МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МИСАК МАКСИМ ПАВЛОВИЧ

Допускається до захисту:
в. о. завідувач кафедри
інформаційних технологій, к.т.н.,
доцент

_____ Оксана ЗЕЛІНСЬКА

«___» _____ 20__ р.

МОБІЛЬНА ІНФОРМАЦІЙНО-ДОВІДНИКОВА СИСТЕМА ДЛЯ ЗДОБУВАЧІВ ФАКУЛЬТЕТУ

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (магістерська) робота
(відповідно до стандарту спеціальності та ОП)

Науковий керівник:
Бабаков Р. М., професор кафедри
інформаційних технологій,
докт. техн. наук, доцент.

_____
(підпис)

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК:_____
(підпис)

Вінниця 2024

АНОТАЦІЯ

Мисак М. П. Мобільна інформаційно-довідникова система для здобувачів факультету. Спеціальність 122 “Комп’ютерні науки”, Освітня програма “Комп’ютерні технології обробки даних”. Донецький національний університет імені Василя Стуса, Вінниця, 2024.

Кваліфікаційна робота містить 97 с. тексту, 35 рисунків, 1 таблиця, посилання на 50 літературних джерел, та 2 додатки.

У кваліфікаційній роботі досліджено існуючі рішення у галузі інформаційно-довідникових систем. Розглянуто загальний підхід до розробки та архітектуру інформаційно-довідникових систем. Після чого розроблено інформаційно-довідникову систему для здобувачів факультету.

Ключові слова: КОРИСТУВАЧ, ПРОГРАМА, ПОСЛУГА, МОБІЛЬНА ІНФОРМАЦІЙНО-ДОВІДНИКОВА СИСТЕМА, FLUTTER, DART.

ABSTRACT

Mysak M. P. Mobile information and reference system for faculty applicants. Specialty 122 "Computer Science", Educational program "Computer Data Processing Technologies". Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

The qualification work contains 97 pages of text, 35 pictures, 1 table, references to 50 literary sources, and 2 appendix.

In the qualification work the existing solutions in the field of information and reference systems are investigated. The general approach to the development and architecture of information and reference systems is considered. After that, an information and reference system for faculty applicants was developed.

Keywords: USER, PROGRAM, SERVICE, MOBILE INFORMATION AND REFERENCE SYSTEM, FLUTTER, DART

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким технологічним розвитком та постійним зростанням доступу до мобільних пристроїв. Студенти активно користуються мобільними додатками для отримання інформації та організації свого навчання. Проте, багато факультетів ще не мають зручних та функціональних мобільних додатків, які відповідали б їхнім потребам. Тому розробка мобільної інформаційно-довідникової системи для здобувачів факультету є актуальною та важливою задачею.

Окрім цього, розробка мобільного додатку для студентів факультету "Факультет інформаційних і прикладних технологій" з використанням нових технологій вносить інновації в цю галузь. Використання сучасних технологій розробки дозволяє створити кросплатформений додаток, який працюватиме на таких мобільних платформах як Android, та iOS, що забезпечить широкий охоплення користувачів.

Таким чином, розробка мобільної інформаційно-довідникової системи для здобувачів факультету відповідає сучасним вимогам студентського середовища та сприятиме покращенню якості навчального процесу. Ця робота має значний потенціал для покращення комунікації, ефективності та задоволення студентів, що робить її актуальною та цінною для дослідження та впровадження.

Предметом дослідження є аналіз предметної області та розробка мобільного додатку для студентів факультету "Факультет інформаційних і прикладних технологій", який надає інформацію про новини факультету, містить карту кабінетів та можливість прив'язки розкладу до кабінетів.

Метою дослідження є аналіз та розробка мобільної інформаційно-довідникової системи, яка задовольнятиме потреби студентів факультету у зручному та функціональному інструменті для отримання актуальної інформації та організації навчання.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– Провести аналіз існуючих рішень в схожих ;

– здійснити постановку задачі;

– вибрати інструменти для розробки системи;

– проектування архітектури мобільної інформаційно-довідникової системи

– розробка мобільної інформаційно-довідникової системи

– підсумувати результати дослідження

Наукова новизна дослідження полягатиме в розробці мобільної інформаційно-довідникової системи, що комбінує функціонал новин факультету, карту кабінетів та прив'язку розкладу до кабінетів. Також, використання Dart/Flutter для реалізації системи забезпечить кросплатформеність та зручний інтерфейс для користувачів.

Теоретична цінність полягатиме у систематизації знань про розробку мобільних додатків та рекомендаційних систем. Практична цінність виявиться у створенні функціонального мобільного додатку для студентів факультету, що сприятиме покращенню їхнього навчального процесу та доступу до необхідної інформації.

Розроблену мобільну інформаційно-довідникову систему планується апробувати шляхом впровадження її на факультеті "Факультет інформаційних і прикладних технологій" та отримання відгуків від користувачів.

Розділ 1. Аналіз предметної області

1.1 Аналіз існуючих рішень у галузі мобільних інформаційно-довідникових систем

У розділі проводиться детальний огляд і аналіз різних мобільних додатків, які використовуються як інформаційні довідники в різних галузях.

Виконуються такі кроки аналізу:

1. Огляд наявних мобільних додатків: Проводиться пошук та вивчення різних мобільних додатків, що пропонують інформаційні довідки. Звертається увага на додатки, які використовуються в різних галузях, таких як освіта, культура, туризм тощо.
2. Аналіз функціональності: Досліджується основна функціональність кожного додатку. Розглядається, які типи інформації надаються (наприклад, новини, події, карти, розклади тощо), які функції доступні для користувача (пошук, фільтри, сповіщення, налаштування тощо).
3. Дизайн та інтерфейс: Вивчається дизайн і користувацький досвід кожного додатку. Аналізується, наскільки естетичний та зручний інтерфейс додатків, як впливає на навігацію та використання.
4. Функціональні можливості: Визначаються додаткові функції та особливості кожного додатку. Це можуть бути такі функції, як синхронізація з іншими пристроями, можливість збереження вибраної інформації, можливість коментування та оцінювання тощо.
5. Рейтинг та відгуки користувачів: Аналізуються рейтинги та відгуки користувачів кожного додатку. Враховується загальна задоволеність користувачів, виявлені переваги та недоліки додатків.
6. Інноваційність: Вивчається, чи містять додатки новаторські функції або підходи, які можуть бути використані в мобільній інформаційно-довідниковій системі для здобувачів факультету.

Після проведення аналізу існуючих рішень у галузі мобільних інформаційно-довідникових систем будуть виділені їх переваги, недоліки та можливості для подальшого впровадження та адаптації до потреб факультету "Факультет інформаційних і прикладних технологій". Такий аналіз надасть науково обгрунтовану базу для розробки мобільної інформаційно-довідникової системи, яка буде оптимально відповідати потребам студентів та забезпечувати їм зручний доступ до необхідної інформації.

Давайте розглянемо кроки аналізу на прикладах чотирьох мобільних додатків, що використовуються як інформаційні довідники у різних галузях.

Приклад 1: "Електронний розклад СумДПУ" - мобільний додаток для студентів університету.

1. Аналіз функціональності: "Електронний розклад СумДПУ" надає інформацію про розклад занять, оголошення, новини університету, доступ до бібліотеки та ресурсів для навчання. Крім того, додаток має можливість сповіщення про зміни в розкладі та нагадування про події.

2. Дизайн та інтерфейс: "Електронний розклад СумДПУ" має застарілий та не зручний інтерфейс з легким доступом до основних функцій. Навігація є інтуїтивно зрозумілою, а кольорова схема відповідає корпоративним кольорам університету.

3. Функціональні можливості: Додаток дозволяє студентам зберігати особисті нотатки, відстежувати успішність та переглядати історію оцінок. Він також має функцію пошуку викладачів та інших студентів.

4. Рейтинг та відгуки користувачів: Додаток має середній рейтинг у магазині додатків та змішані відгуки студентів. Користувачі відзначають його зручність, не надійність та корисність.

5. Інноваційність: "Електронний розклад СумДПУ" має інноваційну функцію "Студентський портфоліо", де студенти можуть збирати свої досягнення та проекти для подальшого представлення потенційним роботодавцям.

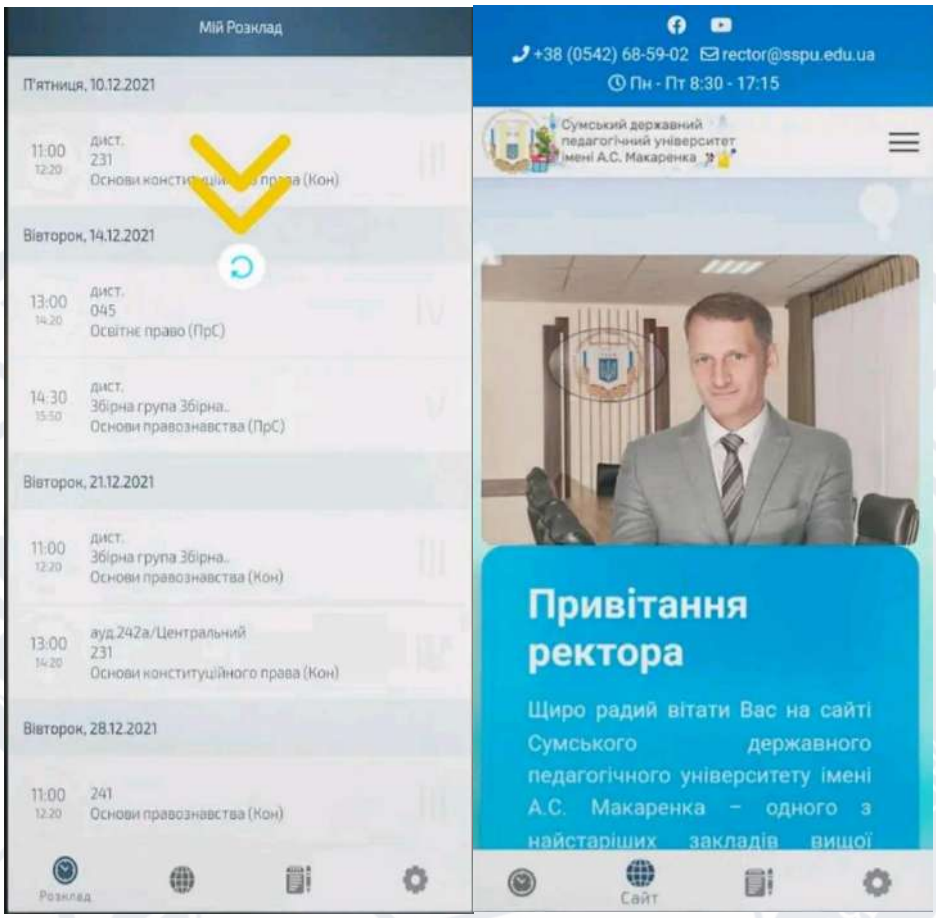


Рисунок 1.1 – Дизайн інтерфейсу "Електронний розклад СумДПУ"

Приклад 2: "Museum Dt." - мобільний додаток для відвідувачів музею.

1. Аналіз функціональності: "Museum Dt." надає докладну інформацію про експонати, історію музею, години роботи та ціни на квитки. Крім того, додаток може пропонувати аудіогід та інтерактивні екскурсії.

2. Дизайн та інтерфейс: "Museum Dt." має привабливий та естетичний дизайн, який відображає характер музею. Інтерфейс є зручним та легким у використанні.

3. Функціональні можливості: Додаток може надавати інтерактивні картки експонатів, дозволяти відправляти цікаві факти про експозиції на соціальні мережі та надавати можливість розширеної реальності для покращеного досвіду відвідування.

4. Рейтинг та відгуки користувачів: Додаток має позитивні відгуки від відвідувачів музею, які високо оцінюють зручність та цінність інформації, яку надає "Museum Dt.".

5. Інноваційність: Додаток може використовувати технології доповненої реальності для створення інтерактивних ефектів та розширення досвіду відвідувачів.

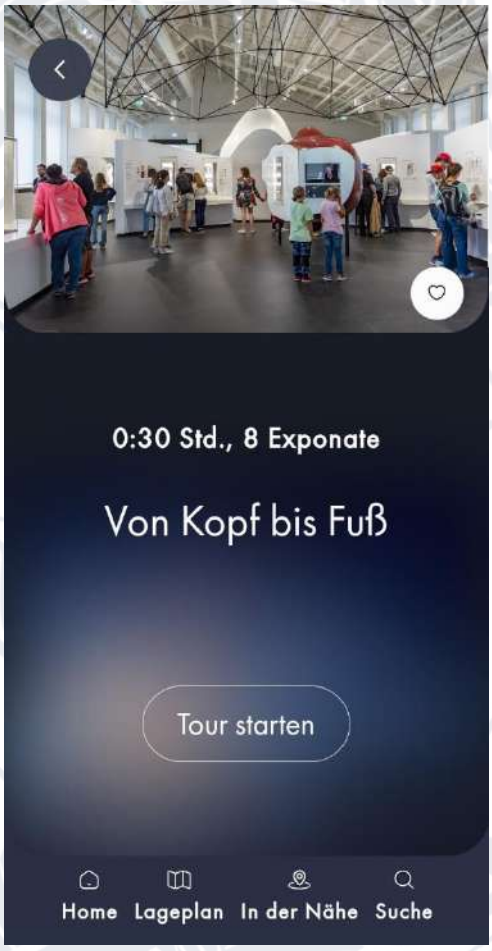


Рисунок 1.2 – Дизайн інтерфейсу "Museum Dt."

Приклад 3: "izi.TRAVEL" – мобільний додаток в сфері туризму.

1. Огляд функціональності: Дослідження можливостей додатка, він має функціонал пошуку та бронювання готелів, вказівки щодо місць відпочинку та визначних пам'яток, інформацію про транспортні засоби та маршрути, мовні переклади та перекладач, оцінки та відгуки користувачів.

2. Оцінка дизайну та інтерфейсу: Аналіз зовнішнього вигляду додатка, зручності навігації, використання графічних елементів та фотографій, відповідність дизайну атмосфері подорожей та туризму.

3. Технологічний аналіз: Дослідження використаних технологій розробки, підтримку різних платформ (Android, iOS), інтеграцію з зовнішніми сервісами (картографічні сервіси, системи онлайн-бронювання), продуктивність та швидкодію додатка.

4. Рейтинг та відгуки користувачів: Аналіз оцінок та коментарів користувачів, що використовували альтернативний додаток. Оцінка задоволеності користувачів, виявлення переваг та недоліків, які вони вказують у своїх відгуках.

5. Інноваційність: Оцінка рівня інноваційності альтернативного додатка в порівнянні з іншими рішеннями на ринку. Аналіз новаторських функцій, унікальних підходів до вирішення проблем користувачів та технологічних рішень.

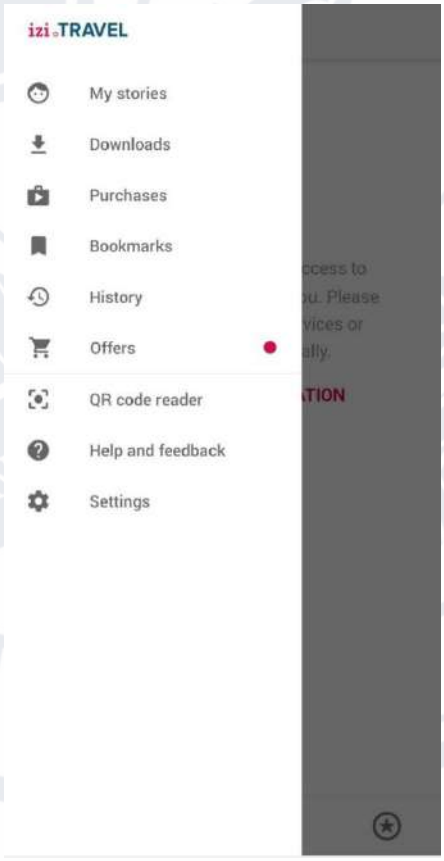


Рисунок 1.3 – Дизайн інтерфейсу "izi.TRAVEL"

Приклад 4: "Samsung Health" – мобільний додаток в сфері здоров’я

1. Аналіз функціональності: Оцінка доступних функцій і можливостей альтернативного додатка, таких як трекінг активності, калорій, супровід тренувань, дієтологічні поради, сповіщення та нагадування, спільноти та соціальні функції. Порівняння з іншими додатками в сфері фітнесу та здоров'я.

2. Дизайн та інтерфейс: Аналіз візуального оформлення та зручності користування альтернативного додатка. Оцінка естетичних аспектів дизайну, інтуїтивність і зрозумілість інтерфейсу для користувачів.

3. Функціональні можливості: Дослідження наявних функцій, які дозволяють користувачам відстежувати свої досягнення, ставити цілі, отримувати поради та підказки для поліпшення здоров'я. Аналіз інноваційних рішень, таких як використання датчиків, штучного інтелекту та інших технологій.

4. Рейтинг та відгуки користувачів: Аналіз рейтингу та відгуків користувачів, які використовували альтернативний додаток. Оцінка задоволеності користувачів, виявлення позитивних та негативних аспектів, які вони відзначають у своїх відгуках.

5. Інноваційність: Інтеграція з фітнес-трекерами, смарт-годинниками та іншими пристроями для отримання точних даних щодо активності та здоров'я користувача. Використання алгоритмів машинного навчання для персоналізації рекомендацій, аналізу результатів та прогнозування подальшого розвитку користувача.

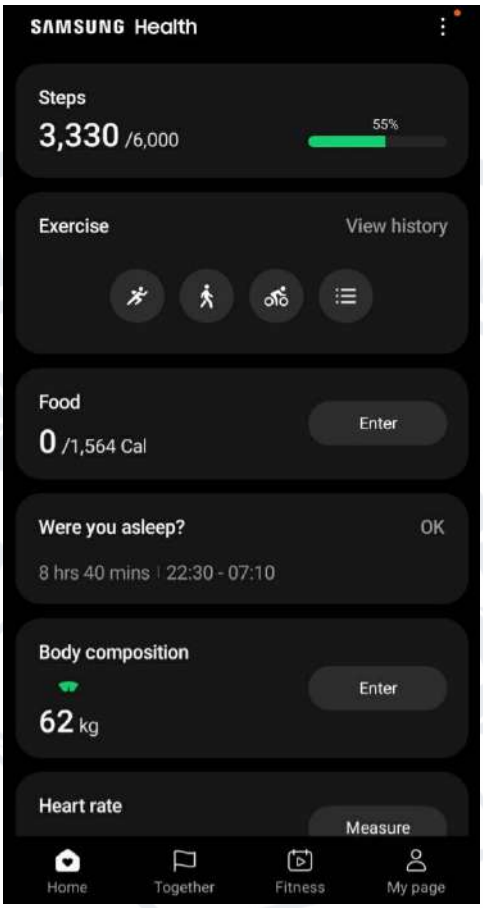


Рисунок 1.4 – Дизайн інтерфейсу

Таблиця 1.1 - Порівняння існуючих рішень в різних сферах

Критерії	*Електронний розклад СумДПУ*	*Museum Dt.*	*izi.TRAVEL*	*Samsung Health*
Виконання поставлених задач	+	+	+	+
Можливість ітеграції	-	-	-	+
Наявність компонентів збору даних	+	-	+	+
Наявність модуля аналізу	-	-	+	+

Аналіз існуючих рішень у галузі мобільних інформаційно-довідникових систем дозволяє отримати інформацію про успішні функціональності, дизайн, користувацький досвід та інноваційні рішення, які можуть бути використані при розробці мобільної інформаційно-довідникової системи для здобувачів факультету "Факультет інформаційних і прикладних технологій". Такий аналіз допомагає виокремити найкращі практики та принципи, які можна впровадити для створення ефективного та корисного додатку для студентів.

