Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота бакалавра
з теми: “Модель мобільної платформи для індивідуального навчального плану здобувача вищої освіти”

Виконала: здобувачка вищої освіти
групи KN1-B21
спеціальності 122 Комп’ютерні науки
Собачинська Анна Віталіївна

Керівник: Іванюк Віталій Анатолійович,
доктор технічних наук, доцент,
завідувач кафедри комп’ютерних наук

Рецензент:
Геселева Катерина Григорівна, кандидат фізико-математичних наук, декан фізико-математичного факультету

м. Кам’янець-Подільський – 2025 р.

АНОТАЦІЯ

Собачинська А.В. Модель мобільної платформи для індивідуального навчального плану здобувача вищої освіти. − Кваліфікаційна робота бакалавра, 2025.

Кваліфікаційна робота присвячена розробці моделі мобільної платформи для управління індивідуальним навчальним планом студентів. Метою роботи є створення кросплатформного рішення, яке усуває проблеми фрагментації даних, відсутності синхронізації та обмеженого доступу до академічної інформації в існуючих системах.

У роботі проведено аналіз сучасних платформ для навчання, визначено ключові недоліки та сформульовано вимоги до нової системи. Розроблено мобільну платформу на базі Flutter/Dart із використанням Firebase для зберігання даних та автентифікації. Платформа інтегрується з Google Workspace для автоматичного оновлення даних та забезпечує доступ до індивідуального навчального плану, оцінок, інформації про викладачів та дисципліни.

Запропоноване рішення сприяє підвищенню ефективності навчального процесу завдяки зручному доступу до персоналізованої академічної інформації. Практичне значення роботи полягає у можливості впровадження платформи у закладах вищої освіти для покращення цифрового супроводу студентів.

Ключові слова: мобільна платформа, індивідуальний навчальний план, Flutter, Firebase, Google Workspace, вища освіта, управління навчанням.

ABSTRACT

Sobachynska A.V. Model of a Mobile Platform for Individual Curriculum of Higher Education Students. − Bachelor's Qualification Thesis, 2025.

The qualification thesis is devoted to developing a model of a mobile platform for managing students' individual curricula. The work aims to create a cross-platform solution that addresses the issues of data fragmentation, lack of synchronization, and limited access to academic information in existing systems.

The study analyzes modern learning platforms, identifies key shortcomings, and formulates requirements for the new system. A mobile platform was developed using Flutter framework with Firebase for data storage and authentication. The platform provides access to individual curriculum, grades, information about teachers and courses, and integrates with Google Workspace for automatic data updates.

The proposed solution enhances the efficiency of the educational process through convenient access to personalized academic information. The practical significance of the work lies in the potential implementation of the platform in higher education institutions to improve digital support for students.

Keywords: mobile platform, individual curriculum, Flutter, Firebase, Google Workspace, higher education, learning management.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій значно трансформує освітній процес, сприяючи зміні традиційних підходів до організації навчання. В умовах цифрової трансформації освіти зростає потреба в персоналізації навчальних процесів, що є важливим аспектом для вищих навчальних закладів. Одним із ключових інструментів такої персоналізації є індивідуальні навчальні плани (ІНП), які дозволяють студентам формувати власні траєкторії навчання, вибирати дисципліни, керувати навантаженням та інтегрувати освітні ресурси.

Завдяки розвитку цифрових технологій, інформаційні системи та мобільні платформи стали важливими інструментами для підвищення ефективності освітнього процесу. На жаль, традиційні підходи до управління ІНП, які досі використовують статичні таблиці або обмежені веб-системи, не забезпечують належної автоматизації та синхронізації між студентами, адміністрацією та викладачами. Це створює певні труднощі при оновленні даних та обміні інформацією.

Актуальність дослідження полягає в необхідності створення ефективної мобільної платформи, що забезпечить студентам зручний доступ до їхніх академічних даних в реальному часі. Таке рішення дозволить не лише автоматизувати процес збору та відображення інформації, але й спростить обмін даними між різними учасниками освітнього процесу.

Мета дослідження – розробити модель мобільної платформи для студентів, що надасть доступ до актуальних академічних результатів та ІНП через мобільний застосунок або веб-сайт.

Для досягнення поставленої мети були поставлені наступні завдання:

- провести аналіз існуючих систем доступу до академічних даних студентів;
- визначити основні функціональні вимоги до мобільної платформи;
- розробити модель інтеграції з сучасними хмарними сервісами для об’єднання даних з різних джерел у єдину базу;

- реалізувати основні функції мобільного додатку, виключно авторизацію через Google та виведення результатів навчання;
- протестувати систему і оцінити її зручність та ефективність.

Об’єкт дослідження – система доступу до академічних даних студентів у вищій освіті.

Предмет дослідження – мобільна платформа для перегляду ІНП та екзаменаційних результатів, що інтегрується з Firebase та Google Workspace.

Методи дослідження включають аналіз існуючих освітніх ІТ-рішень, розробку та тестування мобільної платформи, інтеграцію з хмарними сервісами та оцінку ефективності роботи системи.

Практичне значення одержаних результатів: Запропонована мобільна платформа може бути впроваджена у заклади вищої освіти для покращення доступу студентів до навчальної інформації. Система забезпечує зручний перегляд індивідуального навчального плану та результатів успішності, що сприяє підвищенню ефективності навчального процесу. Рішення є гнучким і може бути адаптоване до особливостей різних навчальних програм, а також інтегроване з іншими інформаційними ресурсами університету.

Результати роботи використовуються в роботі кафедри комп’ютерних наук Кам’янець-Подільського національного університету імені Івана Огієнка.

Структура роботи: Кваліфікаційна робота складається з вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі аналізуються теоретичні аспекти управління індивідуальними навчальними планами та мобільні платформи для освіти. Другий розділ присвячений проектуванню мобільної платформи, вибору технологій і архітектури. Третій розділ присвячений практичній частині − розробці мобільного застосунку, що включає створення інтерфейсу, інтеграцію з навчальним планом і проведення тестування. Список використаних джерел складається із 18 джерел.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ТЕХНОЛОГІЧНІ РІШЕННЯ

1.1 Системи управління індивідуальним навчальним планом

Система управління індивідуальним навчальним планом студентів відіграє важливу роль в організації освітнього процесу у закладах вищої освіти. Індивідуальний навчальний план (ІНП) містить перелік дисциплін, які студент має опанувати протягом навчання, включаючи як обов'язкові, так і вибіркові курси. Однак на практиці доступ до цієї інформації залишається ускладненим для студентів, що створює низку проблем.

На сьогоднішній день основним джерелом інформації про дисципліни, які здобувач вищої освіти має вивчати, є офіційні сайти університету. Там розміщені навчальні плани, що містять лише обов'язкові дисципліни для кожної спеціальності. Однак студент не має змоги дізнатися з цього джерела повний перелік дисциплін, які він фактично обрав для вивчення. Проблема стає особливо гострою, коли студент бажає отримати інформацію про свої вибіркові дисципліни або про дисципліни, які він уже вивчав у попередніх семестрах та підсумкові оцінки за них.

Щоб дізнатися про перелік дисциплін, які він вивчав або вивчатиме, студент змушений звертатися до деканату. Там він може отримати свій паперовий індивідуальний навчальний план або переглянути залікову книжку, якщо вона ведеться в університеті. Такий підхід має значні недоліки:

1. Фрагментарність інформації. На сайті університету відображаються лише загальні навчальні плани без урахування індивідуальних вибіркових дисциплін студентів. Також, здобувачі вищої освіти не завжди пам'ятають, які саме вибіркові дисципліни вже були ними обрані або вивчені.
2. Проблеми з доступністю даних. Якщо студенту терміново потрібна інформація про дисципліни, які він вивчав у попередніх семестрах, йому доводиться особисто відвідувати деканат, що не завжди зручно через обмежений графік роботи адміністрації. Крім того, студент

може фізично перебувати далеко від навчального закладу – наприклад, бути на практиці, навчальній мобільності, дистанційному навчанні або проживати в іншому місті. У таких випадках отримання необхідних даних стає ще більш ускладненим або й зовсім неможливим без зовнішньої допомоги.

3. Обмежені можливості для аналізу навчального процесу. Студент не має швидкого способу переглянути свої оцінки за попередні курси, зв’язатися з викладачами або переглянути свої успіхи у навчанні.

Таким чином, чинна система управління навчальними планами дещо ускладнює ефективний доступ до інформації. Вона базується на традиційних паперових документах і обмежених цифрових ресурсах, що суттєво ускладнює студентам взаємодію з власним освітнім процесом.

Для вирішення вищезазначених проблем необхідне впровадження цифрової платформи, яка дозволить студентам отримувати всю необхідну інформацію про їхній індивідуальний навчальний план у зручному форматі [18]. Така система повинна забезпечувати наступні можливості:

1. Доступ до повного переліку дисциплін
   - Студент матиме змогу переглядати всі дисципліни, які він вивчав у попередніх семестрах, включно з вибірковими курсами.
   - Інформація буде доступною в будь-який момент без потреби відвідувати деканат чи звертатися до викладачів.
2. Інтеграція з університетською базою даних
   - Система повинна отримувати актуальні дані про дисципліни безпосередньо з університетських баз даних або електронних документів.
   - Оновлення бази даних дозволяє отримувати студенту інформацію фактично в реальному часі її заповнення.
3. Перегляд навчального прогресу
   - Відображення оцінок за всі попередні семестри у структурованому форматі.

- Можливість бачити викладачів, які вели певні дисципліни, разом із контактною інформацією (електронною поштою).

4. Прогнозування майбутнього навчального процесу
   - Студент зможе побачити дисципліни, які заплановані для нього на майбутні семестри, включно з обов’язковими та вибірковими дисциплінами (якщо вони ще не обрані, система позначатиме їх, наприклад «Дисципліна 1», «Дисципліна 2» і т.д.).
5. Зручний інтерфейс та мобільний доступ
   - Інформація має бути доступною як у веб-версії, так і через мобільний застосунок.
   - Студент повинен мати можливість швидко знаходити необхідні дані без складної навігації.

Впровадження такої системи не лише спростить доступ до індивідуального навчального плану, а й зробить навчальний процес більш прозорим і зручним для студентів. Автоматизація цього процесу дозволить студентам бути краще поінформованими про свій навчальний шлях.

Таким чином, проблема управління індивідуальним навчальним планом студентів потребує оновлення підходів. Хоча відповідно до законодавства інформація зберігається в паперовому вигляді, доступ до електронних даних значно покращить якість процесу і забезпечить зручність для студентів. Варто зазначити, на кафедрі комп’ютерних наук Кам'янець-Подільського національного університету імені Івана Огієнка вже існує певна електронна документація, яку використовують викладачі для обліку оцінок, результатів заліків і екзаменів. Ці дані заповнюються викладачами у зручному для них форматі, і хоча формат таблиць може варіюватися, сам процес введення інформації займає мало часу.

Крім того, є вже наявні електронні ресурси, такі як списки студентів групи та навчальні плани, що дозволяють адміністрації та викладачам ефективно управляти інформацією. Завдяки існуючим даним, перенесення оцінок та іншої інформації до застосунку не вимагатиме значних додаткових

зусиль. Основний акцент зроблений на тому, що зібрані дані можна зручно перенести в єдину систему, що дозволить студентам мати доступ до своїх оцінок та навчальних планів без зайвих затримок, не впливаючи суттєво на роботу викладачів і адміністрації. Таким чином, реалізація такого інструменту потребує лише мінімальних додаткових кроків, зокрема налаштування формату, що дозволить максимально знизити часові витрати на введення даних і зробить доступ до інформації зручним для всіх учасників навчального процесу.

1.2 Аналіз існуючих мобільних платформ для освітніх цілей

У сучасному світі мобільні платформи стали важливими інструментами для студентів, надаючи їм зручний доступ до навчальних матеріалів, розкладів, оцінок та іншої важливої інформації. З кожним роком мобільні застосунки для освіти розвиваються, а їх функціонал збагачується новими можливостями, що дозволяє студентам бути в курсі подій та ефективно організовувати своє навчання. У цьому контексті варто розглянути існуючі мобільні платформи, що надають подібну інформацію студентам, та порівняти їх з функціоналом, який пропонується в розробленій платформі.

Перш за все, варто звернути увагу на мобільні платформи, призначені для управління освітнім процесом. Існують різні мобільні застосунки, які спрямовані на підтримку студентів в організації їх навчання. Одним з таких популярних платформ є Google Classroom. Ця платформа широко використовується для організації матеріалів, зв'язку між викладачами та студентами, а також для організації завдань і оцінювання [11]. Однак, хоча Google Classroom надає можливість студентам переглядати навчальні матеріали та виконувати завдання, вона не включає в себе детальну інформацію про розклад занять, індивідуальний навчальний план або оцінки за попередні семестри.

Інші популярні мобільні платформи, такі як Moodle та Edmodo, також мають функції для організації навчання, зокрема для створення завдань, тестів

та взаємодії з іншими студентами [16]. Однак, вони знову ж таки не орієнтовані на представлення персональних даних студентів, таких як індивідуальний навчальний план, а також не забезпечують зручного доступу до всіх дисциплін, вивчених студентом, і оцінок за ці дисципліни.

Багато університетів мають власні мобільні платформи, призначені для студентів, що дозволяють доступ до основних функцій, таких як перевірка розкладу занять, актуальних новин, оцінок, та ін. Наприклад, платформи, такі як Student Portal або CampusNet, надають студентам можливість переглядати розклад занять, отримувати доступ до електронних журналів та оцінок, а також звертатися до викладачів.

Проте, ці платформи часто мають обмежений функціонал, зокрема вони можуть не надавати можливості переглядати інформацію про дисципліни, що були обрані на попередніх курсах, або не забезпечують інтеграції з іншими освітніми системами університету, що обмежує зручність використання таких платформ для студента [3].

Такі платформи, як UniApp або MyUni, надають функціонал для перегляду всіх дисциплін за всі курси. Вони також можуть включати профіль студента, що дає можливість переглядати основну інформацію про студента, включаючи його курси, групу, спеціальність та інші корисні дані. Проте в більшості таких платформ немає інтеграції з розкладом занять або з іншими системами університету, що призводить до необхідності отримання окремої інформації в різних місцях.

Окрему нішу займають платформи для комунікації та організації зустрічей, як-от Zoom або Microsoft Teams, надають студентам можливість брати участь у відкритих лекціях або онлайн-зустрічах. Вони можуть бути корисними для підтримки взаємодії між студентами і викладачами, але не забезпечують можливості для зручного перегляду історії навчальних дисциплін, оцінок або побудови індивідуальних навчальних планів. Такі платформи більш орієнтовані на онлайн-навчання і взаємодію в реальному часі, але вони не замінюють повноцінну систему управління освітнім

процесом, де студенти могли б зберігати та аналізувати інформацію про свою освіту.

