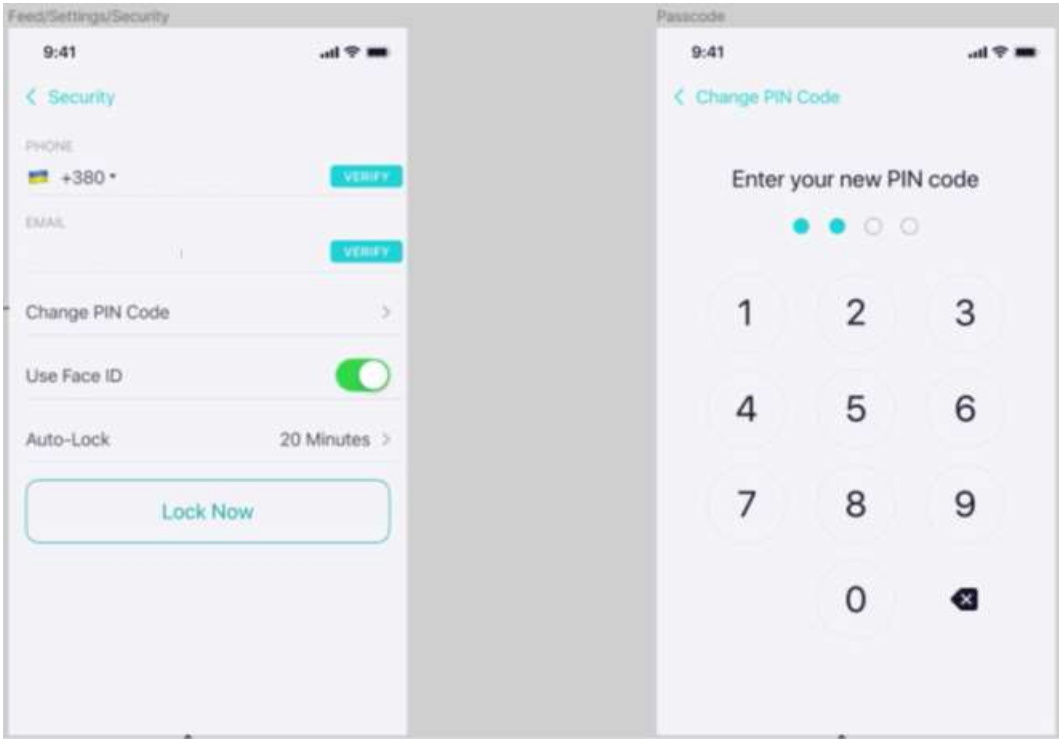


Рисунок 2.14 – Дизайн екранів верифікації електронної пошти, номера телефону та екрану введення пін-коду для входу в додаток

Таким чином було розроблено інтерфейс мобільного додатку, можна приступати до його програмної реалізації. Визначено мову програмування та програмні засоби за допомогою яких буде вестись розробка програмного додатку, а також визначено патерн програмування за допомогою якого буде створено ядро системи. Проведено проектування алгоритму створення спогадів за допомогою аналізу зображень та агрегування медіа-контенту. Визначено ключові параметри за якими буде проведено відбір медіа-контенту, який попаде до вибірки спогадів, а також створено алгоритм створення назви спогаду за допомогою таких параметрів як, час створення, локація та знайдені об’єкти на зображенні. Створено структуру бази даних для зберігання спогадів та подальшого створення резервної копії. Спроектовано інтерфейс системи за допомогою програмного засобу Figma, розроблено логотип, дизайн стартового, головного екрану, а також екранів редагування та перегляду спогадів.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Аналіз засобів реалізації програмного продукту

На основі проведеного у першому розділі аналізу методів розв’язання поставленої задачі, для створення мобільного додатку було обрано програмний засіб Flutter. На сьогоднішній день у розробників існує широкий вибір програмних засобів для створення мобільного додатку. Для вибору засобів реалізації кваліфікаційної роботи магістра розглянемо найвідоміші способи реалізації мобільних додатків. В першу чергу необхідно визначити, що однією з задач кваліфікаційної роботи магістра є створення мобільного додатку, який може бути доступним для основних сучасних операційних систем, призначених для смартфонів, а саме iOS та Android. Тому для пришвидшення розробки було надано перевагу до кросплатформних програмних засобів, оскільки їх використання забезпечує використання однієї програмної бази, яку можливо скомпілювати під обидві платформи. Серед таких програмних засобів можна виділити наступні:

Kotlin Multiplatform. Технологія Kotlin Multiplatform розроблена для спрощення розробки кросплатформних проектів [39 - 42]. Це скорочує час, витрачений на написання та підтримку того самого коду для різних платформ, зберігаючи при цьому гнучкість і переваги нативного програмування. Спільне використання коду між мобільними платформами є одним із основних варіантів використання Kotlin Multiplatform. За допомогою Kotlin Multiplatform Mobile є можливість створювати кросплатформні мобільні додатки та спільно використовувати спільний код між Android та iOS, наприклад бізнес-логіку, підключення тощо. Загальний код, написаний на Kotlin, можна впровадити з мінімальними зусиллями у готовий додаток, компілюється у звичні для платформ бібліотеки. Для Android це jar чи aar-бібліотека, для iOS — Universal Framework. Підключення та подальша робота не сильно відрізняються від взаємодії з будь-якою нативною бібліотекою. Синтаксис мови Kotlin близький до Swift. Схожість мов знижує поріг входу для iOS-розробників. Обидві мови поділяють схожу

ідеологію – швидкість та зручність роботи для розробника. Зрозуміти, що відбувається у загальному коді, і доповнити його зможе будь-хто в команді.

Недоліком даного підходу є його відносно не великий час на ринку, таким чином підхід має не велику кількість підтримки спільноти, а також доволі велику кількість помилок, оскільки він ще не встиг протестуватись спільнотою як слід.

React Native. React Native (також відомий як RN) — це популярна платформа для мобільних додатків на основі JavaScript, яка дозволяє створювати нативно відтворені мобільні додатки для iOS та Android [43]. Фреймворк дозволяє створювати програми для різних платформ, використовуючи ту саму кодову базу. React Native був вперше випущений Facebook як проект з відкритим вихідним кодом у 2015 році. Всього за пару років він став одним із найкращих рішень для мобільної розробки. Розробка React Native використовується для роботи деяких провідних мобільних застосунків, зокрема Instagram, Facebook і Skype. Є кілька причин глобального успіху React Native. По-перше, використовуючи React Native, компанії можуть створити код лише один раз і використовувати його для роботи своїх додатків для iOS і Android. Це означає величезну економію часу та ресурсів. По-друге, React Native було створено на основі React – бібліотеки JavaScript, яка вже була надзвичайно популярною на момент випуску мобільного фреймворку. По-третє, фреймворк надав можливість розробникам інтерфейсу, які раніше могли працювати лише з веб-технологіями, створювати надійні, готові до виробництва програми для мобільних платформ. Цікаво, що, як і багато інших революційних винаходів, React Native був розроблений як відповідь на технологічну помилку. React Native був успішно прийнятий сотнями компаній у всьому світі, включаючи Uber, Microsoft і Facebook, і використовується в багатьох галузях. React Native дозволяє веб-розробникам створювати мобільні програми, які виглядають і відчуваються «рідними», не виходячи з бібліотеки JavaScript (JS), React. Ця простота використання зробила React Native одним із найпопулярніших мобільних фреймворків, але це також є причиною деяких внутрішніх проблем із продуктивністю.

