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

Дипломна робота
бакалавра
з теми: **«ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ “ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН СТУДЕНТА”»**

Виконав: студент 4 курсу, групи KN1-B18
спеціальності 122 Комп’ютерні науки
Чернявський Артем Антонович

Керівник:
Пилипюк Т.М.,
кандидат фізико-математичних наук,
доцент, доцент кафедри комп’ютерних наук

Рецензент:
Газдюк К.П.,
доктор філософії за
спеціальністю 121 Інженерія програмного
забезпечення, асистент кафедри
програмного забезпечення комп’ютерних
систем Чернівецького національного
університету імені Юрія Федьковича

Кам’янець-Подільський – 2022

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі існує тенденція на спрощення доступу до інформації за допомогою сучасних комп'ютерних технологій. За останні роки процес отримання доступу до даних став набагато простішим: інформація, доступ до якої раніше міг бути ускладненим, тепер вільно розповсюджується.

Кожна людина має певну кількість документів, що мають велику цінність, і які повинні ретельно зберігатися. Зазвичай такі документи існують у паперовому вигляді, що не є дуже надійним, а їх використання – не сильно зручним. Тому сьогодні існує очевидний тренд на перенесення особистих документів у цифровий вигляд. Це не тільки збільшує надійність, а й покращує ефективність їх використання, особливо у випадках, коли цей документ використовують часто. Прикладом тренду на оцифрування документів є впроваджений Міністерством цифрової трансформації України електронний сервіс «Дія», що дозволяє зберігати та використовувати громадянину України особисті документи, такі як паспорти та посвідчення, за допомогою мобільного додатку. Використання документів за допомогою мобільного додатку є чи не найзручнішим способом їх застосування, тому що через розповсюдженість смартфонів у сучасному суспільстві кожна людина може мати всі необхідні документи «у кишені», тобто майже завжди з собою. У випадку паперових документів таке було б фізично неможливо.

Окрім документів, що прийняті на загальнодержавному рівні, існують документи більш локального призначення, наприклад для компанії або навчального закладу. Якщо розглянути, наприклад, документи закладу вищої освіти, серед них є такі, ціллю яких є донесення певної інформації до студентів. Частина з них публікується у вільному доступі, такі як накази, розпорядження або оголошення. Деякі з них так і залишаються лише у паперовому вигляді і доступ до них обмежений.

На фізико-математичному факультеті Кам’янець-Подільського національного університету імені Івана Огієнка прикладом такого документу, до якого доступ студента є ускладненим, є індивідуальний навчальний план студента. Тим не менш, цей документ є корисним для кожного студента, і спрощення доступу до певної інформації у ньому покращило б умови навчання студенів на факультеті.

Мета дипломної роботи – створення повноцінного програмного продукту «Індивідуальний навчальний план студента», що буде готовим для використання у межі фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка та дозволить студенту вільно переглядати інформацію з цього документу за допомогою мобільного додатку.

Для досягнення мети потрібно було виконати завдання:

- проаназізувати види мобільних додатків;
- розглянути використання фреймворку Flutter для мобільної розробки, структуру проєктів на Flutter на особливості мови програмування Dart;
- охарактеризувати системи Material Design, сервіс Google Sheets, інструменти Google API
- здійснити проєктування розробку мобільного додатку;
- організувати взаємодію користувачів з фінальним продуктом.

Об’єктом дослідження є програмний продукт для смартфонів «Індивідуальний навчальний план студента».

Предметом дослідження є: фреймворк Flutter, мова програмування Dart, сервіс Google Sheets, інструменти Google API для створення мобільного додатку.

Структура роботи. Дипломна робота бакалавра складається зі вступу, двох розділів, висновків, списку використаних джерел та додатку.

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

1.1. Мобільна розробка. Види мобільних додатків. Крос-платформна розробка

Розробка мобільних додатків — це набір процесів та процедур, що беруть участь у написанні програмного забезпечення для невеликих бездротових обчислювальних пристроїв, таких як смартфони та інші кишенькові пристрої. Це створення програмного забезпечення, призначеного для запуску на мобільних пристроях та оптимізовано, щоб скористатися унікальними функціями та обладнанням цих продуктів.

Розробка мобільних додатків стрімко розвивається. Від роздрібної торгівлі, телекомунікацій та електронної комерції до страхування, охорони здоров'я та уряду — організації в різних галузях повинні відповідати очікуванням користувачів щодо зручних способів здійснення операцій та доступу до інформації в реальному часі. Сьогодні мобільні пристрої — і мобільні додатки, які розкривають їх цінність — є найпопулярнішим способом підключення людей і компаній до Інтернету. Щоб залишатися актуальними, чуйними та успішними, організації повинні розробляти мобільні додатки, які потребують їхні клієнти, партнери та співробітники [7].

Типи мобільних додатків, які створюють розробники, включають нативні додатки, гібридні програми та додатки HTML5.

У перші роки розвитку мобільних додатків єдиним способом забезпечити оптимальну роботу додатка на будь-якому пристрої було розробка нативного додатка. Це означало, що новий код потрібно було писати спеціально для конкретного процесора кожного пристрою. Сьогодні більшість розроблених мобільних додатків не залежать від пристроїв.

Нативні програми. Це додатки, створені для певної платформи (iOS або Android), використовуючи інструменти розробки та мови, що підтримуються цими

операційними системами. iOS використовує Xcode та Objective-C, тоді як Android використовує Eclipse та Java. Розробники часто віддають перевагу нативним додаткам через їх здатність використовувати повний потенціал пристрою. Коли розумні домашні пристрої стають все більш поширеними, розробники створюють унікальні програми, які інтегрують такі речі, як Інтернет-датчики та розумні екрани для персоналізованого досвіду. Звичайно, розробка для кожної платформи-це дорогий і трудомісткий процес, який підходить не для всіх підприємств.

HTML5 додатки. Додатки, що базуються на майже-універсальних стандартах веб-технологій — HTML5, JavaScript та CSS. Програми, розроблені в цих фремйворках, сумісні з багатьма платформами і вимагають лише мінімальних змін, щоб забезпечити повну функціональність на кожній операційній системі. Програми HTML5 все ще можуть надсилати сповіщення на робочому столі та тригерні взаємодії за допомогою електронної пошти та інших напрямків. Не потрібно принижувати зручність веб-додатків, але потрібно мати на увазі, що споживачі швидше використовують мобільний додаток. Дослідження Oracle виявило, що користувачі витрачають 90% свого мобільного часу в додатках та тільки 10% у веб-браузерах.

Гібридні програми. Це веб-програми, які працюють як нативні програми. Вони розроблені з використанням таких технологій, як HTML, JavaScript і CSS. Розробка гібридних додатків економніша, ніж розробка нативних програм, і їх можна створювати швидше, але вони не такі багатофункціональні, як нативні програми.

Мобільний додаток на основі фреймворків та бібліотек. Розробник може використовувати код багаторазового використання, написаний кимось іншим, щоб прискорити розробку мобільного додатка.

Плюси і мінуси кросплатформної мобільної розробки:

Переваги:

- Створення окремих нативних додатків для кожної платформи коштує дорого, тоді як гібридна програма використовує єдиний загальний код, який допомагає вам утримуватися в межах свого бюджету.
- Витрати зменшуються, оскільки для розробки та підтримки програми потрібна лише одна команда програмістів. Більше того, достатньо базових знань стандартних мов — інструменти розробки зроблять решту роботи.
- Крос-платформні програми мають оригінальний зовнішній вигляд та відчуття, що чудово підходить для користувача.
- Гібридна розробка — це, безумовно, шлях для компаній, які хочуть залучити користувачів різних мобільних пристроїв і швидше випустити продукт на ринок за меншу ціну.

Однак є деякі проблеми, з якими розробник можете зіткнутися:

- Більш складний код гібридних рішень поєднує в собі нативні та не-нативні компоненти, що може вплинути на швидкість роботи.
- Крос-платформні додатки не можуть підтримувати всі функції та функції мобільних пристроїв, які є лише нативними, такі як складна графіка та анімація або 3D-ефекти. Це призводить до обмеженої функціональності та погіршення дизайну програми.
- Коли Google і Apple додають нові функції на платформи Android і iOS, рідні рішення можуть негайно почати їх використовувати. Але гібридним додаткам доводиться чекати, поки ці оновлення будуть адаптовані до обраного кросплатформного фреймворку. Таким чином, завжди буде затримка оновлення.

Тим не менш, найзручнішим для розробника способом створення мобільного додатку, що дозволить створити додаток для використання багатьма користувачами, та не буде витрачати багато ресурсів розробника, є створення мобільного крос-платформного додатку за допомогою фреймворку.

Кросплатформна розробка підходить для рішень, які: не вимагають складного дизайну; не потрібно постійно обробляти вхідні дані онлайн; не потрібен доступ до всіх функцій пристрою.

Існують такі популярні фреймворки для створення крос-платформних додатків [13]:

React Native — створений і підтримуваний Facebook, є доступнми крос-платформним фреймворком для розробки додатків, який швидко став улюбленою опцією програмістів. React Native дозволяє розробляти мобільні додатки для Android та iOS. Найбільш взірцевими прикладами програм на фреймфорку React Native є програми від відомих компаній, таких як Tesla, Airbnb, Skype або Amazon Prime. Головна привабливість React Native полягає в тому, що він забезпечує швидшу розробку та впровадження. Багаторазові елементи, взаємодія із сторонніми розширеннями, а також створення графічного інтерфейсу на основі компонентів для інтерфейсних додатків є ще одними важливими характеристиками React Native. Особливості React Native — виняткова продуктивність, компоненти, які можна використовувати повторно, сумісність зі сторонніми розширеннями.

Flutter — це відкритий і безкоштовний фреймворк від Google, який дозволяє створювати нативні програми для Android та iOS за допомогою простої кодової бази. Це інноваційний пакет SDK для розробки крос-платформних додатків, що відрізняє його тим, що він використовує новий спосіб створення додатків, схожих на нативні. Flutter — це всеосяжний і точний фреймворк, який містить віджети, механізм візуалізації, налагодження та інтеграцію API, а також ресурси, які допомагають розробникам створювати та розгортати красиві мобільні додатки. Flutter використовувався рядом відомих організацій, включаючи Google і Abbey Road Studios.