Аналіз існуючих мобільних платформ для освітніх цілей показав, що хоча багато з них надають важливу функціональність для студентів, вони часто не повністю відповідають потребам студентів у зручному доступі до всіх важливих даних про навчання. Існуючі платформи, такі як Google Classroom, Moodle, Edmodo, та університетські додатки, здебільшого орієнтовані на взаємодію між студентами та викладачами, а також організацію навчальних завдань і матеріалів. Водночас, вони не забезпечують універсальний доступ до повної історії навчальних дисциплін, оцінок, та персональних даних студента.

1.3 Вибір технологій та середовища розробки

Для розробки кросплатформених мобільних та веб-інтерфейсів вибрано Flutter − фреймворк від компанії Google, який вперше був випущений у 2017 році. Цей інструмент дозволяє створювати інтерфейси з нативним виглядом і відчуттям на платформах Android, iOS, Web та Desktop. Flutter базується на мові програмування Dart і використовує власну бібліотеку віджетів, що включає компоненти Material Design та Cupertino, а також має сильний рушій рендерингу [9]. Однією з ключових особливостей є Ahead-of-Time (AOT) компіляція у нативний код, що забезпечує високу продуктивність додатків. Google постійно вдосконалює Flutter, підвищуючи його ефективність, зокрема за допомогою впровадження нового графічного рушія Impeller, що оптимізує використання GPU та знижує енергоспоживання. Активна спільнота розробників сприяє поширенню технології, а прикладом успішного застосування є повна розробка інтерфейсу застосунку Google Pay India на базі Flutter [2].

Переваги Flutter і Dart включають високу продуктивність додатків та багатий набір UI-віджетів, які забезпечують однаковий вигляд інтерфейсу на різних платформах. Flutter підтримує швидкий цикл розробки з можливістю використання функції hot reload, має офіційну документацію і значний попит

на ринку праці. Версії Flutter 2 і 3 додали підтримку веб- та настільних операційних систем, а також передбачають вбудовану офлайн-роботу з базами даних за умови відповідного налаштування [2].

Серед недоліків Flutter варто зазначити більший розмір фінального APK чи IPA файлу порівняно з нативними додатками. Також Flutter-додаток не є цілком нативним: хоч він і забезпечує однорідність інтерфейсу, відчуття деякої відмінності від рідних компонентів все ж присутнє. Екосистема пакетів, хоча і швидко зростає, наразі поступається за кількістю пакетів таким технологіям, як JavaScript/Node (npm має сотні тисяч пакетів). Крім того, мова Dart менш відома серед розробників у порівнянні з JavaScript, тому на початкових етапах необхідне додаткове навчання команди. Відповідно до огляду BrowserStack, Flutter відзначається «чудовим UI, численними віджетами, швидкістю додатків та простотою створення однакового інтерфейсу на різних пристроях» [2].

Альтернативи Flutter:

- React Native (JavaScript) – також кросплатформений фреймворк, що використовує JSX та нативні компоненти UI. На відміну від Flutter, React Native компілюється Just-in-Time (JIT) і потребує міст (bridge) між JavaScript і нативними модулями. Це дає широку спільноту та гнучкий JavaScript-стек, але може знижувати продуктивність через затримки моста [2]. React Native легший до опанування для тих, хто вже знає React/JavaScript, але важче реалізувати точний інтерфейс на всіх платформах.
- Xamarin (C#/.NET) – створює кросплатформені UI, але менш поширений для мобільних.
- Native (Android/iOS) – забезпечує максимум продуктивності й контролю, але потребує подвійної розробки (Java/Kotlin + Swift).
- Web (PWA) – можна було б зробити прогресивний веб-додаток, але для нативного UX обрали саме Flutter.

У цілому, Flutter обрано за його продуктивність, єдиний код для різних платформ і стабільну підтримку від Google.

Для бекенд-функціональності застосовано набір сервісів Firebase: Authentication, Firestore Database та Hosting. Firebase реалізує принцип BaaS (Backend as a Service), що значно спрощує розробку мобільного додатку. Згідно з офіційною документацією Firebase для Flutter, інтеграція Firebase з Flutter сприяє швидшому виведенню продукту на ринок і забезпеченню цінності для користувачів із меншими витратами.

Firebase Authentication надає готові бекенд-сервіси та SDK для автентифікації користувачів, включно з авторизацією через email-пароль, Google, Facebook, телефон та інші способи. Це значно спрощує процес забезпечення безпеки, порівняно з розробкою власних рішень на базі Node.js/Express, що вимагали б реалізації складних процедур валідації. Крім того, Firebase є простим у налаштуванні та пропонує систему правил безпеки (Security Rules) для контролю доступу до даних.

Firebase включає дві основні бази даних:

1. Realtime Database − це NoSQL-хмарна база у форматі JSON, оптимізована для синхронізації даних у реальному часі з мінімальними затримками (близько 10 мс). Вона добре підходить для застосунків із потребою негайного оновлення інформації, наприклад, чатів [1].

2. Cloud Firestore є гнучкою документною NoSQL-базою, що зберігає дані у вигляді документів і колекцій. Firestore підтримує розширені можливості запитів та індексування, автоматично масштабується до великої кількості підключень і запитів, а також забезпечує офлайн-доступ до даних [1].

У проєкті можливо комбінувати обидві бази, наприклад, використовуючи Realtime Database для оперативних подій, а Firestore − для структурованих записів. Переваги Firebase включають готові сервіси, швидкий старт, безкоштовний рівень для початкового навантаження, автоматичне масштабування, реплікацію даних у реальному часі та офлайн-підтримку. Недоліками є складність міграції даних та обмежена гнучкість запитів у

Realtime Database через її деревоподібну структуру, можливі високі витрати при великому навантаженні. Порівняно з власним бекендом на Node.js і СУБД, Firebase суттєво економить час розробки.

Для розгортання веб-застосунку (PWA) використовують Firebase Hosting − промисловий сервіс з глобальною CDN (Content Delivery Network), що забезпечує швидку й безпечну доставку контенту. [5] Розгортання проходить однією командою CLI, автоматично застосовується SSL, сервіс оптимізований для SPA та статичних додатків, з підтримкою динамічного бекенду через Cloud Functions чи Cloud Run. У порівнянні з Heroku, Firebase Hosting простіший і швидший у деплойменті, не вимагає Git-репозиторію. Heroku краще підходить для серверних застосунків з масштабуванням, але для статичного фронтенду Firebase є більш зручним і вигідним варіантом.

Для зберігання деяких даних, таких як довідкові таблиці чи результати контролю знань, застосовується Google Sheets із програмним доступом через Google Apps Script − серверну JavaScript-платформу, яка дозволяє розширювати функціонал продуктів Google Workspace [10]. За допомогою Apps Script можна програмно створювати, читати і змінювати файли Google Sheets, а також розгортати скрипти як веб-служби, які повертають дані у форматі JSON через HTTP-запити. Це забезпечує зручний інтерфейс для наповнення та читання даних у реальному часі, що особливо важливо для освітніх процесів. Альтернативою могла б стати повна робота з Firebase або традиційні SQL-бази, однак Google Sheets надають знайомий користувачу інтерфейс і гнучкість змін, хоч і мають обмеження у продуктивності при великих обсягах даних та відсутність складних транзакцій.

Для комунікації та організації освітнього процесу використовується Google Workspace for Education (рис. 1.1), який включає набори інструментів, таких як Gmail, Календар, Документи, Таблиці, Meet, Classroom та інші [14].

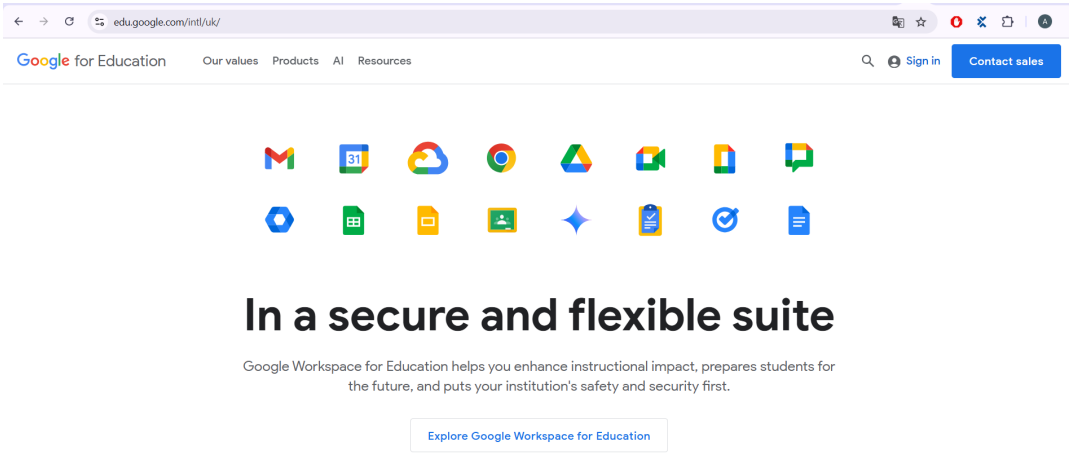


Рис. 1.1 *Панель сервісів Google Workspace for Education*

Це дозволяє викладачам і студентам ефективно співпрацювати у реальному часі. Google Classroom інтегрований з Workspace і слугує єдиною платформою для управління курсами. Згідно з офіційними джерелами, інструменти Workspace покликані спростити колаборацію та навчання, забезпечуючи надійну хмарну інфраструктуру з високою доступністю (99,9% SLA). Більшість сервісів надається безкоштовно для освітніх закладів.

Щодо альтернатив: аналогічні можливості пропонує Microsoft 365 Education (Word, Excel, Teams, Outlook тощо), однак перевагу надано Google Workspace через його простіший інтерфейс, автоматичну синхронізацію документів у хмарі та зручну інтеграцію з Google Sheets. Також розглядаються інші системи управління навчанням або сховища даних, але саме екосистема Google забезпечує більш цілісне рішення для потреб освітнього процесу.\

Для реалізації проєкту було обрано редактор коду Visual Studio Code (VS Code) – зручне та ефективне кросплатформене середовище з широкими можливостями для розширення функціоналу. Завдяки своїй продуктивності та гнучкості, VS Code активно використовується для розробки різноманітних застосунків, зокрема Flutter-додатків. Як зазначено в офіційній документації Flutter, цей редактор ідеально підходить для створення, запуску та налагодження мобільних застосунків за допомогою відповідного розширення. Багато розробників визнають VS Code одним із найпопулярніших середовищ

(IDE): за опитуваннями, воно входить у топ-2 найпоширеніших редакторів разом із Visual Studio [17].

Серед ключових переваг Visual Studio Code − підтримка численних мов програмування та розширень, зручний інтерфейс, вбудований термінал і інтеграція з Git. На відміну від Android Studio чи IntelliJ IDEA, VS Code менш вимогливий до ресурсів і запускається швидше, хоча поступається за кількістю вбудованих засобів рефакторингу. Це зумовлено орієнтацією на зовнішні розширення замість повноцінної внутрішньої системи аналізу коду. Альтернативами є Android Studio − офіційне середовище для Flutter, яке потребує більше ресурсів, або редактори на зразок Atom чи Sublime Text з обмеженим функціоналом. Вибір VS Code у цьому проєкті обґрунтований його універсальністю, безкоштовністю та активною спільнотою користувачів, що забезпечує швидкий доступ до підтримки та численних ресурсів.

Для контролю версій та колаборації використовується GitHub. GitHub – хмарна платформа для зберігання Git-репозиторіїв із вбудованими засобами спільної роботи (issues, pull requests тощо). Він широко поширений: близько 100 млн розробників по всьому світу використовують GitHub, а понад 90% компаній із списку Fortune100 мають на ньому свій код [15]. Це гарантує надійний сервіс і підтримку. Через інтеграцію GitHub Actions з Firebase можна автоматизувати CI/CD: при внесенні змін у репозиторій скрипт CI (GitHub Actions) автоматично компілює проєкт і розгортає його на Firebase Hosting чи інших сервісах.

Для GitHub існують альтернативи: GitLab (подібна функціональність, з власним CI/CD) і Bitbucket (частіше використовується в екосистемі Atlassian). Однак GitHub є найбільшим за спільнотою та кількістю інтеграцій, що спрощує співпрацю команди над проєктом. Також обрано розгортання через GitHub Pages (для статичних сайтів) або Firebase Hosting.

Висновок до розділу 1

Огляд існуючих мобільних платформ (Google Classroom, Moodle, Edmodo, Student Portal тощо) показав, що вони частково вирішують окремі задачі − організацію завдань, спілкування чи перегляд розкладу, − але не пропонують комплексного рішення для управління ІНП. Вони часто мають обмежену інтеграцію з університетськими системами та недостатню адаптивність до специфічних потреб студентів, що підкреслює необхідність розробки власного інструменту.

Для реалізації обрано технологічний стек, який оптимально відповідає поставленим вимогам: Flutter для кросплатформового інтерфейсу з адаптивним дизайном, Firebase для хмарного зберігання даних, автентифікації та реального часу (Firestore), а також Google Sheets для зручного введення та оновлення структурованих даних викладачами. Ця комбінація забезпечує високу продуктивність, безпеку, масштабованість і мінімальні витрати на підтримку, спираючись на готові рішення та інтеграцію з існуючими університетськими ресурсами. Використання GitHub для контролю версій і Firebase Hosting для розгортання веб-версії додатково полегшує розробку та оновлення.

РОЗДІЛ 2. ПРОЕКТУВАННЯ МОДЕЛІ МОБІЛЬНОЇ ПЛАТФОРМИ

2.1 Загальна архітектура та компоненти системи

Проєктування архітектури мобільної платформи є одним із ключових етапів розробки, що визначає структуру системи, взаємодію її компонентів, а також забезпечує гнучкість, масштабованість і подальшу підтримуваність програмного продукту.

У рамках даної роботи архітектурна модель структурування коду базується на підході MVVM (Model–View–ViewModel), який відповідає сучасним рекомендаціям щодо розробки застосунків середнього та великого масштабу на базі Flutter.

