МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Мобільний додаток для персонального читання новин»

Здобувача групи ІТ-03 Іванова Олега Володимировича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олег ІВАНОВ
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Володимир НАГОРНИЙ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2024 р.

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

Іванову Олегу Володимировичу

1 Тема роботи *Мобільний додаток для персонального читання новин*

керівник роботи *Нагорний Володимир В'ячеславович, к.т.н., доцент*,

затверджені наказом по університету від «*07*» травня 2024 р. №0482-VI

2 Строк подання студентом роботи «26» травня 2024 р.

3 Вхідні дані до роботи технічне завдання на розробку мобільного додатку для персонального читання новин

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та проектування, практична реалізація мобільного додатку, тестування

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) аналіз проблеми читання новин, визначення актуальності розробки мобільного додатку для персонального читання новин, аналіз додатків аналогів, визначення функціональних вимог, контекстна діаграма IDEF0, діаграма декомпозиції першого рівня, діаграма варіантів використання, логічна модель даних, архітектура мобільного додатку, програмна реалізація, презентація роботи мобільного додатку, тестування додатку

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 01.04.2024

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	04.04.2024 – 08.04.2024	Виконано
2	Оформлення планування робіт	08.04.2024 – 10.04.2024	Виконано
3	Створення технічного завдання	10.04.2024 – 16.04.2024	Виконано
4	Вибір засобів реалізації	16.04.2024 – 20.04.2024	Виконано
5	Проектування мобільного додатку	20.04.2024 – 28.04.2024	Виконано
6	Розробка мобільного додатку	28.04.2024 – 16.05.2024	Виконано
7	Тестування мобільного додатку	16.05.2024 – 17.05.2024	Виконано
8	Оформлення пояснювальної записки	17.05.2024 – 20.05.2024	Виконано

Студент _____ Олег ІВАНОВ
(підпис)

Керівник роботи _____ к.т.н., доц. Володимир НАГОРНИЙ
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Мобільний додаток для персонального читання новин».

Пояснювальна записка складається з вступу, трьох розділів, висновку, списку використаних джерел із 24 найменувань та трьох додатків. Загальний обсяг робіт становить 88 сторінок, що включає 34 сторінок основного тексту в якому знаходиться 16 ілюстрацій та 2 таблиці, 3 сторінки списку використаних джерел та 50 сторінок додатків.

Актуальність розробки полягає у створенні зручного та персоналізованого мобільного додатку для читання новин, який відповідає індивідуальним потребам користувачів та покращує їхній досвід споживання інформації.

Метою роботи є розробка мобільного додатку для читання персональних новин, який забезпечує інтуїтивно зрозумілий інтерфейс, швидкий доступ до актуальної інформації та можливість налаштування новинного потоку відповідно до інтересів користувача.

У першому розділі проведено аналіз предметної області, визначивши актуальність цієї проблеми та розглянувши додатки аналоги. На основі результатів аналізу були визначені мета та постановка завдання.

Другий розділ присвячений структурно-функціональному моделюванню. Було розроблено контекстну діаграму в нотації IDEF0 та її декомпозицію, а також діаграму варіантів використання для акторів, процесів та їх взаємозв'язків. Також було створено структуру бази даних та описано архітектуру мобільного додатку.

У третьому розділі детально описано програмну реалізацію додатку та варіанти його використання для різних видів користувачів. Крім того, були проведені юніт-тести, результати яких представлені у вигляді таблиці.

Ключові слова: мобільний додаток, розробка, моделювання, персоналізоване читання новин, Android Studio, Kotlin, Firebase.

ЗМІСТ

ВСТУП

За останні роки світ пережив значний ріст у сфері цифрових технологій, про що свідчить стрімке зростання продажів смартфонів та активні інвестиції у розвиток мобільних технологій. Цей розквіт відкриває нові перспективи для розробників програмного забезпечення, зокрема в галузі мобільних додатків [1].

У такому світі, насиченому інформацією та новинами, отримання персональних рекомендацій набуває великого значення. Використання технологій штучного інтелекту та алгоритмів машинного навчання відкриває нові можливості для створення інноваційних продуктів, що адаптуються до індивідуальних інформаційних потреб користувачів [2].

Мобільний додаток для персоналізованого читання новин є важливим інструментом для зручного отримання актуальної інформації. Метою проекту є створення додатку, який наддасть користувачам можливість персоналізувати свій досвід читання новин, відповідно до їхніх індивідуальних інтересів та вимог.

В ході виконання кваліфікаційної роботи бакалавра необхідно виконати наступні задачі:

- провести огляд останніх досліджень і публікацій з теми персоналізованого читання новин;
- проаналізувати мобільні додатки-аналоги;
- визначити функціональні та нефункціональні вимоги;
- виконати моделювання та проектування мобільного додатку;
- розробити мобільний додаток;
- провести тестування мобільного додатку

Практична цінність проекту полягає у покращенні читацького досвіду новин.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Останні дослідження та публікації в галузі розробки мобільних додатків для персонального читання новин свідчать про активний інтерес користувачів до інноваційних рішень у цій області.

Опитування USAID-Internews щодо споживання медіа, понад 82% користувачів мобільних пристроїв в Україні отримують новини через мобільні додатки [3]. Це свідчить про значний вплив мобільних платформ на спосіб споживання новин і потреби користувачів.

Однією з ключових тенденцій є зростання популярності персоналізованих медіа-платформ. Згідно з дослідженням, як змінилося медіа споживання українців в умовах повномасштабної війни, попит на індивідуалізовані стрічки новин постійно зростає [4]. Це свідчить про необхідність розвитку мобільних додатків, які надають користувачам можливість отримати доступ до актуальної інформації, відповідної їхнім інтересам.

В останніх дослідженнях про роль алгоритмів машинного навчання в системах рекомендацій додатків для Android зазначено, що однією з ключових особливостей розробки є створення персоналізованих рекомендацій щодо новин за допомогою штучного інтелекту. Машинне навчання надає можливість аналізувати величезні обсяги даних користувачів і надавати точніші та більш актуальні рекомендації. Це дозволяє створювати більш персоналізований та цікавий досвід для користувачів, що в свою чергу може збільшити залучення користувачів, завантаження додатків і покращити загальне задоволення користувачів [5].

Огляд останніх досліджень і публікацій свідчить про важливість розвитку мобільних додатків для читання новин та необхідність впровадження інноваційних підходів у цій області.

1.2 Аналіз програмних продуктів – аналогів

В сучасному світі, де інформаційний потік безперервно зростає, мобільні додатки для отримання новин стають все більш популярними. Серед них виділяються такі гіганти, як «Google News», «BBC News» та «CNN», кожен із яких пропонує свій унікальний підхід до представлення та організації інформації.

У даному розділі буде проведемо детальний аналіз кожного з цих додатків, враховуючи функціональні можливості. Це дозволить з'ясувати їхні переваги та недоліки, а також визначити, які саме аспекти можна врахувати при розробці власного продукту.

1.2.1. «Google News»

Додаток «Google News» [6] - це мобільний додаток, розроблений компанією Google для швидкого доступу до актуальних новин. Завдяки алгоритмам штучного інтелекту платформа надає персоналізований вміст на основі інтересів (рис. 1.1).

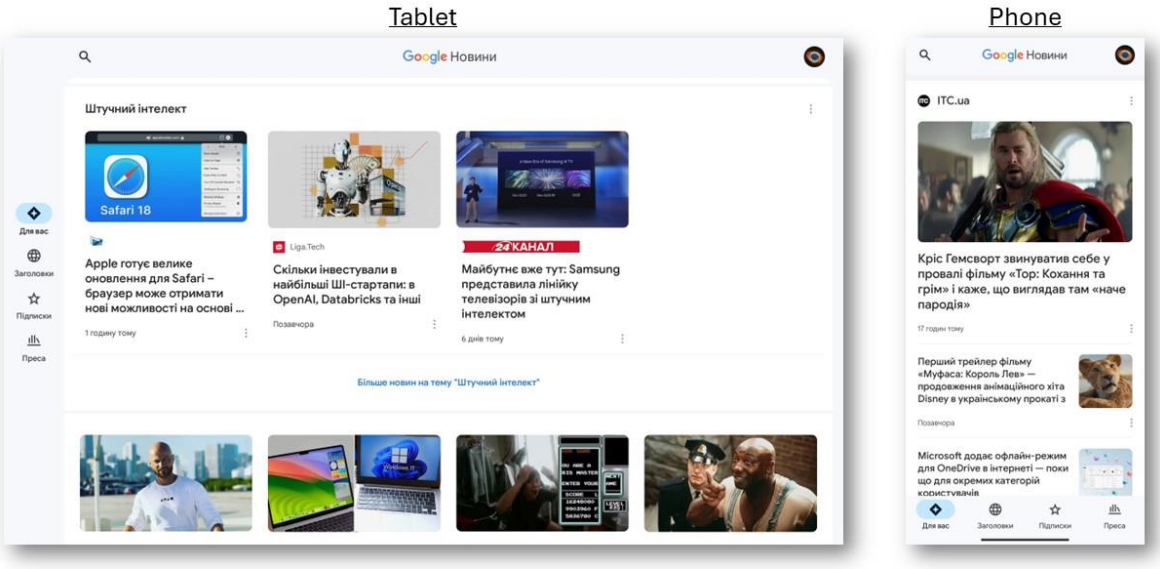


Рисунок 1.1 – Головна сторінка додатку «Google News»

До функціональних можливостей додатку входять пошук новин за ключовими словами, розбиття новин за категоріями, зручна навігація по додатку, можливість зареєструватись у додатку, можливість адаптуватись під різні види пристроїв та адаптивна система рекомендацій на основі алгоритмам штучного інтелекту.

До основного недоліку додатку можна віднести відсутність перегляду статистики використання додатка, тобто перегляд статистики прочитаних новин.

1.2.2. «BBC News»

Додаток «BBC News» [7] - це додаток від Британської корпорації BBC, який надає доступ до останніх новин з усього світу (рис. 1.2).

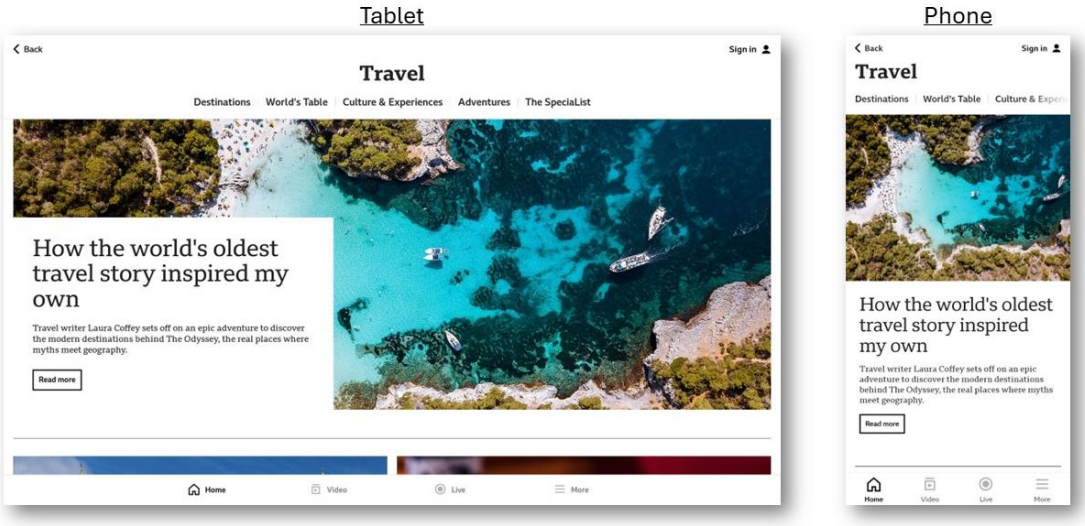


Рисунок 1.2 – Головна сторінка додатку «BBC News»

До функціональних можливостей додатку входять розбиття новин на категорії, доступ до різного виду контенту статі, фото, відео та прямі трансляції, можливість підписуватись на теми, зберігати цікаві статті в закладках, реєстрація користувачів.

До недоліків додатку можна віднести відсутня можливість пошуку статей за ключовими словами, відсутній алгоритм рекомендацій на основі користувацького досвіду та відсутній переглянути статистику прочитаних новин.

1.2.3. «CNN»

Додаток «CNN» [8] - це мобільний додаток, який надає доступ до широкого спектру новинних матеріалів від телеканалу CNN (рис. 1.3).

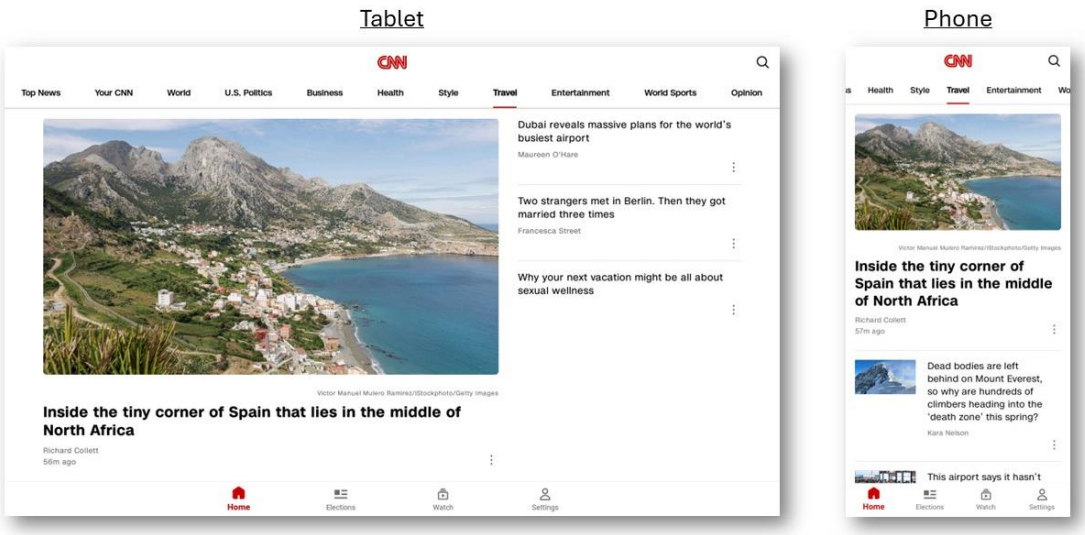


Рисунок 1.3 – Головна сторынка додатку «CNN»

До основні особливості цього додатку входять актуальні новини, відео та аудіо матеріали, персоналізовані новини, повідомлення про надзвичайних подій, пошук новин за ключовими словами, зручна навігація та адаптивний дизайн.

До недоліків можна віднести те, що не реалізована функція рекомендацій на основі користувацького досвіду та перегляду статистики прочитаних новин.

Після детального аналізу аналогів додатків з новинами, було визначено їх переваги та недоліки. Результати аналізу представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристик аналогів

Назва критерію	Google News	BBC News	CNN	Власна розробка
Розбиття новин за категоріями	+	+	+	+
Пошук новини за ключовими словами	+	-	+	+

Продовження таблиці 1.1

Назва критерію	Google News	BBC News	CNN	Власна розробка
Перегляд статистики прочитаних новин	-	-	-	+
Зручна навігація по додатку	+	+	+	+
Реєстрація користувачів	+	+	+	+
Адаптивність під різні види пристроїв	+	+	+	+
Наявність алгоритму рекомендацій новин	+	-	-	+

На основі порівняльної характеристики аналогів було визначено наступні функціональні вимоги:

- Пошук новин за ключовими словами;
- Перегляд новин розсортованих за категоріями;
- Перегляд статистики з прочитаних новин;
- Реєстрація користувачів;
- Отримувати персональні рекомендації на основі користувацького досвіду.

Більш детальний опис функціональних вимог зазначено в технічному завданні в додатку А.

1.3 Мета та задачі дослідження

Метою даного проекту є розробка мобільного додатка для персоналізованого читання новин.

У ході роботи необхідно:

- провести аналіз предметної області для визначення актуальності розробки.
- проаналізувати наявні мобільні додатки-аналоги для виділення їх особливостей. Результатом роботи буде сформований перелік функціональних та системних вимог, на основі яких створюватиметься структура мобільного додатку;
- виконати проектування структури мобільного додатку. По закінченню буде виконано структурно-функціональне моделювання, результатом якого стане створена діаграма IDEF0 та діаграма декомпозиції функціональної моделі. У межах цього етапу також буде розроблена діаграма варіантів використання, щоб проілюструвати взаємозв'язки між системою та її користувачами. Результатом діяльності стане виконане проектування логічної моделі бази даних. Усі етапи проектування будуть необхідними для створення мобільного додатку з конкретними вимогами, що забезпечить точність у процесі розробки функціональних вимог мобільного додатку;
- виконати реалізацію мобільного додатку. Результатом якої стане робочий мобільний додаток з відповідним функціоналом;
- виконати тестування мобільного додатку. Результатом якого стане детальний аналіз наявності критичних помилок у системі.

На основі проведеної роботи виконано планування, яке детально описано в додатку Б.

