

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: "Розробка мобільного додатку для організації фотонотаток"

Виконав: студент групи

 Чугунов К. А.
(підпис)

Керівник роботи:

старший викладач

(посада, науковий ступінь вчене звання)

 Павленко А. Ю.
(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

Спеціальність 124 "Системний аналіз"

Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


к.т.н, доц. Гайдаєнко О. В.

" ____ " _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Чугунову Кирилу Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: "Розробка мобільного додатку для організації фотонотаток"

Керівник роботи Павленко Альона Юріївна

Затверджені наказом ректора № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем

4. Перелік питань, що належать до розробки (найменування розділів) аналіз предметної галузі, постановка задачі, розробка проекту інформаційної підсистеми, реалізація проекту підсистеми, охорона праці

5. Перелік презентаційних матеріалів титульна сторінка, вступ, мета, актуальність, схема запропонованої концепції, діаграма варіантів використання, концептуальна модель даних, логічна модель даних, фізична модель даних, діаграма компонентів, діаграма переходу станів, діаграма розгортання системи, результати роботи, висновок

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Павленко А. Ю., ст. викладач	03.02.2025	22.02.2025
2	Павленко А. Ю., ст. викладач	22.02.2025	21.03.2025
3	Павленко А. Ю., ст. викладач	21.03.2025	25.04.2025
4	Павленко А. Ю., ст. викладач	25.04.2025	07.06.2025

7. Дата видачі завдання р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Чугунов К. А.
(ПІБ)

Керівник роботи


(підпис)

Павленко А. Ю.
(ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота із створення мобільного додатку для організації фотонотаток.

У результаті розвитку мобільних інформаційних систем, виникла потреба у зручному для користувача способі отримання та зберігання даних на пристрої. Фокусування уваги найбільше зосереджується на додатках-послугах(супер-додатків) та додатках-функціях, які мають виконувати лише одну дію та не мати зайвого функціоналу. Найбільше юзери використовують телефони як пристрій для фотографування. Саме тому маємо мету - розробити такий додаток, який міг би полегшити процес отримання фото та пов'язати ці дані із моментом їх створення.

У ході виконання роботи можна побачити наступні технології: Android, Kotlin, Jetpack Compose, Room Database, Google Map API, GMS Location API.

В результаті виконання роботи, було розроблено додаток, що відповідає вимогам користувачів, та який має ефективні алгоритми збереження даних та їх поєднання.

Мобільний додаток можна використати як засіб зберігання особливо важливих моментів, спостерігаючи отримані результати не тільки на фотографії, а й переглядати моменти із прив'язкою до карти. Оскільки додаток не є перевантаженим надлишковими функціями, маємо легкий спосіб взаємодії із даними у пару кліків.

Робота складається із 88 сторінок, 56 рисунків, 10 таблиць, 3 додатків та 16 літературних джерел.

Ключові слова: *фото, галерея, нотатки, мобільний додаток, база даних, фотонотатки, дизайн.*

ABSTRACT

Qualification work on creating a mobile application for organizing photo notes.

As a result of the development of mobile information systems, there is a need for a user-friendly way to receive and store data on the device. The focus is most focused on service applications (super applications) and function applications that should perform only one action and not have unnecessary functionality. Most users use phones as a device for taking photos. That is why our goal is to develop an application that could facilitate the process of receiving photos and link this data to the moment of their creation.

During the work, the following technologies can be seen: Android, Kotlin, Jetpack Compose, Room Database, Google Map API, GMS Location API.

As a result of the work, the application was developed according to requirements of users and has effective algorithms for saving data and combining them.

The mobile application can be used as a means of storing especially important moments, observing the results obtained not only on the photo, but also viewing moments with a link to the map. Since the application is not overloaded with redundant functions, we have an easy way to interact with the data in a couple of clicks.

The work consists of 88 pages, 56 figures, 10 tables, 3 appendices and 16 literary sources.

Keywords: *photo, gallery, notes, mobile app, database, photonotes, design.*

ЗМІСТ

ВСТУП

РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ. ПОСТАНОВКА ЗАДАЧІ

- 1.1. Дослідження та аналіз предметної галузі
- 1.2. Аналіз існуючих рішень
- 1.3. Визначення проблем
- 1.4. Розробка концепцій рішення проблем
- 1.5. Постановка задачі

РОЗДІЛ ІІ. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

- 2.1. Загальносистемні рішення
 - 2.1.1. Загальний опис системи
 - 2.1.2. Розробка діаграми варіантів використання
- 2.2. Рішення з інформаційного забезпечення
 - 2.2.1. Визначення сутностей інформаційної підсистеми
 - 2.2.2. Розробка логічної моделі даних
 - 2.2.3. Розробка фізичної моделі даних
- 2.3. Рішення з технічного забезпечення
 - 2.3.1. Вимоги до технічного забезпечення
 - 2.3.2. Склад технічного забезпечення
- 2.4. Рішення з програмного забезпечення
 - 2.4.1. Розробка схеми структури інформаційної підсистеми
 - 2.4.2. Розробка діаграми компонентів
 - 2.4.3. Розробка діаграми переходів станів
 - 2.4.4. Розробка проекту інтерфейсу інформаційної підсистеми
 - 2.4.5. Розробка діаграми розгортання підсистеми

РОЗДІЛ ІІІ. РЕАЛІЗАЦІЯ ПРОЕКТУ ПІДСИСТЕМИ

- 3.1. Аргументація вибору засобів розробки
- 3.2. Розробка структури програмного забезпечення

3.3. Результати розробки програмного забезпечення

РОЗДІЛ IV. ОХОРОНА ПРАЦІ

4.1. Загальні питання охорони праці

4.2. Структура управління питаннями охорони праці на підприємстві

4.3. Шкідливі та небезпечні фактори, що супроводжують працю програміста

4.4. Ергономічні вимоги до робочого місця при роботі за комп'ютером

4.5. Нормативні вимоги до робочої площі на одного працюючого за комп'ютером

ВИСНОВОК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А –Технічне завдання

ДОДАТОК Б – Програмний код

ДОДАТОК В – Інструкція користувача

ВСТУП

Сьогодні мобільні додатки займають важливе місце у повсякденному житті людей. Їх популярність невпинно зростає, охоплюючи найрізноманітніші сфери: від розважальних ігор, що допомагають "вбити час", до сервісів, які дозволяють зручно керувати особистими справами — музикою, фільмами, банківськими рахунками тощо.

Окрему нішу займають невеликі, вузькоспеціалізовані додатки, які виконують одну або кілька конкретних функцій. До таких можна віднести фоторедактори, відеоредактори, програми-шифрувальники, аудіо-рекордери, бази даних та інші. Ці рішення дозволяють користувачам створювати, зберігати та обробляти власні дані, не покладаючись на великі централізовані системи, які часто стають об'єктами кібератак.

У цьому контексті особливо цінними є застосунки, що дозволяють зручно зберігати особисту інформацію, зокрема — фото та пов'язані з ними спогади. Саме тому було прийнято рішення створити мобільний застосунок для збереження фотонотаток.

Фотографії вже давно стали способом зафіксувати важливі миті життя. За допомогою застосунку користувачі можуть не лише зберігати знімки, але й прикріплювати їх до конкретного місця події, створюючи повноцінну цифрову нотатку. Це дозволить не просто згадати момент, а й відчувати його знову — в контексті часу, місця та емоцій.

РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Дослідження та аналіз предметної галузі

Організація фотонотаток — це процес взаємодії користувача з великою кількістю власних візуальних даних. У сучасному світі щодня створюються мільйони знімків, що породжує потребу у зручному, структурованому та безпечному зберіганні фотографій. Особливої уваги набуває захист приватності: як фотографії, так і облікові дані користувача містять конфіденційну інформацію, що потребує відповідного рівня шифрування та захисту.

На ринку вже існує велика кількість рішень для збереження та організації фотографій — від стандартних галерей на смартфонах до хмарних сервісів (Google Photos, iCloud, OneDrive тощо). Вони мають як переваги, так і недоліки. Зокрема, одним із найбільших недоліків є надлишковий функціонал, що ускладнює інтерфейс, створює плутанину у пошуку потрібного знімка та збільшує обсяг збережених даних, часто непотрібних для користувача.

У зв'язку з цим виникає ідея створення компактного, інтуїтивно зрозумілого мобільного додатку, що дозволяє не просто зберігати фотографії, а й доповнювати їх контекстом — місцем, нотатками. Такий підхід дозволяє персоналізувати фотоархів та зробити його значно зручнішим для перегляду і збереження спогадів. Запропонований застосунок не лише задовольняє базову потребу в організації фотографій, але й пропонує новий рівень приватності та зручності. Він не дублює функціонал великих платформ, а зосереджується на простоті, безпеці та персоналізації, що робить його потенційно конкурентоспроможним рішенням у своїй ніші.

Для захисту користувача використовується багато різних методів. Так наприклад, використання сервісів для резервування даних є одним із популярних рішень, проте є свої недоліки. Якщо машина, що зберігає резервні дані буде взламано, маємо витік даних користувача. Якщо ми використовуємо носій (будь-

то флешку, диски) для цієї цілі, маємо ще запровадити додаткові способи захисту, та також, не маємо чіткої впевненості, що резервування не буде мати "подарунки". Є, наприклад, спосіб резервування за допомогою копіювання файлів з одного місця, до іншого в межах однієї машини. Це може бути використано, наприклад, якщо взаємодія між користувачем та базою даних виконується за допомогою кешу. В результаті, якщо кешовані дані є важливими, та ми маємо їх зберегти, можемо використати цей метод.

У захисті Android пристроїв також працює цей підхід. Ми маємо створити додаток таким чином, щоб користувач використовував зовнішні сервіси не для зберігання інформації, а для її відтворення. Тобто, якщо дані зберігаються, то тільки на пристрою користувача, що зменшить діапазон зони захисту, який треба контролювати. Як результат, відмовляємося від Firebase Realtime Database, AWS та інших хмарних рішень.

З іншої сторони, використання Google Maps дозволить користувачу отримати актуальні дані карти світу, не зберігаючи власних даних користувача у хмарі.

Із впровадженням локації користувача виникає складне питання. По-перше, маємо визначити як саме буде отримано локацію. Тут є два шляхи: надання власне користувачем своїх даних, або використання зовнішнього провайдера, у ролі якого може виступати будь-що(сервер, інший користувач, публічні свідки та ін). Розглянемо переваги та недоліки таких підходів. Якщо використовувати власне користувача, виникає питання орієнтування користувача. Можна скористатися, наприклад, телефоном користувача. Тоді, за допомогою датчиків смартфона можна знайти приблизну позицію користувача. Недоліком цього підходу є неточність вимірювання (датчики телефонів не завжди коректно відпрацьовують із визначенням локації юзера). Також, можна зазначити, що час, впродовж якого виконується підрахунок локації користувача є відносно тривалим. Але, великою перевагою є незалежність від сторонніх сервісів, що підвищує захист користувача. Якщо спробувати обрати другий шлях

та почати використовувати зовнішніх провайдерів як GMS Location, маємо: залежність від зовнішніх сервісів, відповідно залежність від мережі. Також, це залежність від Google. Але, із переваг, це і відносно швидке отримання більш-менш точних даних користувача, і залежність від перевіреного серверу, який має свою систему захисту. Але, все-таки впровадження даних власне через користувача все одно залишається найбільш конфіденційною схемою.

Система реагування відгуків має також велике значення для кінцевого клієнта. Відгук - це посилення, яке ми маємо донести до виробника додатку, із метою покращення додатку або/та виправлення непрацюючих фіч. Так, це має бути зручним для обох сторін. Комунікація між клієнтом та виробником додатку може здійснюватися лише через мережу та не можемо використовувати внутрішню пам'ять пристрою для листування виробника із кінцевим користувачем. У цьому випадку основною опорою може стати лише зовнішній сервіс. Є велика кількість сервісів, які можемо використовувати для задачі: сервери, поштові сервіси, самописний варіант тощо. Поштові сервіси є досить зручним варіантом, оскільки вони популярні у використанні, та майже у кожному Android пристрої вони пошта підтримується з початку встановлення системи.

Питання облікових записів. У додатку буде підтримуватися функція створення облікових записів, які зберігаються локально на пристрої. Які ще були варіанти? Ми можемо перенести облікові записи на хмарне збереження, але тут постає питання коштів, адже зберігання даних у хмарі вартує грошей, також оптимізація бази даних, спосіб зберігання фото користувача. Можна написати власний сервер або базу даних та перенести їх на іншу машину. Проблема у залежності від іншої машини. Для збільшення конфіденційності будемо зберігати облікові дані користувача на пристрої. Таким чином, доступ - це ще одна ланка сортування у додатку, яка підтримує додаткові способи захисту інформації(використання логіну та пароллю). В результаті, сортування даних користувача буде здійснюватися буде мати наступну структуру (рисунок 1.1).



Рисунок 1.1 - Структура сортування даних користувача

1.2 Аналіз існуючих рішень

Формат фотонотаток поєднує в собі функціональність галереї та нотатника. Базовий варіант такого застосунку має бути простим, інтуїтивно зрозумілим і не перевантаженим функціями, з можливістю подальшого розширення до повноцінного багатофункціонального сервісу — за умови реального попиту серед користувачів. Однією з ключових вимог є наявність камери "під рукою", а також зручного та швидкого інтерфейсу.

Проаналізуємо плей маркет для аналізу на існуючі рішення.

Google Photos (Google LLC) - потужне рішення, проте не підходить для тих, хто прагне мінімалізму (рисунок 1.2). Розберемо особливості.

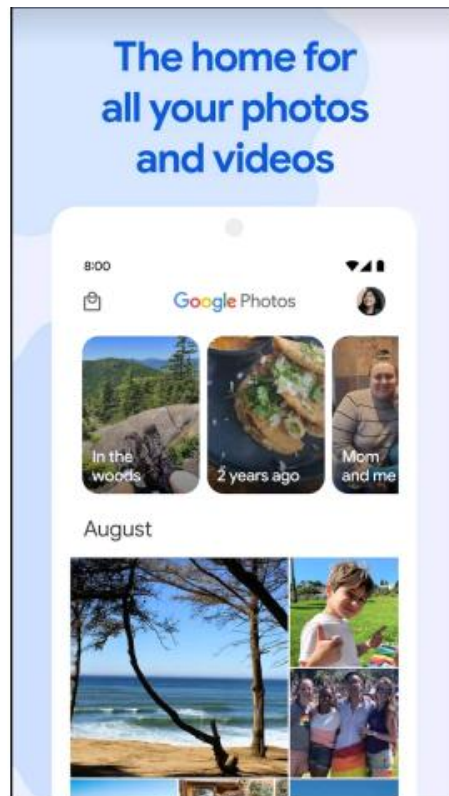


Рисунок 1.2 - Google Photos

Переваги:

- Один із найпопулярніших додатків для зберігання фотографій.
- Забезпечує автоматичне резервне копіювання даних у хмару.
- Інтеграція з іншими сервісами Google.
- Потужна система пошуку, сортування, виявлення подій та згадок.
- Зручний та сучасний інтерфейс.

Недоліки:

- Надмірна інтеграція з іншими сервісами — ускладнення інтерфейсу.
- Велика кількість фіч може бути зайвою для користувача, який шукає простоту.

Галерея — Фотогалерея, Альбом (PhotoZen Studio) - непоганий варіант для збереження фото, проте недолік у вигляді реклами (рисунок 1.3).

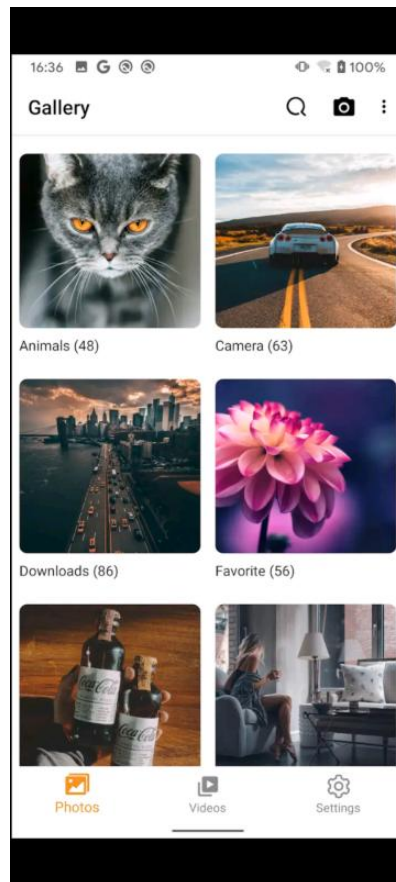


Рисунок 1.3 - Галерея - Фотогалерея, Альбом

Переваги:

- Легка у використанні, підходить для базового перегляду фото.
- Підтримує відображення метаданих зображень.
- Наявні базові можливості сортування та категоризації.
- Проста структура інтерфейсу.

Недоліки:

- Реклама, яка змушує користувача платити за комфортне використання.
- Відсутність інтеграції з іншими сервісами.
- Відсутність можливості перегляду місця зйомки на карті.
- Інтерфейс містить велику кількість налаштувань, що може перевантажити простого користувача.

Gallery (Google LLC) - простий і зручний додаток, проте не охоплює потребу в додатковій інформації про фото, що важливо для формату фотонотатків (рисунок 1.4).

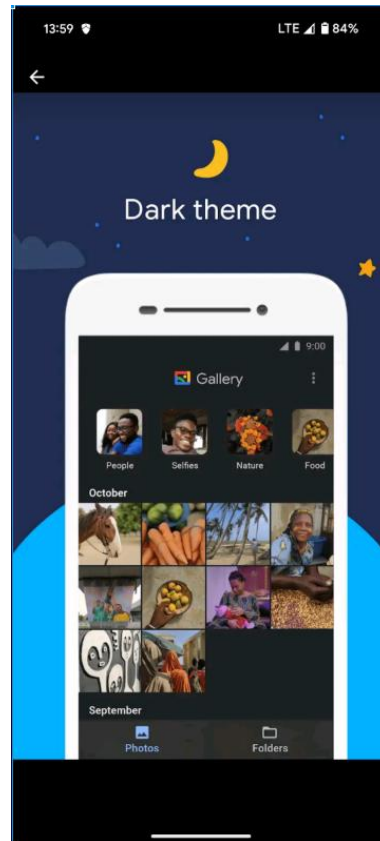


Рисунок 1.4 - Gallery

Переваги:

- Мінімалістичний дизайн та компактний функціонал.
- Зручна система сортування та архівації.
- Швидка робота без потреби постійного доступу до інтернету.