1.2 Загальний підхід до розробки мобільного додатку інформаційно-довідникових систем

Ключові етапи процесу розробки програми [25]:

Рисунок 1.5 - Процес розробки мобільних додатків

1. Затвердження ідеї:

- Виявлення потреби або проблеми, яку мобільний додаток буде вирішувати.

- Вивчення можливостей та обмежень затвердження ідеї, включаючи аналіз ресурсів, часу та бюджету.

2. Дослідження ринку:

- Аналіз схожих додатків, конкурентів та їх функціональності.

- Вивчення цільової аудиторії та її потреб у мобільному додатку.

- Виявлення можливостей для інновацій та відмінностей від конкурентів.

3. Створення прототипа:

- Розробка інтерактивної моделі додатку, що демонструє його основні функції та інтерфейс.

- Збір відгуків і вражень від потенційних користувачів для вдосконалення прототипу.

4. Вибір технологій розробки:

- Вибір мобільної платформи (iOS, Android, Windows Phone тощо) для розробки додатку.

- Вибір мови програмування, фреймворків та інструментів розробки відповідно до потреб і можливостей додатку.

5. Розробка:

- Реалізація функціональності додатку, включаючи взаємодію з базою даних, обробку даних, навігацію тощо.

- Розробка користувацького інтерфейсу, враховуючи дизайн, ергономіку та зручність використання.

6. Тестування функціоналу:

- Проведення тестів для перевірки правильності роботи функціональних модулів та їх взаємодії.

- Виявлення та усунення помилок, оптимізація роботи додатку.

7. Тестування інтерфейсу:

- Оцінка зручності використання інтерфейсу, включаючи навігацію, взаємодію з елементами, відповідність дизайну стандартам та очікуванням користувачів.

- Збір відгуків користувачів та внесення виправлень для поліпшення інтерфейсу.

8. Запуск бета-версії додатку:

- Випуск обмеженої версії додатку для тестування серед обраної групи користувачів.

- Збір відгуків та даних про використання для остаточного вдосконалення перед публічним запуском.

9. Допрацювання і запуск додатку:

- Аналіз отриманих відгуків та внесення необхідних змін у додаток.

- Підготовка до публічного запуску, включаючи маркетингову кампанію, підтримку користувачів та розповсюдження додатку через відповідні канали.

Загальний підхід до розробки мобільного додатку інформаційно-довідникових систем включає всі етапи, починаючи з аналізу потреб користувачів і закінчуючи підтримкою та впровадженням додатку. Кожен етап вимагає ретельного планування, використання відповідних технологій та інструментів, а також забезпечення високої якості і зручності використання для користувачів.

1.3 Загальний огляд типових рішень при розробці мобільних додатків

Нативний додаток: Розробка окремих додатків для кожної мобільної платформи (iOS, Android, Windows Phone).Дозволяє максимально використовувати можливості платформи та надає найвищу продуктивність. Вимагає додаткових зусиль і ресурсів для розробки та підтримки окремих версій додатка для кожної платформи [30, 31, 36-38,].

Хмарні додатки: Розробка додатка, який використовує хмарні сервіси для зберігання даних та виконання обчислень. Дозволяє забезпечити синхронізацію даних між різними пристроями та забезпечити доступ до актуальних даних з

будь-якого місця. Вимагає налагодження та підтримки хмарного сервісу, а також наявності постійного інтернет-з'єднання.

Гібридні додатки: Розробка додатка, який комбінує веб-технології (HTML, CSS, JavaScript) з можливостями мобільних платформ. Дозволяє створити один додаток, який працює на різних платформах, зменшуючи зусилля для розробки та підтримки. Може втратити деяку продуктивність і функціональність, порівняно з нативними додатками.

Progressive Web Apps (PWA): Веб-додатки, які виглядають та працюють як мобільні додатки, але використовуються через веб-браузер. Дозволяє забезпечити швидкий доступ до додатка без необхідності установки з магазинів додатків. Обмеженіше використання функціональності пристрою та можливостей платформи.

Кросплатформені фреймворки: Використання фреймворків, які дозволяють розробляти додатки для різних платформ за допомогою спільного коду. Дозволяє забезпечити одночасну розробку додатків для кількох платформ, зменшуючи витрати часу та ресурсів. Може мати обмеження в продуктивності та доступному функціоналу платформи.

Low-Code/No-Code платформи: Використання платформ, що дозволяють розробляти додатки без необхідності програмування на високому рівні. Дозволяє швидко створювати прототипи та базові додатки з використанням готових компонентів та інструментів. Може бути обмежена в можливостях настроювання та розширення функціоналу. Кожен з цих підходів має свої переваги та обмеження, і вибір залежить від конкретних вимог, ресурсів та цілей проекту.

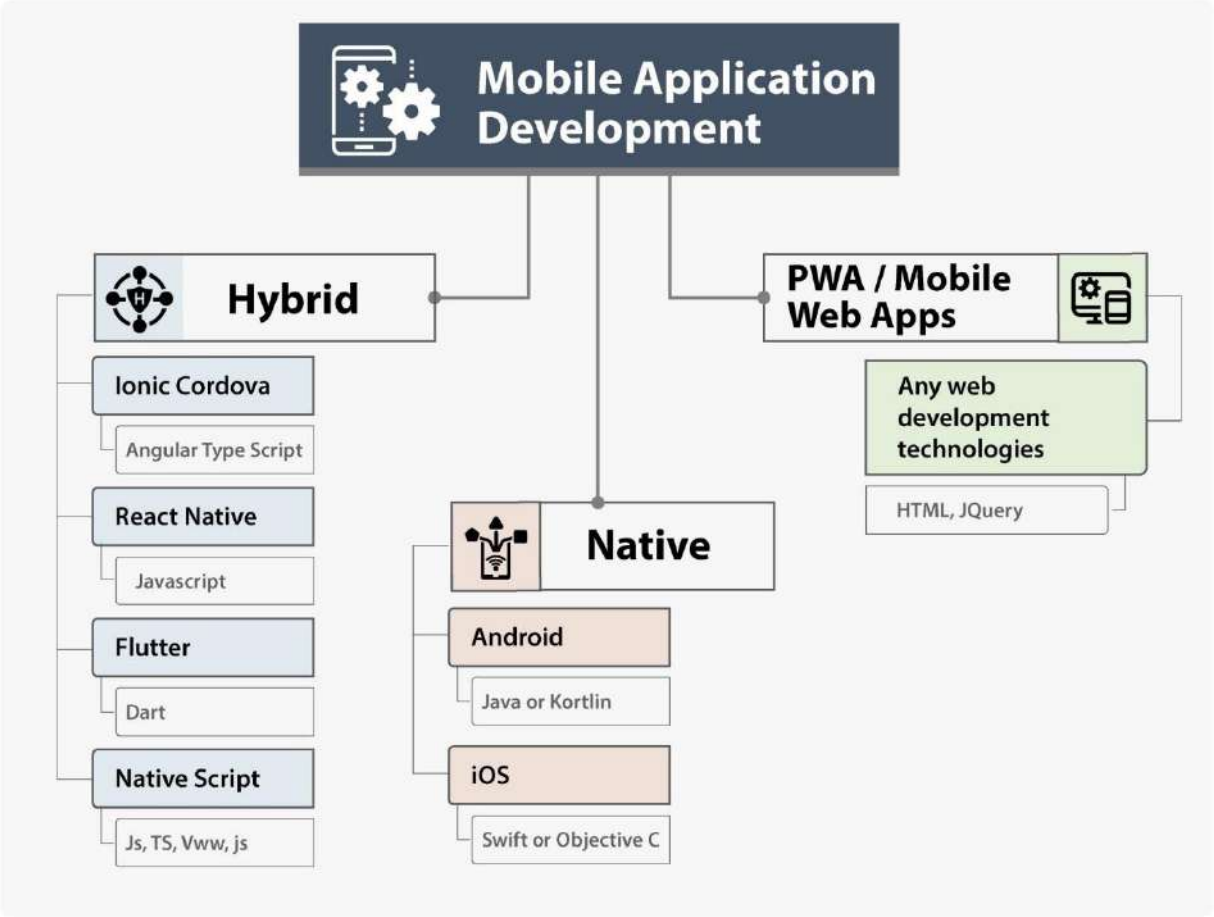


Рисунок 1.6 – Три підходи до розробки мобільних додатків

При розробці мобільного додатку інформаційно-довідникових систем, існує кілька важливих факторів, які варто враховувати. Нижче наведено головні з них:

Цільова аудиторія: Розуміння потреб та вимог цільової аудиторії є ключовим фактором. Варто з'ясувати, які функції та можливості будуть найкориснішими та цікавими для користувачів.

Функціональність: Детально визначте функції та можливості, які має мати додаток. Це можуть бути пошукові функції, категорії, фільтри, сповіщення, можливість зберігання улюблених елементів та багато іншого [31, 32].

Дизайн та інтерфейс: Створення зручного та привабливого дизайну є важливим елементом. Інтерфейс додатку повинен бути простим, інтуїтивно зрозумілим та забезпечувати зручну навігацію користувача.

Кросплатформеність: Розгляньте можливість розробки додатку, який працюватиме на різних мобільних платформах, таких як iOS та Android. Це дозволить досягти більшої аудиторії користувачів та зменшити витрати на розробку та підтримку.

Безпека: Забезпечення захисту даних користувачів є важливим фактором. Використання шифрування, захищених протоколів передачі даних та інших заходів безпеки є необхідними для запобігання несанкціонованому доступу до інформації.

Тестування: Важливо провести ретельне тестування додатку, щоб переконатися, що він працює стабільно, без помилок та забезпечує зручний досвід користувача. Тестування може включати функціональне тестування, тестування відмово стійкості, тестування швидкості та інші види тестів.

Підтримка та оновлення: Розробка мобільного додатку - це довгостроковий процес. Варто планувати підтримку та оновлення додатку, включаючи виправлення помилок, вдосконалення функцій та додавання нового контенту.

Ці фактори допоможуть створити ефективний та користувачам зручний мобільний додаток інформаційно-довідникових систем.

1.4 Аналіз потреб та вимог здобувачів факультету

Для розробки ефективного мобільного додатку інформаційно-довідникової системи для здобувачів факультету, важливо провести опитування та дослідження, щоб визначити конкретні потреби та вимоги студентів та викладачів. Це можна зробити шляхом створення анкет або проведення фокус-груп, де учасники зможуть висловити свої думки та пропозиції щодо функціональності та можливостей додатку.

Під час аналізу потреб здобувачів факультету, важливо визначити, яку інформацію студенти та викладачі найчастіше шукають, які процеси організації навчання є важливими для них та якими засобами вони користуються для отримання цієї інформації. Це можуть бути розклади занять, матеріали лекцій,

відомості про викладачів, інформація про академічні досягнення та багато іншого.

На основі отриманих відгуків, опитувань та аналізу потреб, важливо виявити пріоритетні функції та можливості, які мають бути реалізовані у мобільному додатку. Це можуть бути функції, які дозволять студентам переглядати розклад занять, завдання та матеріали лекцій, здійснювати комунікацію з викладачами, отримувати сповіщення про оновлення та багато іншого. Важливо вибрати ті функції, які найбільше відповідають потребам користувачів та сприятимуть поліпшенню їх навчального процесу.

Під час аналізу потреб та вимог здобувачів факультету також варто враховувати технічні можливості та обмеження мобільних платформ. Наприклад, якщо більшість студентів використовує пристрої з операційною системою Android, то розробка додатку для цієї платформи може бути пріоритетною. Також варто враховувати особливості використання мобільних пристроїв, наприклад, ергономічні особливості дотикового інтерфейсу, розмір екрану та його роздільну здатність.

Крім того, під час аналізу потреб і вимог важливо звернути увагу на рейтинги та відгуки користувачів існуючих додатків, що використовуються в освітніх установах. Це може допомогти зрозуміти, які функції та можливості є популярними та бажаними серед користувачів, а також виявити потенційні недоліки та проблеми, які можна уникнути у власному додатку.

У результаті аналізу потреб та вимог здобувачів факультету будуть отримані цінні висновки та рекомендації, які стануть основою для подальшого проектування та розробки мобільного додатку інформаційно-довідникової системи.

1.5 Тенденції розвитку мобільних інформаційно-довідникових систем

В останні роки мобільні інформаційно-довідникові системи пройшли значний шлях розвитку та трансформації. Деякі з найважливіших тенденцій у цій галузі включають:

Розширення функціональності: Сучасні мобільні інформаційно-довідникові системи стають все більш функціональними і потужними. Вони надають користувачам широкі можливості, такі як доступ до розкладу занять, новин факультету, карти кабінетів, сповіщення про події тощо. Також в них можуть бути вбудовані інструменти для спілкування, обміну матеріалами та колективної роботи.

Персоналізація та адаптація: Мобільні додатки стають все більш адаптивними до потреб і вимог користувачів. Вони надають можливість налаштовувати вигляд, функціональність та контент додатку відповідно до індивідуальних вподобань студента. Також вони можуть здійснювати персоналізовану рекомендацію контенту та послуг, що полегшує навігацію та забезпечує зручність використання.

Інтеграція з іншими системами: Мобільні інформаційно-довідникові системи все частіше взаємодіють з іншими системами та сервісами, що дозволяє студентам отримувати доступ до різноманітної інформації та функціоналу. Наприклад, інтеграція з системами управління навчальним процесом, бібліотеками, системами електронного документообігу тощо.

Використання інноваційних технологій: Сучасні мобільні додатки використовують новітні технології, такі як штучний інтелект, машинне навчання, розпізнавання образів та голосу, віртуальна та доповнена реальність, що розширюють їх можливості та надають нові враження користувачам.

Забезпечення безпеки та конфіденційності: З огляду на зростаючу кількість особистої і конфіденційної інформації, яку користувачі передають через

мобільні додатки, забезпечення безпеки та конфіденційності стає надзвичайно важливим аспектом розробки. Мобільні додатки повинні мати механізми шифрування, аутентифікації та захисту даних.

Мультимедійність та взаємодія: Сучасні мобільні додатки активно використовують мультимедійні елементи, такі як фотографії, відео, анімацію, що допомагає візуалізувати інформацію та забезпечує більш ефективну комунікацію. Також, взаємодія з користувачем стає більш інтуїтивною та зручною завдяки використанню жестів, віджетів та інших інтерактивних елементів.

Використання хмарних технологій: Зростаюча популярність хмарних технологій дозволяє зберігати дані та ресурси на віддалених серверах і забезпечує швидкий та безперебійний доступ до них через мобільний додаток. Це спрощує процес розробки та управління додатком, а також забезпечує гнучкість та масштабованість.

Використання аналітики та штучного інтелекту: Застосування аналітики даних та штучного інтелекту дозволяє мобільним додаткам надавати корисні рекомендації, аналізувати поведінку користувачів та покращувати функціональні можливості. Наприклад, додаток може рекомендувати студентам найкращі матеріали для вивчення, оптимізувати розклад занять або надавати індивідуальні поради.

Інтеграція з глобальними платформами та сервісами: Мобільні додатки все частіше взаємодіють з глобальними платформами, такими як соціальні мережі, платіжні системи, картографічні сервіси тощо. Це розширює можливості додатку, дозволяючи користувачам обмінюватися інформацією, здійснювати покупки, знаходити розташування тощо.

Підтримка різних платформ: З урахуванням розповсюдженості різних мобільних платформ, таких як iOS та Android, важливо розробляти додатки, які підтримують обидві платформи або мають відповідні версії для кожної з них. Це

дозволяє максимально охопити аудиторію користувачів та забезпечити їм зручність використання додатку.

Ці фактори визначають важливі аспекти при розробці мобільних додатків. Розробники повинні бути свідомими цих тенденцій та враховувати їх при виборі технологій, функціоналу, безпеки та інших аспектів розробки, щоб створити високоякісний та конкурентоздатний мобільний додаток.

1.6 Постановка задачі

Огляд сучасних підходів та засобів до проектування та розроблення програмного забезпечення дозволив обрати для створення власної системи ефективні технології та інструментальні засоби: IDE – Visual Studio 2022; Мова програмування – Dart; Фреймворк Flutter.

На основі виконаного аналізу предметної області можна сформулювати постановку задачі.

Розробити інформаційно-довідникову систему. Забезпечити внесення даних про розклад, можливість додавання новин адміністратору, закріплення важливої інформації та підкріплення розкладу до карти кабінетів.

Створити навігаційне меню для можливості швидкого та зручного отримання доступу до потрібної функції в системі.

Розробити супровідну документацію до створеної системи.

Дипломна робота припускає розробку додатка засобами об’єктно орієнтованого середовища програмування .

Реалізація додатка виконується з використанням фреймворком Flutter та мови програмування Dart. Результат – мобільний додаток (Android, IOS).

Вона повинна містити в собі:

Класи додатку:

– учень ;

– староста;

– куратор;

Введення:

Вводиться і редагується: інформація про новини університету, розклад.

Фіксація:

– фіксуються важливу інформацію.

Результат роботи:

– Надання послуг студенту.

Серед усіх функцій, які виконує система можна виділити загальні операції:

– реакція програми на вибір меню користувача;

– реакція програми на натискання кнопок в програмі;

– реакція програми на введення неправильних даних.

Функціональні вимоги:

– можливість додавати, редагувати, видаляти дані про новини, розклад та нагадування;

– виведення на екран інформацію, закріплення нагадувань.

Нефункціональні вимоги:

– для роботи програми на комп’ютері повинна бути встановлена бібліотека класів .NET Framework 4.7;

– для забезпечення роботи програми потрібно мати лише мобільний телефон на Android чи IOS.

Висновок до розділу 1

У ході виконання розділу 1 було ретельно розглянуто та вирішено багато ключових завдань, спрямованих на розробку мобільної інформаційно-довідкової системи для факультету "Факультет інформаційних і прикладних технологій." Процес створення додатку був докладно структурований та включав в себе наступні етапи.

