Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного додатку для продажу онлайн-курсів з використанням Java, Firebase Realtime Database та Android Studio

Виконав(ла): студент(ка) 4 курсу, групи СП-41
спеціальності 121
Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Містерман П. М.
(прізвище та ініціали)

Керівник (підпис) Петрик М. Р.
(прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М.
(прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р.
(прізвище та ініціали)

Рецензент (підпис) Дмитроца Л. П.
(прізвище та ініціали)

Тернопіль
2024

АНОТАЦІЯ

Розробка мобільного додатку для продажу онлайн-курсів з використанням Java, Firebase Realtime Database та Android Studio // Кваліфікаційна робота освітнього рівня «Бакалавр» // Містерман Петро Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-41 // Тернопіль, 2024 // С. - 93, рис. - 42, табл. - 0, додат. - 3, бібліогр.-.17.

Ключові слова: Android Studio, Firebase Realtime Database , Java, мобільний додаток, адміністратор, користувач, база даних, курс, головна сторінка, інтерфейсу користувача, аналіз вимог, проектування, розробка, тестування.

На початковому етапі було проведено комплексний аналіз предметної області для визначення вимог та викликів розробки мобільного додатку для продажу онлайн-курсів. На основі цього аналізу було розроблено детальні функціональні вимоги, якими керувався під час реалізації проекту.

Етап проектування та розробки був зосереджений на перетворенні вимог у функціональний та зручний мобільний додаток. Для забезпечення ясності були створені детальні проектні документи, включаючи діаграму класів. Додаток було створено з використанням сучасних фреймворків та найкращих практик, з акцентом на модульність, масштабованість та безпеку.

На завершальному етапі було проведено ретельне тестування для забезпечення надійності та продуктивності додатку. Це включало тестування інтерфейсу на зручність використання, тестування функціональності для перевірки можливостей і тестування навантаження для оцінки продуктивності. Результати підтвердили, що додаток відповідає вимогам, забезпечуючи надійну, ефективну та зручну роботу.

ANNOTATION

Development of a mobile application for the sale of online courses using Java, Firebase Realtime Database and Android Studio // Qualification work for the educational level "Bachelor" / Misterman Petro Mykhailovych // Ternopil National Technical University named after Ivan Puluj, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SP-41 // Ternopil, 2024 // Pages. - 93, figures. - 42, table - 0, addition - 3, literature. - 17.

Keywords: Android Studio, Firebase Realtime Database, Java, mobile application, administrator, user, database, course, home page, user interface, requirements analysis, design, development, testing.

At the initial stage, a comprehensive analysis of the subject area was carried out to determine the requirements and challenges of developing a mobile application for selling online courses. Based on this analysis, detailed functional requirements were developed to guide the project implementation.

The design and development phase focused on translating the requirements into a functional and user-friendly mobile application. Detailed design documents, including a class diagram, were created to ensure clarity. The application was built using modern frameworks and best practices, with a focus on modularity, scalability and security.

At the final stage, thorough testing was carried out to ensure the reliability and performance of the application. This included UI testing for usability, functionality testing to verify capabilities, and load testing to assess performance. The results confirmed that the application met the requirements, providing a reliable, efficient and user-friendly experience.

ЗМІСТ

ВСТУП

За останні роки ландшафт освіти і навчання зазнав значних змін завдяки поширенню цифрових платформ і зростаючій доступності інтернету. Попит на гнучкі та зручні навчальні рішення призвів до зростання кількості онлайн-курсів. Як наслідок, розробка мобільних додатків, призначених для продажу онлайн-курсів, стала важливим аспектом сучасних освітніх технологій, що задовольняє потреби як учнів, так і викладачів.

Перехід від традиційного навчання в класі до онлайн середовища був зумовлений потребою в більш доступних та адаптивних освітніх рішеннях. Традиційні методи навчання, часто обмежені географічними та часовими рамками, все частіше доповнюються або замінюються онлайн-курсами, які пропонують більшу доступність і гнучкість. Мобільні додатки, зокрема, відіграють важливу роль у цій трансформації, забезпечуючи портативний і цікавий засіб для надання освітнього контенту та управління ним.

Незважаючи на зростання популярності онлайн-курсів, залишаються значні проблеми, особливо у сфері розробки додатків для продажу цих курсів. Однією з головних проблем є відсутність синхронізації в режимі реального часу та ефективних функцій управління курсами в багатьох існуючих освітніх додатках. Ці обмеження можуть заважати користувачеві, ускладнюючи пошук і купівлю курсів, а провайдерам – ефективне управління пропозиціями курсів. Цей дипломний проєкт спрямований на усунення цих прогалин шляхом розробки мобільного додатку з використанням Java, бази даних Firebase Realtime Database та Android Studio, що забезпечує безперебійну та інтерактивну платформу для купівлі та управління онлайн-курсами.

Цілі цього дослідження є багатогранними. По-перше, воно спрямоване на розробку надійного мобільного додатку для продажу онлайн-курсів за допомогою Java та Android Studio, використовуючи можливості бази даних Firebase Realtime Database для синхронізації даних у режимі реального часу. По-друге, додаток

включатиме функції автентифікації користувачів та управління курсами, що забезпечить безпечне та організоване середовище як для покупців, так і для продавців. По-третє, проєкт оцінить продуктивність, зручність використання та надійність розробленого додатку за допомогою ретельного тестування.

Дане дослідження може покращити досвід купівлі та управління онлайн-курсами шляхом усунення ключових обмежень існуючих освітніх додатків. Використовуючи сучасні інструменти та методи розробки, запропонований додаток має на меті запропонувати ефективне рішення для продажу та управління онлайн-курсами, що принесе користь як учням, так і викладачам. Очікується, що інтеграція синхронізації даних у режимі реального часу та зручний функціонал управління курсами забезпечить більш динамічну та інтерактивну платформу для продажі онлайн-курсів.

Розробка мобільного додатку, який ефективно інтегрує синхронізацію даних у реальному часі та комплексні функції управління курсами, має на меті зробити свій внесок у розвиток освітніх технологій. Очікується, що запропоноване рішення забезпечить практичну і масштабовану модель для майбутніх освітніх додатків, демонструючи переваги використання сучасних фрейм ворків розробки і баз даних для створення ефективних інструментів навчання.

Основною метою цієї роботи є аналіз та розробка мобільного додатку для продажу онлайн -курсів завдяки ефективному використанню Java та СУБД Firebase Realtime Database в середовищі Android Studio. Додаток має на меті безпечну автентифікацію користувачів, ефективне управління пропозиціями курсів та зручний інтерфейс, що покращує загальний досвід купівлі. Ця кваліфікаційна робота спрямована на показ потенціалу мобільних додатків у сфері продажу онлайн курсів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз вимог до системи

У сучасну цифрову епоху різко зріс попит на гнучкі та доступні рішення для онлайн-навчання. Щоб задовольнити цей попит, необхідно розробити мобільний додаток для продажу IT-курсів з акцентом на зручність, ефективність та функціональність. Додаток повинен забезпечувати безперебійну роботу як для адміністраторів, так і для користувачів, гарантуючи, що всі необхідні операції, від перегляду курсу до його купівлі, будуть легкодоступними.

Авторизовані користувачі, в першу чергу адміністратори та зареєстровані користувачі, потребують доступу до ряду функцій, які підтримують управління курсами, управління користувачами та аналіз даних. Система повинна надавати наступні функціональні можливості для авторизованих користувачів:

1) Створення та зміна профілів користувачів (лише для ролі адміністратора):
   - Адміністратори повинні мати можливість створювати нові профілі користувачів, що включає введення даних користувача, таких як ім'я, номер телефону та пароль.
   - Адміністратори також повинні мати можливість змінювати профілі користувачів
2) Перегляд списку всіх існуючих профілів користувачів (тільки для ролі адміністратора):
   - Система повинна надавати адміністраторам вичерпний список усіх профілів користувачів, відображаючи необхідну інформацію для легкого управління.
3) Редагування особистої інформації (лише для адміністраторів):
   - Адміністратори повинні мати можливість редагувати особисту інформацію для будь-якого профілю користувача, включаючи такі атрибути, як ім'я, номер телефону та пароль.

4) Вхід до свого облікового запису:
   - Зареєстровані користувачі повинні мати змогу увійти до свого облікового запису, використовуючи безпечні методи автентифікації, що гарантує захист їхніх персональних даних.
5) Зручне управління категоріями:
   - Додаток повинен дозволяти користувачам ефективно керувати категоріями курсів, даючи їм змогу легко переглядати різні категорії онлайн-курсів.
6) Швидке оновлення списку:
   - Користувачі повинні мати змогу швидко оновлювати список курсів, щоб мати доступ до найактуальнішої інформації.
7) Попередній перегляд продукту:
   - Система повинна відображати інформацію про курс у компактній формі на міні-картках, включаючи фотографію, назву курсу, рівень і дату початку, щоб надати користувачам швидкий огляд.
8) Зручний перегляд атрибутів:
   - Користувачі додатку мають мати можливість легко переглядати атрибути курсу, щоб мати доступ до потрібної інформації без необхідності переходити між кількома екранами.
9) Перегляд сторінки курсу з детальною інформацією:
   - Додаток повинен надавати можливість переглянути сторінку курсу на якій вказана детальна інформація про курс.
10) Додавання курсу до кошика:
   - Користувачі додатку мають мати можливість додавати бажані курси до свого кошика, що полегшить процес покупки.
11) Перегляд курсів у кошику:
   - Система повинна дозволяти користувачам переглядати курси, які вони додали до свого кошика.

12) Отримання консультації:
    - Користувачі повинні мати можливість запросити консультацію для отримання додаткової інформації про курси, щоб вони могли приймати зважені рішення.
13) Перегляд аналітики (лише для адміністраторів):
    - Адміністратори повинні мати доступ до інструментів аналітики, які надають інформацію про активність користувачів, завантаження і сховище.

Неавторизовані користувачі, в першу чергу потенційні студенти, які ще не зареєструвалися або не ввійшли в систему, потребують доступу до базових функцій, які дозволять їм вивчити програму та здійснити процес реєстрації. Система повинна надавати наступні функціональні можливості для неавторизованих користувачів:

1) Автентифікація:
    - Потенційні користувачі повинні мати можливість увійти в систему, використовуючи облікові дані, щоб отримати доступ до додаткових функцій і персоналізованого контенту.
2) Реєстрація:
    - Нові користувачі повинні мати можливість зареєструвати обліковий запис, ввівши ім'я, номер телефону та пароль, щоб отримати доступ до повного спектру функцій програми.

Аналіз системних вимог висвітлює критичні функціональні можливості, які мобільний додаток повинен надавати для ефективного продажу онлайн-курсів. Забезпечуючи доступ до основних функцій як авторизованим, так і неавторизованим користувачам, додаток може запропонувати зручну та ефективну платформу для купівлі та управління онлайн-курсами. Детальні вимоги слугують основою для наступних етапів проектування та реалізації, спрямовуючи процес розробки на задоволення потреб користувачів та досягнення цілей проєкту.

1.2 Опис методології проектування та технологій системи

При розробці мобільного додатку для продажу онлайн-курсів вибір відповідної методології проектування та використання відповідних технологій мають вирішальне значення для забезпечення ефективності та функціональності. Обрані методології та технології формують процес розробки, впливають на дизайнерські рішення та визначають продуктивність додатку.

Була обрана методологія аналізу вимог і гнучкий процес розробки ПЗ ICONIX, що ґрунтується на варіантах використання. ICONIX надає більше вимог та проектної документації і прагне уникнути паралічу аналізу. Процес ICONIX використовує лише чотири діаграми на основі UML у чотири-етапному процесі, який перетворює текст прецеденту на робочий код.

Принциповою відмінністю ICONIX є використання аналізу надійності, методу подолання розриву між аналізом та проектуванням. Аналіз стійкості зменшує неоднозначність в описах варіантів використання, гарантуючи, що вони написані в контексті предметної області, що супроводжується моделями. Цей процес значно спрощує проектування, тестування та оцінку варіантів використання.

Процес ICONIX поділено на чотири етапи. На кожному етапі робота з попередньої віхи переглядається та оновлюється.

Етап 1: Огляд вимог. Перед початком процесу ICONIX необхідно провести певний аналіз вимог. На основі цього аналізу можна визначити варіанти використання, створити модель предметної області та створити кілька прототипів графічних інтерфейсів.

Етап 2: Попередній огляд проекту. Після визначення варіантів використання можна написати текст, що описує взаємодію користувача та системи. Виконується аналіз надійності, щоб знайти потенційні помилки у тексті варіанта використання, та відповідним чином оновлюється модель предметної області. Текст прецеденту важливий для визначення того, як користувачі будуть взаємодіяти з системою.

Вони також надають розробнику перевірити правильність результатів аналізу вимог.

Етап 3: Детальний аналіз проекту. На цьому етапі процесу ICONIX модель предметної області та текст прецеденту з етапу 2 використовуються для проектування системи, що будується. Діаграма класів створюється з моделі предметної області, а текст варіантів використання використовується створення діаграм послідовності.

Етап 4: Розгортання. Модульні тести написані для перевірки того, що система відповідатиме тексту варіанта використання та діаграм послідовності. Нарешті, код пишеться з використанням діаграм класів та послідовностей в якості настанови [1].

Ця методологія була обрана тому, що дозволить швидко та ефективно виконати задачі поставлені перед розробником проекту, йому не доведеться витратити більшу частину свого часу на формування діаграм, він зможе якісно описати усі вимоги та спроектувати систему згідно них, після чого протестувати її. Ця методологія не є найлегшою, проте знаходиться в золотій середині між швидкістю та ефективністю. Після того як проект пройде через усі чотири необхідні етапи, він буде завершеним та готовим до експлуатації.

Для побудови мобільного додатку було обрано Android Studio.

Android Studio – це інтегроване середовище розробки (IDE) для розробки мобільних додатків. Розроблена компанією Google, Android Studio надає повний набір інструментів для створення додатків для Android. Вона підтримує різні платформи, включаючи Windows, Mac і Linux, і побудована на IntelliJ IDEA, пропонуючи надійне і гнучке середовище для розробки.

Основні можливості Android Studio включають [2,3]:

1) Підтримка декількох платформ Android: Android Studio дозволяє розробляти для смартфонів, планшетів, Android TV, Android Wear тощо.
2) Інтеграція контролю версій: Підтримує системи контролю версій, такі як GIT, що забезпечує ефективне управління проектами для команд будь-якого розміру.

3) Безперервна інтеграція: Android Studio сприяє безперервній інтеграції, виявляючи проблеми на ранніх стадіях процесу розробки та забезпечуючи стабільну якість коду.
4) Моніторинг продуктивності: Інструменти для оцінки продуктивності, використання пам'яті та заряду акумулятора допомагають оптимізувати додаток для кращого користувацького досвіду.
5) Хмарна інтеграція: Сумісність з Google App Engine полегшує інтеграцію хмарних API та сервісів.

Для написання застосунку у Android Studio було обрано високорівневу мову програмування Java.

Java – це основна мова програмування, яка використовується для розробки додатків в Android Studio. Об'єктно-орієнтована мова, випущена компанією Sun Microsystems у 1995 році, Java не залежить від платформи і відома своєю надійністю та універсальністю. Java-додатки компілюються у байт-код, який інтерпретується віртуальною машиною Java (JVM), забезпечуючи крос-платформну сумісність.

Об'єктно-орієнтовані функції Java та великі бібліотеки роблять її придатною для розробки складних додатків, включаючи мобільні додатки. Її синтаксис, запозичений з мов C та C++, забезпечує звичне середовище для розробників, скорочуючи час навчання та підвищуючи ефективність розробки [4].

Для зберігання даних за стосунку було обрано базу даних Firebase Realtime Database.

Придбана компанією Google у 2014 році, Firebase пропонує набір послуг для розробки мобільних та веб-додатків, включаючи синхронізацію даних у реальному часі та хмарне зберігання.

Основні можливості Firebase Realtime Database включають [5,6]:

1) Синхронізація даних у реальному часі: Забезпечує безперебійну синхронізацію даних між клієнтами та хмарою, гарантуючи, що користувачі завжди мають доступ до найсвіжішої інформації.

2) Аутентифікація: Надає надійні варіанти автентифікації користувачів, включаючи соціальні мережі та автентифікацію за допомогою електронної пошти/пароля.
3) Хмарне сховище: Підтримує зберігання створеного користувачем контенту, наприклад, зображень і відео, у хмарному сховищі Google Cloud Storage, що забезпечує надійність і масштабованість.
4) Безпека: Пропонує правила безпеки на стороні сервера для захисту даних і контролю доступу на основі автентифікації користувача.