Flutter. Flutter - це набір програмного забезпечення для розробки інтерфейсу користувача з відкритим вихідним кодом, створений Google [44]. Він використовується для розробки кросплатформних додатків для Android, iOS, Linux, macOS, Windows, Google Fuchsia та Інтернету з єдиної кодової бази. Вперше анонсований у 2015 році, Flutter був випущений у травні 2017 року. Перша версія Flutter була відома як "Sky" і працювала на операційній системі Android. Він був представлений на саміті розробників Dart у 2015 році. Під час основної доповіді Google Developer Days у Шанхаї у вересні 2018 року Google оголосила Flutter Release Preview 2, останній великий випуск перед Flutter 1.0. 4 грудня того ж року на заході Flutter Live був випущений Flutter 1.0, що означає першу стабільну версію фреймворку. 11 грудня 2019 року на заході Flutter Interactive був випущений Flutter 1.12. 6 травня 2020 року було випущено набір для розробки програмного забезпечення Dart версії 2.8 і Flutter 1.17.0, додавши підтримку Metal API, який покращує продуктивність на пристроях iOS приблизно на 50%, а також нові віджети Material і мережеве відстеження. інструменти розробки. 3 березня 2021 року Google випустив Flutter 2 під час онлайн-заходу Flutter Engage. Це велике оновлення принесло офіційну підтримку веб-додатків із новим рендерером CanvasKit та веб-віджетами, підтримкою настільних програм раннього доступу для Windows, macOS та Linux та покращеними API-інтерфейсами Add-to-App. У цій версії також використовувався Dart 2.0, який забезпечував звукову нуль-безпеку, що спричинило багато критичних змін і проблем із багатьма зовнішніми пакетами; однак команда Flutter включила інструкції та інструменти для пом’якшення цих проблем. 8 вересня 2021 року Google випустила Dart 2.14 і Flutter 2.5. Оновлення принесло покращення в повноекранний режим Android і останню версію Material Design від Google під назвою Material You. Dart отримав два нових оновлення, позначаючи підтримку Apple Silicon як стабільну. Поточна стабільна версія Flutter - 3.3.9, а версія Dart - 2.18.5.

Основні компоненти Flutter включають:

- Dart платформа;

- Рушій Flutter (Skia Graphics Engine);

- Базова бібліотека;

- Спеціальні віджети для дизайну;

- Допоміжні засоби розробки Flutter DevTools.

Під час написання та налагодження програми Flutter працює у віртуальній машині Dart, яка має механізм виконання «якраз вчасно» (англ. “just in time”). Це забезпечує швидкий час компіляції, а також «гарячого перезавантаження» (англ. “hot reload”), за допомогою якого можна вносити зміни до вихідних файлів у запущену програму. Flutter розширює це ще більше за допомогою підтримки гарячого перезавантаження з урахуванням стану, коли в більшості випадків зміни вихідного коду відразу відображаються в запущеній програмі без необхідності перезавантаження або втрати стану. Для кращої продуктивності можна випускати версії додатків Flutter на всіх платформах за допомогою завчасної компіляції. Рушій Flutter, написаний переважно на C++, забезпечує низькорівневу підтримку візуалізації за допомогою графічної бібліотеки Skia від Google. Крім того, він взаємодіє з пакетами SDK для певної платформи, такими як Android та iOS. Flutter Engine - це портативна програма для розміщення програм Flutter. Він реалізує основні бібліотеки Flutter, включаючи анімацію та графіку, файловий і мережевий ввід-вивід, підтримку доступності, архітектуру плагінів, а також ланцюг інструментів Dart для виконання та компіляції. Більшість розробників взаємодіють з Flutter через Flutter Framework, яка забезпечує реактивну структуру та набір віджетів платформи, макета та основи. Дизайн інтерфейсу користувача у Flutter будують з віджетів. Віджет у Flutter являє собою незмінний об'єкт, який описує частину інтерфейсу користувача. Вся графіка, текст, фігури та анімації створюють за допомогою віджетів. Складні віджети створюють шляхом об'єднання простих. На поточний час Flutter містить два набори віджетів, які відповідають відповідним принципам побудови:

- віджети Material Design використовують дизайн Google;

- віджети Cupertino імітують дизайн Apple iOS.

Як було описано раніше, Flutter використовує Dart як основну мову програмування для написання програмного коду додатку. Dart - мова програмування, яку розробляє компанія Google, позиціонуючи як мову структурованого програмування для Веб [45 - 47]. Розробники вважали, що в довгостроковій перспективі Dart може стати прогресивною заміною JavaScript, котрий потерпає від наявних в даний час проблем з розширюваністю, продуктивністю і підтримкою розробки складних застосунків. Мова має схожий на Java синтаксис, не вимагає явного визначення типів і її можна використовувати для створення серверних та клієнтських застосунків.

Мова має схожий на Java синтаксис, не вимагає явного визначення типів і може використовуватися для створення серверних і клієнтських застосунків. Для запуску всередині браузера код мовою Dart може бути перетворений в JavaScript-подання або запущений безпосередньо під управлінням спеціального JavaScript-інтерпретатора Dartboard. Підтримується вбудовування коду мовою Dart в HTML-сторінки, використовуючи MIME тип "application/dart". На стороні сервера застосунок на мові Dart може бути виконаний всередині спеціальної віртуальної машини, яка забезпечує продуктивність виконання близьку до компільованих в машинний код мов.

Мова підходить як для розробки одним програмістом невеликих скриптів без жорсткої структури, так і для створення високо масштабованих великих модульних проектів, підтримуваних великим колективом з потребою більш явної типізації для того, щоб уникнути плутанини і помилок. При цьому явне задання типів не обов'язкове, наприклад, можна почати розробку без вказання типів, а надалі при необхідності додати їх (наприклад, спочатку написати "var x", а потім замінити на "num x"). Код Dart завжди виконується тільки в рамках одної потоку, для організації паралельного виконання пропонується використовувати класи з атрибутом isolate. У кожному скрипті використовується власний простір імен, для використання зовнішніх об'єктів, функцій або змінних слід їх явно імпортувати за допомогою конструкції "import". Всі змінні, початково, діють тільки в межах

поточного скрипту і не експортуються глобально. Для спрощення розробки мовою Dart поставляється SDK, який включає в себе компілятор dart2js, віртуальну машину Dart VM, пакетний менеджер pub, статичний аналізатор коду dart_analyzer, і набір бібліотек. Для виконання і відлагодження застосунків на мові Dart, без компіляції в JavaScript, поширюється Dartium - складання браузера Chromium з інтегрованою віртуальною машиною Dart VM.