Flutter був випущений Google у травні 2017 року. Він використовується як інструмент розробки мобільних додатків для платформ Android, iOS, Mac, Microsoft Windows, Linux та Google Fuchsia. Ця структура підтримує мови кодування C, C++ і Dart. Dart — це об’єктно-орієнтована мова програмування з відкритим вихідним кодом із таким же стилем синтаксису, що й мова програмування C.

VueJS — це фреймворк на основі JavaScript, створений для створення інтерфейсів користувача та односторінкових програм. Ви можете просто розпочати створення програми та поступово додавати необхідні інструменти та функції під час процесу розробки. VueJS надає спосіб створювати частини, які містять стан або дані у вашому JavaScript. Їх можна підключити до шаблонів у HTML. Одні й ті самі введені дані завжди вироблятимуть однаковий візуальний вихід інтерфейсу користувача. У VueJS компонент автоматично адаптується під різні платформи, що чудово підходить для програмістів, які займаються розробкою мобільних додатків. VueJS це одни із найновіших фреймворків і швидко розвивається, через це те, що ви використовували сьогодні, може не працювати завтра. Це дуже ускладнює оновлення програм. Це може дратувати, але не буде серйозною проблемою. Більшість спільноти VueJS походить з Китаю. З огляду на це, звичайний розробник, який спілкується переважно англійською або своєю рідною мовою, може не знайти допомоги, якщо вона буде йому потрібна.

Xamarin — це крос-платформний фреймворк для розробки додатків для Android та iOS. Оскільки він використовує мову програмування C#, додаткам потрібно менше рядків коду. В результаті процес написання коду відбувається швидше. Крім того, це дозволяє нам швидко використовувати скрипти між системами, такими як Windows і macOS. Xamarin була придбана Microsoft. Оскільки сьогодні розробка програми відбувається набагато швидше, можна припустити, що при швидшій розробці ми жертвуємо якістю та конструкцією. Однак програми на основі Xamarin забезпечують бездоганну вбудовану функціональність у сенсі якості та ефективності. В результаті його зв’язок з Microsoft Visual Studio є плюсом з точки зору управління розробкою програм і продуктивності.

Існує ще безліч фреймворків для крос-платформної розробки, але мій вибір пав на фреймворк Flutter.

Flutter — це безкоштовний інтерфейс користувача з відкритим кодом від Google для створення нативних мобільних додатків. Випущений у 2017 році, Flutter

дозволяє розробникам створювати мобільні додатки з єдиною кодовою базою та мовою програмування. Ця можливість робить створення додатків для iOS і Android простіше та швидше.

1.2. Актуальність Flutter. Використання фреймворку Flutter для мобільної розробки

Існує ще безліч фреймворків для крос-платформної розробки, але мій вибір пав на фреймворк Flutter.

Flutter — це безкоштовний фреймворк та інтерфейс користувача з відкритим кодом від Google для створення нативних мобільних додатків. Випущений у 2017 році, Flutter дозволяє розробникам створювати мобільні додатки з єдиною кодовою базою та мовою програмування. Ця можливість робить створення додатків для iOS і Android простіше та швидше. Це актуальний та сучасний комплект розробки кросплатформних застосунків який, хоч з'явився відносно нещодавно, швидко здобув популярності та став одним з найефективніших способів розробки мобільних застосунків. Flutter працює на основі іншої розробки Google — мові програмування Dart [3].

Розробка додатків за допомогою інструментів Flutter є не тільки актуальною, а й перспективною. Багато розробників вважають, що темпи, з якими Flutter розвивається як на ринку, так і у плані функціоналу, зроблять його майбутнім кросплатформної розробки мобільних додатків (рис. 1.1). Минуло лише кілька років відтоді, як Google запустив Flutter, але швидкість інновацій, з якою оновлюється платформа, вже на кілька кроків випередила деякі з найкращих кросплатформних фреймворків, які існують у сфері розробки мобільних додатків [7].

Основні особливості фрейворку:

Flutter — це єдиний фреймворк для розробки кодової бази. Програму, яку ви створюєте на Flutter, може бути опублікована для використання на Android, iOS, комп'ютері та в Web'і. Для бізнесу, який хоче вийти в цифровий простір з мінімальними зусиллями та часом, розробка міжплатформних додатків Flutter є чудовою відправною точкою.

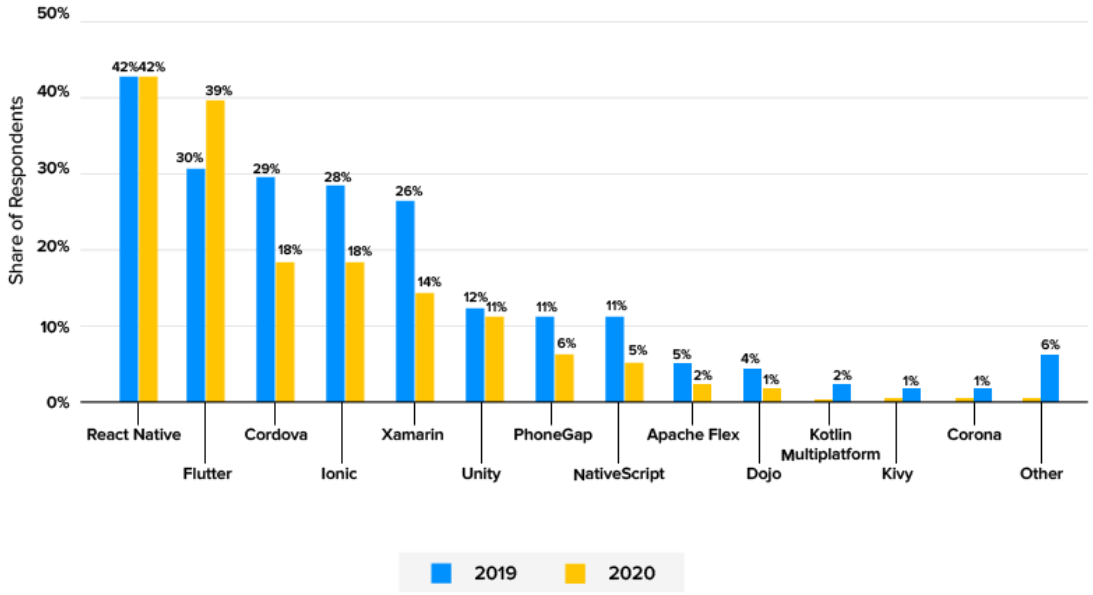

Рис. 1.1. Темпи розвитку фреймворків

Джерело: Flutter Is the Future of Cross-platform App Development. — URL: https://appinventiv.com/blog/flutter-is-cross-platform-development-future/

Повністю настроювані віджети. Однією з найкращих переваг Flutter є той факт, що він підтримує віджети, орієнтовані на платформу. Два набори віджетів Flutter – Cupertino та Material Design – дозволяють компаніям, які розробляють мобільні додатки на Flutter, розробляти користувацький інтерфейс/UX для платформи, надаючи користувачам інтерфейс, що найбільше схожий до нативного додатку на цій платформі.

Швидша розробка додатків. Ще одна річ у фреймворку — це його здатність прискорювати процес розробки та тестування. Той факт, що це єдиний фреймворк кодової бази, разом з фукцією hot reload, полегшує розробникам створення програми під час тестування її в режимі реального часу. Це, у свою чергу, також відіграє вирішальну роль у зниженні вартості розробки мобільних додатків.

Функція Hot Reload — функція «гарячого перезавантаження» Flutter допомагає швидко та легко експериментувати, створювати інтерфейси

користувача, додавати функції та виправляти помилки. Гаряче перезавантаження працює шляхом введення оновлених файлів вихідного коду у запущену віртуальну машину Dart (VM). Після того, як віртуальна машина оновлює класи новими версіями полів і функцій, фреймворк Flutter автоматично перебудовує дерево віджетів, дозволяючи швидко переглядати наслідки ваших змін. На практиці це означає, що при внесенні змін ми бачимо їх на проєкті майже одразу, що економить нам купу часу, який ми могли б витратити на постійну перекомпіляцію додатку.

Для Flutter існує велика кількість пакетів з відкритим кодом — розробники з усього світу постійно додають свою роботу до бібліотеки пакетів, роблячи фреймворк більш функціональним. [11]

Отже, я обрав фреймоврк Flutter для розробки свого мобільного додатку через його актуальність, перспективність та ефективність. Він дозволить мені швидко та зручно створити свій мобільний додаток.

1.3. Структура проєктів на Flutter на особливості мови програмування Dart

Flutter працює як багатошарова, розширювана система. Вона функціонує як послідовність незалежних бібліотек, причому кожна з них залежить від базового шару. Кожна частина рівня фреймворку є змінною, і жоден шар не має привілейованого доступу до нижнього шару. Як правило, розробники працюють з Flutter через фреймворк Flutter, написаний мовою Dart. Він включає в себе макет, багатий набір платформ, а також основні бібліотеки, що складаються з декількох шарів — шар фундаментальних блоків та сервісів, шар рендерингу (показу) та шар віджетів [14].

Основний принцип Flutter звучить як «все — це віджет». Віджети складаються у ієрархію та взаємодіють між собою. Віджетом є як, наприклад, і кнопка, так і текст на ній, так і її стиль, та її розміщення [2, с. 14].

