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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка застосунку для допомоги студентам у покращенні їх організації навчання та підходу до виконання академічних завдань

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

(підпис) (прізвище та ініціали)

Керівник (підпис) (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2023

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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)

« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Романюку Дмитру Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка застосунку для допомоги студентам у покращенні їх організації навчання та підходу до виконання академічних завдань

Керівник роботи Мудрик Іван Ярославович, д.ф., ст. викладач
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи вимоги до функціоналу системи, аналіз методик організації, використання функціональної мови JavaScript

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз предметної області та огляд конкурентів, технічний аспект проблеми, розробка моделі предметної області та бізнес моделі, проєктування архітектури додатку і його конструювання, тестування програмного забезпечення та оцінка його якості, розділ безпеки життєдіяльності та основ охорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)
Діаграма використання, діаграма сутностей(класів), діаграма сервісів, діаграма послідовності, діаграма станів, зображення графічного інтерфейсу додатку, слайди презентації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності			
Основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі		
2	Огляд існуючих додатків		
3	Розробка моделі та програмного комплексу		
4	Тестування програмного комплексу		
5	Підготовка пояснювальної записки		
6	Розділ безпеки життєдіяльності та основ охорони праці		
7	Підготовка презентації та доповіді		
8	Попередній захист		
9	Нормоконтроль, рецензування		
10	Допуск до захисту у зав. кафедри		
11	Захист роботи		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення», ТНТУ, 2023. Сторінок 69, рисунків 12, презентація.

Тема: Розробка застосунку для допомоги студентам у покращенні їх організації навчання та підходу до виконання академічних завдань.

В кваліфікаційній роботі бакалавра адресовано поширену, серед студентів, проблему поганої організації навчання. Встановлено причини, фактори, які на це впливають, та проведено опитування серед студентів щодо їх особистого досвіду. Як інструмент вирішення цієї проблеми було спроектовано та створено мобільний додаток, основною задачею якого є перетворення процесу навчання з гнітючого та складного в організований та ефективний. Він пропонує технологічні рішення та надає розширені функції користувачам для підвищення академічної успішності, покращуючи їх організацію навчання. Опираючись на актуальну літературу та визначивши потреби студентів було розроблено мобільний додаток. В розробці та дизайні використано ітеративний та користувач-орієнтований підходи, використано сучасні фреймворки та технології як ReactNative та Firebase. Тестування додатку проведено методами досвіду функціонального тестування та юніт-тестування.

Ключові слова: навчальні технології, менеджер завдань, академічна успішність, організація часу, персоналізоване навчання.

ANNOTATION

Bachelor’s qualification work. Ternopil Ivan Puluj National Technical University, Department of Software Engineering, specialty 121 “Software Engineering”, TNTU, 2023, Pages 69, figures 12, presentation.

Topic: Development of an app to help students in improving their study organization and academical tasks accomplishment approach.

The bachelor’s qualification work addresses the prevalent, among students, issue of poor study management. Reasons and factors, that influence this issue, were identified and addressed, also the query among students was held, to showcase their personal experience. As instrument to help solving this problem, a mobile app was designed and created, main task of which is to transform study process from daunting and hard to manageable and efficient. It offers technological solutions and advanced features, to help users achieve higher academic performance, by improving their study organization. Based on relevant literature and the identified need of students a mobile app was developed. The application’s design and development utilized an iterative and user-centric approach, using modern frameworks and technologies such as ReactNative and Firebase. Function and unit testing methods were used.

Keywords: educational technologies, task manager, academic performance, time management, personalized learning.

ЗМІСТ

ВСТУП

Враховуючи сучасні тенденції зростання ролі технологій в галузі освіти, відбувається значна зміна того, як студенти навчаються та організовують своє академічне життя. Великим чинником академічного успіху є ефективний менеджмент навчання, але багато студентів мають труднощі з осягненням концептів організації свого часу і створення ефективних стратегій навчання. Вони часто стикаються з великим академічним тиском та проблемами з ментальним здоров'ям, які надалі знижують їх академічну успішність. Беручи до уваги ці фактори та актуальність проблеми, було прийнято рішення створити мобільний додаток для допомоги студентам у покращенні їх організації навчання та підходу до виконання академічних завдань, який надає персоналізований підхід до навчання.