Недоліки:

- Відсутність зайвої інтеграції з іншими сервісами.
- Можливість редагування фотографій.

Отже, створимо такий додаток, який поєднує відображення картинок відповідно до місця події, містить додавання опису, простий та зручний інтерфейс. Для цього використаємо таблицю порівняння функцій додатків.

Таблиця 1.1 - Порівняння функцій додатків

Функція	Google Photos	Галерея - Фотогалерея, Альбом	Gallery	Photo Notes
Резервне копіювання	+	-	-	-
Сортування фото	+	+	+	+
Пошук	+	+	-	-
Зручний інтерфейс	+	+	+	+
Використання карти	+	-	-	+
Редагування фотографій	+	+	+	-
Інтеграція із камерою	-	+	-	+
Зміна мови	+	+	+	+
Управління системними налаштуваннями	+	+	+	+
Використання реклами	-	+	-	-
Надіслання відгуку	+	+	+	+
Керування обліковими записами	+	-	-	+
Простота у використанні	-	+	+	+
Додаток-функція	-	-	-	+

1.3 Визначення проблем

Основною проблемою є питання інтерфейсу. Якщо додаток перевантажений зайвими елементами або функціями, це ускладнює його використання та може викликати відчуття плутанини. Саме тому інтерфейс повинен бути максимально простим, інтуїтивно зрозумілим і зручним навіть для не досвідчених користувачів. Основні функції, зокрема створення фотонотатків, їх перегляд і редагування, повинні бути доступні в один-два кліки.

Функції, які має виконувати програма повинні бути чітко сформульовані. Кожен андроїд додаток, крім супер-додатків, виконує якусь свою функцію. Якщо користувач не може виконати необхідну функцію або якщо інтерфейс перевантажений, маємо погані результати.

Останньою проблемою є ефективність програми. Дані необхідно зберігати так, щоб користувач не почував перевантаження мобільного пристрою через декілька хвилин використання додатку. Також, процеси завантаження мають виконуватися швидко. Для цього будемо використовувати ефективні алгоритми зберігання даних та мінімалістичний інтерфейс, який не буде вимагати великого обсягу оперативної пам'яті.

1.4 Розробка концепцій рішення проблем

Розглянемо концепції інформаційних систем раніше розглянутих існуючих рішень.

Інформаційну систему додатку Google Photos(Google LLC) наведено на рисунку 1.5.

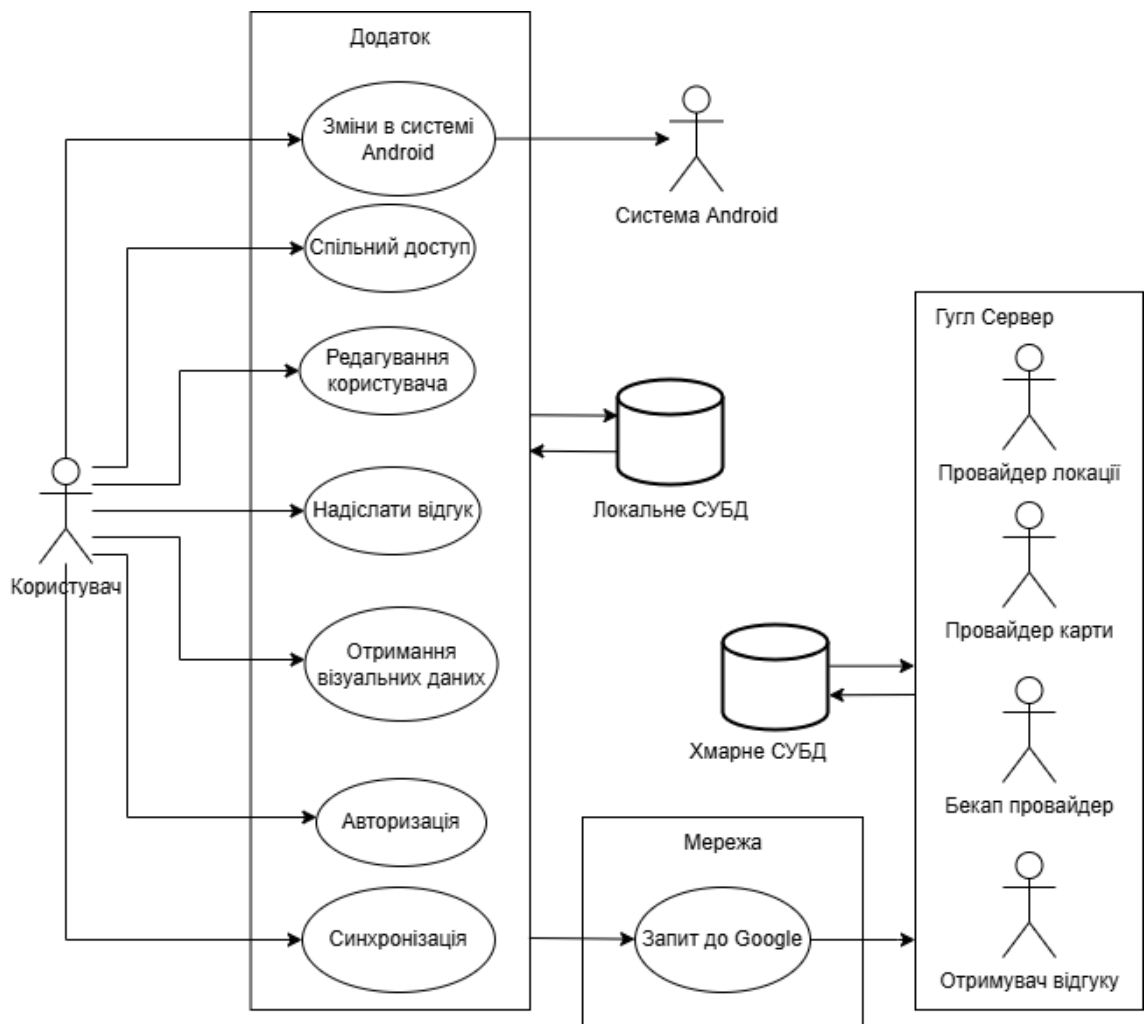


Рисунок 1.5 - Концепція інформаційної системи Google Photos

Додаток складається з двох основних компонентів - додатка та гугл серверу, які поєднуються за допомогою мережі. Додаток використовує наступні функції: авторизацію, спільний доступ, редагування користувача, надіслання відгуку, отримання візуальних даних, змінює налаштування в системі Android та синхронізує дані із гугл сервером.

Інформаційну систему додатку Галерея - Фотогалерея, Альбом (PhotoZen Studio) наведено на рисунку 1.6.

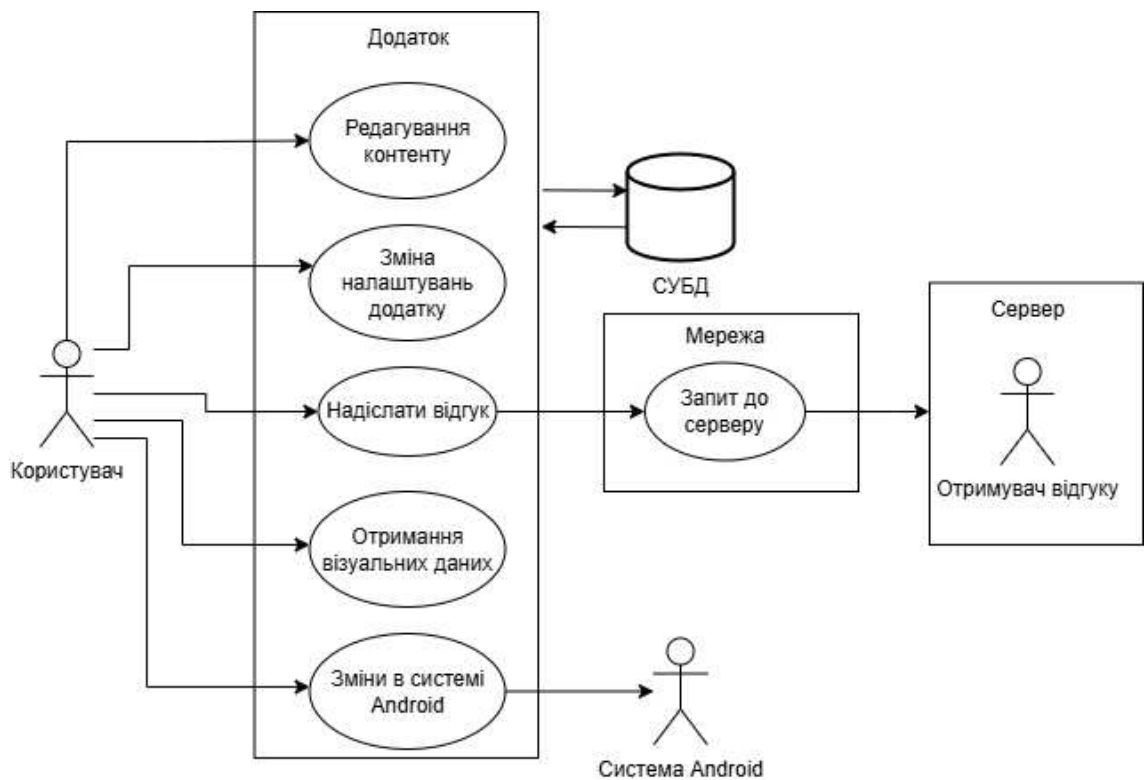


Рисунок 1.6 - Концепція інформаційної системи Галерея - Фотогалерея, Альбом

Додаток створено для використання офлайн. Таким чином, маємо лише один компонент - додаток. Основними функціями додатку є: редагування контенту, зміна налаштувань додатку, надіслання відгук, отримання візуальних даних та зміни у системі Android. Усі зміни зберігаються у локальній базі даних.

Інформаційна система додатку Gallery (Google LLC) тотожна інформаційній системі додатку Галерея - Фотогалерея, Альбом (PhotoZen Studio).

Запропонована концепція реалізації мобільного застосунку фотонотатків (рисунок 1.7) має низку ключових особливостей та організаційних рішень, що дозволяють забезпечити зручність і безпеку користування.

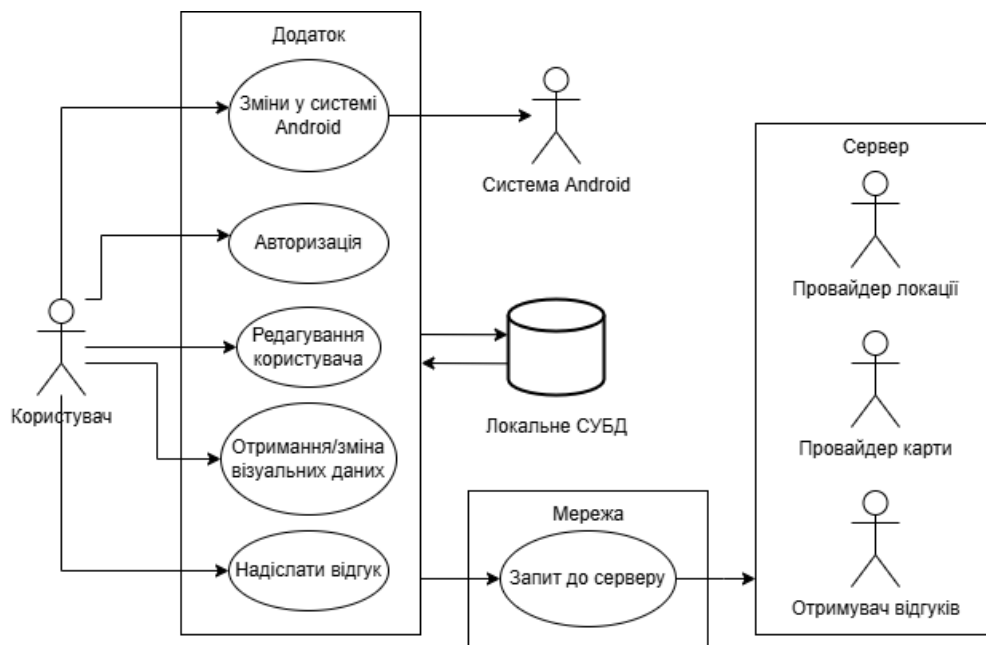


Рисунок 1.7 - Концепція інформаційної системи мобільного додатку

Загальна схема функціонування системи передбачає локальне збереження користувачів і даних, що забезпечує повну незалежність від сторонніх платформ у базових функціях.

Зокрема, користувачі мають змогу проходити авторизацію та редагувати власні дані, що реалізується через локальну базу даних пристрою. Користувач може не лише працювати зі своїм профілем, а й мати можливості адміністрування інших локальних облікових записів. Це особливо корисно в умовах офлайн-середовища.

Щодо реалізації функції зворотного зв'язку, вона виконується поза межами СУБД. Для цього передбачена інтеграція з окремими зовнішніми сервісами — наприклад, через поштові клієнти або Telegram-ботів. Така інтеграція не є критичною для основної роботи додатку, оскільки здійснюється лише за потреби, на запит користувача. Також отримання локації для фотографії та карти буде здійснено за допомогою зовнішнього сервісу.

Функціональні компоненти системи:

1. Система Android відповідає за зміну мови інтерфейсу, запити дозволів (зокрема на використання камери). Мова зберігається у базі даних, запити зберігаються у системних налаштуваннях.
2. Користувач має змогу: переглядати та змінювати свої дані, працювати з папками та фотографіями (створення, редагування, перегляд), надсилати зворотний зв'язок, керувати локальними профілями (у межах пристрою). Усі ці дані зберігаються у базі даних, що гарантує повний контроль користувача над інформацією.
3. Отримувач відгуків — це зовнішній модуль або сервіс, який приймає повідомлення (фідбек) від користувача. Його реалізація залежить від обраного каналу комунікації (електронна пошта, бот тощо).
4. Провайдер локації - зовнішній модуль або сервіс, який надає локацію користувача.
5. Провайдер карти - зовнішній модель або сервіс, який надає карту користувачу.

Для зручної навігації та логічного поділу функціональності додаток умовно розділяється на три основні частини:

1. Блок авторизації

- Екран входу до облікового запису.
- Екран створення нового облікового запису.

2. Основні екрани додатку

- Екран списку фотографій та папок.
- Екран використання камери для створення фото.
- Екран відображення карти для перегляду місць, пов'язаних із фото.
- Екран зворотного зв'язку.
- Екран редагування інформації про фото (назва, опис, метадані).

3. Профіль

- Екран перегляду поточних даних користувача.

- Екран зміни логіну.
- Екран зміни паролю.
- Екран зміни назви облікового запису.
- Екран вибору мови інтерфейсу.

Внутрішня структура додатку зображена на рисунку 1.8.

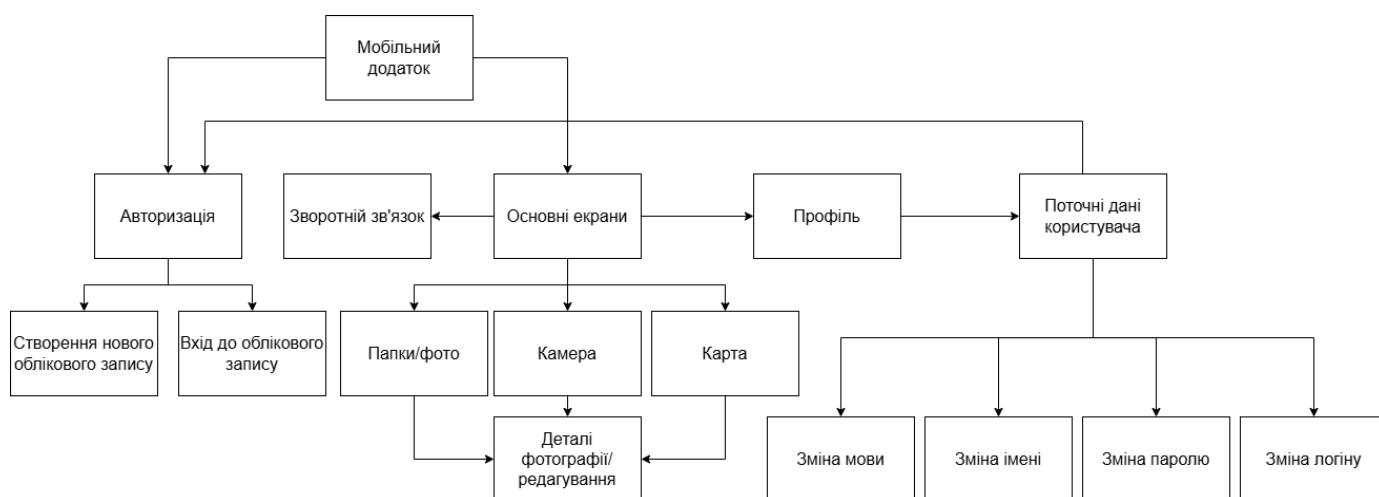


Рисунок 1.8 - Внутрішня структура додатку

1.5 Постановка задачі

Головною метою проекту є створення функціонального та зручного мобільного додатку для збереження, перегляду та організації фотографій користувача. Додаток повинен забезпечувати не лише базове зберігання знімків, але й додаткові можливості — такі як фіксація місця події, редагування метаданих, сортування фотонотатків та взаємодія з інтерфейсом карти. Для досягнення цієї мети проект поділяється на кілька підзадач:

По-перше, це задачі, пов'язані безпосередньо з роботою з фотографіями. Сюди входить реалізація можливості створення фотографій за допомогою камери, їх збереження у локальній пам'яті та обробка метаданих (країна, місто, опис тощо), редагування фотонотатків, а також зручне відображення — як у

вигляді списку, так і на карті. Важливо також передбачити можливість надання користувачем фідбеку щодо окремих записів або функціоналу.