У Flutter існують дві категорії віджетів: stateless та stateful. Їх можна показати як «віджет без стану» та «віджет зі станом» відповідно. Головна різниця між цими

типами віджетів це те, що stateless не змінює свій стан, а stateful постійно перебудовується. Процес побудови, перебудови і оновлення віджету називається циклом життя віджету. [3]

Структура проєкту на Flutter — це деревовидна ієрархічна система віджетів, що накладаються один на одного. Основним віджетом сторінки буде клас певного вікна програми. Над ним — основний віджет графіки, такий як Scaffold. Розробник має змогу робити будь які розгалуження віджетів, як наприклад розділення вікна програми на верхню панель, тіло, та нижню панель, у кожній з яких будуть свої віджети. Більшість віджетів мають власні параметри, які можна змінювати, серед яких — параметри «діти», у які можна розмістити інші віджети. [5]

Мова Dart була розроблена як крос-платформна мова програмування. Вона була оптимізована для роботи на різних типах систем з однаковою ефективністю. Серед особливостей мови Dart є такі як, наприклад, sound null safety — що можна перекласти як «надійна нульова безпека». Вона дозволяє ефективно боротися з критичними помилками, які виникають через використання null — тобто пустих, неіснуючих об’єктів у коді. Використання ізоляцій — особливого способу роботи з ядрами процесора, що збільшує ефективність у багато поточних завданнях. [9] Асинхронне програмування — це концепт у програмування, при якому реалізується можливість викликати функції, що спрацюють через деякий час, не зупиняючи роботу інших частин додатку. Ефективно для роботи з даними, що ми отримуємо з мережі інтернет. Синтаксис мови є С-подібним, тому знайомим з такими мовами буде не важко швидко розібратися у Dart. [2]

1.4. Системи Material Design, сервіс Google Sheets, інструменти Google API

Для створення зовнішнього виду у фреймворку Flutter використовується поняття дизайн-систем. Дизайн система — це візуальна мова нашого продукту, деякий спільний стиль, до якого будуть підрівняні елементи дизайну програми, її анімації та сам стиль, навіть тип інтерактивності користувача з програмою. [12]

Flutter дозволяє користувачам створювати власні дизайн-системи або використовувати інші користувацькі. Тим не менш, в першу чергу Google пропонує обрати основну дизайн систему додатку з двох основних — Cupertino та Material Design , з розрахунку на те, яка з платформ є в пріоритеті, IOS чи Android відповідно.

Для свого проєкту я обрав Material Design. Material — це система дизайну, створена Google, щоб допомогти командам створювати високоякісний цифровий інтерфейс для Android, iOS, Flutter та WEB. Material є популярним способом проєктування Android додатків, його дотримуються додатки самого Google.

У Flutter ми можемо створювати додаток з Material, завантаживши пакет material.dart на писавши код на основі класу Material Design.

Електронні таблиці як і раніше є досить популярним інструментом для роботи з даними, а серед різних процесорів електронних таблиць найбільш популярними є Таблиці Google. По-перше, це безкоштовний інструмент, а по-друге, функціонал Google Sheets досить широкий, і вони надають вам можливість доступу до даних онлайн.

Google дає широкий спектр інструментів для того, щоб розробники могли зв’язувати свої програми з різноманітними сервісами Google. Програмний інтерфейс Google Sheets API дозволяє використовувати дані з таблиць Google у власних проєктах, в тому числі у створених на Flutter додатках. У свому практичному завданні я буду використовувати Google API для того, щоб зв’язати свій Flutter проєкт з таблицями Google. [10]

РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА

2.1. Постановка задачі

Темою моєї дипломної роботи є «Проєктування мобільного додатку “Індивідуальний навчальний план студента”». Відповідно до Положення про Індивідуальний навчальний план здобувача вищої освіти Кам’янець-Подільського національного університету імені Івана Огієнка, Індивідуальний навчальний план здобувача вищої освіти — це документ, що містить інформацію про перелік, обсяг, та результати опанування/засвоєння освітніх компонентів відповідної освітньої (освітньо-професійної/освітньо-наукової) програми, зокрема обраних здобувачем вищої освіти навчальних дисциплін [1].

Отже, додаток має функціонувати як зручний та швидкий спосіб студенту отримати доступ до інформації про навчальні дисципліни, які студент опановує упродовж навчального року, навчальне навантаження у годинах/кредитах ЄКТС. Для цього студент повинен мати змогу подивитись інформацію індивідуально для себе, отже обрати себе за власними даними (за ім’ям та прізвищем). Студенти мають бути розділені по групам, у яких вони навчаються. До студента має бути прив'язаний список дисциплін, обов'язкових та вибіркових. Таким чином студент зможе подивитись обсяг годин (кредитів ЄКТС), що входять до дисципліни. Вони мають бути розділені на обсяг аудиторних годин, до яких входять кількість годин, що призначені для лекційних занять, практичних занять, семінарських занять та лабораторних занять. Окремо від аудиторних годин мають бути вказані години, призначені на самостійну роботу. Має бути вказаний тип семестрового контролю (залік/екзамен). Потрібно вказати прикріплений до дисципліни викладач або прикріплені викладачі. Також замість дисципліни може бути вказаний проєкт, такий як курсова робота. В такому випадку замість викладача має бути вказаний науковий керівник роботи. Окремо має відзначатись практика: там повинна бути вказана кількість кредитів ЄКТС, місце проходження на дати проходження.

На фізико-математичному факультеті Кам’янець-Подільського національного університету імені Івана Огієнка широко використовуються сервіси Google для збільшення ефективності навчального процесу. Платформа Moodle, що використовується університетом, потребує від студентів та викладачів наявності Google аккаунту, прив’язаного до домену університету, часто використовується сервіс Google Meet для відеодзвінків, нещодавно була запроваджена інтеграція сервісу Google Календар, тощо. Тому я вирішив використовувати у якості бази даних сервіс Google Sheets (Google Таблиці), що дозволить вільно та зручно редагувати дані керівництвом навчального закладу. Це відкидає необхідність використовувати сторонні ресурси. Також використання сервісу Google Sheets у таких цілях є для мене вільним та безкоштовним.

Для коректного використання даних мобільним додатком дані у таблиці потрібно вносити за певними правилами, що вказані далі у роботі (розд. 2.4).

2.2. Основа проєктування мобільного додатку. Інструменти розробки. Підключення Google Sheets API до проєкту

Хоч я і розроблюю крос-платформний додаток, що призначений для роботи на платформах Android та IOS, основний фокус роботи у мене буде на платформу Android. Через політику компанії Apple щодо програмного забезпечення, виникають труднощі зі збіркою додатку та його розповсюдження. Для цього Apple потребує від розробника створювати продукт виключно на операційній системі MacOS (що постачається з комп’ютерною технікою Apple), та мати аккаунт розробника Apple, що є платним і коштує біля 100$ на рік. Тим не менш, при наявності усього необхідного, наш додаток можна дуже швидко зібрати як додаток для операційної системи IOS, з мінімальними змінами в коді.

В свою чергу, Android додаток може бути встановлений на будь який Android телефон, що відповідає системним вимогам додатку, за допомогою файлу з

розширенням apk. Отже, через зазачені вище причини, я вирішив сфокусуватись на операційній системі Android як основній.

Проєкт був створений у середовищі розробки Android Studio. Для початку роботи з Dart/Flutter у цьому SDK, необхідно встановити на комп'ютер спочатку набір інструментів розробки Dart SDK, щоб мати можливість писати програми на мові програмування Dart, після цього встановити Flutter SDK, щоб перейти до роботи з безпосередньо фреймворком. Далі потрібно встановити з бібліотеки доповнень Android Studio плагіни Dart та Flutter. З цього моменту середовище розробки готово до створення проєктів на Flutter.

Android Studio має потужну інтеграцію з Dart/Flutter, у середовищі працює функція "Hot Reload", та підказки що відсилають на офіційну документацію Flutter. Перевіряти внесені зміни на самому екрані телефону можна за допомогою емуляторів телефонів Android, які можна створити у меню середовища розробки.

Усі знімки екрану, що використовуються як ілюстрації у цій дипломній роботі, були зроблені на емуляторі телефону Google Pixel 4 (Операцій на система — Android 11), що був створений за допомогою менеджеру девайсів середовища Android Studio.

Проєкт був створений з використанням фреймвойку Flutter версії 2.10.5, на мові програмування Dart версії 2.16.2.

Отже, перший крок для створення нашого додатку — підготовка до під'єднання Google Sheets API до проєкту. Для приєднання таблиць Google Sheets до проєкту Flutter можна використати користувацький пакет з назвою «gsheets». Для пошуку користувацьких пакетів необхідно використовувати офіційний репозиторій за посиланням https://pub.dev. У моїй програмі був використаний пакет версії 0.4.2.

В першу чергу нам необхідно створити таблиці. Переходимо до сервісу Google Sheets за посиланням https://docs.google.com/spreadsheets/. Для роботи з сервісом необхідно мати аккаунт Google. Після входу у меню «Нова електронна

таблиця» потрібно натиснути на шаблон «пустий». Таким чином ми створюємо та відкриваємо саму таблицю. Нагорі веб-сторінки ми можемо ввести її назву.

Наступний крок — це використання Google Sheets API для під'єднання документу до проєкту мобільного додатку. Щоб скористуватись прикладним програмним інтерфейсом для сервісу Google Таблиці потрібно використати інструменти які надає Google у рамках свого сервісу Google Cloud. Через меню керування інструментами Google Cloud ми і зможемо користуватись безкоштовними API для інших Google сервісів. Отже, для початку роботи нам необхідно перейти за веб адресою https://console.cloud.google.com/. Це сервіс з користувацьким інтерфейсом, який можна використовувати для керування власними Google Cloud проєктами. Після переходу на сайт ми з'являємось на головній сторінці сервісу. Нагорі сторінки ми обираємо меню з проєктами, після чого натискаємо на кнопку «New Project». Переходимо у меню створення нового проєкту. Вводимо назву проєкту, у моєму випадку «FlutterSheetsApi», натискаємо «Create» (створити). Через невеликий проміжок часу проєкт створиться.

У новоствореному проєкті Google Cloud Console переходимо до стрічки пошуку, та вводимо у ній «Google Sheets API». Переходимо до нього, та вмикаємо для поточного проєкту — натискаємо на кнопку Enable. Коли API буде підключено, знаходимо у боковому меню зліва пункт «Credentials» (Реквізити для входу). Натискаємо на кнопку «Create Credentials», у контекстному меню, що з'явилося, обираємо пункт «Create service account» (Створити аккаунт сервісу). Тут нам потрібно створити спеціальний акаунт Google для API, щоб той зміг отримати доступ до всієї потрібної інформації. Вводимо його назву, натискаємо кнопки Create and Continue та Done, щоб завершити створення. На сторінці Credentials, на яку ми знову потрапили, шукаємо новостворений аккаунт сервісу, та натискаємо на олівець біля його назви щоб перейти до його налаштування. Переходимо до меню «Keys» (ключі). Натискаємо на Add Key (Додати/створити ключ), у контекстному меню вибираємо пункт Create new key (Створити новий ключ). Обираємо пункт меню «JSON», щоб створити реквізити у відповідному форматі, та натискаємо на

кнопку «Create». Після чого вам на комп'ютер завантажиться файл з розширенням .json. Його можна відкрити більшістю текстових редакторів.