Завдяки вибору Flutter процес розробки для мобільної платформи значно спрощується. На публічному ресурсі бібліотек «pub.dev» завжди можна знайти корисну бібліотеку чи пакунок який допоможе вирішити різноманітні проблеми пов’язані з розробкою даного застосунку. Після виконання аналізу засобів реалізації програмного додатку, розглянувши усі можливі рішення було здійснено висновок, що для розробки даного програмного застосунку найкращим варіантом для розробки є Flutter. Головною причиною вибору саме Flutter є його висока продуктивність за рахунок компіляції в нативні мови програмування Kotlin для Android та Swift для iOS, для прикладу на рисунку 3.1 зображено тест швидкості роботи Flutter та React Native, де Flutter в декілька разів швидше виконує операції ніж React Native, при цьому працюючи не сильно повільніше ніж нативний код [48].

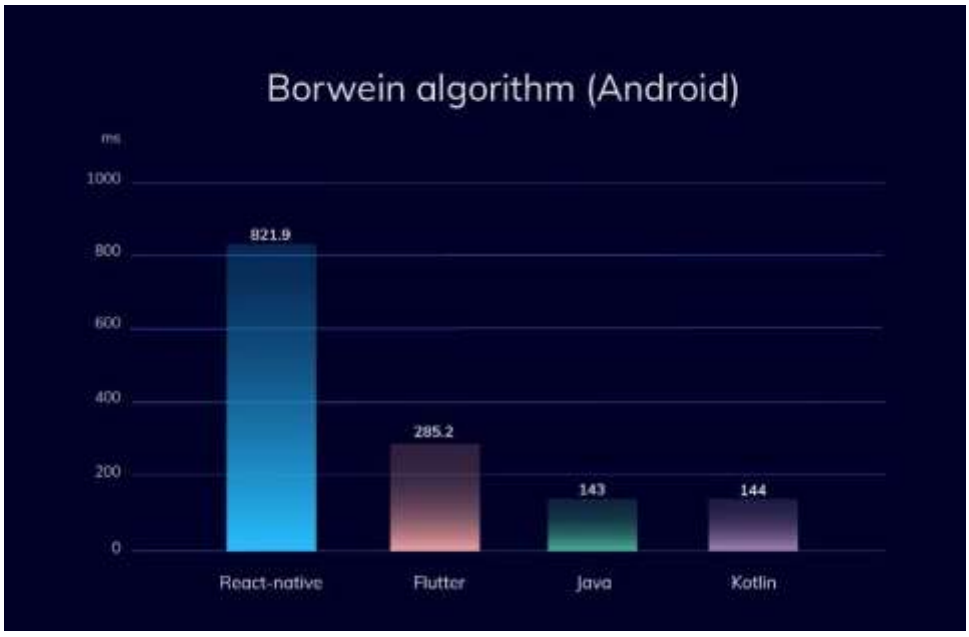


Рисунок 3.1 - Порівняння швидкості роботи Flutter та React Native

3.2 Програмна реалізація мобільного додатку

Програмну реалізацію необхідно розпочати з завантаження Flutter Framework з офіційного сайту, для розробки було обрано версію 3.0.5 як актуальну версію на момент початку розробки. Розпакувавши архів з даними фреймворку, та прописавши необхідні команди для встановлення фреймворку згідно інструкцій можна приступати до створення проекту в середовищі Android Studio. Для цього необхідно відкрити середовище розробки та перейти на вкладку New Project -> New Flutter Project, що зображено на рисунку 3.2, та обрати необхідні платформи, під які буде розроблятись даний проект. В нашому випадку необхідно обрати Android та iOS. Та натиснути далі, таким чином буде створено проект з програмним кодом.

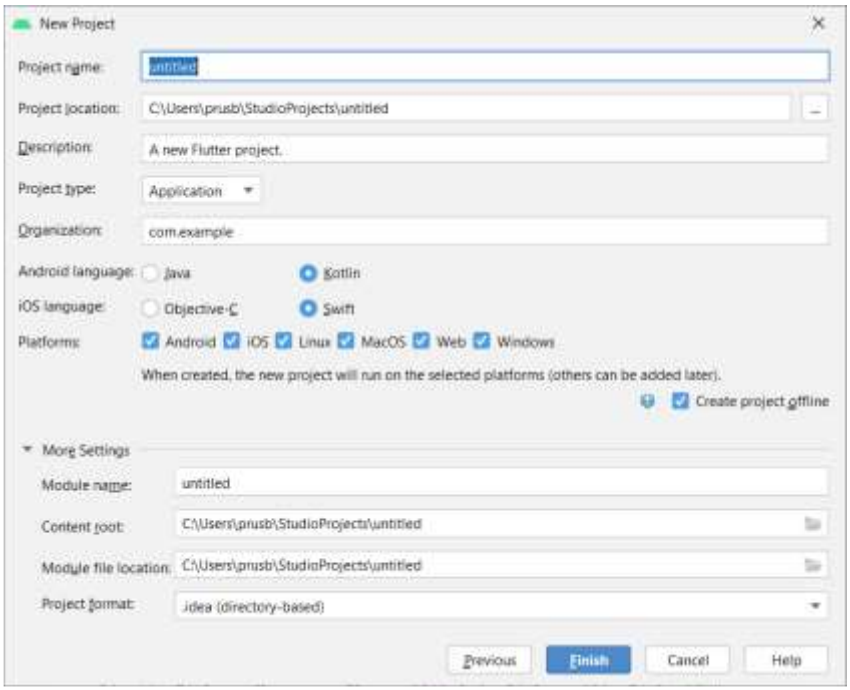


Рисунок 3.2 - Створення проекту в середовищі Android Studio

Наступним етапом необхідно реалізувати підключення проекту до Firebase. Firebase - це Backend-as-a-Service (Baas). Він надає розробникам різноманітні інструменти та послуги, які допомагають їм розробляти якісні програми, розширювати базу користувачів і отримувати прибуток. Він побудований на інфраструктурі Google. Firebase класифікується як програма бази даних NoSQL, яка

зберігає дані в JSON-подібних документах [49 - 50]. Для цього створимо проект Firebase, та додамо додаток до створеного Firebase проєкту - рисунок 3.3.