По-друге, необхідно реалізувати систему управління користувачами. Це включає створення та видалення профілів, а також можливість перегляду й зміни персональних даних. У перспективі це дозволить персоналізувати інтерфейс, зберігати індивідуальні налаштування.

По-третє, серед технічних задач стоїть питання адаптації мобільного додатку до різних пристроїв, розмірів екранів та виробників. Необхідно забезпечити коректну роботу інтерфейсу на різних версіях операційної системи Android.

Висновок за розділом 1

Аналізуючи існуючі рішення із впровадження фотогалереї, що підтримує функцію нотатків, було виявлено, що додатки або розростаються до великих додатків, тим самим підтримуючи велику кількість функціоналу, який не є суттєвим, або не містять прив'язки фотографій до карти, що є мінусом, через недостатню зручність перегляду фотографій.

Інтерфейс користувача було вирішено спростити до мінімальних потрібних вимог, які будуть відповідати назві додатку. Звідси, не маємо можливості редагувати фотографії - це вимога до інших додатків.

Додаток буде створено із підтримкою усіх необхідних функцій перегляду, сортування та редагування фотографій.

РОЗДІЛ II. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1 Загальносистемні рішення

2.1.1 Загальний опис системи

Фотонотатки — це мобільний застосунок для Android, створений для зручного збереження та організації фотографій користувача із можливістю прив'язки знімків до конкретного місця. Основною метою розробки є забезпечення інтуїтивного інтерфейсу та функціонального середовища, яке дозволяє легко створювати, переглядати, структурувати та редагувати фотонотатки.

Додаток реалізований як компактне рішення, що дає змогу швидко взаємодіяти з візуальними даними. Головний акцент зроблено на простоті — як у використанні, так і в технічній реалізації. Завдяки цьому, за потреби, застосунок можна легко доповнювати новими функціями або адаптувати під інші вимоги.

Інтерфейс побудований на базі знайомих елементів, таких як списки, що значно підвищує зручність навігації. Дані у додатку розміщуються у логічно впорядкованому вигляді. Користувач може переглядати свої фотографії, додавати нові, змінювати метадані зображень, видаляти непотрібні файли, створювати тематичні папки для групування фото та керувати тегами.

У межах системи також реалізовано керування користувачами. Передбачено можливість створення нового облікового запису, перегляду та редагування даних поточного користувача, зміну логіну, паролю, назви профілю, а також вибір мови інтерфейсу. За потреби, можна видаляти або додавати нові облікові записи.

2.1.2 Розробка діаграми варіантів використання

Діаграма варіантів використання (рисунок 2.1) відображає взаємодію між основними акторами системи: користувачем, операційною системою Android, отримувачем зворотного зв'язку, провайдером локації та провайдером карти. Ці взаємозв'язки демонструють, як кожен учасник взаємодіє із функціональністю застосунку, а також визначають залежності між модулями системи.

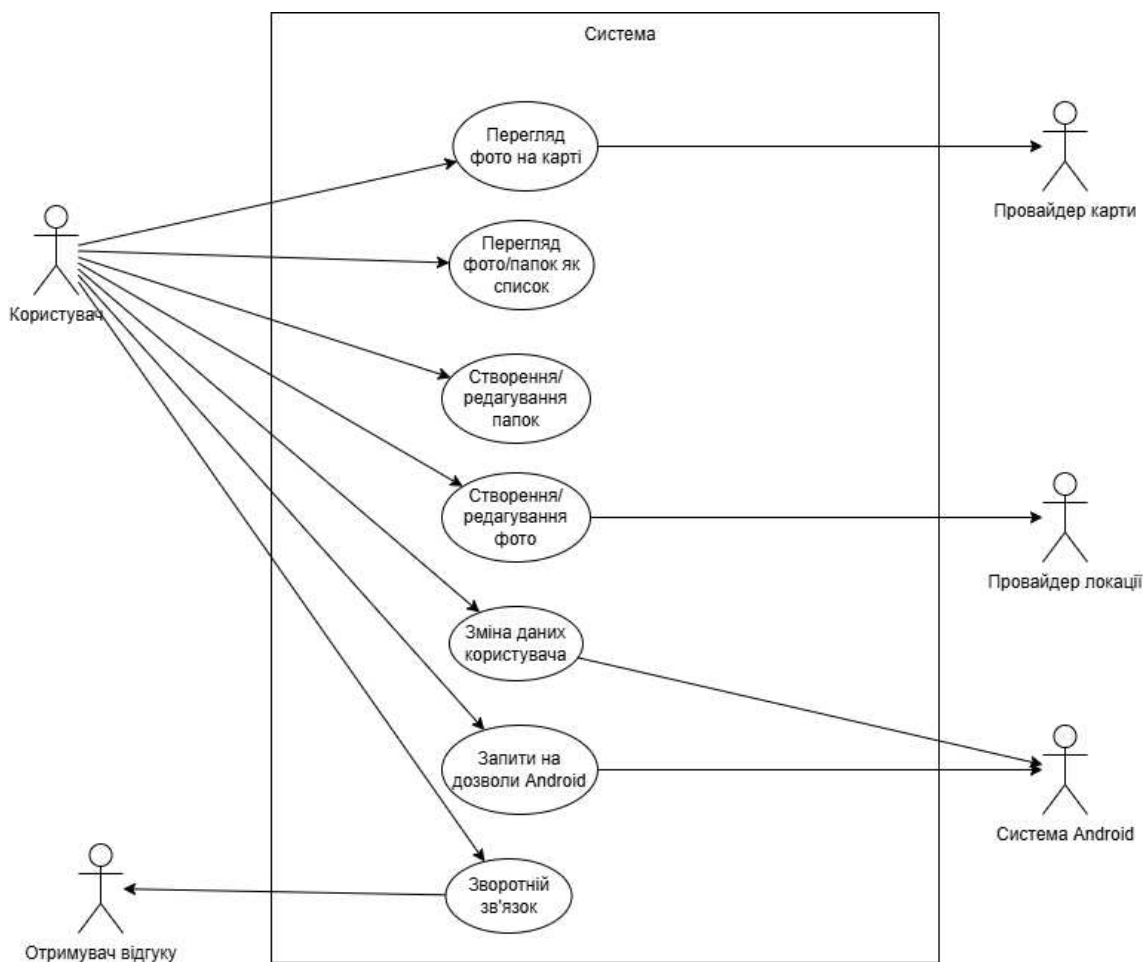


Рисунок 2.1 - Діаграма варіантів використання

Функціональність системи умовно поділяється на два рівні: внутрішні функції, які доступні завжди, незалежно від зовнішніх чинників, та зовнішні функції, що залежать від додаткових умов, зокрема доступу до Інтернету або специфіки Android-платформи.

Внутрішній функціональний рівень охоплює основні можливості програми, які доступні в офлайн-режимі. До них належать дії з обліковими записами користувача: створення, редагування, перегляд і видалення. Подібні операції також можливі для папок і фотографій — користувач може створювати нові записи, редагувати метадані фотографій, переглядати вміст, а також видаляти непотрібне. Окрім цього, система підтримує додавання та видалення тегів для фотографій, що полегшує їхню організацію.

Зовнішній функціональний рівень охоплює ті можливості, які залежать від зовнішніх умов. Наприклад, відображення фотографій на карті (при умові використання зовнішніх сервісів для виконання цієї задачі) потребує активного Інтернет-з'єднання. Аналогічно, надсилання зворотного зв'язку реалізується лише за наявності доступу до мережі. Деякі параметри взаємодії із системою Android, зокрема вибір мови інтерфейсу або надання дозволів на використання камери, залежать від налаштувань та типу локалі операційної системи. Як і з картами, надання локації може залежати від наявності Інтернет-з'єднання у користувача.

У моделі акторів маємо п'ять ролей. Користувач є центральною фігурою системи: він взаємодіє з фотонотатками, керує обліковими записами, виконує налаштування і працює з інтерфейсом. Система Android відповідає за обробку специфічних системних запитів, таких як зміна мови, надання дозволів або реагування на локальні налаштування пристрою. Отримувач зворотного зв'язку є умовною зовнішньою сутністю, яка приймає повідомлення від користувача. Провайдер локації також є зовнішньою сутністю, яка надає місце розташування пристрою для зв'язування фотографії із місцем. Аналогічно провайдер карти відповідає за малювання карти у додатку.

2.2 Рішення з інформаційного забезпечення

2.2.1 Визначення сутностей інформаційної підсистеми

На основі діаграми варіантів використання (рисунок 2.1) можна визначити основні сутності інформаційної підсистеми. Ключовою сутністю є користувач, оскільки система передбачає взаємодію з різними типами користувачів. Кожен користувач має логін (електронну пошту), пароль, ім'я, шлях до фотографії (фото зберігається як шлях до файлу, а не як саме зображення), стан входу в систему (індикатор, чи користувач авторизований), а також обрану мову інтерфейсу.

Сутність мови визначає мову інтерфейсу, яку використовує користувач у системі. Вона містить лише назву мови.

Користувач має змогу створювати ієрархічну структуру даних у вигляді папок(папки у папках заборонено), у яких зберігаються фотографії. Папка визначається лише своєю назвою.

Фотографія є окремою сутністю, яка має назву, шлях до зображення (за тим самим принципом, що і фото користувача), а також опис.

Кожна фотографія може мати декілька тегів. Сутність тега містить назву тега.

Оскільки фотографії мають відображатися на мапі, кожна з них повинна бути прив'язана до певної локації. Сутність локації визначає країну, місто, широту та довготу.

На рисунку 2.2 наведено концептуальну модель даних.

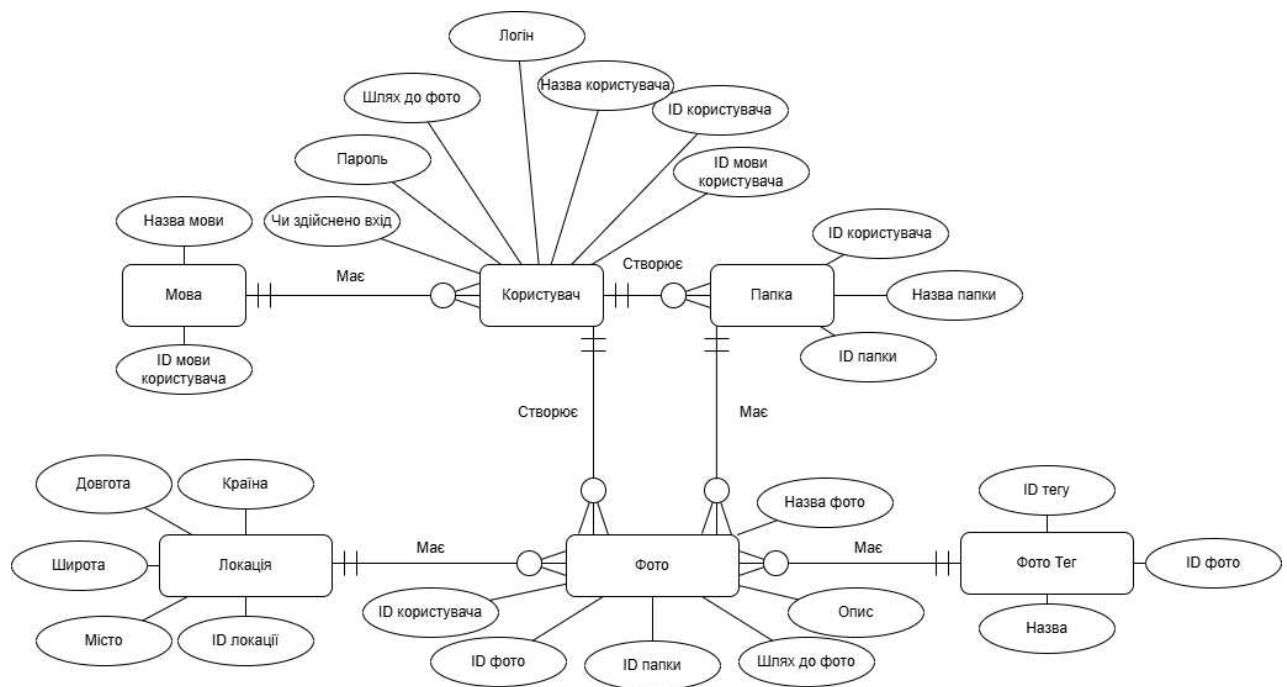


Рисунок 2.2 - Концептуальна модель даних

2.2.2 Розробка логічної моделі даних

На цьому етапі важливо визначити логічну модель[8] для попередньо виділених сутностей. Формування логічної моделі необхідне для побудови правильної структури проекту та забезпечення ефективного зберігання даних у системі.

Логічна модель дозволяє уточнити склад атрибутів кожної сутності, а також встановити зв'язки між ними. Це допомагає не лише сформувати повноцінну структуру бази даних, але й гарантує її працездатність і ефективність при подальшій реалізації.

Логічну модель даних наведено на рисунку 2.3.

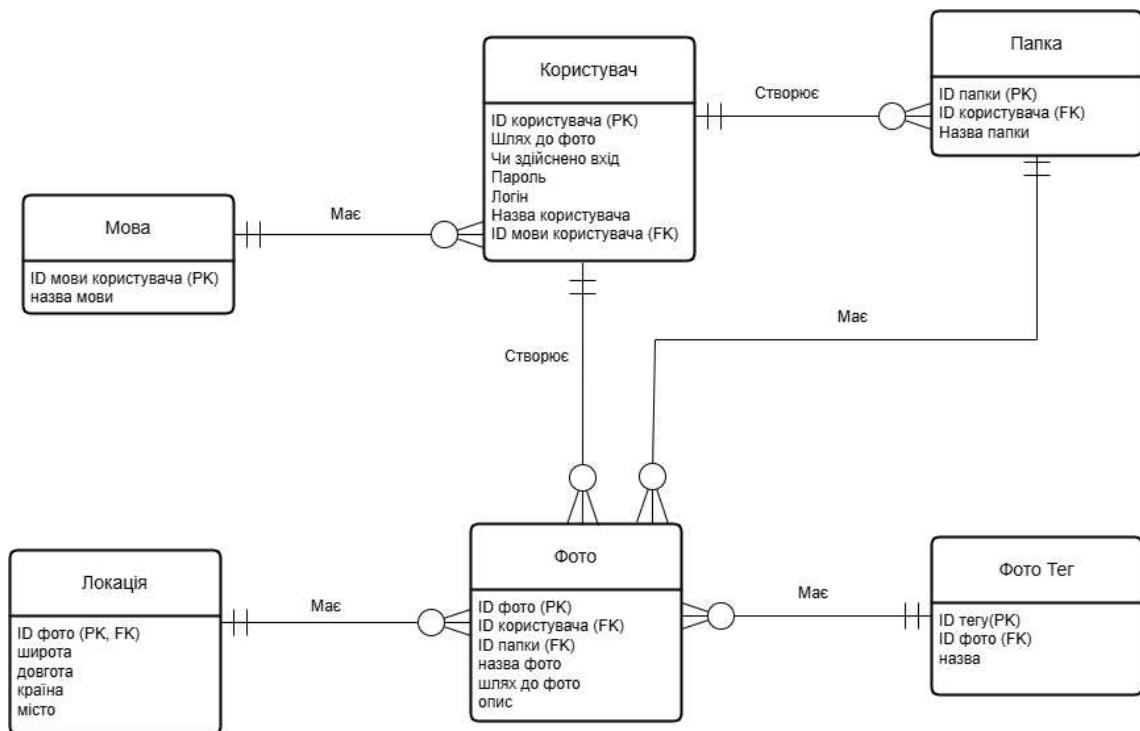


Рисунок 2.3 - Логічна модель даних

Як результат, використовуючи раніше визначені сутності інформаційної системи маємо:

Користувач (ID користувача, шлях до фото, чи здійснено вхід, пароль, логін, назва користувача, ID мови користувача). ID мови користувача та сутність мови пов'язані між собою. Таким чином, відношення 1 (мова) до багатьох (користувачів). ID користувача є головним ключем сутності.

Мова (ID мови користувача, назва мови). ID мови користувача є головним ключем сутності.

Папка (ID папки, ID користувача, назва папки). ID користувача і сутність користувача пов'язані між собою. Таким чином, відношення 1 (користувач) до багатьох (папок). ID папки є головним ключем сутності.

Фото (ID фото, ID користувача, ID папки, назва фото, шлях до фото, опис). ID користувача і сутність користувача пов'язані між собою. Таким чином, відношення 1 (користувач) до багатьох (фото). ID папки і сутність папки

пов'язані між собою. Таким чином, відношення 1 (папка до багатьох (фото)). ID фото є головним ключем сутності.

Локація (ID фото, широта, довгота, країна, місто). ID фото є тотожним ID фото у сутності фото. Таким чином, відношення 1 (локація) : 1 (фото). ID фото є головним ключем сутності.

Фото Тег (ID тегу, ID фото, назва). ID фото і сутність фото пов'язані між собою. Таким чином, відношення 1 (фото) до багатьох (фото тег). ID тегу є головним ключем сутності.

2.2.3 Розробка фізичної моделі даних

Наступним кроком буде створення фізичної моделі на основі логічної. Для цього визначимо обмеження на дані та тип даних.

У таблицях 2.1-2.6 наведено обмеження для сутностей фізичної моделі даних.