Останній підготовчий крок — це надання доступу новоствореному акаунту сервісу до файлів Google Sheets. Зі сторінки реквізитів входу Credentials на сторінці проєкту Google Cloud Console копіюємо адресу електронної пошти, під’єднаної до акаунту сервісу. Повертаємось до сторінки документу Google Sheets. Натискаємо на пункти головного меню Файл — Поділитися — Надати доступ іншим. Вводимо скопійовану електронну адресу, обираємо зі списку меню пункт «Може редагувати», на натискаємо на кнопку «Надіслати». Ми успішно надали доступ Google Sheets API до нашого файлу Google. Покрокову інструкцію по підключенню показано на рис. 2.1.

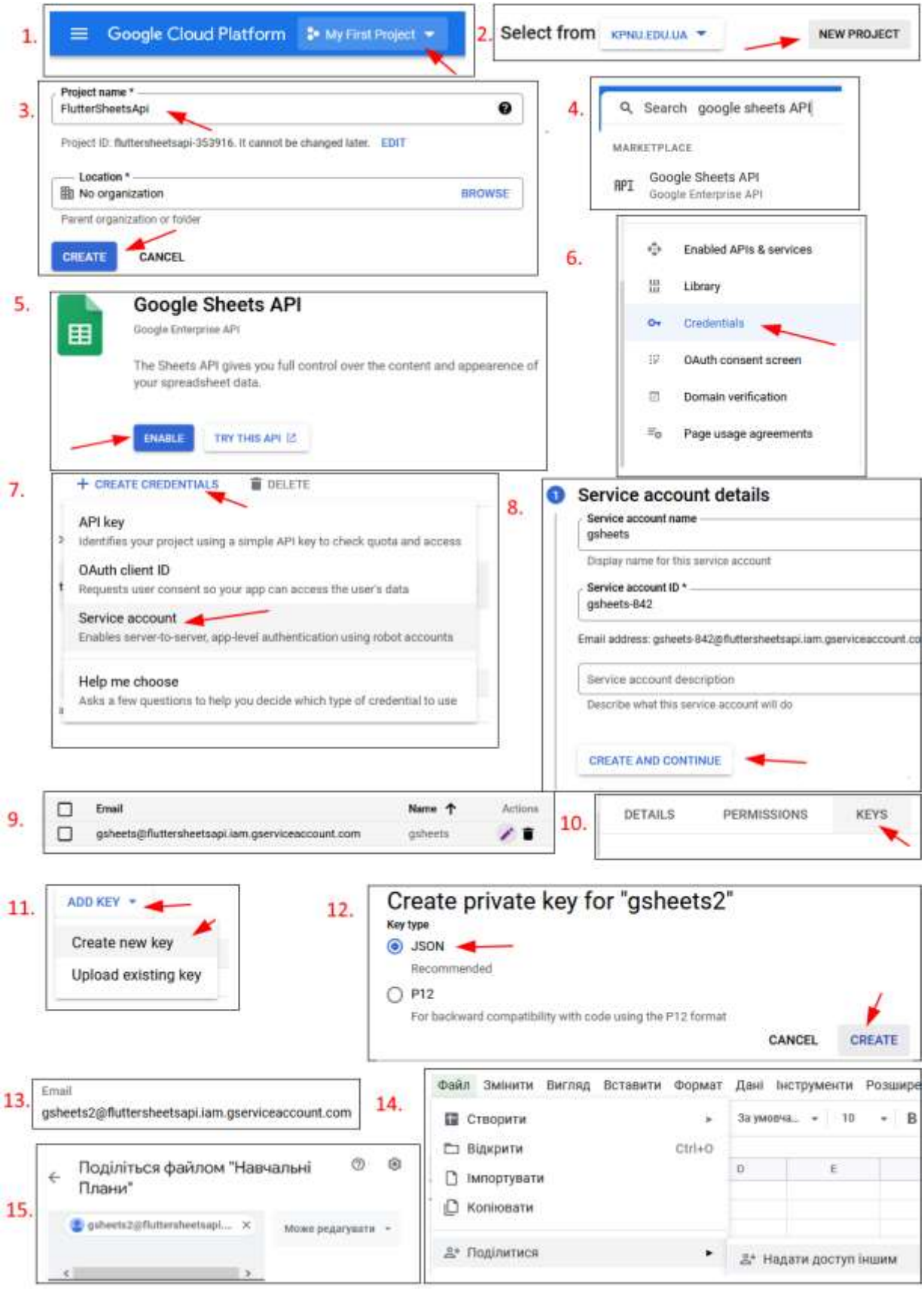

Рис. 2.1. Покрокова інструкція по підключенню Google Sheets API до проєкту

2.3. Проєктування та створення мобільного додатку

Для початку роботи створюємо новий Dart / Flutter проєкт у середовищі розробки Android Studio. Видаляємо весь код, що створюється разом зі створенням нового проєкту по замовчуванню. Перше наше завдання — це підключення Google Sheets API до нашого проєкту. Отже, необхідно підключити користувацький пакет gsheets до нашого проєкту. Для початку роботи пакет приєднується за допомогою змін у файл pubspec.yaml у кореневій папці проєкту. Там ми у пункт «dependencies» (залежності) дописуємо назву пакету та його версію, у нашому випадку — «gsheets: ^0.4.2». Середовище розробки Android Studio запропонує додати залежності у проєкт — вибираємо відповідний пункт меню, що з'явилось. Цей фрагмент у файлі pubspec.yaml виглядає так:

```
dependencies:
  flutter:
    sdk: flutter
  gsheets: ^0.4.2
```

Тепер у папці проєкту lib створюємо новий файл, з назвою google_sheets_api.dart. Це буде основним файлом класу GoogleSheetsApi, у якому буде реалізовуватись ініціалізація таблиць та сторінок таблиць, отриманих за допомогою Google Sheets API, а також більшість методів програми, що реалізують отримання певних конкретних даних з таблиць за певними критеріями. Саме тут будуть писатись ті функції, до яких найкраще буде звертатись через екземпляр класу, в залежності від контексту.

У коді файлу google_sheets_api.dart ми спочатку імпортуємо користувацький пакет gsheets, для того щоб можна було користуватись тими методами, що включаються у цей пакет. Наше наступне завдання — це безпосередньо зв'язати API з нашим проєктом. Для цього ми релізуємо клас GoogleSheetsApi, та створюємо в середені нього константу credentials (Реквізити для входу). В цю константу нам необхідно внести дані ключа, що ми отримали раніше у вигляді файлу з

розширенням json. Потрібно скопіювати з цього файлу абсолютно всі дані, включаючи дуже довгий приватний ключ, і занести їх у константу.

Далі потрібно ініціалізувати безпосередньо документи (таблиці), з якими програма буде працювати. Так як за виведеною умовою таблиці має бути дві, створюємо дві змінні. У кожну змінну як літерал ми повинні внести ID таблиць. Його можна знайти у URL сторінки з таблицею — переходимо на обидва документи, та копіюємо з адресної стрічки веб-браузера довгий код, що знаходиться між /d/ та /edit. Вносимо ідентифікатори у змінні spreadsheetId та spreadsheet2Id відповідно.

Користувацький пакет gsheets потребує від нас створення екземпляру класу GSheets, у аргументи якого буде внесені реквізити для входу (credentials). Наступне що ми робимо — це створюємо змінні класу Worksheet, які відповідатимуть за те, яка сторінка відкрита у обраному документі. Створюємо дві змінні — worksheet та worksheet2 відповідно.

Функціонал пакета не буде працювати без ініціалізації, отже ми створюємо функцію, що вибере перші сторінки документів, з якими ми будемо працювати. Так як ми працюємо з API, а отже беремо дані з інтернету, ми не можемо одразу отримати потрібні нам дані після запуску додатку. Повинен пройти деякий час перед тим, як екземпляр додатку отримає ці дані. Для того щоб ми могли працювати з такими даними, ми повинні використовувати таке поняття мови Dart і фреймворку Flutter як асинхронні функції. Вони дозволяють виконати дії у додатку тільки після того, як отримує результат від методів у середині, при чому сама програма в той час може працювати, маючи це на увазі. Значить, що у нашому випадку нам потрібно використати асинхронну функцію з ключовим словом Future для того, щоб ініціалізувати перші сторінки документів. Завдяки тимчасовим об’єктам ми переносимо у змінні worksheet інформацію про поточну сторінку. Обираємо сторінку ми методом з пакету gsheets worksheetByIndex, де ми обираємо першу сторінку документів по індексу, у нашому випадку 0, щоб працювати з першими