Методологія ICONIX у поєднанні з потужними можливостями Android Studio, Java та СУБД Firebase Realtime Database забезпечує міцну основу для розробки мобільного додатку для продажу онлайн-курсів. Цей підхід забезпечує добре структурований процес розробки, від початкового аналізу вимог до фінальної реалізації, в результаті чого створюється надійний, зручний та ефективний додаток. Обрані технології доповнюють методологію проектування, підвищуючи продуктивність, безпеку та масштабованість додатку, що в кінцевому підсумку забезпечує високоякісний продукт для кінцевих користувачів.

1.3 Проектування відношень між акторами і прецедентами

При розробці мобільного додатку для продажу IT-курсів розуміння взаємодії між різними акторами та системою має вирішальне значення для створення надійного та зручного для користувача додатку. У цьому розділі детально описано дизайн взаємозв'язків між основними акторами і варіантами використання (прецедентами), які описують їхню взаємодію з системою. Відображаючи ці взаємозв'язки, можна гарантувати, що додаток відповідає функціональним вимогам і забезпечує безперебійну роботу для всіх зацікавлених сторін.

В описаній системі виділено три актори:

1) Користувач – авторизувавшись, переглядає курси і шукає йому потрібні і додає їх в корзину якщо хоче його купити. Якщо в нього виникають питання чи проблеми, то він може запросити консультацію для отримання додаткової інформації про курси чи вирішення проблеми.
2) Адміністратор – слідкує за роботою бази даних, аналізує та відповідає на повідомлення користувачів. Приймає міри щодо вирішення помилок та покращення роботи системи.
3) База даних – зберігає та надає реєстраційні дані користувачів (дані надаються додатку задля автоматичної авторизації після реєстрації).

Користувачів у системі є два типи: адміністратор та користувач. Різниця у користуванні полягає у тому, що у адміністратора є доступ до бази даних.

На рисунку 1.1 зображено типовий приклад використання.

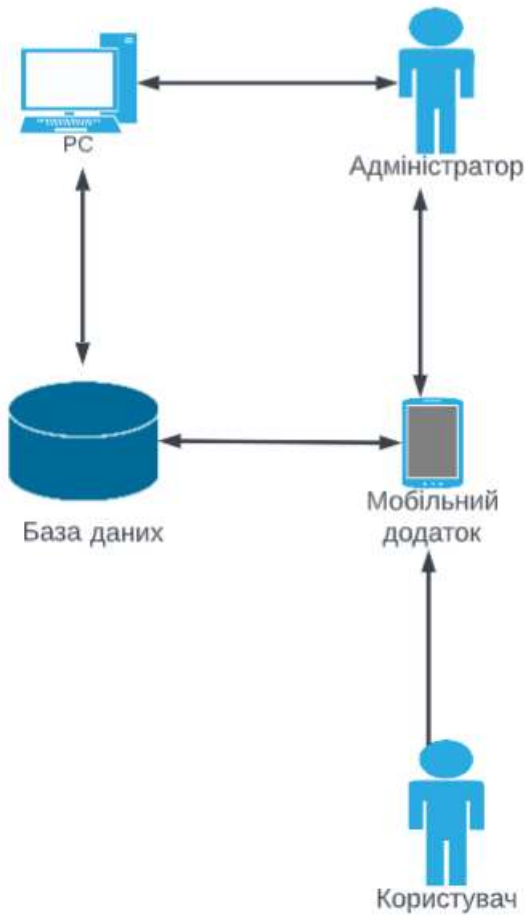


Рисунок 1.1 – Типовий приклад використання

Рисунок 1.1 дає чітке уявлення про взаємодію в системі, виділяючи ролі користувача та адміністратора. Користувач взаємодіє з системою через мобільний додаток, виконуючи такі завдання, як перегляд курсів, перегляд деталей, додавання курсів до кошика та запити на консультації. Адміністратор, з іншого боку, використовує ПК для управління базою даних, аналізу даних користувачів, надання консультацій та підтримки системи. Така структурована взаємодія забезпечує безперебійну роботу користувачів та ефективне управління системою з боку адміністраторів.

Для даних акторів системи можна виявити такі варіанти використання:

1) Реєстрація нового користувача в додатку

Користувач відкриває додаток. Якщо в нього немає акаунту, він нажимає на кнопку «Реєстрація» і вводить своє ім’я, номер телефону, пароль, нажимає на кнопку «Створити акаунт».

2) Вхід користувача в додаток

Користувач відкриває додаток. Якщо в нього є акаунт, він нажимає на кнопку «Вхід» і вводить номер телефону, пароль, нажимає на кнопку «Ввійти».

3) Вибір категорії

Користувач авторизувався у свій профіль вибирає потрібну йому категорію.

4) Перегляд інформації про курс

Користувач переглядає курси. Знайшовши потрібний йому курс він нажимає на нього. Відкривається сторінка курсу з інформацією про нього.

5) Додання курсу в корзину

Користувач переглядає курси. Знайшовши потрібний курс, переходить на сторінку курсу і нажимає кнопку «Додати в корзину».

6) Перегляд курсів в корзині

Користувач авторизувався у свій профіль нажимає на кнопку “Корзина”. Користувач бачить всі курси, які додав в корзину.

7) Отримання консультації

В користувача виникає питання щодо певного курсу. Він переходить на сторінку контактів, щоб дізнатися номер телефону чи адрес електронної пошти підтримки магазину. Користувач надсилає лист на електронну пошту підтримки магазину або телефонує в центр підтримки магазину для отримання консультації.

8) Перевірка відгуків адміністратором

Адміністратор заходить на електронну пошту підтримки магазину. Вивчивши причини відгуків, дає на них відповіді.

9) Створення та зміна профілів користувачів

Адміністратор ввійшовши в свій акаунт в Firebase, переходить в Realtime Database. Адміністратор створює профіль користувача вказуючи ім’я користувача, телефон користувача і пароль.

10) Перегляд списку усіх існуючих профілів користувачів

Адміністратор, авторизувавшись, переходить в Realtime Database іпереглядає профілі користувачів.

11) Редагування персональної інформації

Адміністратор ввійшовши в свій акаунт в Firebase, переходить в Realtime Database. Адміністратор вибирає користувача і редагує інформацію про нього.

12) Перегляд аналітики

Адміністратор ввійшовши в свій акаунт в Firebase, переходить на сторінку проекту мобільного додатку. Адміністратор переглядає аналітику мобільного додатку.

13) Записати у базу даних.

При надсиланні даних на запис користувачем, база даних отримує на це запит та виконує його.

14) Отримати/надати доступ до бази даних.

База даних отримує запит про надання доступу до даних та виконує його для користувача чи адміністратора.

Для кращого візуального сприйняття зв’язків між акторами та варіантами використанням, за допомогою мови UML можна створити діаграму варіантів

використання [7]. На діаграмі варіантів використання, на якій зображено актори та варіанти використання, можна побачити той функціонал, який необхідно реалізувати в системі. Діаграмі варіантів використання зображена на рисунку 1.2.

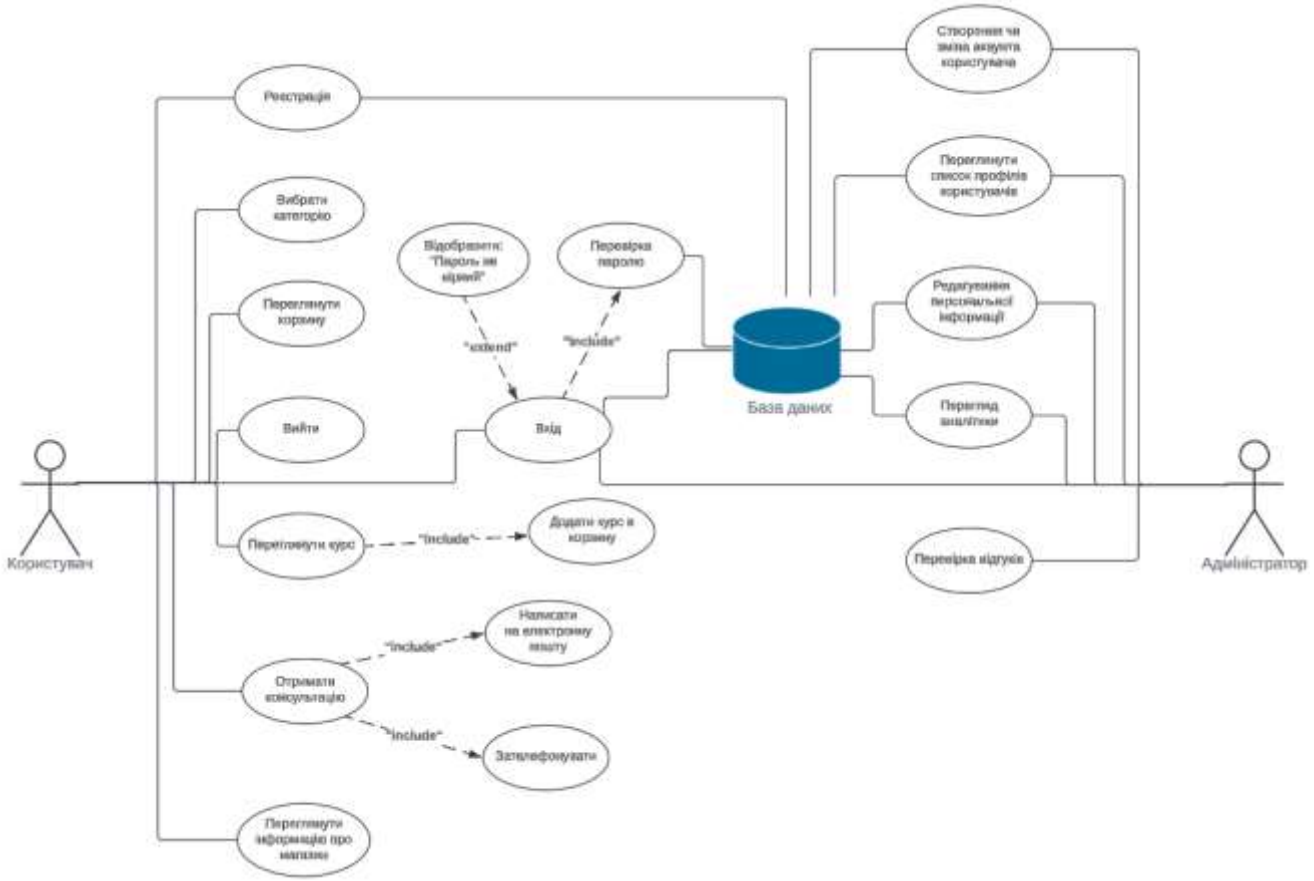


Рисунок 1.2 – Рисунок 1.2 – Діаграма варіантів використання

Розробка взаємозв'язків між акторами та прецедентами є критично важливим кроком у забезпеченні відповідності мобільного додатку для продажу онлайн-курсів потребам його користувачів та адміністраторів. Створити систему, яка буде інтуїтивно зрозумілою, ефективною та здатною забезпечити якісний користувацький досвід можна тільки чітко визначивши взаємодію та обов'язки кожного актора. Детальні сценарії використання слугують планом для розробки, керують процесом впровадження і забезпечують виконання всіх функціональних вимог і безперебійну роботу додатку.

1.4 Виявлення класів сутності

Визначення класів сутностей є важливим кроком у проектуванні та розробці будь-якої програмної системи. Ці класи представляють ключові елементи системи і визначають структуру та поведінку даних. У контексті мобільного додатку для продажу онлайн-курсів визначено кілька основних класів сутностей, які є важливими для функціональності системи.

Проаналізувавши опис даної системи, виявлено наступні сутності предметної області.

Сутність User – призначена для збереження інформації про користувача. Це одна з основних сутностей, що містить у собі основні дані для авторизації, які визначають тип сутності.

Сутність Course – призначена для збереження інформації про курс, яка буде відображатися на сторінці курсу. Кожен користувач повинен мати змогу переглянути всю інформацію про курс. Ця сутність має місити наступні особливості: назву курсу, короткий опис, дата початку і рівень складності.

Сутність Category – призначена для збереження інформації про категорії курсів, так як кожен курс має свою певну категорію.

Сутність Order – містить список курсів доданих в корзину.

Таким чином, визначені класи сутностей – Користувач, Курс, Категорія та Замовлення складають основу мобільного додатку для продажу курсів онлайн. Кожна сутність відіграє певну роль у забезпеченні безперебійної та ефективної роботи системи, надаючи користувачам безперебійний досвід від перегляду курсів до завершення покупки. Правильне визначення та реалізація цих елементів має вирішальне значення для загального успіху додатку.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення класів системи

Розробка мобільного додатку для продажу онлайн-курсів вимагає чіткого визначення класів системи. Маючи всі відомості про систему з першого розділу, можна приступити до проектування архітектури системи, а саме визначення цих класів. Вони формують основу додатку, містять як дані, так і функціональність, необхідну для забезпечення безперебійної роботи додатку. По цих класах встановлюється структура і поведінка системи, гарантуючи, що вона відповідає заданим вимогам і ефективно функціонує.

В даній системі всі основні класи наслідуються від AppCompatActivity. У розробці Android клас AppCompatActivity відіграє вирішальну роль, слугуючи базовим класом для дій в Android-додатках. AppCompatActivity є прямим підкласом класу FragmentActivity і непрямим підкласом класу Activity. Як частина бібліотеки AndroidX, AppCompatActivity надає зворотно-сумісні функції та сучасні вдосконалення, забезпечуючи безперебійну роботу додатків на різних версіях операційної системи Android. Розуміння AppCompatActivity є важливим для будь-якого розробника, який працює над додатками для Android, оскільки клас дозволяє розробникам використовувати сучасні функції Android на старих версіях Android, забезпечуючи узгоджений користувацький досвід на широкому спектрі пристроїв [8][9].

Визначено такі класи системи:

1) MainActivity – головний клас додатку з такими атрибутами:
   - joinButton – кнопка для реєстрації користувача.
   - loginButton – кнопка для входу користувача.
   - onCreate() – задає початкові параметри при ініціалізації активності.
2) RegisterActivity – клас, що реєструє нового користувача з такими атрибутами:
   - usernameInput – атрибут класу для запису імені користувача.

- phoneInput – атрибут класу для запису номера телефону.
- passwordInput – атрибут класу для запису пароля.
- registerBtn – кнопка реєстрації.
- onCreate().
- CreateAcount() – створює акаунт
- ValidatePhone() – перевірює чи введений номер використовується.

3) LoginActivity – клас для входу в акаунт з такими атрибутами:
   - phoneInput – атрибут класу для запису номера телефону.
   - passwordInput – атрибут класу для запису пароля.
   - loginBtn – кнопка входу.
   - onCreate().
   - loginUser – відповідає за вхід користувача в акаунт.
   - ValidateUser – звіряє введені дані з базою даних.
4) Users – клас користувачів з такими атрибутами:
   - name – атрибут для запису імені.
   - phone – атрибут для запису телефону
   - password – атрибут для запису паролю
   - Users() – конструктор.
   - GetName() – повертає ім’я користувача.
   - SetName() – ініціалізує ім’я користувача.
   - GetPhone() – повертає номер телефону.
   - SetPhone() – ініціалізує номер телефону.
   - GetPassword() – повертає пароль.
   - SetPassword() – ініціалізує пароль.
5) HomeActivity – клас домашньої сторінки додатку з такими атрибутами:
   - courseAdapter: CourseAdapter – об’єкт класу CourseAdapter.
   - categoryAdapter: CategoryAdapter – об’єкт класу CategoryAdapter.
   - courseList: List – список курсів певної категорії.
   - fullCourseList: List – список всіх курсів.
   - onCreate().

- openShopingCart() – відкрити сторінку корзини.
- openContactPage() – відкрити сторінку контактів.
- openAboutPage() – відкрити сторінку про мобільний додаток.
- showCoursesByCategory() – показати курси певної категорії.
- setCourseRecycler() – налаштування виводу курсів на головній сторінці.
- setCategoryRecycler() – налаштування виводу категорій на головній сторінці.

6) CourseAdapter – клас, який реалізовує список курсів з такими атрибутами:
   - context: Context – змінна для збереження сторінки, яка буде поміщатися в список.
   - corses: List – список курсів.
   - CorsesAdapter() – конструктор.
   - onCreateViewHolder() – створює і відображає список курсів.
   - CorsesViewHolder() – відображення елементів на сторінці курсів.
   - getItemCount() – повертає кількість курсів.
   - onBindViewHolder() – метод підстановки елементів на сторінку курсу.
7) CategoryAdapter – клас, який реалізовує список категорій з такими параметрами:
   - context: Context – змінна для збереження сторінки, яка буде поміщатися в список.
   - categories: List – список категорій.
   - CategoryAdapter() – конструктор.
   - onCreateViewHolder() – створює і відображає список категорій.
   - CategoryViewHolder() – відображення назви категорії.
   - onBindViewHolder() – метод підстановки тексту в категорії і виклику методу для виводу курсів певної категорії.
   - getItemCount() – повертає кількість категорій.
8) Course – клас курсу з такими атрибутами:
   - id – id курсу.
   - category – категорія курсу.