Проведено аналіз існуючих мобільних додатків та інформаційно-довідкових систем для з'ясування їхніх переваг та недоліків. Визначено загальні принципи та підходи до розробки мобільних інформаційно-довідникових систем. Розглянуто типові рішення, які можуть бути використані при розробці мобільних додатків. Визначено потреби та вимоги користувачів факультету щодо мобільної інформаційно-довідкової системи. Розглянуто актуальні тенденції у розвитку мобільних інформаційно-довідникових систем, що допомогли визначити напрямки подальшої розробки. Сформульовано завдання на розробку мобільної інформаційно-довідкової системи для факультету "Факультет інформаційних і прикладних технологій," враховуючи аналіз і вимоги користувачів та сучасні тенденції.

Розділ 2. Теоретична частина розробки

2.1 Вибір технологій та інструментів

Хоча Kotlin є офіційною мовою для Android, існує багато інших мов, які можна використовувати для розробки програм Android. Подробиці про них наведено нижче, щоб допомогти вам прийняти зважене рішення [9].

1. Java

Спочатку Java була офіційною мовою для розробки додатків для Android (але тепер її замінив Kotlin), і, отже, вона також є найбільш використовуваною мовою. Багато програм у магазині Play створено на Java, і це також найбільш підтримувана мова Google. На додаток до всього цього, Java має чудову онлайн-спільноту для підтримки у разі будь-яких проблем (і повірте мені, проблеми будуть!).

Однак Java є складною мовою для початківців, оскільки вона містить такі складні теми, як конструктори, винятки нульового покажчика, паралелізм, перевірені винятки тощо. Крім того, Android Software Development Kit (SDK) підвищує складність на новий рівень!

Загалом, Java — чудова мова, щоб відчути всі переваги розробки додатків для Android. Однак це може бути трохи складніше для початківців, які воліють почати з чогось легшого, а потім повернутися до цього.

2. Kotlin

Зараз Kotlin є офіційною мовою для розробки додатків для Android, оголошеною Google у 2019 році. Kotlin — це кросплатформна мова програмування, яка може використовуватися як альтернатива Java для розробки додатків для Android. У 2017 році він також був представлений як додаткова «офіційна» мова Java. Kotlin може взаємодіяти з Java і працює на віртуальній машині Java.

Єдина значна відмінність полягає в тому, що Kotlin видаляє зайві функції Java, такі як винятки нульового покажчика. Це також усуває необхідність закінчувати кожен рядок крапкою з комою. Коротше кажучи, початківцям набагато простіше випробувати Kotlin порівняно з Java, і його також можна використовувати як «точку входу» для розробки додатків для Android.

3. C++

C++ можна використовувати для розробки додатків Android за допомогою Android Native Development Kit (NDK). Однак програму неможливо створити повністю за допомогою C++, і NDK використовується для реалізації частин програми у рідному коді C++. Це допомагає використовувати бібліотеки коду C++ для програми за потреби.

Хоча в деяких випадках C++ корисний для розробки додатків для Android, його набагато складніше налаштувати та він набагато менш гнучкий. Це також може призвести до збільшення кількості помилок через підвищену складність. Отже, краще використовувати Java порівняно з C++, оскільки вона не забезпечує достатнього приросту, щоб компенсувати необхідні зусилля.

4. C#

C# дуже схожий на Java, тому він ідеально підходить для розробки програм для Android. Як і Java, C# також реалізує збирання сміття, тому є менше шансів витоку пам’яті. Крім того, C# має чистіший і простіший синтаксис, ніж Java, що робить кодування з його допомогою порівняно легшим.

Раніше найбільшим недоліком C# було те, що він міг працювати лише в системах Windows, оскільки використовував .NET Framework. Однак цю проблему вирішив Xamarin. Android (раніше Mono для Android) — це кросплатформна реалізація спільної мовної інфраструктури. Тепер Xamarin. Інструменти Android можна використовувати для написання рідних програм для Android і спільного використання коду на кількох платформах.

5. Python

Python можна використовувати для розробки додатків Android, навіть якщо Android не підтримує власну розробку Python . Це можна зробити за допомогою різних інструментів, які перетворюють програми Python на пакети Android, які можна запускати на пристроях Android.

Прикладом цього є Kivy, бібліотека Python з відкритим кодом, яка використовується для розробки мобільних програм. Він підтримує Android, а також заохочує швидку розробку програм (що, на мій погляд, є безпрограшною ситуацією!). Однак недоліком цього є те, що Kivy не матиме нативних переваг, оскільки він не підтримується нативно.

6. HTML, CSS, JavaScript

Програми для Android можна створювати за допомогою HTML , CSS і JavaScript за допомогою фреймворку Adobe PhoneGap, який підтримує Apache Cordova. Фреймворк PhoneGap в основному дозволяє використовувати навички веб-розробки для створення гібридних програм, які відображаються через «WebView», але упаковані як програма.

Хоча фреймворку Adobe PhoneGap достатньо для базових завдань у сфері розробки програм для Android, він навряд чи вимагає багато програмування, окрім JavaScript. І оскільки для створення пристойної програми потрібно багато працювати, краще використовувати інші мови в цьому списку, якщо ви хочете, щоб вас називали справжнім розробником Android (Так…Це річ!). Але якщо вам зручно працювати з Javascript, ви можете вивчити React Native, який є фреймворком з відкритим вихідним кодом, який зараз дуже затребуваний. Ви можете розробляти гарні та потужні гібридні програми за допомогою нативної системи React, тобто ваша програма буде працювати як на Android, так і на iOS. Розробка гібридних додатків стає такою популярною, тому навчання реагування може допомогти вам стати кар’єрою в розробці програмного забезпечення.

7. Dart

Ігнорування Dart як мова програмування в сучасному контексті було б схоже на ігнорування горили в кімнаті (тому що слон — це java). Dart — це мова програмування з відкритим вихідним кодом, яка є основою фреймворку Flutter, який сьогодні набуває великої популярності завдяки своїй здатності створювати гарні та продуктивні програми для Інтернету, комп'ютера та мобільних пристроїв за менший час. Ключова перевага Dart полягає в тому, що він розроблений Google як клієнтська оптимізована мова для швидких програм на будь-якій платформі. Dart головним чином зосереджується на полегшенні розробки інтерфейсу користувача для розробників за допомогою таких функцій, як гаряче перезавантаження, яке дозволяє розробникам миттєво бачити зміни під час роботи над програмою. Dart також відомий своєю високою продуктивністю, він компілюється в машинний код ARM і x64 для мобільних пристроїв, комп'ютерів і серверної частини. І до JavaScript для веб-програм. [7, 8, 10, 11]

8. Corona

Corona — це набір для розробки програмного забезпечення, який можна використовувати для розробки програм Android за допомогою Lua. Він має два режими роботи, а саме Corona Simulator і Corona Native. Corona Simulator використовується для безпосереднього створення програм, тоді як Corona Native використовується для інтеграції коду Lua з проектом Android Studio для створення програми з використанням нативних функцій.

Хоча Lua трохи обмежена порівняно з Java, вона також набагато простіша та має легшу криву навчання. Крім того, є функції монетизації збірки, а також різні ресурси та плагіни, які збагачують досвід розробки додатків. Corona в основному використовується для створення графічних програм та ігор, але не обмежується цим.

Однією з оптимальних технологій, яку ми обираємо для розробки нашої магістерської роботи, є Dart/Flutter [1, 6, 21]. Dart підтримує об'єктно-орієнтоване та функціональне програмування, а також має вбудовану систему збирання сміття, що сприяє ефективному використанню ресурсів. Flutter, у свою чергу, є

фреймворком розробки користувацького інтерфейсу. Він базується на мові Dart і надає можливості для швидкої та ефективної розробки кросплатформенних мобільних додатків для платформ Android та iOS. Однією з особливостей Flutter є "гаряче перезавантаження" (hot reload), що дозволяє розробникам швидко бачити зміни, внесені в код, без необхідності повного перекомпілювання додатку.

Оптимальність вибору Dart/Flutter для нашої магістерської роботи обумовлена кількома факторами [2-5, 21]. По-перше, ця технологія дозволяє розробляти кросплатформенні додатки, що означає, що ми зможемо випускати один додаток для обох основних платформ - Android та iOS. Це зменшує затрати на розробку та підтримку додатку.

По-друге, Dart має чистий синтаксис і простий у вивченні, що дозволить швидко оволодіти мовою програмування для всіх учасників команди розробки. Крім того, Flutter надає велику кількість готових компонентів і віджетів, які допоможуть ефективно будувати інтерфейс користувача і забезпечити його красивий вигляд та високу продуктивність [47].

Також варто зазначити, що Dart/Flutter має активну спільноту розробників, що забезпечує підтримку, постійне оновлення та вдосконалення фреймворку [12, 13]. Це означає, що ми отримаємо актуальні інструменти та рішення для розробки нашого мобільного додатку.

Узагальнюючи, вибір технології Dart/Flutter для нашої магістерської роботи є раціональним рішенням, оскільки він дозволяє розробляти кросплатформенні додатки, має простий синтаксис, надає широкі можливості для розробки інтерфейсу користувача та має активну спільноту розробників [49]. Це допоможе нам створити ефективний, функціональний та привабливий мобільний додаток для нашої магістерської роботи.

Ринок баз даних є давно насиченим ринком і все ще переживає двозначне зростання . Більша частина цього зростання пов’язана з базами даних NoSQL і новітніми технологіями баз даних, такими як бази даних часових рядів або бази

даних графіків. У міру того як обчислення переходять у бік децентралізованих обчислень на межі , у центрі уваги стають локальні бази даних , які підтримують децентралізовані потоки даних на мобільних пристроях, IoT та інших вбудованих пристроях. Деякі з них походять зі світу збереження даних Flutter, і ми розглянемо їх за секунду.

В рамках нашої магістерської роботи ми розглядаємо декілька варіантів баз даних для використання у мобільному додатку. Ось короткий опис кожної з них:

Firebase Realtime Database ця база даних надає реально-часну синхронізацію даних між різними платформами та пристроями. Завдяки своїй властивості автоматичної синхронізації змін, вона є ідеальним варіантом для додатків, що потребують змін у реальному часі, наприклад, чати або спільні списки [14-20, 22].

Hive - це легка та швидка база даних, яка забезпечує швидкий доступ до даних, зберігаючи їх локально на пристрої. Це робить Hive привабливим варіантом для додатків, які працюють зі збереженими локально даними та потребують швидкої реакції.

Moor - це бібліотека баз даних, яка використовує аннотації та SQL-подібну мову запитів. Вона забезпечує зручну роботу з базою даних, дозволяючи розробникам визначати схему бази даних та виконувати запити. Moor допомагає управляти складними відносинами між даними, що може бути корисним у складних мобільних додатках.

ObjectBox - це швидка та ефективна база даних, спеціально створена для мобільних пристроїв. Вона пропонує високу швидкодію та ефективну роботу зі складними даними, що робить її привабливою для мобільних додатків з високою процесорною потужністю.

SQLite є бібліотекою баз даних SQLite, яка забезпечує доступ до локальних даних на мобільному пристрої. SQLite є широко використовуваною та надійною

базою даних, що підтримуєся на багатьох платформах. SQLite дозволяє просто працювати зі структурованими даними та виконувати SQL-запити.

Серед різних варіантів для нашого мобільного додатку ми вирішили обрати Firebase Realtime Database з таких причин:

Реальний час: Firebase Realtime Database пропонує синхронізацію даних в реальному часі. Це означає, що зміни, внесені на одному пристрої, автоматично відображаються на всіх інших підключених пристроях. Ця можливість особливо корисна для додатків, які потребують миттєвого оновлення даних, таких як чати, спільні списки або потоки подій [24].

Простота використання: Firebase Realtime Database має простий API, що дозволяє легко взаємодіяти з базою даних без необхідності писати складні SQL-запити. Це робить розробку швидшою і полегшує роботу з даними для розробників.

Широкі можливості: Firebase Realtime Database не обмежується лише збереженням структурованих даних. Він дозволяє зберігати й обробляти різні типи даних, включаючи текст, зображення, аудіо та відео. Це дає можливість створювати багатофункціональні додатки з різноманітними типами контенту.

Швидкодія та масштабованість: Firebase Realtime Database має високу швидкодію завдяки своїй реалізації на основі протоколу WebSocket. Він також може автоматично масштабуватись залежно від навантаження, що дозволяє оптимально працювати з даними в додатку незалежно від його розміру.

Інтеграція та розширення: Firebase Realtime Database легко інтегрується з іншими сервісами Firebase, такими як аутентифікація користувачів, збереження файлів, аналітика та багато іншого. Це дозволяє створювати повноцінні додатки з розширеними можливостями без необхідності використовувати різні сервіси окремо.

Підтримка та документація: Firebase має широку спільноту розробників і надає детальну документацію та приклади використання. Це спрощує процес навчання та вирішення можливих проблем.

Загалом, Firebase Realtime Database є потужним і гнучким інструментом для роботи з базою даних мобільного додатку. Його здатність працювати в реальному часі, простота використання, широкі можливості та інтеграція з іншими сервісами роблять його відмінним вибором для нашого проекту.

Також важливим є обрати потрібне IDE для якісної розробки мобільного додатку. Серед IDE ми розглянемо [26]:

Visual Studio Code:

- Простота використання: Visual Studio Code має інтуїтивний інтерфейс, легку навігацію та простий встановлювач. Його розширення та налаштування можна виконувати без зайвих зусиль, що робить його дружнім для початківців.
- Розширюваність: Visual Studio Code має потужну систему розширень, що дозволяє додавати функціональність, включаючи підтримку різних мов програмування, автоматичне завершення коду, налаштування тем та багато іншого.
- Продуктивність: Visual Studio Code працює швидко і ефективно, навіть з великими проектами. Він має вбудовану систему контролю версій і потужні інструменти для роботи з Git, що полегшує спільну роботу в команді.
- Здатність до налагодження: Visual Studio Code надає розширені можливості для налагодження коду, включаючи точки зупинки, крокування по коду та перегляд змін змінних у режимі реального часу.

Android Studio:

- Простота використання: Android Studio спеціально розроблене для розробки мобільних додатків на базі Android, що робить його добре зорієнтованим

для розробників, що працюють з цією платформою. Воно надає інтуїтивний інтерфейс та шаблони проектів, які полегшують початок роботи.

- Розширюваність: Android Studio підтримує розширення, які дозволяють налаштувати середовище розробки під свої потреби. Він також має велику базу плагінів, що розширюють його функціональність.
- Продуктивність: Android Studio оптимізовано для роботи з проектами Android, що дозволяє розробникам ефективно виконувати завдання, такі як збірка, компіляція та розгортання додатків. Воно також надає інструменти для профілювання та оптимізації додатків.
- Здатність до налагодження: Android Studio має розширені можливості для налагодження Android-додатків. Він підтримує відстеження стеку викликів, перегляд змін змінних, профілювання продуктивності та багато іншого.

IntelliJ IDEA Community Edition:

- Простота використання: IntelliJ IDEA має чистий та добре організований інтерфейс, що дозволяє розробникам швидко орієнтуватися у середовищі розробки. Він надає розумні підказки, автоматичне завершення коду та інші інструменти, що сприяють простоті використання.
- Розширюваність: IntelliJ IDEA має потужну систему плагінів, що дозволяє розширити його функціональність. Розробники можуть встановлювати різноманітні плагіни для підтримки різних мов програмування, фреймворків та інструментів.
- Продуктивність: IntelliJ IDEA працює швидко та ефективно, забезпечуючи широкі можливості для розробки проектів різного розміру і складності. Воно надає інструменти для рефакторингу, аналізу коду та роботи з системами контролю версій.
- Здатність до налагодження: IntelliJ IDEA має вбудовану підтримку для налагодження коду. Воно надає можливість встановлювати точки зупинки, крокувати по коду, виводити значення змінних та виконувати інші дії, необхідні для відлагодження програм.

Emacs:

- Простота використання: Emacs має велику кількість команд та скорочень клавіш, що може вимагати трохи часу для оволодіння. Проте, коли ви освоїте його, Emacs надає потужні можливості редагування інтерактивного тексту.
- Розширюваність: Emacs базується на Lisp і надає розширену систему для написання скриптів та додатків. Розробники можуть розширювати функціональність Emacs, встановлюючи різноманітні пакети, створені спільнотою.
- Продуктивність: Emacs відомий своєю потужною системою редактора, що дозволяє розробникам ефективно редагувати та переглядати код. Він має розширені можливості для автоматичного вирішення завдань та виконання рутинних операцій.
- Здатність до налагодження: Emacs має базову підтримку для налагодження, але він не має таких розширених можливостей, як інші спеціалізовані IDE. Проте, його можна налаштувати для співпраці з іншими інструментами для налагодження, що забезпечують більш розширені можливості.

Під час аналізу різних варіантів IDE для нашого додатку, ми прийшли до висновку, що Visual Studio Code є найкращим вибором з нашого списку. Ось декілька причин, чому ми обираємо Visual Studio Code [23]:

Простота використання: Visual Studio Code має чистий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко розпочати роботу з ним. Він має прості інструменти редагування коду, автоматичне завершення коду та широку підтримку різних мов програмування.

Розширюваність: Visual Studio Code має потужну систему розширень, що дозволяє налаштувати його під власні потреби. Існує велика кількість розширень, доступних в магазині Visual Studio Code, що дозволяють розширити функціональність IDE, додати підтримку конкретних фреймворків та мов програмування.

Продуктивність: Visual Studio Code працює швидко і легко масштабується для роботи з проектами різного розміру та складності. Він має підтримку вбудованої системи керування версіями Git, інструменти для рефакторингу коду та розумне виявлення помилок.

Здатність до налагодження: Visual Studio Code надає вбудовану підтримку для налагодження коду, зокрема точок зупинки, крокування по коду та перегляду значень змінних. Це дозволяє ефективно відлагоджувати та виявляти помилки в додатку.

Спільнота та підтримка: Visual Studio Code є дуже популярним серед розробників і має активну спільноту, що допомагає вирішувати проблеми та надає різноманітні ресурси для самовдосконалення. Крім того, Visual Studio Code має широку документацію та підтримку з боку Microsoft.

Загалом, Visual Studio Code є потужним та гнучким інструментом, який надає нам усі необхідні можливості для розробки нашого додатку. Його простота використання, розширюваність, продуктивність та здатність до налагодження роблять його ідеальним вибором для нашого проекту.

2.2 Особливості розробки інтерфейсу користувача

Інтерфейс користувача в сучасних мобільних додатках відіграє вирішальну роль у забезпеченні успіху та популярності програми. Він є обличчям додатку, першим, що бачить користувач, і основним засобом спілкування між ним та програмним забезпеченням. Добре спроектований інтерфейс здатний не лише полегшити взаємодію з додатком, але й створити неперевершені враження, заохочуючи користувачів використовувати його регулярно. У контексті нашої магістерської роботи, яка присвячена розробці мобільної інформаційно-довідникової системи для студентів факультету "Факультет інформаційних і прикладних технологій," важливість інтерфейсу надається особливого значення.