сторінками документів. Отже, файл google_sheets_api.dart повинен виглядати таким чином:

```
class GoogleSheetsApi {
  static const _credentials = r'''
{
  "type": "service_account",
  "project_id": "fluttersheetsapi",
  "private_key_id": "62bf0085be27f3508fea8187b6658f90a38f4d22",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC197m2J7I6Cd+F\n
qEuxG+7i7tTBCySSThitEYHKjnn4XW7+dqqhP8yRhSPtJ01w79Qjo7Ha7dq31h4W\n+p
7toGvPjkEONE76D7L6t0et1sVIMPGtHt/QL88KLtp1Tsybgw8NPFwK2T2Klt9A\nlVQC
jDmFUIhgyuGaRFVMldpCDhxzKcKK4qR7BR8mVKgNeDHOXNSLSdsPjjU+eqGz\n26jEZP
Gmpv5sJ8eyYyQUoeS2l2dt2rRCtOZQlnT/zcVApguDSm5f9iiRx1qJO9Vd\nAUlfn7tn
Tj4RP/7lfOOXEIVdgMLmiKBLLSQwZFQBHMnkr97OJSvgvQNt/T9kTMZ/\ntY0LETUDAg
MBAAECggEARpcIqJTCq4gr+W4dmuGyrL8ospCPieAY636Yoord3w7j\n4XwIWlBlG+iR
1MyD2rV2zrKQatUNnXZHKvPjVlx+pkBRlWCOaKxRarhR6qqf0o8G\nGfoZqen5/8HIRm
ygrA1N9/z72cuHd61qguPK+MgMAl8I4L+jgBRVIMYbKrfNqpCX\npu8K2DuN22/UmNp+
pi719V123YPiT5SpjBdECTM6Yn60gHfRwKgSOg+4MoHJhJ6o\npA1jd0FP7hnKLzSB19
yNxsjZngSWMCiptl2cTzvB5gLweji1/YTbFmFbCXJixacR\nayPCfXO7a8+Ll2mU/nhN
6JJyl8mE9xH7/C53q3hwNQKBgQDb0gYafRH/t1WZRoup\nGDOLTbxae8/F90hqN4ox/p
ls6hvgJXpF3XndLah67ODL42pF5fa2AVaDhBJTuGI/\nRGoDZpDM8erFRh6ullf7OUec
YN+M+E0+pMoraUeW6pz3HLyYOmM8KHR/XwYmUw0o\nr3+p/Gki/5fXFNwhrscGkXxChQ
KBgQDT6s3KIFzr26svIh8u5ox9FvPPcOlG3Bus\nz98De60kqrrOnXkUFx6V7ke5vzib
zv1zZRXMLDhx2rCPg5iBoeunaZpwmv20UFTc\nonLNgR/bYY1fmVTdhbC5gXE2ZZV4BT
VNNIdAIZyT78obdf5HVdv0OKz1jz8Z0C7i\nh6wQRUUj5wKBgQCRxWEOx8nbzfQRxNpw
s//abR5u0pQiL63x4grexHtx4n3B0piX\nJBvgNJr5iYQ5MQIUgD9uysxYQoj0w91DH1
c+EhIMFpEdaTaHiGOn0YBPj14v2ak/\n1X/L8fDrf4G/zWqzeGS+TAWVQaqOV5zuUhNS
4nPGJcDHfKnOHF0XWfsREQKBgQCS\nKbc6nEu62c4eOjirzpRDitsqzsP2/cWQ0ecNrf
Xj/mXOMErckn2kB68s7DkNLh0d\nxDDem/lqfd2dfrRnZNj2pR3CYQRvo6CEdP8dwtvI
Imax8z3XdGhJ6n+9dx86osz6\nrSpED2rBlngdrKpvZUbKEXUqAV0SXsaXs9/3cO+ZLQ
KBgCPV3dKQK2RtNgf/sfBe\nQ16vxK9HctY3Icv7uOAo5HkdFXrqdm5PddqxBRBLv9xF
AeJMaJR+a6g1ScVXCoEN\nSEZyd23vm2EN2fh978S4q3fNw04YF9wT1m3kp9AZ8yEIeO
SgX2Zal2gtpWs8eE7f\nPA9p1fBXToonyBe/y+KIeHWF\n-----END PRIVATE KEY--
---\n",
  "client_email":
"gsheets@fluttersheetsapi.iam.gserviceaccount.com",
  "client_id": "103629475708125275827",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/gsheets%40flutter
sheetsapi.iam.gserviceaccount.com"
}
''';

  static final _spreadsheetId = '1MrRLNwWKg7N5coWm-kaaOphUiE9OP4b-
```

```
PFxK9HjC8lM';
  static final _spreadsheet2Id =
'1_sBKCshEnmgxrE2DqEB3nNHfchAU7SI_snu8yRwGPuY';
  static final _gsheets = GSheets(_credentials);
  static Worksheet? _worksheet;
  static Worksheet? _worksheet2;


  static Future init() async {
    final ss = await _gsheets.spreadsheet(_spreadsheetId);
    _worksheet = ss.worksheetByIndex(0);

    final ss2 = await _gsheets.spreadsheet(_spreadsheet2Id);
    _worksheet2 = ss2.worksheetByIndex(0);
  }
}
```

У нашому мобільному додатку повинні бути декілька сторінок, між якими користувач буде переключатися. Необхідно створити домашню сторінку — першу сторінку програми, яку користувач буде бачити одразу після запуску мобільного додатку, та на якій студент зможе вибрати групу, у якій навчається, та власне ім’я по списку групи. Після цього він повинен потрапити до наступної сторінки, де він побачить список дисциплін, до яких приєднаний. З цього списку він буде мати змогу обрати дисципліну, яка його цікавить, і після вибору потрапити до наступної сторінки з безпосередньо інформацією про обрану дисципліну. Отже, потрібні щонайменш 3 сторінки — домашня (стартова) сторінка, сторінка с дисциплінами, та сторінка з інформацією про дисципліну.

Створюємо три нових файли у папці проєкту відповідно для цих цілей, з назвами home_page.dart, subjects_page.dart та plan_page.dart. Хоч запуск додатків у Dart та Flutter починається з файлу main.dart, реалізовувати програмний інтерфейс з цього файлу не рекомендується, тому ми і створюємо домашню сторінку home_page.dart, на яку при запуску додатку файл main.dart і буде перенаправляти. Для того, щоб у додатку можна було зручно переключатись між сторінками, та мати змогу переносити динамічні дані з однієї сторінки на іншу, нам потрібно буде реалізувати функцію роутингу.

Створюємо у папці проєкту нашого додатку новий файл route_generator.dart. До функцій у ньому ми будемо звертатись кожен раз, коли будемо перемикатись між сторінками у додатку. На початку файлу нам потрібно підключити всі пакети файлів, які будуть використовуватись при роутингу. Одразу після цього ми створюємо клас RouteGenerator, де і пишемо функцію маршрутизації. Код файлу route_generator.dart у моїй програмі після написання функціоналу виглядає таким чином:

```
import 'package:flutter/material.dart';
import 'package:student_plan/home_page.dart';
import 'package:student_plan/plan_page.dart';
import 'package:student_plan/subjects_page.dart';

class RouteGenerator {
  static Route<dynamic> generateRoute(RouteSettings settings) {
    final args = settings.arguments;

    switch (settings.name)
    {
      case '/': return MaterialPageRoute(builder: (_) =>
HomePage());
      case '/subjects':
        if (args is String) {
          return MaterialPageRoute(builder: (_) => Subjects(data:
args),);
        }
        return _errorRoute();
      case '/plan':
        if (args is String) {
          return MaterialPageRoute(builder: (_) => PlanPage(data:
args));
        }
    }
    return _errorRoute();
  }

  static Route<dynamic> _errorRoute() {
    return MaterialPageRoute(builder: (_) {
      return Scaffold(
        appBar: AppBar(
          title: Text('Error'),
        ),
        body: Center(
          child: Text('ERROR'),
        ),
```

```
        );
      });
    }
}
```

У даному коді ми реалізуємо можливість отримати дані, які можна буде перенести до наступної сторінки програми. Ми використовуємо віджет Flutter RouteSettings, щоб прийняти дані як аргумент у функцію, після цього заносимо отримані дані у змінну. У switch case функції ми робимо розгалуження, тут реалізується перехід на іншу сторінку за допомогою метода builder. У аргументах назви класів ми передаємо дані на наступну сторінку, при цьому перевіряючи, чи є вони необхідним типом даних. Якщо з якихось причин при виклику функції generateRoute аргументом буде передаватись сторінка, не вказана у конструкції switch, програма покаже вікно з помилкою.

Для того щоб безпосередньо при запуску додатку нас перенаправляли на домашню сторінку, потрібно вказати у файлі main.dart першочерговий маршрут, перед цим підключивши файл route_generator.dart. Єдине що ще потрібно зробити — викликати метод генерації маршрутів. Отже, код файлу main.dart виглядає у нашій програмі таким чином:

```
import 'package:flutter/material.dart';
import 'package:student_plan/route_generator.dart';

void main() => runApp(MaterialApp(
      initialRoute: '/',
      onGenerateRoute: RouteGenerator.generateRoute, ));
```

Для створення домашньої (стартової) сторінки додатку необхідно створити її інтерфейс. В першу чергу ми підключаємо до проєкту пакет віджетів дизайну та користувацького інтерфейсу типу Material Design, що реалізовані у файлі material.dart. Так як стан віджетів на цій сторінці повинен змінюватись динамічно в залежності від даних, що надходять до неї, ми реалізуємо програмний інтерфейс

на основі віджету Stateful Widget (віджети зі станом). Отже, я створюю клас HomePage, що розширюється від класу StatefulWidget. У віджеті створення я повертаю віджет Scaffold, на основі якого і будую користувацький інтерфейс.

Реалізація функціоналу на домашній сторінці була запланована таким чином: потрібно, щоб користувач міг обрати зі списку груп, що буде отриманий з таблиці Google Sheets, свою групу, після чого студент повинен отримати можливість обрати себе з списку студентів що навчаються у цій групі. Коли користувач обирає потрібні дані, він зможе окремою кнопкою перейти до наступної сторінки, дані на якій будуть залежати від того, що користувач обрав на домашній сторінці. Так як ми отримуємо дані асинхронно, користувач не зможе одразу їх обирати, тому що потрібен час, щоб інформація з мережі інтернет дійшла до клієнта — мобільного додатку. Отже потрібно також реалізувати індикатор завантаження даних.

Угорі сторінок програми буде AppBar. Елементи ми будуємо вертикально через віджет Column, та по центру, через віджет Center. Були додані віджети користувацького інтерфейсу, такі як текстові віджити.

Для реалізації меню вибору групи та конкретного студента ми обрали віджет DropdownButton. Він показує вибраний користувачем елемент, а після дотику по ньому розкриває повне меню з усіма елементами в списку. Цей віджет є компактним, принцип роботи інтуїтивно зрозумілим, та підходящим для задекларованих цілей.