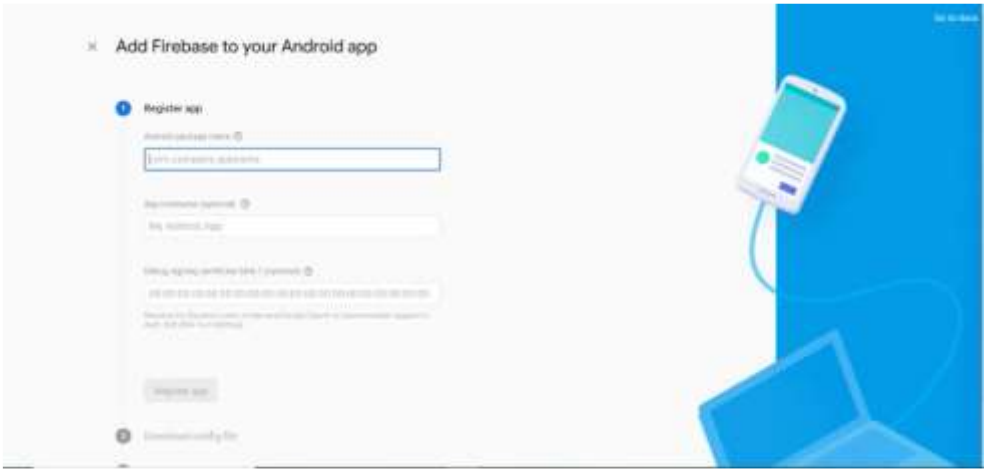


Рисунок 3.3 - Додавання Android додатку проекту Firebase

Далі, відповідно інструкцій Firebase, необхідно створити підключення програмного коду до серверів Firebase. Для цього, на сторінці pub.dev знаходимо бібліотеки firebase core та firebase auth, що буде мінімально необхідно для створення авторизації через firebase. Для їх підключення використовуємо файл проекту pubspec.yaml, в якому знаходяться налаштування проекту, а також всі його залежності. Додамо зазначені бібліотеки до списку залежностей – рисунок 3.4.

```
#firebase
firebase_storage: 11.0.6
firebase_analytics: 10.0.6
firebase_core: 2.3.0
firebase_crashlytics: 3.0.6
firebase_auth: 4.1.3
firebase_messaging: 14.1.1
firebase_dynamic_links: 5.0.6
```

Рисунок 3.4 - Підключення бібліотек firebase до залежностей проєкту

Наступним етапом створимо програмний модуль, для реалізації авторизації. Для цього створимо абстрактний клас BaseAuth в якому створимо логіку роботи з Firebase Auth та імплементацію для нього у класі Auth.

```
class Auth implements BaseAuth {
final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
Future<String> signIn(String email, String password) async {
FirebaseUser user = await _firebaseAuth.signInWithEmailAndPassword(email: email,
password:
password);
return user.uid;
}
Future<String> signUp(String email, String password) async {
FirebaseUser user = await _firebaseAuth.createUserWithEmailAndPassword(
email: email, password: password);
return user.uid;
}
Future<FirebaseUser> getCurrentUser() async {
FirebaseUser user = await _firebaseAuth.currentUser();
return user;
}
Future<void> signOut() async {
return _firebaseAuth.signOut();
}
Future<void> sendEmailVerification() async {
FirebaseUser user = await _firebaseAuth.currentUser();
user.sendEmailVerification();
}
Future<bool> isEmailVerified() async {
FirebaseUser user = await _firebaseAuth.currentUser();
return user.isEmailVerified;
}
@override
Future<void> changePassword(String password) async {
FirebaseUser user = await _firebaseAuth.currentUser();
user.updatePassword(password).then((_) {
print("Succesfull changed password");
}).catchError((error) {
print("Password can't be changed" + error.toString());
});
return null;
}
}
```

В даному методі реалізовані такі методи, як:

- signIn – дозволяє виконувати авторизацію за допомогою електронної пошти та паролю;

- signUp – дозволяє виконувати реєстрацію через Firebase Auth за допомогою електронної пошти та паролю;

- getCurrentUser – дозволяє отримати модель конкретного користувача, використовується для перевірки чи користувач авторизований у системі, або для отримання авторизаційних даних;

- signOut – дозволяє вийти з аккаунту;

- sendEmailVerification – виконується для підтвердження електронної пошти;

- isEmailVerified – виконується для перевірки чи була підтверджена електронна пошта користувача;

- changePassword – виконується для зміни паролю аккаунту.

Слід зазначити, що після стартового екрану, коли користувач вперше відкриває додаток і переходить на головний екран, виконується так звана анонімна реєстрація аккаунту. Це необхідно для того, щоб була можливість ідентифікувати користувача у системі, проте на даному етапі, поки користувач не здійснює жодних дій з програмою, в тому числі і створення резервної копії. Додавання електронної адреси до аккаунту буде виконуватись вже після дій користувача в додатку, таких як автоматична агрегація спогадів, або створення їх власноруч. Для цього необхідно налаштувати проєкт Firebase у консолі. Перейдемо до вкладки «Authentication», та на вкладці «Settings» необхідно увімкнути можливість об’єднання аккаунтів – рисунок 3.5.

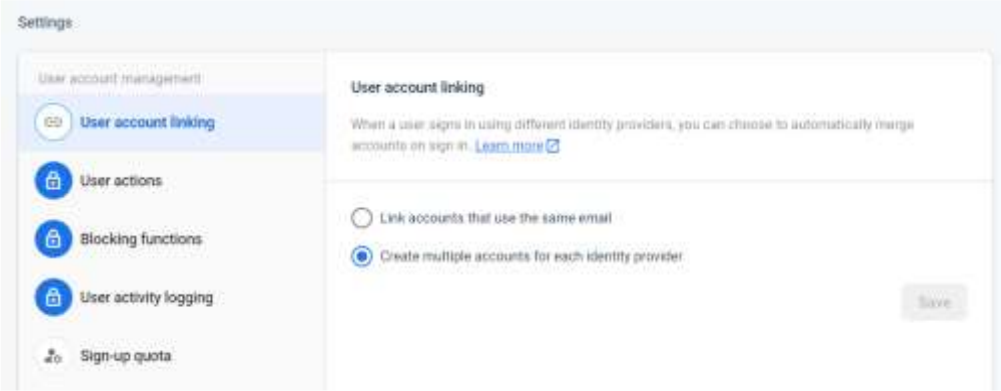


Рисунок 3.5 – Налаштування об’єднання аккаунтів у консолі Firebase

На даному етапі ми маємо підключену авторизацію через Firebase, що допоможе нам ідентифікувати користувача, та реалізувати наступні функції програмного забезпечення. Далі можемо переходити до реалізації ядра системи. Для цього буде створено клас CoreApp, який буде викликатись при запуску додаткF, та в ньому буде рендеретись інтерфейс та реалізована навігація. Далі, згідно дизайну створимо та реалізуємо інтерфейс відповідних екранів, таких як:

- Main View;
- Story View;
- Story Editor View;
- Audio Editor View;
- Canvas Editor View;
- Tags View;
- Map View;
- Splash View;
- SignUp View.

Описані вище екрани будуть імплементацією базового екрану:

```
abstract class BaseControlledStateNSA<B extends StatefulWidget,
C extends BaseControllerNSA> extends BaseStateNSA<B> {
C getController();
@protected
late C controller;
BaseControlledStateNSA({this.tag});
Future<void> _initializeController() async {
controller = getController();
if (Get.isRegistered<C>()) {
await Get.delete<C>();
}
if (mounted) {
setState(() {
Get.put<C>(controller, tag: tag);
_isInitialized = true;
});
}
}
@override
```

```
Widget build(BuildContext context) {
return Scaffold(
body: getProgress(background: false),
);
}
@override
void dispose() {

controller.dispose();
Get.delete<C>(tag: tag);
super.dispose();
}
}
```

Даний клас реалізує поведінку шаблону MVC, який було обрано в якості архітектурного шаблону програмного засобу. Метод getController повертає для класу представлення контролер, з якого представлення використовує дані, які необхідно показати на екран. За допомогою методу update() контролер може змінювати стан екрану.