Основною метою розділу "Розробка інтерфейсу користувача" є створення інтерфейсу, який відповідає потребам і очікуванням користувачів факультету,

забезпечуючи їм зручний, інтуїтивно зрозумілий та ефективний спосіб взаємодії з додатком. Ми прагнемо створити інтерфейс, який стане не лише інформаційним довідником, але й зручним супутником для студентів у навчанні та повсякденному житті [39].

В даному розділі ми розглянемо ключові аспекти розробки інтерфейсу користувача, включаючи визначення головних цілей та методологію проектування. Ми також вивчимо інструменти та технології, які будуть використовуватися при створенні інтерфейсу, та розглянемо питання тестування та підтримки.

Важливість Інтерфейсу Користувача.

Інтерфейс користувача є невід'ємною частиною сучасного мобільного додатку, і від його якості залежить багато факторів. Ось кілька ключових аспектів, які роблять ІК таким важливим [40, 42]:

Позитивне перше враження: інтерфейс є першим, що бачить користувач, і від нього залежить перше враження від додатку. Перший враження може визначити, чи продовжить користувач використовувати додаток чи вибере конкурента.

Взаємодія та зручність: Добре спроектований інтерфейс робить взаємодію з додатком зручною та приємною. Він дозволяє користувачам легко здійснювати потрібні дії і знаходити необхідну інформацію.

Залученість та вірність: Привабливий та інтуїтивно зрозумілий інтерфейс може стати фактором, який збуджує цікавість та зацікавленість користувача. Він спонукає їх використовувати додаток регулярно та надійно.

Бренд та репутація: Якість інтерфейсу впливає на сприйняття бренду та репутацію компанії. Професійно розроблений інтерфейс може підвищити довіру користувачів.

Ефективність: Ефективний інтерфейс допомагає користувачам швидко досягати своїх цілей, що важливо для додатків, спрямованих на досягнення конкретних завдань.

Дизайн інтерфейсу з використанням фреймворку Flutter.

Наш загальний стиль і візуальна концепція інтерфейсу базуються на таких принципах:

Мінімалізм: Ми прагнемо до мінімалістичного дизайну, який спрощує сприйняття інтерфейсу та робить його інтуїтивно зрозумілим для користувачів. Чисті лінії, простий шрифт та мінімум зайвих деталей дозволяють концентруватися на важливій інформації.

Кольорова палітра: Наша кольорова палітра базується на офіційних кольорах факультету, що сприяє ідентифікації та розпізнаваності додатку серед користувачів. Вона також включає нейтральні кольори для забезпечення консистентності та зручності читання.

Шрифти: Ми використовуємо чіткі та читабельні шрифти, що дозволяють користувачам зручно читати текст та отримувати інформацію без зайвих зусиль. Іконки у нашому додатку відповідають стандартам Material Design, що полегшує їхнє розпізнавання та розуміння.

Просторий Дизайн: Ми використовуємо просторий дизайн, що дозволяє інформації "дихати" та зменшує візуальний шум. Просторий дизайн також зробить інтерфейс більш адаптивним для різних розмірів екранів мобільних пристроїв.

Вибір Кольорової Палітри, Шрифтів та Іконок.

Кольорова палітра: Наша кольорова палітра включає основні фарби факультету, такі як синій та жовтий, які символізують інформаційні технології та знання. Ці яскраві кольори використовуються для виділення важливих елементів і акцентів в додатку. Ми також використовуємо нейтральні кольори, такі як сірий і білий, для фонів та контрасту.

Шрифти: Наші шрифти обрані на основі їхньої читабельності та елегантності. Для заголовків та акцентних елементів використовується більш жирний шрифт, який виділяється на фоні загального тексту.

Іконки: Ми використовуємо набір стандартних іконок з бібліотеки Material Icons для забезпечення консистентності та зручності розпізнавання.

Використання Принципів Material Design. Ми дотримуємося принципів Material Design, розроблених Google, для створення нашого інтерфейсу. Ці принципи включають в себе різноманітні ефекти анімації, роблять інтерфейс інтуїтивно зрозумілим, та підтримують відповідність розмірів та пропорцій. Ми використовуємо такі елементи, як тіні, рух, зміну розміру, для того, щоб забезпечити зручну та привабливу взаємодію користувачів з додатком.

Взаємодія користувача

Опис взаємодії користувача з додатком на різних етапах використання.

Взаємодія користувача з нашим мобільним додатком для факультету "Факультет інформаційних і прикладних технологій" розглядається на різних етапах використання, включаючи перший запуск, реєстрацію, авторизацію, основний функціонал та виходи з додатку.

1. Перший запуск додатку:

Після завантаження додатку користувач бачить екран вітання, на якому представлена основна інформація про можливості додатку та запросила на реєстрацію або авторизацію. Користувач може перейти до реєстрації або авторизації, або просто переглянути інформацію.

2. Реєстрація:

Користувач обирає опцію реєстрації і заповнює необхідні поля, такі як ім'я, прізвище, адреса електронної пошти та пароль. Після успішної реєстрації, користувач отримує підтвердження та може перейти до авторизації.

3. Авторизація:

Користувач вводить свою адресу електронної пошти та пароль для входу в додаток. Після успішної авторизації користувач переходить до головного екрану додатку.

4. Головний екран:

На головному екрані користувач отримує доступ до основного функціоналу додатку, такого як перегляд розкладу занять, завдань, новин та інших корисних інформаційних розділів. Користувач може переходити між різними секціями додатку, використовуючи навігаційну панель або відповідні кнопки.

5. Виходи з додатку:

Користувач може завершити сеанс, натиснувши на відповідну опцію в меню. В цьому випадку користувач виходить зі свого облікового запису і повертається на екран авторизації.

Розробка сценаріїв взаємодії та управління переходами між екранами

Сценарій реєстрації:

- Користувач вибирає опцію реєстрації.
- Відкривається форма реєстрації з полями для введення даних.
- Користувач вводить свої особисті дані та натискає кнопку "Зареєструватися".
- Після успішної реєстрації, користувач переходить на екран авторизації.

Сценарій авторизації:

- Користувач вводить свою адресу електронної пошти та пароль.
- Натискає кнопку "Увійти".
- При успішній авторизації, користувач переходить на головний екран додатку.

Сценарій взаємодії на головному екрані:

- Користувач має доступ до різних секцій (розклад занять, завдання, новини тощо) через навігаційну панель або меню.
- Вибравши секцію, користувач переходить на відповідний екран з можливістю перегляду вмісту та взаємодії з ним.

Сценарій виходу з додатку:

- Користувач може завершити сеанс на будь-якому екрані додатку.
- Після виходу користувач повертається на екран авторизації.

Розробка сценаріїв взаємодії та управління переходами між екранами забезпечить зручну та інтуїтивно зрозумілу взаємодію користувачів з нашим додатком, що є важливим аспектом розробки інтерфейсу користувача.

Інтерфейсні елементи

Під час розробки інтерфейсу нашого мобільного додатку ми вибрали різні інтерфейсні елементи з урахуванням їхньої придатності та зручності для користувачів. Нижче пояснено вибір окремих елементів:

Кнопки "Увійти" та "Зареєструватися": Ці кнопки використовуються для створення або входу в обліковий запис користувача. Вони розташовані на головному екрані додатку для зручності доступу. Кнопки "Вийти" або "Вихід": Ці кнопки дозволяють користувачам завершити сеанс. Вони можуть використовуватися на багатьох екранах для зручного виходу з облікового запису.

Заголовки та підзаголовки: Використовуються для найменування розділів і секцій додатку, щоб користувачі могли швидко зрозуміти, що вони переглядають. Тексти опису: Використовуються для надання користувачам додаткової інформації, пояснень або інструкцій.

Списки завдань та новин: Використовуються для відображення інформації у вигляді списку, де користувачі можуть швидко переглядати та вибирати потрібні елементи.

Розгляд структури інтерфейсу та розміщення елементів на сторінці

Структура інтерфейсу нашого додатку ретельно виважена з метою забезпечення зручності та логічного розташування елементів. Основні принципи розміщення елементів на сторінці включають:

Меню та Навігація розміщені зверху або відкриваються за допомогою кнопки "Меню". Воно містить посилання на різні секції додатку, що полегшує навігацію.

Заголовок - назва розділу чи секції розміщена у верхній частині екрана для ідентифікації поточного контексту.

Елементи вмісту, такі як списки завдань чи новин, розміщені в центральній частині сторінки.

Кнопки для дій (наприклад, "Вийти" або "Створити завдання") розташовані в нижній частині екрана, щоб залучити увагу користувача після перегляду вмісту.

Інші елементи - тексти опису або інформаційні блоки можуть бути вбудовані в контент для надання додаткової інформації.

Ця структура дозволить користувачам легко переглядати та взаємодіяти з додатком, забезпечуючи при цьому приємний інтерфейс користувача.

Адаптивний дизайн.

Адаптивність інтерфейсу до різних розмірів екранів та орієнтацій є однією з ключових вимог до нашого мобільного додатку [43]. Для досягнення цієї мети ми використовуємо фреймворк Flutter, який надає потужні інструменти для розробки адаптивних інтерфейсів.

Основні підходи до адаптивного дизайну включають наступне:

Резиновий дизайн: Ми використовуємо резинові макети та компоненти, які розтягуються та зменшуються відповідно до розміру екрану. Це дозволяє

зберігати логічну структуру елементів і зручність використання, навіть на дуже малих або великих екранах.

Адаптивні шрифти і розміри: Ми використовуємо відносні одиниці для задання розмірів тексту і інших елементів, що дозволяє їм автоматично адаптуватися до розміру екрану.

Завантаження контенту за запитом: Деякі елементи контенту, наприклад, списки або зображення, завантажуються за запитом, враховуючи обмеження швидкості Інтернету та розмір екрану. Це дозволяє зменшити споживання ресурсів і покращити швидкість завантаження.

Орієнтація екрану: Наш додаток підтримує обидві орієнтації екрану (горизонтальну та вертикальну) і надає користувачам однаково зручний доступ до функціоналу в обох режимах.

Завдяки цим підходам, наш додаток забезпечує консистентну та зручну взаємодію з користувачем, незалежно від пристрою, на якому він використовується.

Під час розробки інтерфейсу нашого мобільного додатку для студентів факультету, ми маємо перед собою кілька головних цілей:

1. Зручність використання: Наш інтерфейс повинен бути легким у використанні та інтуїтивно зрозумілим для всіх користувачів. Ми створюємо інструмент, який спрощує їхнє життя та навчання.

2. Інформаційна зрозумілість: Ми надаємо студентам доступ до важливої інформації про факультет та навчання, і ця інформація має бути доступною та зрозумілою.

3. Стимулювання використання: Наша мета - створити інтерфейс, який стимулює студентів регулярно користуватися додатком, надаючи їм корисну та цікаву інформацію.

4. Відповідність потребам: Інтерфейс повинен відповідати потребам наших користувачів, надаючи їм необхідну функціональність та можливості.

У цьому розділі ми детально розглянули процес розробки інтерфейсу для нашого мобільного додатку. Інтерфейс користувача є критично важливою частиною нашого проекту, оскільки він є основним інструментом взаємодії користувача з додатком.

В завершенні, важливо відзначити, що інтерфейс користувача є ключовою складовою нашого додатку і має прямий вплив на зручність та задоволення користувачів від його використання. Відповідно до наших цілей розробки, ми прагнемо створити інтерфейс, який допоможе користувачам ефективно використовувати нашу мобільну інформаційно-довідкову систему та досягати найкращих результатів у своєму навчанні та діяльності на факультеті.

2.3 Особливості бази даних

Першим і важливим кроком у розробці бази даних є вибір типу бази даних, який найкраще підходить для наших потреб. В нашому випадку, ми використовуєм базу даних Firebase Realtime Database.

База даних Firebase Realtime — це база даних, розміщена в хмарі. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним клієнтом. Коли ви створюєте міжплатформні програми за допомогою наших платформ Apple, Android і JavaScript SDK, усі ваші клієнти спільно використовують один екземпляр Realtime Database і автоматично отримують оновлення з найновішими даними [27, 29].

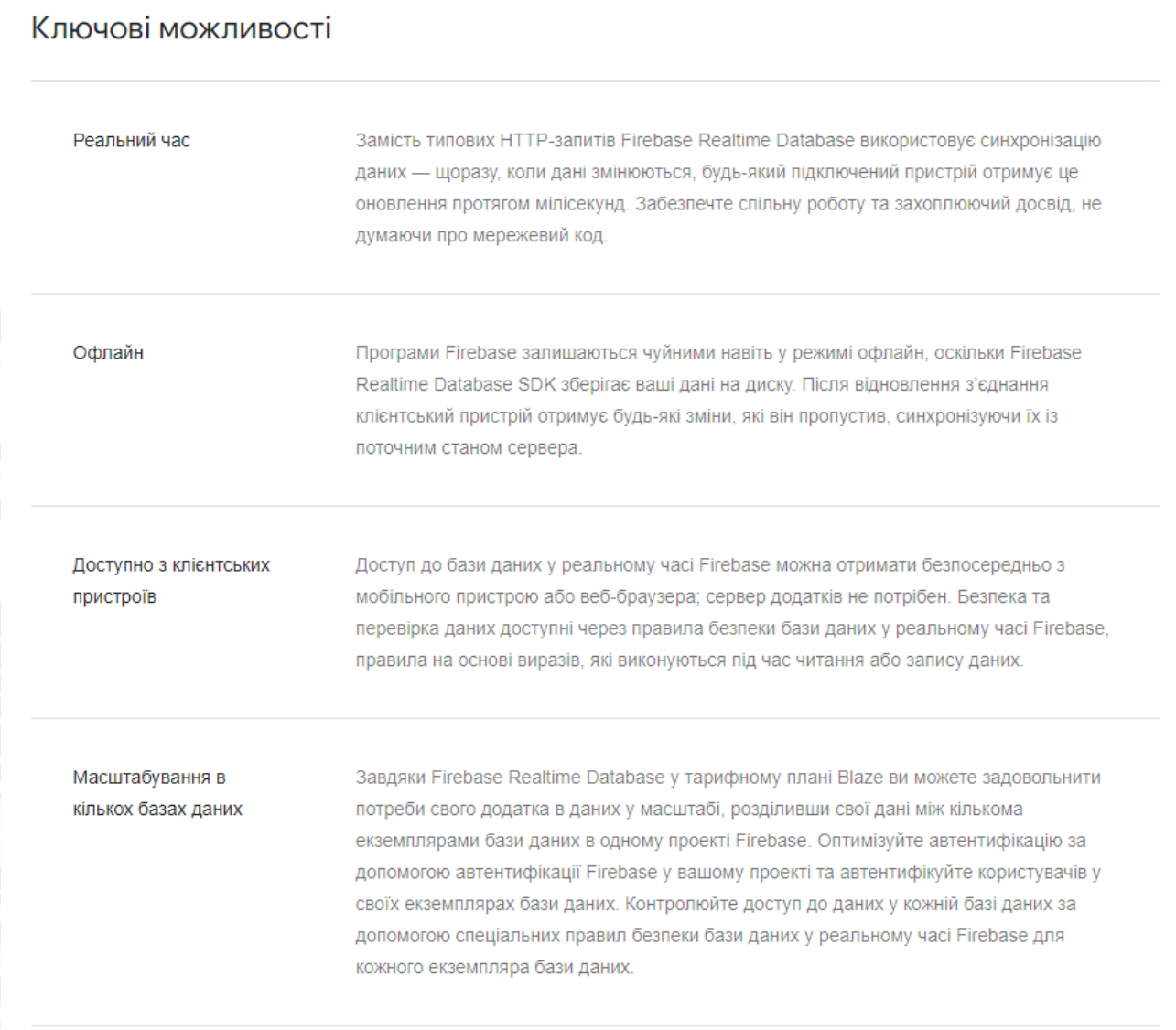


Рисунок 2.1 – Ключові можливості Firebase Realtime Database

Для нашого додатку "Мобільна інформаційно-довідникова система" база даних має виконувати кілька важливих завдань:

Збереження даних про користувачів - ми повинні мати можливість реєстрації та аутентифікації користувачів факультету. Firebase Realtime Database дозволяє зберігати дані користувачів та забезпечувати безпеку входу до системи.

Збереження актуальної інформації - база даних має зберігати дані про розклад занять, новини, оголошення і іншу актуальну інформацію для здобувачів. Firebase Realtime Database надає можливість оновлення цих даних в реальному часі, щоб користувачі завжди мали доступ до оновленої інформації.

Синхронізація між пристроями - оскільки наш додаток має підтримувати використання на різних мобільних пристроях та платформах, важливо мати базу даних, яка забезпечить синхронізацію даних між усіма пристроями користувачів.

Масштабованість - наш додаток може зростати та отримувати все більше користувачів з часом. Firebase Realtime Database дозволяє масштабувати обсяг даних та кількість одночасних підключень, що робить її відмінним вибором для майбутнього розвитку нашого додатку.

Загалом, Firebase Realtime Database відповідає всім цим вимогам і надає нам потужну інфраструктуру для збереження та управління даними, які необхідні для успішної реалізації нашого проекту.

2.4 Особливості тестування та налагодження

Тестування та налагодження є важливою частиною розробки мобільного додатку для нашої інформаційно-довідкової системи факультету інформаційних і прикладних технологій. Цей розділ розглядає основні аспекти та процеси тестування та налагодження нашого додатку.

Тестування на реальних пристроях. Ми будемо проводити тестування на реальних мобільних пристроях з різними версіями операційних систем, розмірами екранів і характеристиками. Це дозволить виявити можливі проблеми, які можуть виникнути на конкретних пристроях і забезпечить сумісність додатку з різними конфігураціями [28, 33-35].

Тестування на емуляторах. Окрім тестування на реальних пристроях, ми також використовуватимемо емулятори для перевірки додатку на різних версіях Android і iOS. Це дозволить нам переконатися, що додаток працює правильно на різних платформах.

Знаходження та виправлення помилок. Важливою частиною процесу тестування є знаходження та виправлення помилок. Коли помилка виявляється, ми відразу ж фіксуємо її відповідно до найкращих практик розробки:

Записуємо опис помилки та умови, в яких вона виникла.

Встановлюємо точку зупинки (breakpoint) в коді, де відбулася помилка, для подальшого аналізу.

Аналізуємо стек викликів та значення змінних для з'ясування причини помилки.

Вносимо необхідні зміни в код для виправлення помилки.

Повторно проводимо тести, щоб переконатися в виправленні помилки і відсутності нових проблем.

Тестування в реальних умовах. Крім розробника, тестуванню також підлягає інтерфейсний дизайн і функціональність програмного забезпечення в реальних умовах. Наприклад, ми можемо оцінити, наскільки швидко завантажується додаток при поганих мережевих умовах, які можуть виникнути в реальному використанні.

Перевірка документації. Документація є важливою частиною процесу тестування. Ми переконуємося, що всі функції і можливості додатку описані в документації для користувачів, щоб забезпечити їхню доступність та зрозумілість.