- img – зображення (прев’ю) курсу.
- title – назва курсу.
- date – початок курсу.
- level – рівень складності.
- color – колір заднього фону сторінку курсу.
- text – опис курсу.
- Products() – конструктор.
- GetId() – повертає id курсу.
- SetId() – ініціалізує id курсу.
- GetCategory() – повертає категорію курсу.
- SetCategory() – ініціалізує категорію курсу.
- GetImg() – повертає картинку курсу
- SetImg() – ініціалізує картинку курсу.
- GetTitle() – повертає назву курсу.
- SetTitle() – ініціалізує назву курсу.
- GetDate() – повертає початок курсу.
- SetDate() – ініціалізує початок курсу.
- GetLevel() – повертає рівень складності курсу.
- SetLevel() – ініціалізує рівень складності курсу.
- GetColor() – повертає колір заднього фону сторінку курсу.
- SetColor() – ініціалізує колір заднього фону сторінку курсу.

9) Category – клас категорій з такими атрибутами:
   - id – id категорії.
   - title – назва категорії.
   - GetId() – повертає id категорії.
   - SetId() – ініціалізує id категорії.
   - GetTitle() – повертає назву категорії.
   - SetTitle() – ініціалізація назви категорії.

10) Order – клас, в якому містяться .курси додані в корзину. Клас має такі методи:
    - items_id – список id курсів, які додано в корзину.
11) CoursePage – клас сторінки курсу з такими атрибутами:
    - onCreate().
    - addToCart() – додає курс в корзину.
    - openHomePage() – повертає на головну сторінку.
    - openContactsPage() – відкриває сторінку з контактами мобільного додатку.
    - openAboutPage() – відкриває сторінку з інформацією про мобільний додаток.
12) OrderPage – клас корзини з такими атрибутами:
    - onCreate().
    - openHomePage().
    - openContactsPage().
    - openAboutPage().
13) ContactsPage – клас контактів мобільного додатку з такими атрибутами:
    - onCreate().
    - openHomePage().
14) AboutPage – клас з описом про мобільний додаток. Клас містить такі атрибути:
    - onCreate().
    - openHomePage().

На основі усіх вищеописаних класів, за допомогою мови UML, побудовано діаграму класів, яка буде використана під час розробки застосунку [10]. Діаграма класів зображена на рисунку 2.1.

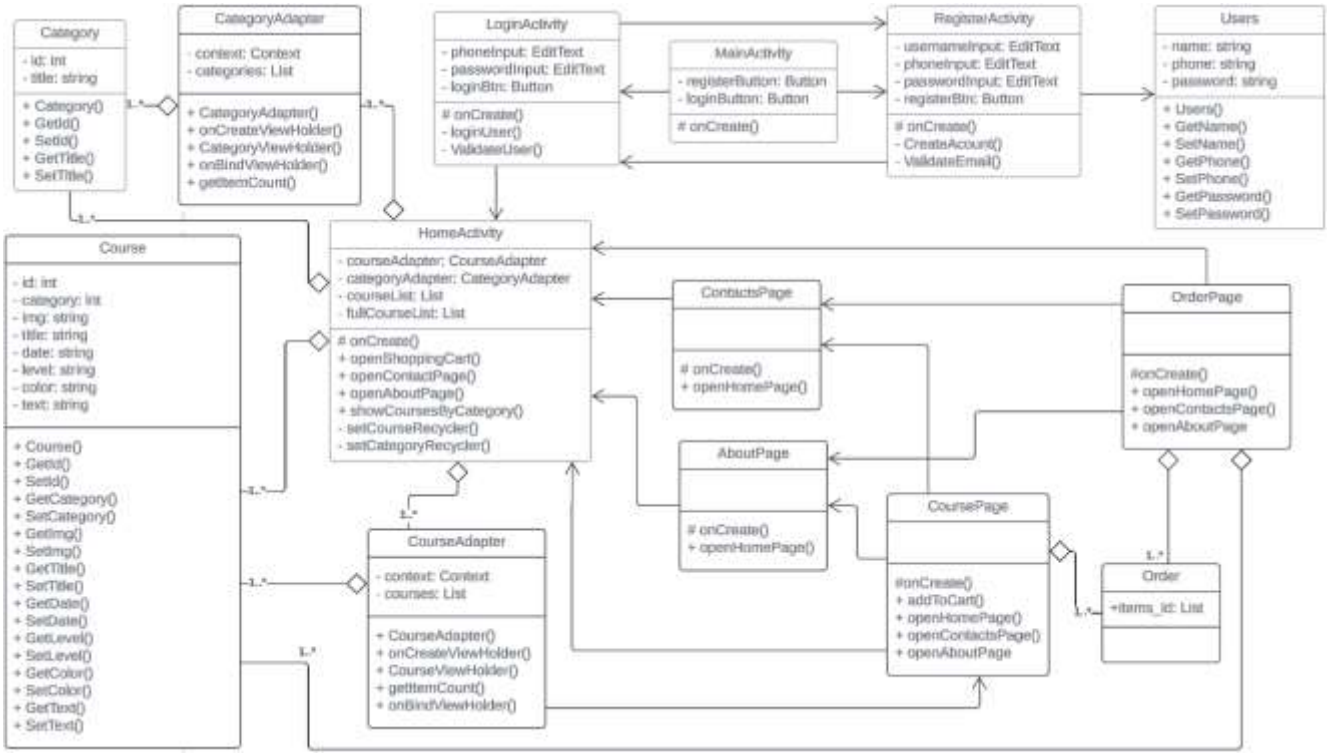


Рисунок 2.1 – Діаграма класів системи

Визначення класів системи є ключовим кроком у процесі проектування програмного забезпечення. Він надає план архітектури додатку, спрямовує процес розробки та гарантує, що всі необхідні функції будуть охоплені. Класи, визначені в цьому розділі, містять основні елементи мобільного додатку, від управління користувачами до управління курсами та обробки замовлень. Глибоко розуміючи та документуючи ці класи, можна закласти міцний фундамент для розробки надійного та ефективного мобільного додатку для продажу онлайн-курсів.

2.2 Розробка мобільного додатку

Розробка мобільного додатку для продажу IT-курсів передбачає реалізацію кількох ключових класів і методів. Ці класи виконують різні функції, починаючи від реєстрації користувача і входу в систему до перегляду курсів і додавання їх до

кошика. Кожен клас розроблений зі специфічними атрибутами та методами, щоб забезпечити безперебійну роботу додатку.

Клас MainActivity слугує точкою входу в Android-додаток, надаючи користувачеві можливість приєднатися або увійти в систему. Він визначає дві кнопки, joinButton і loginButton, які ініціалізуються в методі onCreate(). Цей метод також встановлює вид графічного інтерфейсу activity_main і налаштовує вставки вікна для відображення від краю до краю. Кнопкам joinButton і loginButton призначено події OnClickListener для обробки взаємодії з користувачем: натискання loginButton запускає LoginActivity, а натискання joinButton запускає RegisterActivity, таким чином перенаправляючи користувача на відповідні екрани для входу та реєстрації. Код методу onCreate() зображений на рисунку 2.2.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    joinButton = (Button) findViewById(R.id.main_join_btn);
    loginButton = (Button) findViewById(R.id.main_login_btn);

    loginButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent loginIntent = new Intent( packageContext: MainActivity.this, LoginActivity.class);
            startActivity(loginIntent);
        }
    });

    joinButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent registerIntent = new Intent( packageContext: MainActivity.this, RegisterActivity.class);
            startActivity(registerIntent);
        }
    });
}
```

Рисунок 2.2 – Код методу onCreate() класу MainActivity

Клас LoginActivity керує процесом входу для користувачів у додатку Android. Він визначає такі елементи інтерфейсу, як Button loginBtn, EditText phoneInput і passwordInput, а також використовує ProgressDialog loadingBar для відображення статусу завантаження. Метод onCreate() ініціалізує активність, встановлює макет і налаштовує віконні вставки для повноекранного інтерфейсу. Він налаштовує кнопку входу для запуску методу loginUser(), який перевіряє поля введення і показує відповідні повідомлення, якщо вони порожні. Код методу onCreate() зображений на рисунку 2.3.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_login);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    loginBtn = (Button)findViewById(R.id.login_btn);
    phoneInput = (EditText) findViewById(R.id.login_phone_input);
    passwordInput = (EditText) findViewById(R.id.login_password_input);
    loadingBar = new ProgressDialog( context: this);

    loginBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { loginUser(); }
    });
}
```

Рисунок 2.3 – Код методу onCreate() класу LoginActivity

Метод loginUser() також показує панель завантаження і викликає ValidateUser() для перевірки автентичності користувача у базі даних Firebase. Код методу loginUser() зображений на рисунку 2.4. ValidateUser() перевіряє, чи існує номер телефону користувача в базі даних, і перевіряє пароль. Якщо облікові дані правильні, він закриває панель завантаження і переходить до HomeActivity; в іншому випадку, він показує повідомлення про помилку або перенаправляє до RegisterActivity для створення облікового запису.

```
private void loginUser() {
    String phone = phoneInput.getText().toString();
    String password = passwordInput.getText().toString();

    if(TextUtils.isEmpty(phone))
    {
        Toast.makeText( context: this,  text: "Введіть номер", Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(password))
    {
        Toast.makeText( context: this,  text: "Введіть пароль", Toast.LENGTH_SHORT).show();
    }
    else
    {
        loadingBar.setTitle("Вхід в додаток");
        loadingBar.setMessage("Будь ласка зачекайте...");
        loadingBar.setCanceledOnTouchOutside(false);
        loadingBar.show();

        ValidateUser(phone, password);
    }
}
```

Рисунок 2.4 – Код методу loginUser()

Клас RegisterActivity відповідає за реєстрацію нових користувачів у додатку. Він містить три поля EditText для введення імені користувача, номера телефону та пароля, а також кнопку Button для надсилання реєстраційної форми. При створенні активності за допомогою методу onCreate() встановлюється вид її вмісту activity_register і налаштовуються вставки вікна для відображення від краю до краю. Код методу onCreate() зображений на рисунку 2.5. Коли натискається кнопка registerBtn, викликається метод CreateAccount() для перевірки полів введення. Код методу CreateAccount() зображений на рисунку 2.6. Якщо введені дані дійсні, відображається діалог ProgressDialog, а метод ValidatePhone() перевіряє, чи номер телефону вже зареєстровано в базі даних Firebase. Якщо номер не зареєстровано, дані користувача зберігаються, і відображається повідомлення про успіх; в іншому випадку відображається повідомлення про помилку. Користувач буде перенаправлений на сторінку LoginActivity.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_register);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    registerBtn = (Button) findViewById(R.id.register_btn);
    usernameInput = (EditText) findViewById(R.id.register_username_input);
    phoneInput = (EditText) findViewById(R.id.register_phone_input);
    passwordInput = (EditText) findViewById(R.id.register_password_input);
    loadingBar = new ProgressDialog( context: this);

    registerBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { CreateAccount(); }
    });
}
```

Рисунок 2.5 – Код методу onCreate() класу RegisterActivity

```
private void CreateAccount() {
    String username = usernameInput.getText().toString();
    String phone = phoneInput.getText().toString();
    String password = passwordInput.getText().toString();

    if(TextUtils.isEmpty(username))
    {
        Toast.makeText( context: this,  text: "Введіть ім'я", Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(phone))
    {
        Toast.makeText( context: this,  text: "Введіть номер", Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(password))
    {
        Toast.makeText( context: this,  text: "Введіть пароль", Toast.LENGTH_SHORT).show();
    }
    else
    {
        loadingBar.setTitle("Створення аккаунту");
        loadingBar.setMessage("Будь ласка зачекайте...");
        loadingBar.setCanceledOnTouchOutside(false);
        loadingBar.show();

        ValidatePhone(username, phone, password);
    }
}
```

Рисунок 2.6 – Код методу CreateAccount()

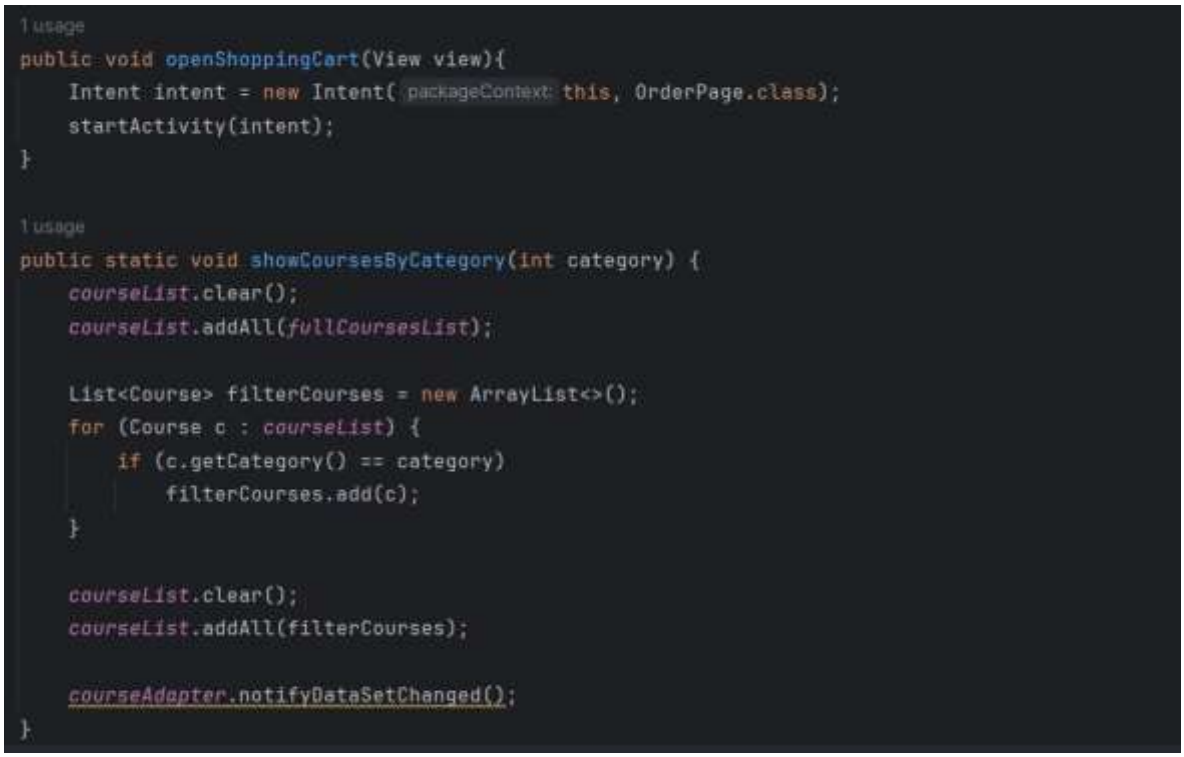
Клас HomeActivity слугує основним інтерфейсом для домашнього екрану програми, відображаючи список категорій та курсів. Він використовує компоненти RecyclerView для представлення категорій та курсів горизонтально. При створенні метод onCreate() встановлює вид вмісту activity_home, налаштовує вставки вікон для відображення від краю до краю та ініціалізує два адаптери RecyclerView, CategoryAdapter і CourseAdapter. Клас визначає статичний список курсів courseList і fullCoursesList і заповнює їх попередньо визначеними курсами. Код методу onCreate() зображений на рисунку 2.7.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_home);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    List<Category> categoryList = new ArrayList<>();
    categoryList.add(new Category( id: 1,  title: "Ігри"));
    categoryList.add(new Category( id: 2,  title: "Сайти"));
    categoryList.add(new Category( id: 3,  title: "Мови"));
    categoryList.add(new Category( id: 4,  title: "Інше"));

    setCategoryRecycler(categoryList);

    courseList.add(new Course( id: 1,  img: "java",  title: "Професія Java\nрозробник",  date: "1 серпня",  level: "початко
    courseList.add(new Course( id: 2,  img: "python",  title: "Професія Python\nрозробник",  date: "10 липня",  level: "поч
    courseList.add(new Course( id: 3,  img: "unity",  title: "Професія Unity\nрозробник",  date: "1 липня",  level: "початк
    courseList.add(new Course( id: 4,  img: "front_end",  title: "Професія Front-end\nрозробник",  date: "15 липня",  lev
    courseList.add(new Course( id: 5,  img: "back_end",  title: "Професія Back-end\nрозробник",  date: "20 липня",  level:
    courseList.add(new Course( id: 6,  img: "full_steck",  title: "Професія Full Stack\nрозробник",  date: "10 серпня",
    courseList.add(new Course( id: 7,  img: "node_js",  title: "Професія JS розробник\nрозробник",  date: "12 серпня",
    courseList.add(new Course( id: 8,  img: "csharp",  title: "Професія C# розробник\nрозробник",  date: "1 серпня",  lev

    fullCoursesList.addAll(courseList);

    setCourseRecycler(courseList);
}
```