Таким чином було реалізовано користувацький інтерфейс програмного продукту. Програмний код всіх вікон наведено у додатках. Реалізовані екрани зображено на рисунках 3.6 – 3.9.

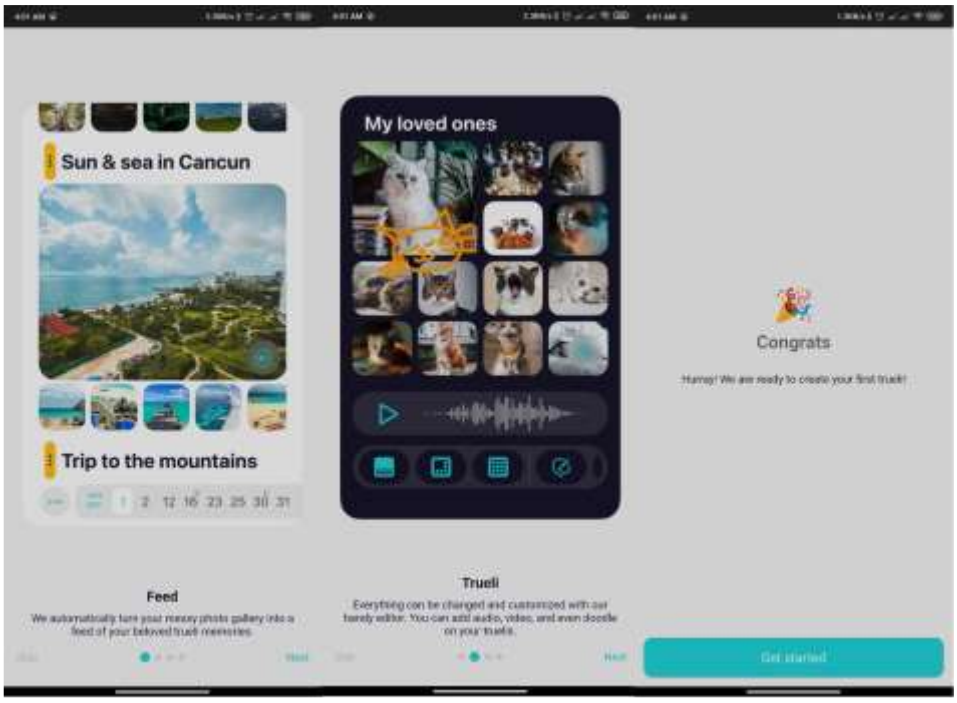


Рисунок 3.6 - Реалізація стартових екранів

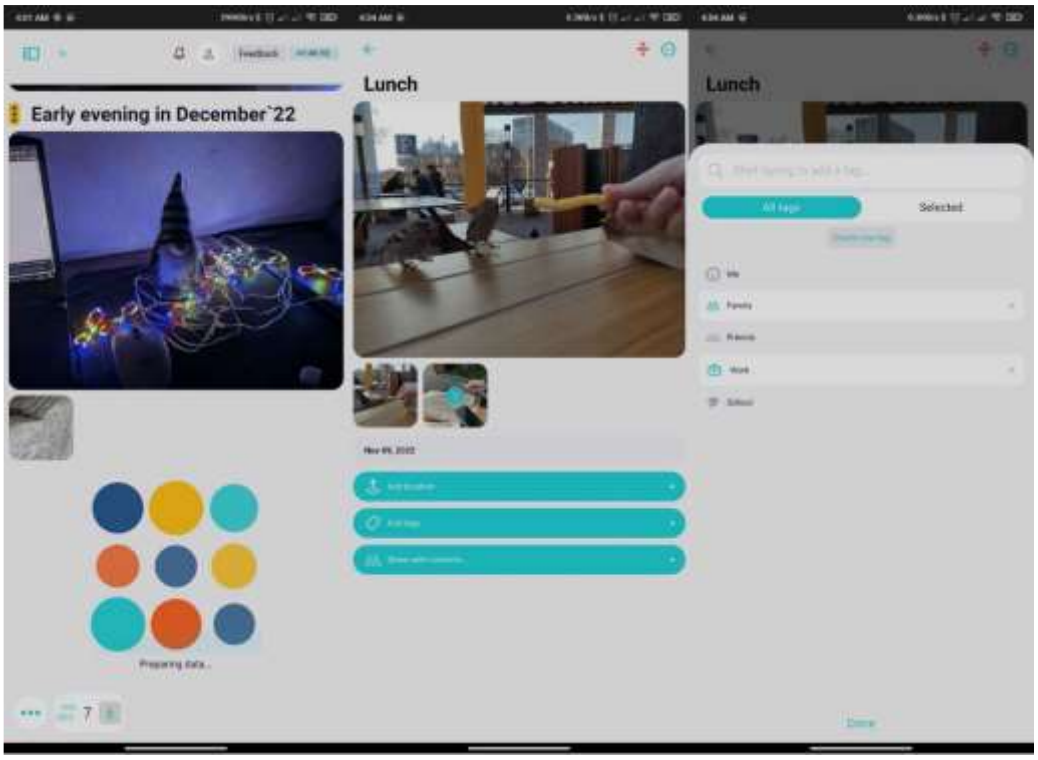


Рисунок 3.7 - Реалізація головного екрану та екрану перегляду спогаду

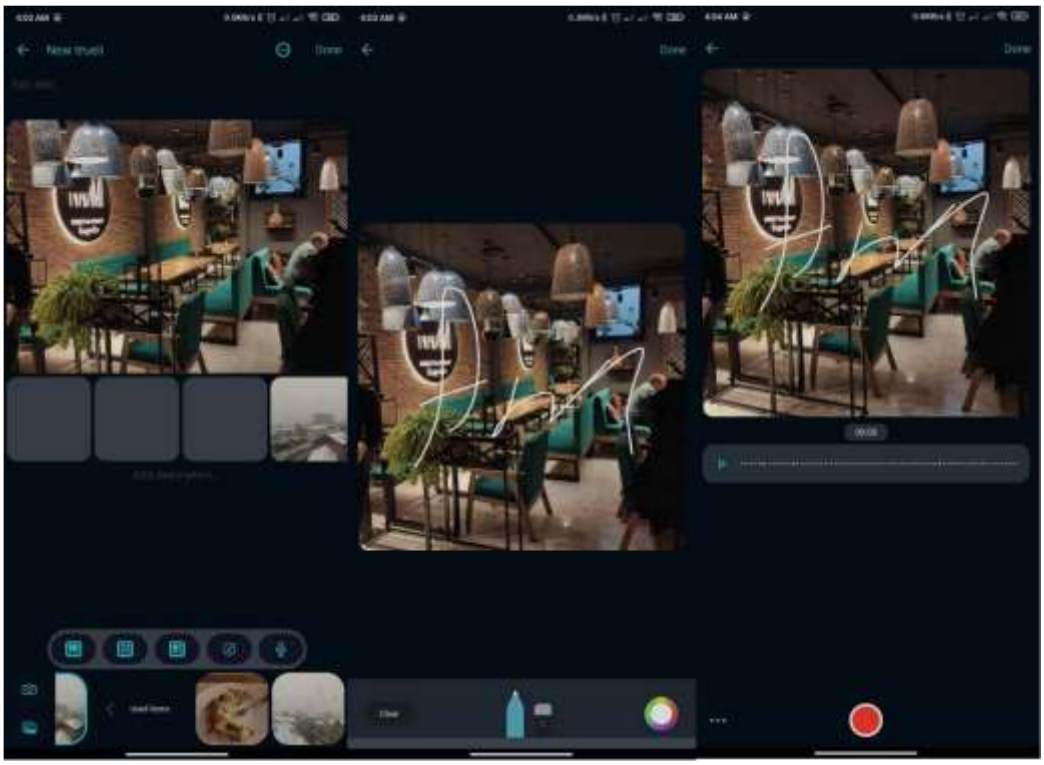


Рисунок 3.8 - Реалізація екрану редагування спогаду та суміжних з ним екранів

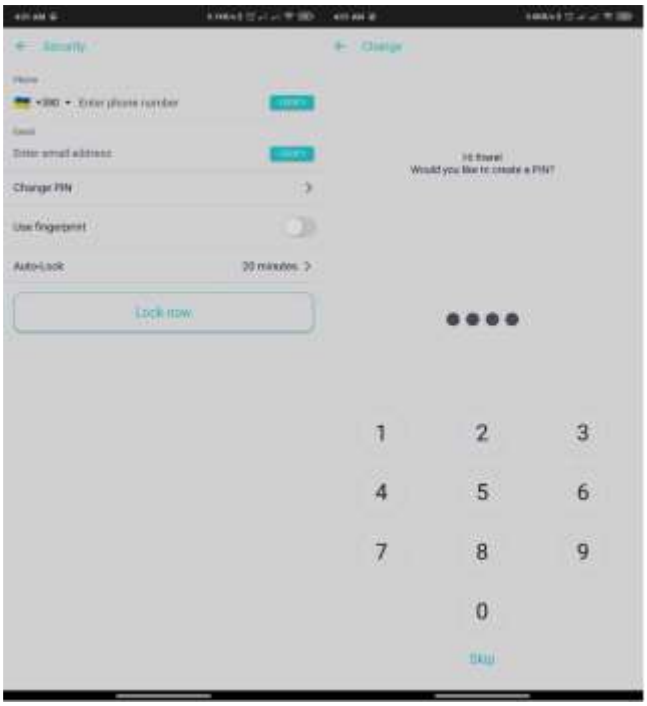


Рисунок 3.9 - Реалізація екранів верифікації електронної адреси та введення пін-коду

3.3 Програмна реалізація алгоритму агрегування медіа-контенту

Для агрегування медіа контенту відповідно алгоритму, який було описано у другому розділі створимо клас AutoCreateStoryManager. У ньому метод getAllAvailableContentAndStartCreate. У даному методі реалізуємо алгоритм роботи агрегування медіа контенту. Для початку дістаємо медіа контент з галереї пристрою за допомогою бібліотеки PhotoManager:

```
final List<AssetEntity> _content = await PhotoManagerService()
.readAssetsFromStorage(startDate: startDate, endDate: endDate);
```

Визначивши список медіа контенту переходимо до його агрегування. Виконаємо фільтрування медіа за часом, відкинувши таким чином фотографії та відео які були створені серією, щоб уникнути можливого дубляжу:

```
contentModel =
await ContentDateFilter().filterSameContent(contentModel);
```

Далі переходимо до створення моделі спогаду та обробки зображень за допомогою засобів розпізнавання зображень визначимо об’єкти на медіа контенті:

```
StoryModel? storyModel =
await CreateStoryModelManager().createStoryModel(contentModel);
storyModel = await ObjectsDetector().processImages(storyModel);
```

У методі processImages визначимо об’єкти та виконаємо фільтрування згідно таблиці 2.2.

```
final DetectedObject detectedObject = DetectedObject(
recognitions[j]['detectedClass'],
recognitions[j]['confidenceInClass']);
if (detectedObject.confidence! *
ObjectThresholdMapper.thresholdCorrection >=
ObjectThresholdMapper().objectThreshold[detectedObject.name]!) {
detectedObjects.add(detectedObject);
                                                    }
```