1.4 Методи дослідження та інструменти реалізації

Перед тим як розпочати реалізацію будь-якого проекту, проводяться численні процеси планування та прийняття стратегічних рішень. Ця фаза має критичне значення, оскільки може суттєво вплинути на подальший розвиток проекту. Вибір

оптимального стеку технологій може визначити час розробки, якість програмного продукту, а також його масштабованість. Тому важливо приділяти належну увагу аналізу всіх можливих варіантів, ретельно враховуючи переваги та недоліки кожного. Стек технологій представляє собою комплекс інструментів, які використовуються при розробці проектів, включаючи мови програмування, фреймворки, системи управління базами даних та інше. Вибір конкретного стеку технологій визначає продуктивність роботи. У ході аналізу було вибрано наступні інструменти реалізації:

- Kotlin [9] – це мова програмування, що зорієнтована на розробку програмного забезпечення для платформи Android. Одним з її аналогів є Java [10], яка вже давно є стандартом для розробки Android-додатків. Однак Kotlin відрізняється більшою ефективністю та зручністю в порівнянні з Java. Kotlin пропонує більш безпечну та експресивну синтаксичну конструкцію, що дозволяє швидше та ефективніше писати код. Вибір Kotlin обумовлений бажанням скоротити час розробки та підвищити якість програмного продукту.
- Android Studio [11] – це інтегроване середовище розробки (IDE) для створення Android-додатків. Його аналогом є IntelliJ IDEA [12]. Проте, Android Studio переважно вибирається розробниками за його широкі можливості, оновлення та інтегровані інструменти для розробки, таким як емулятор Android та підтримка Kotlin. Вибір Android Studio обумовлено бажанням мати доступ до найсучасніших інструментів розробки та найкращої підтримки Kotlin.
- Figma [13] – це онлайн-інструмент для дизайну та прототипування. Його альтернативою є Adobe XD [14]. Figma відрізняється можливістю спільної роботи над проектом в реальному часі без необхідності установки додаткового програмного забезпечення. Це робить його ідеальним вибором для командної роботи та спільної розробки дизайну.

– Firebase [15] - це інтегрована платформа Google для розробки мобільних та веб-додатків, яка надає широкі можливості для збереження та синхронізації даних в реальному часі. Його аналогом є Supabase [16]. Однак використання Firebase обумовлено його простотою в інтеграції, масштабованістю та можливістю реалізації функцій реального часу, що є важливими для розробки Android-додатків. Вибір Firebase дозволить мати потужну та ефективну систему управління базами даних з розширеними можливостями синхронізації та роботи в реальному часі.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Моделювання

Структурно-функціональне моделювання відіграє ключову роль при створенні програмного продукту. Це дозволяє краще зрозуміти систему та оптимізувати її, що допомагає заощадити час і ресурси.

Для представлення функціонального моделювання часто використовують методологію IDEF0. Ця методологія призначена для опису систем і процесів, що мають між собою взаємозв'язки [17]. Вона складається з блоків, що представляють процеси та функції, а також стрілок, які вказують на вхідні та вихідні дані.

На (рис. 2.1) зображено функціональна діаграма, яка ілюструє процес функціонування мобільного додатку для персонального читання новин.

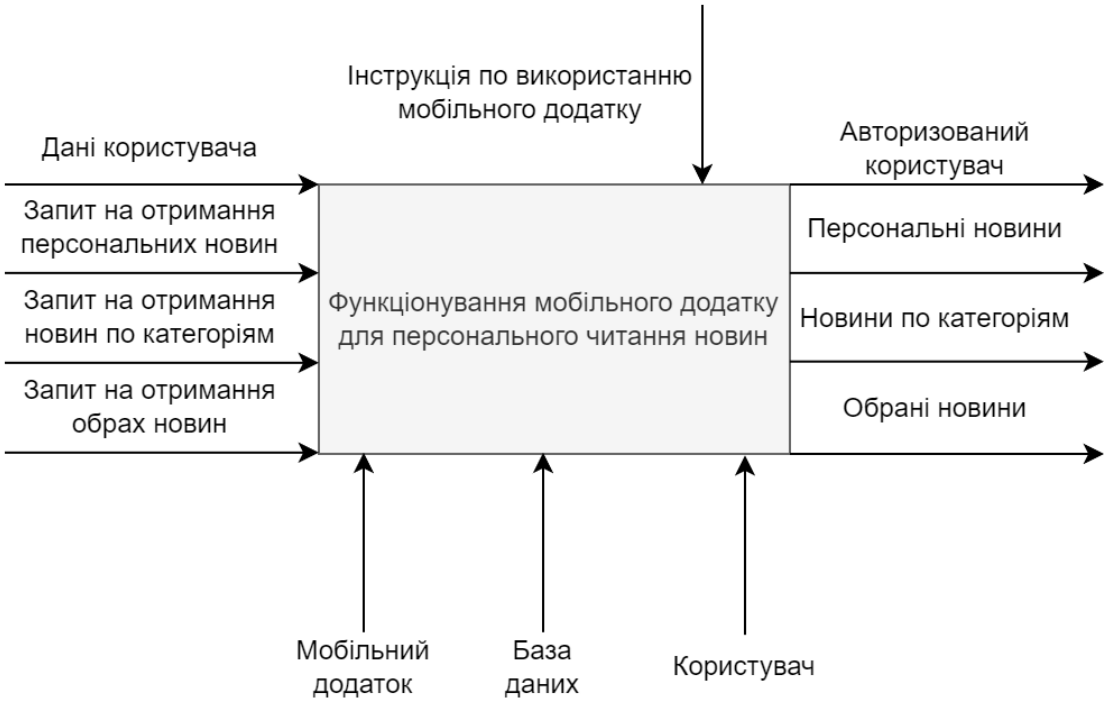


Рисунок 2.1 – Контекстна діаграма IDEF0

На (рис. 2.1) наведена Діаграма IDEF0 яка складається з наступних елементів:

– Вхідні дані: дані користувача, запит на отримання персональних новин, запит на отримання новин по категоріям, запит на отримання обраних новин
– Управління: інструкція по використанню мобільного додатку
– Механізми: мобільний додаток, база даних, користувач
– Вихід: авторизований користувач, персональні новини, новини по категоріям, обрані новини

Для відображення детальних процесів застосовують декомпозицію функціонального проектування. Цей підхід дає змогу розділити функції на більш дрібні та прості елементи. Декомпозиція не обмежується одним рівнем і може тривати доти, доки не буде виокремлено конкретні під процеси.

На (рис. 2.2) наведено декомпозицію функціональної моделі.

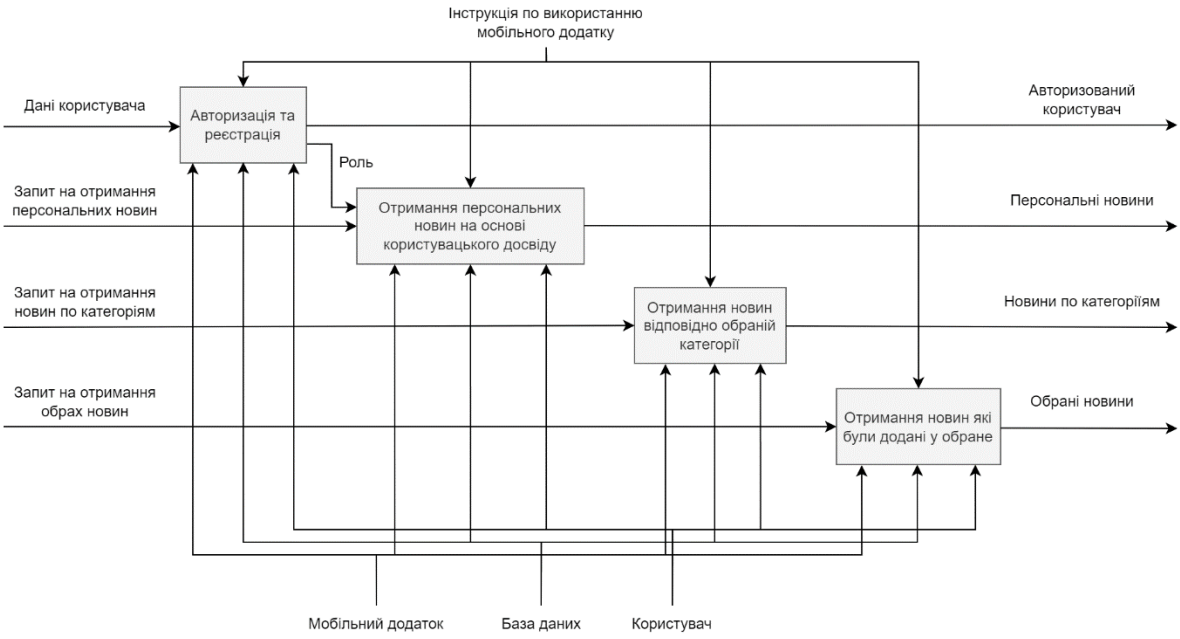


Рисунок 2.2 – Діаграма першого рівня декомпозиції функціональної моделі

Дана декомпозиція розбита на 4 процеси: авторизація та реєстрація, отримання персональних новин на основі користувацького досвіду, отримання новин відповідно обраній категорії, отримання новин які були додані у обране.

2.2 Проектування інформаційної системи

Діаграма варіантів використання (рис. 2.3) є ключовим компонентом у процесі проектування інформаційної системи. Вона демонструє взаємозв'язки між акторами та системою, а також ілюструє функціональні вимоги з боку користувача [18].

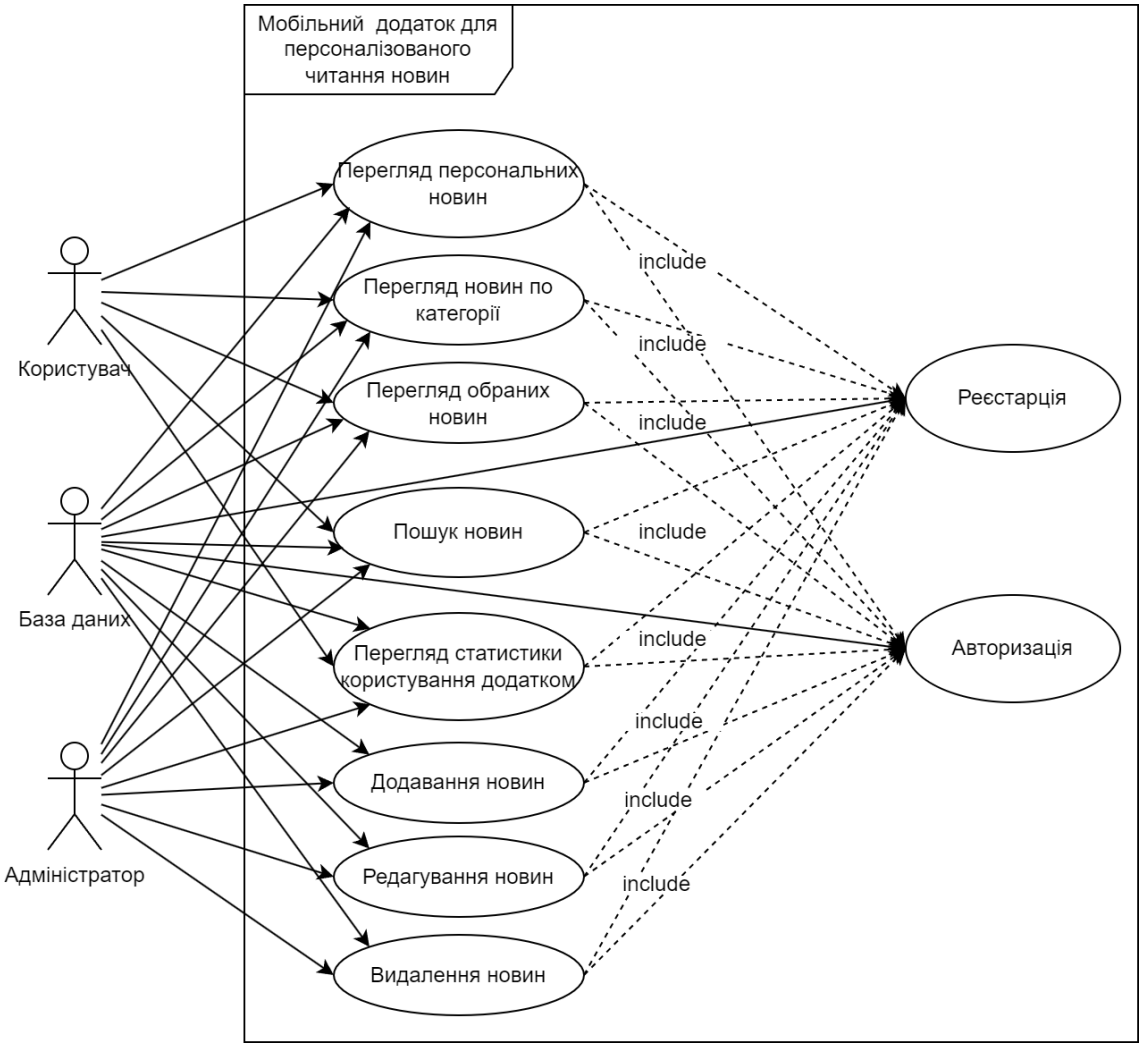


Рисунок 2.3 – Діаграма варіантів використання додатку

Як зазначено на (рис. 2.3), було визначено три типи користувачів:

- Користувач
- Адміністратор

– База даних

На рисунках також показані такі варіанти використання додатку:

– Перегляд персональних новин
– Перегляд новин по категорії
– Перегляд обраних новин
– Пошук новин
– Перегляд статистики користування додатком
– Додавання новин
– Редагування новин
– Видалення новин

2.3 Проектування моделі бази даних

Логічна модель даних розробляється для представлення абстрактної структури інформаційної сфери [19]. Вона відображає сутності, їхні атрибути та взаємозв'язки між ними. Для роботи з базою даних було обрано сервіс Firebase. Далі наведено сутності системи:

Users: сутність яка зберігає в собі дані користувачів, має такі поля:

– admin: поле для встановлення типу актора
– email: поля для збереження пошти користувача
– photo: поле збереження назви фото користувача
– username: поле імені користувача

News: сутність яка зберігає в собі дані про новини, має такі поля:

– category: поле назви категорії новини
– date: поля дати публікації новини
– imageURL: поле з посиланням на фото новини
– sourceName: поле з іменем джерела
– title: поле с заголовком новини

– url: поле з посиланням на новину

Favorite: сутність яка зберігає id новини яка потрібна для розділу «Обране», має такі поля:

– UsersID: поле з id номером користувача з додані новин у розділі «Обране»
– NewsID: поле з id номеров новини для розділу «Обране»

Interests: сутність яка зберігає кількість кликів по категоріям для створення системи рекомендацій, має такі поля:

– business: поле для збереження кількості кликів по категорії бізнес
– movie: поле для збереження кількості кликів по категорії кіно
– medicine: поле для збереження кількості кликів по категорії медицина
– politics: поле для збереження кількості кликів по категорії політика
– sports: поле для збереження кількості кликів по категорії спорт
– technology: поле для збереження кількості кликів по категорії технології
– UsersID: поле з id номером користувача за яким закріплені його інтереси

На (рис. 2.4) наведено логіну модель бази даних.

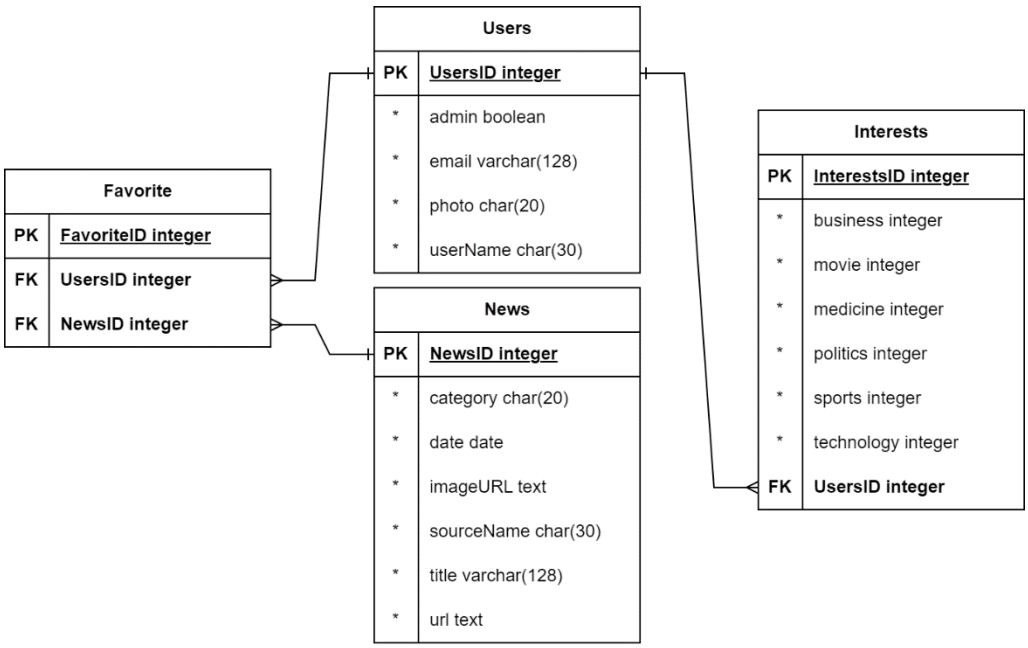


Рисунок 2.4 – Логічна модель бази даних

2.4 Архітектура додатку

Для структурування мобільного додатку використовується шаблон проектування MVVM (Model-View-ViewModel). Цей шаблон надає можливість відокремити рівень представлення даних від бізнес-логіки, що значно полегшує розробку та підтримку додатку.

Model – відповідає за управління даними застосунку. Вона займається отриманням, зберіганням та обробкою даних з Firebase. У нашому додатку моделлю є клас «News». Всі дані щодо новин були зібрані вручну, і для цього проекту використовується власна невелика база даних новин.

View – охоплює користувацький інтерфейс і визначає, як дані показуються користувачеві. У нашому випадку це такі фрагменти, як «ForYou», «Categories», «Favorite» та «UserPage». Ці фрагменти відповідають за відображення відповідних екранів для користувачів і забезпечують їх взаємодію з додатком.

ViewModel – виступає посередником між Model та View. Вона містить логіку представлення, яка дозволяє View взаємодіяти з Model. Клас «MyAdapter» відіграє роль цього посередника, забезпечуючи методи для отримання, обробки та надання даних з бази даних до View. «MyAdapter» відповідає занесення даних рекомендацій для користувачів на основі їхніх інтересів та взаємодій з додатком, а також за управління іншими аспектами логіки додатку.

Використання шаблону MVVM забезпечує чітку структуру проекту, що значно спрощує розробку та підтримку мобільного додатку.