Основним зв'язком з дослідженнями в інженерії програмного забезпечення є розробка застосунків у галузі освітніх технологій та систем менеджменту навчання, які забезпечують різноманітні потреби та вподобання студентів. Розуміючи моделі навчання студентів, їх звички та пов'язані фактори, інженери програмного забезпечення зможуть створювати більш персоналізовані, привабливі та ефективні навчальні інструменти.

Метою цього дослідження є використання зібраних даних про моделі навчання студентів, задля обміркованого та ефективного проєктування та створення навчальних інструментів. Основними задачами є: збір даних та їх аналіз, інтерпретація та виведення висновків, проєктування, моделювання та створення додатку, тестування та оцінка кінцевої якості, визначення потенціалу та подальшої перспективи. Об'єктом дослідження є моделі навчання студентів в сучасному освітньому середовищі та вплив на них з боку соціальних, психологічних та особистих факторів. Предметом дослідження є розробка застосунку для допомоги студентам у покращенні їх організації навчання та

підходу до виконання академічних завдань.

Як методи дослідження було проведено аналіз предметної області, проблем та перешкод з якими стикаються студенти під час навчання, для цього було проведено та аналізовано детальне опитування серед студентів, задля висвітлення їх особистого досвіду у нинішній системі освіти. Після цього оглянуто та досліджено існуючих представників на ринку освітніх технологій, проведено аналіз їх сильних та слабких сторін, бізнес моделі, тощо. Визначено, які потреби студентів не є реалізовані на цьому ринку. Проаналізовано актуальну наукову літературу. Основними ідейними принципами додатку визначено заохочення саморегульованої освіти та ефективних навчальних технік, цілями яких є покращення загальної академічної успішності здобувачів освіти. Визначено його потенційну роль в галузі освітніх технологій. Проведено аналіз соціальних та психологічних факторів ризику з якими можуть стикатись студенти.

Наступним було обґрунтовано обране середовище розробки, технології та мови програмування для реалізації даного додатку. Розглянуто технічний аспект проблеми створення такого додатку. Обрано оптимальну методологію розробки програмного забезпечення. Змодельовано та спроєктовано архітектуру застосунку, створено діаграми варіантів використання та інші. Після чого застосунок було розроблено та протестовано. Потреби студентів грали визначну роль в процесі моделювання та розробки.

З точки зору наукової новизни цього дослідження можна визначити міжгалузевий підхід до дослідження, інтегруючи принципи психології, соціології, галузі освіти та інженерії програмного забезпечення для визначення ширшого розуміння проблеми та можливого її вирішення. Вивчення та поставлення в пріоритет студента та його потреб, методом прямого опитування, дає цілісне розуміння для адресації проблеми і потенційних рішень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд конкурентних додатків

Розглядаючи ринок освітніх технологій варто брати до уваги найбільш успішні додатки з широкою користувацькою базою. Одним з таких найбільших додатків, який має цікавий та унікальний підхід до навчання є ‘Quizlet’(рис. 1.1).

Рис. 1.1 — Логотип компанії конкурента Quizlet

Quizlet – це навчальний онлайн-додаток, який надає користувачам можливість вивчати різноманітні теми за допомогою навчальних інструментів та ігрових елементів. Заснована у 2005, ця платформа містить мільйони навчальних збірок, створених користувачами. Додаток є безкоштовним, з можливістю оформлення преміального місячного плану, який прибирає рекламу і надає більше просунутих функцій.

Розглянемо його основні функції та можливості, які він надає:

– Флеш-карти. Основою функціоналу Quizlet є цифрові флеш-карти(рис. 1.2). Користувачі можуть створювати власні флеш-карти або шукати створені іншими користувачами, вони можуть містити в собі зображення та текст.

– Режими навчання. Додаток пропонує різноманітні та адаптивні режими навчання за використання флеш-карт, від базових вивчення слів та їх

вимови до ігрових тестів на час.