Таким чином ми отримаємо список медіа файлів та список знайдених на них об’єктів. Далі за допомогою алгоритму генерації назви згенеруємо назву спогаду. Для цього використовуємо метадані медіа файлу, зокрема локацію та час. За допомогою geocoder визначаємо назву локації, а за допомогою часу створення медіа файлу та таблиці 2.1 визначаємо частину дня, яка буде використана для назви. Визначені об’єкти відносимо до категорій. Реалізація створення назви спогаду:

```
Future<StoryModel> getTitle(StoryModel story) async {
story.titleGeneratorDTO!.dateTime = story.mainContent!.dateCreateOriginal;
final Locale locale = await PreferenceManager.getLocale();
try {
story.titleGeneratorDTO!.dateTime = story.mainContent!.dateCreateOriginal;
if (story.locationEntity != null) {
final Pair<String, String>? locationName =
await _getTitleLocationName(story.locationEntity!);
if (locationName != null) {
```

```
story.titleGeneratorDTO!.locationName = locationName.right;
}
}
final Pair<String, StoryTitleChunksEntity> titleResult =
await StoryTitleGenerator(
).getTitleForStory();
        story.name = titleResult.left;
story.storyTitleChunksEntity = titleResult.right;
} catch (err) {
story.name = L.s.trueli;
}
return story;
        }
```

Реалізація визначення назви локації для назви спогаду:

```
Future<Pair<String, String>?> _getTitleLocationName(
final LocationEntity location) async {
String? locationName;
if (location.placeName.isNotNullOrEmpty()) {
} else if (location.city.isNotNullOrEmpty()) {
}
if (locationName == null) {
return null;
}
final List<CustomLocationEntity> _locationsByName =
.getCustomLocationsByAutogeneratedPlaceName(locationName);
await LogService().logRealtime(
"Locations: Searched ${_locationsByName.length} locations with autogenerated name
```

```
'$locationName'");
for (final CustomLocationEntity element in _locationsByName) {
if (element.calculateDistance(location.latitude!, location.longitude!)! <=
Get.find<ConfigResponse>()
.settings!
.maxStoryTitleLocationRadiusMeters!) {
return Pair<String, String>(element.preposition, element.locationName);
}
}
return Pair<String, String>(null, locationName);
}
```

Під час створення спогаду у локальний файл logs.txt записуються усі дії алгоритму, для того щоб можна було відслідковувати роботу алгоритму, а також визначити можливі покращення його роботи. Процес створення спогаду за проміжком часу у вигляді файлу логу програми зображено на рисунку 3.10.