Отже, щоб реалізувати частину функціоналу додатку у цьому вікні нам в першу чергу потрібно отримати дані про групи, щоб занести їх у перше меню DropdownButton. Для цього потрібно використовувати методи з класу GoogleSheetsApi. Але ми поки що не можемо працювати з Google таблицями, тому що метод ініціалізації GoogleSheetsApi ще досі не був викликаний. Найефективніше буде викликати його при створенні (побудові) домашньої сторінки додатку. Щоб реалізувати цей функціонал потрібно перевантажити неявний метод initState. Цей метод відповідає за той функціонал, що віджет сторінки виконує при першій її побудові. У класі стану класу HomePage ми

створюємо новий метод loadFirstData, та перевантажуємо метод initState, щоб він викликав цей метод. Метод loadFirstData не повинен повертати дані, отже має бути типу void, та отримувати дані з мережі інтернет, отже він отримає дані не відразу після виклику. Це значіть, що метод повинен буди асинхронними. У методі loadFirstData ми викликаємо статичну функцію ініціалізації з класу GoogleSheetsApi. Фрагмент коду:

```
...
class _HomePageState extends State<HomePage> {
void loadFirstData() async {
  WidgetsFlutterBinding.ensureInitialized();
  await GoogleSheetsApi.init();
}

@override
void initState() {
  super.initState();
  loadFirstData();
}
...
```

Тепер ми можемо викликати всі функції, що потрібно виконати при запуску додатку з середини метода loadFirstData. Відштовхуючись від цього створюємо функціонал для отримання списку груп. На сторінці створюємо змінну типу List<String> (Список рядків). У класі GoogleSheetsApi створюємо метод, що буде повертати нам список груп з першої сторінки документу. Для цього використовуємо методи пакету gsheets. Метод робимо статичним (ключовим словом static на початку методу), щоб до нього можна було звернутись без створення екземплярів класів з будь якої сторінки, до якої підключений файл google_sheets_api.dat. Клітинки, які ми вибираємо у якості аргументів для методів пакету gsheets, є незмінними та вибрані розробником. Алгоритми додатку були написані з урахуванням певного визначеного розміщення даних у таблиці, тому для коректної роботи додатку таблиці з даними повинні бути заповнені за певними правилами, розписаними далі у роботі. Приклад коду створеного методу у класі GoogleSheetsApi:

```
...
static Future<List<String>> getGroups() {
  return _worksheet!.values.column(2, fromRow: 3);
}
...
```

У файлі домашньої сторінки ми викликаємо метод з класу та заносимо отримані дані у створений раніше список. Не забуваємо при цьому використовувати ключове слово await:

```
...
List<String> groupsdata = [];
void loadFirstData() async {

...
groupsdata = (await GoogleSheetsApi.getGroups());

}
```

Таким чином ми отримали дані про групи у змінну типу список. Тепер ми можемо реалізувати меню з отриманими даними. Фрагмент коду:

```
class _HomePageState extends State<HomePage> {
...
@override
Widget build(BuildContext context) {
  DropdownMenuItem<String> buildMenuItem(String item) =>
DropdownMenuItem(
        value: item,
        child: Text(
          item,
          style: TextStyle(
            fontWeight: FontWeight.bold,
            fontSize: 20,
          ),
        ),
      );
...
DropdownButton<String>(
  value: value,
  iconSize: 36,
  isExpanded: true,
  items: groupsdata.map(buildMenuItem).toList(),
  onChanged: (value){
```

```
    setState(() => this.value = value);
    value2 = null;
    isVisible2 = true;
    selectedGroup = value;
    loadStudentData(selectedGroup!);
  },
...
```

Також у коді був реалізований показ індикатора завантаження. Він працює як текст, який покладений у віджет Visibility. Коли дані завантажуються, булева функція isVisible змінює значення, і текст зникає. Щоб стан віджетів на екрані змінювався, потрібно писати до функціоналу метод setState.

Після того як користувач обере групу, програма має знайти студентів з цієї групи, та перенести їх список у змінну. Для цього програма спочатку має перейти до сторінки в документі певної групи, переданої як аргумент, та повернути клітинки, де вказані імена студентів. Фрагмент коду:

```
...
static Future<List<String>> getGroupStudents(String groupname)
async{
  {
    final ss = await _gsheets.spreadsheet(_spreadsheetId);
    _worksheet = ss.worksheetByTitle(groupname);

    final ss2 = await _gsheets.spreadsheet(_spreadsheet2Id);
    _worksheet2 = ss2.worksheetByTitle(groupname);

    return _worksheet!.values.row(1, fromColumn: 3);
  }
}
...
```

Ці функції будуть викликатись не при ініціалізації, а при виборі першої групи, тобто у методі loadStudentData, з вибраною групою у якості аргумента. Цей метод також є асинхронним, тому також контролює стан власного індикатора завантаження.

Фрагмент коду:

```
...
void loadStudentData(String groupname) async {
  WidgetsFlutterBinding.ensureInitialized();
  studentsdata = (await
GoogleSheetsApi.getGroupStudents(groupname));
  isVisible2 = false;
  setState(() {});
}
...
```

Коли користувач обере всі необхідні дані, він зможе натиснути на кнопку «Перейти далі», що відправить його на наступну сторінку класу Subjects, передавши при цьому дані про поточного студента.

Щоб отримати на новій сторінці дані з іншої сторінки, потрібно визначити потрібні змінні у конструкторі основного класу. Фрагмент коду:

```
...
class Subjects extends StatefulWidget {

  final String? data;

  const Subjects({Key? key, @required this.data,}) : super(key: key);


  @override
  State<Subjects> createState() => _SubjectsState();
}
...
```

На сторінці зі списком дисциплін теж повинен бути метод первинної ініціалізації loadFirstData. В першу чергу у ньому ми можемо передати з нашого конструктора у змінну. Викликаємо його через об’єкт widget. Наступним кроком є отримання списку дисциплін. Звертаємось до нового методу з класу GoogleSheetsApi, відправивши у якості аргументу дані про студента.

Отже, у класі ми створюємо метод, що буде повертати дисципліни, приєднані до певного студента. Але для цього потрібно спочатку знайти стовпець, що

відповідає цьому студенту — значить цей метод має викликати інший метод, що поверне потрібний номер стовпця, для використання з функціоналом пакету gsheets. Цей метод буде виглядати таким чином:

```
...
static Future<int?> getStudentRow(String? sudent) async {
  {
    var ListOfSuds = await _worksheet!.values.row(
        1, fromColumn: 2);
    for (var i = 0; i < ListOfSuds.length; i++) {
      if (ListOfSuds[i] == sudent) {
        return await i;
      }
    }
  }
}
...
```

А виклик цього методу з основної функції:

```
...
var column  = await _worksheet!.values.column((await getStudentRow(student))!+2
), fromRow: 2);
...
```

Так як нам необхідно створити певну кількість віджетів-кнопок, але ми не знаємо, яку кількість до того, як не отримаємо дані, нам потрібно використовувати клас який динамічно створить нам необхідну кількість віджетів. Для таких цілей використовується віджет ListView, з конструктором типу Builder. Отже, ми реалізуємо у тілі нашого класу сторінки Subjects віджет ListView:

```
ListView.builder(
    itemCount: (subjectsData.length),
    itemBuilder: (context, index) {
      return SubjectButton(child: subjectsData[index],);
    }),
```

Він буде створювати елементи SubjectButton, виведені окремо з child як конструктором.

При виборі кнопки з назвою дисципліни дані про неї відправляються до конструктора класу PagePlan. При ініціалізації на сторінці генеруються дані, отримані з полів документу вибраної групи. Необхідна сторінка групи відкривається ще під час ініціалізації у класі HomePage. У таблиці спочатку шукається поле дисципліни, після чого з відповідного сегменту зчитуються дані та показуються на сторінці плану. Якщо якась з необов'язкових клітинок не заповнена — контейнер для неї не показується на самій сторінці.

2.4. Вимоги до оформлення Google Sheet документу

Для коректного зчитування даних з таблиць Google Sheet, необхідно заповнювати таблиці з даними за певними правилами. Це необхідно для того щоб алгоритми програми змогли знайти саме потрібні дані та передати їх у додаток.

Дані мають знаходитись у вигляді двох Google Sheet таблиць. Розділення створено з метою того, щоб не довелося орієнтуватися у занадто великій кількості сторінок у одному документі.

Перший документ повинен бути поділений на аркуші, де кожен аркуш, окрім першого, має бути названим кодом усіх груп, інформацію про які ми хочемо отримувати в додаток.

Перша ж сторінка (назва може бути довільною) має виглядати як відношення порядкового номеру аркуша та коду групи, інформація про яку знаходиться на цьому аркуші. Найменування стовпців має бути на другому рядку, а список груп починатись з третього. Тільки перші два стовпці — порядковий номер сторінки та кодом групи — є обов'язковими. Після останнього рядку таблички клітинки мають бути пустими. Додаткова перша сторінка необхідна через те, що Google Sheets API не дає нам технічної можливості ітерації через назву аркушу в таблиці. А без цього ми змогли б у майбутньому видаляти групи та додавати нові.

Приклад оформлення першого аркуша подано на рис. 2.2.

	A	B	C	D	E
1	Список груп та відповідна кожній сторінка у документі (порядковий номер)				
2	Аркуш, №	Група	Курс	Семестр	Рівень В.О.
3	2	КВ1-В18	4	2	Бакалавр

Рис. 2.2. Приклад оформлення першого аркуша у документі 1

Сторінки груп мають мати назву, ідентичну до назв груп на першій сторінці. Сторінки груп мають бути оформлені за таким шаблоном (рис. 2.3).

	A	B	C	D
1		Студент 1	Студент 2	Студент 3
2	0	Семестр 1	Семестр 1	Семестр 1
3	1			
4	2	Дисципліна 1	Дисципліна 1	Дисципліна 1
5	3	Дисципліна 2	Дисципліна 2	Дисципліна 2
6	4	Дисципліна 3	Дисципліна 3	Дисципліна 3
7	5	Дисципліна 4	Дисципліна 4	Дисципліна 4
8	6	Семестр 2	Семестр 2	Семестр 2
9	7			
10	8	Дисципліна 5	Дисципліна 1	Дисципліна 1
11	9	Дисципліна 6	Дисципліна 1	Дисципліна 1
12	10	Семестр 3	Семестр 3	Семестр 3
13	11			
14	12	Дисципліна 7	Дисципліна 7	Дисципліна 7
15	13	Дисципліна 8	Дисципліна 8	Дисципліна 8
16	14	Дисципліна 9	Дисципліна 9	Дисципліна 9
17	15	Семестр 4	Семестр 4	Семестр 4
18	16			
19	17	Дисципліна 10	Дисципліна 10	Дисципліна 10
20	18	Дисципліна 11	Дисципліна 11	Дисципліна 11
21	19	Дисципліна 12	Дисципліна 12	Дисципліна 12
22	20	Дисципліна 13	Дисципліна 13	Дисципліна 13
23	21	Дисципліна 14	Дисципліна 14	Дисципліна 14

Рис. 2.3. Приклад оформлення аркуша з кодом групи у документі 1

Другий документ теж повинен бути поділений на аркуші, де кожен аркуш, має бути названим кодом усіх груп. Сторінки груп мають бути оформлені за таким шаблоном (рис. 2.4):

Рис. 2.4. Приклад оформлення аркуша з кодом групи у документі 2

2.5 Взаємодія користувачів з фінальним продуктом

Після запуску мобільного додатку студент потрапляє до стартового екрану (рис. 2.5). Тут, у першому розгортаємому меню, він повинен обрати свою групу, у якій навчається, після чого зможе знайти своє ім’я та прізвище у другому меню. Після вибору елементу у обох меню стане доступною кнопка «Перейти далі», що перенесе користувача на другу, головну сторінку додатку.