– Навчальні збірки та плани. Користувачі можуть організовувати свої флеш-карти у збірки та папки, полегшуючи орієнтацію і керування ними.

– Співпраця між користувачами. Користувачі можуть поширювати флеш-карти одне з одним, створювати або приєднуватись до груп, які вивчають якусь одну тему, працюючи в колективі. Також в функціоналі є інтерактивні ігри, які мають можливість гри в групі.

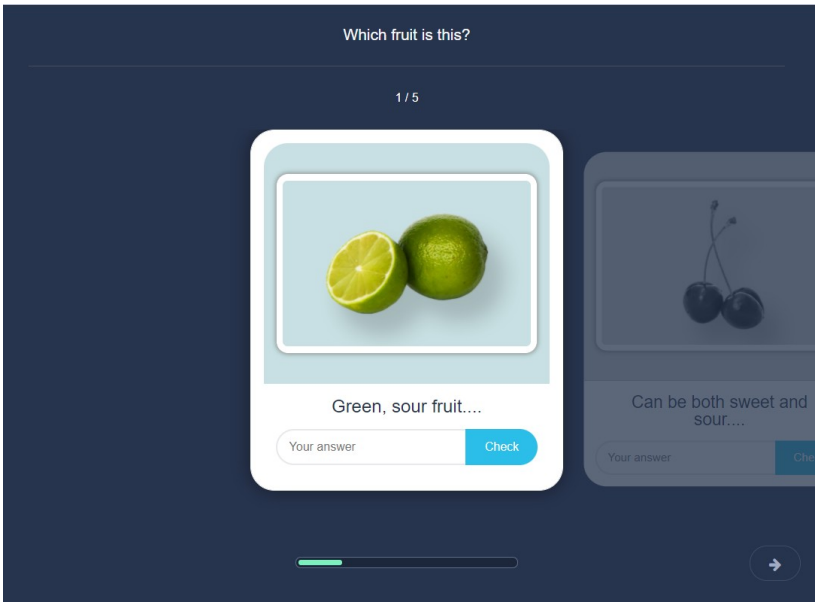

Рис. 1.2 — Приклад флеш-карти

Як сильні сторони цього додатку можна виділити: створений користувачами контент, різноманітні навчальні режими, нагадування, доступність, можливість співпраці користувачів. Завдяки своєму функціоналу Quizlet вирішує проблеми запам’ятовування інформації, індивідуального підходу до навчання, доступу до різноманітного вмісту, заохочення користувача. Серед слабких сторін варто зазначити, що покладання на створення контенту користувачами може призводити до невідповідності інформації та недостатнього контролю за її якістю. Також як недолік можна вказати обмеженість функцій безкоштовної версії.

У висновку, Quizlet є цікавим та успішним додатком у своїй ніші, він пропонує унікальні рішення існуючих освітніх проблем, але унікальність цих рішень обмежує його до простіших тем для навчання, що не є ефективним конкретного академічного навчання.

Також варто відмітити інший додаток, Evernote (рис. 1.3) , який прямо не відноситься до освітніх технологій, але дуже активно використовується здобувачами освіти, завдяки своєму зручному функціоналу.

Рис. 1.3 — Логотип Evernote

Evernote це універсальний додаток для нотаток та їх організації, основною ідеєю якого є допомога користувачам у записуванні ідей, створенні листів задач та організації проєктів. Запущений у 2008, за цей час, Evernote виріс до одного з найбільш популярних додатків у своїй категорії.

Його основними функціями можна визначити:

– Нотатки та нотатники. В своїй суті Evernote це додаток для нотаток. Користувачі можуть створювати текстові нотатки, добавляти зображення та вкладені документи, записувати аудіо нотатки. Ці нотатки можуть бути організовані в нотатники.

– Веб завантаження. Evernote дає змогу користувачам зберігати онлайн статті, файли та інший зміст мережі прямо в нотатники.

– Сканування документів. Використовуючи камеру свого пристрою, користувачі можуть сканувати свої документи та інші папери прямо в Evernote.