Після створення моделі спогаду додаємо посилання на медіа-файли, та додаємо запис до таблиці StoryTable, а також на головний екран. Далі переходимо до наступної вибірки зображень, і працюємо за алгоритмом, що описаний вище. Програмний код класів для агрегування медіа контенту наведений у додатках.

```
2022-09-21 21:27:19.948100 ------------------------------2022-09-21 21:27:20.247006 ------------------------------
2022-09-21 21:27:20.247009 started story images processing
2022-09-21 21:27:20.247009 globalStoryId = 6DEC13EF-5009-42C2-8676-911F53B2ED6A_b1c2597f-3f9b-4844-bfa5-e456363bc6af
2022-09-21 21:27:20.247010 story dates = 2022-03-19 19:03:31.000 -- 2022-03-19 18:28:01.000
2022-09-21 21:27:20.247013 story created at = 2022-09-21 21:27:20.246745
2022-09-21 21:27:20.247015 Images count before processing = 4
2022-09-21 21:27:20.247016 -> started detection
2022-09-21 21:27:20.247018 -> -> got image = /var/mobile/Media/DCIM/100APPLE/IMG_8302.HEIC
2022-09-21 21:27:20.250717 -> -> -> image saved with milliseconds = 3
2022-09-21 21:27:20.264290 -> -> -> Tflite.detectObjectOnImage finished with milliseconds = 13 2022-09-21 21:27:20.264290 and returned 1 detected objects
2022-09-21 21:27:20.264313 -> -> -> -> added detected object = DetectedObject{name: tv, confidence: 0.5238375}
2022-09-21 21:27:20.264318 -> -> got image = /var/mobile/Media/DCIM/100APPLE/IMG_8301.HEIC
2022-09-21 21:27:20.268690 -> -> -> image saved with milliseconds = 4
2022-09-21 21:27:20.281197 -> -> -> Tflite.detectObjectOnImage finished with milliseconds = 12 2022-09-21 21:27:20.281197 and returned 1 detected objects
2022-09-21 21:27:20.281218 -> -> -> -> ignored DetectedObject{name: book, confidence: 0.57421875}
2022-09-21 21:27:20.281224 -> -> got image = /var/mobile/Media/DCIM/100APPLE/IMG_8300.HEIC
2022-09-21 21:27:20.284872 -> -> -> image saved with milliseconds = 3
2022-09-21 21:27:20.297353 -> -> -> Tflite.detectObjectOnImage finished with milliseconds = 12 2022-09-21 21:27:20.297353 and returned 2 detected objects
2022-09-21 21:27:20.297391 -> -> -> -> ignored DetectedObject{name: book, confidence: 0.66796875}
2022-09-21 21:27:20.297403 -> -> -> -> ignored DetectedObject{name: clock, confidence: 0.64921875}
2022-09-21 21:27:20.297405 -> -> got image = /var/mobile/Media/DCIM/100APPLE/IMG_8299.HEIC
2022-09-21 21:27:20.301164 -> -> -> image saved with milliseconds = 3
2022-09-21 21:27:20.313749 -> -> -> Tflite.detectObjectOnImage finished with milliseconds = 12 2022-09-21 21:27:20.313749 and returned 5 detected objects
2022-09-21 21:27:20.313773 -> -> -> -> ignored DetectedObject{name: refrigerator, confidence: 0.51171875}
2022-09-21 21:27:20.313785 -> -> -> -> added detected object = DetectedObject{name: dining table, confidence: 0.51171875}
2022-09-21 21:27:20.313794 -> -> -> -> ignored DetectedObject{name: chair, confidence: 0.4609375}
2022-09-21 21:27:20.313803 -> -> -> -> added detected object = DetectedObject{name: cup, confidence: 0.4140625}
2022-09-21 21:27:20.313811 -> -> -> -> ignored DetectedObject{name: oven, confidence: 0.3671875}
2022-09-21 21:27:20.313813 -> finished detection with millisecond = 66
2022-09-21 21:27:20.314265 -> excluded 2 images because of no detected objects on them
2022-09-21 21:27:20.314269 -> defined main image if could: ContentModel(assetId: E1490B8E-9F97-4692-94E0-0C482204C352/L0/001, , pathOrigin: /var/mobile/M
2022-09-21 21:27:20.314276 -> prepared TitleGeneratorDTO
2022-09-21 21:27:20.314277 finished story images processing. Millisecond = 67
```

Рисунок 3.10 - Лог роботи алгоритму агрегування медіа контенту

ВИСНОВОК

У кваліфікаційній роботі магістра проведено розробку методів та засобів розпізнавання зображень для створення медіа-контенту. У результаті виконання роботи було створено мобільний застосунок за допомогою фреймворку Flutter, який доступний для платформ iOS та Android. Відмінністю розробленого застосунку є його реалізація для основних мобільних платформ, а також забезпечення приватності користувацьких даних за допомогою шифрування.

У результаті аналізу предметної області було виявлено, що в сучасній сфері мобільних застосунків існує потреба у реалізації мобільного додатку, який допоможе здійснити агрегацію медіа контенту з галереї мобільного пристрою для створення спогадів, які будуть включати в себе лише ті фото та відео матеріали, які мають певну естетичну цінність. Враховуючи ряд вимог до розробки мобільного застосунку було вирішено обрати архітектуру MVC та здійснити реалізацію додатку за допомогою мультиплатформенного фреймворку Flutter на мові Dart. Для організації менеджеру станів екранів та архітектури додатку було обрано менеджер станів GetX. Для реалізації алгоритму агрегування медіа-контенту було обрано відкриту бібліотеку TensorFlow, яка допомагає здійснювати аналіз зображень на предмет виявлення на них об'єктів. Для здійснення локального збереження даних було обрано відкриту бібліотеку SqfLite, яка основана на мові запитів SQLite.

Проведено порівняння з такими аналогами, як Google Photos та Apple Photos. В процесі здійснення аналізу було визначено слабкі та сильні сторони аналогів та визначено відмінність розроблюваного програмного забезпечення від конкурентів.

Створивши таблицю порівняння аналогів було зроблено висновок, що система є конкурентоспроможною та має ряд переваг які роблять її кращою за аналогічні системи, зокрема головною перевагою є забезпечення приватності користувацьких даних, оскільки вся обробка зображень здійснюється локально на пристрої без використання сторонніх сервісів чи серверів.

В процесі проектування було розроблено діаграму використання мобільного додатку. Спроектовано логічну та фізичну структури мобільного додатку, які

допомагають розробнику оптимізувати свою роботу, а також будуть зрозумілими для інших фахівців, що працюють над проектом. Проведено розробку структури інтерфейсу, розроблений дизайн мобільного додатку спрямований на належне представлення користувачеві всіх візуальних складових та забезпечення правильної роботи усього функціоналу застосунку.

Проведено аналіз найвідоміших засобів реалізації програмного продукту, які мають можливість одночасної розробки для різних платформ. Було проаналізовано такі засоби, як Kotlin Multiplatform, React Native та Flutter. Після виконання аналізу засобів реалізації було здійснено вибір в користь розробки мобільного додатку за допомогою фреймворку Flutter.

Проведено аналіз середовищ розробки, таких як Visual Studio Code, Android Studio та Xcode. Після проведення аналізу середовищ було обрано Android Studio для здійснення основної розробки мобільного додатку для платформи Android, а також середовище Xcode для здійснення розробки для платформи iOS.