Таблиця 2.1 - Таблиця сутності “Користувач” (“User”)

Назва атрибуту в логічній моделі	Назва атрибуту в фізичній моделі	Тип	Обмеження	Тип ключа
ID користувача	id	Integer	NOT NULL	PK
ID мови користувача	lang_id	Integer	NOT NULL	FK (id:Language)
Шлях до фото	photo_path	Varchar	NULL	
Чи здійснено вхід	entered	Integer	NOT NULL	
Пароль	password	Varchar	NOT NULL	
Логін	login	Varchar	NOT NULL	
Назва користувача	username	Varchar	NOT NULL	

Таблиця 2.2 - Таблиця сутності “Мова” (“Language”)

Назва атрибуту в логічній моделі	Назва атрибуту в фізичній моделі	Тип	Обмеження	Тип ключа
ID мови користувача	id	Integer	NOT NULL	PK
Назва мови	name	Varchar	NOT NULL	

Таблиця 2.3 - Таблиця сутності “Папка” (“Folder”)

Назва атрибуту в логічній моделі	Назва атрибуту в фізичній моделі	Тип	Обмеження	Тип ключа
ID папки	id	Integer	NOT NULL	PK
ID користувача	user_id	Integer	NOT NULL	FK (id:User)
Назва папки	title	Varchar	NOT NULL	

Таблиця 2.4 - Таблиця сутності “Фото” (“Photo”)

Назва атрибуту в логічній моделі	Назва атрибуту в фізичній моделі	Тип	Обмеження	Тип ключа
ID фото	id	Integer	NOT NULL	PK
ID користувача	user_id	Integer	NOT NULL	FK (id:User)
ID папки	folder_id	Integer	NULL	FK (id:Folder)
Назва фото	title	Varchar	NOT NULL	
Шлях до фото	photo_path	Varchar	NOT NULL	
Опис	description	Varchar	NULL	

Таблиця 2.5 - Таблиця сутності “Фото тег” (“Photo Tag”)

Назва атрибуту в логічній моделі	Назва атрибуту в фізичній моделі	Тип	Обмеження	Тип ключа
ID тегу	id	Integer	NOT NULL	PK
ID фото	photo_id	Integer	NOT NULL	FK (id:Photo)
Назва	name	Varchar	NOT NULL	

Таблиця 2.6 - Таблиця сутності “Локація” (“Location”)

Назва атрибуту в логічній моделі	Назва атрибуту в фізичній моделі	Тип	Обмеження	Тип ключа
ID локації	id	Integer	NOT NULL	PK,FK (id:Photo)
широта	latitude	Float	NOT NULL	
довгота	longitude	Float	NOT NULL	
країна	country	Varchar	NOT NULL	
Місто	town	Varchar	NOT NULL	

Фізичну модель даних наведено на рисунку 2.4.

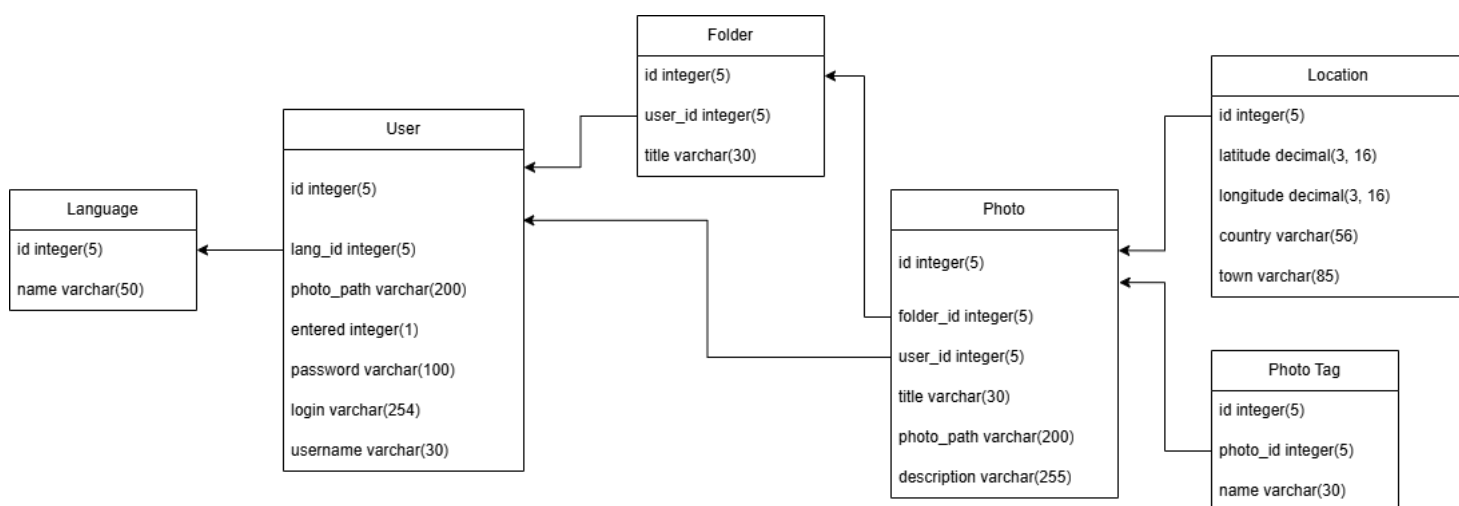


Рисунок 2.4 - Фізична модель даних

2.3 Рішення з технічного забезпечення

2.3.1 Вимоги до технічного забезпечення

Програма буде розроблена для використання у системі Android. Для забезпечення ефективної роботи необхідно врахувати наступні технічні вимоги пристрою:

- Операційна система: Android
- Мінімальний сдк: 28
- Максимальний сдк: 35
- Оперативна пам'ять: 2 Гб
- Пам'ять пристрою: 7 Мб
- Підключення до мережі (опціонально)
- Наявність камери

2.3.2 Склад технічного забезпечення

Для забезпечення коректної роботи мобільного додатку користувач повинен мати смартфон, що працює під керуванням операційної системи Android. Пристрій має підтримувати базові технічні вимоги, необхідні для повноцінного функціонування застосунку.

Обов'язковою умовою є наявність вбудованої камери, оскільки частина функціоналу додатку пов'язана з створенням фотографій. Також пристрій має бути підключений до мережі Інтернет (через Wi-Fi або мобільні дані), що необхідно для використання розширених можливостей додатку, таких як передача даних для відгуку користувача, взаємодія із сервером для отримання карти тощо.

2.4 Рішення з програмного забезпечення

2.4.1 Розробка схеми структури інформаційної підсистеми

Користувач додатка Photo Notes може взаємодіяти із базою даних, виконувати функції: авторизації, адміністрування користувачів, обробки фото, та взаємодії із системою. Це ті запити, які ми маємо обробляти за допомогою бази даних та зберігати результати. Наступною операцією є надсилання відгуку. Зворотній зв'язок буде реалізовано через використання зовнішнього сервісу. Отримання локації також буде зовнішнім сервісом, оскільки так буде більш ефективніше, особливо на телефонах з мінімальними технічними характеристиками. Зображення карти також потребує зовнішнього втручання через те, що малювання карти “від руки” не буде містити актуальної інформації та буде додатково вантажити телефон користувача.

Важливо зазначити, що база даних не використовує зовнішнього сервісу, тому усе буде зберігатися на пристрої користувача.

Результат схеми взаємодії програмних модулів наведено на рисунку 2.5.

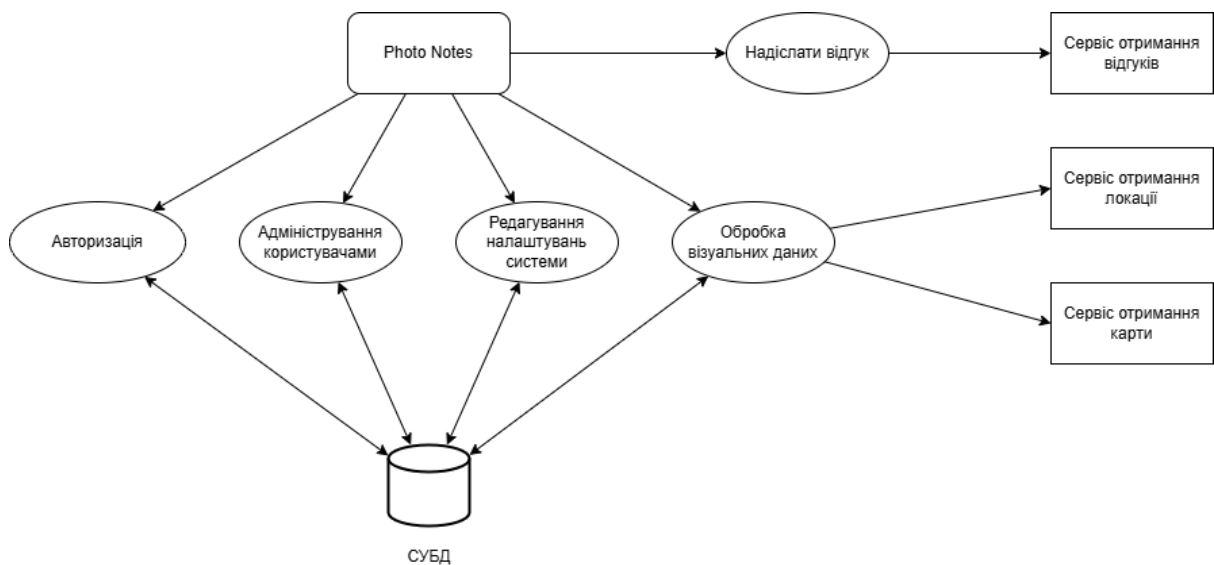


Рисунок 2.5 - Схема взаємодії програмних модулів

2.4.2 Розробка діаграми компонентів

Для того, щоб виявити залежності та змінні у додатку, потрібно побудувати діаграму компонентів (рисунок 2.6).

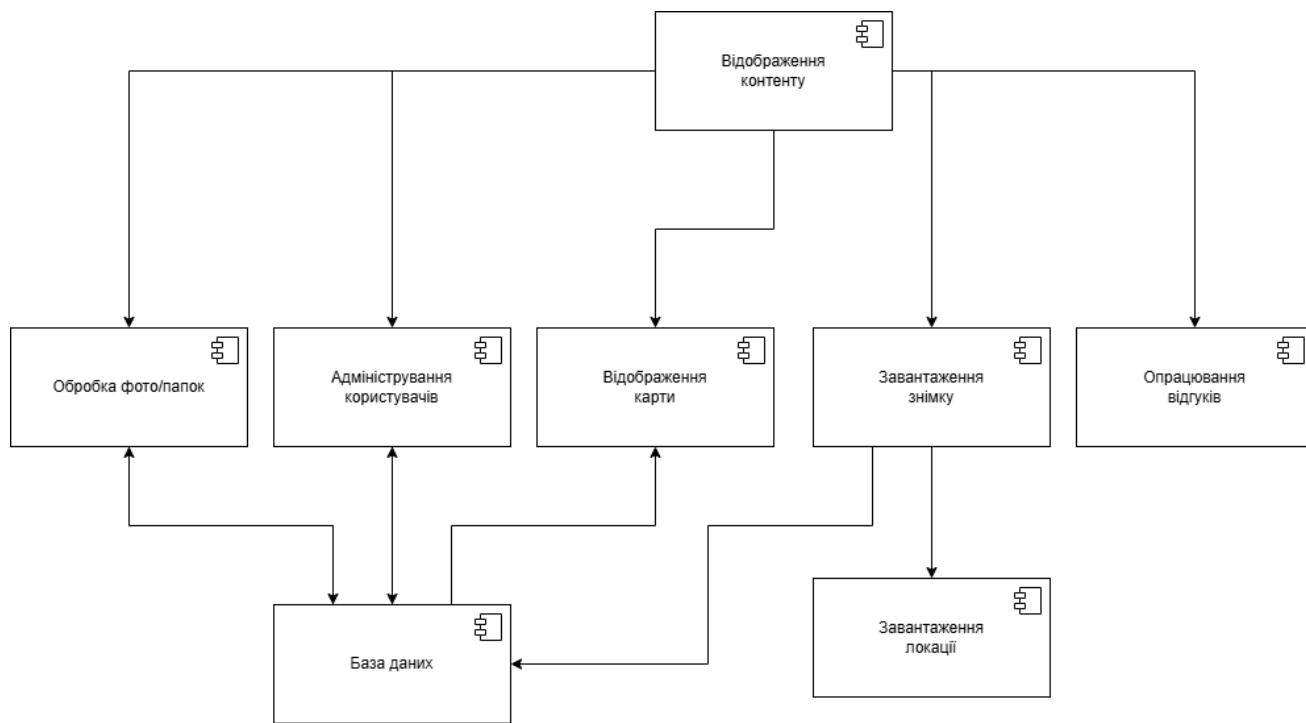


Рисунок 2.6 - Діаграма компонентів

Змінними компонентами можуть бути: відображення контенту (використовуємо Compose чи View), відображення карти (використовуємо Google Maps, власну реалізацію, малюнок), бази даних (реалізація може бути будь-якою), оброблення фото/папок (визначення їх формату), адміністрування користувачів(також визначає їх формат), завантаження знімку (використання cameraLauncher, cameraX тощо), завантаження локації (використання сервісу або власна реалізація), опрацювання відгуків (Telegram, google drive тощо.).

Тепер визначимо залежності між цими компонентами.

Для реалізації обробки фото/папок, адміністрування користувачами, відображення карти, завантаження знімку та опрацювання відгуку необхідно використовувати візуальний інтерфейс.

Обробка фото/папок, адміністрування користувачів та відображення карти потребують даних від бази даних.

Обробка фото/папок, адміністрування користувачів та завантаження знімку може змінювати дані у базі даних.

Для поєднання знімку із локацією будемо використовувати залежність завантаження знімку від завантаження локації.

2.4.3 Розробка діаграми переходів станів

Діаграма переходів станів була спроектована таким чином, щоби захопити усі шари навігації (рисунок 2.7).

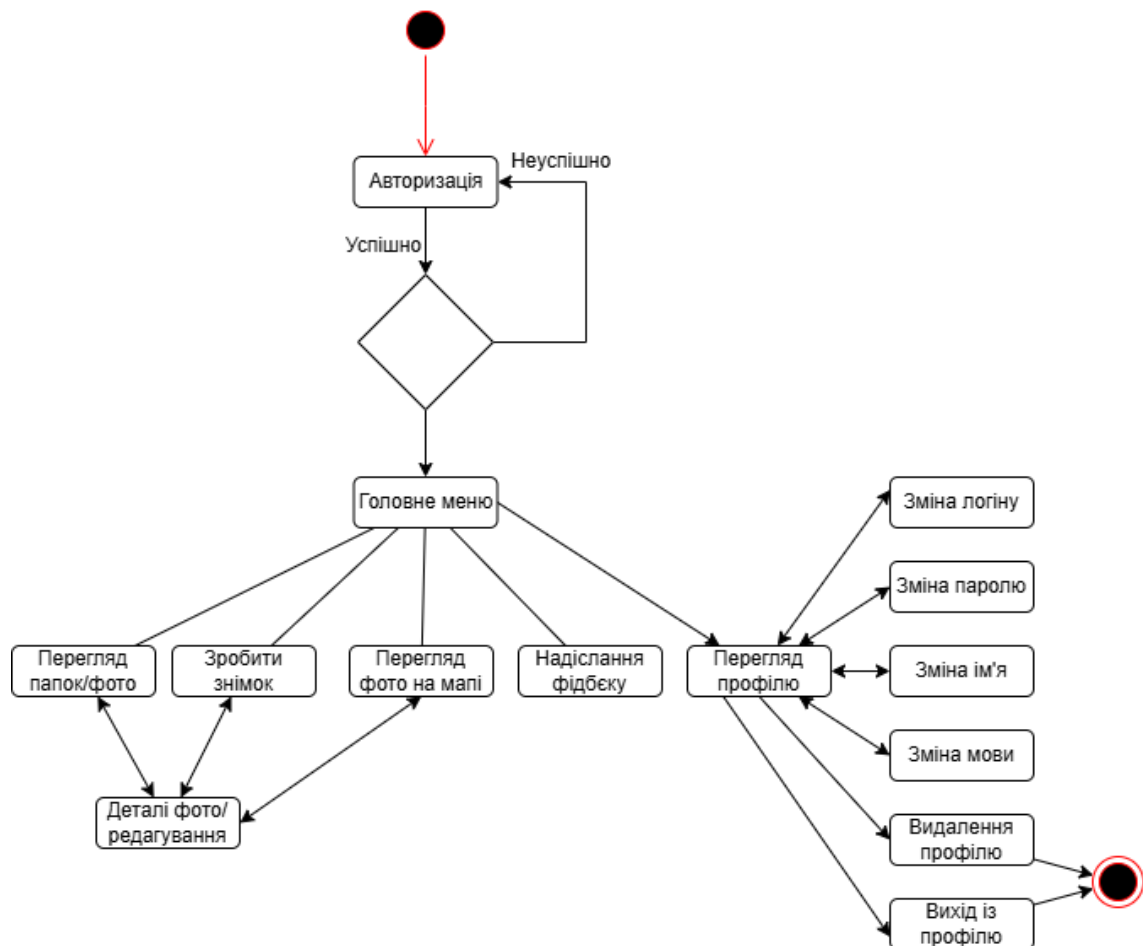


Рисунок 2.7 - Діаграма переходу станів

Користувач починає із авторизації. Якщо було здійснено вхід до існуючого облікового запису або створено новий, то він потрапляє до головного меню.

Головне меню містить інформацію про перегляд фото/папок, використання камери, карту із фотографіями та надсилання зворотнього зв'язку.

Після того як користувач зробив знімок, натиснув на фото на карті або обрав фото із списку із фотографіями, користувач потрапляє до екрану деталей обраного знімку.

Також, користувач може потрапити із головного меню до екрану перегляду профілю. Тут можна знайти всю інформацію про користувача, змінити її за допомогою екранів зміни логіну, пароллю, імені. Якщо видалити профіль або вийти із облікового запису стан користувача потрапляє у кінцевий стан.

2.4.4 Розробка проекту інтерфейсу інформаційної підсистеми

У мобільних додатках велика увага приділяється інтерфейсу користувача. Тому треба його розробити таким чином, щоб орієнтуватися у програмі було досить легко.