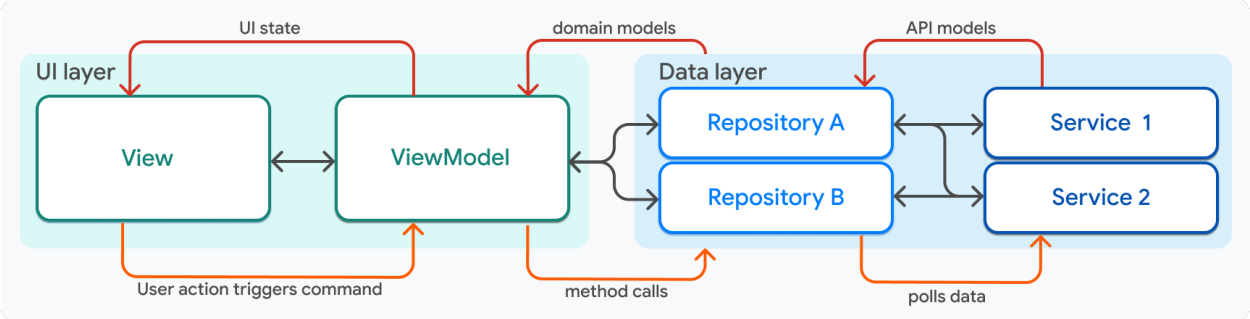


Рис. 2.1 *Діаграма архітектури MVVM у Flutter*

Вибір архітектури MVVM (рис. 2.1) обґрунтовується її здатністю чітко розмежовувати відповідальність між трьома основними компонентами: інтерфейсом користувача (View), логікою відображення станів (ViewModel) та бізнес-логікою разом із взаємодією з джерелами даних (Model). Така структура сприяє підвищенню підтримуваності коду, полегшує процес тестування та впровадження змін у подальшій експлуатації програмного продукту.

Структурні компоненти системи

- View (інтерфейс користувача) – забезпечує графічне відображення елементів застосунку та організовує взаємодію користувача з системою, зокрема функціонал перегляду оцінок, доступ до індивідуального навчального плану та профілю.
- ViewModel – відповідає за керування станами інтерфейсу, зокрема за оновлення списку дисциплін після отримання даних, обробку

подій користувача, таких як натискання кнопок або перемикання між вкладками.

- Model − реалізує бізнес-логіку та забезпечує взаємодію з віддаленими джерелами даних, зокрема Firebase Cloud Firestore, де зберігається актуальна інформація про студентів, дисципліни та оцінки (Cloud Firestore Overview, Google Firebase).

Дані завантажуються безпосередньо із Firestore через мобільний клієнт. Для організації синхронізації з Google Sheets застосовуються Google Apps Script, які можуть виконуватися як вручну, так і за розкладом.

Архітектура системи передбачає чітко визначені взаємозв'язки між її основними компонентами:

1. Flutter-застосунок та Firebase: мобільний і веб-клієнти на Flutter використовують офіційні FlutterFire-бібліотеки (firebase_auth, cloud_firestore, firebase_database) для аутентифікації користувачів і доступу до бази даних Firestore, що забезпечує безпечний і ефективний обмін інформацією.

2. Google Sheets та Firebase: обмін даними між Google Таблицями і Firebase реалізовано за допомогою Google Apps Script, що зчитує інформацію з Google Sheets і оновлює Firestore. Flutter-застосунок отримує дані виключно з Firestore, без прямої взаємодії з Google Sheets або Apps Script.

3. Кодова база, GitHub та Firebase Hosting: репозиторій з кодом розміщено на GitHub. Кожен пуш у гілку main ініціює процес CI/CD через GitHub Actions, який автоматично виконує збірку Flutter Web-застосунку і розгортає його на Firebase Hosting. Для мобільних версій (iOS/Android) код також зберігається в репозиторії, однак збірка і публікація у відповідних магазинах застосунків здійснюється вручну.

4. Google Workspace і система: викладачі використовують хмарні сервіси Google Workspace (Gmail, Calendar, Meet тощо) для організації навчального процесу. Пряма інтеграція з Flutter-застосунком реалізована лише через Google Sheets; інші сервіси функціонують автономно і не мають безпосереднього зв'язку з кодовою базою.

Таким чином, впровадження архітектурної моделі MVVM у поєднанні з інтеграцією хмарних сервісів Firebase та Google Sheets забезпечує створення надійної, масштабованої та зручно підтримуваної мобільної платформи.

2.2 Забезпечення захисту персональних даних користувачів

Мобільна платформа опрацьовує персональні дані студентів − такі як прізвище, ім’я, електронна адреса, група, оцінки, індивідуальний план навчання − тому надзвичайно важливим є питання їхнього захисту.

Для зберігання та обробки персональних даних використовується Firebase Cloud Firestore, який сертифікований за міжнародними стандартами безпеки, зокрема ISO/IEC 27001, SOC 2, GDPR. Використання сервісів Google дає змогу довіряти збереженню та обробці даних на високому рівні, адже компанія інвестує значні ресурси у забезпечення безпеки, конфіденційності та відповідності нормативним вимогам.

Процес автентифікації реалізується через Google Sign-In, що дозволяє забезпечити безпечний вхід без зберігання паролів на стороні застосунку або сервера [13]. Це дозволяє користувачу входити в систему через свою освітню пошту, що є також елементом перевірки його приналежності до навчального закладу. Надання довіри Google у питанні автентифікації гарантує надійність та захищеність цього процесу.

Ключові заходи безпеки:

- Обмеження доступу: доступ до навчальної інформації отримує лише автентифікований користувач, і лише до своїх власних даних. Це реалізовано через правила безпеки Firebase Firestore [6], [7].
- Мінімізація даних: зберігаються лише необхідні дані для функціонування системи.
- Шифрування: усі дані передаються за протоколом HTTPS та зберігаються у зашифрованому вигляді [4].

Таким чином, захист персональних даних користувачів забезпечується не лише технічними засобами, а й завдяки довірі до перевірених зовнішніх платформ, таких як Google, які постійно оновлюють свої системи безпеки та підтримують високий рівень захисту інформації.

2.3 Розробка інтерфейсу користувача

Інтерфейс мобільного застосунку розроблено з використанням Google Material Design 3 у середовищі Flutter. Material Design – відкрита система дизайну від Google, де остання версія Material 3 підтримує створення персоналізованих, адаптивних і виразних інтерфейсів. Наявність підтримки Material 3 у Flutter (за замовчуванням у Flutter 3.16 та новіших) надає можливості динамічного підбору кольорів, підвищеної доступності та адаптації макета під різні розміри екранів. Для забезпечення єдності стилю та послідовності оформлення всі сторінки застосунку використовують загальну тему (ThemeData) Flutter. Глобальна тема задає скін та колірну палітру компонентів (барів, кнопок, тексту тощо) на всіх екранах програми, що гарантує конзистентність візуального стилю [9].

Структура інтерфейсу охоплює чотири основні сторінки: *Головна, Оцінки, Поточний семестр* та *Профіль* користувача. Кожна з цих сторінок виконує певну функцію в системі та оформлена у рамках єдиного дизайну згідно з Material Design 3. Використання глобальної теми та єдиної системи стилів забезпечує, що колірні схеми, шрифти та компоненти управління виглядають однорідно на всіх розділах застосунку.

Для навігації між основними розділами застосунку застосовується нижня панель навігації – компонент BottomNavigationBar. Згідно з документацією Flutter, цей матеріальний віджет призначений для відображення внизу екрана декількох пунктів (звичайно від трьох до п’яти) для перемикання між різними екранами застосунку. BottomNavigationBar складається з іконок і підписів (NavigationDestination) і «забезпечує швидку навігацію між представленнями верхнього рівня додатка». У Flutter він

зазвичай передається властивості bottomNavigationBar віджета Scaffold, що автоматично розміщує його внизу екрану [8]. Кожен пункт на нижній панелі відповідає одній з чотирьох основних сторінок, забезпечуючи користувачеві миттєвий доступ до ключових функцій застосунку.

З погляду UX-дизайну та логіки взаємодії такий підхід є цілком виправданим і зручним. Розміщення навігаційних елементів у нижній частині екрана корисне для мобільних пристроїв, адже дозволяє користувачу інтуїтивно дістатись до основних пунктів меню великим пальцем під час звичайного тримання смартфона. Така нижня навігація відповідає сучасним рекомендаціям адаптивного дизайну і робить інтерфейс більш інтуїтивно зрозумілим. Водночас аналогічна структура навігації залишається зручною і у веб-версії додатку: вона наочно виокремлює головні розділи та спрощує доступ до них, що підвищує юзабіліті сайту. Таким чином, використання Flutter та Material Design 3 у поєднанні з нижньою панеллю навігації забезпечує адаптивний, послідовний і користувацький дружній інтерфейс застосунку.

2.4 Планування функціональних можливостей застосунку

Під час проєктування мобільного застосунку особлива увага приділялася попередньому визначенню функціональних можливостей, які б відповідали реальним потребам здобувачів вищої освіти в організації та контролі власного навчального процесу. Основною метою цього етапу стало формування чіткої структури застосунку, що забезпечуватиме доступ до актуальної, персоналізованої та структурованої інформації. На основі аналізу навчального середовища студентів та особливостей освітнього процесу було окреслено перелік функціональних блоків, які мають формувати логічну архітектуру майбутнього застосунку.

Одним із ключових елементів структури визначено модуль перегляду індивідуального навчального плану. У межах цього розділу передбачалося відображення пройдених та запланованих дисциплін, упорядкованих за

навчальними роками чи семестрами. Також планувалося надання користувачу можливості ознайомитися з інформацією про викладачів, які закріплені за відповідними курсами. Особливу увагу було приділено зручності фільтрації навчального навантаження, що мало сприяти кращому сприйняттю структури освітнього процесу.

Окремо розглядалася функціональність, пов’язана з академічною успішністю студента. Планувалося створити електронний аналог залікової книжки, у якому студент зможе переглядати отримані оцінки, типи підсумкового контролю (екзамен, залік тощо), а також викладачів, що оцінювали відповідні дисципліни. Такий підхід мав на меті забезпечення прозорості та зручності у відстеженні результатів навчання.

Для полегшення орієнтації у межах поточного семестру було заплановано окремий функціональний блок. Його інформаційне наповнення передбачало перелік дисциплін, що вивчаються саме у цей період. Така структура мала б забезпечити стислий і водночас змістовний огляд актуального навчального навантаження студента.

У контексті інформаційної взаємодії між студентом та навчальним закладом розглядалася інтеграція блоку новин та оголошень. Передбачалося, що в цьому розділі буде зібрана інформація про актуальні події, важливі оголошення та інші повідомлення адміністрації. Контент, включно з візуальним наповненням (банери, фотографії), не створюється у межах застосунку, а синхронізується з наявними джерелами, що вже функціонують у межах інформаційної екосистеми закладу освіти.

Важливою складовою структури було також передбачено персональний профіль користувача. Його концепція базувалася на ідеї створення цифрового еквівалента студентського профілю, що включатиме такі дані, як прізвище, ім’я, по батькові користувача, освітній рівень, спеціальність, курс, група та форма навчання.

Окрім основних модулів, на етапі планування розглядалася також можливість впровадження окремого розділу для перегляду розкладу занять.

Ідея передбачала відображення розкладу за днями тижня з можливістю фільтрації за курсом або академічною групою. Водночас, з огляду на пріоритетність інших функцій та обмеження в ресурсах, ухвалено рішення відкласти реалізацію цього функціонального блоку до подальших етапів розробки.

Загальне планування функціональності здійснювалося з урахуванням принципів інтуїтивної навігації, логічної структури інтерфейсу та мінімізації кількості дій, необхідних для досягнення користувацької цілі. Визначальним технологічним рішенням із самого початку стало інтегроване використання інструментів Google Workspace. У якості основи для роботи з даними було заплановано застосування Google Таблиць та, в загальному, Google Диск як джерела структурованої та централізованої інформації (наприклад, дані про дисципліни, оцінки, розклад). Використання Google Apps Script передбачалося для автоматизації оновлення даних, обробки запитів, взаємодії між окремими сервісами та забезпечення синхронізації з мобільним застосунком у режимі реального часу.

Застосування цього технологічного рішення забезпечувало глибоку інтеграцію мобільного застосунку з уже наявною, хоча й початковою, цифровою інфраструктурою кафедри. Це дозволяло організувати надійне зберігання даних, ефективне адміністрування та своєчасне оновлення інформації без потреби у додаткових ресурсах або складних технічних рішеннях.

Уся концепція застосунку була сформована виключно з орієнтацією на потреби студентської аудиторії. Початково не передбачалося створення функціоналу для адміністративного або викладацького використання, що дозволило зосередитися на створенні адаптованого та ефективного інструменту для повсякденного освітнього використання.

2.5 Проектування бази даних: структура та моделювання

У межах розробки мобільного застосунку для формування індивідуального навчального плану здобувача вищої освіти особлива увага приділяється проєктуванню та структурі бази даних. Основою інформаційної моделі слугують таблиці Google Sheets, які містять вихідні дані щодо студентів, академічної успішності та навчальних планів. Вихідна інформація розподілена між двома основними папками: «Магістратура» та «Бакалаврат» (рис. 2.2).

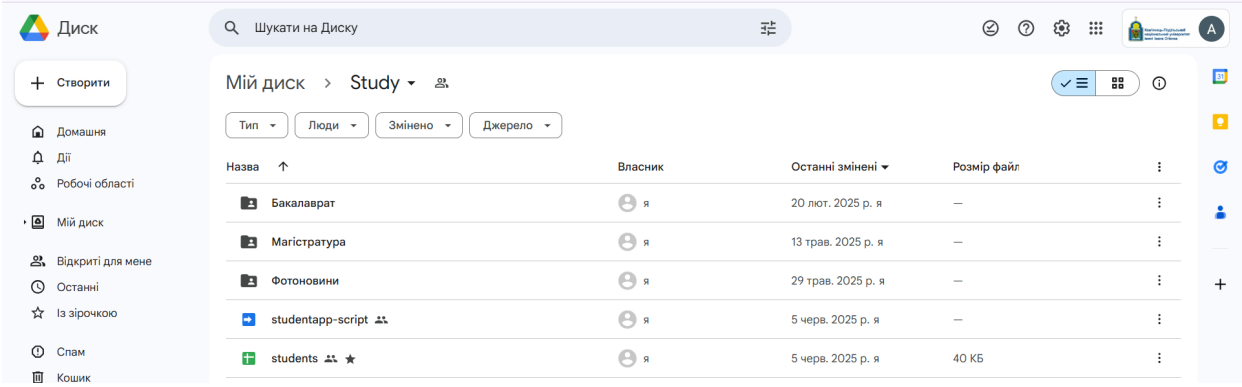


Рис. 2.2 *Структура папки Study на Google Диску*

Усередині кожної папки створено вкладені директорії відповідно до назв академічних груп (наприклад, KN1–B21 (рис. 2.3)), які містять три основні типи файлів:

– академічні довідки по групі (по семестрах);

– списки студентів;

– навчальні плани (за роком вступу).