Здійснено розробку програмного забезпечення за допомогою фреймворку Flutter та мови Dart. Проведено розробку візуальної та логічної частини усіх екранів, розробку модулів агрегування медіа-контенту, створення спогадів, авторизації у системі.

Кваліфікаційна робота магістра має практичну цінність, оскільки розроблені модулі агрегування медіа-контенту можуть використовуватись для інших проєктів, а також має високу конкурентоспроможність на ринку, оскільки має не велику кількість прямих конкурентів, при цьому маючи низку вагомих переваг.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Smartphones vs Cameras: Closing the gap on image quality [Електронний ресурс] – Режим доступу: https://www.dxomark.com/smartphones-vs-cameras-closingthe-gap-on-image-quality.
2. Photos for iOS and iPadOS [Електронний ресурс] – Режим доступу: https://www.apple.com/ios/photos.
3. Google Photos [Електронний ресурс] – Режим доступу: https://www.google.com/photos/about.
4. Google Photos. Growth [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Google_Photos#:~:text=In%20May%202017%2C%20Google%20announced%20that%20Google%20Photos%20has%20over,photos%20and%20videos%20are%20uploaded.
5. Google Drive [Електронний ресурс] – Режим доступу: https://www.google.com/drive.
6. Apple iCloud [Електронний ресурс] – Режим доступу: https://www.icloud.com.
7. Теорія розпізнавання образів [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Теорія_розпізнавання_образів.
8. Tensor Flow [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/TensorFlow.
9. Pramod Singh. Learn Tensor Flow 2.0. 1st edition / Apress, 2020. – 343 с.
10. UML. Короткий посібник по UML (UML Distilled: A Brief Guide to the Standard Object Modeling Language.) 1999.
11. Архітектура модель-вид-контролер [Електронний ресурс] – режим доступу: https://www.codecademy.com/articles/mvc.
12. Архітектура мобільних додатків [Електронний ресурс] – Режим доступу: https://koloro.ua/arkhitektura-prilozheniya.html.
13. Проектування мобільного додатку [Електронний ресурс] – Режим доступу: https://wezom.com.ua/ua/blog/proektirovanie-mobilnogo-prilozheniya.

14. Photo Manager package [Електронний ресурс] – Режим доступу: https://pub.dev/packages/photo_manager.

15. Sqflite package [Електронний ресурс] – Режим доступу: https://pub.dev/packages/sqflite.

16. G.J. CHI, L. LIU, J.L., Z. JIANG and G. ZHANG: Machine Vision Based Automatic Detection Method of Indicating Values of a Pointer, Mathematical Problems in Engineering, 2015, (2015), Article ID 283629, 19 pages.

17. J. CANNY: Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6), (1986), 679–698.

18. J. BERNSEN: Dynamic Thresholding of Grey-Level Images, Proc. of the 8th Int. Conf. on Pattern Recognition, Paris (1986), 1251–1255.

19. R. LISHCHUK, V. KUCHERUK and I.P. KURYTNIK: Adaptive binarization with non-uniform image illumination, Pomiary Automatyka Robotyka, 6, (2014), 72–76.

20. T. HUANG, G. YANG and G. TANG: A fast two-dimensionalmedian filtering algorithm, IEEE Trans. Acoust., Speech, Signal Processing, 27(1), (1979), 13–18.

21. R.O. DUDA and P.E. HART: Use of the Hough Transformation to Detect Lines and Curves in Pictures, Comm. ACM, 15, (1972), 11–15.

22. Розробка дизайну інтерфейсу мобільних додатків [Електронний ресурс] – Режим доступу: https://web24.pro/android-rozrobka/rozrobka-dyzajnu-interfejsumobilnyh-dodatkiv.

23. Theo Mendel. Golden Rules to User Interface Design / DMK 2001 – 54 с.

24. Figma [Електронний ресурс] – Режим доступу: https://www.figma.com.

25. Kotlin Multiplatform [Електронний ресурс] – Режим доступу: https://kotlinlang.org/docs/multiplatform.html.

26. React Native [Електронний ресурс] – Режим доступу: https://reactnative.dev.

27. Flutter [Електронний ресурс] – Режим доступу: https://flutter.dev.

28. Dart – programming language [Електронний ресурс] – Режим доступу: https://dart.dev.

29. Flutter vs Native vs React-Native: Examining performance [Електронний ресурс]

– Режим доступу: https://medium.com/swlh/flutter-vs-native-vs-react-nativeexamining-performance-31338f081980.

30. Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com.

31. GitHub [Електронний ресурс] – Режим доступу: https://github.com.

32. Android Studio [Електронний ресурс] – Режим доступу: https://developer.android.com/studio.

33. Google I/O 2013 [Електронний ресурс] – Режим доступу: https://developers.google.com/events/io/2013.

34. Xcode [Електронний ресурс] – Режим доступу: https://developer.apple.com/xcode.

35. Firebase [Електронний ресурс] – Режим доступу: https://firebase.google.com.

36. Розробка Web-додатків [Електронний ресурс] - Режим доступу до ресурсу: http://clashdash/directions/web-applications

37. Using Service Workers [Електроний ресурс] - Режим доступу до ресурсу: https://developer.mozilla.org/API/Service_Worker_API/Using_Service_Workers

38. PostgreSQL [Електроний ресурс] - Режим доступу до ресурсу: https://builtin.com/learn/tech-dictionary/postgresql

39. Single-page application vs. multiple-page application [Електроний ресурс] - Режим доступу до ресурсу: https://medium.com/@NeotericEU/single-pageapplication-vs-multiple-page-application-2591588efe58

40. Node.js Documentation [Електронний ресурс] - Режим доступу до ресурсу:https://nodejs.org/dist/latest-v14.x/docs/api/

41. TypeScript [Електронний ресурс] - Режим доступу до ресурсу: https://www.typescriptlang.org/

42. Тестування мобільних додатків [Електронний ресурс] – Режим доступу: https://qagroup.com.ua/publications/testuvannia-mobilnykh-dodatkiv-vid-a-do-ia.

43. Bluetooth: технологія і її використання. [Електронний ресурс]. Режим доступу: https://www.ixbt.com/mobile/review/bluetooth-2.shtml

44. Все що потрібно знати про технології Bluetooth. [Електронний ресурс]. Режим доступу: http://motorvision.com.ua/vse-shho-potribno-znaty-pro-tehnologiyi-bluetooth/

45. Java: A Beginner's Guide, Eighth Edition / Herbert Schildt – 2018

46. The C++ programming language \ Bjarne Stroustrup – 2013

47. Getting started with WebStorm [Електронний ресурс] : echo / Режим доступу: https://www.jetbrains.com/help/webstorm/getting-started-withwebstorm.html

48. Visual Studio Code Tips and Tricks [Електронний ресурс]. – Режим доступу: до ресурсу: https://code.visualstudio.com/docs/getstarted/tips-and-tricks

49. How to Use Sublime Text: A Quick Overview [Електронний ресурс]. – Режим доступу до ресурсу: https://kinsta.com/blog/how-to-use-sublime-text/

50. Java is a powerful general-purpose programming language. [Електронний ресурс]. – Режим доступу: https://www.programiz.com/javaprogramming