Шрифти є важливою складовою у інтерфейсі, що дозволяє користувачу краще сприймати текст у додатку[1]. Але шрифти не будуть працювати без коректної гами[11], тому додаток буде виконано у блакитно-пастельних тонах (використано шрифт Comic Relief).

Так, для вводу даних (логіну, пароллю, текст відгуку тощо) будемо використовувати текстове поле (рисунок 2.8).



Рисунок 2.8 - Приклад текстового поля для мобільного додатку

Для відображення папок та фото на одній сторінці будемо використовувати списки. Тому що вони досить часто використовуються та є інтуїтивно зрозумілим для кінцевого користувача (рисунок 2.9).



Рисунок 2.9 - Приклад списків

Для виконання будь-якої дії з боку користувача будемо використовувати кнопки (рисунок 2.10).



Рисунок 2.10 - Приклад кнопки

Для привертання уваги, що фото є змінним будемо використовувати відповідно значок редагування (рисунок 2.11).



Рисунок 2.11 - Приклад значку змінного фото

2.4.5 Розробка діаграми розгортання підсистеми

Діаграма розгортання - це діаграма, яка демонструє взаємозв'язок додатку із різними сервісами. Діаграма відображає фізичну модель поєднання (зв'язків) між компонентами.

Діаграму розгортання наведено на рисунку 2.12.

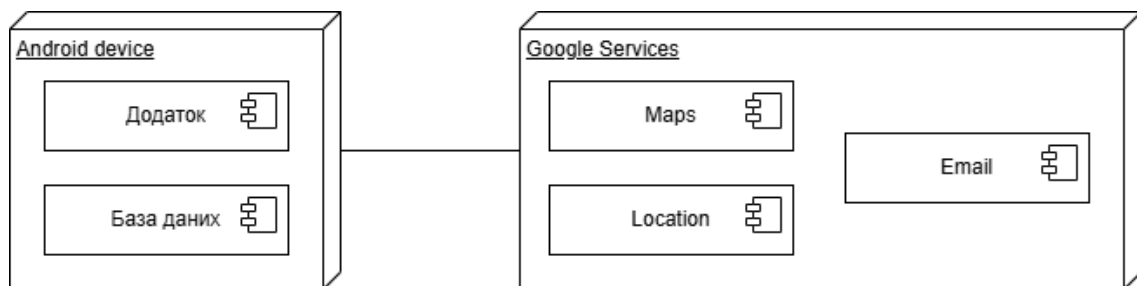


Рисунок 2.12 - Діаграма розгортання системи

Схема передбачає використання таких гугл сервісів, як Google Maps, GMS Location, Gmail (Google Email). На девайсі користувача буде розташовано лаунчер мобільного додатку. Усередині додатку буде використовуватися база даних. Як результат, додаток має дві складові: внутрішня - мобільний пристрій та зовнішній - гугл сервіси.

Висновок за розділом 2

Було спроектовано інформаційну систему згідно аналізу. Визначено функції користувача у системі та створено діаграму варіантів використання. Згідно із цих даних, створено концептуальну модель даних, з'ясовано обмеження даних та їхню залежність. Встановлено логічну та фізичну моделі даних. Для коректної роботи встановлено вимоги до пристрою.

Вирішено використовувати зовнішні сервіси для відтворення інформації, а не для збереження. Для забезпечення більшої конфіденційності даних базу даних буде розміщено напряду у додатку. Для відображення інтерфейсу користувачу

буде використовуватися знайомі елементи дизайну із гармонійною гамою. Додаток має лише одного користувача, який може контролювати облікові записи та змінювати контент записів усередині них.

РОЗДІЛ III. РЕАЛІЗАЦІЯ ПРОЕКТУ ПІДСИСТЕМИ

3.1 Аргументація вибору засобів розробки

Для створення мобільного додатку для Android девайсів будемо користуватися середою розробки Android Studio, оскільки це є найбільш популярним рішенням для розробки мобільних додатків, що підтримує Android Framework. Також ця середа активно підтримується Google. Таким чином, можна використовувати останні стабільні фічі для розробки проекту.

Основними мовами програмування для додатків під Android є Java, Kotlin. Зосередимося на їх перевагах та недоліках.

При використанні мови програмування Java перевагами є велика ефективність алгоритмів та пам'яті. Також Java є більш зрозумілою мовою через невикористання додаткових синтаксичних інструментів, які потребують уваги. Останньою перевагою є швидкість компіляції проекту.

При використанні мови програмування Kotlin перевагами є швидкість написання проектів через використання синтаксичних інструментів[12]. Також, важливо зауважити, що синтаксичні інструменти складають дуже велику частину kotlin-проектів, що дає можливість використовувати делегати, розширення, nullable типи даних тощо.

Кожна мова підтримує ООП.

Було прийнято рішення про необхідність вибору мови Kotlin. По-перше через її зручність, по-друге, тому що більшість android проектів зараз мігрує із Java до Kotlin, тож обрано рух у напрямку тенденцій.

Є два способи відображати контенту у Android. За допомогою View/XML та Jetpack Compose. Розберемо, що обирати.

View/xml - підхід, що раніше підтримувався Google та який є основою для відображення у android. Значною перевагою є простий спосіб оптимізації

створених xml. Але, через необхідність мати доступ до view та роботи із станом цієї view, часто виникає плутанина у коді, яка поєднує шар відображення із шаром збереження.

Jetpack Compose - спосіб, що зараз активно просувається Google. Має декларативну форму написання. Вирішив проблему доступу та стану до відображення через реактивність. Має складність у оптимізації через систему рекомпозицій.

Jetpack Compose є більш вдалим варіантом, оскільки декларативний тип написання дуже зменшить кількість часу для проекту. Проблема оптимізації не буде значною через використання статичних елементів.

Наступним кроком є визначення бази даних. Її можна створити за допомогою програм-серверів, таких як: PostgreSQL, MySQL, MariaDB тощо. Або використати бібліотеки, які надають реалізацію бази даних напряду у додатку. Такими бібліотеками можуть бути: Room, Realm, Firebase Database.

Найкращим способом для даного проекту є реалізація напряду бази даних через бібліотеку Room[4]. Тут є і підтримка від Google, і перевірка запитів на етапі компіляції, і підтримка реактивного підходу, і зручності із написання коду.

Далі, треба обрати як саме буде відображатися карта. Тут також є невелика кількість підходів. Але, на мій погляд, найефективнішим рішенням є відображення за допомогою Google Map[6]. Це завжди актуальна інформація, це підтримка від google при написанні коду, що спрощує його реалізацію. Недоліком є необхідність мережевого з'єднання.

Реалізація завантаження знімків

Оскільки немає необхідності у створенні фоторедактора, маємо можливість використати спрощене API(camera launcher), яке використовує вбудований додаток камери для отримання фотографій[13].

Реалізація завантаження локації

Є два популярних способа отримати локацію у додатку android. Першим є використання location manager, але ми його не будемо використовувати через

нестабільний результат. Другим способом є використання gms Location API[2], який є дуже швидким, але також зберігається недолік із необхідністю мати мережеве з'єднання.

Реалізація опрацювання відгуків

Для реалізації системи відгуків також буде використано зовнішній додаток, а саме gmail [14]. Таким чином, це також функція що потребує мережевого з'єднання. Цей спосіб є легким у написанні та зрозумілим для всіх користувачів.

3.2 Розробка структури програмного забезпечення

Початковий стан програми - це користувач додатку, який знаходиться на сторінці з авторизацією. Користувач може створити новий обліковий запис та увійти до нього.

Після цього, користувач потрапляє до сторінки папок та фото. Навігація буде зроблена за допомогою Bottom Navigation Bar, що є інтуїтивно зрозумілим. Нижня навігація має розділ папок, розділ, камери, розділ карти, розділ зворотнього зв'язку.

Із розділу папок та фото, якщо користувач обирає будь-яке фото, він переходить до екрану деталей обраного фото.

На екрані деталей користувач може редагувати дані фотографії, видалити її, додавати та видаляти теги, пересувати фото до папки.

На екрані камери, користувач буде мати лише одну кнопку “зробити знімок”. Спочатку вона буде запрошувати запити на дозволи для роботи із камерою та роботою із локацією. Якщо користувач відхилить запит на камеру 2 рази, то він побачить повідомлення про те, що необхідно ввімкнути дозвіл для роботи цього екрану. Якщо дозвіл на роботу із камерою було надано, користувач

взаємодіє із камерою. Після успішного знімку, користувач переходить до екрану деталей із створеним знімком.

Екран карти відображає ті фотографії, до яких було можливо отримати локацію. Якщо обрати фотографію на карті, користувач переходить до екрану деталей.

Екран відгуку має поля, необхідні для створення відгуку та кнопку для надсилання запиту.

Окрім нижньої навігації, використовується Top Bar, який містить ім'я облікового запису та його фотографію. При натисканні на фото, користувач переходить до екрану профілю.

Профіль має ім'я облікового запису та фотографію. При натисканні на фотографію, користувач може змінити її. Також екран містить декілька кнопок із наступними функціями(перехід до екранів): зміна логіну, зміна паролю, зміна імені, зміна мови, видалення профілю, вийти із профілю. Результатом останніх двох функцій є перехід користувача до екрану авторизації.

3.3 Результати розробки програмного забезпечення

Мета проекту була досягнута як результат виконання роботи. Програма містить такі екрани: авторизація (рисунок 3.1), папки, деталі фото (рисунок 3.2), камера, карта, відгук, профіль, зміна логіну, зміна паролю, зміна імені, зміна мови (рисунок 3.3). Програма має наступні функції: додавання тегів та їх видалення, використання камери для знімків, додавання знімків до папок, редагування фото, перегляд фото на карті, надсилання відгуків, зміна даних користувача, видалення профілю, зміна облікових записів.

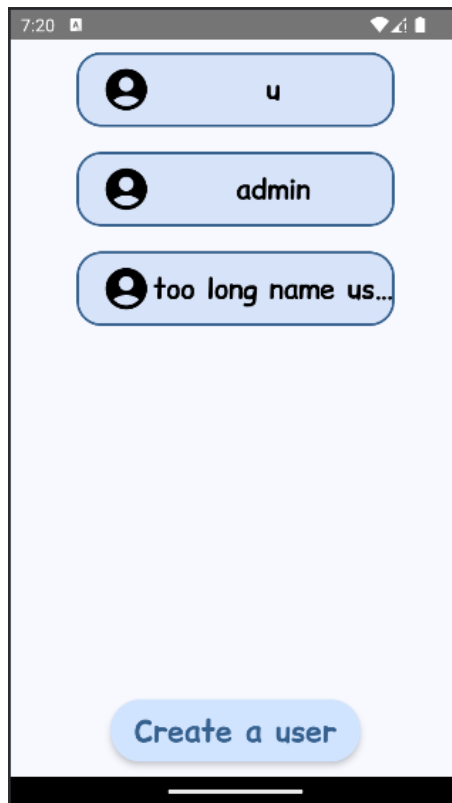


Рисунок 3.1 - Реалізація авторизації у додатку

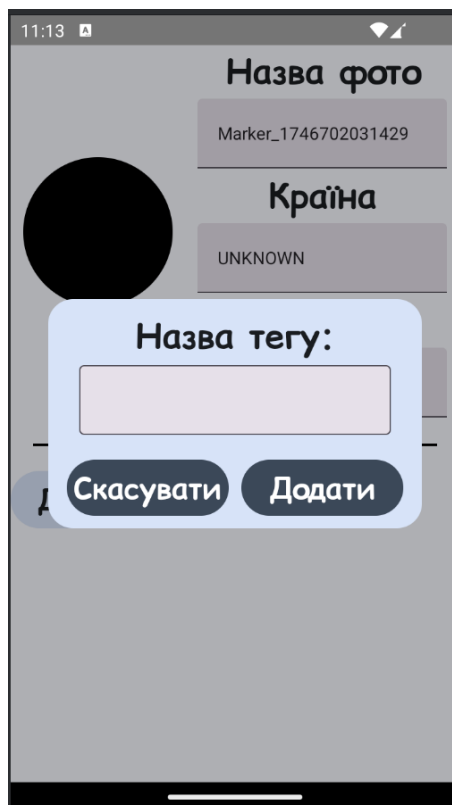


Рисунок 3.2 - Реалізація деталей фотографії у додатку

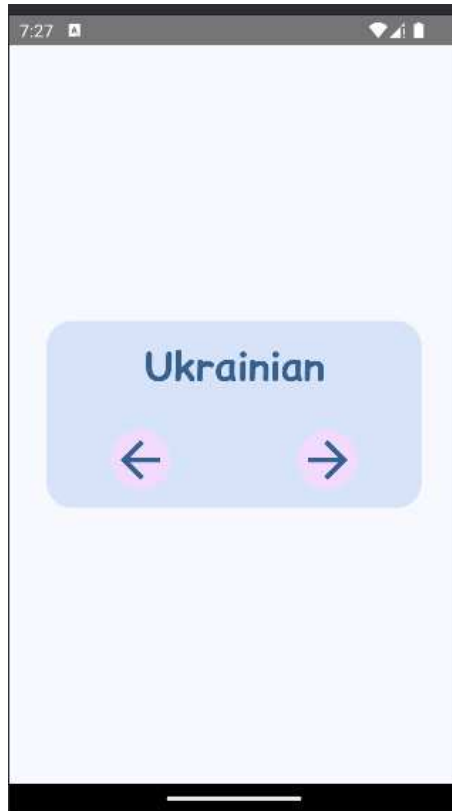


Рисунок 3.3 - Реалізація зміни мови у додатку

Текст програми наведено в додатку Б.

Інструкцію користувача наведено в додатку В.

Висновок за розділом 3

Отже, було порівняно та проаналізовано різні технології для написання додатків під Android. Обрані технології є найефективнішими. Серед них маємо наступні відповідники: середовище розробки - Android Studio, мова програмування - Kotlin, спосіб відображення - Jetpack Compose, база даних - Room Database, відображення карти - Google Map, реалізація камери - camera launcher API, реалізація відгуків - gmail.

Структуру програмного забезпечення та навігаційні компоненти розроблено згідно проектування програми. Логіка навігації програми коректно відпрацьовує.

Як результат, мета проекту була досягнута та програма запускається відповідно до вимог до програмного забезпечення. Усі функції, закладені у програмі, виконуються відповідно до логіки програми. Додаток швидко працює, що є показником ефективності обраних технологій та поєднання їх між собою.

РОЗДІЛ IV. ОХОРОНА ПРАЦІ

4.1 Загальні питання охорони праці

При розробці будь-якого застосунку постає питання охорони праці. Це є важливим, оскільки забезпечення безпеки безпосередньо впливає на кінцевий результат. Під безпекою найчастіше мають на увазі створення таких умов праці, які не завдаватиме травматизму, захворюванням.

Питання охорони праці дотримуються як у підприємствах, так і індивідуально.

Для того, щоб задовольнити питання про охорону праці в Україні використовується Закон “про охорону праці”. Так наприклад, якщо це підприємницька діяльність, роботодавець має створити такі умови, які б регулювалися нормативними актами та сприяли плідній довгостроковій, безпечній праці. Працівник, навпаки, повинен використовувати створені умови, для власної безпеки та безпеки своїх товаришів.

Сучасними методами охорони праці можна вважати: обробка даних про виробничі ризики, аналіз; санітарні заходи, у межах системи, в якій діє охорона праці; аналіз нещасних випадків, та коригування подальшого плану згідно із результатом.

4.2 Структура управління питаннями охорони праці на підприємстві

Для того, щоб на підприємстві підтримувати охорону праці створюють організаційну структуру, функціями якої є забезпечення безпеки для працівників (на всіх рівнях) та створення безпечних умов праці. Якщо вимоги виконуються та структура відповідає вимогам чинного законодавства, трудова діяльність у даному підприємстві вважаються можливою.

Структурно управління питаннями охорони праці на підприємстві можна поділити на керівників, служби та працівників.

До складу керівників входить керівник підприємства, який займається питаннями з охороною праці, та який несе відповідальність за все підприємство. Також, це керівники підрозділів у підприємстві. Їх обов'язками є проведення інструктажів та дотримання правил охорони безпеки в межах свого підрозділу.

Служби - це окремий підрозділ, метою якої є розробка політики з питань охорони праці. Здебільшого вона займається контролем за дотриманням правил, перевіряє технічний стан речей на підприємстві, веде документації із політики правил. Іноді служби виконують функції комісії, функціями якої є розгляд пропозицій щодо покращення умов праці.

Працівники мають дотримуватися своїх обов'язків у сфері охорони праці. Також, крім вимог політики служби, кожен працівник має користуватись індивідуальними засобами захисту, та, якщо комітет був створений, запропонувати способи покращення умов праці.

4.3 Шкідливі та небезпечні фактори, що супроводжують працю програміста

Хоча програміст - це не та праця, яка пов'язана із якимось фізичними умовами праці, які є очевидно небезпечними, все одно це завжди велика кількість стресу, напруги.

По-перше, весь час, який продовжується праця, найчастіше відбувається в сидячому положенні. Так, зараз деякі програмісти почали використовувати конструкції, які дозволяють змінювати положення під час написання коду. Так наприклад, ми можемо балансувати між сидячим та стоячим графіками. Але, велика кількість так і працює сидячи, що призводить до проблем із кровообігом.

Також, якщо програміст не слідкує за осанкою, в майбутньому це можуть бути проблеми із шиєю.

Великою проблемою є використання гнучких графіків у компаній. Досить часто це переривання сну програміста через важливі фікси у 3 часа ночі. Це може спричинити порушення режиму сну, згодом до дратівливості.

Велика кількість проблем виникає через складність задач, які постають перед користувачами. Так наприклад, з'являється необхідність відстежування доступів (власних та колег). Чим більше прав ми маємо, тим більшим стресом це для нас впливає. Ще одним фактором є дедлайни, які змушують програміста сидіти більше часу перед монітором, “швидше” думати, що є дуже великою напругою.

Не варто недооцінювати роботу кисті. Використання не спеціалізованих клавіатур зумовлює хронічні захворювання. Використання мишки зумовлює напруження м'язів, що має той самий результат.