На (рис. 2.5) наведено архітектуру мобільного додатку.

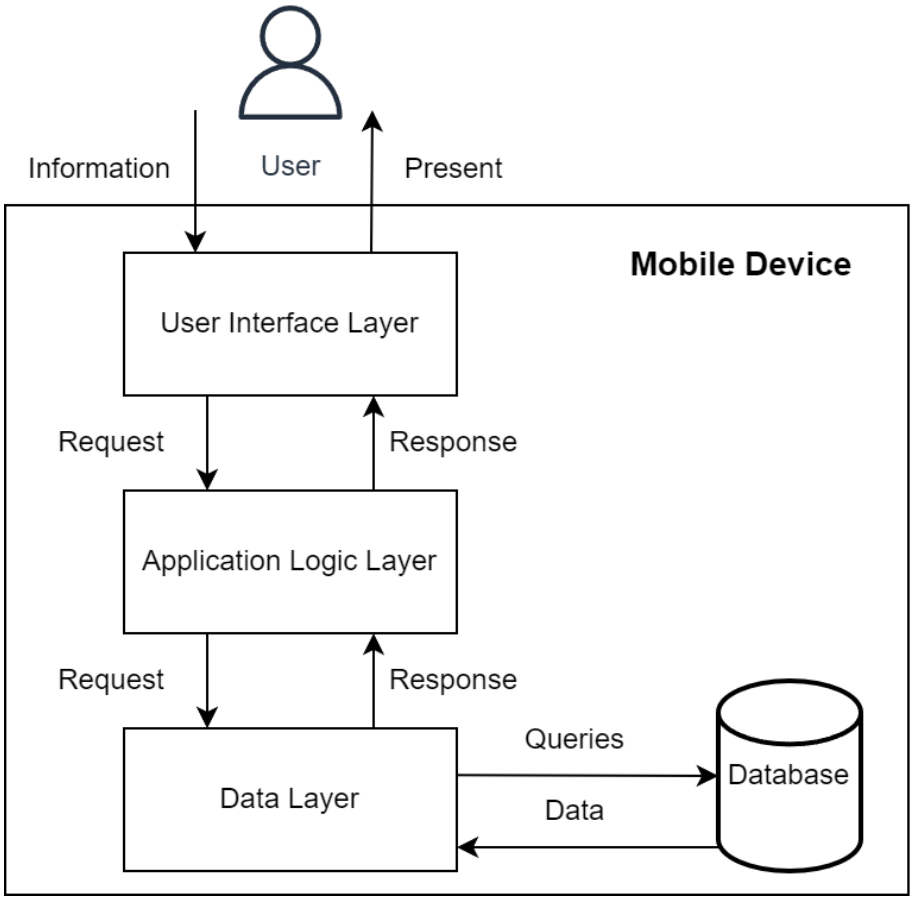


Рисунок 2.5 – Архітектура мобільного додатку

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Програмна реалізація

Розробка мобільного додатку здійснювалася в середовищі Android Studio з використанням мови програмування Kotlin та дизайн сторінок було створено с використанням мови розмітки XML. Мобільний додаток включає в себе чотирьох основних сторінок, зв’язаних між собою навігаційним меню. Переміщення між екранами відбувається за допомогою контролерів, розміщено в нижній частині інтерфейсу.

Для початку роботи над проектом необхідно створити та заповнити базу даних новинами. Спочатку потрібно перейти на сайт firebase.google.com, зареєструйтесь та натисніть кнопку "Створити проект" (рис. 3.1). Далі ввести ім'я проекту та натиснути "Продовжити".

У новому вікні вибрати вкладку "Realtime Database", де створити та заповнити базу даних новинами (рис. 3.2). Після цього відкрити Android Studio, перейти до меню Tools -> Firebase та, дотримуючись підказок, підключіть базу даних до проекту.

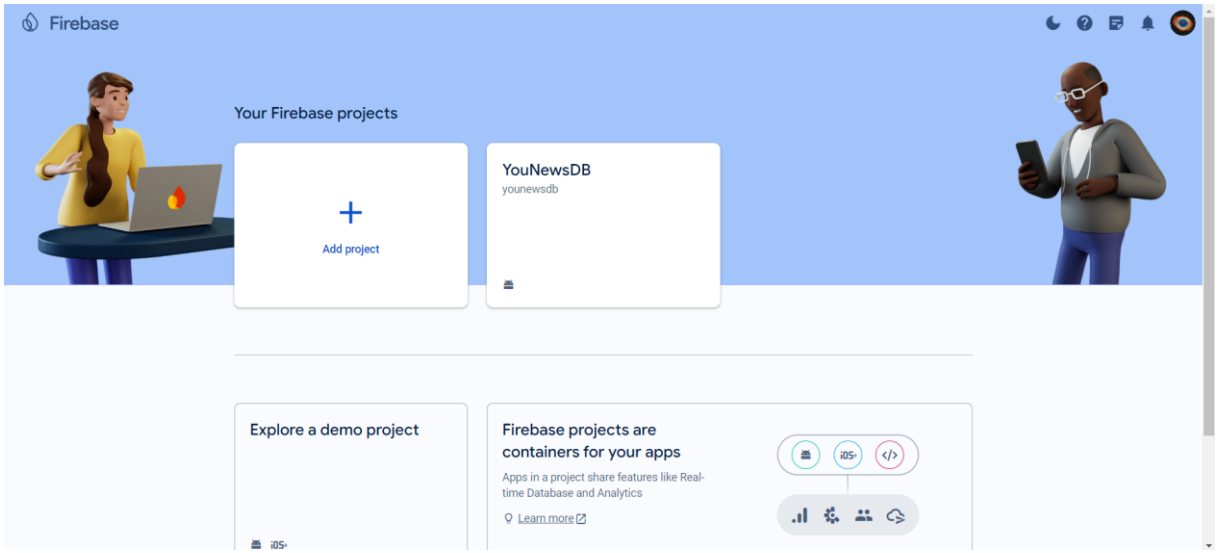


Рисунок 3.1 – Головна сторінка сайту Firebase

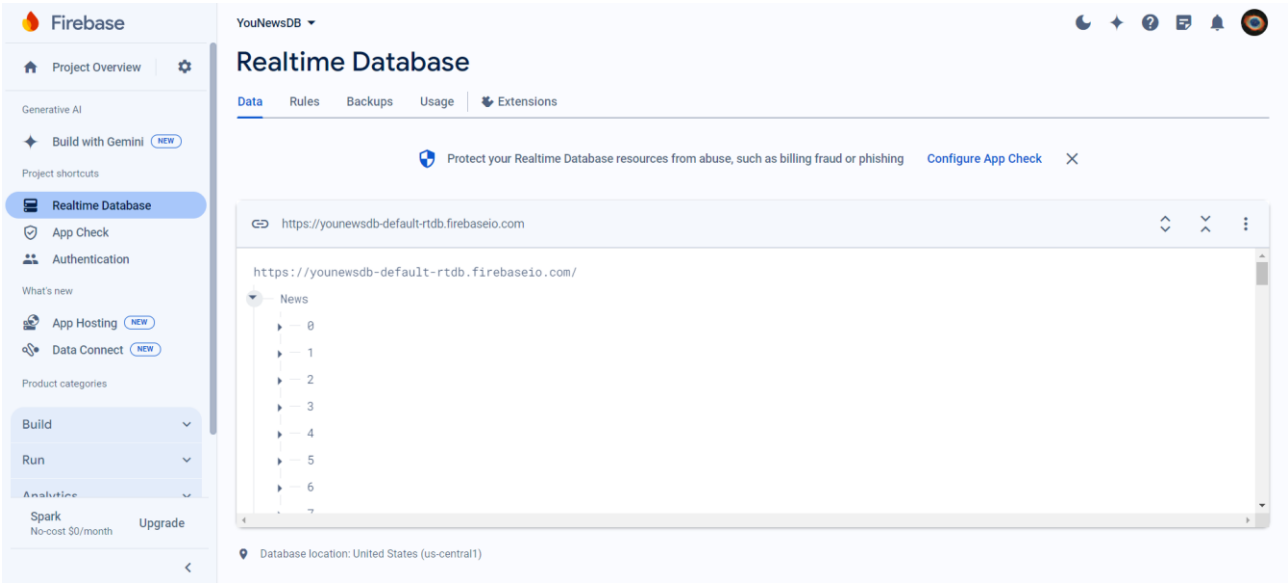


Рисунок 3.2 – Сторінка для створення бази даних

Для авторизації та реєстрації користувачів було створено два класи Login в якому реалізовано метод авторизації та Register в якому реалізовано функцію registerUser. Повний код для класів Login та Register можна знайти у додатку В.

Фрагмент коду класу Login для авторизації:

```
bindingClass.bLogin.setOnClickListener {
            val email = bindingClass.edLEmail.text.toString()
            val password = bindingClass.edLPassword.text.toString()

            if (email.isBlank() || password.isBlank()) {
                Toast.makeText(
                    applicationContext,
                    "Поля не можуть бути порожніми",
                    Toast.LENGTH_SHORT
                ).show()
            ...
```

Фрагмент коду класу Register для реєстрації:

```
private fun registerUser(email: String, password: String, userName: String) {
        FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    val userId = FirebaseAuth.getInstance().currentUser?.uid ?: ""
                    val userInfo = HashMap<String, Any>().apply {
                        put("admin", false)
                        put("email", email)
                        put("userName", userName)
                    ...
```

Всі дані по новинам були зібрані вручну, для цього проекту використовується власна невелика база даних з новинами. Для роботи з новинами було створено клас News:

```
data class News(
    var id: String? = null,
    var category: String? = null,
    var imageURL: String? = null,
    var title: String? = null,
    var sourceName: String? = null,
    var date: String? = null,
    var url: String? = null
)
```

Для зручності роботи з новинами та їхнього виведення на екрани в додатку було створено клас MyAdapter, який керує всіма діями з картками новин. Цей клас відіграє ключову роль у реалізації користувацького інтерфейсу, забезпечуючи зв'язок між даними новин та їхнім відображенням у застосунку. Повний код для класу MyAdapter можна знайти у додатку В.

Фрагмент коду класу MyAdapter:

```
class MyAdapter(private val newsList : ArrayList<News>) :
RecyclerView.Adapter<MyAdapter.MyViewHolder>() {

    class MyViewHolder(itemView : View) : RecyclerView.ViewHolder(itemView) {
        val image: ImageView = itemView.findViewById(R.id.Image)
        val title: TextView = itemView.findViewById(R.id.tvTitle)
        val source: TextView = itemView.findViewById(R.id.tvSourceName)
        val data: TextView = itemView.findViewById(R.id.tvData)
        val star: ImageView = itemView.findViewById(R.id.imgVStar)
        val edit: ImageView = itemView.findViewById(R.id.imgVEdit)
        val delete: ImageView = itemView.findViewById(R.id.imgVDelete)
    }
    ...
```

Для пункту меню "Для вас" був створений клас ForYou, у якому реалізовано функцію loadRecommendedNews для формування персоналізованих рекомендацій на основі користувацького досвіду. Система рекомендацій працює на основі аналізу кількості кликів користувача на окремі категорії новин. Алгоритм рекомендацій починається з отримання даних про кількість кликів для кожної категорії, при умові, що кількість кликів більше 0. Після цього створюється відсоткове відношення між категоріями. Це відношення у відсотках помножується на 5 і визначає кількість рекомендованих новин для кожної категорії. Новини сортуються за датою для

відображення від найсвіжіших до найстаріших. Повний код функції можна знайти у додатку В класу ForYou .

Фрагмент коду функції loadRecommendedNews:

```
private fun loadRecommendedNews() {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =
FirebaseDatabase.getInstance().getReference("Users").child(userId).child("interests")

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    val interestsMap = snapshot.value as? Map<String, Long>
                    interestsMap?.let { interests ->
                        val sortedInterests = interests.filter { it.value > 0 }
                            .toList()
                            .sortedByDescending { (_, value) -> value }

                        if (sortedInterests.isEmpty()) return
                    ...
```

Для другого пункту меню «Категорії» у класі Categories було створено функцію getNewsDataByCategory для фільтрації та виводу новин по категоріям. Повний код функції можна знайти у додатку В класу Categories .

Фрагмент коду функції getNewsDataByCategory:

```
private fun getNewsDataByCategory(category: String) {
        newsArrayList.clear()

        dbref = FirebaseDatabase.getInstance().getReference("News")
        dbref.orderByChild("category").equalTo(category)
            .addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(snapshot: DataSnapshot) {
                    if (snapshot.exists()) {
                        for (newsSnapshot in snapshot.children) {
                            val news = newsSnapshot.getValue(News::class.java)
                            newsArrayList.add(news!!)
                        }
                        newsRecyclerView.adapter = MyAdapter(newsArrayList)
                        ...
```

У третьому пункті меню «Обране» класу Favorite було створено функцію getFavoriteNews для відображення новин яких користувач додав у цей пункт. Повний код функції можна знайти у додатку В класу Favorite.

Фрагмент коду функції getFavoriteNews:

```
private fun getFavoriteNews() {
        val databaseReference: DatabaseReference =

FirebaseDatabase.getInstance().getReference("Users").child(userId).child("favorite")

        databaseReference.addValueEventListener(object : ValueEventListener {
```

```
        override fun onDataChange(snapshot: DataSnapshot) {
            if (snapshot.exists()) {
                favoriteNewsList.clear()
                for (newsSnapshot in snapshot.children) {
                    val newsId = newsSnapshot.key ?: ""
                    FirebaseDatabase.getInstance().getReference("News").child(newsId)
                        .addListenerForSingleValueEvent(object : ValueEventListener {
                            override fun onDataChange(newsDataSnapshot: DataSnapshot) {
                                val news = newsDataSnapshot.getValue(News::class.java)
                            ...
```

У останньому пункті меню «Користувач» класу UserPage було створено функція loadUserInterestsAndDetails для завантаження інформації про користувача, його фото аватара, ім’я та данні ко користуванню додатком для створення кругової діаграми. Дані, що використовуються для створення кругової діаграми, це кількість кликів користувача по кожній категорії новин. Кругова діаграма створена за допомогою бібліотеки MPAndroidChart. Повний код функції можна знайти у додатку В класу UserPage.

Фрагмент коду функції loadUserInterestsAndDetails:

```
private fun loadUserInterestsAndDetails() {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef = FirebaseDatabase.getInstance().getReference("Users").child(userId)

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                // Load user interests
                val interestsSnapshot = snapshot.child("interests")
                val interestsMap = interestsSnapshot.value as? Map<String, Long>
                interestsMap?.let {
                    val filteredInterestsMap = interestsMap.filter { it.value > 0 }
                    val total = filteredInterestsMap.values.sum().toFloat()
                    val entries = filteredInterestsMap.entries.map { entry ->
                        PieEntry(entry.value.toFloat() / total * 100, entry.key)
                    }
                    ...
```

Для пошуку новин по ключовим словам було створено функцію searchNewsByTitle для основного класу MainActivity. Повний код функції можна знайти у додатку В класу MainActivity.

Для відображення новин у весь екран при натисканні на неї було створено клас UrlNewsActivity в який передається посилання на новину для її відображення.

Для додавання новин адміністратором було створено функцію addNews у класі Admin. Повний код функції можна знайти у додатку В класу Admin.

Фрагмент коду функції searchNewsByTitle:

```
private fun searchNewsByTitle(title: String) {
        newsArrayList.clear()

        dbref = FirebaseDatabase.getInstance().getReference("News")
        dbref.orderByChild("title")
            .addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(snapshot: DataSnapshot) {
                    if (snapshot.exists()) {
                        for (newsSnapshot in snapshot.children) {
                            val news = newsSnapshot.getValue(News::class.java)
                            if (news != null && news.title != null && news.title!!.contains(
                                    title,
                                    ignoreCase = true
                                )
                            ...
```

Клас UrlNewsActivity:

```
class UrlNewsActivity : AppCompatActivity() {
    private lateinit var binding: ActivityUrlNewsBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityUrlNewsBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val url = intent.getStringExtra("URL")
        binding.webView.loadUrl(url!!)
    }
}
```

Фрагмент коду функції addNews:

```
private fun addNews() {
        val category = bindingClass.etCategory.text.toString().trim()
        val imageURL = bindingClass.etImageURL.text.toString().trim()
        val title = bindingClass.etTitle.text.toString().trim()
        val sourceName = bindingClass.etSourceName.text.toString().trim()
        val date = bindingClass.etDate.text.toString().trim()
        val url = bindingClass.etURL.text.toString().trim()

        if (category.isBlank() || !isValidCategory(category)) {
            Toast.makeText(this, "Такої категорії немає", Toast.LENGTH_SHORT).show()
            return
        }
        if (imageURL.isBlank() || !isValidImageURL(imageURL)) {
            Toast.makeText(this, "Некоректний URL зображення", Toast.LENGTH_SHORT).show()
            return
        }
        if (title.isBlank()) {
            Toast.makeText(this, "Заголовок не може бути порожнім", Toast.LENGTH_SHORT).show()
            return
        }
        ...
```

3.2 Використання розробленого продукту

Для початку роботи у додатку потрібно авторизуватись або зареєструватись якщо немає облікового запису (рис. 3.3).