Оптимізація та виправлення продуктивності. Оптимізація продуктивності також є важливим аспектом тестування. Ми перевіряємо швидкість виконання додатку на різних пристроях і забезпечуємо, щоб він працював рівномірно і без затримок.

Тестування безпеки. Забезпечення безпеки додатку - ще одна важлива складова тестування. Ми проводимо тестування на вразливості (vulnerability testing) та тестування на відповідність стандартам безпеки (security standards compliance testing) для захисту даних користувачів і забезпечення конфіденційності.

Регресійне тестування. Після внесення змін або виправлення помилок ми виконуємо регресійне тестування для переконання в тому, що внесені зміни не вплинули на раніше працюючі частини додатку.

Звітність та документування. Під час тестування ми створюємо звіти, в яких фіксуються всі виявлені помилки та виконані тести. Ці звіти допомагають нам відстежувати прогрес тестування та робити прийняття рішень щодо випуску додатку. Також ми документуємо процес тестування для подальшого аналізу та можливої оптимізації.

Процес тестування та налагодження є важливою частиною розробки мобільного додатку для нашої інформаційно-довідкової системи факультету інформаційних і прикладних технологій. Від правильно проведеного тестування залежить якість, надійність та безпека додатку. Ми вдосконалюватимемо та виправлятимемо додаток на всіх етапах розробки, щоб забезпечити задоволення користувачів та оптимальний досвід використання.

2.5 Особливості забезпечення безпеки

В сучасному світі зростаюча кількість мобільних додатків і значущість цифрових даних покладають на нас велику відповідальність за забезпечення безпеки відомостей користувачів. У нашому додатку забезпечення безпеки є однією з найважливіших складових.

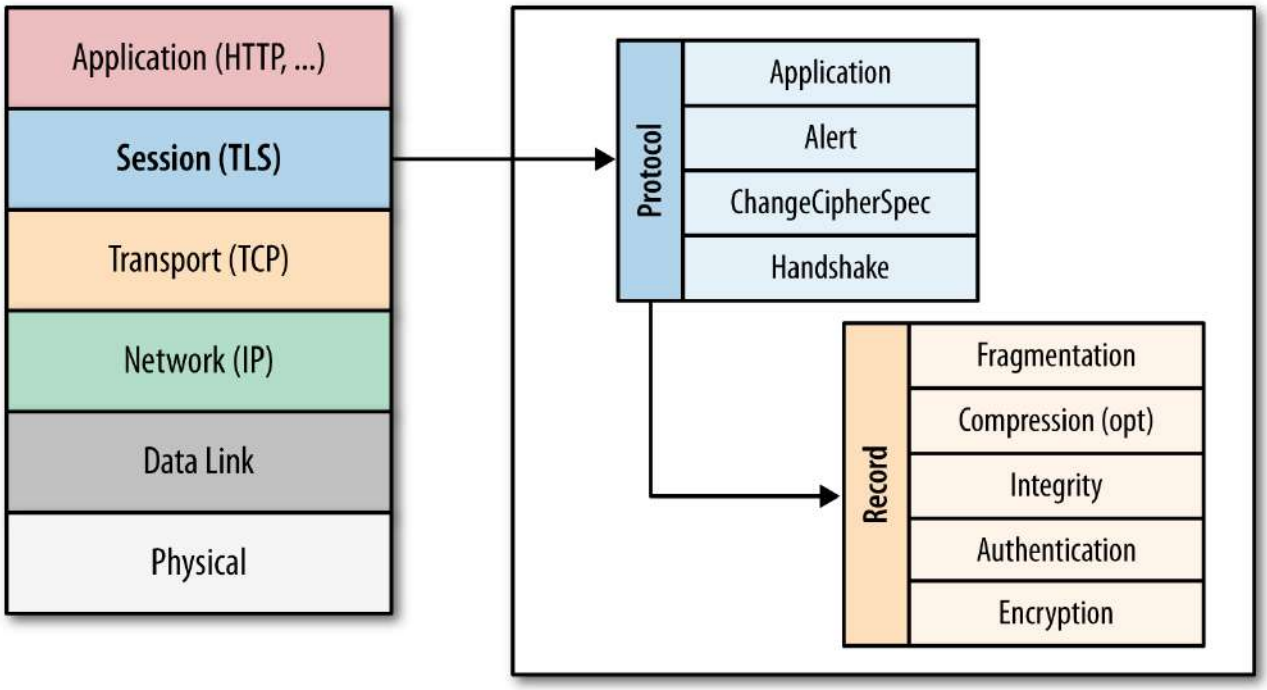


Рисунок 2.2 – Місце протоколу TLS (SSL) в стеці протоколів Інтернету

Аналіз загроз і ризиків.

Першим кроком у забезпеченні безпеки нашого додатку був аналіз потенційних загроз і ризиків. Ми визначили можливі сценарії атак та вразливості, які можуть використовувати зловмисники для доступу до даних користувачів. Це включає в себе атаки на автентифікацію, витоки даних, перехоплення сесій, а також можливість використання додатку для відправки шкідливих запитів на сервери [44-46].

Захист даних користувачів.

Однією з основних мет цієї роботи є захист особистих даних користувачів. Для цього ми використовуємо сучасні методи шифрування при зберіганні і передачі даних. Всі дані, які передаються через мережу, шифруються з використанням протоколів HTTPS та TLS, що гарантує конфіденційність та цілісність інформації.

Аутентифікація та авторизація.

Для захисту від несанкціонованого доступу ми використовуємо сильну систему аутентифікації та авторизації користувачів. Це включає в себе вимогу до

пароля для доступу до особистого кабінету, використання біометричних даних для підтвердження особи, та обмеження прав доступу користувачів до різних функцій додатку.

Моніторинг та виявлення атак.

Ми розробили систему моніторингу, яка відстежує підозрілу активність та можливі атаки нашого додатку. Це дозволяє нам вчасно виявляти порушення та втручатися для захисту даних користувачів. Також ми використовуємо внутрішні журнали для реєстрації подій та дій користувачів для подальшого аналізу.

Навчання та оновлення.

Оскільки загрози безпеці постійно змінюються, ми зосереджуємося на постійному навчанні та оновленні наших заходів безпеки. Ми слідкуємо за новими вразливостями та методами атак і вживаємо відповідних заходів для їх запобігання.

Забезпечення безпеки нашого мобільного додатку - це наш пріоритет. Ми розуміємо важливість конфіденційності та цілісності даних користувачів і приділяємо особливу увагу захисту їх інформації. Наша система безпеки включає в себе шифрування даних, сильну автентифікацію, моніторинг та виявлення атак, а також постійне навчання та оновлення. Наша мета - зробити наш додаток надійним та безпечним для всіх користувачів.

2.6 Вибір оптимальних рішень

Вибір технологій та інструментів є однією з ключових складових успішної розробки мобільного додатку. Правильно обрані технології можуть позитивно позначитися на продуктивності, якості та майбутньому розвитку проекту. У цьому розділі ми розглянемо процес вибору технологій та інструментів для нашого мобільного додатку, враховуючи вимоги проекту, наявні ресурси команди та перспективи майбутнього розвитку.

Рисунок 2.3 – Етапи життєвого циклу програмного забезпечення

Першим кроком в процесі вибору технологій є детальний аналіз вимог проекту. Ми ретельно вивчили потреби користувачів, функціональність, яку має підтримувати додаток, та інші специфікації проекту. На цьому етапі визначилися основні вимоги до технологічного стеку [48].

Наступним важливим кроком є врахування ресурсів команди розробників. Ми оцінили знання та досвід нашої команди у використанні конкретних технологій і інструментів. Це дозволило нам визначити, наскільки комфортно і швидко команда зможе приступити до розробки на обраних технологіях.

Ми також обрали технології, які підходять для майбутнього розвитку проекту. Розглядаючи перспективи розширення та вдосконалення додатку, ми обрали технології, які дозволять нам ефективно впроваджувати нові функції та зберігати сумісність з майбутніми версіями операційних систем.

На основі проведеного аналізу та врахування всіх факторів ми прийняли рішення щодо технологічного стеку для нашого мобільного додатку. Ми вирішили використовувати Flutter - фреймворк для розробки нативних мобільних додатків для платформ Android та iOS. Flutter надає можливість створення високоякісного інтерфейсу та швидкої розробки на обох платформах, що відповідає вимогам нашого проекту.

Вибір оптимальних технологій та інструментів є важливим кроком у розробці мобільного додатку. Враховуючи вимоги проекту, ресурси команди та перспективи майбутнього розвитку, ми визначилися з використанням Flutter як основного фреймворку для розробки. Це рішення дозволить нам створити високоякісний та ефективний додаток, який задовольнить потреби наших користувачів.

Висновок до розділу 2

У ході виконання розділу 2 було ретельно розглянуто та вирішено багато ключових завдань, спрямованих на розробку мобільної інформаційно-довідкової системи для факультету "Факультет інформаційних і прикладних технологій." Процес створення додатку був докладно структурований та включав в себе наступні етапи.

Аналіз вимог, передбачав аналіз потреб та вимог користувачів факультету. Це дозволило визначити основні функціональність і особливості системи, які мають відповідати вимогам користувачів.

Вибір технологій, було проведено глибокий аналіз та обрано оптимальний технологічний стек для розробки додатку. Мова програмування Dart і фреймворк Flutter були вибрані через їхню зручність, продуктивність та можливості створення крос-платформеного додатку.

Розділ 3. Практична частина

3.1 Розробка схеми архітектури

Перед початком розробки нашої інформаційно-довідкової системи, ми вживаємо кроків для чіткого визначення функціональності системи та способів взаємодії з нею. Цей процес допомагає нам створити структурований план розробки та забезпечити те, щоб система відповідала потребам користувачів. Один з головних інструментів, який ми використовуємо для цього, - діаграма використання (use-case).

Актори в нашій діаграмі відображають різні сутності, які будуть взаємодіяти з нашою системою. У нашому випадку це:

Студент: Це один з основних користувачів системи, який може використовувати багато функцій, таких як реєстрація, авторизація, перегляд розкладу та карти кабінетів.

Староста: Це ще один користувач системи, якому надається та ж сама функціональність, що і студенту, але додається ще публікування, редагування та закріплення важливої інформації.

Адміністратор: Цей актор відповідає за публікацію новин університету та іншу діяльність, пов'язану з редагуванням та публікацією важливої інформації.

Використання use-case представляють собою конкретні функції або можливості, які система надає своїм користувачам. Нижче перераховані важливі функції нашої системи:

Зареєструватись: Ця функція дозволяє користувачам створити обліковий запис в системі.

Авторизація: Вона дозволяє користувачам увійти в систему, підтвердивши свою ідентичність.

Розклад занять: Користувачі можуть переглядати розклад занять.

Карта кабінетів: Ця функція дозволяє користувачам переглядати карту кабінетів в університеті.

Редагування номерів кабінетів: Адміністратор має можливість редагувати та оновлювати інформацію про номери кабінетів.

Публікація новин університету: Адміністратор може публікувати новини та важливу інформацію на головній сторінці.

Видача ролі «староста»: Адміністратор системи може надати користувачеві роль "староста", що дає деякі додаткові права.

Публікація важливої інформації: Староста може публікувати важливу інформацію для користувачів.

Закріплення важливої інформації: Староста може закріпити важливу інформацію, щоб вона була видно користувачам.

Редагування важливої інформації: Староста може редагувати і оновлювати важливу інформацію.

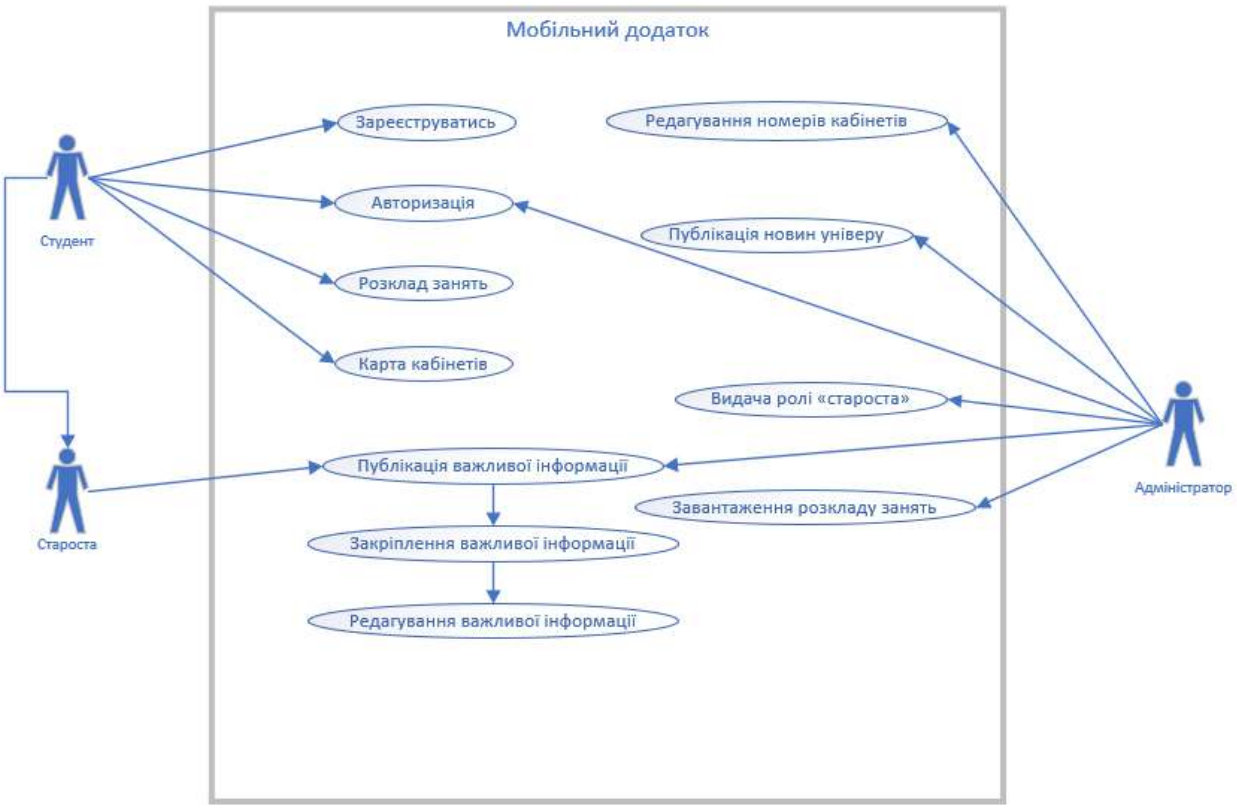


Рисунок 3.1 – Діаграма use-case

Діаграма використання допомагає нам визначити, які функції системи мають бути реалізовані та які актори взаємодіють з системою на різних етапах її роботи. Вона служить основою для подальшого проектування та розробки системи, забезпечуючи відповідність функціональності потребам користувачів [50].

3.2 Розробка дизайну додатку

Вибір мінімалістичного дизайну

При розробці дизайну нашого мобільного додатку для факультету "Факультет інформаційних і прикладних технологій" було прийнято рішення зосередитися на мінімалістичному дизайні. Мінімалістичний дизайн став популярним завдяки своїй простоті, зрозумілості і чіткості, що дозволяє користувачам легко взаємодіяти з додатком і зосереджувати увагу на основних функціях. Основні принципи мінімалістичного дизайну включають:

Простота, ми використовуємо мінімальну кількість кольорів, форм і декоративних елементів, щоб запобігти перенасиченому вигляду та розсіюванню уваги користувача.

Чіткість, всі елементи і інтерфейс повинні бути чіткими і легко розпізнаваними. Ми використовуємо зрозумілі піктограми та значки для швидкого розуміння функцій [30].

Легкість навігації, мінімалістичний дизайн сприяє легкій навігації і простому доступу до важливих функцій.

Дизайн авторизації

Для сторінки авторизації ми використовуємо мінімалістичний підхід з метою забезпечити користувачам зручність і швидкість входу.

На сторінці авторизації користувачу надається два основних поля для введення інформації: логін і пароль. Ці поля мають простий білий фон і чіткий чорний

шрифт, що забезпечує високу читабельність. Кнопка "Увійти" розміщена внизу форми і виділена контрастним кольором, щоб вона була видно і легко натискати.

Рисунок 3.2 – Макет авторизації

На сторінці реєстрації ми також залишаємося в межах мінімалістичного дизайну з акцентом на простоту та зручність.

Користувачеві пропонується заповнити кілька основних полів: Ім'я, Прізвище, логін, пароль і поле для повторного введення паролю для підтвердження. Поля для введення інформації мають аналогічний дизайн з чіткими місцями для введення тексту.

Ми враховуємо, що це важлива частина додатку, тому робимо її максимально зрозумілою і зручною для користувачів. Всі елементи розташовані таким чином, щоб користувач міг легко заповнити дані і перейти до використання додатку.

Такий дизайн дозволяє користувачам швидко та безпечно авторизуватися та реєструватися в додатку, не відволікаючись на зайві деталі і сприяє загальному позитивному враженню від використання додатку.

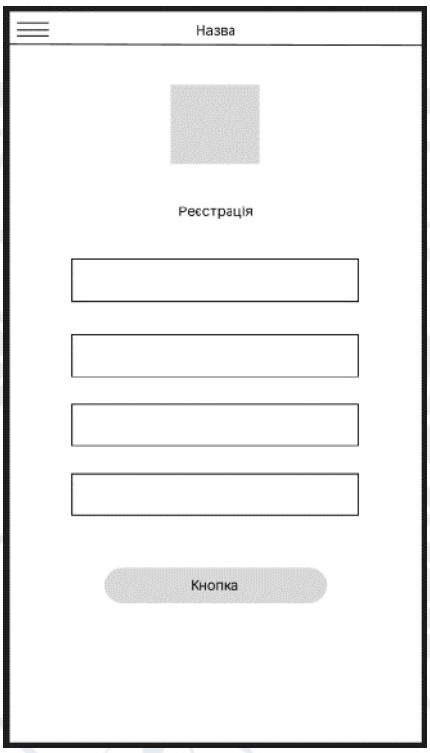


Рисунок 3.3 – Макет реєстрація

На головній сторінці мобільного додатку розміщено основні елементи і інформацію, яка дозволить користувачам швидко зорієнтуватися та отримати актуальну інформацію:

Верхнє меню, правому верхньому куті розміщено три горизонтальні смужки, що вказують на наявність більш докладного меню. Цей зручний інтерфейсний елемент дозволяє користувачам відкрити головне меню додатку з будь-якої сторінки.

Зображення, центральна частина сторінки відводиться для важливих зображень або фотографій, які можуть бути новинами, подіями чи іншою актуальною інформацією.

Підпис з новинами, нижче зображень розміщено короткий текстовий підпис, що супроводжує фотографії або інші елементи. В даному випадку, це може бути блок з новинами університету.

Важлива інформація, під блоком з новинами розташовано розділ з важливою інформацією. Ця інформація може включати оголошення, повідомлення від куратора чи старости, або інші актуальні дані.