Рис. 2.5. Стартове меню додатку

Головне меню додатку являє собою список дисциплін, до яких приєднаний студент. Дисципліни розділені по навчальним семестрам. Студент може вибрати дисципліну, щоб отримати детальну інформацію про дисципліну.

Приклад основного меню додатку подано на рис. 2.6.

Рис. 2.6. Приклад основного меню додатку

Після вибору дисципліни, що цікавить студента, він переходить до меню з інформацією про навчальну дисципліну (рис. 2.7). Тут він зможе побачити всі заповнені навчальним керівництвом дані, про назву курсу, викладача, обсяг курсу, тип семестрового контролю. Також вказані посилання на завантаження силабусу курсу, сторінку курсу на платформі Moodle. Також тут відображається примітка

про курс, якщо така присутня. Якщо якісь з додаткових даних про курс, як контакти викладача або посилання, вони на сторінці не відображаються.

Рис. 2.7. Приклад сторінки навчальної дисципліни

ВИСНОВКИ

Результатом дипломної роботи став повноцінний мобільний додаток «Індивідуальний навчальний план студента». Програма була створена на базі платформи Flutter, з використанням сервісу Google Sheets для роботи з даними.

Готову програму можна використовувати за прямим призначенням у межах Фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка.

У рамках дипломного проєкту я ознайомився з інструментами, що можуть використовуватись для створення мобільних додатків, розглянув ринок сучасної мобільної розробки, вивів їх основні атрибути та особливості. Завдяки цьому я розібрався у актуальності напрямків мобільної розробки, та виходячи з цієї інформації обрав інструменти розробки, які можуть мені знадобитися у майбутньому.

За час написання роботи я отримав великий об’єм знань та навичок, необхідних для створення крос-платформних додатків з використанням інструментів Flutter та Dart. Практична частина роботи допомогла мені поглибитись в усі деталі створення готових проєктів на основі фреймворку Flutter. Отриманні знання дозволять мені і надалі працювати над проєктами, що використовують ці технології.

Окрім практики у роботі зі специфічними інструментами для розробки, я також отримав досвід у розробці програмного забезпечення в широкому сенсі. Я покращив навички у пошуку причин проблем під час написання програмного коду та їх вирішенню, постановці завдання для програмного продукту, створенні користувацької документації, а самостійне вивчення обраних мною інструментів розробки — у пошуку інформації та отриманню нових знань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Положення про Індивідуальний навчальний план здобувача вищої освіти Кам'янець-Подільського національного університету імені Івана Огієнка (Нова редакція) від 31.08.2021р.

2. Eric Windmill. (2020). Flutter In Action. Shelter Island, NY: Manning Publications Co.

3. Marco L. Napoli. (2019). Beginning Flutter: a Hands On Guide to App Development. Wrox; John Wiley & Sons.

4. Martin Sikora. (2015). Dart Essentials. UK: Packt Publishing.

5. Mike Katz, Kevin D Moore, Vincent Ngo, Vincenzo Guzzi. (2021). Flutter Apprentice. Learn to Build Cross-Platform Apps.

6. Simone Alessandria, Brian Kayfitz. (2021). Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart. UK: Packt Publishing.

7. URL:https://appinventiv.com/blog/flutter-is-cross-platform-development-future/

8. URL:https://www.businessnewsdaily.com/5155-mobile-app-development.html.

9. URL:https://dart.dev/guides/language/language-tour

10. URL:https://developers.google.com/sheets/api.

11. URL: https://flutter.dev/development.

12. URL: https://material.io/develop.

13. URL:https://technostacks.com/blog/mobile-app-development-frameworks/.

14. URL:https://surf.dev/flutter-architecture-guide/

ДОДАТОК

Програмний код мобільного додатку

«Індивідуальний навчальний план студента»

Вміст файлу pubspec.yaml

```
name: student_plan
description: A new Flutter project.

publish_to: 'none'


version: 1.0.0+1

environment:
  sdk: ">=2.16.2 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  gsheets: ^0.4.2
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter


  flutter_lints: ^1.0.0



  uses-material-design: true

  assets:
    - assets/moodle-logo.png
```

Вміст файлу main.dart

```
import 'package:flutter/material.dart';
import 'package:student_plan/route_generator.dart';

void main() => runApp(MaterialApp(
      initialRoute: '/',
      onGenerateRoute: RouteGenerator.generateRoute, ));
```

Вміст файлу route_generator.dart

```
import 'package:flutter/material.dart';
import 'package:student_plan/home_page.dart';
import 'package:student_plan/plan_page.dart';
import 'package:student_plan/subjects_page.dart';

class RouteGenerator {
  static Route<dynamic> generateRoute(RouteSettings settings) {
    final args = settings.arguments;

    switch (settings.name)
    {
      case '/': return MaterialPageRoute(builder: (_) =>
HomePage());
      case '/subjects':
        if (args is String) {
          return MaterialPageRoute(builder: (_) => Subjects(data:
args),);
        }
        return _errorRoute();
      case '/plan':
        if (args is String) {
          return MaterialPageRoute(builder: (_) => PlanPage(data:
args));
        }
    }
    return _errorRoute();
  }

  static Route<dynamic> _errorRoute() {
    return MaterialPageRoute(builder: (_) {
      return Scaffold(
        appBar: AppBar(
          title: Text('Error'),
        ),
        body: Center(
          child: Text('ERROR'),
        ),
      );
    });
  }
}
```

Вміст файлу home_page.dart

```
import 'package:flutter/material.dart';
import 'package:student_plan/google_sheets_api.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key
      })
      : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {

  List<String> groupsdata = [];
  List<String> studentsdata = [];
  String? value;
  String? selectedGroup;
  String? value2;
  String? currentStudent;
  bool isVisible = true;
  bool isVisible2 = false;

  void loadFirstData() async {
    WidgetsFlutterBinding.ensureInitialized();
    await GoogleSheetsApi.init();

    groupsdata = (await GoogleSheetsApi.getGroups());
    isVisible = !isVisible;
    setState(() {});

  }

  void loadStudentData(String groupname) async {
    WidgetsFlutterBinding.ensureInitialized();
    studentsdata = (await
GoogleSheetsApi.getGroupStudents(groupname));
    isVisible2 = false;
    setState(() {});
  }

  @override
  void initState() {
    super.initState();
    loadFirstData();
  }

  @override
  Widget build(BuildContext context) {
    DropdownMenuItem<String> buildMenuItem(String item) =>
```

```
DropdownMenuItem(
          value: item,
          child: Text(
            item,
            style: TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 20,
            ),
          ),
        );

    return Scaffold(
      appBar: AppBar(
        elevation: 20,
        backgroundColor: Colors.deepOrange[900],
        automaticallyImplyLeading: false,
      ),
      body: Center(
        child: Column(
          children: [
            Container(
              padding: const EdgeInsets.fromLTRB(10, 10, 10, 20),
              child: Column(
                children: [
                  Text("Фізико-математичний факультет КПНУ",
                      style: TextStyle(
                        fontSize: 14,
                        color: Colors.grey[900],
                      )),
                  SizedBox(
                    height: 10.0,
                  ),
                  Text("ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН",
                      textAlign: TextAlign.center,
                      style: TextStyle(
                        fontSize: 16,
                        color: Colors.grey[900],
                        fontWeight: FontWeight.bold,
                      )),
                ],
              ),
            ),
            Visibility(
              visible: isVisible,
              maintainSize: true,
              maintainAnimation: true,
              maintainState: true,
              child: Container(
                  padding: const EdgeInsets.fromLTRB(10, 10, 10,
10),
                  child: Text("Завантажуються дані...",
```

```
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 12,
                          color: Colors.red,
                        ))),
              ),
              Container(
                  child: Text("ГРУПА",
                      style: TextStyle(
                        fontWeight: FontWeight.bold,
                        fontSize: 30,
                        color: Colors.deepOrange[900],
                      ))),
              Container(
                padding: const EdgeInsets.fromLTRB(100, 10, 100, 60),
                child: Container(
                  decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(8),
                      border: Border.all(color: Colors.black, width:
4)),
                  padding: EdgeInsets.symmetric(horizontal: 12,
vertical: 4),
                  child: DropdownButtonHideUnderline(
                    child: DropdownButton<String>(
                      value: value,
                      iconSize: 36,
                      isExpanded: true,
                      items: groupsdata.map(buildMenuItem).toList(),
                      onChanged: (value){
                        setState(() => this.value = value);
                        value2 = null;
                        isVisible2 = true;
                        selectedGroup = value;
                        loadStudentData(selectedGroup!);
                      },
                    ),
                  ),
                ),
              ),
              Visibility(
                visible: isVisible2,
                maintainSize: true,
                maintainAnimation: true,
                maintainState: true,
                child: Container(
                    padding: const EdgeInsets.fromLTRB(10, 10, 10,
10),
                    child: Text("Завантажуються дані...",
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 12,
```

```
                        color: Colors.red,
                      ))),
              ),
              Container(
                  child: Text("СТУДЕНТ",
                      style: TextStyle(
                        fontWeight: FontWeight.bold,
                        fontSize: 30,
                        color: Colors.deepOrange[900],
                      ))),
              Container(
                padding: const EdgeInsets.fromLTRB(20, 20, 20, 30),
                child: Container(
                  decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(6),
                      border: Border.all(color: Colors.black, width:
3)),
                  height: 55,
                  padding: EdgeInsets.symmetric(horizontal: 10,
vertical: 4),
                  child: DropdownButtonHideUnderline(
                    child: DropdownButton<String>(
                      value: value2,
                      iconSize: 26,
                      isExpanded: true,
                      items: studentsdata.map(buildMenuItem).toList(),
                      onChanged: (value){
                        setState(() => this.value2 = value);
                        currentStudent = value;
                      },
                    ),
                  ),
                ),
              ),
              Container(
                alignment: Alignment.center,
                padding: EdgeInsets.fromLTRB(10, 5, 10, 5),
                child: ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    primary: Colors.deepOrange[900],
                    minimumSize: Size(330, 80),
                    elevation: 20,
                    shape: new RoundedRectangleBorder(
                      borderRadius: new BorderRadius.circular(18.0),
                    ),
                  ),
                  child: Text(
                    'ПЕРЕЙТИ ДАЛІ',
                    style: TextStyle(fontSize: 32),
                  ),
                  onPressed: () {
```

```
                    Navigator.of(context).pushNamed('/subjects',
arguments: currentStudent);
                  },
                ),
              ),
            ],
          ),
        ),
      );
  }
}
```