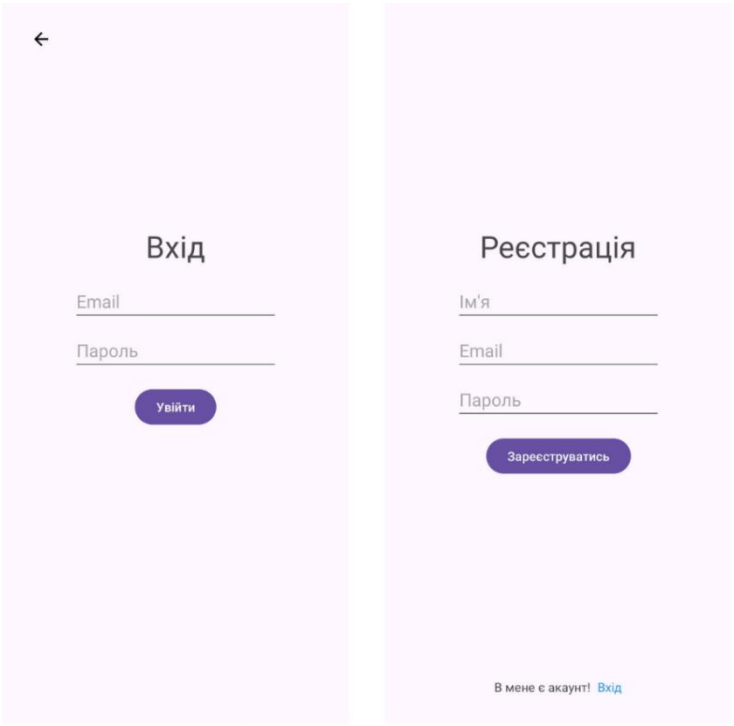


Рисунок 3.3 – Авторизація та Реєстрація

Після входу в систему користувач потрапляє на головний екран і в меню «Для вас» (рис. 3.4), де він може переглянути персоналізовані новини на основі своїх інтересів. За потреби користувач може натиснути на новину, щоб відкрити її на весь екран для зручнішого перегляду (рис. 3.4). У картці з новиною є біла зірка: при натисканні вона стає жовтою, і новина додається до розділу «Обране» (рис. 3.5). Також на головній сторінці є іконка з лупою. При натисканні на неї відкривається нове вікно з полем для введення інформації. Після введення тексту і натискання на кнопку пошуку на клавіатурі система знайде новини, що відповідають введеному тексту (рис. 3.5).

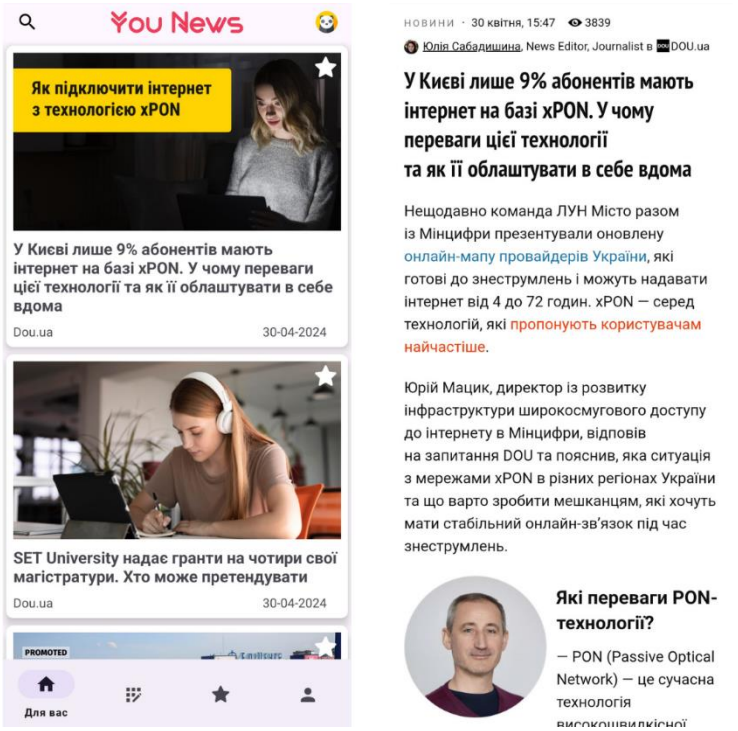


Рисунок 3.4 – Розділ «Для вас» та перегляд новини

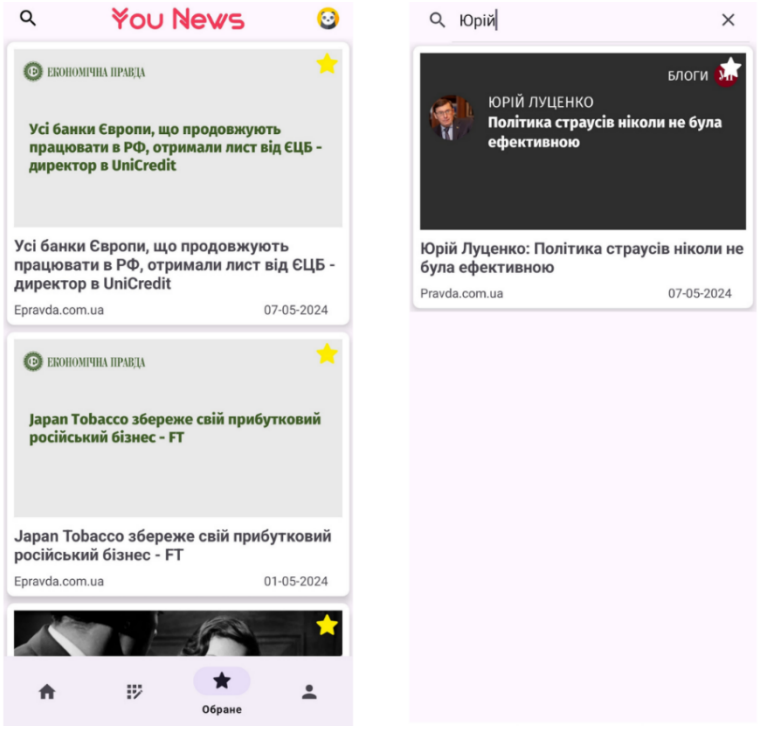


Рисунок 3.5 – Розділ «Обране» та сторінка пошуку

У розділі меню «Категорії» відображаються всі новини. Користувач може вибрати одну з категорій, щоб переглядати новини лише з цієї категорії (рис. 3.6).

У розділі «Користувач» відображається фото профілю користувача, його ім’я, кнопка для виходу з акаунту та кругова діаграма, яка показує статистику використання додатку. Діаграма відображає процентне співвідношення категорій новин, які зацікавили користувача (рис. 3.6).

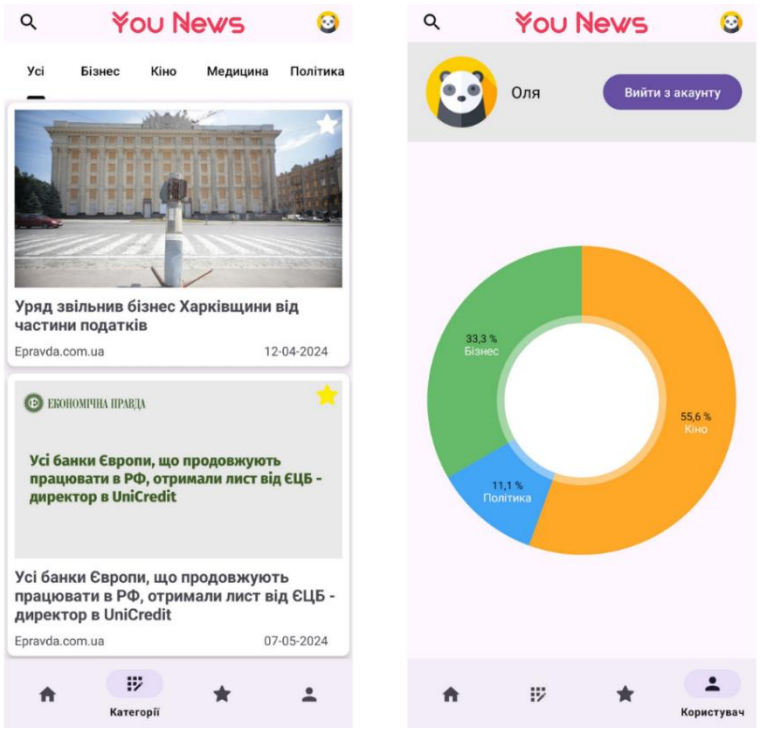


Рисунок 3.6 – Розділ «Категорії» та «Користувач»

Якщо в систему входить адміністратор зі своєю електронною поштою та паролем, дизайн додатку трохи змінюється. У картках новин з’являються дві нові кнопки: редагування та видалення (рис. 3.7). При натисканні на іконку редагування відкривається вікно з даними новини, які можна змінити та зберегти, або скасувати зміни, натиснувши відповідну кнопку. При натисканні на іконку видалення відкривається вікно з підтвердженням дії та двома кнопками: "Так" та "Ні". Також на сторінці «Користувач» з’являється нова кнопка «Панель адміністратора», яка відкриває вікно для додавання новин (рис. 3.8).

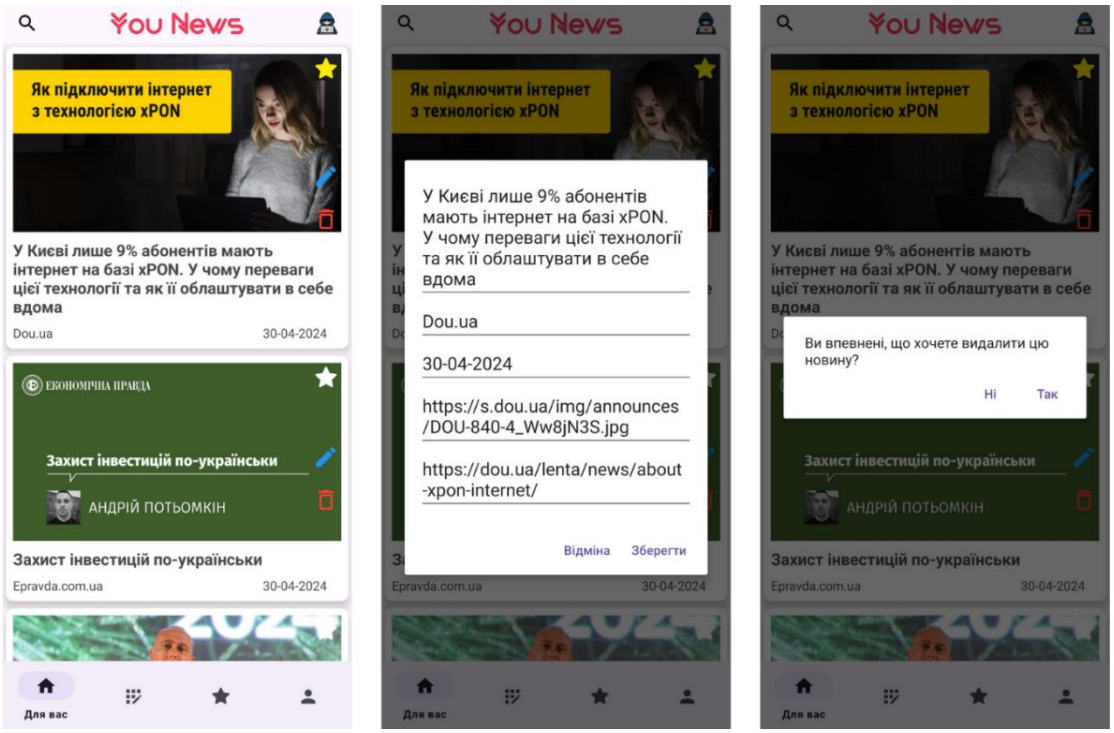


Рисунок 3.7 – Редагування на видалення новин

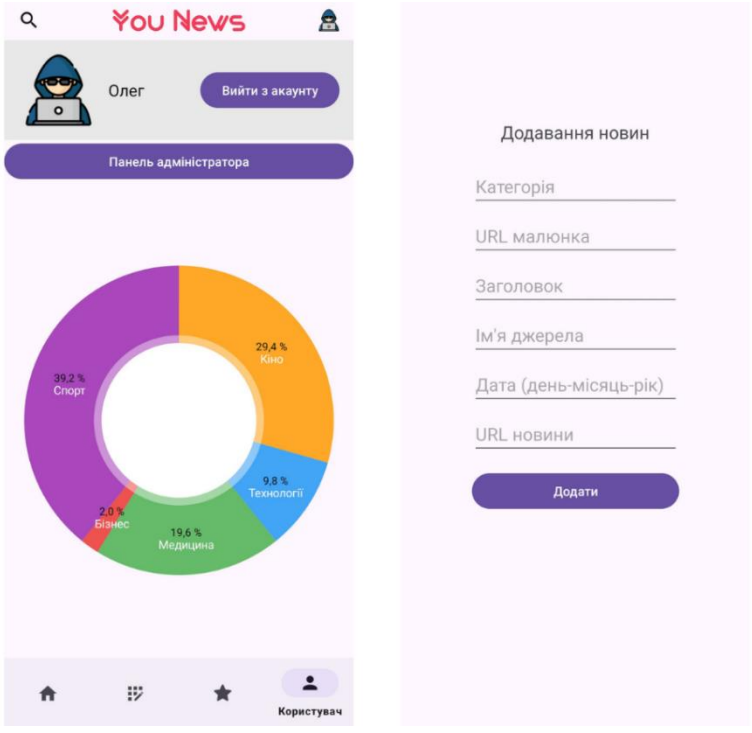
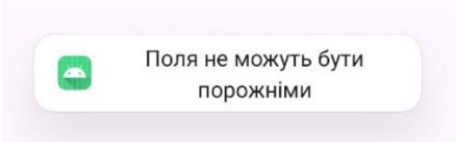
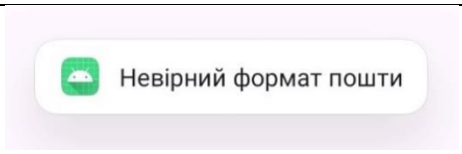
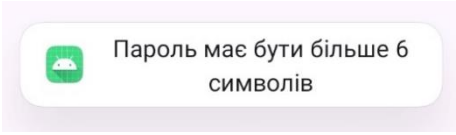
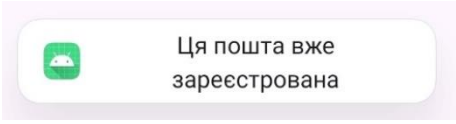
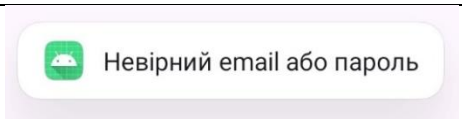
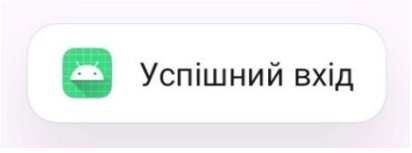


Рисунок 3.8 – Розділ «Користувач» та сторінка додавання новин

3.3 Тестування розробки

Даний етап розробки є ключовим для забезпечення якості продукту, оскільки тестування дозволяє переконатися, що програмне забезпечення працює належним чином і відповідає вимогам користувачів. Важливим аспектом тестування є виконання юніт-тестів, які допомагають переконатися, що окремі частини програми функціонують правильно. Юніт-тести зазначені у таблиці 3.1

Таблиця 3.1 – Юніт-тести

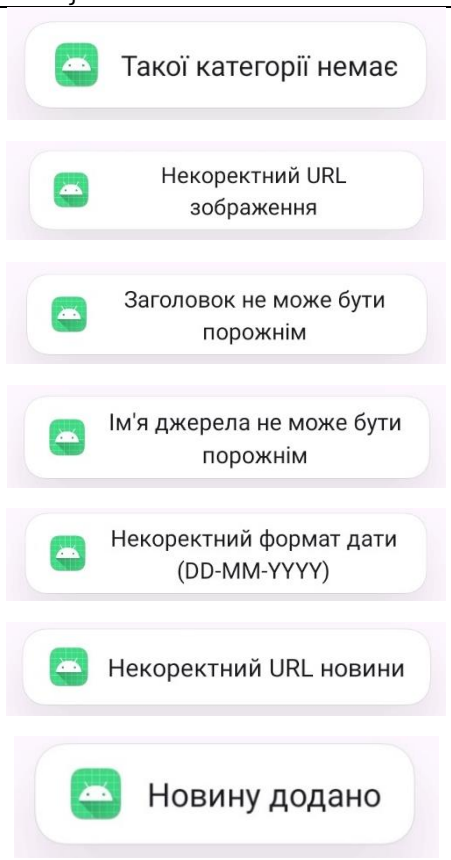
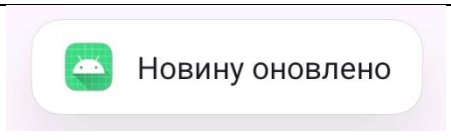
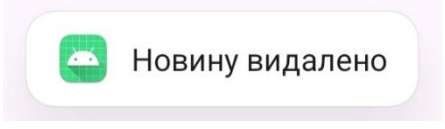
Назва тесту	Очікуваний результат	Фактичний результат	0/1
Перевірка шо поля не порожні	Отримання повідомлення що поля порожні		1
Введення некоректної даних при реєстрації	Отримати повідомлення про некоректні дані	 	1
Зареєструвати пошту що вже є в системі	Отримання повідомлення, що така пошта вже зареєстрована		1
Перевірка даних при авторизації	Отримати сповіщення про стан авторизації	 	1

Продовження таблиці 3.1

Назва тесту	Очікуваний результат	Фактичний результат	0/1

Помилка отримання даних з бази даних	Отримати сповіщення про помилку отримання даних	`override fun onCancelled(error: DatabaseError) {`
		`Toast.makeText(requireContext(), "Помилка отримання даних", Toast.LENGTH_SHORT)`
		`.show()`
		`}`
			1

Перевірка що користувач авторизувався	Отримати доступ до функціоналу додатку	`if (FirebaseAuth.getInstance().currentUser == null) {`
		`val intent = Intent(this, Register::class.java)`
		`startActivity(intent)`
		`finish()`
		`}`
			1

Перевірка на коректність ведених даних при додаванні новини	Отримати сповіщення про некоректні дані		1
Редагування новини	Отримати сповіщення про стан даних		1
Видалення новини	Отримати сповіщення про видалення новини		1

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було створено мобільний додаток для персоналізованого читання новин, що значно покращує досвід користувачів у цій галузі.