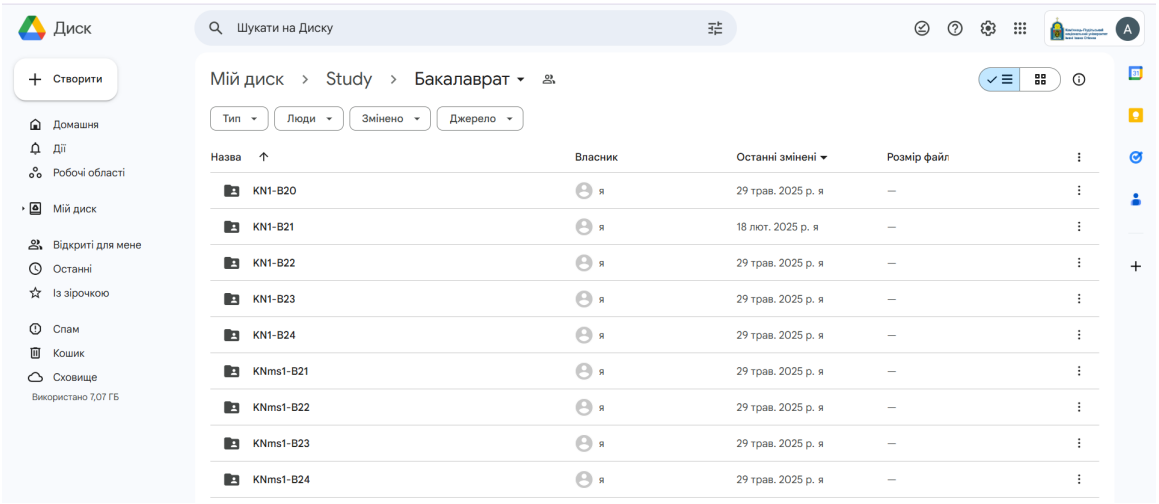


Рис. 2.3 *Папки навчальних груп у розділі «Бакалаврат» на Google Диску*

Обробка даних між Google Таблицями та Firestore реалізована за допомогою чотирьох скриптів (рис. 2.4), написаних на Google Apps Script:

1. З навчального плану до списку студентів. Автоматично переносить інформацію про дисципліни (назви, типи, семестри, форму контролю тощо) з файлу навчального плану до відповідного файлу списку студентів.
2. З академічної довідки до списку студентів. Синхронізує отримані оцінки студентів за дисциплінами з файла академічної довідки у файл списку студентів, прив'язуючи їх до конкретних предметів.
3. Із списку студентів до Google-таблиці students. Формує єдину агреговану таблицю з усіх груп у зручному для подальшого експорту форматі: кожен рядок містить дані про студента та його оцінки з усією необхідною інформацією.
4. З Google-таблиці Students до Firestore. Завантажує оброблені дані до Firebase Firestore. При цьому автоматично створюється структура з колекціями, документами та вкладеними записами.

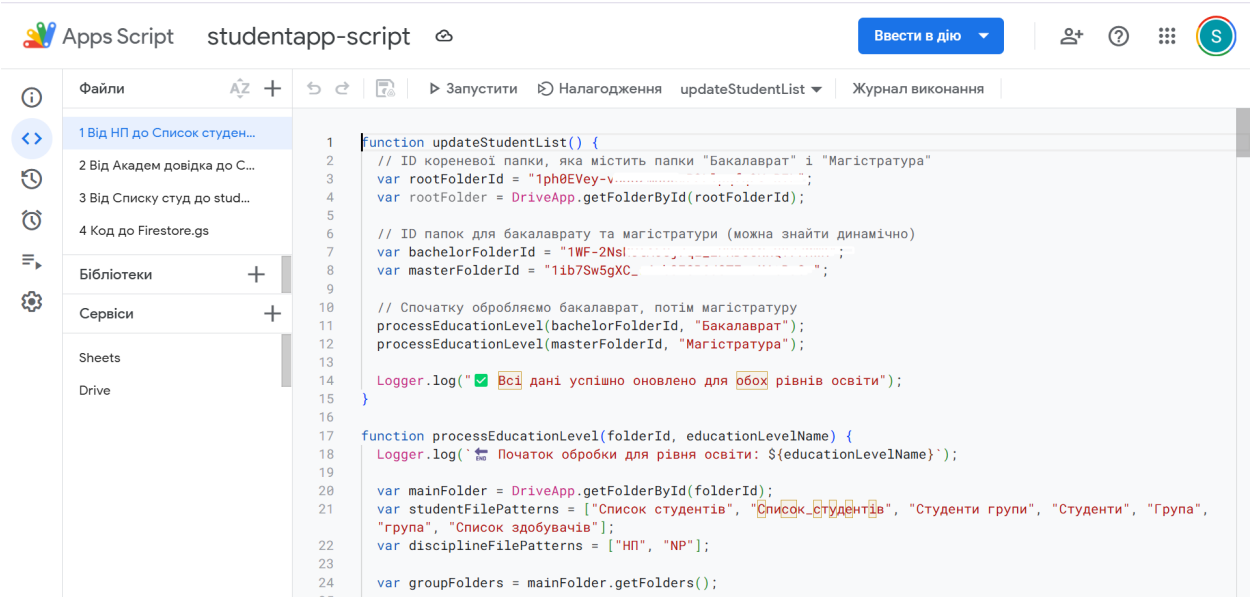


Рис. 2.4 *Зберігання скриптів у середовищі Apps Script*

У Firestore структура бази організована у вигляді вкладених колекцій та документів. Кореневою колекцією є groups, в якій зберігаються документи з назвами академічних груп (наприклад, KN1-B21, KN1-M23 тощо). Кожен

документ містить вкладену колекцію students_[назва групи], де зберігаються окремі документи студентів (рис. 2.5).

Кожен студентський документ включає основну інформацію:

– student_name – ПІБ студента;
– student_email – електронна адреса;
– education_level – освітній рівень.

Крім цього, студент має вкладену структуру additional_records, що є масивом об’єктів з деталями щодо кожної дисципліни, яку студент вивчав. Кожен запис містить:

– discipline_name – назва дисципліни;
– discipline_type – тип дисципліни (обов’язкова або вибіркова);
– exam_type – тип підсумкового контролю;
– grade – оцінка;
– semester – семестр вивчення;
– teacher_email – електронна пошта викладача;
– teacher_name – ПІБ викладача;
– year – навчальний рік.

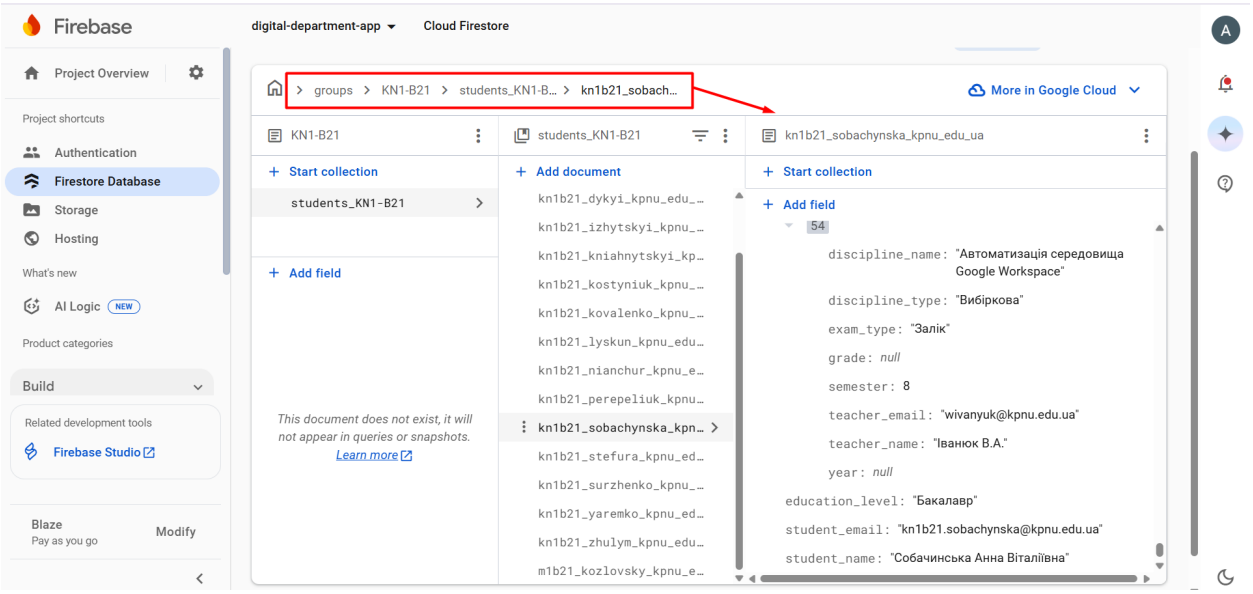


Рис. 2.5 *Архітектура зберігання даних в Firestore Database*

Застосунок здійснює автентифікацію користувача через Google Sign-In. Після входу за корпоративною поштою (з доменом @kpnu.edu.ua), система

виконує пошук відповідного запису в Firestore. Спочатку перевіряється група, яка відповідає префіксу електронної адреси за шляхом:

/groups/[назва групи]/students_[назва групи]/[email без спецсимволів].

Якщо запис не знайдено − здійснюється пошук по всіх групах бази.

Таким чином, система забезпечує гнучке, автоматизоване та масштабоване зчитування структурованої інформації з Google Sheets у Firestore з подальшою інтеграцією у мобільний застосунок. Це дозволяє забезпечити студентів актуальними персоналізованими даними про їхню освітню траєкторію.

Висновки до розділу 2

Проєктування моделі мобільної платформи ґрунтується на архітектурі MVVM, яка забезпечує чітке розмежування компонентів системи: інтерфейсу користувача (View), логіки відображення (ViewModel) та бізнес-логіки (Model). Такий підхід сприяє підвищенню підтримуваності, тестованості та масштабованості коду, що є критично важливим для розробки застосунків середнього та великого масштабу. Використання Flutter як основної технології розробки забезпечує кросплатформну сумісність і високу продуктивність, а застосування Material Design 3 гарантує створення адаптивного, інтуїтивно зрозумілого та візуально конзистентного інтерфейсу.

Інтеграція з хмарними сервісами, зокрема Firebase Cloud Firestore та Google Sheets, за допомогою Google Apps Script забезпечує автоматизовану синхронізацію даних, що дозволяє підтримувати актуальність інформації про студентів, дисципліни та оцінки. Firebase, як сертифікована платформа, гарантує високий рівень безпеки завдяки шифруванню даних, обмеженню доступу через правила безпеки та автентифікації через Google Sign-In, що відповідає міжнародним стандартам. Це забезпечує захист персональних даних студентів і надійність системи.

Функціональні можливості застосунку, такі як перегляд індивідуального навчального плану, академічної успішності, поточних дисциплін та профілю

користувача, розроблені з урахуванням потреб студентів, що сприяє зручності та прозорості в управлінні освітнім процесом. Структура бази даних, побудована на основі вкладених колекцій у Firestore та агрегованих даних із Google Sheets, забезпечує гнучке та ефективне управління інформацією. Загалом, запропонована модель мобільної платформи є надійною, масштабованою та орієнтованою на користувача основою для створення сучасного освітнього інструменту.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОБІЛЬНОЇ ПЛАТФОРМИ

3.1 Реалізація інтерфейсу користувача

Інтерфейс користувача мобільної платформи розроблено з урахуванням принципів інтуїтивності, доступності та адаптивності, що забезпечує його ефективність як інструменту підтримки студентів у процесі навчання. Застосунок надає користувачам зручний доступ до індивідуального навчального плану, оцінок, інформації про поточні дисципліни та персонального профілю, поєднуючи функціональні можливості із сучасними дизайнерськими підходами. Візуальне оформлення інтерфейсу ґрунтується на стандартах Material Design 3, що проявляється у використанні чіткої карткової структури, заокруглених кутів елементів та виразної кольорової палітри. Використання яскравого помаранчевого відтінку (0xFFFF940) у комбінації з білим і сірим фоном створює енергійний, водночас лаконічний дизайн, який підкреслює освітню спрямованість платформи та забезпечує комфортне сприйняття інформації.

Константи кольорів для мобільного застосунку:

```
abstract final class AppColors {
  // Основні кольори
  static const primary = Color(0xFFFF9406);
  static const primaryDark = Color(0xFFD97905);
  static const appBarBackground = Color(0xFFFF9406);
  static const scaffoldBackground = Colors.grey;
  // Тексти та іконки
  static const iconColor = Colors.white;
  static const textPrimary = Color(0xFF212121);
  static const textSecondary = Colors.grey;
  // Колір для посилань
  static const linkColor =  Colors.grey;
  static const primaryyellow =Color.fromARGB(255, 255, 180, 6);
}
```

Навігаційна система реалізована через фіксовану нижню панель (BottomNavigationBar), що включає чотири основні розділи: «Головна», «Оцінки», «Поточний семестр» та «Профіль». Кожен розділ позначено інтуїтивно зрозумілими піктограмами та текстовими мітками, при цьому

активна вкладка відображається помаранчевим маркером, що сприяє швидкій орієнтації користувача в інтерфейсі. Така організація навігації дозволяє ефективно переходити між ключовими функціями, особливо у мобільному середовищі, де взаємодія переважно здійснюється великим пальцем. У веб-версії застосунку навігаційна панель зберігає аналогічний функціонал, що забезпечує послідовність користувацького досвіду на різних платформах.

Початковим етапом взаємодії з платформою є екран авторизації, який містить логотип університету, назву закладу та аватар користувача. Центральною функціональною складовою є кнопка «Увійти через Google», стилізована в помаранчевому кольорі з заокругленими кутами (рис. 3.1), що забезпечує швидкий та безпечний вхід через Firebase Authentication з використанням Google Sign-In [13].