Рисунок 2.7 – Код методу onCreate() класу HomeActivity

Клас містить методи для фільтрації та відображення курсів за категоріями showCoursesByCategory(), відкриття кошика для покупок openShoppingCart() та переходу на інші сторінки, такі як контакти та про курс openContactsPage(),

openAboutPage(). Код методів showCoursesByCategory() і openShoppingCart() зображені на рисунку 2.8. Методи setCategoryRecycler() і setCourseRecycler() налаштовують менеджери макетів і адаптери для компонентів RecyclerView, щоб керувати відображенням категорій і курсів відповідно. Код методів setCategoryRecycler() і setCourseRecycler() зображені на рисунку 2.9.



```
1 usage
public void openShoppingCart(View view){
    Intent intent = new Intent( packageContext: this, OrderPage.class);
    startActivity(intent);
}

1 usage
public static void showCoursesByCategory(int category) {
    courseList.clear();
    courseList.addAll(fullCoursesList);

    List<Course> filterCourses = new ArrayList<>();
    for (Course c : courseList) {
        if (c.getCategory() == category)
            filterCourses.add(c);
    }

    courseList.clear();
    courseList.addAll(filterCourses);

    courseAdapter.notifyDataSetChanged();
}
```

Рисунок 2.8 – Код методів showCoursesByCategory() і openShoppingCart()

```
private void setCourseRecycler(List<Course> courseList) {
    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager( context: this, RecyclerView.HORIZONTAL,  reverseLayout: false);

    courseRecycler = findViewById(R.id.courseRecycler);
    courseRecycler.setLayoutManager(layoutManager);

    courseAdapter = new CourseAdapter( context: this, courseList);
    courseRecycler.setAdapter(courseAdapter);
}

1 usage
private void setCategoryRecycler(List<Category> categoryList) {//налаштування виводу категорій горизонтально
    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager( context: this, RecyclerView.HORIZONTAL,  reverseLayout: false);

    categoryRecycler = findViewById(R.id.categoryRecycler);
    categoryRecycler.setLayoutManager(layoutManager);

    categoryAdapter = new CategoryAdapter( context: this, categoryList);
    categoryRecycler.setAdapter(categoryAdapter);

}
```

Рисунок 2.9 – Код методів setCategoryRecycler () і setCourseRecycler ()

Клас CoursePage слугує активністю для відображення детальної інформації про курс. У своєму методі onCreate() він налаштовує макет і обробляє віконні вставки, щоб забезпечити коректне відображення вмісту у системних панелях. Він отримує дані, передані з попередньої дії, щоб заповнити різні елементи інтерфейсу, такі як колір фону, зображення, заголовок, дата, рівень і текст, пов'язані з курсом. Код методу onCreate() зображений на рисунку 2.10.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_course_page);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    ConstraintLayout courseBg = findViewById(R.id.coursePageBg);
    ImageView courseImage =findViewById(R.id.coursePageImage);
    TextView courseTitle = findViewById(R.id.coursePageTitle);
    TextView courseDate = findViewById(R.id.coursePageDate);
    TextView courseLevel = findViewById(R.id.coursePageLevel);
    TextView courseText = findViewById(R.id.coursePageText);

    courseBg.setBackgroundColor(getIntent().getIntExtra( name: "courseBg",  defaultValue: 0));
    courseImage.setImageResource(getIntent().getIntExtra( name: "courseImage",  defaultValue: 0));
    courseTitle.setText(getIntent().getStringExtra( name: "courseTitle"));
    courseDate.setText(getIntent().getStringExtra( name: "courseDate"));
    courseLevel.setText(getIntent().getStringExtra( name: "courseLevel"));
    courseText.setText(getIntent().getStringExtra( name: "courseText"));

}
```

Рисунок 2.10 – Код методу onCreate() класу CoursePage

Крім того, він включає такі методи, як addToCart() для додавання поточного курсу до кошика, відображення вітального повідомлення в разі успіху, а також методи для переходу до інших активностей, таких як HomeActivity, ContactsPage і AboutPage. Цей клас визначає функціональність, пов'язану з відображенням деталей курсу та взаємодією з навігацією у додатку. Код методу addToCart() зображений на рисунку 2.11.

```
public void addToCart(View view){
    int item_id = getIntent().getIntExtra( name: "courseId",  defaultValue: 0);
    Order.items_id.add(item_id);
    Toast.makeText( context: this,  text: "Додано", Toast.LENGTH_LONG).show();
}
```

Рисунок 2.11 – Код методу addToCart()

Клас OrderPage призначений для відображення списку замовлених курсів. У своєму методі onCreate() він ініціалізує активність макетом сторінки activity_order_page і налаштовує віконні вставки для забезпечення належного відображення вмісту. Він заповнює ListView назвами курсів, доданих до замовлення, беручи їх зі статичного списку ідентифікаторів курсів, що зберігаються в класі Order, і порівнюючи їх з повним списком курсів, доступним з HomeActivity. Код методу onCreate() зображений на рисунку 2.12. Клас також надає методи для переходу до інших активностей, таких як HomeActivity, ContactsPage та AboutPage, що полегшує навігацію користувача в додатку.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_order_page);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    ListView orders_list = findViewById(R.id.orders_list);

    List<String> coursesTitle = new ArrayList<>();
    for(Course c: HomeActivity.fullCoursesList){
        if(Order.items_id.contains(c.getId()))
            coursesTitle.add(c.getTitle());
    }

    orders_list.setAdapter(new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, coursesTitle));
}
```

Рисунок 2.12 – Код методу onCreate() класу OrderPage

Клас AboutPage надає інформацію про програму. У методі onCreate() він ініціалізує активність за допомогою макета activity_about_page і налаштовує вигляд для правильної обробки віконних вставок, забезпечуючи коректне відображення

вмісту навколо системних панелей. Код методу onCreate() зображений на рисунку 2.13. Клас також містить метод переходу до HomeActivity, що дозволяє користувачам повернутися на головний екран програми. Цей клас передусім зосереджений на представленні статичного вмісту програми та забезпеченні плавного переходу на домашню сторінку.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_about_page);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
}
```

Рисунок 2.13 – Код методу onCreate() класу AboutPage

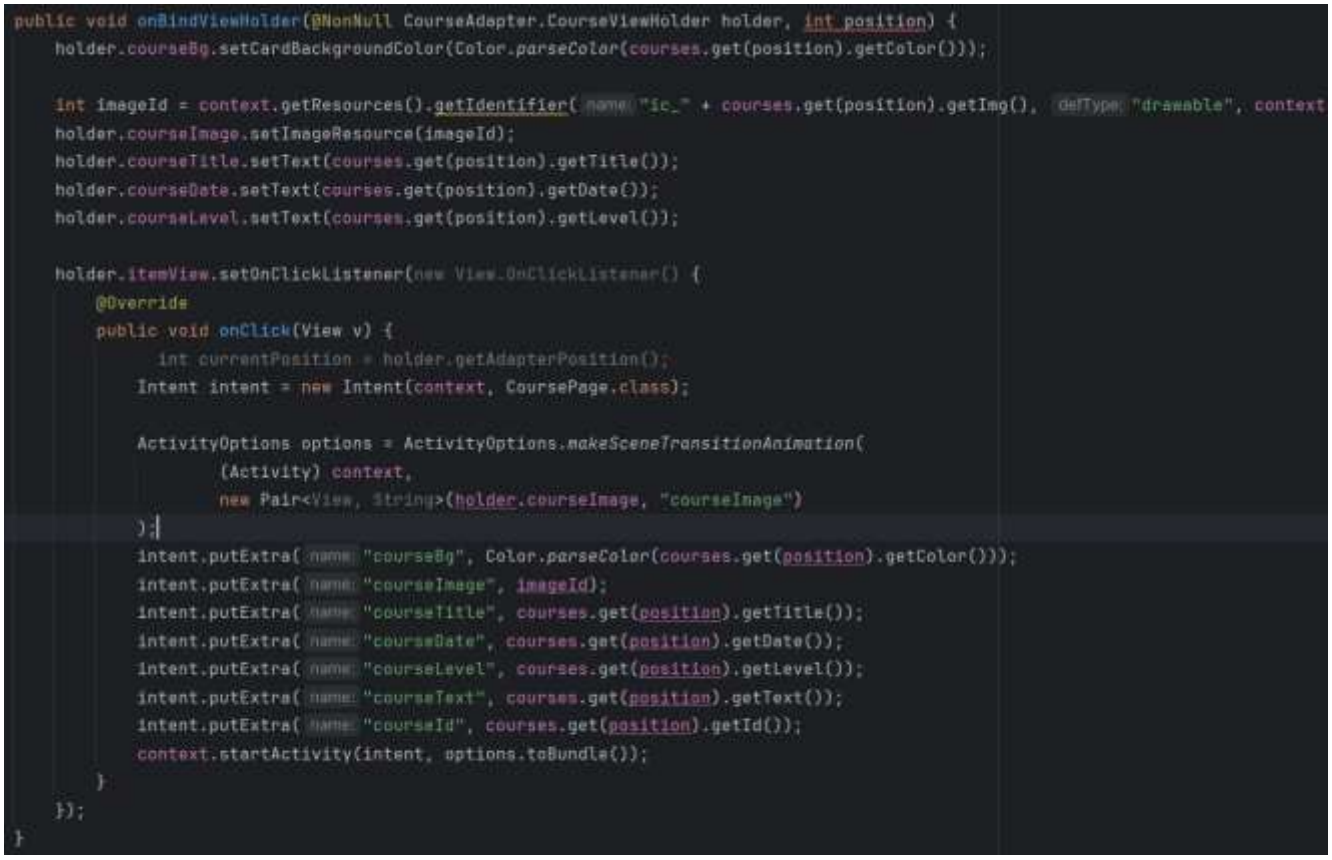
Клас ContactsPage відповідає за відображення контактної інформації в додатку. Метод onCreate() встановлює активність з макетом сторінки activity_contacts_page і налаштовує вигляд для обробки віконних вставок, гарантуючи, що вміст коректно відображається навколо системних панелей. Крім того, клас надає метод для переходу назад до HomeActivity, що полегшує користувачеві навігацію до домашнього екрана. Код методу onCreate() зображений на рисунку 2.14.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_contacts_page);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
}
```

Рисунок 2.14 – Код методу onCreate() класу ContactsPage

Цей клас передусім зосереджується на представленні контактних даних та підтримці узгодженої взаємодії з користувачем за допомогою належної навігації та налаштувань макета.

Клас CourseAdapter – це спеціальний адаптер для RecyclerView, який відображає список курсів у додатку Android. Цей адаптер містить контекст і список об'єктів курсу, а також використовує внутрішній клас CourseViewHolder для керування окремими представленнями елементів курсу. Метод onCreateViewHolder() розгортає макет course_item для створення вигляду для кожного курсу, а onBindViewHolder() встановлює дані для кожного подання, такі як колір фону курсу, зображення, заголовок, дата і рівень. Він також встановлює OnClickListener() для обробки подій кліку, які запускають активність CoursePage з переходами спільних елементів і передають відповідні деталі курсу за допомогою intent.putExtra(). Код методу onBindViewHolder() зображений на рисунку 2.15.



```
public void onBindViewHolder(@NonNull CourseAdapter.CourseViewHolder holder, int position) {
    holder.courseBg.setCardBackgroundColor(Color.parseColor(courses.get(position).getColor()));

    int imageId = context.getResources().getIdentifier( name: "ic_" + courses.get(position).getImg(),  defType: "drawable", context
    holder.courseImage.setImageResource(imageId);
    holder.courseTitle.setText(courses.get(position).getTitle());
    holder.courseDate.setText(courses.get(position).getDate());
    holder.courseLevel.setText(courses.get(position).getLevel());

    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            int currentPosition = holder.getAdapterPosition();
            Intent intent = new Intent(context, CoursePage.class);

            ActivityOptions options = ActivityOptions.makeSceneTransitionAnimation(
                    (Activity) context,
                    new Pair<View, String>(holder.courseImage, "courseImage")
            );
            intent.putExtra( name: "courseBg", Color.parseColor(courses.get(position).getColor()));
            intent.putExtra( name: "courseImage", imageId);
            intent.putExtra( name: "courseTitle", courses.get(position).getTitle());
            intent.putExtra( name: "courseDate", courses.get(position).getDate());
            intent.putExtra( name: "courseLevel", courses.get(position).getLevel());
            intent.putExtra( name: "courseText", courses.get(position).getText());
            intent.putExtra( name: "courseId", courses.get(position).getId());
            context.startActivity(intent, options.toBundle());
        }
    });
}
```

Рисунок 2.15 – Код методу onBindViewHolder()

Метод getItemCount() повертає загальну кількість курсів у списку. Клас CourseViewHolder визначає та ініціалізує елементи подання, які відображають інформацію про кожен курс. Код класу CourseViewHolder зображений на рисунку 2.16.

```
public static final class CourseViewHolder extends RecyclerView.ViewHolder{

    2 usages
    CardView courseBg;
    3 usages
    ImageView courseImage;
    2 usages
    TextView courseTitle, courseDate, courseLevel;
    1 usage
    public CourseViewHolder(@NonNull View itemView){
        super(itemView);

        courseBg = itemView.findViewById(R.id.courseBg);
        courseImage = itemView.findViewById(R.id.courseImage);
        courseTitle = itemView.findViewById(R.id.courseTitle);
        courseDate = itemView.findViewById(R.id.courseDate);
        courseLevel = itemView.findViewById(R.id.courseLevel);

    }
}
```

Рисунок 2.16 – Код класу CourseViewHolder

Клас CategoryAdapter – це спеціальний адаптер для RecyclerView, який відображає список категорій у додатку Android. Він містить контекст і список об'єктів категорії та використовує внутрішній клас CategoryViewHolder для керування окремими поданнями елементів категорії. Метод onCreateViewHolder() розгортає макет category_item для створення подання для кожної категорії, тоді як onBindViewHolder() встановлює заголовок категорії і призначає кожному елементу обробник подій кліку. Код методів onCreateViewHolder() і onBindViewHolder() зображені на рисунку 2.17 і 2.18 відповідно. Цей обробник кліків запускає метод showCoursesByCategory() у класі HomeActivity, який фільтрує та відображає курси на основі вибраної категорії. Метод getItemCount повертає загальну кількість

категорій у списку. Клас CategoryViewHolder визначає та ініціалізує елемент подання, який відображає заголовок категорії. Код класу CategoryViewHolder зображений на рисунку 2.19.

```
public CategoryViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View categoryItems = LayoutInflater.from(context).inflate(R.layout.category_item, parent, attachToRoot: false);
    return new CategoryViewHolder(categoryItems);
}
```

Рисунок 2.17 – Код методу onCreateViewHolder()

```
public void onBindViewHolder(@NonNull CategoryViewHolder holder, int position) {
    holder.categoryTitle.setText(categories.get(position).getTitle());
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            HomeActivity.showCoursesByCategory(categories.get(position).getId());
        }
    });
}
```

Рисунок 2.18 – Код методу onBindViewHolder()

```
public static final class CategoryViewHolder extends RecyclerView.ViewHolder {

    2 usages
    TextView categoryTitle;

    1 usage
    public CategoryViewHolder(@NonNull View itemView) {
        super(itemView);

        categoryTitle = itemView.findViewById(R.id.categoryTitle);
    }
}
```

Рисунок 2.19 – Код класу CategoryAdapter

Отже, розробка мобільного додатку для продажу онлайн-курсів передбачає створення та інтеграцію декількох класів, кожен з яких виконує певні функції. Від

автентифікації користувачів до управління курсами, ці класи та їхні відповідні методи формують основу додатку, забезпечуючи його функціональність, зручність та ефективність.

Дизайн цих класів підкреслює модульність і можливість багаторазового використання, при цьому кожен клас містить певну функціональність, яка сприяє загальній архітектурі додатку. Такий модульний підхід не лише спрощує розробку та підтримку, але й забезпечує можливість масштабування в майбутньому. Наприклад, додавання нових функцій або модифікація існуючих може бути здійснена з мінімальним впливом на інші частини програми.

На додаток до цих основних класів, додаток використовує сучасні практики розробки Android, такі як Edge-to-Edge UI дизайн, який забезпечує безшовний та захоплюючий користувальницький інтерфейс. Інтеграція Firebase для управління базами даних забезпечує надійне внутрішнє рішення, що дозволяє синхронізувати дані в режимі реального часу та безпечно зберігати їх.