На початку роботи було проведено глибокий аналіз актуальних досліджень і вивчення предметної області. Ці дослідження підтвердили важливість розробки додатку для індивідуального читання новин. Було також виконано огляд існуючих аналогів, що дозволило визначити їхні сильні та слабкі сторони, аби врахувати їх під час розробки. В результаті аналізу було визначено функціональні та системні вимоги до додатку. На основі технічного завдання було створено детальний план виконання робіт. Цей план включав постановку мети за методом SMART, планування робіт за допомогою WBS та структуру організації OBS, а також побудову діаграми Ганта і аналіз ризиків.

У процесі проектування була розроблена контекстна діаграма в нотації IDEF0, яка чітко відображає ключові процеси системи та їх взаємодію. Для більш детального відображення кожного процесу було виконано декомпозицію першого рівня, яка дозволила розбити процеси на підпроцеси. Взаємодію користувачів з мобільним додатком було описано за допомогою діаграми варіантів використання, що дало змогу чітко окреслити всі можливі способи взаємодії з системою. Також було розроблено логічну модель додатку для зображення сутностей у базі даних. Описано архітектуру додатку для розуміння взаємодії модулів між собою. Проведено детальний аналіз розробки продукту з описом варіантів використання для різних типів користувачів та виконано юніт-тестування для перевірки працездатності додатку, результати якого були занесені в таблицю.

Результатом роботи став готовий мобільний додаток для персонального читання новин, що значно підвищує якість користувацького досвіду у читанні новин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іздебський В. 15 трендів мобільної розробки у 2024 році. *Wezom*. URL: https://wezom.com.ua/ua/blog/trendi-mobilnoyi-rozrobki-na-2023-rik (дата звернення: 29.04.2024).

2. Штучний інтелект у розробці мобільних додатків. *Stfalcon*. URL: https://stfalcon.com/uk/blog/post/the-impact-of-artificial-intelligence-on-mobile-app-development (дата звернення: 29.04.2024).

3. Ставлення населення до ЗМІ та споживання різних типів медіа 2022. URL: https://internews.in.ua/wp-content/uploads/2023/10/Ukrainski-media-stavlennia-ta-dovira-2023r.pdf (дата звернення: 29.04.2024).

4. Снопок О. Дивимося, читаємо, слухаємо: як змінилося медіаспоживання українців в умовах повномасштабної війни. *Українська правда*. URL: https://www.pravda.com.ua/columns/2022/06/22/7353987/ (дата звернення: 29.04.2024).

5. The role of machine learning algorithms in Android app recommendation systems. *MoldStud*. URL: https://moldstud.com/articles/p-the-role-of-machine-learning-algorithms-in-android-app-recommendation-systems (дата звернення: 29.04.2024).

6. Google News - Daily Headlines. *Google Play*. URL: https://play.google.com/store/apps/details?id=com.google.android.apps.magazines&hl=en_US&pli=1 (дата звернення: 29.04.2024).

7. BBC News. *Google Play*. URL: https://play.google.com/store/apps/details?id=bbc.mobile.news.uk&hl=en_US (дата звернення: 29.04.2024).

8. The global newsgathering power of CNN. *CNN*. URL: https://edition.cnn.com/specials/mobile-apps (дата звернення: 29.04.2024).

9. Kotlin. URL: https://kotlinlang.org/ (дата звернення: 29.04.2024).

10. Ardito L. Effectiveness of Kotlin vs. Java in android app development tasks. *ScienceDirect*.

URL: https://www.sciencedirect.com/science/article/abs/pii/S0950584920301439#previewsection-references (дата звернення: 29.04.2024).

11. Android Studio: переваги та особливості. *Qagroup*. URL: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/ (дата звернення: 29.04.2024).

12. Android Studio vs. IntelliJ IDEA: Which is Better for Kotlin Development?. *Dopebase*. URL: https://dopebase.com/android-studio-intellij-idea-better-kotlin-development (дата звернення: 29.04.2024).

13. Особливості Figma. *Web4u*. URL: https://web4u.in.ua/blog/osoblivost-figma-34 (дата звернення: 29.04.2024).

14. Прототипування за допомогою Figma, Sketch та Adobe XD: порівняльний аналіз та практичні прийоми роботи. *Mate academy*. URL: https://mate.academy/blog/ui-ux-design/prototyping-tools/ (дата звернення: 29.04.2024).

15. Firebase | Google's Mobile and Web App Development Platform. Firebase. URL: https://firebase.google.com/ (дата звернення: 29.04.2024).

16. Firebase vs Supabase: Choosing the Right Tool for Your Project. *Flatirons*. URL: https://flatirons.com/blog/firebase-vs-supabase/#:~:text=Firebase,%20offers%20a%20range%20of,custom%20functions%20using%20PL/pgSQL. (дата звернення: 29.04.2024).

17. Створення схем IDEF0. *Microsoft*. URL: https://support.microsoft.com/uk-ua/topic/створення-схем-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc#:~:text=IDEF0%20–%20це%20визначення%20інтеграції%20для,Це%20тип%20блок-схеми. (дата звернення: 17.05.2024).

18. Каграманова Ю. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. *Dou*. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 17.05.2024).

19. Maxym Z. Моделювання даних (Data Modelling). *Maxzosim*. URL: https://www.maxzosim.com/data-modelling/ (дата звернення: 17.05.2024).

20. Гончарук М. Що таке концепція SMART. *Lemon School*. URL: https://lemon.school/blog/shho-take-kontseptsiya-smart (дата звернення: 29.04.2024).

21. Zosym Maxym. Структура розбиття робіт (Work Breakdown Structure - WBS). *Maxzosim*. URL: https://www.maxzosim.com/struktura-rozbittia-robit/ (дата звернення: 29.04.2024).

22. Organization Breakdown Structure (OBS). *Upland*. URL: https://uplandsoftware.com/psa/resources/glossary/organization-breakdown-structure-obs/ (дата звернення: 29.04.2024).

23. What are Gantt charts?. *Atlassian*. URL: https://www.atlassian.com/agile/project-management/gantt-chart (дата звернення: 29.04.2024).

24. Що таке аналіз ризику: визначення та інструменти. *Visure*. URL: https://visuresolutions.com/uk/блог/аналіз-ризику/ (дата звернення: 29.04.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Мобільний додаток для персоналізованого читання новин»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій

_____ Нагорний В.В.

Студент групи ІТ-03

_____ Іванов О.В.

Суми 2024

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ МОБІЛЬНОГО ДОДАТКА

1.1 Призначення мобільного додатку

Призначення мобільного додатку полягає в забезпеченні користувачам персоналізованого та зручного доступу до новин.

1.2 Мета створення мобільного додатку

Головна мета проекту - це створення зручного мобільного додатку для читання новин, який враховує індивідуальні інтереси та вподобання кожного користувача.

1.3 Цільова аудиторія

Цільова аудиторія мобільного додатку охоплює широкий спектр користувачів, які полюбляють персоналізований підхід до читання новин та бажають отримувати актуальну інформацію відповідно до своїх індивідуальних інтересів.

2 ВИМОГИ ДО МОБІЛЬНОГО ДОДАТКУ

2.1 Вимоги до мобільного додатку в цілому

2.1.1 Вимоги до структури й функціонування

Мобільний додаток повинен бути реалізований за допомогою сучасних можливостей мобільної розробки, тобто використовувати новітній графічний інтерфейс у парі з мовою програмування Kotlin та забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проекту має бути високоякісним мобільний додаток, який функціонує з пристроями на базі ОС Android та містити інформативний контент та графічні матеріали.

2.1.2 Вимоги до персоналу

Для роботи з мобільним додатком персонал закладу не потребує особливих навичок. Єдиною вимогою є вміння користуватися мобільним телефоном.

2.1.3 Вимоги до збереження інформації

Уся інформацію повинна зберігатися у СКБД Firebase.

2.1.4 Вимоги до розмежування доступу

Мобільний додаток повинен бути загальнодоступним у мережі Інтернет. Для розмежування доступу визначаються два основні групи користувачів: адміністратор та користувач.

Адміністратор має необмежений доступ до даних з правами перегляду, додавання, редагування та видалення даних. Доступ до адміністративної панелі надається за спеціальним логіном та паролем.

Користувач додатку може переглядати новини на сторінках, зберігати налаштування та історію у хмару, переглядати персональні новини, дивитися новини за категоріями, додавати новини у обране та шукати новини за ключовими словами.

2.2 Структура мобільного додатку

2.2.1 Загальна інформація про структуру мобільного додатку

Загальна структура мобільного додатку передбачає логічний розподіл основних елементів інтерфейсу з метою забезпечення зручного та ефективного користування.

Нижче наведено перелік основних сторінок та елементів:

- Головний екран «Для вас» – Вітає користувача та надає швидкий доступ до основних розділів додатку. Відображає персоналізовані рекомендації новин на основі інтересів користувача.
- Розділ "Категорії" – Надає користувачам можливість переглядати останні новини за категоріями. Забезпечує функцію пошуку та фільтрації новин за ключовими словами.
- Розділ "Обране" – Зберігає новини, вибрані користувачем для подальшого перегляду.

- Розділ "Користувач" – Включає особистий кабінет користувача. В ньому знаходиться меню налаштувань, що містить додаткові опції, такі як зміна мови, нагадування та інші налаштування.
- Елемент "Пошук" – Надає можливість користувачам використовувати розширений пошук новин за різними критеріями.
- Сторінка адміністратора – це прихований розділ, який надає можливість редагувати контент у додатку за допомогою доданих елементів управління. Доступ до цієї сторінки має лише адміністратор з власним логіном та паролем, що забезпечує конфіденційність і безпеку даних.

2.2.2 Навігаційне меню

Для зручності навігації повинно бути створено меню, що забезпечить швидке переміщення користувача по додатку. Меню повинно бути закріплене і розташовуватись знизу екрану на кожній сторінці.

2.2.3 Управління контентом

Керування контентом всередині мобільного додатку здійснюватиметься через панелі адміністратора. Вся інформація міститиметься в системі управління базою даних.

2.2.4 Дизайн мобільного додатку

Дизайн додатку має бути виконаний у мінімалістичному та сучасному стилі. Кольорову схему додатку можна буде налаштувати у пункті «Налаштування».

Види і розміри шрифтів повинні бути комфортними для перегляду. Інформаційні блоки, графічні матеріали та інші елементи сторінок повинні мати зручне і логічне розташування. Шаблон майбутнього програмного продукту зображено на рисунку А.1

Рисунок А.1 – Шаблон дизайну мобільного додатку

2.2.5 Система навігації (карта мобільного додатку)

Карта додатку зображена на рисунку А.2.

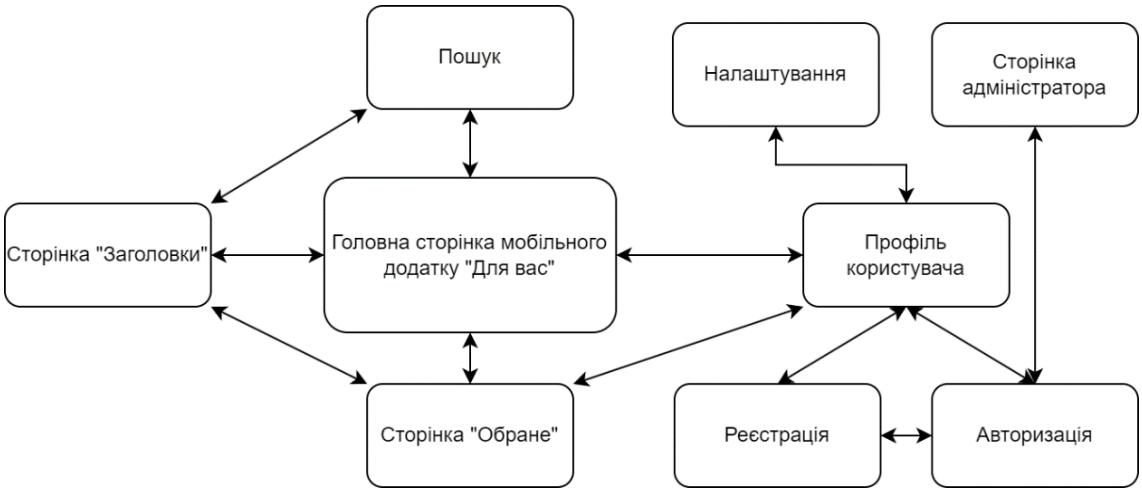


Рисунок А.2 – Система навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Мобільний додаток повинен бути реалізований українською мовою.

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи мобільного додатку пристрій повинен бути під операційною системою Android та не старіше версії 8.0 (Oreo).

2.4 Вимоги до функціонування системи

2.4.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреба користувача	Джерело
UN-01	Перегляд новин	Користувач, Адміністратор
UN-02	Збереження уподобань та налаштувань	Користувач, Адміністратор
UN-03	Отримання рекомендацій на основі читацьких інтересів	Користувач, Адміністратор
UN-04	Перегляд новин по категоріям	Користувач, Адміністратор
UN-05	Додавання новин у обране	Користувач, Адміністратор
UN-06	Перегляд статистики та історії персоналізованих новин	Користувач, Адміністратор
UN-07	Редагування даних	Адміністратор

2.4.2 Функціональні вимоги

З урахуванням потреб користувачів мобільного додатку та завдань, передбачених для персоналу, визначено наступні функціональні вимоги:

- Реалізація системи реєстрації та авторизації користувачів для доступу до персоналізованих новин та функціоналу додатку.
- Можливість налаштування та збереження особистих уподобань користувача для персоналізованого підбору новин.
- Динамічний пошук та перегляд актуальних новин, враховуючи індивідуальні інтереси користувача.
- Перегляд статистики та історії персоналізованих новин для кращого розуміння читацьких вподобань.
- Використання алгоритму для підбору та рекомендацій новин.

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ МОБІЛЬНОГО ДОДАТКУ

Детальний опис етапів створення додатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення мобільного додатку

№	Склад і зміст робіт	Строк розробки
1	Визначення вимог та аналіз ринку мобільних додатків	5 днів
2	Розробка технічного завдання та концепції додатку	6 днів
3	Розробка шаблону додатку	5 днів
4	Проектування інтерфейсу та визначення функціональності	5 днів
5	Розробка основної структури додатку та його модулів	20 днів
6	Створення та підключення БД	4 дні
7	Тестування та виправлення помилок	5 днів
8	Створення документації до мобільного додатку	3 дні
9	Реліз мобільного додатку	1 день
	Загальна тривалість робіт	54 дні

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВВЕДЕННЯ МОБІЛЬНОГО ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Мобільний додаток повинен бути затверджений та бути розміщений на платформі Google Play.

ДОДАТОК Б
Планування робіт

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є мобільний додаток доповненої для персоналізованого читання новин.

Результати деталізації мети методом SMART [20] наведено у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific (Конкретність)	Створити мобільний додаток, що автоматично адаптує новини для користувачів на основі їхніх індивідуальних інтересів та переваг.
Measurable (Вимірюваність)	Забезпечити можливість користувачам оцінювати та взаємодіяти з новинами, здійснювати персоналізований вибір контенту та забезпечувати метрики ефективності на основі їхніх взаємодій.
Achievable (Досяжність)	Розробка додатка на основі сучасних технологій, таких як мова програмування Kotlin, фреймворки Android SDK та Android Studio, а також інструменти для дизайну, такі як Figma. Наявність кваліфікованих фахівців та належні ресурси дозволяють здійснити проект відповідно до встановлених критеріїв.
Relevant (Узгодженість, важливість)	Реалізація додатка важлива для поліпшення користувацького досвіду читання новин та залучення більшої аудиторії до використання додатка.
Time-framed (Визначеність за термінами)	Завершити розробку та впровадження додатка до 30 травня 2024 року.

Планування змісту робіт. Планування робіт розпочинається зі створення WBS (Work Breakdown Structure) діаграми – це графічне представлення групованих

елементів проекту у вигляді пакетів робіт, які ієрархічно пов’язані з проектом. На першому рівні WBS фіксується сам продукт проекту, який відповідає основній меті. Наступні рівні відображають дії та заходи, необхідні для досягнення цілей проекту [21]. На рисунку Б.1 представлено WBS-діаграма з розробки мобільний додаток.

Планування структури виконавців. Для розподілу завдань та графічного відображення організаційної структури проекту використовуються діаграма OBS (Organization Breakdown Structure) [22]. Вона допомагає чітко визначити ролі та відповідальності кожного учасника проекту. OBS діаграма наведена на рисунку Б.2.