Рисунок 3.4 – Макет головної сторінки

Меню, якщо користувач натисне на три смужки у верхньому правому куті, він отримає доступ до головного меню додатку. В меню розміщені вкладки для різних функцій та розділів:

Головна сторінка, повернення на головну сторінку, де користувач може переглянути новини та важливу інформацію.

Карта кабінетів, відкриття інтерактивної карти з розташуванням кабінетів та інших важливих місць на факультеті.

Розклад, перегляд розкладу занять для студентів, що дозволяє користувачам визначити час і місце проведення занять.

Налаштування додатку, де користувач може змінити особисті налаштування, які включають мову додатку, повідомлення та інше.

Вихід з облікового запису, опція виходу з облікового запису користувача, яка дозволяє забезпечити безпеку та конфіденційність даних.

Ця структура меню дозволяє легко навігувати додатком і швидко здійснювати доступ до різних функцій і розділів, сприяючи приємному користувацькому досвіду.

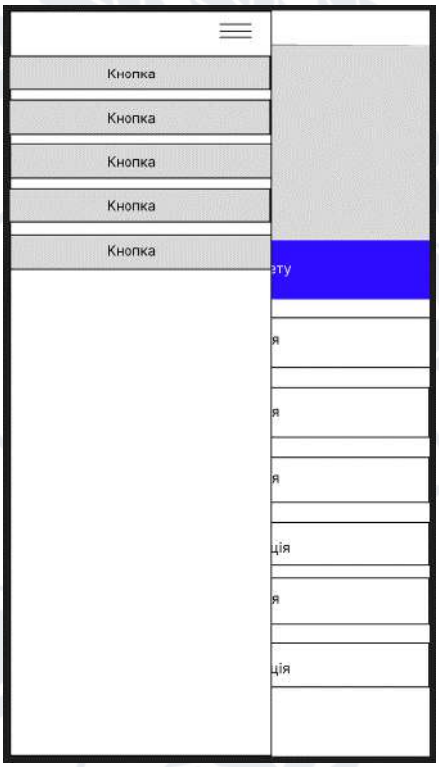


Рисунок 3.5 – Макет меню

В результаті роботи над інтерфейсом був створений дизайн, який сприяє зручному взаємодії користувачів з додатком і допомагає їм легко знаходити необхідну інформацію. Розглянуті дизайн-елементи дозволять користувачам швидко орієнтуватися в додатку та максимально використовувати його можливості.

3.3 Розробка функціоналу додатка

Розробка функціоналу додатка на основі мови програмування Dart та фреймворку Flutter є ключовим аспектом процесу створення мобільної інформаційно-довідкової системи для факультету "Факультет інформаційних і прикладних технологій." Правильний вибір технологій та ефективна розробка на

цій платформі визначають успіх проекту та його здатність задовольнити потреби користувачів.

Одноразовий код для різних платформ - Dart є мовою програмування, яка використовується для розробки на платформі Flutter. Однією з головних переваг є можливість створювати мобільні додатки для iOS та Android з використанням одного і того ж коду. Це спрощує розробку, підтримку та поновлення додатку.

Широкі можливості для UI - Flutter надає багато можливостей для створення інтерфейсу користувача відповідно до потреб проекту. За допомогою багатьох вбудованих інструментів і бібліотек, розробники можуть створювати красивий та інтуїтивно зрозумілий інтерфейс [21].

Швидкість та продуктивність - Dart має швидкий час виконання коду, що дозволяє створювати відзначено продуктивні додатки з плавним і відзначено швидким інтерфейсом.

Реєстрація та авторизація.

Реєстрація:

- Користувач запускає додаток та обирає функцію реєстрації.
- Вводить свої особисті дані, такі як ім'я, прізвище, обраний логін та пароль.
- Система перевіряє унікальність логіну та інші обов'язкові поля.
- Після успішної реєстрації користувач отримує підтвердження та може увійти в систему.

Авторизація:

- Після реєстрації або після запуску додатку користувач обирає функцію авторизації.
- Вводить свій логін та пароль.
- Система перевіряє введені дані з базою даних.
- У разі успішної авторизації користувач має доступ до особистого облікового запису та всіх функцій додатку.

Створення інтерфейсу для реєстрації та авторизації

Спочатку створимо потрібні сторінки інтерфейсу для реєстрації та авторизації. Ми можемо використовувати різні віджети, такі як TextField для введення інформації та RaisedButton для кнопок.

```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';

class RegistrationScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Реєстрація'),
      ),
      body: Center(
        child: Column(
          children: <Widget>[
            TextField(
              decoration: InputDecoration(labelText: 'Ім\'я Прізвище'),
            ),
            TextField(
              decoration: InputDecoration(labelText: 'Email'),
            ),
            TextField(
              decoration: InputDecoration(labelText: 'Пароль'),
              obscureText: true,
            ),
            TextField(
              decoration: InputDecoration(labelText: 'Повторіть пароль'),
              obscureText: true,
            ),
            RaisedButton(
                child: Text('Зареєструватись'),
                    onPressed: () async {
                        try {
                          await registerUser(nameController.text, emailContr
                          }
                          catch (e) {
                              print('Помилка реєстрації: $e');
```

Рисунок 3.6 – Програмна реалізація форми «Реєстрація»

Аналогічно, створіть сторінку для авторизації, замінивши RegistrationScreen на LoginScreen у коді вище.

Додавання бізнес-логіки.

Для обробки реєстрації та авторизації ми можемо використовувати Firebase, який надає надійний механізм авторизації та зберігання даних користувачів. Для використання Firebase в Flutter, додайте відповідні плагіни до «pubspec.yaml».

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^latest_version
  firebase_auth: ^latest_version
```

Рисунок 3.7 – Програмна реалізація додавання “pubspec.yaml”

Після цього ініціалізуйте Firebase у нашому додатку:

```
import 'package:firebase_core/firebase_core.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
```

Рисунок 3.8 – Програмна реалізація ініціалізації бази даних

Реєстрація та авторизація з використанням Firebase.

Для реєстрації та авторизації користувача використовуємо FirebaseAuth.

```
Future<void> registerUser(String email, String password) async {
  try {
    await FirebaseAuth.instance.createUserWithEmailAndPassword(
        name: name,
        email: email,
        password: password,
    );
  } catch (e) {
    print('Помилка реєстрації: $e');
```

Рисунок 3.9 – Програмна реалізація реєстрації користувача в базу даних

```
Future<void> signInUser(String email, String password) async {
  try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: email,
      password: password,
    );

  } catch (e) {
    print('Помилка авторизації: $e');
  }
```

Рисунок 3.10 – Програмна реалізація авторизації користувача в базу даних

```dart
Future<void> registerUser(String name, String email, String password) async {
  try {
    if (name.isEmpty || email.isEmpty || password.isEmpty) {
      // Перевірка на заповненість всіх полів
      print('Please fill in all fields.');
      return;
    }

    if (password.length < 6) {
      print('Password must be at least 6 characters long.');
      return;
    }

    UserCredential userCredential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(email: email, password: password);

    String userId = userCredential.user!.uid;

    await FirebaseDatabase.instance.reference().child('users').child(userId).set({
      'name': name,
      'email': email,
    });
  } on FirebaseAuthException catch (e) {
```

Рисунок 3.11 – Програмна реалізація перевірки авторизації користувача в базу даних

Розробка висувного меню є важливою частиною інтерфейсу додатка, оскільки вона надає користувачам зручний спосіб доступу до основних функцій та навігації в додатку. Висувне меню може містити різні елементи, такі як пункти меню, інформацію про користувача та інші корисні функції.

Основні елементи висувного меню, які ми реалізували, включають заголовок користувача. UserAccountsDrawerHeader містить ім'я користувача, адресу електронної пошти. Це допомагає ідентифікувати користувача та надає особистий характер.

Пункти меню, представлені за допомогою ListTile, кожен з яких має іконку і текст, які описують функцію або розділ додатку. При натисканні на пункт меню виконуються певні дії, такі як перехід на іншу сторінку додатку.

Відкриття меню кнопкою, для відкриття висувного меню використовується кнопка. При натисканні на цю кнопку висувне меню з'являється зліва.

Закриття меню, після вибору пункту меню або при натисканні поза висувним меню воно закривається, повертаючи користувача до основного вмісту додатку.

Висувне меню дозволяє зробити навігацію більш зручною для користувачів і забезпечити легкий доступ до різних функцій додатку. Це особливо корисно в додатках зі складним функціоналом, де важливо забезпечити структуровану і зрозумілу навігацію.

Процес взаємодії користувача з висувним меню полягає в тому, що користувач натискає на кнопку і меню висовується зліва. Після цього користувач може обирати пункти меню, які відповідають різним функціям додатку. Кожен пункт меню реалізований як окремий взаємодійний елемент, і при натисканні на нього виконується певна дія.

Загальна мета висувного меню - полегшити користувачам роботу з додатком, надаючи зручний доступ до основних функцій і зберігаючи зовнішній вигляд додатку лаконічним та організованим.

Для створення висувного меню, яке з'являється при натисканні кнопки зверху зліва, ми використовуємо віджет Drawer.

Спершу, нам потрібно додати кнопку на верхню панель нашого додатку. Ми можете використовувати для цього віджет AppBar.

```
AppBar(
  title: Text('Інформаційний довідниковий студента ДонНУ'),
  leading: Builder(
    builder: (BuildContext context) {
      return IconButton(
        icon: Icon(Icons.menu),
        onPressed: () {
          Scaffold.of(context).openDrawer();
```

Рисунок 3.12 – Програмна реалізація кнопки меню

Далі, ми використовуємо віджет Drawer для створення самого висувного меню.

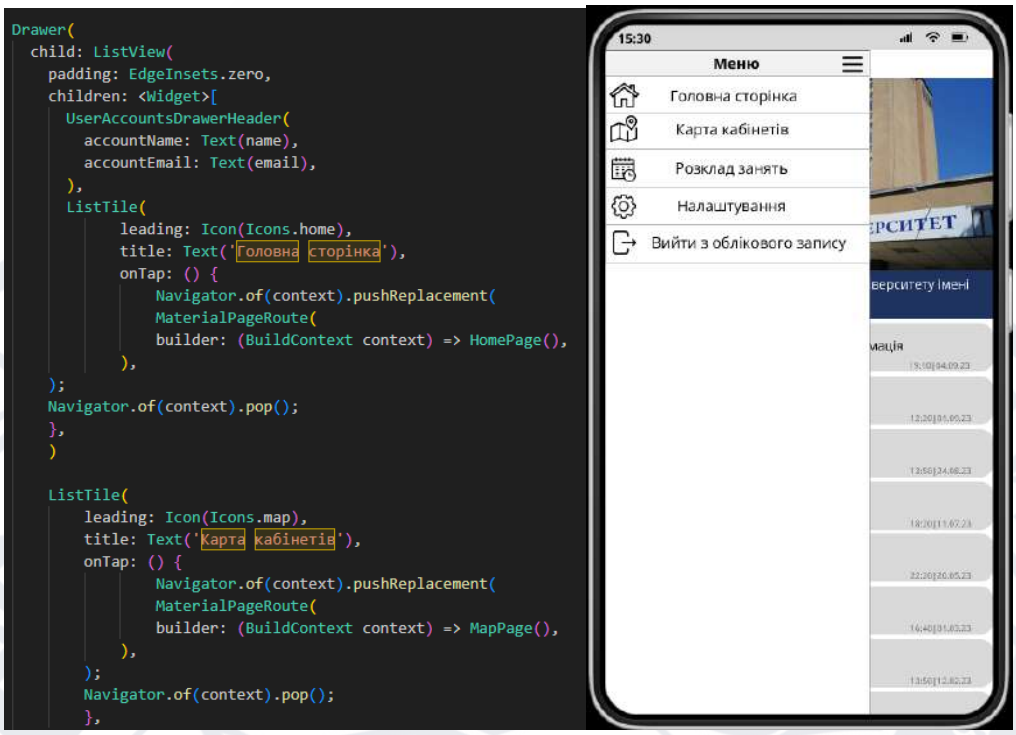


Рисунок 3.13 – Програмна реалізація меню

Таким чином, коли користувач натискає на кнопку в AppBar, висувне меню з'являється, і користувач може вибрати пункт меню для переходу на відповідну сторінку нашого додатку.

На головній сторінці мобільного додатку ми реалізуємо такий функціонал.

Зображення новини - Верхню частину сторінки займе зображення поточної новини. Це зображення відображатиметься зверху посередині сторінки та може бути розміщене як фонове зображення, яке допомагає візуально привертати увагу користувачів. Зображення новини може бути оновлюваним і змінюватися при оновленні новин на головній сторінці.

Текст новини - Нижче зображення новини розташований текст новини. Текст може включати заголовок, короткий опис і основний текст новини. Цей вміст може оновлюватися, коли нові новини додаються до системи.

Список важливої інформації - У нижній частині головної сторінки міститься список важливої інформації. Цей список містить різні важливі повідомлення або оголошення для користувачів. Важлива інформація може включати оголошення про події, оголошення факультету, інформацію від куратора, або будь-яку іншу

важливу інформацію. Користувачі можуть переглядати цей список і взаємодіяти з елементами цього списку для отримання додаткових деталей.

Взаємодія з елементами - Користувачі можуть натискати на елементи на головній сторінці для отримання більш докладної інформації або взаємодії з додатком. Наприклад, якщо користувач натисне на зображення новини, це може відкрити докладну сторінку цієї новини. Якщо користувач натисне на елемент важливої інформації, він може отримати більше деталей або перейти до пов'язаних функцій додатку.

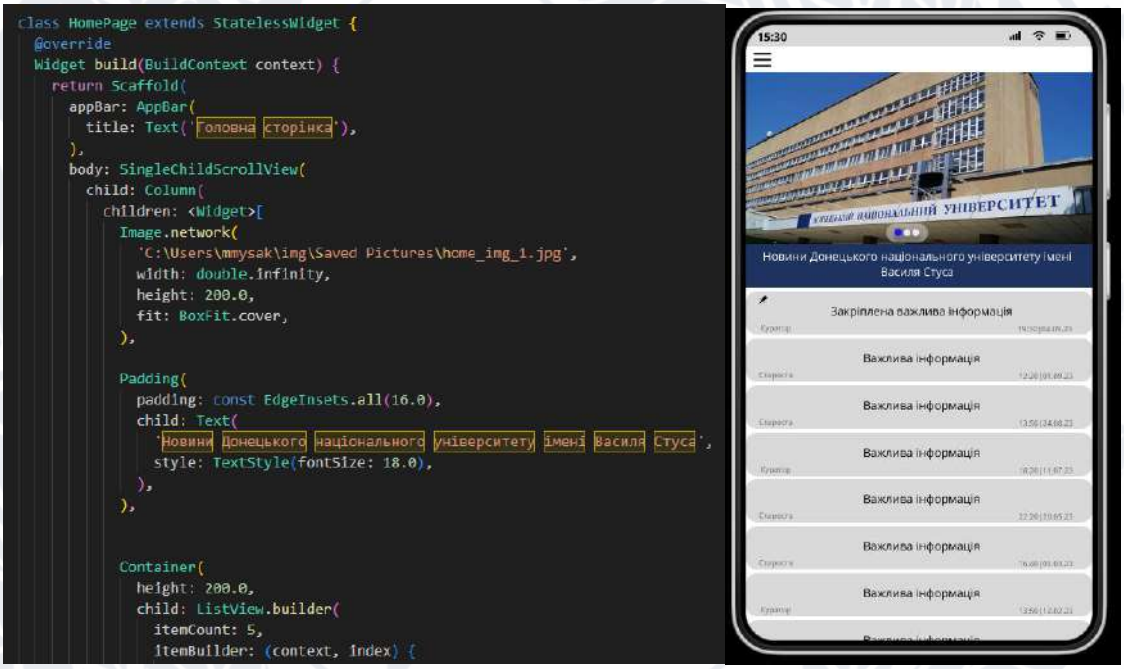


Рисунок 3.14 – Програмна реалізація головної сторінки

Важливою особливістю цієї головної сторінки є її динамічність і актуальність. Змінюючи зображення новини, текст і список важливої інформації, додаток може завжди надавати користувачам актуальну інформацію. Це робить головну сторінку центральним місцем для отримання оновлень та доступу до важливої інформації додатку.

На сторінці розкладу в мобільному додатку реалізовано відображення списку PDF-файлів із розкладом занять. Для цього використовується пакет pdf_flutter, який дозволяє відкривати та переглядати PDF-файли безпосередньо у додатку.