Загалом, мобільний додаток для продажу онлайн-курсів є прикладом комплексного і добре продуманого процесу розробки. Кожен клас і метод ретельно розроблений для виконання певної ролі, що сприяє створенню цілісного та ефективного додатку. Результатом є орієнтована на користувача платформа, яка не тільки задовольняє потреби учнів, а й створює основу для постійного вдосконалення та розширення. Увага до деталей як на етапі планування, так і на етапі кодування підкреслює важливість структурованого підходу в розробці мобільних додатків, що в кінцевому підсумку забезпечує високоякісний продукт, який виділяється на конкурентному ринку онлайн-освіти.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Тестування інтерфейсу

Тестування інтерфейсу – це важливий аспект тестування програмного забезпечення, який гарантує, що користувацький інтерфейс (UI) додатку відповідає найвищим стандартам зручності та функціональності. Для онлайн-сервісу з продажу курсів тестування інтерфейсу фокусується на кількох ключових характеристиках: чіткість, виразність, звичність, релевантність, послідовність, естетичність, продуктивність і "прощення". Кожна з цих характеристик відіграє життєво важливу роль у забезпеченні безперебійного та приємного користувацького досвіду [11]. Інтерфейс домашньої сторінки мобільного додатку зображено на рисунку 3.1.

Рисунок 3.1 – Домашня сторінка застосунку

Щоб досягти ясності, інтерфейс був розроблений таким чином, щоб бути простим і зручним для навігації. Користувачі можуть швидко зрозуміти, що робить кожна частина інтерфейсу, не потребуючи додаткових інструкцій. У головному меню на видному місці відображаються всі функціональні можливості, доступні в додатку, логічно розташовані для полегшення швидкого доступу. Чітке маркування кнопок і елементів керування гарантує, що користувачі можуть передбачити результат своїх дій, зменшуючи ймовірність помилок і підвищуючи загальну ефективність.

Виразність інтерфейсу гарантує, що користувачі можуть виконувати завдання ефективно та результативно. Елементи дизайну чітко відображають про своє призначення, дозволяючи користувачам інтуїтивно взаємодіяти з додатком. Значки, кнопки та інші інтерактивні елементи розроблені з використанням знайомих символів і термінології, що сприяє швидкому розпізнаванню і розумінню. Ця виразність має вирішальне значення для того, щоб провести користувачів через різні функції та можливості додатку без зайвої складності.

Звичність досягається завдяки дотриманню стандартних принципів дизайну, до яких звикли користувачі. Інтерфейс запозичує загальні патерни, які можна побачити в інших успішних додатках, гарантуючи, що користувачам не потрібно вивчати нові способи взаємодії. Таке відчуття знайомства допомагає скоротити час навчання і робить додаток доступним для ширшої аудиторії, в тому числі для тих, хто не є технічно грамотним.

Релевантність інтерфейсу гарантує, що кожен елемент слугує певній меті та сприяє досягненню цілей користувача. Сторонні елементи зводяться до мінімуму, щоб уникнути безладу, дозволяючи користувачам зосередитися на поставленому завданні. Релевантність гарантує, що користувачі не будуть перевантажені інформацією або опціями, що робить інтерфейс більш зручним та ефективним.

Послідовність в інтерфейсі підтримує єдиний вигляд і стиль в усьому додатку. Ця узгодженість проявляється в дизайні кнопок, шрифтів, кольорів і макетів, які відповідають єдиному стилю. Послідовні елементи дизайну

допомагають користувачам створити ментальну модель інтерфейсу, роблячи навігацію більш інтуїтивно зрозумілою і зменшуючи потенційну плутанину.

Естетика також є важливим компонентом інтерфейсу, що сприяє приємному та цікавому користувацькому досвіду. Візуальний дизайн додатку чистий і сучасний, з акцентом на простоті та елегантності. Естетично привабливі інтерфейси можуть підвищити задоволеність користувачів і заохотити їх до подальшого використання програми.

Продуктивність – ще один важливий аспект тестування інтерфейсу. Інтерфейс повинен бути швидким, гарантуючи, що користувачі не відчуватимуть затримок. Швидкий час завантаження та плавні переходи між різними частинами програми є важливими для підтримки залученості та задоволеності користувачів.

"Прощення" в дизайні інтерфейсу означає здатність програми делікатно обробляти помилки користувача. Інтерфейс забезпечує чіткий зворотній зв'язок і можливості для відновлення, коли користувачі роблять помилки. Наприклад, діалоги підтвердження для критично важливих дій, опції скасування та корисні повідомлення про помилки – все це сприяє створенню інтерфейсу, що пробачає. Ця характеристика допомагає зменшити розчарування користувачів і підтримує більш поблажливий і пристосований до них користувацький досвід.

Під час розробки цього проекту значну увагу було приділено чіткості та зручності для забезпечення безперешкодного використання будь-яким користувачем. У головному меню користувачу представлені усі можливі функції, які він може виконати у застосунку. Проте усі елементи розставлені інтуїтивно та зрозуміло для кожного. Підписи на кнопках відповідають дійсності та виконують функції, задані у назві. У користувача не може виникнути плутанини, адже усе стоїть на своїх місцях та виконує свої задачі.

На завершення, тестування інтерфейсу для продажу онлайн-курсів було ретельним, щоб забезпечити відповідність інтерфейсу найвищим стандартам чіткості, виразності, знайомства, релевантності, послідовності, естетики, продуктивності та прощення помилок. Зосередившись на цих характеристиках, додаток забезпечує безперебійний, інтуїтивно зрозумілий і приємний

користувацький досвід, дозволяючи користувачам взаємодіяти з додатком без особливих зусиль і ефективно.

3.2 Тестування функціональності

Функціональне тестування – це критичний етап у процесі тестування програмного забезпечення, який фокусується на перевірці того, що додаток працює відповідно до заданих вимог і точно виконує свої функції. Цей тип тестування гарантує, що всі функції та можливості програми працюють правильно і надійно за різних умов.

1) Реєстрація нового облікового запису

Кроки виконання:

- завантажити мобільний додаток на свій телефон;
- відкрити мобільний додаток
- на початковій сторінці, яка зображена на рисунку 3.2, нажати на кнопку «Реєстрація»;
- у вікні реєстрації, яка зображена на рисунку 3.3, ввести ім’я, номер телефону і пароль;
- нажати на кнопку «Реєстрація».

Очікуваний результат: Новий обліковий запис буде успішно створено, користувач зможе виконувати усі передбачені функції та входити у свій обліковий запис. Також, система сповістить користувача про успішну реєстрацію і пере направить користувача на вікно входу. Це зображено на рисунку 3.4.

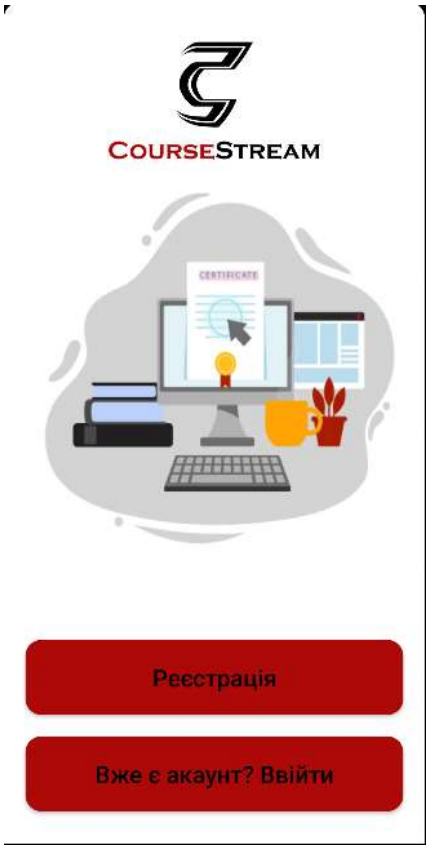


Рисунок 3.2 – Початкова сторінка застосунку

Рисунок 3.3 – Вікно реєстрації

Рисунок 3.4 – Вікно входу

2) Вхід в обліковий запис

Передумови: додаток встановлено і користувач зареєструвався.

Кроки виконання:

- на початковій сторінці (див. рисунок 3.2), нажати на кнопку «Вже є акаунт? Ввійти»;
- у вікні входу (див. рисунок 3.3) ввести номер телефону і пароль вказані при реєстрації;
- нажати на кнопку «Ввійти».

Очікуваний результат: Користувач успішно ввійде у свій обліковий запис. Також, система користувача вас про успішний вхід і перенаправить користувача на домашню сторінку. Це зображено на рисунку 3.5.

Рисунок 3.5 – Домашня сторінка застосунку

3) Введення уже існуючого в базі даних номера телефону при реєстрації

Передумови: додаток встановлено на телефоні.

Кроки виконання:

- відкрити мобільний додаток
- на початковій сторінці (див. рисунок 3.2) нажати на кнопку «Реєстрація»;
- у вікні реєстрації (див. рисунок 3.3) ввести ім’я, уже існуючий в базі даних номер телефону і пароль;
- нажати на кнопку «Реєстрація».

Очікуваний результат: система сповістить про те, що даний номер телефону уже використовується і перенаправить вас на сторінку входу. Це зображено на рисунку 3.6.

Рисунок 3.6 – Сповіщення, що даний номер уже використовується

4) Введення неправильного паролю при вході

Передумови: додаток встановлено на телефоні.

Кроки виконання:

- на початковій сторінці (див. рисунок 3.2), нажати на кнопку «Вже є акаунт? Ввійти»;
- у вікні входу (див. рисунок 3.3) ввести номер телефону і неправильний пароль;
- нажати на кнопку «Ввійти».

Очікуваний результат: система сповістить про неправильно введений пароль. Це зображено на рисунку 3.7.

Рисунок 3.7 – Сповіщення, що пароль не вірний

5) Введення не зареєстрованого в базі даних номера телефону при вході в акаунт

Передумови: додаток встановлено на телефоні.

Кроки виконання:

- на початковій сторінці (див. рисунок 3.2), нажати на кнопку «Вже є акаунт? Ввійти»;
- у вікні входу (див. рисунок 3.3) ввести номер телефону, який не зареєстрований в базі даних і пароль;
- нажати на кнопку «Ввійти».

Очікуваний результат: система сповістить користувача, що акаунта з даним номером телефону не існує і переправить користувача на сторінку реєстрації. Це зображено на рисунку 3.8.

Рисунок 3.8 – Сповіщення, що акаунт з таким номером не існує

6) Вибір категорії курсів

Передумови: додаток встановлено на телефоні і користувач зареєструвався.

Кроки виконання:

- ввійти в акаунт;
- переглянути список категорій;
- вибрати потрібну категорію;
- нажати на вибрану категорію.

Очікуваний результат: система відсортує курси і виведе на екран курси вибраної категорії. Це зображено на рисунку 3.9.

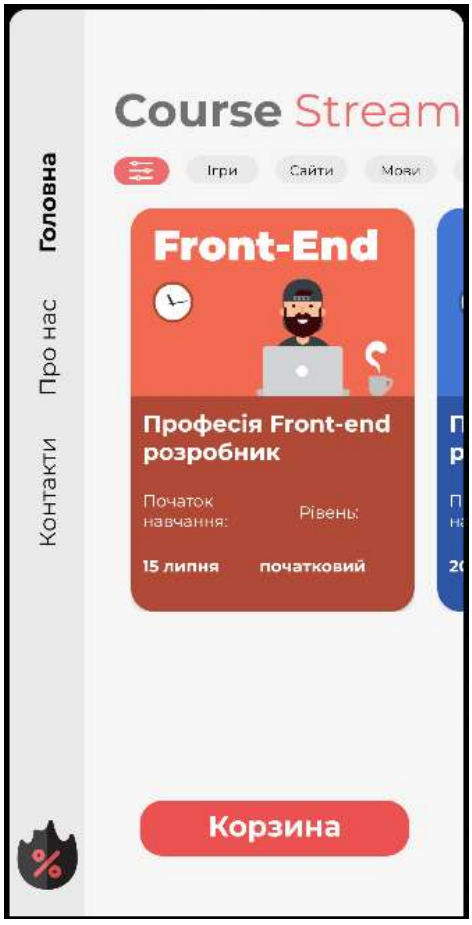


Рисунок 3.9 – Курси вибраної категорії

7) Перегляд детальної інформації про курс

Передумови: користувач успішно увійшов в свій акаунт.

Кроки виконання:

- вибрати потрібну категорію курсів;
- знайти потрібний курс;
- нажати на курс.

Очікуваний результат: відкриється вікно курсу з детальною інформацією.

Процес перегляду детальної інформації про курс «Професія Front-end розробник» зображено на рисунку 3.10.

Рисунок 3.10 – Сторінка курсу

8) Додання курсу в корзину

Передумови: користувач успішно увійшов в свій акаунт.

Кроки виконання:

- знайти потрібний курс;
- нажати на курс;
- нажати на кнопку «Додати в корзину».

Очікуваний результат: Система додасть курс в корзину і сповістить про це.

Процес додаваня в корзину курсу «Професія Back-end розробник» зображено на рисунку 3.11.

Рисунок 3.11 – Додання курсу в корзину

9) Перегляд курсів в корзині

Передумови: користувач успішно увійшов в свій акаунт.

Кроки виконання:

- додати необхідні курси в корзину;
- на домашній сторінці додатку (див. рисунок 3.5) нажати на кнопку «Корзина».

Очікуваний результат: Система перенаправить користувача на сторінку корзини, де відображаються всі додані курси.

Для тесту цієї функціональності було додано декілька курсів в корзину. Вміст корзини зображено на рисунку 3.12.

Рисунок 3.12 – Сторінка корзини з доданими курсами

10) Отримання консультації

Передумови: при виборі курсів у користувача виникли запитання чи проблеми.

Кроки виконання:

- на бічній панелі додатку нажати на кнопку «Контакти»;
- переглянути сторінку контактів мобільного додатку, яка зображена на рисунку 3.13;
- зателефонувати/написати лист в центр підтримки магазину.

Очікуваний результат: Отримання кваліфікованої консультації.

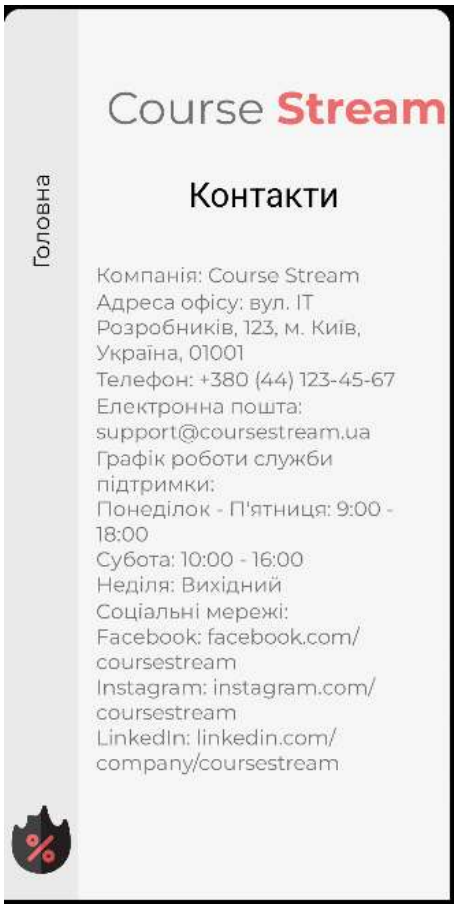


Рисунок 3.13 – Сторінка контактів додатку

11) Перегляд адміністратором існуючих профілів користувачів

Кроки виконання:

- ввійти в акаунт Firebase;
- нажати на кнопку «Go to console»;
- вибрати проект, а саме даний мобільний додаток;
- в лівому меню вибрати «Realtime Database»;
- вибрати пункт «Data».

Очікуваний результат: Адміністратор перегляне існуючі профілі користувачів. Це зображено на рисунку 3.14.

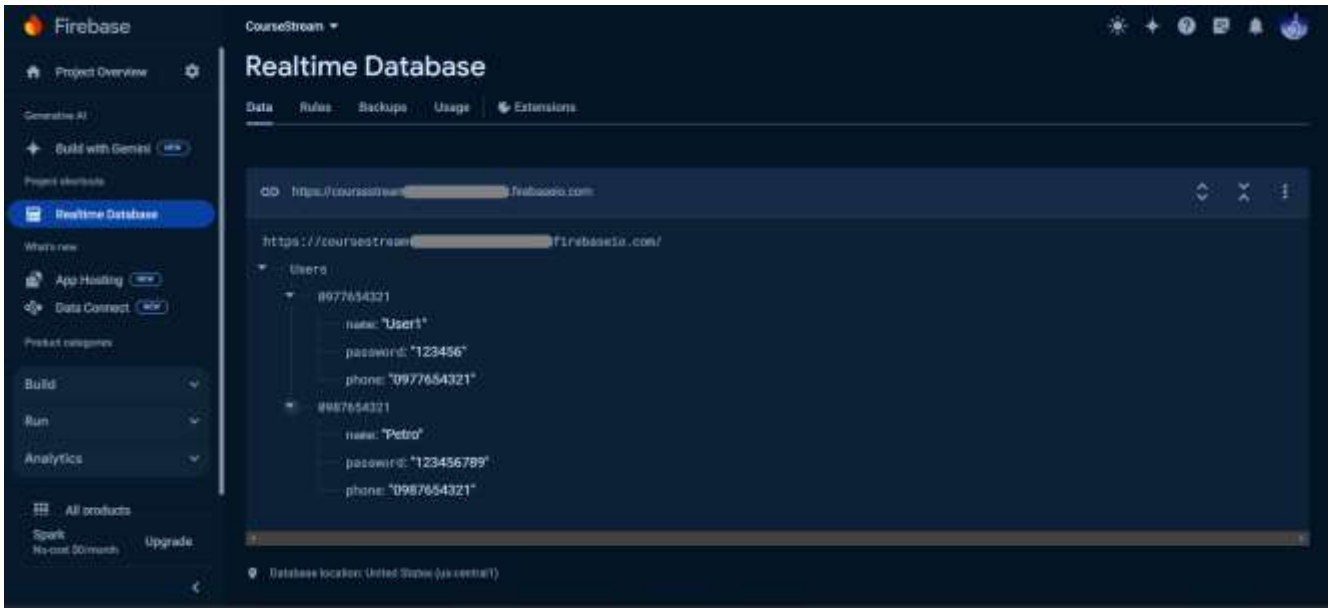


Рисунок 3.14 – Акаунти користувачів

12) Редагування/видалення профілів користувачів

Передумови: адміністратор увійшов в акаунт Firebase.