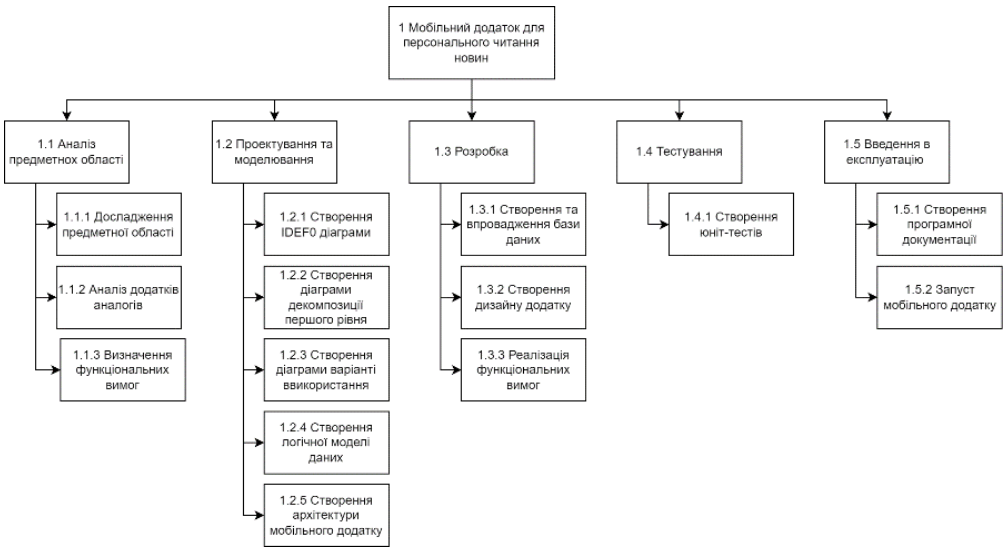


Рисунок Б.1 – WBS-діаграма

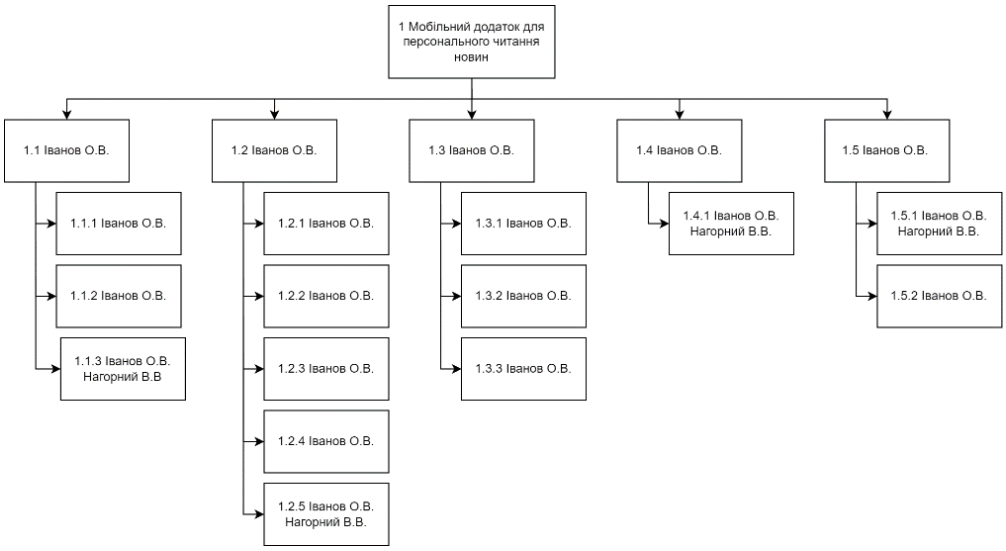


Рисунок Б.2 – OBS-діаграма

Список виконавців проекту знаходиться в Таблиця Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Іванов О.В.	Виконує розробку додатку
Проектувальник	Іванов О.В.	Визначає функціонал системи та інструменти реалізації. Розробляє макет інтерфейсу
Тестувальник	Іванов О.В.	Виконує тестування функціоналу додатку та його дизайну
Керівник проекту	Нагорний В.В.	Формує завдання на розробку проекту
Менеджер проекту	Іванов О.В.	Здійснює контроль за виконанням термінів, розподілом ресурсів та завдань між учасниками. Відповідає за проведення збору та аналізу даних.

Діаграма Ганта. Побудова календарного плану (діаграми Ганта) [23] є одним з важливих етапів планування проекту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят.

Діаграма Ганта представлено на рисунку Б.3.

Аналіз ризиків. Важливим аспектом успішної реалізації проекту є ідентифікація потенційних ризиків. Необхідно проаналізувати їх можливий вплив та розробити стратегії їх усунення. Цей процес включає систематичне визначення, оцінку та управління ризиками з метою мінімізації негативного впливу на проект [24].

У Таблиця Б.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

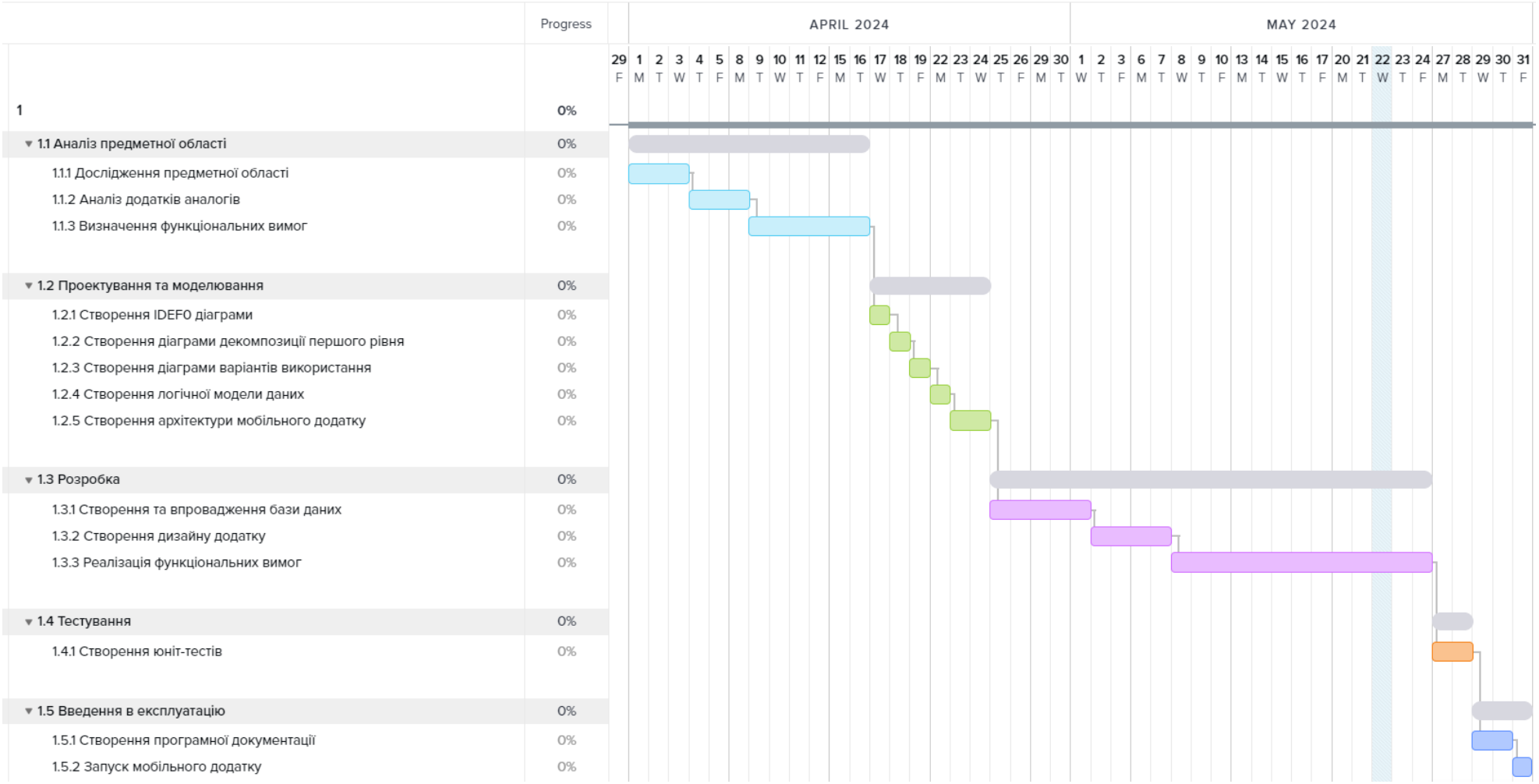


Рисунок Б.3 – Діаграма Ганта

Таблиця Б.3 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятний
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

У результаті планування заходів реагування на ризики проекту було отримано матрицю ймовірності виникнення та впливу ризиків таблиця Б.4. Зеленим кольором на позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі. Класифікація ризиків проекту за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.5. У таблиці Б.6 зазначені ризики та стратегії реагування.

Таблиця Б.4 – Матриця ймовірності та впливу згідно проекту

Ймовірність ризику	Вплив загрози (ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0.05	0.1	0.2	0.4	0.8
0.9	0.045	0.09 R8	0.18	0.36	0.72
0.7	0.035	0.07	0.14 R3, 7	0.28	0.56
0.5	0.025	0.05 R4	0.1 R1	0.2	0.4
0.3	0.015	0.03 R2, 9	0.06 R10	0.12 R5, 6	0.24
0.1	0.005	0.01	0.02	0.04	0.08

Таблиця Б.5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять (номера)
1	Прийнятні	$0.005 \leq R \leq 0.05$	2, 4, 9
2	Виправдані	$0.05 < R \leq 0.14$	1, 3, 5, 6, 7, 8, 10
3	Недопустимі	$0.14 < R \leq 0.72$	

Таблиця Б.6 – Ризики проекту та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
1	Виправданий	Хвороба учасника команди	0.5	0.2	0.1	-	Розподіл завдань і ресурсів між членами команди	Запобігання можливим затримкам у виконанні проекту
2	Прийнятний	Запланована відпустка співробітника	0.3	0.1	0.03	-	Розподіл завдань і ресурсів між членами команди	Узгодження графіку роботи замість відпустки

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
3	Виправданий	Блекаут	0.7	0.2	0.14	Ухилення	Резервне живлення, використання резервних джерел енергії	Розробка плану відновлення роботи після перебою
4	Прийнятий	Стихійне лихо/війна	0.5	0.1	0.05	Передача	Страхування, укладення угод з іншими компаніями для постачання умовах непередбачених ситуацій	Відновлення після припинення роботи

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
5	Виправданий	Скорочення бюджету	0.3	0.4	0.12	Зменшення	Оптимізація витрат, перегляд проектних завдань та обсягів робіт	Запобігання можливим затримкам у виконанні проекту
6	Виправданий	Непорозуміння між замовником і ПМ/ТЗ	0.3	0.4	0.12	-	Регулярні наради та звітності перед замовником	Вдосконалення процесу комунікації
7	Виправданий	Некомпетентність співробітника	0.2	0.7	0.14	-	Навчання та підтримка з боку більш кваліфікованих	Перерозподіл обов'язків або заміна співробітника

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
8	Виправданий	Конкурентний продукт з'явився першим	0.3	0.3	0.09	Зменшення	Прискорення розробки та впровадження продукту	Розробка стратегії маркетингу
9	Прийнятний	Низька швидкість Інтернет-з'єднання	0.3	0.1	0.03	Зменшення	Вибір провайдера з найвищою доступною швидкістю, оптимізація коду та зображень на сторінках додатку	Пошук альтернативних інтернет-провайдерів

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
10	Виправданий	Атаки з боку зловмисників (хакерські атаки)	0.2	0.3	0.06	Ухилення	Використання сучасних засобів безпеки, регулярне оновлення програмного забезпечення та аудит безпеки	Відновлення після кібератаки

ДОДАТОК В

Лістинг програмного коду основних модулів мобільного додатку

MainActivity.kt:

```
package com.example.younewsapp

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import com.example.younewsapp.databinding.ActivityMainBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.*

class MainActivity : AppCompatActivity() {
    lateinit var bindingClass: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindingClass = ActivityMainBinding.inflate(layoutInflater)
        setContentView(bindingClass.root)

        if (FirebaseAuth.getInstance().currentUser == null) {
            val intent = Intent(this, Register::class.java)
            startActivity(intent)
            finish()
        }

        if (savedInstanceState == null) {
            replaceFragment(ForYou())
        }

        loadUserPhoto()

        bindingClass.bottomNavigationView.setOnItemSelectedListener {
            when (it.itemId) {
                R.id.ForYou -> replaceFragment(ForYou())
                R.id.Categories -> replaceFragment(Categories())
                R.id.Favorite -> replaceFragment(Favorite())
                R.id.User -> replaceFragment(UserPage())
            }
            true
        }

        bindingClass.imgVSearch.setOnClickListener {
            val intent = Intent(this, SearchActivity::class.java)
            startActivity(intent)
        }
    }

    private fun replaceFragment(fragmet: Fragment) {
        val fragmentManager = supportFragmentManager
        val fragmentTransaction = fragmentManager.beginTransaction()
        fragmentTransaction.replace(R.id.frame_layout, fragmet)
        fragmentTransaction.commit()
    }

    private fun loadUserPhoto() {
```

```
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef = FirebaseDatabase.getInstance().getReference("Users").child(userId)

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val photoName = snapshot.child("photo").getValue(String::class.java)
                if (photoName != null) {
                    val resId = resources.getIdentifier(photoName, "drawable", packageName)
                    if (resId != 0) {
                        bindingClass.imgVUserLogo.setImageResource(resId)
                    }
                }
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(
                    this@MainActivity,
                    "Помилка завантаження фото користувача",
                    Toast.LENGTH_SHORT
                ).show()
            }
        })
    }
}
```

Admin.kt:

```
package com.example.younewsapp

import android.os.Bundle
import android.util.Patterns
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.younewsapp.databinding.ActivityAdminBinding
import com.google.firebase.database.*

class Admin : AppCompatActivity() {
    lateinit var bindingClass: ActivityAdminBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindingClass = ActivityAdminBinding.inflate(layoutInflater)
        setContentView(bindingClass.root)

        bindingClass.bAdd.setOnClickListener {
            addNews()
        }
    }

    private fun addNews() {
        val category = bindingClass.etCategory.text.toString().trim()
        val imageURL = bindingClass.etImageURL.text.toString().trim()
        val title = bindingClass.etTitle.text.toString().trim()
        val sourceName = bindingClass.etSourceName.text.toString().trim()
        val date = bindingClass.etDate.text.toString().trim()
        val url = bindingClass.etURL.text.toString().trim()

        if (category.isBlank() || !isValidCategory(category)) {
            Toast.makeText(this, "Такої категорії немає", Toast.LENGTH_SHORT).show()
            return
        }
        if (imageURL.isBlank() || !isValidImageURL(imageURL)) {
            Toast.makeText(this, "Некоректний URL зображення", Toast.LENGTH_SHORT).show()
```

```
            return
        }
        if (title.isBlank()) {
            Toast.makeText(this, "Заголовок не може бути порожнім", Toast.LENGTH_SHORT).show()
            return
        }
        if (sourceName.isBlank()) {
            Toast.makeText(this, "Ім'я джерела не може бути порожнім",
Toast.LENGTH_SHORT).show()
            return
        }
        if (date.isBlank() || !isValidDate(date)) {
            Toast.makeText(this, "Некоректний формат дати (DD-MM-YYYY)",
Toast.LENGTH_SHORT).show()
            return
        }
        if (url.isBlank() || !Patterns.WEB_URL.matcher(url).matches()) {
            Toast.makeText(this, "Некоректний URL новини", Toast.LENGTH_SHORT).show()
            return
        }

        val dbRef = FirebaseDatabase.getInstance().getReference("News")
        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                var maxId = 0
                for (newsSnapshot in snapshot.children) {
                    val news = newsSnapshot.getValue(News::class.java)
                    val id = news?.id?.toIntOrNull()
                    if (id != null && id > maxId) {
                        maxId = id
                    }
                }
                val newId = (maxId + 1).toString()

                val news = News(newId, category, imageURL, title, sourceName, date, url)
                dbRef.child(newId).setValue(news).addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        Toast.makeText(this@Admin, "Новину додано", Toast.LENGTH_SHORT).show()
                        finish()
                    } else {
                        Toast.makeText(this@Admin, "Помилка додавання новини",
Toast.LENGTH_SHORT)
                            .show()
                    }
                }
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(this@Admin, "Помилка: ${error.message}",
Toast.LENGTH_SHORT).show()
            }
        })
    }

    private fun isValidCategory(category: String): Boolean {
        val categories = listOf("Бізнес", "Кіно", "Медицина", "Політика", "Спорт",
"Технології")
        return categories.contains(category)
    }

    private fun isValidDate(date: String): Boolean {
        val datePattern = Regex("\\d{2}-\\d{2}-\\d{4}")
        return date.matches(datePattern)
    }
```

```
    private fun isValidImageURL(url: String): Boolean {
        return Patterns.WEB_URL.matcher(url).matches()
    }
}
```

Categories.kt:

```
package com.example.younewsapp

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.tabs.TabLayout
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class Categories : Fragment() {
    private lateinit var dbref: DatabaseReference
    private lateinit var newsRecyclerView: RecyclerView
    private lateinit var newsArrayList: ArrayList<News>

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_categories, container, false)

        newsRecyclerView = view.findViewById(R.id.rvNewsList)
        newsRecyclerView.layoutManager = LinearLayoutManager(requireContext())
        newsRecyclerView.setHasFixedSize(true)
        newsArrayList = arrayListOf<News>()
        getNewsData()

        val tabLayout = view.findViewById<TabLayout>(R.id.include)
        tabLayout.addOnTabSelectedListener(object : TabLayout.OnTabSelectedListener {
            override fun onTabSelected(tab: TabLayout.Tab) {
                when (tab.position) {
                    0 -> getNewsData()
                    1 -> getNewsDataByCategory("Бізнес")
                    2 -> getNewsDataByCategory("Кіно")
                    3 -> getNewsDataByCategory("Медицина")
                    4 -> getNewsDataByCategory("Політика")
                    5 -> getNewsDataByCategory("Спорт")
                    6 -> getNewsDataByCategory("Технології")
                }
            }
            override fun onTabUnselected(tab: TabLayout.Tab?) {}
            override fun onTabReselected(tab: TabLayout.Tab?) {}
        })

        return view
    }

    private fun getNewsData() {
```

```
        newsArrayList.clear()

        dbref = FirebaseDatabase.getInstance().getReference("News")
        dbref.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    for (newsSnapshot in snapshot.children) {
                        val news = newsSnapshot.getValue(News::class.java)
                        newsArrayList.add(news!!)
                    }
                    newsRecyclerView.adapter = MyAdapter(newsArrayList)
                }
            }
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT).show()
            }
        })
    }

    private fun getNewsDataByCategory(category: String) {
        newsArrayList.clear()

        dbref = FirebaseDatabase.getInstance().getReference("News")
        dbref.orderByChild("category").equalTo(category)
            .addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(snapshot: DataSnapshot) {
                    if (snapshot.exists()) {
                        for (newsSnapshot in snapshot.children) {
                            val news = newsSnapshot.getValue(News::class.java)
                            newsArrayList.add(news!!)
                        }
                        newsRecyclerView.adapter = MyAdapter(newsArrayList)
                    }
                }
                override fun onCancelled(error: DatabaseError) {
                    Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT).show()
                }
            })
    }
}
```