Вміст файлу subjects_page.dart

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

import 'google_sheets_api.dart';

class Subjects extends StatefulWidget {

  final String? data;

  const Subjects({Key? key, @required this.data,}) : super(key:
key);



  @override
  State<Subjects> createState() => _SubjectsState();
}


class _SubjectsState extends State<Subjects> {
  List<String> subjectsData = [];
  List<String> semesters = [];
  String? currentStudent;
  int? studRowID;


  void loadFirstData() async {
    WidgetsFlutterBinding.ensureInitialized();
    currentStudent = widget.data;
    subjectsData = await
GoogleSheetsApi.getSemesters(currentStudent);

    setState(() {});
  }


  void createDrpMenu()
  {
```

```
  }


@override
  void initState() {
    super.initState();
    loadFirstData();
  }

  String? value;
  @override
  Widget build(BuildContext context) {

    return Scaffold(
        appBar: AppBar(
          elevation: 20,
          backgroundColor: Colors.deepOrange[900],
          actions: [
            IconButton(
              icon: Icon(Icons.menu),
              onPressed: () {},
            ),
          ],
        ),
        backgroundColor: Colors.grey[200],
        body: Column(
          child:(
                 Expanded(
              child: ListView.builder(
                  itemCount: (subjectsData.length),
                  itemBuilder: (context, index) {
                    return SubjectButton(child:
subjectsData[index],);
                  }),
            ),
          ),
        ));
  }
}

class SubjectButton extends StatelessWidget {
  final String child;
  const SubjectButton({Key? key, required this.child}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Container(
```

```
          child: ElevatedButton(
            style: ElevatedButton.styleFrom(
                primary: Colors.deepOrange[900],
                minimumSize: Size(430, 50)),
            onPressed: () {
              Navigator.of(context).pushNamed('/plan', arguments:
child);
              },
            child: Text(child),
          ),
          height: 40,
          color: Colors.red,
        ),
      );
    }
}
```

Вміст файлу google_sheets_api.dart

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

class GoogleSheetsApi {
  static const _credentials = r'''
{
  "type": "service_account",
  "project_id": "fluttersheetsapi",
  "private_key_id": "62bf0085be27f3508fea8187b6658f90a38f4d22",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC197m2J7I6Cd+F\nqEuxG+7i7tTBCy
SSThitEYHKjnn4XW7+dqqhP8yRhSPtJ01w79Qjo7Ha7dq31h4W\n+p7toGvPjkEONE76D7L6t0et1sVIMP
GtHt/QL88KLtp1Tsybgw8NPFwK2T2Klt9A\nlVQCjDmFUIhgyuGaRFVMldpCDhxzKcKK4qR7BR8mVKgNeD
HOXNSLSdsPjjU+eqGz\n26jEZPGmpv5sJ8eyYyQUoeS2l2dt2rRCtOZQlnT/zcVApguDSm5f9iiRx1qJO9
Vd\nAUlfn7tnTj4RP/7lfOOXEIVdgMLmiKBLLSQwZFQBHMnkr97OJSvgvQNt/T9kTMZ/\ntY0LETUDAgMB
AAECggEARpcIqJTCq4gr+W4dmuGyrL8ospCPieAY636Yoord3w7j\n4XwIWlBlG+iR1MyD2rV2zrKQatUN
nXZHKvPjVlx+pkBRlWCOaKxRarhR6qqf0o8G\nGfoZqen5/8HIRmygrA1N9/z72cuHd61qguPK+MgMAl8I
4L+jgBRVIMYbKrfNqpCX\npu8K2DuN22/UmNp+pi719V123YPiT5SpjBdECTM6Yn60gHfRwKgSOg+4MoHJ
hJ6o\npA1jd0FP7hnKLzSB19yNxsjZngSWMCiptl2cTzvB5gLweji1/YTbFmFbCXJixacR\nayPCfXO7a8
+Ll2mU/nhN6JJyl8mE9xH7/C53q3hwNQKBgQDb0gYafRH/t1WZRoup\nGDOLTbxae8/F90hqN4ox/pls6h
vgJXpF3XndLah67ODL42pF5fa2AVaDhBJTuGI/\nRGoDZpDM8erFRh6ullf7OUecYN+M+E0+pMoraUeW6p
z3HLyYOmM8KHR/XwYmUw0o\nr3+p/Gki/5fXFNwhrscGkXxChQKBgQDT6s3KIFzr26svIh8u5ox9FvPPcO
lG3Bus\nz98De60kqrrOnXkUFx6V7ke5vzibzv1zZRXMLDhx2rCPg5iBoeunaZpwmv20UFTc\nonLNgR/b
YY1fmVTdhbC5gXE2ZZV4BTVNNIdAIZyT78obdf5HVdv0OKz1jz8Z0C7i\nh6wQRUUj5wKBgQCRxWEOx8nb
zfQRxNpws//abR5u0pQiL63x4grexHtx4n3B0piX\nJBvgNJr5iYQ5MQIUgD9uysxYQoj0w91DH1c+EhIM
FpEdaTaHiGOn0YBPj14v2ak/\n1X/L8fDrf4G/zWqzeGS+TAWVQaqOV5zuUhNS4nPGJcDHfKnOHF0XWfsR
EQKBgQCS\nKbc6nEu62c4eOjirzpRDitsqzsP2/cWQ0ecNrfXj/mXOMErckn2kB68s7DkNLh0d\nxDDem/
lqfd2dfrRnZNj2pR3CYQRvo6CEdP8dwtvIImax8z3XdGhJ6n+9dx86osz6\nrSpED2rBlngdrKpvZUbKEX
UqAV0SXsaXs9/3cO+ZLQKBgCPV3dKQK2RtNgf/sfBe\nQ16vxK9HctY3Icv7uOAo5HkdFXrqdm5PddqxBR
BLv9xFAeJMaJR+a6g1ScVXCoEN\nSEZyd23vm2EN2fh978S4q3fNw04YF9wT1m3kp9AZ8yEIeOSgX2Zal2
gtpWs8eE7f\nPA9p1fBXToonyBe/y+KIeHWF\n-----END PRIVATE KEY-----\n",
  "client_email": "gsheets@fluttersheetsapi.iam.gserviceaccount.com",
  "client_id": "103629475708125275827",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
```

```
"https://www.googleapis.com/robot/v1/metadata/x509/gsheets%40fluttersheetsapi.iam.
gserviceaccount.com"
}
''';

  static final _spreadsheetId = '1MrRLNwWKg7N5coWm-kaaOphUiE9OP4b-PFxK9HjC8lM';
  static final _spreadsheet2Id = '1_sBKCshEnmgxrE2DqEB3nNHfchAU7SI_snu8yRwGPuY';
//вторая
  static final _gsheets = GSheets(_credentials);
  static Worksheet? _worksheet;
  static Worksheet? _worksheet2;


  static Future init() async {
    final ss = await _gsheets.spreadsheet(_spreadsheetId);
    _worksheet = ss.worksheetByIndex(0);

    final ss2 = await _gsheets.spreadsheet(_spreadsheet2Id);
    _worksheet2 = ss2.worksheetByIndex(0);
  }


  static Future<List<String>> getGroups() {
    return _worksheet!.values.column(2, fromRow: 3);
  }

  static Future changePage(int page) async {
    final ss = await _gsheets.spreadsheet(_spreadsheetId);
    _worksheet = ss.worksheetByIndex(page);
  }




  static Future loadGroups() async
  {
    var groups = (await GoogleSheetsApi.getGroups());
    return groups;
  }

  static Future<List<String>> getGroupStudents(String groupname) async{
    {
      final ss = await _gsheets.spreadsheet(_spreadsheetId);
      _worksheet = ss.worksheetByTitle(groupname);

      final ss2 = await _gsheets.spreadsheet(_spreadsheet2Id);
      _worksheet2 = ss2.worksheetByTitle(groupname);

      return _worksheet!.values.row(1, fromColumn: 3);
    }
  }


  static Future<int?> getStudentRow(String? sudent) async {
    {
      var ListOfSuds = await _worksheet!.values.row(
          1, fromColumn: 2);
      for (var i = 0; i < ListOfSuds.length; i++) {
        if (ListOfSuds[i] == sudent) {
          return await i;
        }
```

```
      }
    }
  }

  static Future<List<String>> getSemesters(String? student) async{

    int? studRow = (await getStudentRow(student))!+2;

    var column  = await _worksheet!.values.column((studRow), fromRow: 2);

    List semester = [];
    int? first = null;
    int? second = null;
    int? prev = null;
    bool gotfirst = false;

    for (int i = 0; i < column.length; i++) {
      if (column[i] == "") {
        if (gotfirst == false) {
          if (first == null) {
            first = i + 1;
          } else if (first != null) {
            second = i - 2;
            semester.add({first, second});
            gotfirst = true;
            prev = second;
            first = null;
            second = null;
          }
        } else {
          first = prev! + 3;
          second = i - 2;
          semester.add({first, second});
          prev = second;
        }
      }
    }
    semester.add({second! + 3, column.length - 1});

    return column;

  }

}
```