```
dependencies:
pdf_flutter: ^2.0.0
```

Рисунок 3.15 – Ініціалізація pdf_flutter

Основні елементи сторінки розкладу:

Заголовок сторінки. У верхній частині сторінки розташований заголовок, який інформує користувача про те, що це сторінка з розкладом.

Список PDF-файлів. Головна частина сторінки містить список доступних PDF-файлів з розкладом. Кожен файл в списку включає ім'я розкладу та дату його створення.

Обробник натискань. Для кожного елемента списку PDF-файлів доданий обробник події onTap, який реагує на натискання користувача. При натисканні на файл, відкривається відповідний PDF-файл для перегляду.

Відкриття PDF-файлів. При натисканні на файл відкривається вбудований переглядач PDF-файлів. Користувач може переглядати зміст PDF і використовувати функції, такі як збільшення, зменшення, перегортування та інші.

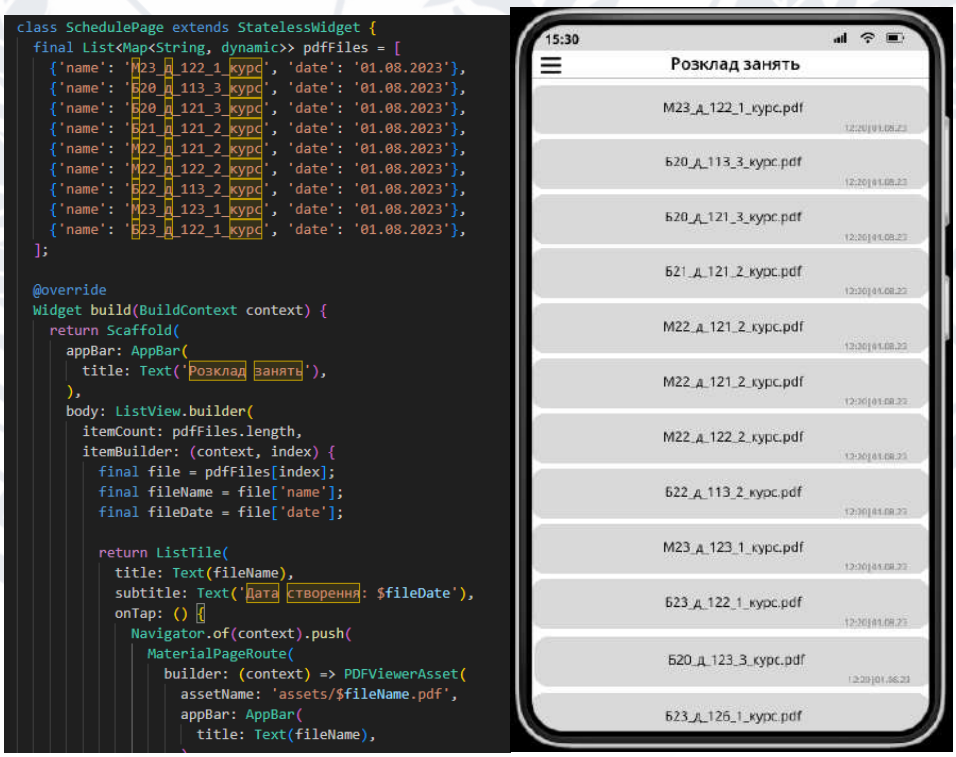


Рисунок 3.16 – Програмна реалізація сторінки Розклад занять

Ця сторінка розкладу створена з урахуванням зручності користувача та надає доступ до важливої інформації про розклад занять на факультеті. Користувач може легко переглядати та завантажувати PDF-файли із розкладом, що спрощує доступ до необхідної інформації.

Однією з ключових частин будь-якого мобільного додатку є його налаштування. Створення інтуїтивно зрозумілої та зручної сторінки "Налаштування" допомагає користувачам ефективно керувати функціоналом додатку.

Сторінка "Налаштування" у нашому мобільному додатку виконує важливі функції для керівництва і контролю за інформацією та ролями користувачів. На сторінці "Налаштування" користувач має можливість побачити свою роль "Студент", "Староста" та "Куратор".

На верхній частині сторінки розміщено заголовок "Налаштування" з іконкою персонажа поруч із текстом "Права: Куратор". Це створює чітку ідентифікацію ролі користувача.

Далі в залежності від права користувача є можливість обрати такі функції як "Функція публікування важливої інформації", "Функція публікування новин університету", "Функція публікування карти кабінетів", "Функція публікування розкладу занять".

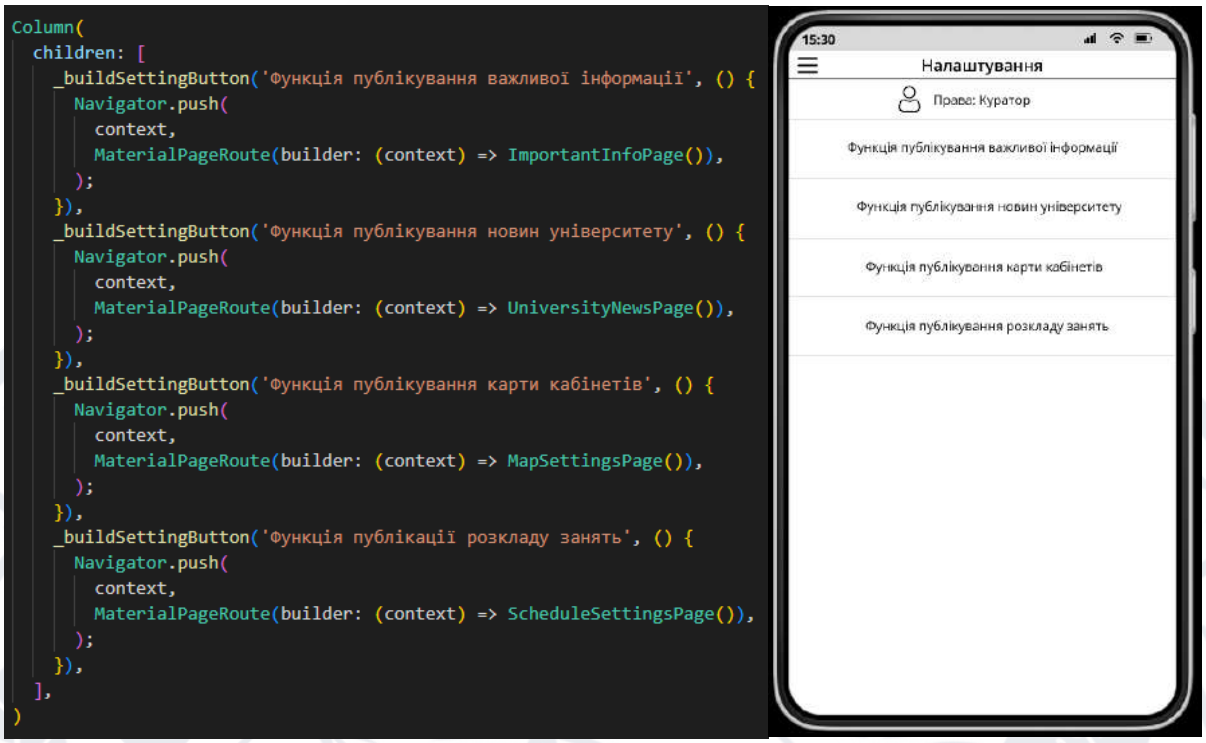


Рисунок 3.17 – Програмна реалізація головної сторінки Налаштування

Після обраної функції "Функція публікування важливої інформації" слідує текстове поле для введення тексту зі спливаючим текстом "Введіть текст". Це поле дозволяє користувачеві вводити необхідний контент.

Після текстового поля використано кнопку "Закріпити/Відкріпити" з текстом "Закріпити повідомлення". Ця кнопка служить для закріплення або відкріплення повідомлення.

Нижче знаходиться випадаючий список, де користувач може обрати, вид публікації із варіантів: "Публікація новини університету", "Публікація важливої інформації", "Публікація карти кабінетів", "Публікація розкладу занять". Це важливо для визначення виду публікації і можливості змінити його за потреби.

У кінці сторінки розміщена кнопка "Відправити", яка ініціює відправлення введеного користувачем контенту на головну сторінку. Ця кнопка запускає обробник для обробки та збереження змін.

За допомогою кнопок, користувач може керувати інформацією на головній сторінці додатку. Кнопка "Публікувати інформацію" дозволяє користувачу додавати важливу інформацію на головну сторінку. Кнопка "Редагувати

інформацію" надає можливість вносити зміни до вже існуючої інформації, а кнопка "Закріпити інформацію" дозволяє закріплювати важливу інформацію у верхній частині списку, щоб вона була легше доступною для всіх користувачів.

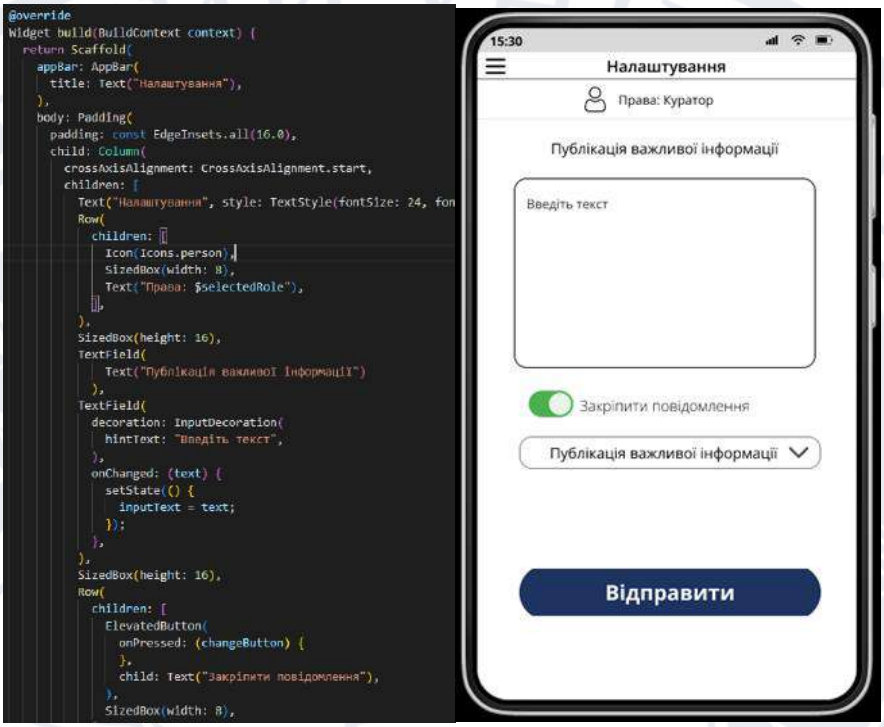


Рисунок 3.18 – Програмна реалізація сторінки Публікація важливої інформації

Також можна обрати таку функцію як "Функція публікування новин університету" слідує текстове поле для введення тексту зі спливаючим текстом "Введіть текст". Це поле дозволяє користувачеві вводити необхідний контент.

Після текстового поля використано кнопку "Закріпити/Відкріпити" з текстом "Закріпити повідомлення". Ця кнопка служить для закріплення або відкріплення повідомлення.

Нижче знаходиться випадаючий список, де користувач може обрати, вид публікації із варіантів: "Публікація новини університету", "Публікація важливої інформації", "Публікація карти кабінетів", "Публікація розкладу занять". Це важливо для визначення виду публікації і можливості змінити його за потреби.

Є змога завантажити картинку для нашої публікації, яка буде доповнювати текстову інформацію.

У кінці сторінки розміщена кнопка "Відправити", яка ініціює відправлення введеного користувачем контенту на головну сторінку. Ця кнопка запускає функцію для обробки та збереження змін.

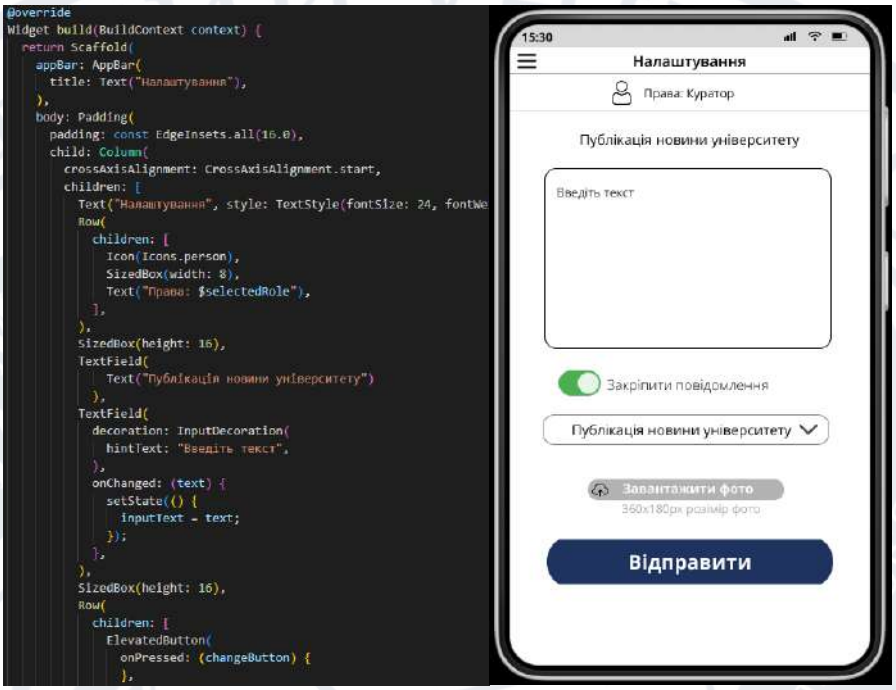


Рисунок 3.19 – Програмна реалізація сторінки Публікація новини університету

Також можна обрати таку функцію як "Функція публікування карти кабінетів" вона дозволяє завантажити, змінити або відредагувати вже існуючу карту. Що дасть можливість завжди мати актуальну карту кабінетів.

Зверху знаходиться назва функції "Публікація карти кабінетів", що інформує користувача на якій сторінці налаштувань він знаходиться.

Нижче знаходиться випадаючий список, де користувач може обрати номер поверху. Це важливо для можливості зміни карти на кожному поверсі корпусу.

Є змога завантажити карту в багатьох форматах для нашої карти кабінетів, яка буде допомагати користувачам знаходити кабінети.

Нижче знаходиться випадаючий список, де користувач може обрати, вид публікації із варіантів: "Публікація новини університету", "Публікація важливої

інформації", "Публікація карти кабінетів", "Публікація розкладу занять". Це важливо для визначення виду публікації і можливості змінити його за потреби.

У кінці сторінки розміщена кнопка "Відправити", яка ініціює відправлення введеного користувачем контенту на сторінку карти кабінетів. Ця кнопка запускає функцію для обробки та збереження змін.

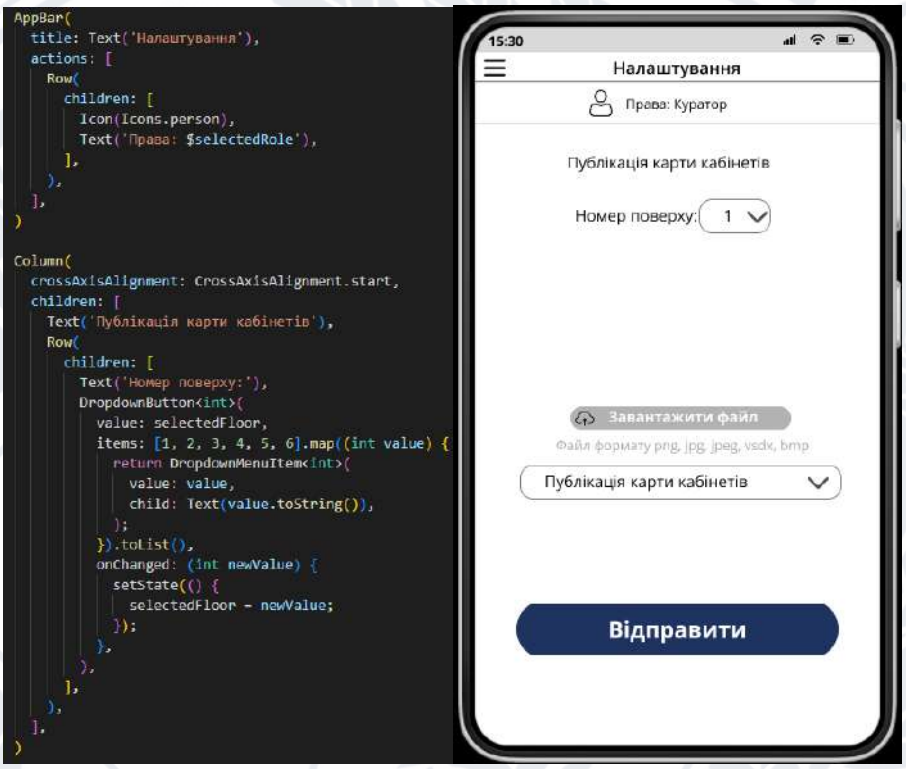


Рисунок 3.20 – Програмна реалізація сторінки Публікація карти кабінетів

Також можна обрати таку функцію як "Функція публікування розкладу занять" вона дозволяє завантажити розклад. Що дасть можливість завжди мати актуальний розклад для студентів.

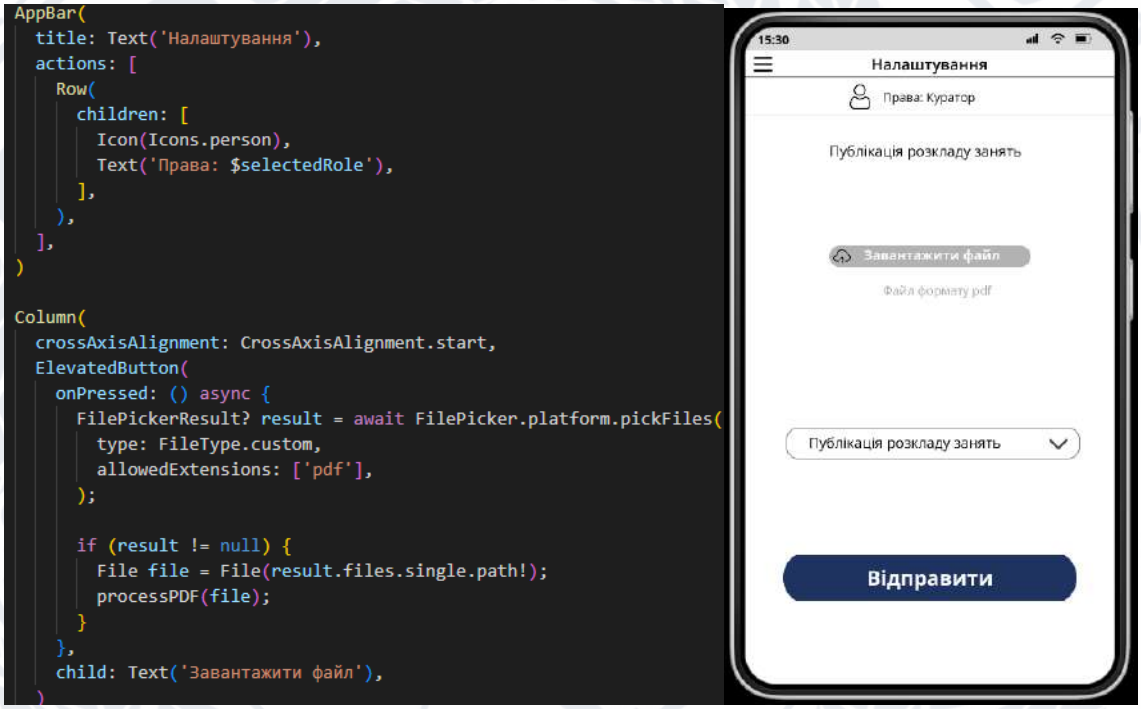
Зверху знаходиться назва функції "Публікація розкладу занять", що інформує користувача на якій сторінці налаштувань він знаходиться.

Нижче знаходиться кнопка завантаження файлу в форматі pdf для додавання його до нашого списку розкладів занять, який буде інформувати користувачів актуальним розкладом занять.

Далі знаходиться випадаючий список, де користувач може обрати, вид публікації із варіантів: "Публікація новини університету", "Публікація важливої

інформації", "Публікація карти кабінетів", "Публікація розкладу занять". Це важливо для визначення виду публікації і можливості змінити його за потреби.

У кінці сторінки розміщена кнопка "Відправити", яка ініціює відправлення введеного користувачем контенту на сторінку карти кабінетів. Ця кнопка запускає функцію для обробки та збереження змін.