Favorite.kt:

```
package com.example.younewsapp

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class Favorite : Fragment() {
```

```
    private lateinit var favoriteRecyclerView: RecyclerView
    private lateinit var favoriteAdapter: MyAdapter
    private lateinit var favoriteNewsList: ArrayList<News>
    private lateinit var userId: String

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_favorite, container, false)

        userId = FirebaseAuth.getInstance().currentUser?.uid ?: ""
        favoriteRecyclerView = view.findViewById(R.id.rvNewsList)
        favoriteRecyclerView.layoutManager = LinearLayoutManager(requireContext())
        favoriteNewsList = ArrayList()
        favoriteAdapter = MyAdapter(favoriteNewsList)
        favoriteRecyclerView.adapter = favoriteAdapter

        getFavoriteNews()

        return view
    }

    private fun getFavoriteNews() {
        val databaseReference: DatabaseReference =

FirebaseDatabase.getInstance().getReference("Users").child(userId).child("favorite")

        databaseReference.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    favoriteNewsList.clear()
                    for (newsSnapshot in snapshot.children) {
                        val newsId = newsSnapshot.key ?: ""
                        FirebaseDatabase.getInstance().getReference("News").child(newsId)
                            .addListenerForSingleValueEvent(object : ValueEventListener {
                                override fun onDataChange(newsDataSnapshot: DataSnapshot) {
                                    val news = newsDataSnapshot.getValue(News::class.java)
                                    if (news != null) {
                                        favoriteNewsList.add(news)
                                        favoriteAdapter.notifyDataSetChanged()
                                    }
                                }
                                override fun onCancelled(error: DatabaseError) {
                                    Toast.makeText(
                                        requireContext(),
                                        "Помилка обробки даних",
                                        Toast.LENGTH_SHORT
                                    ).show()
                                }
                            })
                    }
                }
            }
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT)
                    .show()
            }
        })
    }
}
```

ForYou.kt:

```
package com.example.younewsapp

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.*
import kotlin.math.ceil

class ForYou : Fragment() {
    private lateinit var newsRecyclerView: RecyclerView
    private lateinit var newsAdapter: MyAdapter
    private lateinit var newsList: ArrayList<News>

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_foryou, container, false)
        newsRecyclerView = view.findViewById(R.id.rvNewsList)
        newsList = ArrayList()
        newsAdapter = MyAdapter(newsList)
        newsRecyclerView.adapter = newsAdapter
        newsRecyclerView.layoutManager = LinearLayoutManager(requireContext())

        loadRecommendedNews()

        return view
    }

    private fun loadRecommendedNews() {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =

FirebaseDatabase.getInstance().getReference("Users").child(userId).child("interests")

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    val interestsMap = snapshot.value as? Map<String, Long>
                    interestsMap?.let { interests ->
                        val sortedInterests = interests.filter { it.value > 0 }
                            .toList()
                            .sortedByDescending { (_, value) -> value }

                        if (sortedInterests.isEmpty()) return

                        val totalClicks = sortedInterests.sumByDouble { it.second.toDouble() }

                        val recommendedNews = mutableListOf<News>()
                        loadNewsForCategories(sortedInterests, totalClicks, recommendedNews) {
                            displayRecommendedNews(recommendedNews)
                        }
                    }
                }
            }
```

```
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT)
                    .show()
            }
        })
    }

    private fun loadNewsForCategory(category: String, limit: Int, callback: (List<News>) ->
Unit) {
        val newsRef = FirebaseDatabase.getInstance().getReference("News")
            .orderByChild("category").equalTo(category)

        newsRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(newsSnapshot: DataSnapshot) {
                val newsList = mutableListOf<News>()
                for (newsData in newsSnapshot.children) {
                    val news = newsData.getValue(News::class.java)
                    news?.let { newsList.add(it) }
                }
                callback(newsList.sortedByDescending { it.date }.take(limit))
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT)
                    .show()
            }
        })
    }

    private fun loadNewsForCategories(
        categories: List<Pair<String, Long>>,
        totalClicks: Double,
        recommendedNews: MutableList<News>,
        onComplete: () -> Unit
    ) {
        if (categories.isEmpty()) {
            onComplete()
            return
        }

        val category = categories[0]
        val percentage = category.second / totalClicks
        val limit = ceil(percentage * 5).toInt()

        loadNewsForCategory(category.first, limit) { newsList ->
            recommendedNews.addAll(newsList)
            loadNewsForCategories(categories.drop(1), totalClicks, recommendedNews,
onComplete)
        }
    }

    private fun displayRecommendedNews(recommendedNews: List<News>) {
        newsList.clear()
        newsList.addAll(recommendedNews.sortedByDescending { it.date })
        newsAdapter.notifyDataSetChanged()
    }
}
```

Login.kt:

```
package com.example.younewsapp
```

```
import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.younewsapp.databinding.ActivityLoginBinding
import com.google.firebase.auth.FirebaseAuth

class Login : AppCompatActivity() {
    lateinit var bindingClass: ActivityLoginBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindingClass = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(bindingClass.root)

        bindingClass.imgVBack.setOnClickListener {
            val intent = Intent(this, Register::class.java)
            startActivity(intent)
        }

        bindingClass.bLogin.setOnClickListener {
            val email = bindingClass.edLEmail.text.toString()
            val password = bindingClass.edLPassword.text.toString()

            if (email.isBlank() || password.isBlank()) {
                Toast.makeText(
                    applicationContext,
                    "Поля не можуть бути порожніми",
                    Toast.LENGTH_SHORT
                ).show()
            } else if (!isValidEmail(email)) {
                Toast.makeText(applicationContext, "Невірний формат пошти",
Toast.LENGTH_SHORT)
                    .show()
            } else {
                FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener(this) { task ->
                        if (task.isSuccessful) {
                            Toast.makeText(applicationContext, "Успішний вхід",
Toast.LENGTH_SHORT)
                                .show()
                            val intent = Intent(this, MainActivity::class.java)
                            startActivity(intent)
                            finish()
                        } else {
                            Toast.makeText(
                                applicationContext,
                                "Невірний email або пароль",
                                Toast.LENGTH_SHORT
                            ).show()
                        }
                    }
            }
        }
    }

    private fun isValidEmail(email: String): Boolean {
        return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
    }
}
```

MyAdapter.kt:

```
package com.example.younewsapp

import android.content.Context
import android.content.Intent
import android.util.Patterns
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.MutableData
import com.google.firebase.database.Transaction
import com.google.firebase.database.ValueEventListener

class MyAdapter(private val newsList : ArrayList<News>) :
RecyclerView.Adapter<MyAdapter.MyViewHolder>() {

    class MyViewHolder(itemView : View) : RecyclerView.ViewHolder(itemView) {
        val image: ImageView = itemView.findViewById(R.id.Image)
        val title: TextView = itemView.findViewById(R.id.tvTitle)
        val source: TextView = itemView.findViewById(R.id.tvSourceName)
        val data: TextView = itemView.findViewById(R.id.tvData)
        val star: ImageView = itemView.findViewById(R.id.imgVStar)
        val edit: ImageView = itemView.findViewById(R.id.imgVEdit)
        val delete: ImageView = itemView.findViewById(R.id.imgVDelete)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        var itemView = LayoutInflater.from(parent.context).inflate(R.layout.news_item, parent,
false)
        return MyViewHolder(itemView)
    }

    override fun getItemCount(): Int {
        return newsList.size
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val currentItem = newsList[position]

        checkAdminRights(holder)

        Glide.with(holder.itemView)
            .load(currentItem.imageURL)
            .into(holder.image)

        holder.title.text = currentItem.title
        holder.source.text = currentItem.sourceName
        holder.data.text = currentItem.date

        setStarState(holder.star, currentItem.id!!)

        holder.star.setOnClickListener {
```

```
            toggleFavorite(currentItem.id!!, holder.star)
            setStarState(holder.star, currentItem.id!!)
        }

        holder.edit.setOnClickListener {
            showEditDialog(holder.itemView.context, currentItem)
        }

        holder.delete.setOnClickListener {
            showDeleteConfirmationDialog(holder.itemView.context, currentItem.id!!)
        }

        holder.itemView.setOnClickListener {
            val intent = Intent(holder.itemView.context, UrlNewsActivity::class.java)
            intent.putExtra("URL", currentItem.url)
            holder.itemView.context.startActivity(intent)

            incrementCategoryCount(holder.itemView.context, currentItem.category!!)
        }
    }

    private fun checkAdminRights(holder: MyViewHolder) {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =
FirebaseDatabase.getInstance().getReference("Users").child(userId).child("admin")

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val isAdmin = snapshot.getValue(Boolean::class.java) ?: false
                if (isAdmin) {
                    holder.edit.visibility = View.VISIBLE
                    holder.delete.visibility = View.VISIBLE
                } else {
                    holder.edit.visibility = View.GONE
                    holder.delete.visibility = View.GONE
                }
            }
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(holder.itemView.context, "Помилка обробки даних",
Toast.LENGTH_SHORT).show()
            }
        })
    }

    private fun setStarState(star: ImageView, newsId: String) {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =
FirebaseDatabase.getInstance().getReference("Users").child(userId).child("favorite").child(new
sId)

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    star.setImageResource(R.drawable.favorite_active_icon)
                } else {
                    star.setImageResource(R.drawable.favorite_inactive_icon)
                }
            }
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(star.context, "Помилка отримання даних",
Toast.LENGTH_SHORT).show()
            }
        })
    }
```

```
    private fun toggleFavorite(newsId: String, star: ImageView) {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =
FirebaseDatabase.getInstance().getReference("Users").child(userId).child("favorite").child(new
sId)

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    dbRef.removeValue().addOnSuccessListener {
                        star.setImageResource(R.drawable.favorite_inactive_icon)
                    }
                } else {
                    dbRef.setValue(true).addOnSuccessListener {
                        star.setImageResource(R.drawable.favorite_active_icon)
                    }
                }
            }
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(star.context, "Помилка отримання даних",
Toast.LENGTH_SHORT).show()
            }
        })
    }

    private fun incrementCategoryCount(context: Context, category: String) {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =
FirebaseDatabase.getInstance().getReference("Users").child(userId).child("interests").child(ca
tegory)

        dbRef.runTransaction(object : Transaction.Handler {
            override fun doTransaction(mutableData: MutableData): Transaction.Result {
                val currentValue = mutableData.getValue(Int::class.java) ?: 0
                mutableData.value = currentValue + 1
                return Transaction.success(mutableData)
            }
            override fun onComplete(error: DatabaseError?, committed: Boolean, currentData:
DataSnapshot?) {
                if (error != null) {
                    Toast.makeText(context, "Помилка оновлення даних",
Toast.LENGTH_SHORT).show()
                }
            }
        })
    }

    private fun showEditDialog(context: Context, news: News) {
        val builder = AlertDialog.Builder(context)
        val inflater = LayoutInflater.from(context)
        val dialogView = inflater.inflate(R.layout.dialog_edit_news, null)
        builder.setView(dialogView)

        val etTitle = dialogView.findViewById<EditText>(R.id.etETitle)
        val etSourceName = dialogView.findViewById<EditText>(R.id.etESourceName)
        val etDate = dialogView.findViewById<EditText>(R.id.etEDate)
        val etImageURL = dialogView.findViewById<EditText>(R.id.etEImageURL)
        val etURL = dialogView.findViewById<EditText>(R.id.etEURL)

        etTitle.setText(news.title)
        etSourceName.setText(news.sourceName)
        etDate.setText(news.date)
        etImageURL.setText(news.imageURL)
```

```
        etURL.setText(news.url)

        builder.setPositiveButton("Зберегти") { _, _ ->
            val newTitle = etTitle.text.toString().trim()
            val newSourceName = etSourceName.text.toString().trim()
            val newDate = etDate.text.toString().trim()
            val newImageURL = etImageURL.text.toString().trim()
            val newURL = etURL.text.toString().trim()

            var isValid = true

            if (newTitle.isBlank()) {
                Toast.makeText(context, "Заголовок не може бути порожнім",
Toast.LENGTH_SHORT).show()
                isValid = false
            }
            if (newSourceName.isBlank()) {
                Toast.makeText(context, "Джерело не може бути порожнім",
Toast.LENGTH_SHORT).show()
                isValid = false
            }
            if (newDate.isBlank()) {
                Toast.makeText(context, "Дата не може бути порожньою",
Toast.LENGTH_SHORT).show()
                isValid = false
            } else if (!isValidDate(newDate)) {
                Toast.makeText(context, "Некоректний формат дати. Використовуйте YYYY-MM-DD",
Toast.LENGTH_SHORT).show()
                isValid = false
            }
            if (newImageURL.isBlank()) {
                Toast.makeText(context, "URL зображення не може бути порожнім",
Toast.LENGTH_SHORT).show()
                isValid = false
            } else if (newImageURL.isBlank() || !isValidImageURL(newImageURL)) {
                Toast.makeText(context, "Некоректний URL зображення",
Toast.LENGTH_SHORT).show()
                isValid = false
            }
            if (newURL.isBlank()) {
                Toast.makeText(context, "URL новини не може бути порожнім",
Toast.LENGTH_SHORT).show()
                isValid = false
            } else if (!Patterns.WEB_URL.matcher(newURL).matches()) {
                Toast.makeText(context, "Некоректний URL новини", Toast.LENGTH_SHORT).show()
                isValid = false
            }

            if (isValid) {
                updateNewsData(context, news.id!!, newTitle, newSourceName, newDate,
newImageURL, newURL)
            }
        }

        builder.setNegativeButton("Відміна") { dialog, _ ->
            dialog.dismiss()
        }

        builder.create().show()
    }

    private fun isValidDate(date: String): Boolean {
        val datePattern = Regex("\\d{2}-\\d{2}-\\d{4}")
        return date.matches(datePattern)
```

```
    }

    private fun isValidImageURL(url: String): Boolean {
        return Patterns.WEB_URL.matcher(url).matches()
    }

    private fun showDeleteConfirmationDialog(context: Context, newsId: String) {
        val builder = AlertDialog.Builder(context)
        builder.setMessage("Ви впевнені, що хочете видалити цю новину?")
            .setPositiveButton("Так") { _, _ ->
                deleteNews(context, newsId)
            }
            .setNegativeButton("Ні") { dialog, _ ->
                dialog.dismiss()
            }
        builder.create().show()
    }

    private fun updateNewsData(context: Context, newsId: String, newTitle: String,
newSourceName: String, newDate: String, newImageURL: String, newURL: String) {
        val dbRef = FirebaseDatabase.getInstance().getReference("News").child(newsId)

        val updates = hashMapOf<String, Any>(
            "title" to newTitle,
            "sourceName" to newSourceName,
            "date" to newDate,
            "imageURL" to newImageURL,
            "url" to newURL
        )

        dbRef.updateChildren(updates)
            .addOnSuccessListener {
                Toast.makeText(context, "Новину оновлено", Toast.LENGTH_SHORT).show()
            }
            .addOnFailureListener { e ->
                Toast.makeText(context, "Помилка: ${e.message}", Toast.LENGTH_SHORT).show()
            }
    }

    private fun deleteNews(context: Context, newsId: String) {
        val dbRef = FirebaseDatabase.getInstance().getReference("News").child(newsId)

        dbRef.removeValue()
            .addOnSuccessListener {
                Toast.makeText(context, "Новину видалено", Toast.LENGTH_SHORT).show()
            }
            .addOnFailureListener { e ->
                Toast.makeText(context, "Помилка: ${e.message}", Toast.LENGTH_SHORT).show()
            }
    }
}
```

News.kt:

```
package com.example.younewsapp

data class News(
    var id: String? = null,
    var category: String? = null,
    var imageURL: String? = null,
    var title: String? = null,
    var sourceName: String? = null,
    var date: String? = null,
```

```
    var url: String? = null
)
```

Register.kt:

```
package com.example.younewsapp

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.younewsapp.databinding.ActivityRegisterBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.*

class Register : AppCompatActivity() {
    lateinit var bindingClass: ActivityRegisterBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindingClass = ActivityRegisterBinding.inflate(layoutInflater)
        setContentView(bindingClass.root)

        bindingClass.tvLoginPage.setOnClickListener {
            startActivity(Intent(this, Login::class.java))
        }

        bindingClass.bRegister.setOnClickListener {
            val email = bindingClass.edREmail.text.toString()
            val password = bindingClass.edRPassword.text.toString()
            val userName = bindingClass.edRUserName.text.toString()

            if (email.isBlank() || password.isBlank() || userName.isBlank()) {
                Toast.makeText(this, "Поля не можуть бути порожніми",
Toast.LENGTH_SHORT).show()
            } else if (password.length < 6) {
                Toast.makeText(this, "Пароль має бути більше 6 символів",
Toast.LENGTH_SHORT).show()
            } else if (!isValidEmail(email)) {
                Toast.makeText(this, "Невірний формат пошти", Toast.LENGTH_SHORT).show()
            } else {
                checkEmailExists(email) { exists ->
                    if (exists) {
                        Toast.makeText(this, "Ця пошта вже зареєстрована", Toast.LENGTH_SHORT)
                            .show()
                    } else {
                        registerUser(email, password, userName)
                    }
                }
            }
        }
    }

    private fun checkEmailExists(email: String, callback: (Boolean) -> Unit) {
        val dbRef = FirebaseDatabase.getInstance().getReference("Users")
        dbRef.orderByChild("email").equalTo(email)
            .addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(snapshot: DataSnapshot) {
                    callback(snapshot.exists())
                }

                override fun onCancelled(error: DatabaseError) {
                    Toast.makeText(
```

```
                        this@Register,
                        "Помилка під час перевірки пошти: ${error.message}",
                        Toast.LENGTH_SHORT
                    ).show()
                    callback(false)
                }
            })
    }

    private fun registerUser(email: String, password: String, userName: String) {
        FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    val userId = FirebaseAuth.getInstance().currentUser?.uid ?: ""
                    val userInfo = HashMap<String, Any>().apply {
                        put("admin", false)
                        put("email", email)
                        put("userName", userName)
                        put(
                            "interests", mapOf(
                                "Бізнес" to 0,
                                "Кіно" to 0,
                                "Медицина" to 0,
                                "Політика" to 0,
                                "Спорт" to 0,
                                "Технології" to 0
                            )
                        )
                        put("photo", getRandomPhoto())
                    }

                    FirebaseDatabase.getInstance().getReference()
                        .child("Users")
                        .child(userId)
                        .setValue(userInfo)
                        .addOnCompleteListener { userTask ->
                            if (userTask.isSuccessful) {
                                startActivity(Intent(this, MainActivity::class.java))
                                finish()
                            } else {
                                Toast.makeText(
                                    this,
                                    "Помилка при збереженні інформації про користувача:
${userTask.exception?.message}",
                                    Toast.LENGTH_SHORT
                                ).show()
                            }
                        }
                } else {
                    Toast.makeText(
                        this,
                        "Помилка реєстрації: ${task.exception?.message}",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }
    }

    private fun getRandomPhoto(): String {
        val photos = listOf(
            "cat",
            "bear",
            "panda",
            "rabbit"
```

```
        )
        return photos.random()
    }

    private fun isValidEmail(email: String): Boolean {
        return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
    }
}
```