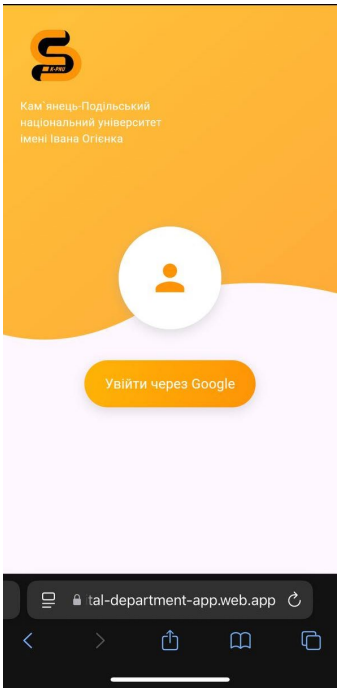
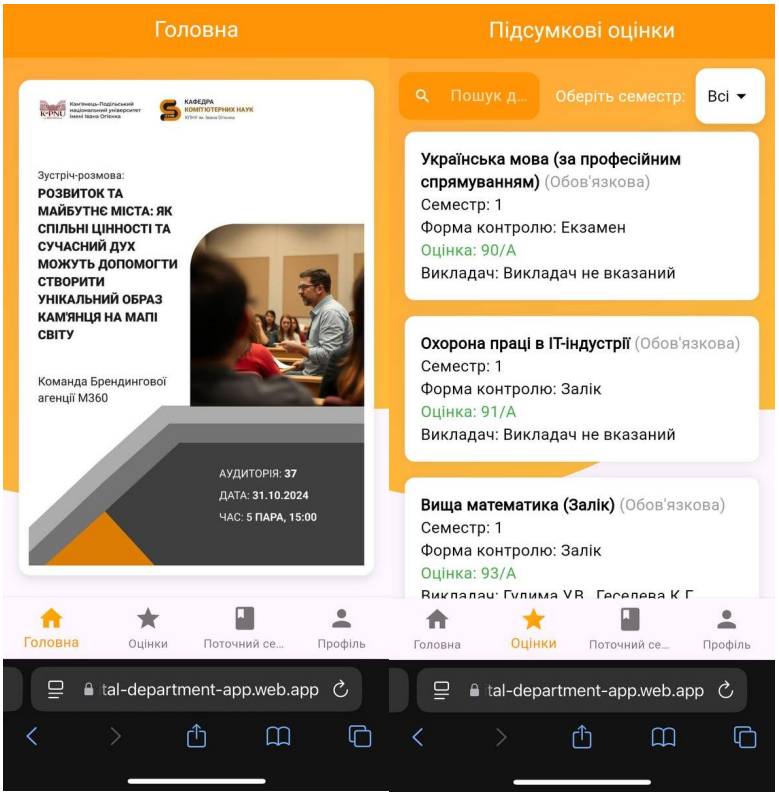


Рис. 3.1 *Вигляд екрану авторизації*

Після успішної авторизації користувач потрапляє на головний екран, який виконує функцію інформаційного центру. На цьому екрані відображаються актуальні новини, події та оголошення у вигляді карток, що містять зображення, назву, короткий опис, а також інформацію про час і місце проведення. Така організація інформації дозволяє студентам оперативно

ознайомлюватись з важливими повідомленнями, не відволікаючись на зайві дії, а також сприяє структурованому сприйняттю контенту.



а) *б)*

Рис. 3.2 *Головна сторінка застосунку а) та сторінка «Оцінки» б)*

Розділ «Оцінки» є центральним елементом застосунку, який надає користувачам повний огляд їхньої академічної успішності. Дані представлені у вигляді карток з білим фоном, де кожна картка містить назву дисципліни, семестр, форму контролю, оцінку за шкалою A–F із відповідним кольоровим маркуванням (наприклад, зелений колір позначає позитивні результати) та ім'я викладача. У верхній частині екрану розміщено поле пошуку та випадаючий список для фільтрації за семестрами, що дозволяє отримати доступ до інформації про оцінки за всі роки навчання, роблячи цей модуль важливим інструментом для аналізу навчальних досягнень (рис. 3.2, *а*).

У розділі «Поточний семестр» надається можливість перегляду дисциплін, які вивчаються у поточному семестрі, із чіткою класифікацією на обов'язкові та вибіркові курси. Незавершені дисципліни відмічені маркером «Нездано», що сприяє контролю навчального прогресу та плануванню

подальших дій (рис. 3.2, *б*). Візуальне оформлення карток відповідає стилю розділу «Оцінки», забезпечуючи єдність і послідовність у дизайні інтерфейсу.

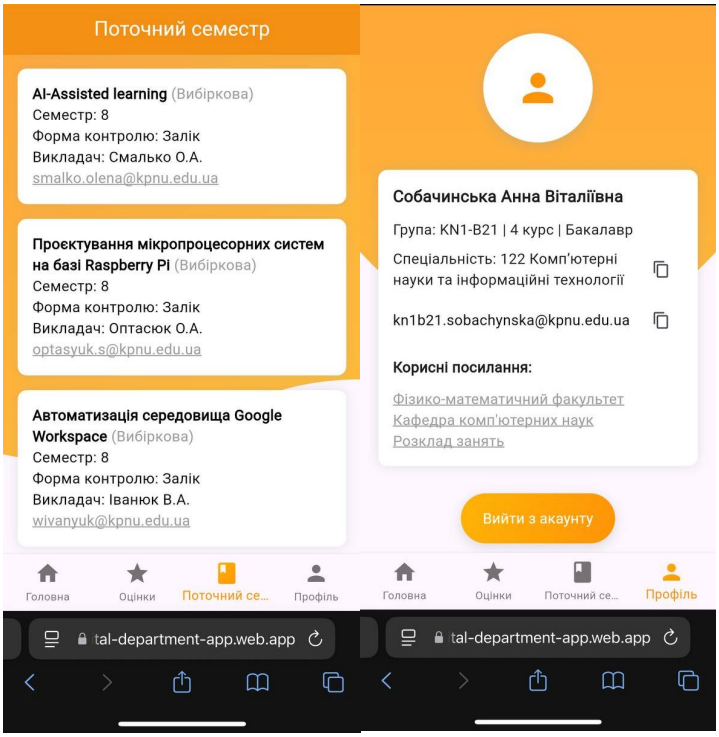


Рис. 3.3 *Сторінки «Поточний семестр» та «Профіль»*

Розділ «Профіль» містить персональні дані користувача: повне ім’я, групу, курс, спеціальність, рівень освіти та академічну електронну пошту. Додатково передбачено посилання на офіційні ресурси університету, зокрема сайти факультету, кафедри та розклад занять, що забезпечує швидкий доступ до релевантної інформації. Кнопка виходу з облікового запису, розташована в нижній частині екрану, забезпечує зручне завершення сесії (рис. 3.3).

Типографічна система інтерфейсу базується на шрифті без засічок (Roboto), із варіативною товщиною накреслення для заголовків, підзаголовків та основного тексту, що формує чітку ієрархію сприйняття інформації. Іконографія навігаційної панелі містить інтуїтивно зрозумілі символи – будинок для «Головна», документ для «Оцінки», силует людини для «Профіль» (рис. 3.4). Плавні анімації та швидка реакція інтерфейсу сприяють комфортному користувальницькому досвіду.

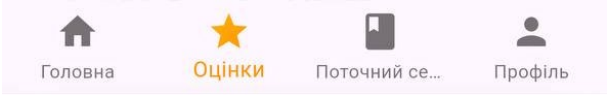


Рис. 3.4 *Вигляд навігаційної панелі*

Технічна реалізація здійснена із застосуванням фреймворку Flutter, що гарантує уніфіковане відображення на платформах Android (рис. 3.5) та iOS (рис. 3.6). Адаптивність макетів, швидкість відгуку та підтримка компонентів Material Design 3 забезпечують не лише функціональність, а й естетичну привабливість інтерфейсу.

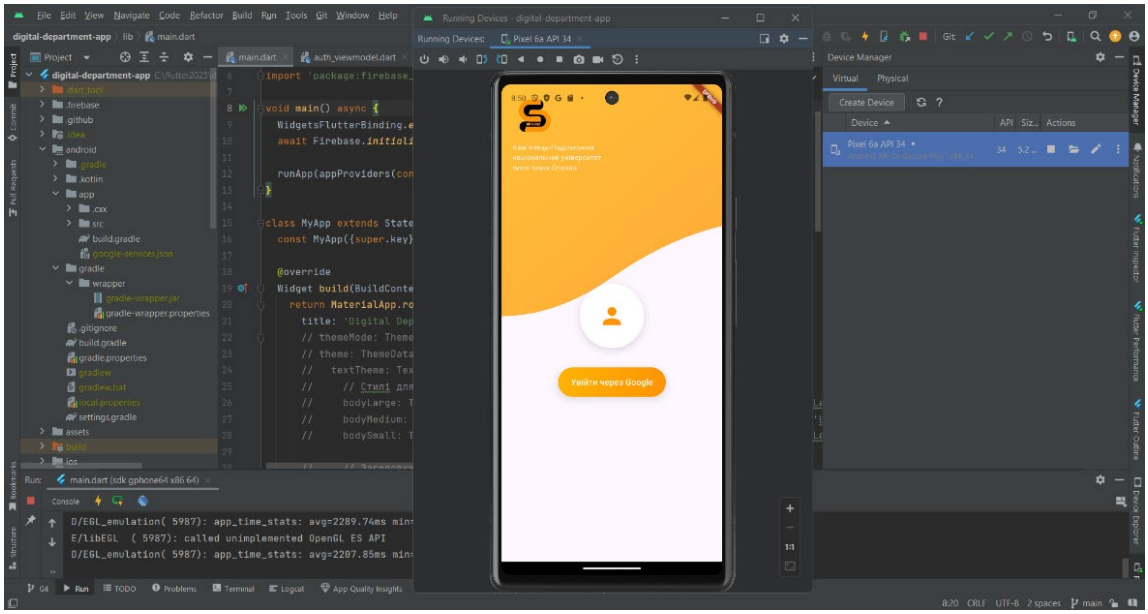


Рис. 3.5 *Запуск застосунку на емуляторі для Android*

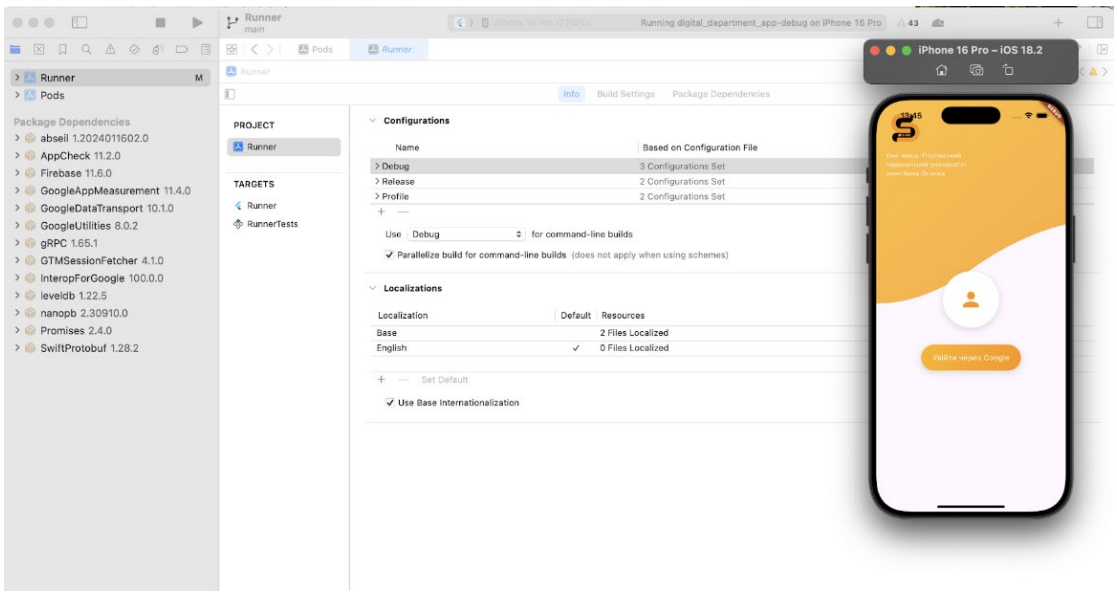


Рис. 3.6 *Запуск застосунку на емуляторі для iOS*

Отже, розроблений інтерфейс користувача поєднує сучасні тенденції, інтуїтивну навігацію та хорошу функціональність, що створює ефективний та зручний інструмент для студентів. Оформлення тексту застосунку забезпечує зручне та зрозуміле сприйняття інформації завдяки чітким шрифтам і логічній

структурі. Він забезпечує оперативний доступ до навчальної інформації, підтримує академічний процес та відповідає високим стандартам користувацького досвіду, формуючи надійну платформу для освітнього середовища.

3.2 Інтеграція автентифікації через Google

У рамках розробки мобільного застосунку для індивідуального навчального плану здобувача вищої освіти було реалізовано автентифікацію користувачів за допомогою Google-акаунтів через сервіс Firebase Authentication. Такий підхід забезпечує надійну і безпечну ідентифікацію студентів, використовуючи сучасні хмарні технології та інфраструктуру компанії Google.

Для реалізації Google Sign-In у мобільному застосунку було виконано низку кроків у Firebase Console:

1. Створення проєкту у Firebase Console
   Вебінтерфейс Firebase Console дозволяє створити новий проєкт, який автоматично інтегрується з Google Cloud Platform.
2. Підключення застосунку до Firebase. У розділі Project Settings було додано новий Android-додаток:
   - Вказано назву пакету (package name)
   - Завантажено файл конфігурації google-services.json, який було додано до проєкту Flutter у директорію android/app/
3. Увімкнення автентифікації через Google. У розділі Authentication → Sign-in method активовано Google як метод входу. Тут також налаштовано підтримку поштових доменів і базові параметри безпеки.

Додаткові налаштування для вебверсії. Для підтримки Google Sign-In у вебверсії застосунку необхідно створити OAuth 2.0 Client ID у Google Cloud Console:

1. Перехід до Google Cloud Console https://console.cloud.google.com/

2. Вибір проєкту, пов’язаного з Firebase. Необхідно переконатися, що вибрано той самий проєкт, який використовується у Firebase.
3. Увімкнення API Google Identity. У розділі APIs & Services → Library увімкнено Google Identity Services API.
4. Створення OAuth-клієнта. У розділі APIs & Services → Credentials було створено OAuth 2.0 Client ID типу Web Application з указанням авторизованих URI для перенаправлення, після чого згенерований Client ID інтегровано до налаштувань вебзастосунку [12].

Застосування автентифікації через Google забезпечує безпеку та конфіденційність даних, водночас надаючи низку важливих переваг:

- Підвищений рівень безпеки – автентифікація здійснюється безпосередньо через інфраструктуру Google, яка має сертифікацію відповідно до міжнародних стандартів безпеки.
- Відсутність необхідності зберігати паролі – застосунок не зберігає жодних облікових даних користувачів, що виключає можливість витоку конфіденційної інформації.
- Спрощення доступу для студентів – достатньо скористатися університетською поштою Google Workspace (@kpnu.edu.ua), щоб отримати миттєвий доступ до навчального профілю.