Кроки виконання:

- нажати на кнопку «Go to console»;
- вибрати проект, а саме даний мобільний додаток;
- в лівому меню вибрати «Realtime Database»;
- вибрати пункт «Data»;
- вибрати профіль користувача який потрібно редагувати/видалити;
- редагувати/видалити профіль користувача.

Очікуваний результат: Адміністратор редагує/видалить профіль користувача. Це зображено на рисунку 3.15.

Рисунок 3.15 – Редагування/видалення профілю користувача

13) Перегляд аналітики адміністратором

Передумови: адміністратор увійшов в акаунт Firebase.

Кроки виконання:

- нажати на кнопку «Go to console»;
- вибрати проект, а саме даний мобільний додаток;
- в лівому меню вибрати «Project Overview»;
- переглянути аналітику.

Очікуваний результат: Адміністратор перегляне і проаналізує аналітику роботи мобільного додатку з продажі онлайн-курсів, оскільки це є необхідним для досягнення успіху в даній сфері діяльності. Аналітика надасть важливі дані, на основі яких можна приймати рішення, вдосконалювати продукт і розробляти ефективні стратегії продажу онлайн-курсів. Аналітика зображена на рисунку 3.16.

Рисунок 3.16 – Сторінка з аналітикою мобільного додатку

На завершення, тестування функціональності має важливе значення для забезпечення того, щоб додаток відповідав функціональним вимогам і надавав користувачам надійний і безперебійний досвід роботи. Систематично перевіряючи кожну функцію, можна виявити і виправити будь-які проблеми, гарантуючи, що додаток буде працювати так, як очікується в реальних умовах. Цей суворий процес

тестування в кінцевому підсумку підвищує якість і надійність програмного забезпечення, надаючи користувачам надійний і ефективний інструмент для їхніх потреб.

3.3 Навантажувальне тестування

Навантажувальне тестування є важливою складовою процесу тестування, що дозволяє оцінити продуктивність та стійкість ПЗ під високим навантаженням. У випадку, коли мобільний додаток підключений до бази даних Firebase Realtime Database, проведення навантажувального тестування бази даних є необхідним, оскільки швидкість відклику мобільного додатку залежить від продуктивності бази даних.

Для проведення навантажувального тестування було обрано браузерний ресурс – BlazeMeter. BlazeMeter – це хмарна платформа для тестування навантаження, що надає можливість тестування веб-додатків та API, з метою визначення продуктивності та масштабованості ваших додатків. BlazeMeter дозволяє створювати та запускати тестові сценарії з великою кількістю користувачів з різних точок світу, щоб перевірити, як додаток поводиться при великому навантаженні. BlazeMeter також надає різноманітні інструменти аналізу результатів тестування, що дозволяють знайти та виправити проблеми з продуктивністю додатка [12].

За допомогою цього ресурсу, було проведено навантажувальне тестування двічі, перший раз з навантаженням 10 користувачів, другий раз 50. Результат тестування показано на рисунках 4.1 і 4.2.

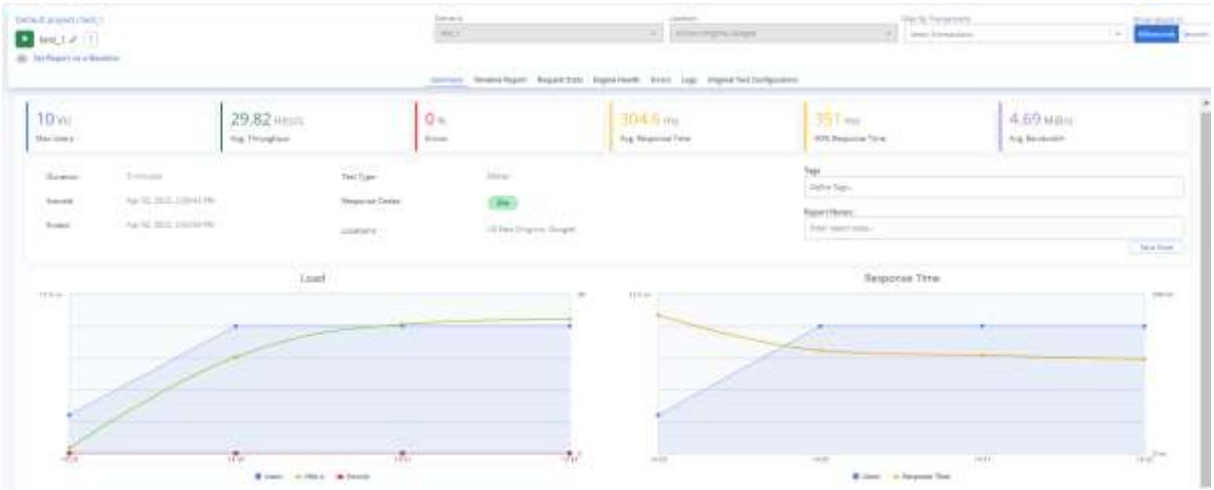


Рисунок 4.1 – Результат першого тестування

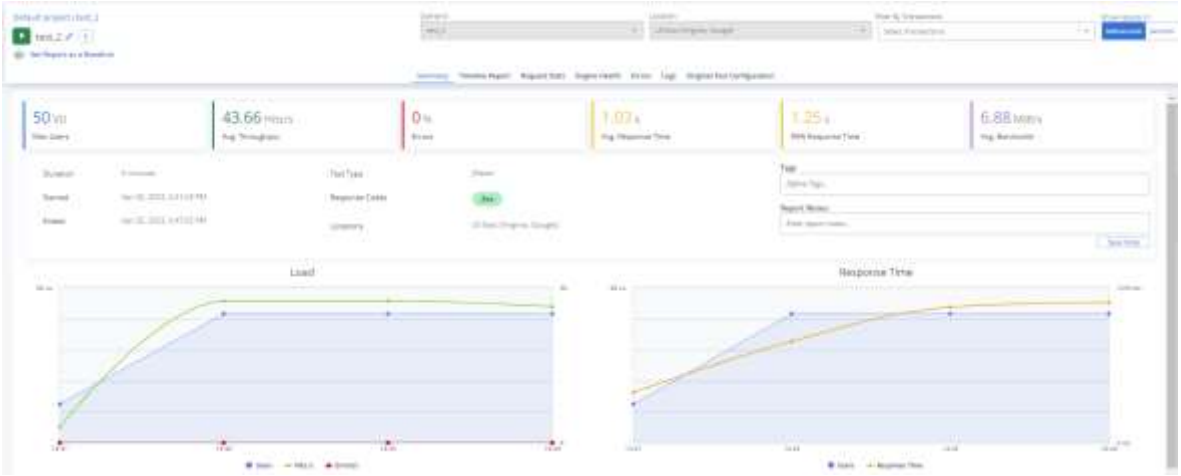


Рисунок 4.2 – Результат другого тестування

Отже, за результатами двох тестувань можна зробити висновок, що обрана база даних в реальному часі працює коректно з різним навантаженням, адже і при одночасній роботі 10 користувачів, і при одночасній роботі 50 користувачів сайт не видавав помилок. Також можна зауважити, що при збільшенні кількості користувачів збільшився і середній час реакції та відгуку, проте це збільшення є незначним та не впливає на роботу бази даних. При збільшенні кількості користувачів, збільшилась також середня пропускна здатність, що не є дивним, адже її збільшення дозволяє обслуговувати більше запитів користувачів, що допомагає забезпечити швидший доступ до бази даних при більшому навантаженні.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Порядок надання домедичної допомоги постраждалим при раптовій зупинці серця

Раптова зупинка серця (РЗС) є однією з найнебезпечніших невідкладних ситуацій, що вимагає негайного втручання. Від своєчасності та правильності надання домедичної допомоги часто залежить життя постраждалого. Цей розділ присвячений порядку дій, які необхідно виконати для надання ефективної домедичної допомоги при раптовій зупинці серця. Ми розглянемо основні принципи, етапи та методи проведення серцево-легеневої реанімації (СЛР), а також використання автоматичних зовнішніх дефібриляторів (АЗД).

Під час зупинки серцевої діяльності у потерпілого настає клінічна смерть. У нього не промацується пульсація сонної артерії на шиї, не прослуховується серцебиття, припиняється дихання, зіниці розширюються, шкірні покриви й слизові оболонки губ синюшні. Через 5-6 хв після настання клінічної смерті відбуваються необоротні зміни в нервових клітинах центрів головного мозку, що керують життєво важливими функціями організму. Внаслідок цього настає біологічна смерть. З клінічної смерті потерпілого можуть вивести лише негайні (не пізніш як через 4-5 хв після її настання) й енергійні заходи, спрямовані на відновлення серцевої діяльності та дихання. З цією метою роблять штучне дихання і непрямий (закритий) масаж серця.

Роблячи непрямий масаж серця, потерпілого кладуть спиною на тверду поверхню (шосе, до рога та ін.). Той, хто надає допомогу, стає на коліна збоку (краще справа) біля потерпілого і, поклавши кисті рук одну на другу в нижній частині груднини, робить енергійний поштовх натискуючи на груднину так, щоб вона зміщувалася приблизно на 4-5 см у напрямі до хребта і штучне дихання способом «рот до рота». При цьому стискується серце між грудниною і хребтом, кров із порожнини серця виштовхується в кровоносні судини. Оскільки стінка грудної клітки еластична, то вона повертається у вихідне положення, а порожнини

серця наповнюються кров'ю. Поштовхи слід робити ритмічно, близько 60 разів на хвилину. Процес надання домедичної допомоги постраждалим при раптовій зупинці серця зображено на рисунку 4.1.

Рисунок 4.1 – Непрямий масаж серця

Дітям, особливо молодшого віку, непрямий масаж серця можна робити однією рукою або навіть двома пальцями, але частоту поштовхів треба збільшити до 100-120 разів на хвилину.

Іноді непрямий масаж серця поєднують із штучним диханням. Щоправда, це потребує чималих зусиль. Тому бажано, щоб потерпілим у стані клінічної смерті допомогу надавали двоє: один робить непрямий масаж серця, другий — штучне дихання. Після кожного вдування повітря в легені потерпілого чотири рази натискують на груднину. Під час вдування повітря не можна стискувати грудну клітку.

Непрямий масаж серця і штучне дихання звичайно може робити й одна людина. У такому разі після кожного вдування повітря в рот або в ніс натискують чотири рази на груднину.

Показником ефективності непрямого масажу серця і штучного дихання є порожевіння шкірних покривів, звуження зіниць, поява на великих артеріях (стегновій, сонній) синхронно натискуванню на груднину пульсових поштовхів і, нарешті, відновлення самостійного дихання й серцебиття. Непрямий масаж серця і штучне дихання слід проводити до відновлення серцевої діяльності й дихання.

Коли вони не дають ефекту, то їх припиняють тільки після огляду потерпілого медичним працівником [13].

Для відновлення нормального ритму серця у разі його раптової зупинки також можна використовувати автоматичні зовнішні дефібрилятори. Автоматичні зовнішні дефібрилятори (АЗД) є пристроями, призначеними для відновлення нормального ритму серця у разі його раптової зупинки шляхом подання електричного імпульсу. Використання АЗД є критично важливим у перші хвилини після зупинки серця, оскільки швидке відновлення серцевого ритму значно підвищує шанси на виживання постраждалого.

Порядок дій при використанні АЗД:

1) Оцінка стану постраждалого: Переконайтеся, що постраждалий дійсно перебуває у стані зупинки серця. Для цього перевірте наявність дихання та пульсу.

2) Виклик екстреної допомоги: Негайно зателефонуйте до служби екстреної допомоги (наприклад, 112) і повідомте про ситуацію.

3) Підготовка АЗД: Увімкніть пристрій і дотримуйтесь голосових або візуальних інструкцій.

4) Прикріплення електродів: Прикріпіть електроди до оголених грудей постраждалого, як показано на схемі, яка зазвичай міститься на самому пристрої або на упаковці електродів.

5) Аналіз ритму серця: Дайте АЗД проаналізувати ритм серця постраждалого. Не торкайтеся постраждалого в цей момент.

6) Виконання дефібриляції: Якщо пристрій рекомендує дефібриляцію, переконайтеся, що ніхто не торкається постраждалого, і натисніть кнопку для подання розряду.

7) Подальші дії: Після розряду продовжуйте СЛР (якщо необхідно) і дотримуйтесь інструкцій АЗД до прибуття медичних працівників або до відновлення постраждалим свідомості.

Отже, своєчасне та правильне виконання алгоритму надання домедичної допомоги при раптовій зупинці серця є ключовим фактором у збереженні життя

постраждалого. Кожна хвилина зволікання значно знижує шанси на успішну реанімацію, тому важливо знати основні етапи СЛР і вміти застосовувати АЗД. Регулярне навчання та практика з підготовки до таких екстрених ситуацій допоможуть забезпечити високий рівень готовності та впевненості у діях рятувальників і пересічних громадян. Пам’ятайте, що ваші знання та вміння можуть стати вирішальними у критичний момент.

4.2 Заходи щодо захисту установки від короткого замикання

Коротке замикання є одним із найбільш поширених та небезпечних видів електричних аварій, що можуть виникнути в будь-якій електричній установці. Одним із основних методів захисту від таких аварій є заземлення, яке забезпечує безпечний шлях для струму в разі короткого замикання і запобігає ураженню людей електричним струмом. Ефективне заземлення сприяє стабільній роботі електричних систем та мінімізує ризик пошкодження обладнання і виникнення пожеж. У цьому розділі детально розглядається принципи роботи заземлення, його типи та методи, а також практичні рекомендації щодо впровадження ефективної системи заземлення для захисту установок від короткого замикання.

Захисне заземлення – це навмисне електричне з`єднання з землею або з її еквівалентом металевих неструмоведучих частин, котрі можуть опинитись під напругою.

Заземлення – умисне сполучення з землею або її еквівалентом металевих частин ЕУ. Якщо з землею з’єднані струмопровідні частини – це робоче заземлення . Заземлення бувають:

- робочими – якщо з землею сполучені струмопровідні частини;
- захисними – неструмопровідні частини (корпуси) , які можуть опинитись під напругою при пошкоджені робочої ізоляції;
- технологічні;

- для блискавкозахисту;
- суміщені.

У відповідності з ПУЭ захисне заземлення виконується у таких випадках:

- для всіх ЕУ – при $U_{НОМ}$ = 380В і вище ~ струму та 440 В і вище = струму;
- для ЕУ, розміщених у приміщеннях з підвищеною і особливою небезпекою та поза приміщеннями – при $U_{НОМ}$ вище 42 В ~ струму і вище 110 В = струму;
- для вибухонебезпечних ЕУ – при будь-якій напрузі ~ чи = струмів.

Призначення захисного заземлення: захист від напруги дотику, тобто від напруги на корпусі електроустановки (при пошкодженні робочої ізоляції і переході напруги металевому корпусі) відносно землі. Схема захисного заземлення в мережах з ізольованою нейтраллю зображена на рисунку 4.2.

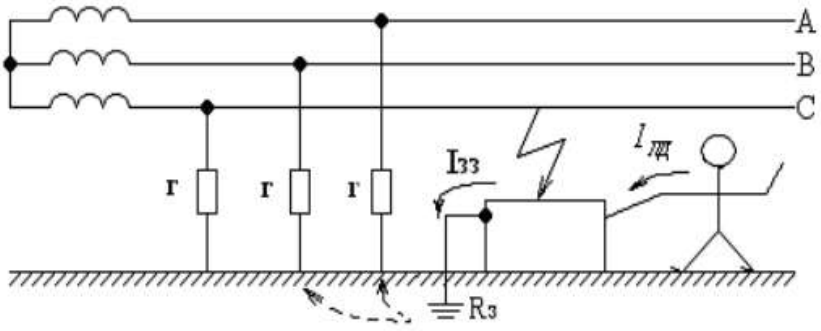


Рисунок 4.2 – Схема захисного заземлення в мережах з ізольованою нейтраллю

Конструктивно заземлюючі пристрої являють сукупність заземлювача і заземлюючих провідників.