SearchActivity.kt:

```
package com.example.younewsapp

import android.content.Context
import android.os.Bundle
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.younewsapp.databinding.ActivitySearchBinding
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class SearchActivity : AppCompatActivity() {
    private lateinit var bindingClass: ActivitySearchBinding
    private lateinit var dbref: DatabaseReference
    private lateinit var newsArrayList: ArrayList<News>
    private lateinit var adapter: MyAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindingClass = ActivitySearchBinding.inflate(layoutInflater)
        setContentView(bindingClass.root)

        newsArrayList = ArrayList()
        adapter = MyAdapter(newsArrayList)
        bindingClass.rvNewsList.layoutManager = LinearLayoutManager(this)
        bindingClass.rvNewsList.adapter = adapter

        setupSearchView()

        bindingClass.svSearch.requestFocus()
        showKeyboard()
    }

    private fun setupSearchView() {
        bindingClass.svSearch.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
            override fun onQueryTextSubmit(query: String?): Boolean {
                if (query != null && query.isNotBlank()) {
                    searchNewsByTitle(query)
                    hideKeyboard()
                }
                return true
            }

            override fun onQueryTextChange(newText: String?): Boolean {
                return false
            }
        })
```

```
        bindingClass.svSearch.setIconifiedByDefault(false)
    }

    private fun searchNewsByTitle(title: String) {
        newsArrayList.clear()

        dbref = FirebaseDatabase.getInstance().getReference("News")
        dbref.orderByChild("title")
            .addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(snapshot: DataSnapshot) {
                    if (snapshot.exists()) {
                        for (newsSnapshot in snapshot.children) {
                            val news = newsSnapshot.getValue(News::class.java)
                            if (news != null && news.title != null && news.title!!.contains(
                                    title,
                                    ignoreCase = true
                                )
                            ) {
                                newsArrayList.add(news)
                            }
                        }
                        adapter.notifyDataSetChanged()
                    }
                }

                override fun onCancelled(error: DatabaseError) {
                    Toast.makeText(
                        this@SearchActivity,
                        "Помилка отримання даних",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            })
    }

    private fun showKeyboard() {
        val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
        imm.showSoftInput(bindingClass.svSearch, InputMethodManager.SHOW_IMPLICIT)
    }

    private fun hideKeyboard() {
        val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
        imm.hideSoftInputFromWindow(bindingClass.svSearch.windowToken, 0)
    }
}
```

UrlNewsActivity.kt:

```
package com.example.younewsapp

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.example.younewsapp.databinding.ActivityUrlNewsBinding

class UrlNewsActivity : AppCompatActivity() {
    private lateinit var binding: ActivityUrlNewsBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityUrlNewsBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
        val url = intent.getStringExtra("URL")
        binding.webView.loadUrl(url!!)
    }
}
```

UserPage.kt:

```
package com.example.younewsapp

import android.content.Intent
import android.graphics.Color
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import com.example.younewsapp.databinding.FragmentUserpageBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.github.mikephil.charting.data.PieData
import com.github.mikephil.charting.data.PieDataSet
import com.github.mikephil.charting.data.PieEntry
import com.github.mikephil.charting.formatter.PercentFormatter

class UserPage : Fragment() {
    private lateinit var bindingClass: FragmentUserpageBinding

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        bindingClass = FragmentUserpageBinding.inflate(inflater, container, false)
        val view = bindingClass.root

        val signOutButton = bindingClass.bExit
        signOutButton.setOnClickListener {
            FirebaseAuth.getInstance().signOut()
            val intent = Intent(requireContext(), Login::class.java)
            startActivity(intent)
            requireActivity().finish()
        }

        bindingClass.bAdminPanel.setOnClickListener {
            val intent = Intent(requireContext(), Admin::class.java)
            startActivity(intent)
        }

        bindingClass.pieChart.setNoDataText("")

        checkAdminRights()
        loadUserInterestsAndDetails()

        return view
    }

    private fun checkAdminRights() {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef =
FirebaseDatabase.getInstance().getReference("Users").child(userId).child("admin")
```

```
        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val isAdmin = snapshot.getValue(Boolean::class.java) ?: false
                if (isAdmin) {
                    bindingClass.bAdminPanel.visibility = View.VISIBLE
                } else {
                    bindingClass.bAdminPanel.visibility = View.GONE
                }
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT)
                    .show()
            }
        })
    }

    private fun loadUserInterestsAndDetails() {
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val dbRef = FirebaseDatabase.getInstance().getReference("Users").child(userId)

        dbRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                // Load user interests
                val interestsSnapshot = snapshot.child("interests")
                val interestsMap = interestsSnapshot.value as? Map<String, Long>
                interestsMap?.let {
                    val filteredInterestsMap = interestsMap.filter { it.value > 0 }
                    val total = filteredInterestsMap.values.sum().toFloat()
                    val entries = filteredInterestsMap.entries.map { entry ->
                        PieEntry(entry.value.toFloat() / total * 100, entry.key)
                    }

                    val dataSet = PieDataSet(entries, "Interests")
                    dataSet.colors = getCategoryColors()
                    dataSet.valueTextSize = 12f
                    dataSet.valueTextColor = Color.BLACK
                    dataSet.valueFormatter = PercentFormatter(bindingClass.pieChart)

                    val pieData = PieData(dataSet)
                    bindingClass.pieChart.data = pieData
                    bindingClass.pieChart.setUsePercentValues(true)
                    bindingClass.pieChart.description.isEnabled = false
                    bindingClass.pieChart.legend.isEnabled = false
                    bindingClass.pieChart.invalidate()
                }

                val userName = snapshot.child("userName").getValue(String::class.java)
                val photoName = snapshot.child("photo").getValue(String::class.java)
                bindingClass.tvName.text = userName
                if (photoName != null) {
                    val resId = resources.getIdentifier(photoName, "drawable",
requireContext().packageName)
                    if (resId != 0) {
                        bindingClass.imgVPhoto.setImageResource(resId)
                    }
                }
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(requireContext(), "Помилка отримання даних",
Toast.LENGTH_SHORT)
```

```
                    .show()
            }
        })
    }

    private fun getCategoryColors(): List<Int> {
        return listOf(
            Color.parseColor("#FFA726"), // Бізнес
            Color.parseColor("#42A5F5"), // Кіно
            Color.parseColor("#66BB6A"), // Медицина
            Color.parseColor("#EF5350"), // Політика
            Color.parseColor("#AB47BC"), // Спорт
            Color.parseColor("#FF7043")  // Технології
        )
    }
}
```

bottom_nav.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/ForYou"
        android:icon="@drawable/baseline_home_24"
        android:title="Для вас" />
    <item
        android:id="@+id/Categories"
        android:icon="@drawable/baseline_app_registration_24"
        android:title="Категорії" />
    <item
        android:id="@+id/Favorite"
        android:icon="@drawable/favorite_inactive_icon"
        android:title="Обране" />
    <item
        android:id="@+id/User"
        android:icon="@drawable/baseline_person_24"
        android:title="Користувач" />
</menu>
```

activity_admin.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Admin">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Додавання новин"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.17000002" />

<EditText
    android:id="@+id/etCategory"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:ems="10"
    android:hint="Категорія"
    android:inputType="text"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/textView"
    app:layout_constraintStart_toStartOf="@+id/textView"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<EditText
    android:id="@+id/etImageURL"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:hint="URL малюнка"
    android:inputType="text"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/etCategory"
    app:layout_constraintStart_toStartOf="@+id/etCategory"
    app:layout_constraintTop_toBottomOf="@+id/etCategory" />

<EditText
    android:id="@+id/etTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:hint="Заголовок"
    android:inputType="text"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/etImageURL"
    app:layout_constraintStart_toStartOf="@+id/etImageURL"
    app:layout_constraintTop_toBottomOf="@+id/etImageURL" />

<EditText
    android:id="@+id/etSourceName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:hint="Ім'я джерела"
    android:inputType="text"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/etTitle"
    app:layout_constraintStart_toStartOf="@+id/etTitle"
    app:layout_constraintTop_toBottomOf="@+id/etTitle" />

<EditText
    android:id="@+id/etDate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:hint="Дата (день-місяць-рік)"
    android:inputType="text"
```

```
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="@+id/etSourceName"
        app:layout_constraintStart_toStartOf="@+id/etSourceName"
        app:layout_constraintTop_toBottomOf="@+id/etSourceName" />

    <EditText
        android:id="@+id/etURL"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:hint="URL новини"
        android:inputType="text"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="@+id/etDate"
        app:layout_constraintStart_toStartOf="@+id/etDate"
        app:layout_constraintTop_toBottomOf="@+id/etDate" />

    <Button
        android:id="@+id/bAdd"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Додати"
        app:layout_constraintEnd_toEndOf="@+id/etURL"
        app:layout_constraintStart_toStartOf="@+id/etURL"
        app:layout_constraintTop_toBottomOf="@+id/etURL" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_login.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Login">

    <ImageView
        android:id="@+id/imgVBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:src="@drawable/baseline_arrow_back_24" />

    <TextView
        android:id="@+id/tvLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Вхід"
        android:textSize="34sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.32999998" />

    <EditText
        android:id="@+id/edLEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="@+id/tvLogin"
        app:layout_constraintStart_toStartOf="@+id/tvLogin"
        app:layout_constraintTop_toBottomOf="@+id/tvLogin" />

    <EditText
        android:id="@+id/edLPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:hint="Пароль"
        android:inputType="textPassword"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="@+id/edLEmail"
        app:layout_constraintStart_toStartOf="@+id/edLEmail"
        app:layout_constraintTop_toBottomOf="@+id/edLEmail" />

    <Button
        android:id="@+id/bLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Увійти"
        app:layout_constraintEnd_toEndOf="@+id/edLPassword"
        app:layout_constraintStart_toStartOf="@+id/edLPassword"
        app:layout_constraintTop_toBottomOf="@+id/edLPassword" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="45dp"
        android:gravity="center_vertical"
        android:orientation="horizontal"
        android:paddingStart="8dp"
        android:paddingTop="10dp"
        android:paddingEnd="8dp"
        android:paddingBottom="10dp"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:id="@+id/imgVSearch"
            android:layout_width="40dp"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            android:src="@drawable/baseline_search_24" />

        <ImageView
            android:id="@+id/imgVLogo"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:src="@drawable/logo" />

        <ImageView
            android:id="@+id/imgVUserLogo"
            android:layout_width="40dp"
            android:layout_height="match_parent"
            android:layout_gravity="end"
            android:src="@drawable/account" />
    </LinearLayout>

    <FrameLayout
        android:id="@+id/frame_layout"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@+id/bottomNavigationView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout"/>

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottomNavigationView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:menu="@menu/bottom_nav">

    </com.google.android.material.bottomnavigation.BottomNavigationView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_register.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Register">

    <TextView
        android:id="@+id/tvRegister"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="Реєстрація"
    android:textSize="34sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.32999998" />

<EditText
    android:id="@+id/edRUserName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:ems="10"
    android:hint="Ім'я"
    android:inputType="text"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/tvRegister"
    app:layout_constraintStart_toStartOf="@+id/tvRegister"
    app:layout_constraintTop_toBottomOf="@+id/tvRegister" />

<EditText
    android:id="@+id/edREmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/edRUserName"
    app:layout_constraintStart_toStartOf="@+id/edRUserName"
    app:layout_constraintTop_toBottomOf="@+id/edRUserName" />

<EditText
    android:id="@+id/edRPassword"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:hint="Пароль"
    android:inputType="textPassword"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="@+id/edREmail"
    app:layout_constraintStart_toStartOf="@+id/edREmail"
    app:layout_constraintTop_toBottomOf="@+id/edREmail" />

<Button
    android:id="@+id/bRegister"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="Зареєструватись"
    app:layout_constraintEnd_toEndOf="@+id/edRPassword"
    app:layout_constraintStart_toStartOf="@+id/edRPassword"
    app:layout_constraintTop_toBottomOf="@+id/edRPassword" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginBottom="32dp"
    android:gravity="center_horizontal"
    android:orientation="horizontal"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

        <TextView
            android:id="@+id/tvText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="В мене є акаунт!" />

        <TextView
            android:id="@+id/tvLoginPage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingStart="8dp"
            android:text="Вхід"
            android:textColor="#2196F3" />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_search.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SearchActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <SearchView
            android:id="@+id/svSearch"
            android:layout_width="match_parent"
            android:layout_height="50dp" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rvNewsList"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            tools:listitem="@layout/news_item" />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_url_news.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UrlNewsActivity"
    android:orientation="vertical">

    <WebView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

dialog_edit_news.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/etETitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Заголовок"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/etESourceName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Джерело"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/etEDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Дата"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/etEImageURL"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="URL зображення"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/etEURL"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="URL новини"
        android:textSize="20sp" />
</LinearLayout>
```

fragment_categories.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".Categories"
    android:orientation="vertical">

    <com.google.android.material.tabs.TabLayout
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:id="@+id/include"
        app:tabTextColor="@color/black"
        app:tabSelectedTextColor="@color/black"
        app:tabIndicatorColor="@color/black"
        app:tabIndicatorHeight="3.5dp"
        android:layout_marginTop="0dp"
        app:tabMode="scrollable"
        android:backgroundTint="@color/white">

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Усі"
            android:id="@+id/all">
        </com.google.android.material.tabs.TabItem>

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Бізнес"
            android:id="@+id/business">
        </com.google.android.material.tabs.TabItem>

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Кіно"
            android:id="@+id/movie">
        </com.google.android.material.tabs.TabItem>

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Медицина"
            android:id="@+id/medicine">
        </com.google.android.material.tabs.TabItem>

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Політика"
            android:id="@+id/politics">
        </com.google.android.material.tabs.TabItem>

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Спорт"
            android:id="@+id/sport">
        </com.google.android.material.tabs.TabItem>

        <com.google.android.material.tabs.TabItem
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Технології"
            android:id="@+id/technologies">
        </com.google.android.material.tabs.TabItem>

    </com.google.android.material.tabs.TabLayout>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvNewsList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/news_item" />

</LinearLayout>
```

fragment_favorite.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Favorite">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvNewsList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/news_item" />

</LinearLayout>
```

fragment_foryou.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ForYou">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvNewsList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/news_item" />

</LinearLayout>
```

fragment_userpage.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    tools:context=".UserPage"
    tools:ignore="MissingClass">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#E8E8E8"
        android:padding="16dp">

        <ImageView
            android:id="@+id/imgVPhoto"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:layout_alignParentStart="true"
            android:layout_marginStart="5dp"
            android:layout_marginEnd="16dp"
            android:elevation="4dp"
            android:scaleType="centerCrop"
            android:src="@drawable/account" />

        <TextView
            android:id="@+id/tvName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_marginEnd="16dp"
            android:layout_toEndOf="@id/imgVPhoto"
            android:text="Ім'я"
            android:textColor="@android:color/black"
            android:textSize="18sp" />

        <Button
            android:id="@+id/bExit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_centerVertical="true"
            android:text="Вийти з акаунту"
            android:textColor="@android:color/white" />

    </RelativeLayout>

    <Button
        android:id="@+id/bAdminPanel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Панель адміністратора"
        android:visibility="gone" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">

        <com.github.mikephil.charting.charts.PieChart
            android:id="@+id/pieChart"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:paddingTop="10dp"/>
    </RelativeLayout>

</LinearLayout>
```

news_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    app:cardCornerRadius="8dp"
    app:cardElevation="8dp">

    <ImageView
        android:id="@+id/imgVStar"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_gravity="end"
        android:elevation="1dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        app:srcCompat="@drawable/favorite_inactive_icon" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_gravity="center_vertical|end"
        android:elevation="1dp">

        <ImageView
            android:id="@+id/imgVEdit"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:elevation="1dp"
            android:paddingEnd="10dp"
            android:visibility="gone"
            app:srcCompat="@drawable/baseline_edit_24" />

        <Space
            android:layout_width="match_parent"
            android:layout_height="8dp" />

        <ImageView
            android:id="@+id/imgVDelete"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:elevation="1dp"
            android:paddingEnd="10dp"
            android:visibility="gone"
            app:srcCompat="@drawable/baseline_delete_outline_24" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="8dp">

        <ImageView
            android:id="@+id/Image"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:scaleType="centerCrop" />
```

```
        <TextView
            android:id="@+id/tvTitle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp"
            android:text="Основний текст"
            android:textSize="18sp"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/tvSourceName"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginEnd="4dp"
                android:layout_weight="1"
                android:text="Джерело"
                android:textSize="14sp" />

            <TextView
                android:id="@+id/tvData"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginEnd="16dp"
                android:text="Дата"
                android:textSize="14sp" />
        </LinearLayout>

    </LinearLayout>

</androidx.cardview.widget.CardView>
```