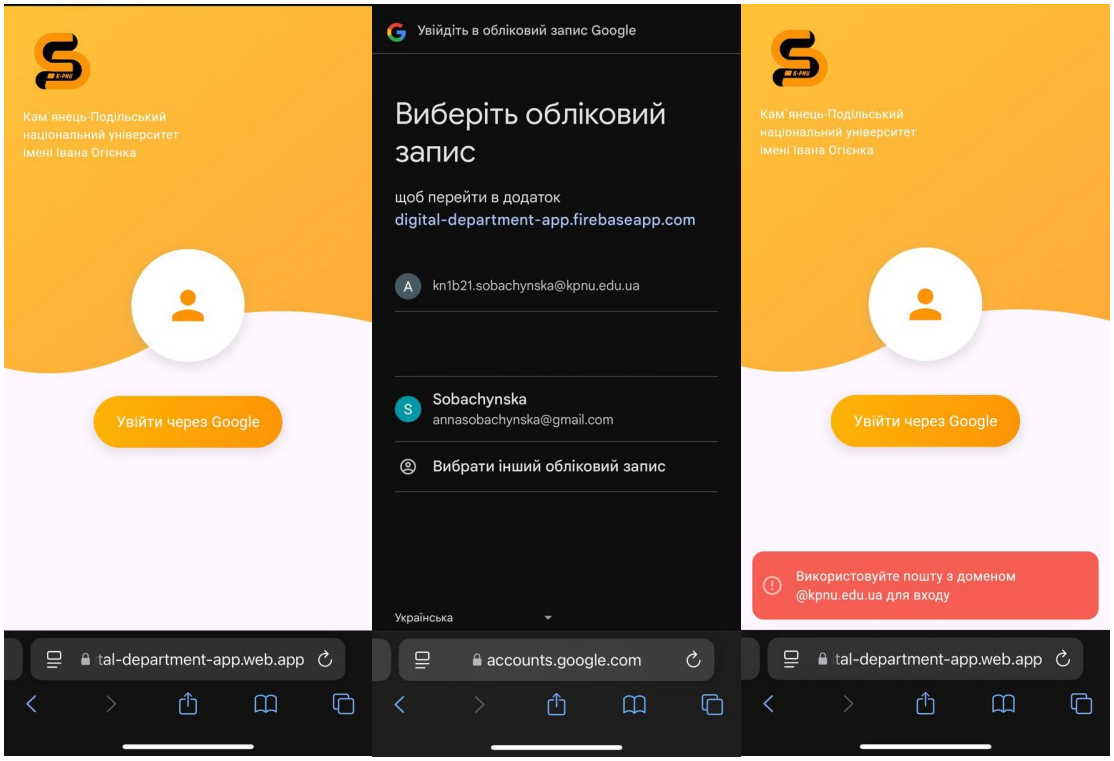
Таким чином, реалізація автентифікації через Google у поєднанні з Firebase забезпечує захист персональних даних студентів, зручність входу та інтегрованість з іншими Google-сервісами, що є важливою перевагою для освітнього цифрового середовища.

3.3 Реалізація функціональних модулів

Функціональні можливості мобільного застосунку розроблені відповідно до специфіки освітнього середовища та потреб користувача, яким у даному випадку є здобувач вищої освіти. Основною увагою при розробці функціональних модулів було забезпечення доступності, персоналізації,

інтеграції з цифровими сервісами закладу та створення комфортних умов для комунікації між студентом і адміністрацією/викладачами.

Автентифікація користувача відбувається через обліковий запис Google (рис. 3.7, *а* та *б*), що дозволяє реалізувати швидкий і безпечний вхід до системи, зберігаючи при цьому єдиний підхід до авторизації, типовий для сучасних освітніх платформ. Якщо користувач намагається увійти в застосунок не з корпоративного облікового запису (з доменом @kpnu.edu.ua), йому висвітиться відповідна помилка (рис. 3.7, *в*).



а) *б)* *в)*

Рис. 3.7 *Авторизація через Google: а) стартовий екран; б) вибір акаунта; в) помилка авторизації через некорпоративну пошту*

Подальший функціонал застосунку реалізовано через поділ на логічно структуровані екрани, між якими здійснюється перемикання за допомогою нижньої навігаційної панелі. Основними функціональними розділами виступають: «Головна», «Оцінки», «Поточний семестр» та «Профіль».

Головна сторінка застосунку виконує функцію інформаційного вузлу, однак не містить специфічних інструментів взаємодії з користувачем. Основною метою цієї сторінки є ознайомлення з актуальними подіями та

новинами навчального закладу. Дані для відображення зберігаються в Firebase Storage у папці «News». Файли підписані датою проведення заходу та збережені у форматі .png (рис. 3.8).

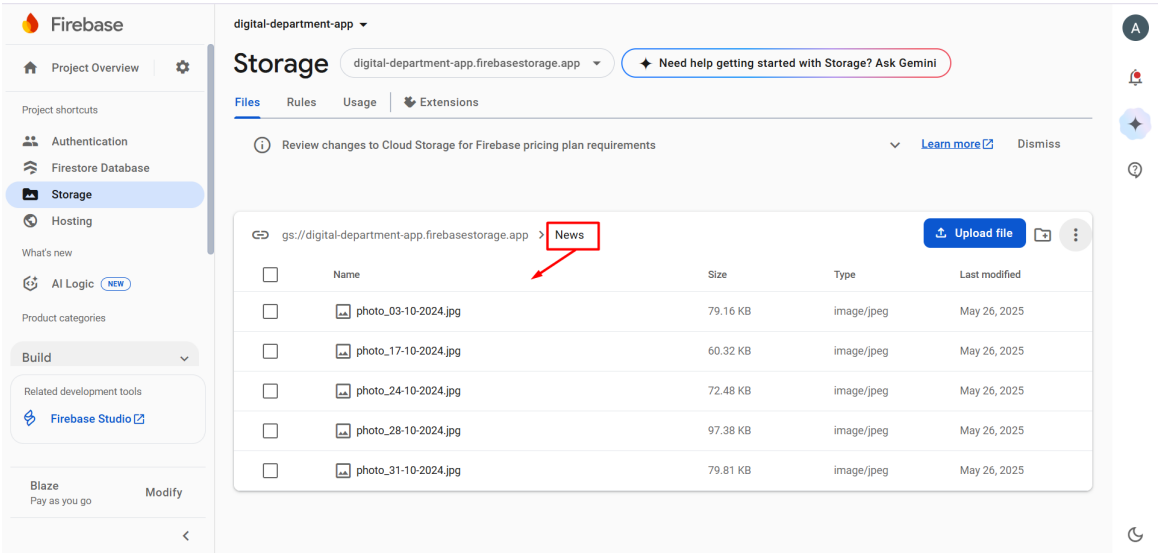


Рис. 3.8 *Зберігання фотокарток новин для Головної сторінки*

Перехід між сторінками реалізовано за допомогою фіксованої нижньої панелі з іконками, що дозволяє інтуїтивно перемикатися між основними функціональними модулями застосунку.

Найбільш насиченим з функціонального боку є розділ «Оцінки», що забезпечує доступ до персоналізованого академічного контенту студента. У цьому розділі відображається повний індивідуальний навчальний план здобувача вищої освіти з відповідними оцінками за пройдені дисципліни. Інформація подана у вигляді впорядкованого списку карток, що включають назви предметів, типи дисциплін (обов'язкові чи вибіркові), семестри, типи підсумкового контролю (екзамен, залік, курсова робота) та підсумкові бали. Така деталізація дає змогу студенту швидко оцінювати власну академічну успішність та динаміку навчання.

Однією з функціональних можливостей цього розділу є реалізація прямого зв'язку з викладачем. Для кожної дисципліни, що відображається у списку, зазначено електронну адресу викладача. У разі потреби студент має змогу ініціювати створення листа у Gmail безпосередньо із застосунку, натиснувши на адресу (рис. 3.9). Це значно спрощує комунікацію, підвищує

швидкість реагування на академічні запити та підтримує сталі канали зворотного зв’язку між учасниками освітнього процесу.

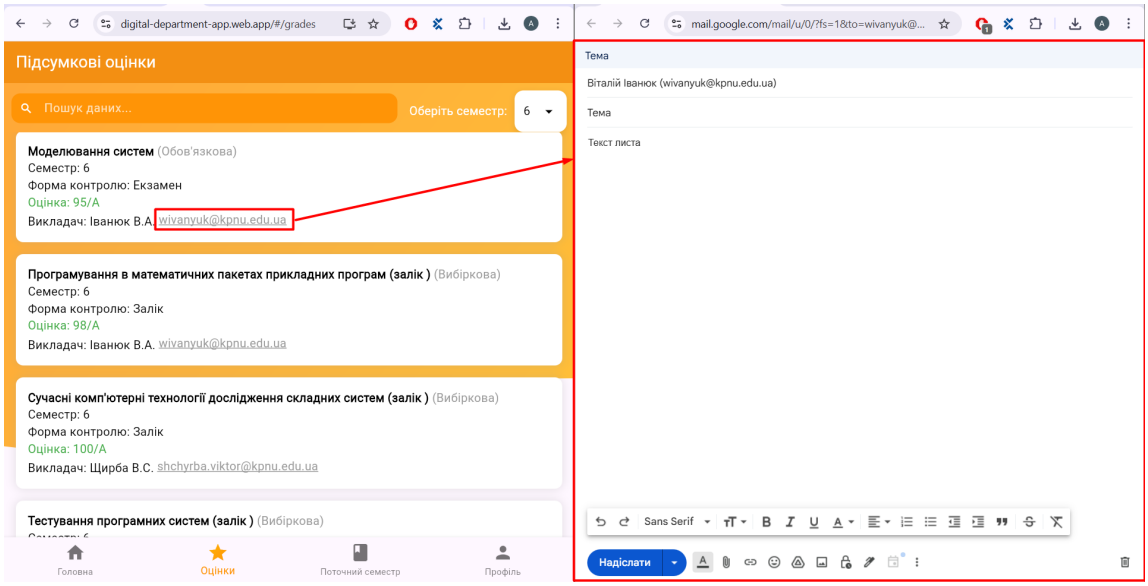


Рис. 3.9 *Демонстрація функціоналу створення листа Gmail після натискання на пошту викладача на платформі*

Крім того, функціонал розділу «Оцінки» доповнений пошуковим полем, що дозволяє здійснювати фільтрацію предметів за назвою, оцінками чи іншими потрібними полями окрім семестру. Такий підхід сприяє зручному доступу до конкретної інформації у разі великого обсягу даних в навчальному плані. Додатково передбачено модуль вибору семестру, за яким користувач бажає переглянути оцінки (рис. 3.10). Інтерфейс дозволяє як перегляд повної історії навчання за всі семестри, так і фільтрацію за окремими семестрами (від першого до восьмого або відповідно до наповнення навчального плану). Це надає користувачеві гнучкість у роботі з навчальною інформацією, дозволяючи фокусуватися на окремих періодах навчання.

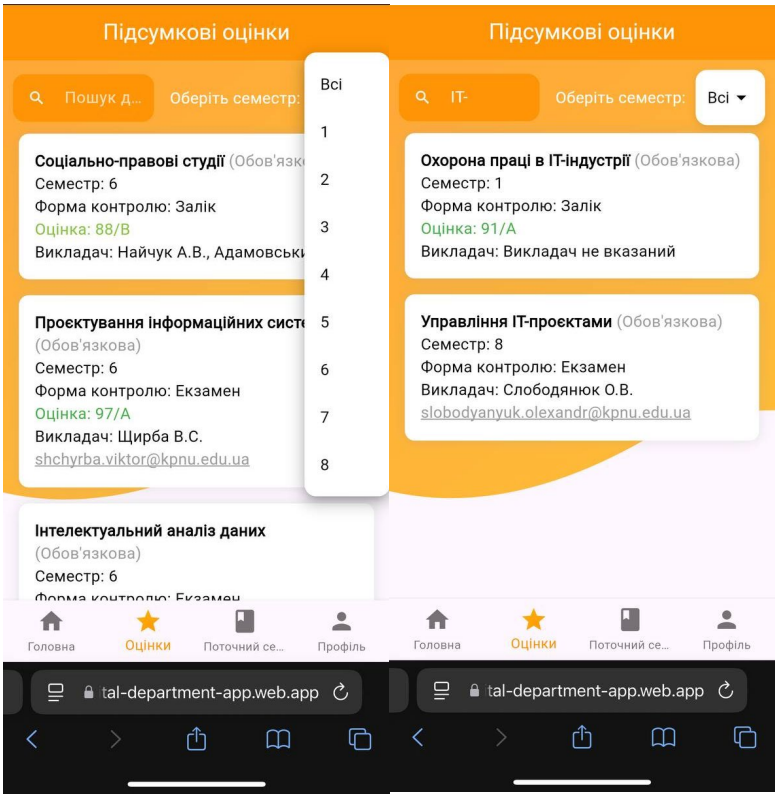


Рис. 3.10 *Функціонал сторінки «Оцінки»*

Наступний розділ під назвою «Поточний семестр» створений передусім для того, щоб студент міг швидко та легко переглянути інформацію саме за цей семестр без необхідності додатково вибирати його на попередній сторінці щоразу. Тут відображається список дисциплін, які студент вивчає в поточному навчальному періоді. Такий підхід допомагає зосередитися на актуальних завданнях і оцінках.

Сторінка «Профіль» містить розширену інформацію про користувача. Тут відображаються такі персональні дані: прізвище, ім’я, по батькові, номер академічної групи, курс, спеціальність, освітній рівень та університетська електронна пошта. За власними спостереженнями, було вирішено додати функцію копіювання назви спеціальності та шифру групи (рис. 3.11).

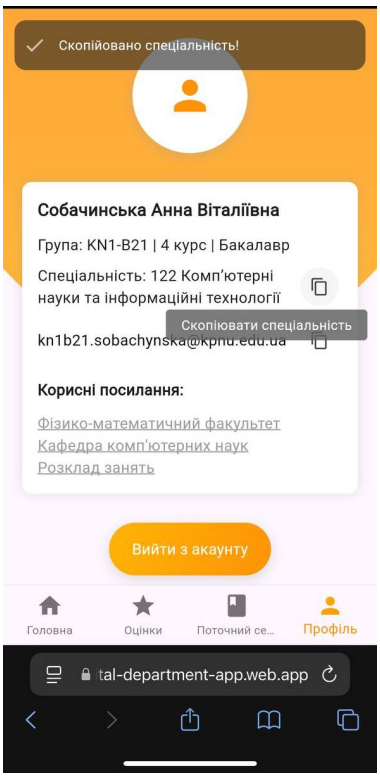


Рис. 3.11 *Функція копіювання на сторінці «Профіль»*

Окрім цього, у профілі реалізовано блок з корисними посиланнями, які спрямовують користувача на офіційні вебресурси університету, зокрема сайт кафедри, головну сторінку університету, а також розклад занять. Завдяки цьому студент може швидко отримати доступ до пов'язаної інформації, не виходячи з меж застосунку (рис. 3.12). Завершується функціонал профілю кнопкою виходу з облікового запису, яка дозволяє безпечно завершити сесію та вийти з системи.