Заземлювач – це металоконструкція, яка розміщена в грунті і має з ним хороший електроконтакт.

Заземлюючий провідник – це провідник, що з’єднує корпус електро-приймача з заземлювачем. Заземлюючий провідник, що має 2 або більше відгалужень, називається магістраллю заземлення.

Заземлювачі поділяються на натуральні та штучні.

Натуральні заземлювачі – це металоконстукції в грунті, які мають з ним гарний контакт, виконують будівельні або технологічні функції і паралельно застосовуються для заземлення. ПУЭ передбачає в першу чергу використовувати натуральні заземлювачі, а саме:

- прокладені в землі металеві трубопроводи, за винятком трубопроводів горючих речовин;
- обсадні труби свердловин;
- підземні металеві чи залізобетонні конструкції будинків і споруд, у тому числі і опор ПЛ;
- свинцеві оболонки кабелів при їх кількості не менше двох, алюмінієві оболонки кабелів використовувати як природні заземлювачі заборонено;
- грозозахисний трос ПЛ, якщо він ізольований від опір, та ін.

Штучні заземлювачі – це спеціально виконані і призначені виключно для заземлення металеві конструкції в грунті.

Найменші розміри сталевих штучних заземлювачів такі:

- діаметр круглих стрижневих заземлювачів: неоценкованих – 10 мм, оценкованих – 6 мм;
- товщина полички кутника – 4 мм;
- площа перерізу прямокутних заземлювачів – 48 мм², товщина – 4 мм.

Найчастіше штучні заземлювачі використовуються з вертикальних електродів, розміщених по периметру або уподовж сторони об’єкта і сполучених горизонтальними електродом. Для вертикальних електродів використовують сталеві прутки діаметром 12-16 мм або кутники з розмірами 32х32х4 мм чи 40х40х5 мм довжиною 3-5 м чи більше. Для горизонтальних електродів застосовуються ті ж прутки або сталева стрічка 4х40 мм².

Як заземлюючі провідники можуть бути використовані спеціально передбачені провідники, металеві конструкції будинків, арматура залізобетонних констукцій, металоконструкції виробничого призначення, сталеві труби, алюмінієві оболонки кабелів та ін.

Схема занулення зображена на рисунку 4.3.

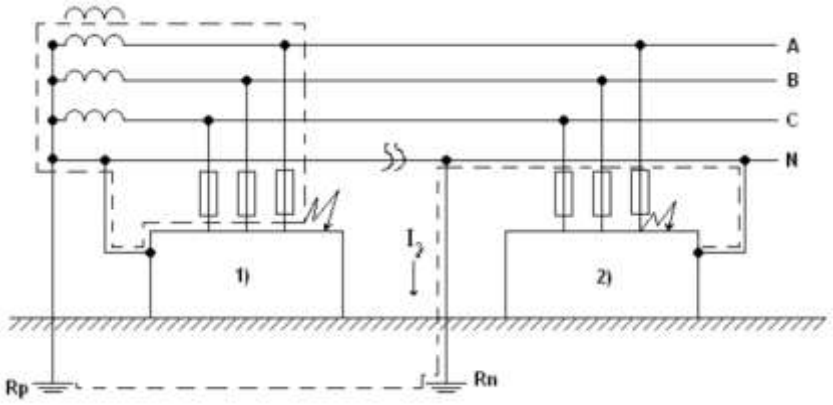


Рисунок 4.3 – Схема занулення

В мережах напругою до 1 кВ з глухозаземленою нейтраллю захисне заземлення не є ефективною мірою захисту. В таких мережах використовують занулення – з’єднання корпусів ЕУ з нульовим проводом трьохфазної мережі з глухозаземленою нейтраллю, з заземленим полюсом в однофазних мережах і з заземленою середньою точку в мережах постійного струму [14].

Отже, заземлення є ключовим елементом захисту електричних установок від короткого замикання, що забезпечує безпеку як для людей, так і для обладнання. Використання правильно спроектованих і встановлених систем заземлення знижує ризик виникнення аварійних ситуацій та їхніх негативних наслідків. Регулярне обслуговування та перевірка систем заземлення гарантує їхню ефективність і надійність. Пам’ятайте, що забезпечення належного заземлення є не тільки технічним, але й життєво важливим аспектом, який сприяє довговічності та безперебійній роботі ваших електричних установок. Завжди дотримуйтесь відповідних стандартів і норм для забезпечення максимальної безпеки вашого електричного обладнання.

ВИСНОВКИ

Дослідження, представлене в цій бакалаврській роботі, заглиблюється в розробку мобільного додатку для продажу онлайн-курсів, що охоплює комплексні етапи планування, проектування та впровадження. Основна мета полягала у створенні ефективного, зручного та надійного додатку, який би відповідав потребам онлайн-освіти, що постійно зростають. Методологія, прийнята для цього дослідження, поєднувала теоретичний аналіз з практичною розробкою додатку, забезпечуючи всебічний підхід до вирішення визначеної проблеми.

Методологія дослідження включала кілька ключових етапів: аналіз вимог, розробка системи, впровадження та тестування. Спочатку було проведено ретельний аналіз вимог, щоб зрозуміти потреби цільових користувачів та визначити функціональні можливості, необхідні для додатку.

Етап проектування системи включав створення детальних архітектурних планів, діаграм потоків даних та макетів користувацького інтерфейсу. Основна увага при проектуванні приділялася забезпеченню масштабованості, безпеки та простоти використання. На етапі реалізації відбулося фактичне кодування додатку з використанням сучасних мов програмування та фреймворків, придатних для розробки мобільних додатків. Акцент був зроблений на написанні чистого, зручного для обслуговування коду та дотриманні найкращих практик програмної інженерії.

Тестування, важливий компонент методології дослідження, проводилося в кілька етапів. Кожен етап був спрямований на виявлення та виправлення помилок, гарантуючи, що додаток функціонує належним чином за різних умов. Особливу увагу було приділено тестуванню користувацького інтерфейсу, тестуванню функціональності та навантажувальному тестуванню, кожне з яких стосувалося різних аспектів продуктивності та зручності використання додатку.

Розроблений додаток має велике значення та потенціал для використання у швидкозростаючій сфері онлайн-освіти. Він забезпечує масштабовану та

настроювану платформу, яку можна адаптувати для різних освітніх сфер. Модульний дизайн додатку дозволяє легко інтегрувати додаткові функції та можливості, задовольняючи різноманітні потреби та вподобання користувачів.

З технічної точки зору, додаток демонструє ефективне використання ресурсів, забезпечуючи оптимальну продуктивність навіть при великому користувацькому навантаженні. Економічна ефективність додатку очевидна з його потенціалу спростити процес продажі онлайн-курсів, зменшити адміністративні витрати та підвищити доступність курсів для ширшої аудиторії. Орієнтований на користувача дизайн та інтуїтивно зрозумілий інтерфейс сприяють підвищенню рівня утримання користувачів та їх задоволеності, що призводить до кращих бізнес-результатів для розробників курсів.

Крім того, дослідження пропонує кілька пропозицій щодо напрямків подальших досліджень. Майбутні дослідження можуть бути зосереджені на розширенні можливостей адаптивного навчання додатку, включно з розширеною аналітикою даних для надання персоналізованого навчального досвіду. Іншою потенційною сферою є інтеграція штучного інтелекту та алгоритмів машинного навчання, щоб рекомендувати курси на основі вподобань користувача та його прогресу в навчанні. Крім того, розширення функціональності додатку для підтримки середовищ спільного навчання та віртуальних класів може значно підвищити його освітню цінність.

Таким чином, ця кваліфікаційна робота бакалавра представляє комплексне дослідження з розробки мобільного додатку для продажу онлайн-курсів, підкреслюючи його технічну, економічну та освітню значимість. Методологія дослідження, що охоплює аналіз вимог, системний дизайн, реалізацію та тестування, забезпечила створення надійного та зручного для користувача додатку. Отримані результати підкреслюють потенціал додатку для широкого використання та створюють основу для майбутнього розвитку технологій онлайн-освіти. Вирішуючи поточні проблеми та використовуючи технологічні інновації, це дослідження робить свій внесок у постійний розвиток цифрових рішень для навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ICONIX - Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ICONIX (дата звернення: 19.06.2024).

2. Android Studio: переваги та особливості [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/ (дата звернення: 19.06.2024).

3. Develop for Android | Android Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/develop (дата звернення: 19.06.2024).

4. Java - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java (дата звернення: 19.06.2024).

5. Firebase - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Firebase (дата звернення: 19.06.2024).

6. Firebase Realtime Database [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/database (дата звернення: 19.06.2024).

7. Інструкція, як будувати UML-діаграми | DOU [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/ (дата звернення: 19.06.2024).

8. AppCompatActivity | Android Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity (дата звернення: 19.06.2024).

9. Android Toolbar for AppCompatActivity [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@101/android-toolbar-for-appcompatactivity-671b1d10f354 (дата звернення: 19.06.2024).

10. Що таке UML Class Diagram, включаючи UML Class Diagram Maker [Електронний ресурс] – Режим доступу до ресурсу:

https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (дата звернення: 19.06.2024).

11. Інтерфейс користувача – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0 (дата звернення: 19.06.2024).

12. BlazeMeter Continuous Testing | BlazeMeter by Perforce [Електронний ресурс] – Режим доступу до ресурсу: https://www.blazemeter.com/ (дата звернення: 19.06.2024).

13. Скобло Ю. С, Соколовська Т. Б., Мазоренко Д. І., Б 40 Тіщенко Л. М., Троянов М. М. Безпека життєдіяльності: Навчальний посібник для вищих навчальних закладів III— IV рівнів акредитації. – Київ: Кондор, 2003. – 424 с.

14. Зеркалов Д. В. Безпека життєдіяльності та основи охорони праці: Навчальний посібник для студентів гуманітарного блоку. – Київ: Основа, 2016. – 267 с.

15. Пастух О. Estimation of shape memory alloys functional properties by methods of artificial intelligence [Електронний ресурс] / О. Пастух, О. Ясній. – 2024. – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/handle/lib/39148 (дата звернення: 19.06.2024).

16. Петрик М. Two-stage random sequential adsorption of discorectangles and disks on a two-dimensional surface [Електронний ресурс] / М. Петрик, І. Бойко. – 2023. – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/handle/lib/44653 (дата звернення: 19.06.2024).

17. Михалик Д. Intellectual information technologies for the study of filtration in multidimensional nanoporous particles media [Електронний ресурс] / Д. Михалик, М. Петрик. – 2022. – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/handle/lib/42018?locale=gl (дата звернення: 19.06.2024).

ДОДАТКИ

ДОДАТОК А – Тези конференції

УДК 004.41
Містерман П. – ст. гр. СП-41**;** Петрик М. – д.ф.-м.н., професор.
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА ДОДАТКА ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ З ПРОДАЖУ ОНЛАЙН КУРСІВ НА JAVA З ВИКОРИСТАННЯМ FIREBASE REALTIME DATABASE ТА ANDROID STUDIO

Misterman P.; Petrik M., PhD,-M.Sc., Professor
Ternopil Ivan Puluj National Technical University

DEVELOPMENT OF AN APPLICATION FPR MOBILE DEVICES FOR SELLING ONLINE COURSES IN JAVA USING FIREBASE REALTIME DATABASE AND ANDROID STUDIO

Ключові слова: мобільний додаток, Java, Firebase Realtime Database, Android Studio
Keywords: mobile app, Java, Firebase Realtime Database, Android Studio

У епоху цифрової трансформації, коли мобільні технології стають невід'ємною частиною нашого повсякденного життя, розробка мобільних додатків набуває все більшої вагомості. Створення мобільних додатків на Java, що інтегрується з Firebase Realtime Database та розробляється в Android Studio, є відображенням сучасних тенденцій у сфері мобільних технологій.

Використання Firebase Realtime Database дозволяє розробникам створювати динамічні та інтерактивні додатки з можливістю миттєвого оновлення контенту без необхідності перезавантаження додатку. Це забезпечує користувачам безперервний доступ до курсів, а також можливість відстежувати свій прогрес у реальному часі.

Android Studio, як потужне середовище розробки, надає інструменти для створення високоякісних мобільних додатків. Його інтеграція з Firebase Realtime Database спрощує процес синхронізації даних та управління користувацькими сесіями.

Основні компоненти такого додатку включають в себе інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко навігувати між різними курсами. Додатково, інтеграція з платіжними системами забезпечує зручність та безпеку транзакцій.

Розробка такого додатку вимагає глибокого розуміння не лише технічних навичок, але й здатності добре розуміти потреби та очікування цільової аудиторії. Вона також відображає стратегічний підхід до використання мобільних технологій для створення цінності та забезпечення конкурентних переваг у даній сфері.

Такий проект не лише відповідає сучасним вимогам ринку, але й відкриває нові горизонти для розвитку інформаційних технологій, забезпечуючи користувачам доступ до курсів в зручний для них спосіб.

Література

1. Firebase Realtime Database Documentation – Офіційна документація для створення та управління базою даних у реальному часі. URL: https://firebase.google.com/docs/database

2. Get Started with Realtime Database – Firebase - Посібник для швидкого початку роботи з Firebase Database.URL: https://firebase.google.com/docs/database/flutter/start

3. Android Studio Overview – Огляд офіційного IDE для розробки додатків Android. URL: https://developer.android.com/studio

ДОДАТОК Б – Лістинг коду мобільного додатку

Файл «MainActivity.java»

```
package com.example.coursestream;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {
    private Button joinButton, loginButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        joinButton = (Button) findViewById(R.id.main_join_btn);
        loginButton = (Button) findViewById(R.id.main_login_btn);

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent loginIntent = new Intent(MainActivity.this,
LoginActivity.class);
                startActivity(loginIntent);
            }
        });

        joinButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent registerIntent = new
Intent(MainActivity.this, RegisterActivity.class);
                startActivity(registerIntent);
            }
        });
    }
}
```

Файл «LoginActivity.java»

```
package com.example.coursestream;

import androidx.annotation.NonNull;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.coursestream.Model.Users;
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;

public class LoginActivity extends AppCompatActivity {

    private Button loginBtn;
    private EditText phoneInput, passwordInput;
    private ProgressDialog loadingBar;

    private String parentDbName = "Users";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        loginBtn = (Button)findViewById(R.id.login_btn);
        phoneInput = (EditText)
findViewById(R.id.login_phone_input);
        passwordInput = (EditText)
findViewById(R.id.login_password_input);
        loadingBar = new ProgressDialog(this);

        loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loginUser();
            }
        });
    }
```

```
    private void loginUser() {
        String phone = phoneInput.getText().toString();
        String password = passwordInput.getText().toString();

        if(TextUtils.isEmpty(phone))
        {
            Toast.makeText(this, "Введіть номер",
Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(password))
        {
            Toast.makeText(this, "Введіть пароль",
Toast.LENGTH_SHORT).show();
        }
        else
        {
            loadingBar.setTitle("Вхід в додаток");
            loadingBar.setMessage("Будь ласка зачекайте...");
            loadingBar.setCanceledOnTouchOutside(false);
            loadingBar.show();

            ValidateUser(phone, password);
        }
    }

    private void ValidateUser(String phone, String password) {
        final DatabaseReference RootRef;
        RootRef = FirebaseDatabase.getInstance().getReference();
        RootRef.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {

if(dataSnapshot.child(parentDbName).child(phone).exists())
                {
                    Users usersData =
dataSnapshot.child(parentDbName).child(phone).getValue(Users.class);
                    if(usersData.getPhone().equals(phone))
                    {
                        if(usersData.getPassword().equals(password))
                        {
                            if(parentDbName.equals("Users")){
                                loadingBar.dismiss();
                                Toast.makeText(LoginActivity.this,
"Успішний вхід!", Toast.LENGTH_SHORT).show();
                                Log.d("LoginActivity", "Navigating
to HomeActivity");

                                Intent homeIntent = new
Intent(LoginActivity.this, HomeActivity.class);
                                startActivity(homeIntent);
```

```
                            }
                        }
                        else {
                            loadingBar.dismiss();
                            Toast.makeText(LoginActivity.this, "Не
правильний пароль", Toast.LENGTH_SHORT).show();
                        }
                    }
                }
                else {
                    loadingBar.dismiss();
                    Toast.makeText(LoginActivity.this, "Аккаунт з
номером " + phone + "не існує", Toast.LENGTH_SHORT).show();
                    Intent registerIntent = new
Intent(LoginActivity.this, RegisterActivity.class);
                    startActivity(registerIntent);
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError
databaseError) {

            }
        });
    }
}
```