Якщо користувач досить близько розташований до монітору, виникає напруження очей, що, у свою чергу, зумовлює біль, спазми. Також, якщо користувач багато часу фокусує свій погляд на екрані, виникає почервоніння очей, подразнення, погіршення зору. Більшість програмістів працюють у закритому приміщенні, що часто зумовлює недостаток світла, що призводить до зниження зорової концентрації. Також, закриті приміщення означає недостатню вентиляцію. Через це може виникати головний біль та проблеми із диханням.

4.4 Ергономічні вимоги до робочого місця при роботі за комп'ютером

Ергономіка робочого місця визначає продуктивність працівників, їх комфорт та здоров'я. Через що велика увага привертається дотриманню правильної організації робочого місця.

Ергономіка при роботі за комп'ютером характеризується рівнем світла, шуму. Основною складовою є розміщення обладнання.

Конкретніше, якщо користувач взаємодіє із монітором, то відстань між екраном та очима повинна бути 50-70 см. Таким чином, можна зменшити кількість негативного опромінення та зберегти зір. Використання старих моніторів також сприяє втомі та напруженню очей. Саме тому застосування сучасних моніторів є важливим.

Оскільки програміст досить часто контактує із клавіатурою та мишею, треба подбати про правильне їх розташування. Рекомендації говорять, що кут між плечем та передпліччям має становити 90 градусів для усунення будь-яких травм у майбутньому.

Використання стільців, крісел, стільниць також є важливою характеристикою, яка впливає на навантаження хребту. Треба зосередити увагу на положенні стільця та його регулюванні, таким чином, щоб кут у колінах був приблизно 90 градусів. Рекомендованою висотою стільниці є 72-75 см. Для збільшення комфортних умов працівника важливо уникати “забруднення” речами робочого місця.

До складової розміщення обладнання входить рівень освітлення робочої зони. Освітлення не має викликати відблисків на екрані монітору. Здебільшого використовується пов'язка із природного освітлення та, у темну пору, штучного освітлення у 300-500 люкс.

Важливою умовою є температура у приміщенні. Маємо рекомендовану температуру 20-24 градусів, вологістю 40-60 відсотків. Повітря має отримуватися природним шляхом або штучним із ретельною системою вентиляції.

Ще ергономіка визначає здійснення перерв протягом праці. Так, кожну годину потрібно виконувати 5 хвилин перерву, розім'яти очі. Це корисно, оскільки ми послаблюємо напруження очей та відволікаємося від рутинного напруження через складні задачі.

1.Останньою характеристикою є психологічна ергономіка. Це і оформлення офісу/робочого місця, яке не потребує додаткового фокусування на елементах.

4.5 Нормативні вимоги до робочої площі на одного працюючого за комп'ютером

Для дотримання комфортних умов щодо робочої площі працівника існують нормативні вимоги, які забезпечують безпечну та продуктивну працю.

Метою цих норм є забезпечення необхідним простором для правильного розміщення обладнання та надання нормованого повітря (системи вентиляцій) та освітленості (природного та штучного).

Відповідно до норм, мінімальною робочою площею працівника має становити 6 м².

Якщо це робочий офіс. Рекомендована відстань між моніторами має складати 1.5 метра. Відстань між сусідніми столами - не менше метру. Якщо столів багато, то рекомендована ширина простору для проходу становить 0.6-0.8 метри.

За порушення нормативних вимог роботодавець може отримувати штрафи, приписи про усунення порушень.

Із додаткових вимог також можна виділити:

- На одну особу має приходити 20 м³ повітря.
- Висота приміщення для офісів складає 2.5 м (для навчальних закладів 2.7 м).
- Швидкість руху повітря в приміщенні складає 0.1-0.2 м/с.
- Стандартний рівень шуму 50 дБ у офісах, 40 дБ у навчальних закладах.
- Необхідним є забезпечення пожежної безпеки.
- Мають бути зони відпочинку.

Висновок за розділом 4

Було проаналізовано загальні питання охорони праці та визначено як відбувається регулювання процесу праці на підприємствах та як у індивідуальному порядку. Досліджено сучасні методи охорони праці.

Потім, було розглянуто питання із структурами управління питаннями охорони праці на підприємстві та виявлено, що керівники та службовці мають свої обов'язки, засоби захисту.

Згодом було визначено шкідливі та небезпечні фактори, що супроводжують програміста. Проаналізовано поради, яких слід дотримуватися під час праці.

Наостанок, проаналізовано ергономічні та нормативні вимоги на підприємствах, правила розміщення та догляд за приміщенням.

ВИСНОВОК

У роботі із розробки мобільного застосунку для фотонотатків було використано поступове проектування та сучасні рішення для реалізації даного проекту.

За мету було поставлено розробка функціонального додатку, що не мав би несумісного наповнення. Завдяки зручному інтерфейсу, додаток може надати користувачам можливість легко отримувати збережені фотографії, коригувати інформацію про них, пов'язувати дані з їх розташуванням.

Під час роботи було проаналізовано додатки-конкуренти, їх переваги та недоліки, що надало можливість спроектувати функціональну частину системи згідно вимог до проекту.

Також, було зазначено які саме рішення були найкращими для виготовлення проекту. Так, було проаналізовано різні реалізації із баз даних, інтерфейсу користувача, систем управління камерою, зворотного зв'язку.

В результаті, мобільний додаток було розроблено за допомогою таких технологій: Jetpack Compose - для зручного коду у декларативному стилі, Room Database - ефективне рішення, яке підтримується Google. Було використано гугл сервіси (GMS Location, Google Maps) для швидкого отримання актуальних даних.

Результатом програми є Android застосунок, який виконує поставлені цілі згідно проектування, має ефективну реалізацію, не потребує багато місця та є легким і зрозумілим у користуванні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google платформа для пошуку шрифтів. URL: <https://fonts.google.com/> (дата звернення 05.04.2025).
2. Використання GMS Location до проекту. URL: <https://developers.google.com/android/reference/com/google/android/gms/location/LocationServices> (дата звернення 07.03.2025).
3. Бібліотека Coil Android. URL: <https://github.com/coil-kt/coil> (дата звернення 17.03.2025).
4. Приклади використання Room Database. URL: <https://developer.android.com/jetpack/androidx/releases/room> (дата звернення 24.03.2025).
5. Приклади трансформацій мовою Kotlin. URL: <https://kotlinlang.org/docs/collection-transformations.html#map> (дата звернення 12.02.2025).
6. Приклади використання Google Maps. URL: <https://developers.google.com/maps/documentation/android-sdk/overview> (дата звернення 10.03.2025).
7. Інструктаж до UML. URL: <https://dou.ua/forums/topic/40575/> (дата звернення 02.01.2025).
8. Логічна модель даних. URL: <https://www.youtube.com/watch?v=SwCiEWBdaBI> (дата звернення 15.02.2025).
9. Реляційні бази даних. URL: <https://foxminded.ua/reliatsiini-bazy-danykh/> (дата звернення 23.03.2025).
10. SQL запити. URL: https://www.youtube.com/watch?v=b_NJb-2ZJEA (дата звернення 23.03.2025).
11. Використання Material3 для UI. URL: <https://m3.material.io/blog/material-theme-builder> (дата звернення 05.04.2025).
12. Переваги Kotlin. URL: <https://dou.ua/forums/topic/45180/> (дата звернення 14.02.2025).
13. Приклад використання камери у Android додатку. URL: <https://medium.com/deuk/android-camera-permission-essentials-streamlining-with-baseactivity-13be6d296224> (дата звернення 09.03.2025).
14. Приклад використання перенаправлення тексту до пошти з додатку Android. URL: <https://stackoverflow.com/questions/4345032/how-to-send-a>

[simple-email-programmatically-exists-a-simple-way-to-do-it](#) (дата звернення 12.04.2025).

15. Розробка додатків під Android. URL: <https://foxminded.ua/rozrobyty-dodatok-na-android/> (дата звернення 27.02.2025).
16. Android Manifest. URL: <https://www.geeksforgeeks.org/application-manifest-file-in-android/> (дата звернення 03.03.2025).

ДОДАТОК А – Технічне завдання

Вступ

Завданням дипломної роботи є розробка мобільного додатку для організації фотонотаток. В таблиці А.1 представлено загальні відомості про систему.

Таблиця А.1 - Загальні відомості про систему

Повна назва системи	Мобільний додаток для організації фотонотаток
Умовне позначення	Мобільний додаток
Розробник	Чугунов Кирило Андрійович
Замовник	Кафедра інформаційних управляючих систем та технологій

1. Підстави для розробки

Мобільний додаток розробляється на підставі завдання для дипломного проектування за спеціальністю 124 - “Системний аналіз”.

2. Призначення розробки

Об’єкт автоматизації: користувач мобільного телефону

Основні задачі користувача є:

- Отримання фото з локацією (без неї).
- Створення користувачів для розподілення даних.
- Сортування фотографій.
- Можливість додавання нових знімків.

3. Мета створення системи

Основною метою створення мобільного додатку є реалізація додатку із зручним інтерфейсом, ефективним механізмом збереження даних, функціями для легкого доступу до даних та можливістю їх модифікування.

4. Вимоги до програми

4.1. Вимоги до функціональних характеристик

Спроектований мобільний додаток має виконувати наступні функції:

- здійснювати авторизацію користувача (вхід, створення нового користувача)
- надавати можливість додати нові знімки
- відображати фото у формі списку
- відображати фото на карті
- обробляти надсилання відгуків
- можливість сортувати фото за допомогою папок
- додавання/видалення тегів до фото
- зміна даних/налаштувань користувача
- видалення користувачів

4.2. Вимоги до надійності

При проектуванні повинні бути використані типові проектні рішення там, де це можливо. На стадії технічного проектування повинні прийматися рішення, що зменшують трудомісткість експлуатації системи.

4.3. Умови експлуатації системи

Мобільний додаток розроблено на мові програмування Kotlin у середовищі Android Studio.

4.4. Вимоги до інформаційної та програмної сумісності

- Клієнт має використовувати мобільний телефон із 28-35 сдк та ОС Android.
- Для використання відображень фото на карті та отримання локації необхідно бути підключеним до мережі, всі інші функції можливо використовувати без Інтернет-з'єднання.

- Оперативна пам'ять: 2 гб
- Пам'ять пристрою: 7мб
- Наявність камери

4.5 Вимоги до технічного забезпечення

Технічні вимоги до мобільних пристроїв визначаються операційною системою Android.

5. Вимоги до програмної документації

Усі регламентуючі документи є елементами мобільного додатку, які безпосередньо входять до цього. Регламент системи описує правила її ефективного використання. В таблиці А.2 наведено склад документів.

Таблиця А.2 - Склад документів організаційного забезпечення

№	Назва	Стадія
1	Опис організаційної структури	Технічне проектування
2	Керівництво користувача	Реалізація системи

6. Стадії та етапи розробки

В таблиці А.3 представлено стадії та етапи по створенню мобільного додатку

Таблиця А.3 - Стадії та етапи робіт по створення додатку

№	Стадія	Документи
1	Аналіз предметної області	Звіт з описом предметної області
2	Системне проектування	Технічне завдання
3	Ескізне проектування	Ескізний проект
4	Технічне проектування	Технічний проект
5	Робоче проектування	Робочий проект

ДОДАТОК Б – Програмний код

Повний код проекту: <https://github.com/Kirishhaa/PhotoNotes>

CameraEvent.kt

```
package com.kirishhaa.photonotes.presentation.home.camerascreen
sealed interface CameraEvent {
    class OnNewImageCaptured(val markerId: Int) : CameraEvent
}
```

CameraScreen.kt

```
package com.kirishhaa.photonotes.presentation.home.camerascreen
import android.annotation.SuppressLint
import android.content.pm.PackageManager
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.offset
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.DisposableEffect
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import androidx.lifecycle.Lifecycle
import androidx.lifecycle.LifecycleEventObserver
import androidx.lifecycle.compose.LocalLifecycleOwner
```



```

import androidx.lifecycle.viewmodel.compose.viewModel
import com.kirishhaa.photonotes.R
import com.kirishhaa.photonotes.clickeffects.pulsateClick
@SuppressLint("UseCheckPermission")
@Composable
fun CameraScreen(
    onNewImageCaptured: (Int) -> Unit
) {
    val viewModel: CameraViewModel = viewModel(factory =
CameraViewModel.Factory)
    val state by viewModel.state.collectAsState()
    LaunchedEffect(0) {
        viewModel.events.collect { event ->
            when (val valueEvent = event.getValue()) {
                is CameraEvent.OnNewImageCaptured ->
onNewImageCaptured(valueEvent.markerId)
                null -> {}
            }
        }
    }
    when {
        state.loadingState -> LoadingScreen()
        else -> {
            CameraScreen(
                state = state,
                onLaunchCameraPermission = {
viewModel.decreaseRequestPermissionCount(android.Manifest.permission.CAMERA) },
                onLaunchLocationPermission = {
viewModel.decreaseRequestPermissionCount(android.Manifest.permission.ACCESS_COARSE_LOCATION) },
                onImageCaptured = viewModel::onImageCaptured
            )
        }
    }
}
@Composable
private fun LoadingScreen() {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center
    ) { CircularProgressIndicator(
        color = colorResource(R.color.primary_color)
    ) }
}
@Composable
private fun CameraScreen(
    state: CameraState,
    onLaunchCameraPermission: () -> Unit,
    onLaunchLocationPermission: () -> Unit,
    onImageCaptured: (String) -> Unit

```

```

) {
    val context = LocalContext.current
    var cameraGranted by remember {
        mutableStateOf(
            ContextCompat.checkSelfPermission(
                context,
                android.Manifest.permission.CAMERA
            ) == PackageManager.PERMISSION_GRANTED
        )
    }
    var locationGranted by remember {
        mutableStateOf(
            ContextCompat.checkSelfPermission(
                context,
                android.Manifest.permission.ACCESS_COARSE_LOCATION
            ) == PackageManager.PERMISSION_GRANTED
        )
    }
    val locationPermissionLauncher =
rememberLauncherForActivityResult(
    contract = ActivityResultContracts.RequestPermission()
) { granted ->
    locationGranted = granted
}
    val cameraPermissionLauncher =
rememberLauncherForActivityResult(
    contract = ActivityResultContracts.RequestPermission()
) { granted ->
    onLaunchLocationPermission()

locationPermissionLauncher.launch(android.Manifest.permission.ACCE
SS_COARSE_LOCATION)
    cameraGranted = granted
}
    var imagePath: String? by remember {
        mutableStateOf(null)
    }
    val cameraLauncher = rememberLauncherForActivityResult(
    contract = ActivityResultContracts.TakePicture()
) { success ->
    if (success) {
        imagePath?.let { path -> onImageCaptured(path) }
    }
}
    val lo = LocalLifecycleOwner.current
    var resumed by remember {
        mutableStateOf(false)
    }
    DisposableEffect(lo) {
        val observer = LifecycleEventObserver { source, event ->

```

```

        resumed =
source.lifecycle.currentState.isAtLeast(Lifecycle.State.RESUMED)
    }
    lo.lifecycle.addObserver(observer)
    onDispose {
        lo.lifecycle.removeObserver(observer)
    }
}
Box(
    contentAlignment = Alignment.Center,
    modifier = Modifier.fillMaxSize()
) {
    when {
        state.createNewMarker || resumed.not() -> {
            CircularProgressIndicator(color =
colorResource(R.color.primary_color))
        }
        state.requestCameraPermissionCount == 0 &&
cameraGranted.not() -> {
            Text(
                text =
stringResource(R.string.to_take_a_photo_you_need_a_permission),
                fontSize = 24.sp,
                color = colorResource(R.color.on_surface),
                fontFamily = FontFamily(Font(R.font.comic)),
                textAlign = TextAlign.Center,
                modifier = Modifier.padding(horizontal = 16.dp)
            )
        }
        resumed -> {
            Box(
                contentAlignment = Alignment.Center,
                modifier = Modifier
                    .width(220.dp)
                    .height(50.dp)
                    .offset(y = -5.dp)
                    .pulseClick(clickable = true, onClick = {
                        if (cameraGranted) {
                            val uri =
ComposeFileProvider.getImageUri(context)
                            imagePath = uri.toString()
                            cameraLauncher.launch(uri)
                        } else {
                            onLaunchCameraPermission()

cameraPermissionLauncher.launch(android.Manifest.permission.CAMERA
)
                        }
                    })
                .background(

```

```

                                color =
colorResource(R.color.on_secondary_container),
                                shape = RoundedCornerShape(24.dp)
                                )
                                ) {
                                    Text(
                                        text =
stringResource(R.string.take_a_photo),
                                        fontFamily =
FontFamily(Font(R.font.comic)),
                                        fontSize = 24.sp,
                                        color = Color.White
                                    )
                                }
                            }
                    }
            }
    }
}

```

CameraState.kt

```

package com.kirishhaa.photonotes.presentation.home.camerascreen
data class CameraState(
    val loadingState: Boolean = true,
    val creatingNewMarker: Boolean = false,
    val requestCameraPermissionCount: Int = 0,
    val requestLocationPermissionCount: Int = 0
)

```

CameraViewModel.kt

```

package com.kirishhaa.photonotes.presentation.home.camerascreen
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.viewModelScope
import androidx.lifecycle.viewmodel.CreationExtras
import com.kirishhaa.photonotes.SingleEvent
import com.kirishhaa.photonotes.domain.markers.CreateMarkerUseCase
import com.kirishhaa.photonotes.domain.permissions.DecreasePermissionCountUseCase
import com.kirishhaa.photonotes.domain.permissions.GetPermissionCountUseCase
import com.kirishhaa.photonotes.toApp
import kotlinx.coroutines.CoroutineExceptionHandler
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow