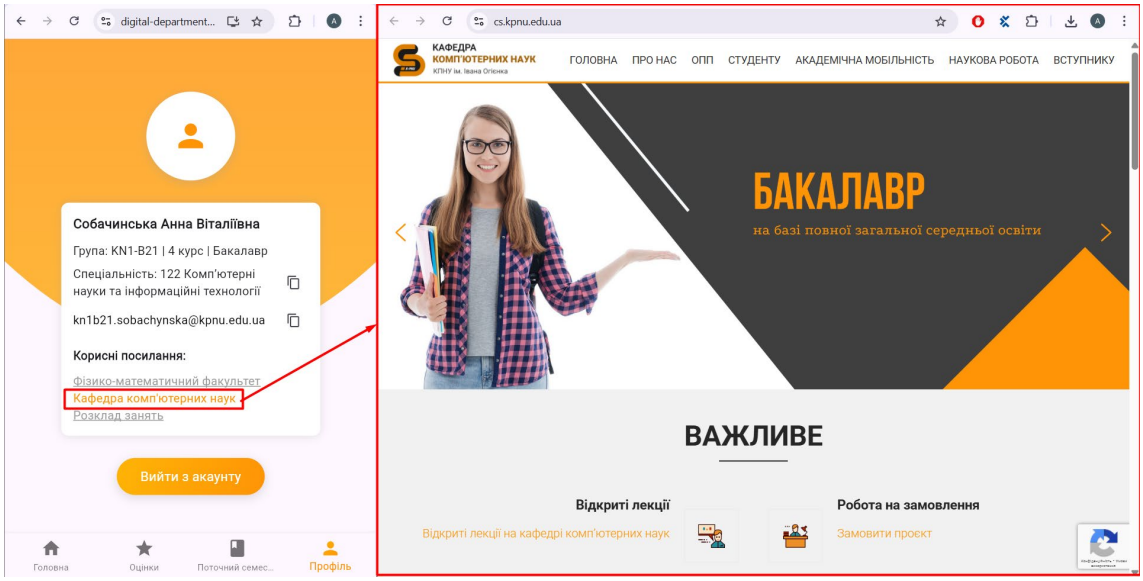


Рис. 3.12 *Перехід за посиланням на сторінці «Профіль»*

Усі функціональні модулі мобільного застосунку взаємодіють із базою даних Firestore, забезпечуючи динамічне оновлення та синхронізацію даних у реальному часі. Таке рішення сприяє безперервному доступу до актуальної інформації, навіть у випадку зміни оцінок, навчального плану чи особистих даних користувача. Архітектура застосунку дозволяє масштабувати функціональність у майбутньому, що забезпечує його гнучкість та відповідність змінним потребам освітнього процесу.

Таким чином, реалізація функціональних модулів мобільного застосунку демонструє комплексний підхід до створення зручного та ефективного інструменту для підтримки навчання. Інтеграція з сервісами Google, доступ до персоналізованого навчального плану, можливість фільтрації та пошуку дисциплін, налагодження зв’язку з викладачами та централізований доступ до важливих ресурсів університету − усе це формує сучасну цифрову освітню екосистему, орієнтовану на потреби студента.

3.4 Тестування та оптимізація мобільної платформи

Розробка мобільної платформи на основі фреймворку Flutter вимагала ретельного тестування для виявлення недоліків, забезпечення правильного відображення даних і підвищення стабільності роботи системи для різних користувачів. Тестування проводилося поетапно, охоплюючи індивідуальні та групові сценарії використання, а також передбачало вдосконалення архітектури платформи для підвищення її ефективності. Через використання Flutter тестування було безперервним процесом, спрямованим на перевірку сумісності з операційними системами та пристроями різних технічних характеристик. Тестування і вдосконалення платформи є ітеративним процесом, оскільки система потребує постійного оновлення для відповідності новим вимогам і розширення функціоналу. Аналіз виявлених проблем і впровадження заходів оптимізації сприяли покращенню функціональності, надійності та зручності платформи.

На першому етапі перевірявся функціонал платформи на власному обліковому записі, щоб переконатися в коректності базових функцій, таких як авторизація, відображення даних і навігація. Для ширшого тестування було залучено ще один обліковий запис із академічної групи випускниці, що дозволило перевірити відображення даних для іншого користувача. Цей етап підтвердив базову працездатність системи, оскільки серйозних проблем не було виявлено.

На другому етапі тестування розширилося за рахунок залучення більшої кількості користувачів із академічної групи KN1-B21, і меншої кількості представників інших академічних груп. Це допомогло виявити низку недоліків, пов’язаних із неправильним відображенням даних та адаптованістю. Безперервне тестування на різних платформах було необхідним через особливості Flutter, адже поведінка програми могла варіюватися залежно від операційної системи чи характеристик пристрою.

У процесі тестування було виявлено кілька вагомих недоліків, які проаналізовано та частково усунуто:

1. Некоректне відображення академічного статусу та семестру.

Під час тестування було виявлено, що платформа в деяких випадках показувала неправильний курс навчання та семестр для певних користувачів. Наприклад, для користувачів іншої академічної групи, які перебували на третьому курсі, платформа помилково відображала дані за восьмий семестр замість шостого, а в інших випадках вказувала невідповідний рік навчання. Ці проблеми виникали через помилки в логіці обробки даних, яка базувалася на аналізі поштових адрес користувачів і поточного часу, а також через неточності в алгоритмі синхронізації даних про поточний семестр. Аналіз причин цих недоліків дозволив внести зміни до логіки обробки та алгоритму синхронізації, що забезпечило коректне визначення академічного статусу та семестру. Повторне тестування підтвердило успішне усунення цих проблем.

2. Періодична помилка доступу до платформи.

У процесі тестування було виявлено періодичну помилку, яка виникала при спробі входу або повторного входу на платформу. Помилка мала такий вигляд:

Unable to process request due to missing initial state.

This may happen if browser sessionStorage is inaccessible or accidentally cleared. Some specific scenarios are:

Using IDP-Initiated SAML SSO.

Using signInWithRedirect in a storage-partitioned browser environment.

Ця помилка виникала нерегулярно: у деяких користувачів вона з’являлася час від часу, а в інших не спостерігалася. Аналіз показав, що проблема не пов’язана тільки з кодом, а й викликана зовнішніми факторами, такими як обмеження роботи sessionStorage у браузерах (Safari), зокрема в середовищах із ізольованим сховищем. Ця проблема, більш ніж на половину, є зовнішньою особливістю, оскільки її вирішення виходить за межі коду застосунка. Робота над пошуком додаткових рішень для цієї проблеми триває.

3. Часткове відображення даних.

Під час тестування було виявлено, що платформа не завжди забезпечує повне відображення даних для всіх користувачів. Іноді окремі академічні дані могли не відображатися або з’являлися некоректно внаслідок неточно сформованої логіки запитів до бази даних. У зв’язку з індивідуальними траєкторіями навчання студентів − зокрема зміною спеціальностей або груп, а також використанням однакових академічних облікових записів на різних освітніх рівнях − можуть виникати окремі випадки некоректного відображення даних. Над вдосконаленням логіки обробки таких ситуацій ведеться поетапна робота.

На основі результатів проведеного тестування було впроваджено низку заходів, спрямованих на підвищення ефективності, стабільності та сумісності мобільної платформи.

З метою усунення помилок, пов’язаних із некоректним відображенням академічного статусу студентів та даних про семестри, було оптимізовано

алгоритми обробки інформації. Зокрема, впроваджено додаткові перевірки для синхронізації між логікою даних та інтерфейсом користувача, що сприяло підвищенню точності відображення. Крім того, для вирішення проблеми часткового або неповного завантаження студентських записів розпочато поетапну модернізацію системи обробки, яка враховує ширший спектр академічних сценаріїв і забезпечує гнучкіший підхід до індивідуальних навчальних траєкторій.

На архітектурному рівні платформа зазнала значних змін. Первинна версія застосунку мала низький рівень модульності, що ускладнювало його підтримку та масштабування. Було здійснено рефакторинг коду із впровадженням архітектурного підходу MVVM, який забезпечив чітке розмежування логіки даних, інтерфейсу та бізнес-процесів. Це дозволило не лише спростити розробку й тестування, але й значно покращити гнучкість та підтримку кросплатформеності, що є критично важливим у середовищі Flutter. Оптимізована структура проєкту також дозволила зменшити дублювання коду та підвищити загальну продуктивність системи.

Приділено увагу продуктивності та сумісності платформи. Оптимізовано деякі запити до джерел даних і зменшено кількість зайвих операцій. Застосунок тестувався на пристроях з різними характеристиками для виявлення можливих проблем з рендерингом та швидкодією. Забезпечено базову стабільність роботи на трьох платформах (Web, Android, iOS).

3.5 Оцінка продуктивності та ефективності (з використанням інструментів Flutter, VS Code)

Для оцінки продуктивності та ефективності розробленої мобільної платформи на основі фреймворку Flutter було використано інструменти, інтегровані в середовище розробки VS Code, зокрема debugger, консоль відлагодження, Widget Tree та Flutter Developer Tools Inspector. Ці інструменти дозволяють детально проаналізувати поведінку додатка, оптимізувати його продуктивність і забезпечити коректну роботу всіх компонентів.

Проєкт було запущено в режимі відлагодження через команду F5 у VS Code, що ініціалізує виконання файлу lib\main.dart у браузері Chrome. У процесі запуску debug-консоль відображає ключову інформацію про підключення до сервісу відлагодження через WebSocket-протокол за певною адресою. Це забезпечує можливість моніторингу стану додатка в реальному часі.

Особливу увагу було приділено виведенню GoRouter, який відповідає за управління навігацією в додатку. У debug-консолі відображається структура маршрутів (рис. 3.13), що включає:

- ShellRoute як кореневий маршрут, який охоплює основні екрани застосунка:
  - / (HomeScreen) – головний екран;
  - /userProfile (з використанням ChangeNotifierProvider <userprofileviewmodel>) – екран профілю користувача; </userprofileviewmodel>
  - /academicDisciplines (з ChangeNotifierProvider <disciplinesviewmodel>) – екран перегляду академічних дисциплін; </disciplinesviewmodel>
  - /grades (з ChangeNotifierProvider <gradesviewmodel>) – екран оцінок. </gradesviewmodel>
- /login (AuthScreen) – екран авторизації, визначений як початковий маршрут.

GoRouter також вивів іменовані маршрути, такі як home → /, userProfile → /userProfile, login → /login, та підтвердив ініціалізацію початкового маршруту /login з конфігурацією MaterialApp. Для перевірки навігації відбувся перехід між екранами через інтерфейс додатка (наприклад, з /login до /userProfile), спостерігаючи за повідомленнями в дебаг-консолі. Використання панелі дебагера (кнопки Continue, Pause, Step Over) дозволило покроково проаналізувати переходи, що підтвердило коректність маршрутів і відсутність помилок у навігації.

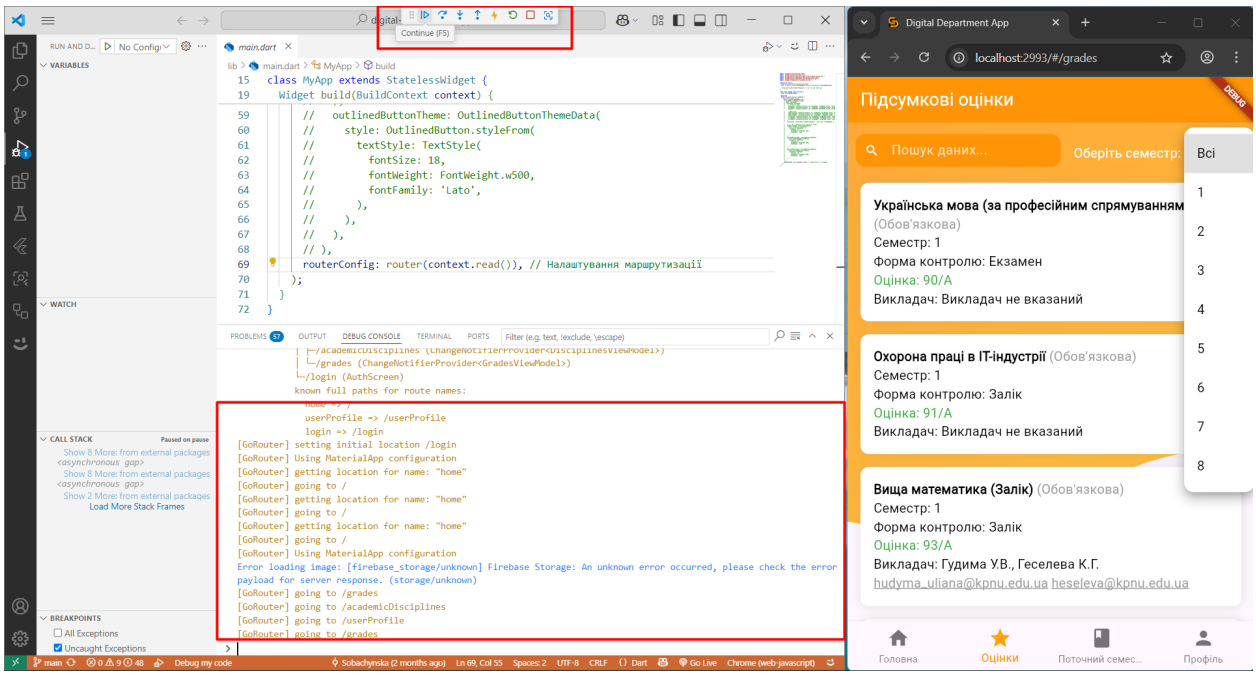


Рис. 3.13 *Виведення маршрутів GoRoute в Debug Console*

Для оцінки структури інтерфейсу використовувався Flutter Developer Tools Inspector, виконавши команду Flutter: Open DevTools через боеовеменю у VS Code. Далі перейшовши до вкладки Widget Tree, щоб візуалізувати ієрархію віджетів. Перевірялися їхні властивості: розміри, відступи, стилі, тощо (рис. 3.14). Це допомогло виявити потенційні проблеми, наприклад, переповнення (overflow) або некоректне позиціонування елементів.