```
AppBar(
  title: Text('Налаштування'),
  actions: [
    Row(
      children: [
        Icon(Icons.person),
        Text('Права: $selectedRole'),
      ],
    ),
  ],
)

Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  ElevatedButton(
    onPressed: () async {
      FilePickerResult? result = await FilePicker.platform.pickFiles(
        type: FileType.custom,
        allowedExtensions: ['pdf'],
      );

      if (result != null) {
        File file = File(result.files.single.path!);
        processPDF(file);
      }
    },
    child: Text('Завантажити файл'),
  )
```


Рисунок 3.21 – Програмна реалізація сторінки Публікація розкладу занять

У мобільному додатку також є карта кабінетів, яка виконує важливі функції, що полегшують користувачам навігацію та забезпечують швидкий доступ до інформації. Ось детальний опис цього функціоналу:

Зверху сторінки розташований заголовок "Карта Кабінетів", що вказує на функціонал цієї сторінки.

Користувач може взаємодіяти з картою: змінювати масштаб, шукати конкретний кабінет за номером або іменем, що робить використання карти зручним та ефективним.

Карта кабінетів відображається поетапно: спочатку зверху розташована загальна карта, а нижче можливість вибору конкретного поверху для подальшого розгляду.

Зліва на сторінці присутнє меню, яке містить швидкі посилання на інші частини додатку, забезпечуючи зручну навігацію.

Нижче карти розташовані дві кнопки для перемикання між поверхами. Текст поруч із кнопками інформує користувача про номер поточного поверху. Ця сторінка надає користувачам засоби для зручної навігації.

Використовуємо Scaffold з AppBar для заголовку сторінки та body для розміщення контенту.

```
Scaffold(
  appBar: AppBar(
    title: Text('Карта Кабінетів'),
  ),
```

Рисунок 3.22 – Код реалізації заголовку сторінки

Використовуємо Image або Image.network для завантаження та відображення карти кабінетів.

```
Image.network(img_map);
```

Рисунок 3.23 – Код реалізації відображення карти кабінетів

Використовуємо Drawer для реалізації меню.

Додаємо ListView з пунктами меню та відповідними обробниками подій.

Внизу карти розташовані дві кнопки для перемикання між поверхами. Текст поруч із кнопками інформує користувача про номер поточного поверху.

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    ElevatedButton(
      onPressed: () {
        beforeMap
      },
      child: Text(numberMap),
    ),
    ElevatedButton(
      onPressed: () {
        afterMap
      },
      child: Text(numberMap),
    ),
```

Рисунок 3.24 – Код реалізації кнопок зміни поверху

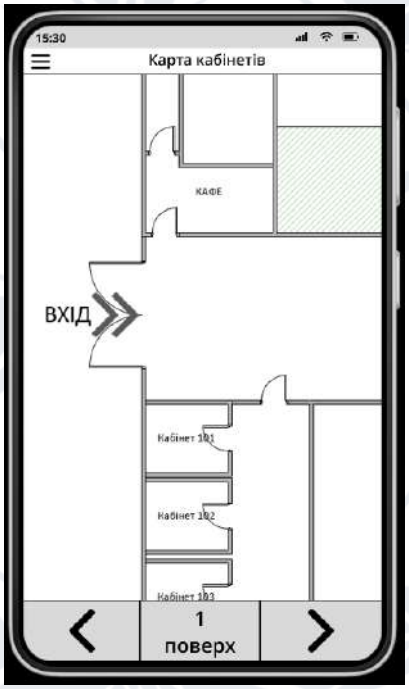


Рисунок 3.25 – Інтерфейс сторінки карти кабінетів

Карта кабінетів виконує важливі функції, полегшує користувачам навігацію в корпусі та забезпечують швидкий доступ до інформації розташування кабінетів. Що є важливим для студентів факультету.

Висновок до розділу 3

У ході виконання розділу 3 було ретельно розглянуто та вирішено багато ключових завдань, спрямованих на розробку мобільної інформаційно-довідкової системи для факультету "Факультет інформаційних і прикладних технологій". Процес створення додатку був докладно структурований та включав в себе наступні етапи.

Велика увага була приділена створенню інтуїтивно зрозумілого та привабливого інтерфейсу. Це включало в себе розробку головної сторінки, сторінку налаштування, сторінок для розкладу та карти кабінетів, які були структуровані та зручні для користувачів.

Було розроблено функціонал для реєстрації та авторизації користувачів, можливість перегляду новин та важливої інформації, завантаження розкладу, перегляд карти кабінетів та інші корисні можливості.

Додаток був підданий ретельному тестуванню на різних пристроях та операційних системах. Були виявлені і виправлені всі виявлені помилки для забезпечення стабільності та надійності.

Важливою частиною проекту була забезпечення безпеки даних та функціоналу. Були вжиті заходи для захисту конфіденційної інформації користувачів та стійкості додатку до потенційних загроз.

В результаті цього комплексного підходу був розроблений мобільний додаток, який відповідає потребам факультету та є потужним інструментом для здобувачів. Процес розробки був важливим етапом в розвитку системи та дав можливість виявити та вирішити ключові завдання проекту.

ВИСНОВКИ

У рамках даної роботи було розглянуто актуальність розробки мобільної інформаційно-довідкової системи для здобувачів факультету. Аналіз існуючих альтернатив інформаційно-довідникових систем в мобільних додатках, визначення потреб користувачів та огляд сучасних тенденцій у розробці додатків підкреслили важливість створення ефективного та функціонального інструмента для студентів та викладачів.

Протягом трьох розділів роботи було вирішено значущі завдання, починаючи від аналізу доцільності розробки і закінчуючи тестуванням та забезпеченням безпеки додатку. Систематичний підхід до процесу розробки дозволив створити продукт, який відповідає вимогам користувачів та враховує сучасні технологічні тенденції.

Крім того, увага до інтуїтивного дизайну та функціональності додатку підкреслила його зручність та привабливість для користувачів. Вибір оптимального технологічного стеку, такого як мова програмування Dart і фреймворк Flutter, сприяв швидкій та ефективній розробці крос-платформеного рішення.

Загалом, результатом цього проекту є мобільний додаток, який відповідає вимогам сучасності та стає важливим кроком у розвитку мобільних інформаційно-довідкових систем для здобувачів факультету. Подальше вдосконалення та розширення функціоналу може сприяти подальшій популяризації та успіху цього інструменту в освітньому середовищі університету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Документація фреймворку Flutter" – URL: https://docs.flutter.dev/ (дата звернення: 18.09.2023)
2. "Flutter in Action" - Eric Windmill. 496 с.
3. "Flutter Complete Reference" - Alberto Miola. 210 с.
4. "Beginning App Development with Flutter" - Rap Payne, Amanda Thornton. 128 с.
5. "Learning Flutter: A Hands-On Guide to App Development" - Paul Deitel, Harvey Deitel. 512 с.
6. "Документація мови програмування Dart" – URL: https://dart.dev/guides (дата звернення: 01.10.2023)
7. "Dart: Up and Running" - Kathy Walrath, Seth Ladd. 162 с.
8. "Learning Dart - Second Edition" - Ivo Balbaert. 446 с.
9. "Top Programming Languages for Developing Mobile Apps in 2023" – URL: https://medium.com/@digitaltm020/top-programming-languages-for-developing-mobile-apps-in-2023-1a50551fb086 (дата звернення: 05.10.2023)
10. "Dart Essentials" - Martin Sikora. 214 с.
11. "Dart: Scalable Application Development" - Michaël Berty. 372 с.
12. "Google Dart: Up and Running" - Cesar Ferrari, Kathy Walrath. 76 с.
13. "Dart: A Simple, Effective, and Powerful Language" - James Ward. 52 с.
14. "Firebase Essentials - Android Edition" - Neil Smyth. 142 с.
15. "Mastering Firebase for Android Development" - Ashok Kumar S. 324 с.
16. "Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase" - Harmeet Singh. 328 с.
17. "Learning Firebase" - Krishnendu Roy, Kaushik Roy. 268 с.
18. "Firebase: Up and Running" - Mike Donald, Sean Grove. 208 с.
19. "Firebase from Scratch" - Greg Sidelnikov. 254 с.
20. "Building Serverless Mobile Applications with Google Cloud Platform and Firebase" - Abhishek Mishra. 372 с.
21. "Flutter vs React Native: A Comparison" – BrowserStack — URL: https://eightify.app/summary/programming-and-artificial-intelligence/flutter-vs-

react-native-a-comparison-of-mobile-app-development-frameworks (дата звернення: 18.10.2023)

22."NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" - Martin Fowler, Pramod J. Sadalage. 192 с.

23."Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux" - Alessandro Del Sole. 400 с.

24."Firebase Essentials - Android Edition" - Neil Smyth. 142 с.

25."Mobile App Development with Ionic 2: Cross-Platform Apps with Ionic 2, Angular 2, and Cordova" - Chris Griffith. 416 с.

26."How to Choose a Good IDE"— URL: https://stackoverflow.com/questions/268528/ (дата звернення: 20.10.2023)

27."Everything You Need to Know About Flutter App Development" – URL: https://www.cometchat.com/blog/flutter-app-development (дата звернення: 23.10.2023)

28."A Complete Guide to Mobile App Development" – URL: https://buildfire.com/understanding-mobile-app-development-lifecycle/ (дата звернення: 25.10.2023)

29. "Документація бази данних Firebase" – URL: https://firebase.google.com/docs

30. "Hooked: How to Build Habit-Forming Products" - Eyal Nir. 327 с.

31."Code Complete: A Practical Handbook of Software Construction" - Steve McConnell. 960 с.

32."Clean Code: A Handbook of Agile Software Craftsmanship" - Robert C. Martin. 464 с.

33."Software Engineering: A Practitioner's Approach" - Roger S. Pressman. 992 с.

34."Tap Into Mobile Application Testing" - Jonathan Kohl. 100 с.

35."Mobile Testing: An ASTQB-BCS Foundation Guide" - Andy Glover, Daniel Knott. 208 с.

36."Top Most Popular Programming Languages for Mobile App" – URL: https://fireart.studio/blog/top-most-popular-programming-languages-for-mobile-app-development/ (дата звернення: 03.11.2023)

37. "Android App Development Languages: Which To Choose" – URL: https://www.codemotion.com/magazine/frontend/mobile-dev/android-app-development-which-language-to-choose/ (дата звернення: 05.11.2023)
38. "What is the best framework for mobile app development in 2023" – URL: https://merge.rocks/blog/what-is-the-best-framework-for-mobile-app-development-in-2023 (дата звернення: 08.11.2023)
39. "Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps" - Theresa Neil. 416 с.
40. "Designing Interfaces: Patterns for Effective Interaction Design" - Jenifer Tidwell. 592 с.
41. "Mobile Usability" - Jakob Nielsen, Raluca Budiu. 240 с.
42. "Handbook of Mobile UX Design: Tap into the Experience of Mobile App Development" - Martina Schell, Ondrej Mirtes, Max Galka. 224 с.
43. "27 Steps to a Successful Mobile App Launch" – Orangesoft – URL: https://orangesoft.co/blog/how-to-launch-an-app (дата звернення: 12.11.2023)
44. "Mobile Application Security: Protecting Mobile Devices and Their Applications" - Himanshu Dwivedi, Chris Clark, David Thiel. 338 с.
45. "iOS Application Security: The Definitive Guide for Hackers and Developers" - David Thiel. 432 с.
46. "Android Hacker's Handbook" - Joshua J. Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, Georg Wicherski. 576 с.
47. "Choosing the Right Mobile App Development Technology" - Todd Moore. 324 с.
48. "Mobile App Development: A Practical Guide" - Pete Houston. 412 с.
49. “Flutter and Dart Cookbook” - автор Річард Роуз. 435 с.
50. "User Story Mapping: Discover the Whole Story, Build the Right Product" - Jeff Patton. 324 с.

ДОДАТОК А

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';

class RegistrationScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text('Реєстрація'),
   ),
   body: Center(
    child: Column(
     children: <Widget>[
      TextField(
       decoration: InputDecoration(labelText: 'Ім\'я Прізвище'),
      ),
      TextField(
       decoration: InputDecoration(labelText: 'Email'),
      ),
      TextField(
       decoration: InputDecoration(labelText: 'Пароль'),
       obscureText: true,
```

```
            ),
            TextField(
              decoration: InputDecoration(labelText: 'Повторіть пароль'),
              obscureText: true,
            ),
            RaisedButton(
               child: Text('Зареєструватись'),
                  onPressed: () async {
                        try {
                          await registerUser(nameController.text,
emailController.text, passwordController.text);
                        }
                        catch (e) {
                            print('Помилка реєстрації: $e');
                        }
                    },
                )
          ],
        ),
      ),
    );
  }
}
```

```
import 'package:firebase_core/firebase_core.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());

}

import 'package:firebase_auth/firebase_auth.dart';

Future<void> registerUser(String email, String password) async {
  try {
    await FirebaseAuth.instance.createUserWithEmailAndPassword(
        name: name,
        email: email,
        password: password,
    );
  } catch (e) {
    print('Помилка реєстрації: $e');
  }
}
```

```
Future<void> signInUser(String email, String password) async {
  try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: email,
      password: password,
    );

  } catch (e) {
    print('Помилка авторизації: $e');
  }
}

AppBar(
  title: Text('Інформаційний довідниковий студента ДонНУ'),
  leading: Builder(
    builder: (BuildContext context) {
      return IconButton(
        icon: Icon(Icons.menu),
        onPressed: () {
          Scaffold.of(context).openDrawer();
        },
      );
```

```
    },
  ),
),

Drawer(
  child: ListView(
    padding: EdgeInsets.zero,
    children: <Widget>[
      UserAccountsDrawerHeader(
        accountName: Text(name),
        accountEmail: Text(email),
      ),
      ListTile(
          leading: Icon(Icons.home),
          title: Text('Головна сторінка'),
          onTap: () {
              Navigator.of(context).pushReplacement(
              MaterialPageRoute(
              builder: (BuildContext context) => HomePage(),
          ),
    );
    Navigator.of(context).pop();
```

```
    },
    )

    ListTile(
      leading: Icon(Icons.map),
      title: Text('Карта кабінетів'),
      onTap: () {
          Navigator.of(context).pushReplacement(
          MaterialPageRoute(
          builder: (BuildContext context) => MapPage(),
        ),
      );
      Navigator.of(context).pop();
      },
    )
    ],
  ),
),

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
appBar: AppBar(
  title: Text('Головна сторінка'),
),
body: SingleChildScrollView(
  child: Column(
    children: <Widget>[
      Image.network(
        'C:\Users\mmysak\img\Saved Pictures\home_img_1.jpg',
        width: double.infinity,
        height: 200.0,
        fit: BoxFit.cover,
      ),

      Padding(
        padding: const EdgeInsets.all(16.0),
        child: Text(
          'Новини Донецького національного університету імені Василя Стуса',
          style: TextStyle(fontSize: 18.0),
        ),
      )

      Container(
```

```
            height: 200.0,
            child: ListView.builder(
              itemCount: 5,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text('Елемент $index'),
                  onTap: () {
                  },
                );
              },
            ),
          ),
        ],
      ),
    ),
  );
 }
}

class SchedulePage extends StatelessWidget {
 final List<Map<String, dynamic>> pdfFiles = [
  {'name': 'М23_д_122_1_курс', 'date': '01.08.2023'},
  {'name': 'Б20_д_113_3_курс', 'date': '01.08.2023'},
```

```
  {'name': 'Б20_д_121_3_курс', 'date': '01.08.2023'},
  {'name': 'Б21_д_121_2_курс', 'date': '01.08.2023'},
  {'name': 'М22_д_121_2_курс', 'date': '01.08.2023'},
  {'name': 'М22_д_122_2_курс', 'date': '01.08.2023'},
  {'name': 'Б22_д_113_2_курс', 'date': '01.08.2023'},
  {'name': 'М23_д_123_1_курс', 'date': '01.08.2023'},
  {'name': 'Б23_д_122_1_курс', 'date': '01.08.2023'},
];

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Розклад занять'),
    ),
    body: ListView.builder(
      itemCount: pdfFiles.length,
      itemBuilder: (context, index) {
        final file = pdfFiles[index];
        final fileName = file['name'];
        final fileDate = file['date'];

        return ListTile(
```

```
            title: Text(fileName),
            subtitle: Text('Дата створення: $fileDate'),
            onTap: () {
              Navigator.of(context).push(
                MaterialPageRoute(
                  builder: (context) => PDFViewerAsset(
                    assetName: 'assets/$fileName.pdf',
                    appBar: AppBar(
                      title: Text(fileName),
                    ),
                  ),
                ),
              );
            },
          );
        },
      ),
    );
  }
}

class SettingsPage extends StatefulWidget {
  @override
  _SettingsPageState createState() => _SettingsPageState();
```

```
}

class _SettingsPageState extends State<SettingsPage> {
  bool isCurator = false;
  bool isModerator = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Налаштування'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Text(
              'Роль користувача:',
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
              ),
```

```
),
Row(
  children: <Widget>[
    Text('Староста'),
    Checkbox(
      value: isCurator,
      onChanged: (bool newValue) {
        setState(() {
          isCurator = newValue;
        });
      },
    ),
  ],
),
Row(
  children: <Widget>[
    Text('Куратор'),
    Checkbox(
      value: isModerator,
      onChanged: (bool newValue) {
        setState(() {
          isModerator = newValue;
        });
```

```
      },
    ),
  ],
),
SizedBox(height: 16),
Text(
 'Управління інформацією:',
 style: TextStyle(
  fontSize: 18,
  fontWeight: FontWeight.bold,
 ),
),
ElevatedButton(
 onPressed: () {
  // Код для публікації інформації
 },
 child: Text('Публікувати інформацію'),
),
ElevatedButton(
 onPressed: () {          },
 child: Text('Редагувати інформацію'),
),
ElevatedButton(
```

```
            onPressed: () {
            },
            child: Text('Закріпити інформацію'),
          ),
        ],
      ),
    ),
  );
 }
}
```

ДОДАТОК Б

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';

Future<void> registerUser(String name, String email, String password) async
{
  try {
    if (name.isEmpty || email.isEmpty || password.isEmpty) {
      // Перевірка на заповненість всіх полів
      print('Please fill in all fields.');
      return;
    }

    if (password.length < 6) {
      print('Password must be at least 6 characters long.');
      return;
    }

    UserCredential userCredential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(email: email, password: password);

    String userId = userCredential.user!.uid;
```

```
    await FirebaseDatabase.instance.reference().child('users').child(userId).set({
      'name': name,
      'email': email,
    });
  } on FirebaseAuthException catch (e) {
    if (e.code == 'weak-password') {
      print('The password provided is too weak.');
    } else if (e.code == 'email-already-in-use') {
      print('The account already exists for that email.');
    }
  } catch (e) {
    print(e);
  }
}
class RegistrationScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Реєстрація'),
      ),
      body: Center(
```

```
child: Column(
 children: <Widget>[
  TextField(
   decoration: InputDecoration(labelText: 'Ім\'я Прізвище'),
  ),
  TextField(
   decoration: InputDecoration(labelText: 'Email'),
  ),
  TextField(
   decoration: InputDecoration(labelText: 'Пароль'),
   obscureText: true,
  ),
  TextField(
   decoration: InputDecoration(labelText: 'Повторіть пароль'),
   obscureText: true,
  ),
  RaisedButton(
     child: Text('Зареєструватись'),
        onPressed: () async {
           try {
              await registerUser(nameController.text,
emailController.text, passwordController.text);
           }
```

```
                catch (e) {
                    print('Помилка реєстрації: $e');
                }
            },
        )
    ]
  )
 )
)
}
```

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)