```

```

import kotlinx.coroutines.launch
class CameraViewModel(
    private val createMarkerUseCase: CreateMarkerUseCase,
    private val decreasePermissionCountUseCase:
DecreasePermissionCountUseCase,
    private val getPermissionCountUseCase:
GetPermissionCountUseCase
) : ViewModel() {
    private val _events =
MutableSharedFlow<SingleEvent<CameraEvent>>(replay = 1)
    val events: Flow<SingleEvent<CameraEvent>> = _events
    private val _state = MutableStateFlow(CameraState())
    val state = _state.asStateFlow()
    private val exceptionHandler = CoroutineExceptionHandler { _,
throwable ->
        _state.value = CameraState(false)
    }
    init {
        viewModelScope.launch {
            val cameraRequestPermissionCount =

getPermissionCountUseCase.execute(android.Manifest.permission.CAME
RA)

            val locationRequestPermissionCount =

getPermissionCountUseCase.execute(android.Manifest.permission.ACCE
SS_COARSE_LOCATION)
            _state.value = CameraState(
                loadingState = false,
                creatingNewMarker = false,
                requestCameraPermissionCount =
cameraRequestPermissionCount,
                requestLocationPermissionCount =
locationRequestPermissionCount
            )
        }
    }
    fun decreaseRequestPermissionCount(name: String) {
        viewModelScope.launch {
            decreasePermissionCountUseCase.execute(name)
            val count = getPermissionCountUseCase.execute(name)
            if (name == android.Manifest.permission.CAMERA) {
                _state.value = _state.value.copy(
                    requestCameraPermissionCount = count
                )
            } else if (name ==
android.Manifest.permission.ACCESS_COARSE_LOCATION) {
                _state.value = _state.value.copy(
                    requestLocationPermissionCount = count
                )
            }
        }
    }
}

```

```

    }
}
fun onImageCaptured(filePath: String) {
    viewModelScope.launch(exceptionHandler) {
        _state.value = CameraState(true)
        val markerId = createMarkerUseCase.execute(filePath)

_events.emit(SingleEvent(CameraEvent.OnNewImageCaptured(markerId))
)
        _state.value = CameraState(false)
    }
}
companion object {
    val Factory = object : ViewModelProvider.Factory {
        override fun <T : ViewModel> create(modelClass:
Class<T>, extras: CreationExtras): T {
            val app = extras.toApp()
            val repoUsers = app.localUsersRepository
            val permsRepo = app.permissionsRepository
            val repoImages = app.markersRepository
            val usecase = CreateMarkerUseCase(repoUsers,
repoImages)
            val decreaseUseCase =
DecreasePermissionCountUseCase(permsRepo)
            val getNumUseCase =
GetPermissionCountUseCase(permsRepo)
            return CameraViewModel(usecase, decreaseUseCase,
getNumUseCase) as T
        }
    }
}
}

```

ComposeFileProvider.kt

```

package com.kirishhaa.photonotes.presentation.home.camerascreen
import android.content.Context
import android.net.Uri
import androidx.core.content.FileProvider
import com.kirishhaa.photonotes.R
import java.io.File
class ComposeFileProvider : FileProvider(R.xml.paths) {
    companion object {
        fun getImageUri(context: Context): Uri {
            val directory = File(context.filesDir, "images")
            directory.mkdirs()
            val file = File.createTempFile(
                "image_" + System.currentTimeMillis().toString(),
                ".jpg",
                directory
            )
        }
    }
}

```

```
        val authority = context.packageName + ".fileprovider"  
        return getUriForFile(context, authority, file)  
    }  
}
```

ДОДАТОК В – Інструкція користувача

Спочатку користувач опиняється на екрані авторизації, без жодного облікового запису (рисунок В.1).

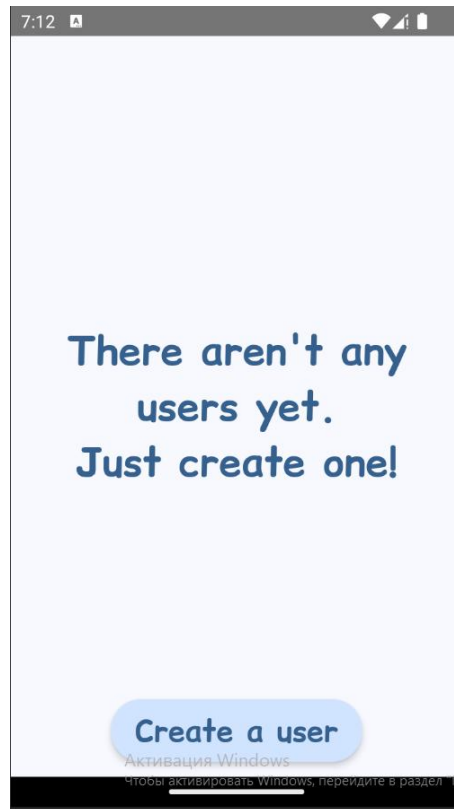


Рисунок В.1 - початковий стан екрану авторизації

Після натискання на кнопку “Create a user”, користувачу показано діалог реєстрації нового користувача (рисунок В.2).

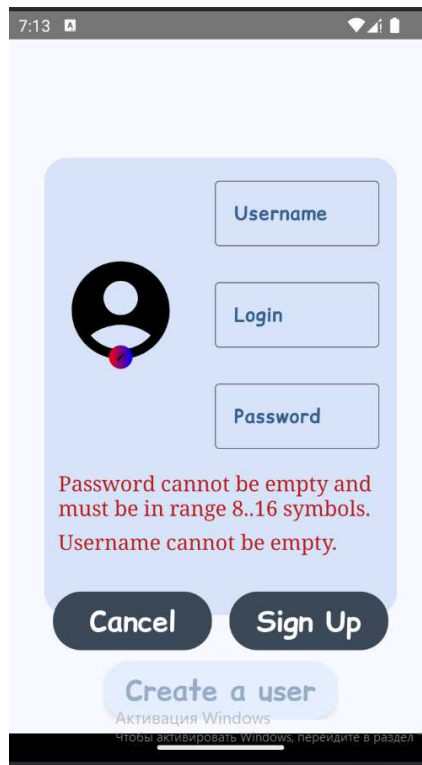


Рисунок В.2 - діалогове вікно створення нового облікового запису

Після успішного створення облікового запису, його буде відображено на екрані авторизації (рисунок В.3).

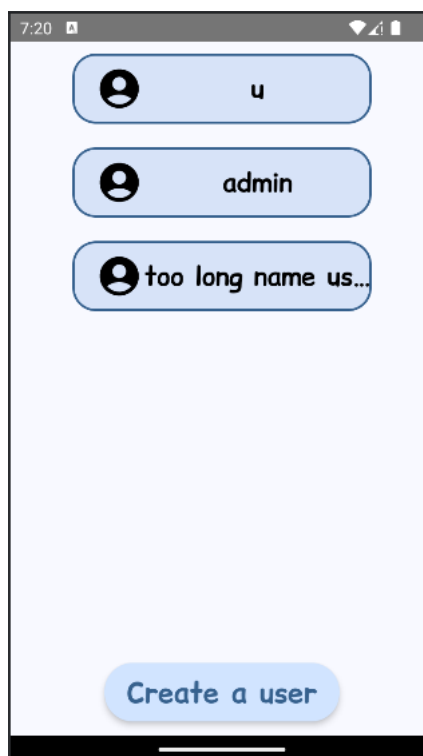


Рисунок В.3 - екран авторизації із трьома створеними обліковими записами

При натисканні на зареєстрований обліковий запис користувачу буде показано діалог входу до облікового запису (рисунок В.4).

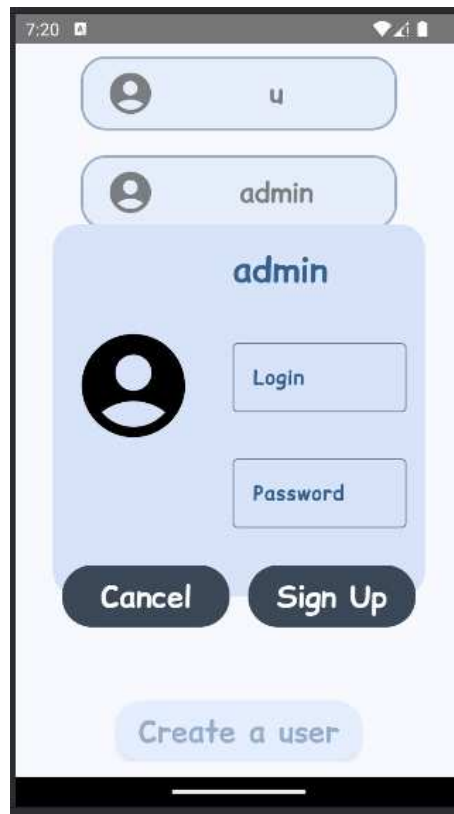


Рисунок В.4 - діалогове вікно входу до облікового запису

Після того, як користувач успішно здійснив вхід до облікового запису, він переходить до екрану головного меню. Спочатку відкрито розділ “Папки” (рисунок В.5).

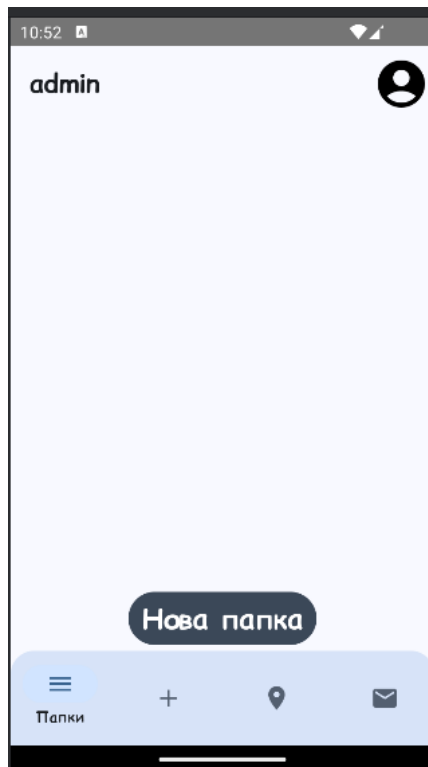


Рисунок В.5 - Головне меню. Початковий стан розділу “Папки”

Після того, як користувач натиснув на додавання нової папки, йому буде показано діалог створення нової папки (рисунок В.6).



Рисунок В.6 - Діалог додавання нової папки

Після того як папку було успішно додано, це буде відображено у розділі “Папки” (рисунок В.7).

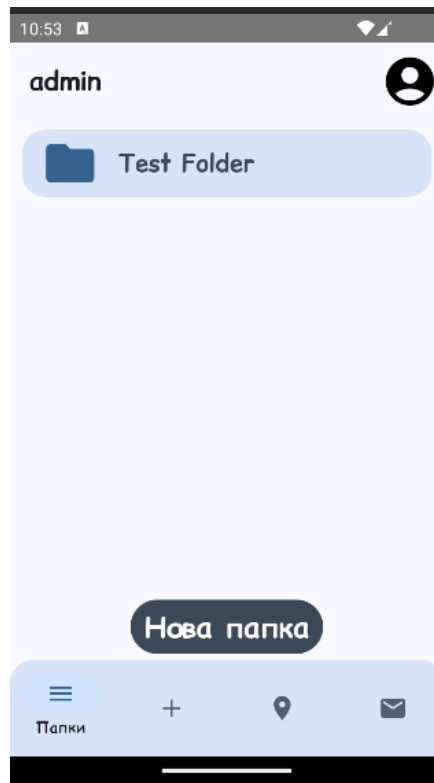


Рисунок В.7 - Головне меню. Розділ “Папки” із однією створеною папкою

Якщо користувач перейде до розділу камери, він побачить кнопку “Зробити знімок” (рисунок В.8).

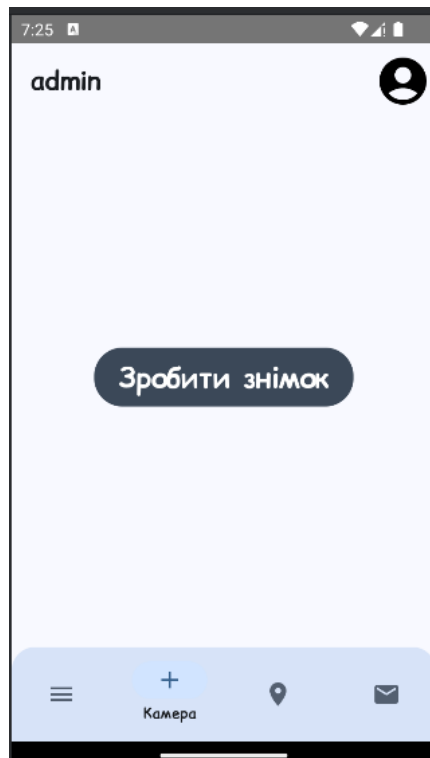


Рисунок В.8 - Головне меню. Розділ “Камера”

Після першого натискання буде відображено запит на дозвіл роботи із камерою (рисунок В.9).

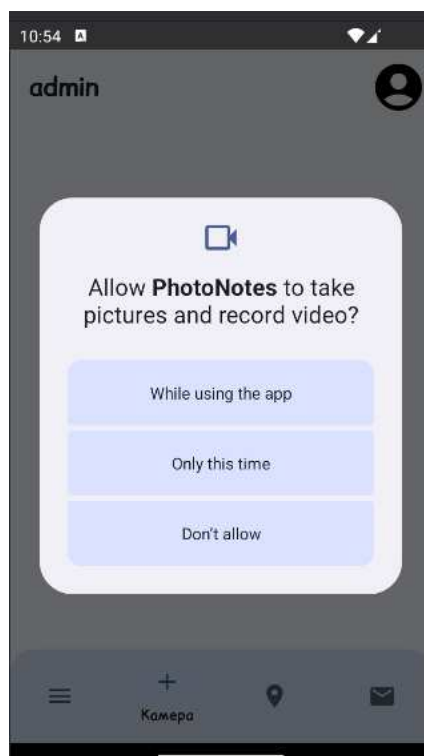


Рисунок В.9 - Запит на дозвіл роботи із камерою

Після чого, буде також запит на дозвіл роботи із локацією.

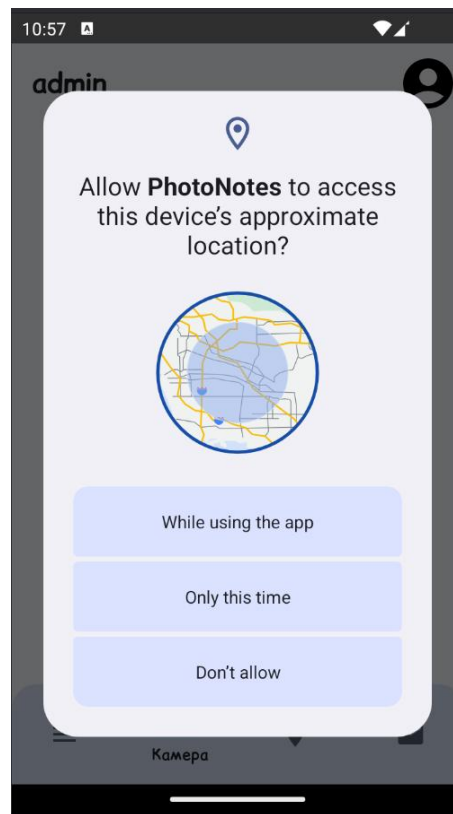


Рисунок В.10 - Запит на дозвіл роботи із локацією

Після цього, якщо дозвіл на камеру було надано, ми можемо скористатися кнопкою для роботи із камерою (рисунок В.11). Оскільки використовується емулятор для зображення результатів, знімок закрито. Якщо дозвіл на використання камери не було надано, буде показано відповідне повідомлення (рисунок В.12).

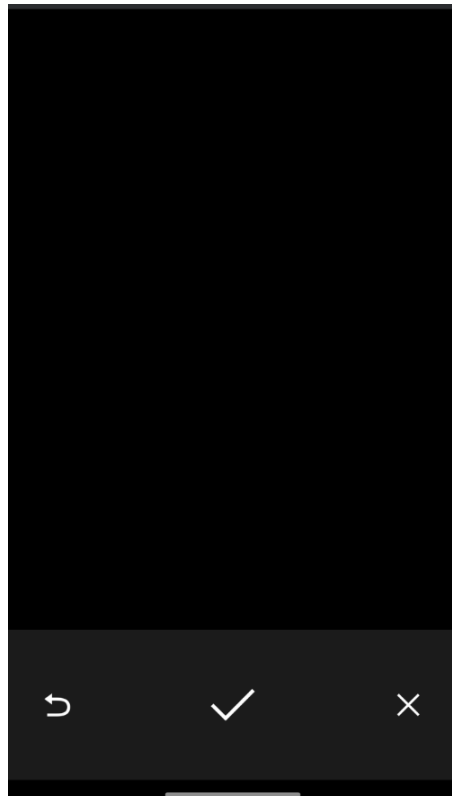


Рисунок В.11 - Екран роботи камери

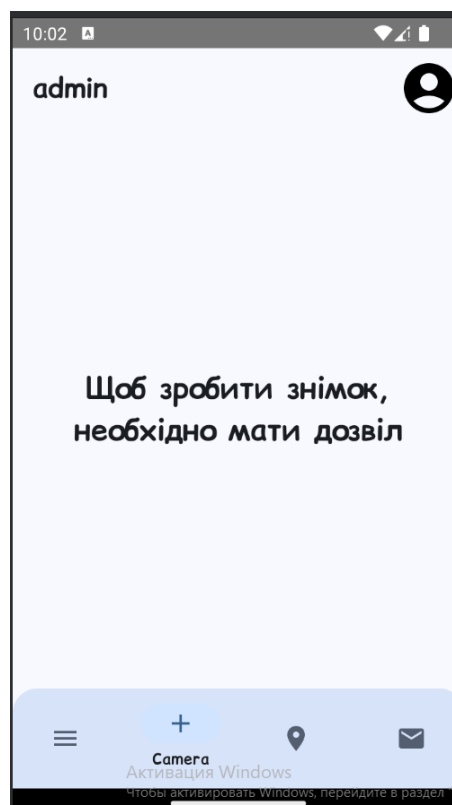


Рисунок В.12 - Сповіщення про відсутність дозволу використання камери

Якщо фото було зроблене успішно, користувач переходить до екрану деталей фото (рисунок В.13). Країна та місто також не відображаються через симуляцію. У звичайних пристроях має писати вашу та місто, якщо було надано дозвіл на роботу з локацією (рисунок В.14).