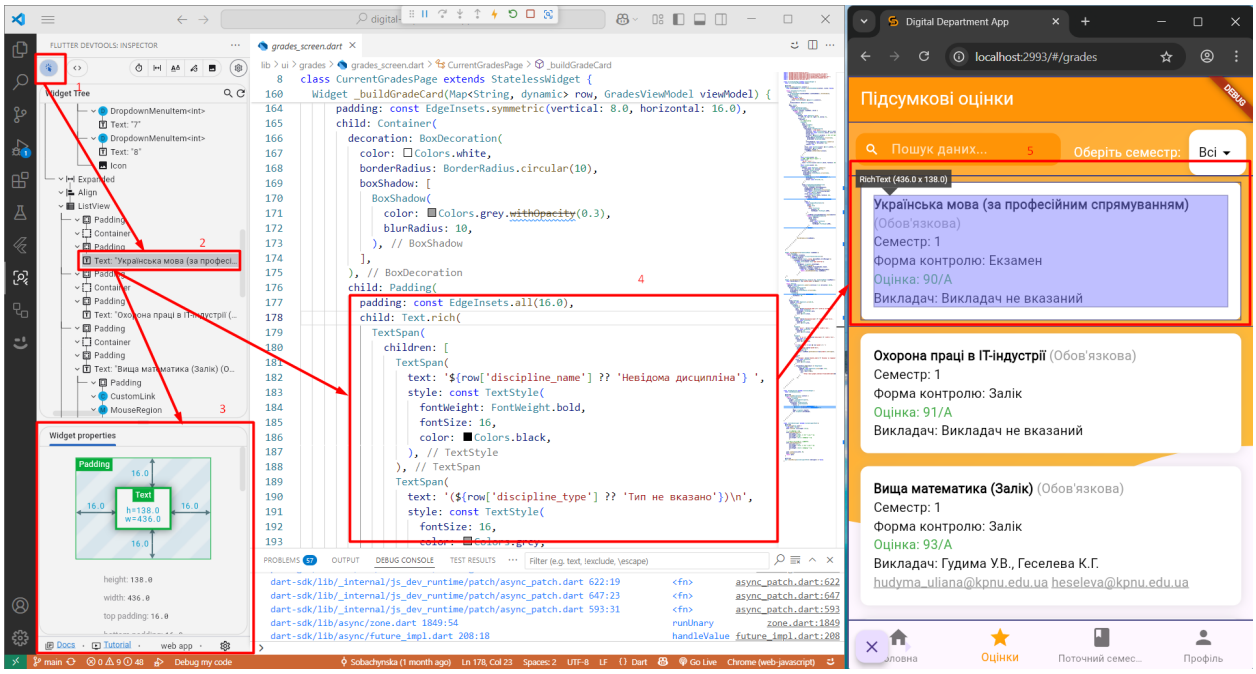


Рис. 3.14 *Показ обрання елемента з Widget Tree*

Для аналізу продуктивності активовано Performance Overlay у вкладці Performance Flutter DevTools (рис. 3.15), запустивши додаток у профільному режимі та відкривши DevTools у VS Code на емуляторі Android. Результати показали середню частоту кадрів 55 FPS, близьку до оптимальних 60 FPS, що вказує на плавне відображення, хоча періодично фіксувалися затримки (Raster Jank) із максимальним часом рендерингу 23.8 ms для кадру 2153.

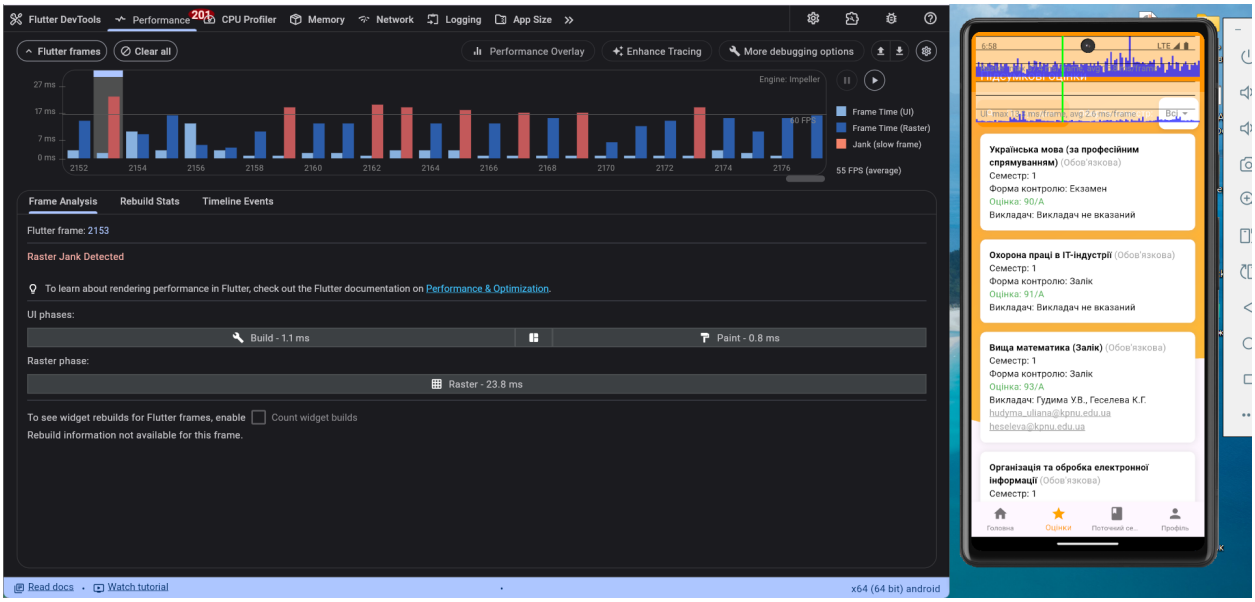


Рис. 3.15 *Аналіз продуктивності в Flutter DevTools*

Аналіз таймлайну вказав, що основна затримка пов’язана з фазою рендерингу (Raster), що може бути спричинено складним фоновим віджетом із ClipPath і градієнтом. Під час переходів між екранами (наприклад, /login до /userProfile) перевірено оновлення віджетів у Widget Tree без значних затримок. Також підтвердено коректність оновлення даних у ChangeNotifierProvider (UserProfileViewModel, GradesViewModel) через інспектор. Планується оптимізація, зокрема аналіз впливу фонового віджета та можливе спрощення геометрії для зменшення часу рендерингу.

У вкладці Memory DevTools (рис. 3.16) споживання пам’яті стабільне (13.3 MB Dart Heap), але піки до 200 MB під час Google Sign-In пов’язані з ChangeNotifierProvider (0.6 KB) і CustomLinkState (0.8 KB). Помилка PlatformException сигналізує про можливі збої з’єднання. Це не критично, але в подальшій роботі потребує перевірки витоків пам’яті.

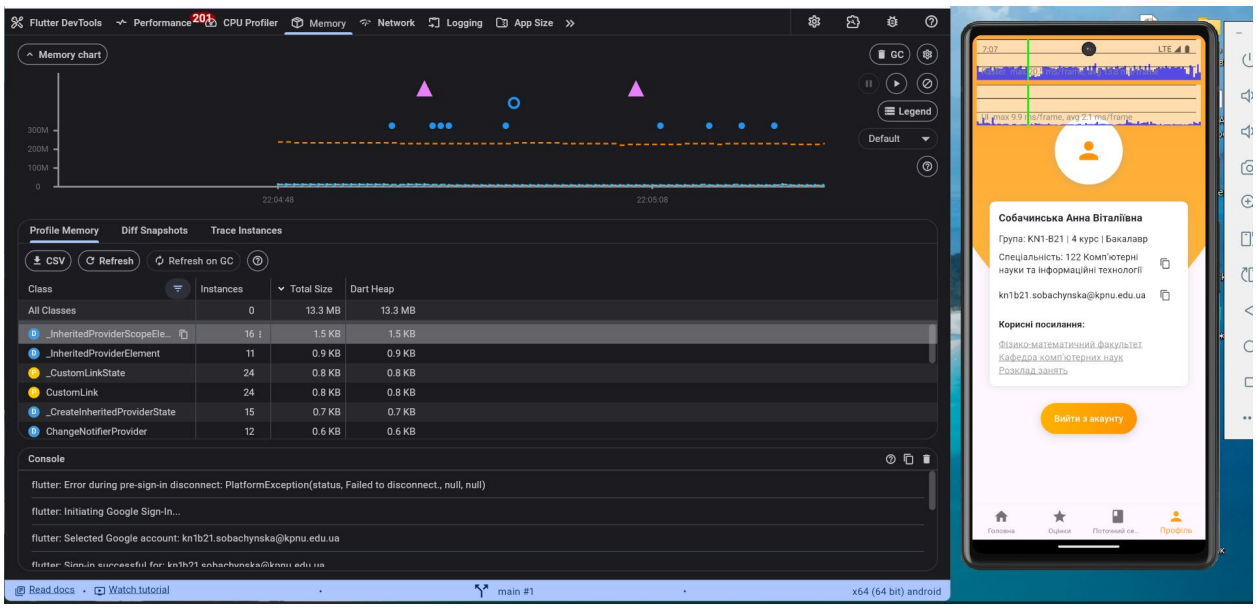


Рис. 3.16 *Аналіз споживання пам'яті в Flutter DevTools*

Висновки до розділу 3

Розроблена мобільна платформа ефективно підтримує студентів в освітньому процесі, поєднуючи сучасний дизайн, інтуїтивну навігацію та функціональність. Інтерфейс, створений за принципами Material Design 3 у Flutter, адаптивний, естетичний і зручний для Android, iOS та веб-версії. Нижня навігаційна панель із розділами «Головна», «Оцінки», «Поточний семестр» і «Профіль» забезпечує швидкий доступ до основних функцій. Карткова структура, чітка типографіка та яскраві кольори покращують сприйняття інформації.

Google Sign-In через Firebase Authentication забезпечує безпечний і зручний вхід, що спрощує доступ до системи. Серед функціональних модулів:

– «Головна» − новини й оголошення;
– «Оцінки» − детальний огляд академічної успішності та зв'язок із викладачами через Gmail;
– «Поточний семестр» − актуальні дисципліни;
– «Профіль» − персональні дані та доступ до університетських ресурсів.

Синхронізація з Firebase Cloud Firestore гарантує актуальність даних у реальному часі. Під час тестування були виявлені й усунуті помилки, зокрема

некоректне відображення статусу та проблеми доступу. Впровадження архітектури MVVM і оптимізація запитів покращили стабільність. Аналіз у Flutter DevTools засвідчив стабільну продуктивність (в середньому 55 FPS) із незначними затримками рендерингу.

ВИСНОВКИ

У межах кваліфікаційної роботи було розроблено мобільну платформу, що забезпечує здобувачам вищої освіти зручний та персоналізований доступ до індивідуального навчального плану, підсумкових оцінок із предметів та супровідної академічної інформації.

Запропоноване рішення поєднує мобільний інтерфейс, хмарну інфраструктуру та інтеграцію з Google Workspace. Для автентифікації користувачів використано вхід через обліковий запис Google, що забезпечує зручність і контрольований доступ до персоналізованої інформації. Дані про дисципліни, індивідуальний навчальний план, підсумкові оцінки, викладачів і академічну історію обробляються автоматизовано та централізовано з урахуванням актуальності й безпеки.

Розробка здійснена з використанням Flutter як інструменту для створення адаптивного застосунку з єдиною кодовою базою. Для зберігання й обробки освітніх даних застосовано хмарну платформу Firebase, що дозволяє гнучко керувати інформацією в режимі реального часу. Такий підхід забезпечує зручність для студентів і водночас не впливає на рівень адміністративного навантаження викладачів і працівників навчальних підрозділів.

Загалом, створена платформа реалізує ключові функції, пов’язані з організацією доступу до ІНП та підсумкових оцінок, й демонструє готовність до практичного впровадження в освітньому середовищі. При цьому архітектурна гнучкість рішення відкриває широкі перспективи для подальшого розвитку − зокрема розширення функціональності та адаптації до різноманітних навчальних програм і технічних середовищ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ben H. (2023). The Pros and Cons of Firestore and Realtime Database. URL: https://giraffemindset.medium.com/the-pros-and-cons-of-firestore-and-realtime-database-in-firebase-fbf812d3438 (дата звернення: 25.11.2024)
2. BrowserStack. (2023). Flutter vs React Native: A Comparison. URL: https://www.browserstack.com/guide/flutter-vs-react-native (дата звернення: 20.11.2024)
3. CampusNet. (2023). CampusNet Platform. URL: https://play.google.com/store/apps/details?id=de.datenlotsen.campusnet.amh&hl=uk (дата звернення: 15.11.2024)
4. Data Encryption in Firebase. URL: https://firebase.google.com/support/privacy#:~:text=Firebase%20services%20encrypt%20data%20in,Cloud%20Firestore (дата звернення: 10.12.2024)
5. Firebase Documentation: Hosting. URL: https://firebase.google.com/docs/hosting (дата звернення: 15.12.2024)
6. Firebase Security & Privacy Overview. URL: https://firebase.google.com/support/privacy (дата звернення: 20.12.2024)
7. Firebase Security Rules Guide. URL: https://firebase.google.com/docs/rules (дата звернення: 25.12.2024)
8. Flutter API: BottomNavigationBar. URL: https://api.flutter.dev/flutter/material/BottomNavigationBar-class.html (дата звернення: 05.01.2025)
9. Flutter Documentation: Material Design. URL: https://docs.flutter.dev/ui/design/material (дата звернення: 10.01.2025)
10. Google Apps Script Documentation. URL: https://developers.google.com/apps-script (дата звернення: 15.01.2025)
11. Google Classroom. URL: https://edu.google.com/products/classroom (дата звернення: 10.11.2024)
12. Google Cloud Console Documentation. URL: https://cloud.google.com/docs (дата звернення: 01.02.2025)

13. Google Sign-In for Flutter. URL: https://pub.dev/packages/google_sign_in (дата звернення: 25.01.2025)
14. Google Workspace for Education. URL: https://edu.google.com (дата звернення: 20.01.2025)
15. Kinsta. GitHub Usage Statistics. URL: https://kinsta.com/blog/github-statistics/ (дата звернення: 30.11.2024)
16. Moodle Community. Moodle LMS. URL: https://moodle.org (дата звернення: 12.11.2024)
17. Stack Overflow. (2024). Developer Survey 2024. URL: https://survey.stackoverflow.co/2024/ (дата звернення: 15.11.2024)
18. Собачинська А. В. Мобільна платформа для індивідуального навчального плану здобувача вищої освіти // Збірник наукових праць студентів та магістрантів Кам'янець-Подільського національного університету імені Івана Огієнка. – Кам'янець-Подільський : Кам'янець-Подільський національний університет імені Івана Огієнка, 2025. – С. 140–142.