Файл «RegisterActivity.java»

```
package com.example.coursestream;

import androidx.annotation.NonNull;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
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;

import java.util.HashMap;

public class RegisterActivity extends AppCompatActivity {

    private Button registerBtn;
    private EditText usernameInput, phoneInput, passwordInput;
    private ProgressDialog loadingBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        registerBtn = (Button) findViewById(R.id.register_btn);
        usernameInput = (EditText)
findViewById(R.id.register_username_input);
        phoneInput = (EditText)
findViewById(R.id.register_phone_input);
        passwordInput = (EditText)
findViewById(R.id.register_password_input);
        loadingBar = new ProgressDialog(this);

        registerBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                CreateAccount();
            }
        });
    }

    private void CreateAccount() {
        String username = usernameInput.getText().toString();
        String phone = phoneInput.getText().toString();
        String password = passwordInput.getText().toString();

        if(TextUtils.isEmpty(username))
        {
            Toast.makeText(this, "Введіть ім'я",
Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(phone))
        {
            Toast.makeText(this, "Введіть номер",
Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(password))
        {
            Toast.makeText(this, "Введіть пароль",
Toast.LENGTH_SHORT).show();
        }
```

```
        else
        {
            loadingBar.setTitle("Створення аккаунту");
            loadingBar.setMessage("Будь ласка зачекайте...");
            loadingBar.setCanceledOnTouchOutside(false);
            loadingBar.show();

            ValidatePhone(username, phone, password);
        }
    }

    private void ValidatePhone(String username, String phone, String
password) {
        final DatabaseReference RootRef;
        RootRef = FirebaseDatabase.getInstance().getReference();

        RootRef.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {

if(!(dataSnapshot.child("Users").child(phone).exists()))
                {
                    HashMap<String, Object> userDataMap = new
HashMap<>();
                    userDataMap.put("phone", phone );
                    userDataMap.put("name", username );
                    userDataMap.put("password", password );


RootRef.child("Users").child(phone).updateChildren(userDataMap)
                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull
Task<Void> task) {
                                    if(task.isSuccessful()) {
                                        loadingBar.dismiss();

Toast.makeText(RegisterActivity.this, "Реєстрація пройшла успішно.",
Toast.LENGTH_SHORT).show();

                                        Intent loginIntent = new
Intent(RegisterActivity.this, LoginActivity.class);
                                        startActivity(loginIntent);
                                    }
                                    else {
                                        loadingBar.dismiss();

Toast.makeText(RegisterActivity.this, "Помилка.",
Toast.LENGTH_SHORT).show();
```

```
                                    }
                                }
                            });
                }
                else {
                    loadingBar.dismiss();
                    Toast.makeText(RegisterActivity.this, "Номер" +
phone + "уже зареєстрований", Toast.LENGTH_SHORT).show();

                    Intent loginIntent = new
Intent(RegisterActivity.this, LoginActivity.class);
                    startActivity(loginIntent);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError
databaseError) {

            }
        });
    }
}
```

Файл «HomeActivity.java»

```
package com.example.coursestream;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.coursestream.Adapter.CategoryAdapter;
import com.example.coursestream.Adapter.CourseAdapter;
import com.example.coursestream.Model.Category;
import com.example.coursestream.Model.Course;

import java.util.ArrayList;
import java.util.List;

public class HomeActivity extends AppCompatActivity {

    RecyclerView categoryRecycler, courseRecycler;
    CategoryAdapter categoryAdapter;
    static CourseAdapter courseAdapter;
```

```
    static List<Course> courseList = new ArrayList<>();
    static List<Course> fullCoursesList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        List<Category> categoryList = new ArrayList<>();
        categoryList.add(new Category(1, "Ігри"));
        categoryList.add(new Category(2, "Сайти"));
        categoryList.add(new Category(3, "Мови"));
        categoryList.add(new Category(4, "Інше"));

        setCategoryRecycler(categoryList);

        courseList.add(new Course(1, "java", "Професія
Java\nрозробник", "1 серпня", "початковий", "#424345", "За програму
ви вивчите побудову графічних програм під ПК, розробку веб-сайтів на
основі Java Spring Boot, вивчите побудову повноцінних Андроїд
додатків і відмінно вивчите саму мову Джава!", 3));
        courseList.add(new Course(2, "python", "Професія
Python\nрозробник", "10 липня", "початковий", "#9FA52D", "Ви вивчите
основи мови Python, бібліотеки для аналізу даних, веб-розробку на
Flask та Django, а також основи машинного навчання.", 3));
        courseList.add(new Course(3, "unity", "Професія
Unity\nрозробник", "1 липня", "початковий", "#651730", "Навчитеся
створювати ігри на движку Unity, опануєте C# для скриптування
ігрових процесів, а також отримаєте навички роботи з анімаціями та
фізикою у 2D та 3D іграх.", 1));
        courseList.add(new Course(4, "front_end", "Професія Front-
end\nрозробник", "15 липня", "початковий", "#B04935", "Опануєте
HTML, CSS, JavaScript, навчитеся створювати адаптивні веб-сайти,
працювати з фреймворками типу React та інструментами для веб-
розробки.", 2));
        courseList.add(new Course(5, "back_end", "Професія Back-
end\nрозробник", "20 липня", "початковий", "#2D55A5", "Ви вивчите
серверне програмування на мові Python (Django), PHP, або Node.js,
опануєте роботу з базами даних та створення API для веб-
застосунків.", 2));
        courseList.add(new Course(6, "full_stack", "Професія Full
Stack\nрозробник", "10 серпня", "початковий", "#FFC007", "Здобудете
навички як у Front-end, так і у Back-end розробці, навчитеся
працювати з сучасними фреймворками та технологіями для створення
повноцінних веб-застосунків.", 2));
        courseList.add(new Course(7, "node_js", "Професія JS
розробник\nрозробник", "12 серпня", "початковий", "#FCA311", "Ви
вивчите основи JavaScript, роботу з Node.js, Express.js для
створення серверних застосунків, а також роботу з базами даних та
створення RESTful API.", 2));
        courseList.add(new Course(8, "csharp", "Професія C#
розробник\nрозробник", "1 серпня", "початковий", "#EE6055",
```

```
"Опануєте мову програмування C#, створення додатків на .NET,
розробку ігор на Unity, а також основи роботи з базами даних та веб-
розробкою на ASP.NET.", 3));

        fullCoursesList.addAll(courseList);

        setCourseRecycler(courseList);
    }

    private void setCourseRecycler(List<Course> courseList) {
        RecyclerView.LayoutManager layoutManager = new
LinearLayoutManager(this, RecyclerView.HORIZONTAL, false);

        courseRecycler = findViewById(R.id.courseRecycler);
        courseRecycler.setLayoutManager(layoutManager);

        courseAdapter = new CourseAdapter(this, courseList);
        courseRecycler.setAdapter(courseAdapter);
    }

    private void setCategoryRecycler(List<Category> categoryList)
{//налаштування виводу категорій горизонтально
        RecyclerView.LayoutManager layoutManager = new
LinearLayoutManager(this, RecyclerView.HORIZONTAL, false);

        categoryRecycler = findViewById(R.id.categoryRecycler);
        categoryRecycler.setLayoutManager(layoutManager);

        categoryAdapter = new CategoryAdapter(this, categoryList);
        categoryRecycler.setAdapter(categoryAdapter);

    }

    public void openShoppingCart(View view){
        Intent intent = new Intent(this, OrderPage.class);
        startActivity(intent);
    }

    public static void showCoursesByCategory(int category) {
        courseList.clear();
        courseList.addAll(fullCoursesList);

        List<Course> filterCourses = new ArrayList<>();
        for (Course c : courseList) {
            if (c.getCategory() == category)
                filterCourses.add(c);
        }

        courseList.clear();
        courseList.addAll(filterCourses);

        courseAdapter.notifyDataSetChanged();
```

```
    }

    public void openContactsPage(View view) {
        Intent intent = new Intent(this, ContactsPage.class);
        startActivity(intent);
    }

    public void openAboutPage(View view) {
        Intent intent = new Intent(this, AboutPage.class);
        startActivity(intent);
    }
}
```

Файл «CoursePage.java»

```
package com.example.coursestream;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

import com.example.coursestream.Model.Order;

public class CoursePage extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_course_page);

        ConstraintLayout courseBg = findViewById(R.id.coursePageBg);
        ImageView courseImage =findViewById(R.id.coursePageImage);
        TextView courseTitle = findViewById(R.id.coursePageTitle);
        TextView courseDate = findViewById(R.id.coursePageDate);
        TextView courseLevel = findViewById(R.id.coursePageLevel);
        TextView courseText = findViewById(R.id.coursePageText);


courseBg.setBackgroundColor(getIntent().getIntExtra("courseBg", 0));

courseImage.setImageResource(getIntent().getIntExtra("courseImage",
0));

courseTitle.setText(getIntent().getStringExtra("courseTitle"));
```

```
courseDate.setText(getIntent().getStringExtra("courseDate"));

courseLevel.setText(getIntent().getStringExtra("courseLevel"));

courseText.setText(getIntent().getStringExtra("courseText"));

    }

    public void addToCart(View view){
        int item_id = getIntent().getIntExtra("courseId", 0);
        Order.items_id.add(item_id);
        Toast.makeText(this, "Додано", Toast.LENGTH_LONG).show();
    }

    public void openHomePage(View view) {
        Intent intent = new Intent(this, HomeActivity.class);
        startActivity(intent);
    }

    public void openContactsPage(View view) {
        Intent intent = new Intent(this, ContactsPage.class);
        startActivity(intent);
    }

    public void openAboutPage(View view) {
        Intent intent = new Intent(this, AboutPage.class);
        startActivity(intent);
    }
}
```

Файл «OrderPage.java»

```
package com.example.coursestream;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.coursestream.Model.Course;
import com.example.coursestream.Model.Order;

import java.util.ArrayList;
import java.util.List;

public class OrderPage extends AppCompatActivity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order_page);

        ListView orders_list = findViewById(R.id.orders_list);

        List<String> coursesTitle = new ArrayList<>();
        for(Course c: HomeActivity.fullCoursesList){
            if(Order.items_id.contains(c.getId()))
                coursesTitle.add(c.getTitle());
        }

        orders_list.setAdapter(new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, coursesTitle));
    }

    public void openHomePage(View view) {
        Intent intent = new Intent(this, HomeActivity.class);
        startActivity(intent);
    }

    public void openContactsPage(View view) {
        Intent intent = new Intent(this, ContactsPage.class);
        startActivity(intent);
    }

    public void openAboutPage(View view) {
        Intent intent = new Intent(this, AboutPage.class);
        startActivity(intent);
    }
}
```

Файл «AboutPage.java»

```
package com.example.coursestream;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

public class AboutPage extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about_page);
```

```
    }

    public void openHomePage(View view) {
        Intent intent = new Intent(this, HomeActivity.class);
        startActivity(intent);
    }
}
```

Файл «ContactsPage.java»

```
package com.example.coursestream;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

public class ContactsPage extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_contacts_page);
    }

    public void openHomePage(View view) {
        Intent intent = new Intent(this, HomeActivity.class);
        startActivity(intent);
    }
}
```

Файл «CategoryAdapter.java»

```
package com.example.coursestream.Adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.coursestream.HomeActivity;
import com.example.coursestream.Model.Category;
import com.example.coursestream.R;
```

```
import java.util.List;

public class CategoryAdapter extends
RecyclerView.Adapter<CategoryAdapter.CategoryViewHolder> {

    Context context;
    List<Category> categories;

    public CategoryAdapter(Context context, List<Category>
categories) {
        this.context = context;
        this.categories = categories;
    }

    @NonNull
    @Override
    public CategoryViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View categoryItems =
LayoutInflater.from(context).inflate(R.layout.category_item, parent,
false);
        return new CategoryViewHolder(categoryItems);
    }

    @Override
    public void onBindViewHolder(@NonNull CategoryViewHolder holder,
int position) {

holder.categoryTitle.setText(categories.get(position).getTitle());
        holder.itemView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

HomeActivity.showCoursesByCategory(categories.get(position).getId())
;
            }
        });
    }

    @Override
    public int getItemCount() {
        return categories.size();
    }

    public static final class CategoryViewHolder extends
RecyclerView.ViewHolder {

        TextView categoryTitle;

        public CategoryViewHolder(@NonNull View itemView) {
```

```
            super(itemView);

            categoryTitle =
itemView.findViewById(R.id.categoryTitle);
        }
    }
}
```

Файл «CourseAdapter.java»

```
package com.example.coursestream.Adapter;

import android.app.Activity;
import android.app.ActivityOptions;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.util.Pair;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.example.coursestream.CoursePage;
import com.example.coursestream.Model.Course;
import com.example.coursestream.R;

import java.util.List;

public class CourseAdapter extends
RecyclerView.Adapter<CourseAdapter.CourseViewHolder>{

    Context context;
    List<Course> courses;

    public CourseAdapter(Context context, List<Course> courses) {
        this.context = context;
        this.courses = courses;
    }

    @NonNull
    @Override
    public CourseViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View courseItems =
```

```
LayoutInflater.from(context).inflate(R.layout.course_item, parent,
false);
        return new CourseAdapter.CourseViewHolder(courseItems);
    }

    @Override
    public void onBindViewHolder(@NonNull
CourseAdapter.CourseViewHolder holder, int position) {

holder.courseBg.setCardBackgroundColor(Color.parseColor(courses.get(
position).getColor()));

        int imageId = context.getResources().getIdentifier("ic_" +
courses.get(position).getImg(), "drawable",
context.getPackageName());
        holder.courseImage.setImageResource(imageId);

holder.courseTitle.setText(courses.get(position).getTitle());
        holder.courseDate.setText(courses.get(position).getDate());

holder.courseLevel.setText(courses.get(position).getLevel());

        holder.itemView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
//                int currentPosition = holder.getAdapterPosition();
                Intent intent = new Intent(context,
CoursePage.class);

                ActivityOptions options =
ActivityOptions.makeSceneTransitionAnimation(
                        (Activity) context,
                        new Pair<View, String>(holder.courseImage,
"courseImage")
                );
                intent.putExtra("courseBg",
Color.parseColor(courses.get(position).getColor()));
                intent.putExtra("courseImage", imageId);
                intent.putExtra("courseTitle",
courses.get(position).getTitle());
                intent.putExtra("courseDate",
courses.get(position).getDate());
                intent.putExtra("courseLevel",
courses.get(position).getLevel());
                intent.putExtra("courseText",
courses.get(position).getText());
                intent.putExtra("courseId",
courses.get(position).getId());
                context.startActivity(intent, options.toBundle());
            }
        });
```

```
    }

    @Override
    public int getItemCount() {
        return courses.size();
    }

    public static final class CourseViewHolder extends
RecyclerView.ViewHolder{

        CardView courseBg;
        ImageView courseImage;
        TextView courseTitle, courseDate, courseLevel;
        public CourseViewHolder(@NonNull View itemView){
            super(itemView);

            courseBg = itemView.findViewById(R.id.courseBg);
            courseImage = itemView.findViewById(R.id.courseImage);
            courseTitle = itemView.findViewById(R.id.courseTitle);
            courseDate = itemView.findViewById(R.id.courseDate);
            courseLevel = itemView.findViewById(R.id.courseLevel);

        }
    }
}
```

Файл «Category.java»

```
package com.example.coursestream.Model;

public class Category {
    int id;
    String title;

    public Category(int id, String title) {
        this.id = id;
        this.title = title;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }
```

```
    public void setTitle(String title) {
        this.title = title;
    }
}
```

Файл «category_item.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:layout_marginEnd="15dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/app_name"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/category" />

    <TextView
        android:id="@+id/categoryTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/montserrat"
        android:gravity="center"
        android:text="Ігри"
        android:textColor="@color/black"
        android:textSize="12sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Файл «Course.java»

```
package com.example.coursestream.Model;

public class Course {

    int id, category;
    String img, title, date, level, color, text;

    public Course(int id, String img, String title, String date,
String level, String color, String text, int category) {
        this.id = id;
        this.img = img;
        this.title = title;
        this.date = date;
        this.level = level;
        this.color = color;
        this.text = text;
        this.category = category;
    }

    public int getCategory() {
        return category;
    }

    public void setCategory(int category) {
        this.category = category;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getImg() {
        return img;
    }

    public void setImg(String img) {
        this.img = img;
    }
```

```
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getLevel() {
        return level;
    }

    public void setLevel(String level) {
        this.level = level;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

Файл «Order.java»

```
package com.example.coursestream.Model;

import java.util.HashSet;
import java.util.Set;

public class Order {
    public static Set<Integer> items_id = new HashSet<>();
}
```

Файл «Users.java»

```
package com.example.coursestream.Model;
```

```
public class Users {
    private String name, phone, password;

    public Users()
    {

    }

    public Users(String name, String phone, String password) {
        this.name = name;
        this.phone = phone;
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

ДОДАТОК В – Диск із кваліфікаційною роботою бакалавра