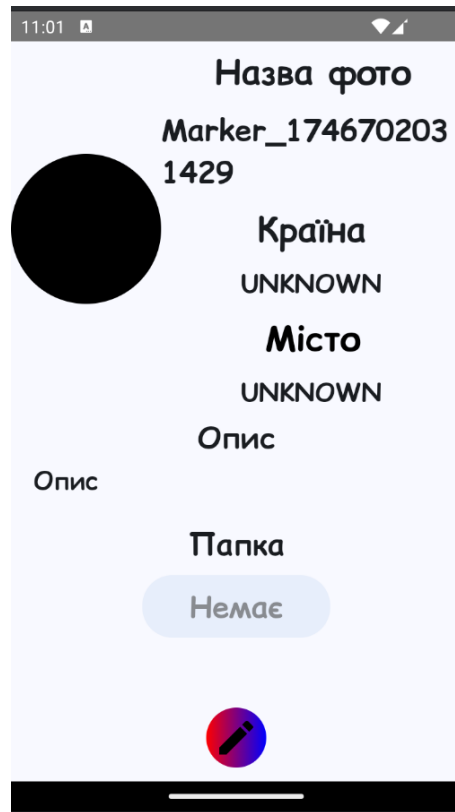


Рисунок В.13 - Екран деталей фото. Відсутня локація



Рисунок В.14 - Экран деталей фото. Прив'язана локація

Якщо користувач натисне на кнопку редагування знизу, екран буде змінено до режиму редагування (рисунок В.15).

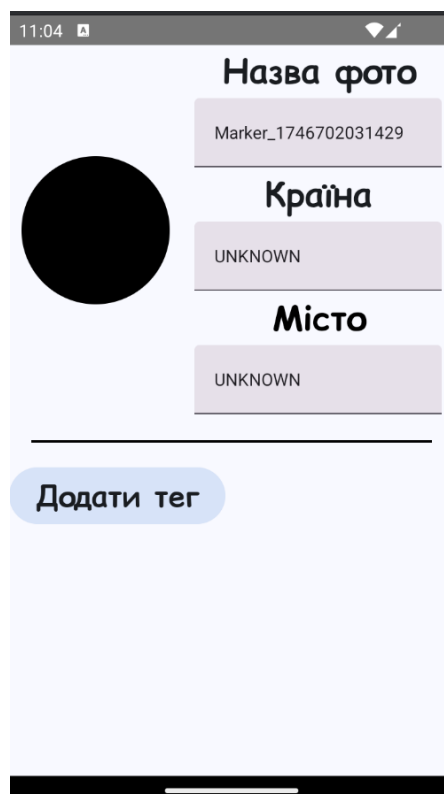


Рисунок В.15 - Экран деталей фото. Режим редагування

Якщо користувач натисне “Додати тег”, йому буде показано діалог додавання тегу (рисунок В.16).

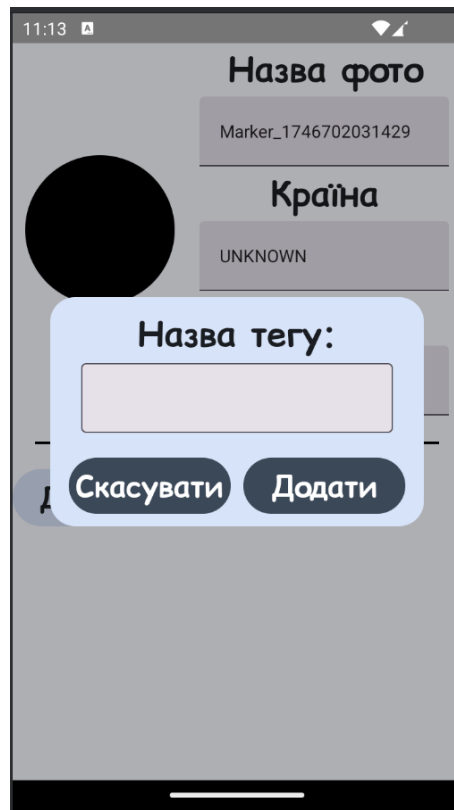


Рисунок В.16 - Діалог додавання нового тегу

Якщо тег було успішно додано, його буде відображено (рисунок В.17).



Рисунок В.17 - Екран деталей фото. Додано 3 теги

Друга частина екрану деталей фото (рисунок В.18). Ми можемо тут обрати папку(якщо папка існує), натиснувши “Немає” (рисунок В.19, рисунок В.20). Можемо зберегти зміни або видалити фотографію.



Рисунок В.18 - Экран деталей фото. Друга частина



Рисунок В.19 - Экран деталей фото. Вибір папки

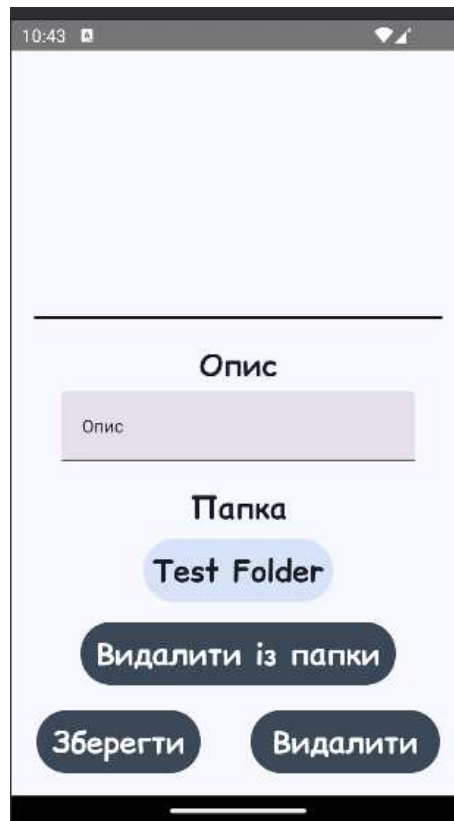


Рисунок В.20 - Екран деталей фото із обраною папкою “Test Folder”

Після додавання фото до папки “Test Folder” розділ “Папки” буде змінено відповідно до ієрархії (рисунок В.21, рисунок В.22).

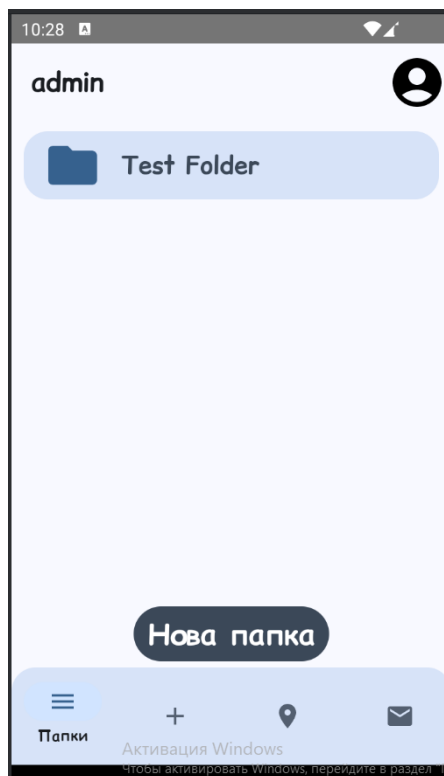


Рисунок В.21 - Головне меню. Розділ “Папки”

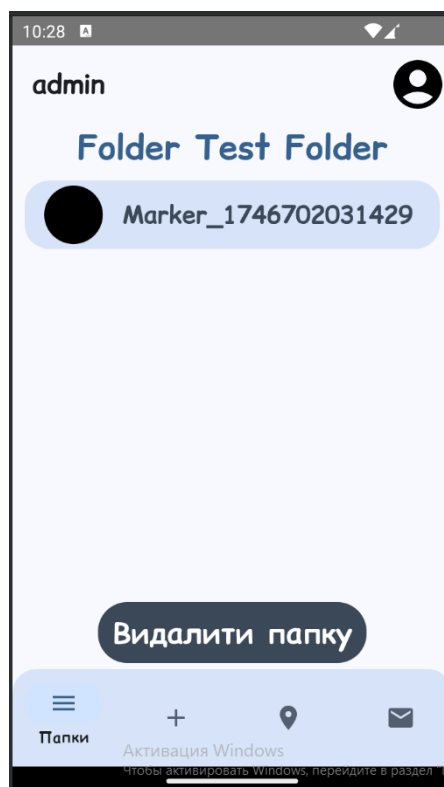


Рисунок В.22 - Головне меню. Розділ “Папки”. Результат переходу до папки
“Test Folder”

Якщо фото не було додано до жодної папки, його можна буде побачити у розділу “Папки” (рисунок В.23).

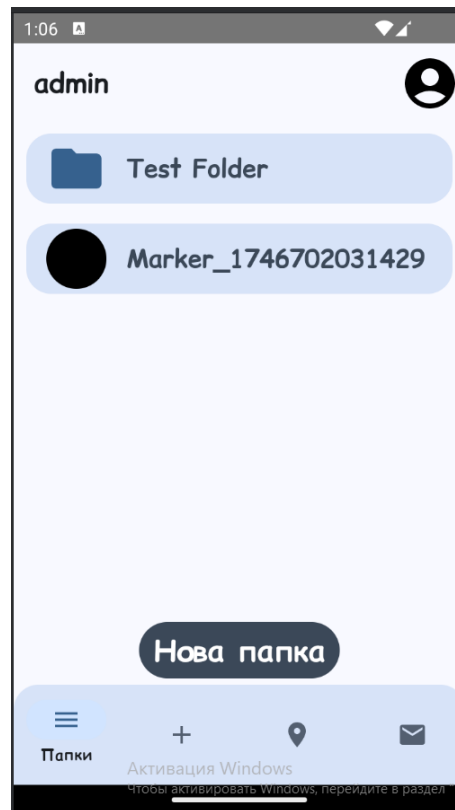


Рисунок В.23 - Головне меню. Розділ “Папки”

Якщо користувач перейде до розділу “Карта”, він побаче карту, на якій будуть розташовані фото, до яких вдалося підчепити локацію.



Рисунок В.24 - Головне меню. Розділ “Карта”. Варіант без фото

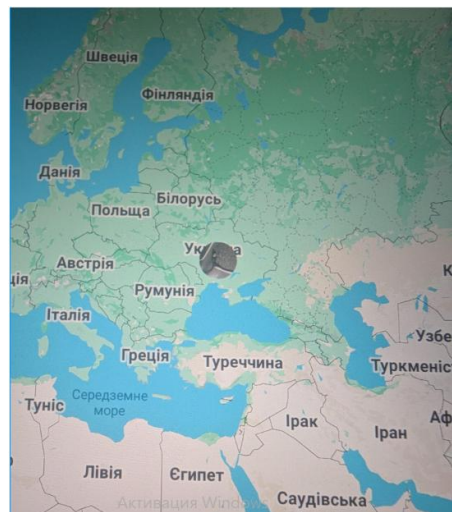


Рисунок В.25 - Головне меню. Розділ “Карта”. Варіант фото із локаціями

Якщо користувач перейде до розділу “Відгук” він буде мати 3 поля (ім'я, запитання та спосіб зворотного зв'язку) та кнопку для відправлення даних (рисунок В.26, рисунок В.27).

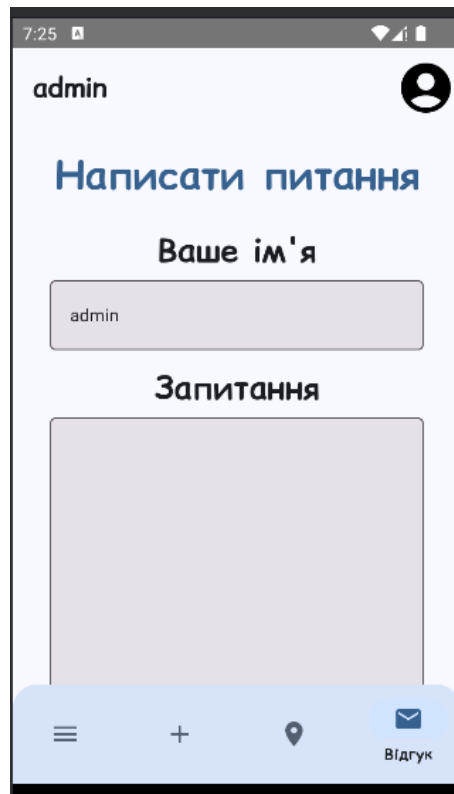


Рисунок В.26 - Головне меню. Розділ “Відгук”. Частина перша

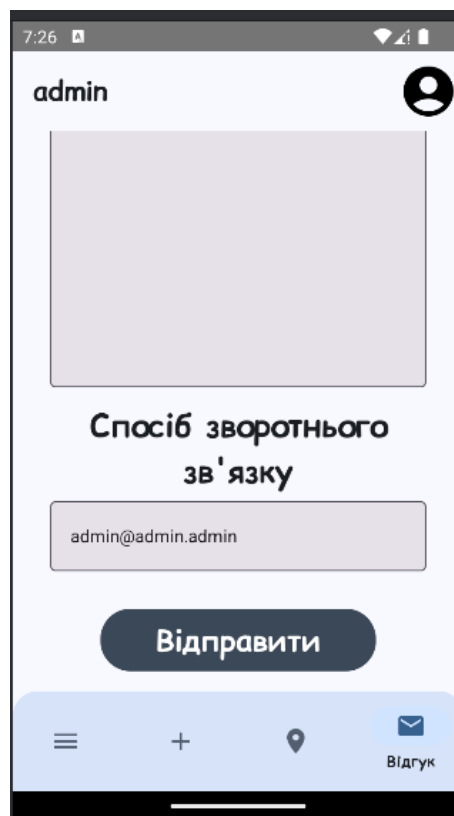


Рисунок В.27 - Головне меню. Розділ “Відгук”. Частина друга

Результат кнопки “Відправити” наведено на рисунку В.28.

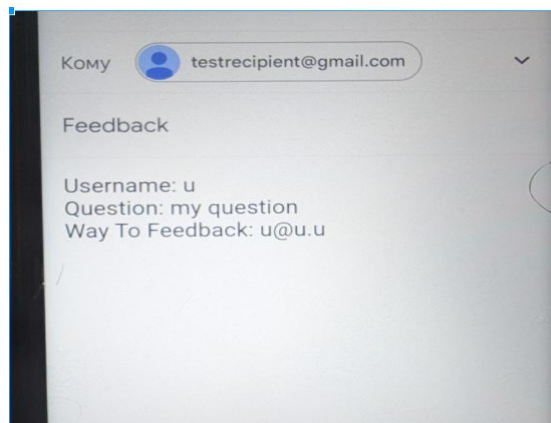


Рисунок В.28 - Результат відправлення відгуку

Якщо користувач натисне на іконку облікового запису, то він здійснить перехід до екрану профілю (рисунок В.29).

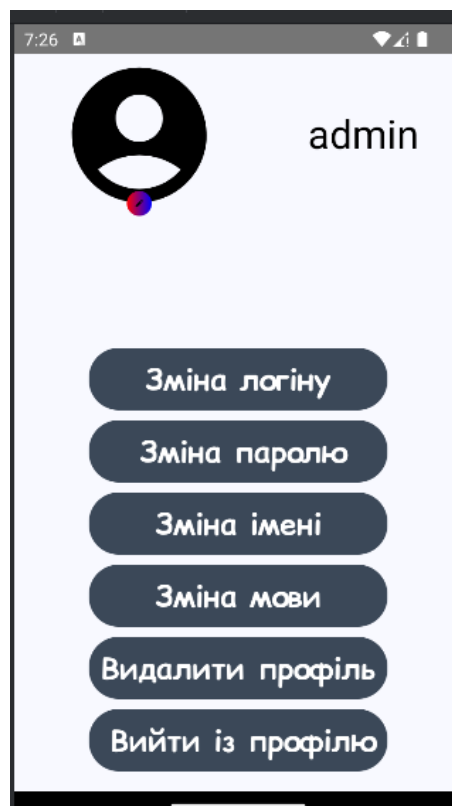


Рисунок В.29 - Екран профілю користувача

Якщо користувач натисне на зміну логіну, він перейде до екрану зміни логіну (рисунок В.30).

7:26

Змінити ваш логін

Поточний логін

admin@admin.admin

Новий логін

Повторення нового логіну

Змінити

Рисунок В.30 - Екран зміни логіну

Якщо користувач натисне на зміну паролю, то він перейде до екрану зміни паролю (рисунок В.31).

7:27

Змінити ваш пароль

Поточний пароль

Новий пароль

Повторення нового паролю

Змінити

Рисунок В.31 - Екран зміни паролю

Якщо користувач натисне на зміну імені, то він перейде до екрану зміни імені (рисунок В.32).



Рисунок В.32 - Екран зміни ім'я

Якщо користувач натисне на зміну мови, то він перейде на екран зміни мови (рисунок В.33).

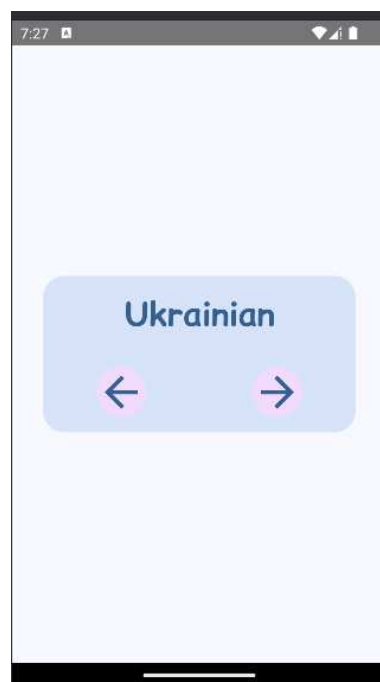


Рисунок В.33 - Екран зміни мови