– Інтеграції з такими сервісами як: Google Drive, Microsoft Teams і Slack.

Загалом додаток є універсальною та зручною платформою, з можливістю якісно організувати інформацію, доступом до нотаток з будь якого пристрою, зручними інструментами праці з нотатками та можливістю групової праці. Сильною стороною є доступ на всіх платформах та можливості інтеграції з іншими сервісами. Попри це його мінусами є його цінова модель, де безкоштовна версія є доволі обмежена, а преміальна задорога, особливо для студентів. Також складний інтерфейс може відлякувати нових та недосвідчених користувачів, деякі з яких повідомляли про проблеми з роботою додатку.

1.2 Обґрунтування обраної предметної області

Задля обґрунтування обраної предметної області розглянуто такі аспекти як: попит на ринку та його ріст, потреби користувачів, розвиток відповідних технологій. Також було проведено опитування серед студентів на рахунок їх потреби у додатку для допомоги їм у організації навчання, проблем з якими вони стикаються під час навчання, їх особистих вподобань та стилів вивчення матеріалу і виконання практичних завдань.

Враховуючи тенденції цифровізації, яка невпинно зростає, відповідно збільшується частка ринку освітніх технологій. Станом на 2021 рік розмір ринку освітніх технологій становив 254.8 мільярдів доларів США і прогнозований ріст на 2027 рік до 605.4 мільярдів доларів США [1]. Цей ріст зумовлений поширеним впровадженням онлайн інструментів для навчання, мобільних додатків і веб ресурсів, серед навчальних закладів, викладачів та студентів. Виходячи з результатів опитування та проведеного дослідження, студенти мають потребу у

інструментах, які допоможуть їм в організації навчання, виробленні корисних звичок, збільшенні продуктивності, менеджменту часу та зниженні стресу, адже багато студентів мають складнощі з цими проблемами, що в свою чергу може мати вплив на їх академічну успішність та загальне благополуччя.

Також варто зазначити, що пандемія COVID-19 значно прискорила процес цифрової трансформації освіти як галузі [2]. Де дистанційне навчання, онлайн-ресурси та додатки, які допомагають у навчанні, стають його важливою та невід'ємною частиною. Тренди у розвитку освіти та технологій пов'язаних з нею, а також відповідна потреба студентів відображають актуальність дослідження та розвитку цієї галузі. Протягом останніх років галузь розробки мобільних застосунків зазнала значного розвитку. З використанням алгоритмів машинного навчання та нейромереж, додатки можуть надавати персоналізований підхід до навчання, визначати слабкі сторони студента та пропонувати стратегії щодо їх покращення.

Як важливий аспект варто відмітити, що галузь ринку освітніх технологій збільшує справедливу доступність освіти по світу. За допомогою таких технологій студенти, які живуть у різних соціоекономічних умовах, можуть отримати доступ до якісних освітніх ресурсів, що в свою чергу збільшить рівність умов та можливостей у суспільстві.

У висновку, враховуючи сучасні тренди, галузь розробки мобільних застосунків для допомоги студентам у навчанні є актуальною та перспективною з огляду на її ринкову частку та прогнозований ріст, на них є очевидний попит серед здобувачів освіти, адже вони адресують поширені проблеми з якими вони стикаються. Цей проєкт втілює цінності покращення якості освіти та її доступності, що робить його не тільки перспективно прибутковим, але й суспільно корисним. Беручи до уваги новітні технологічні здобутки цей проєкт зможе допомагати кожному здобувачу освіти на особистому рівні, що тільки збільшить його ефективність а й отже – актуальність.

1.3 Аналіз опитування серед студентів

В процесі дослідження предметної області, було проведено анонімне опитування серед студентів декількох закладів освіти, з метою визначення їх: особистих стилів навчання, незадоволених потреб, проблем та труднощів з якими вони стикаються, їх потреби та зацікавленості у додатку, який буде допомагати їм з організацією навчання; збору відгуків та пропозицій щодо проєкту та контактів для подальшої дистрибуції та потенційної співпраці [3].

Опитування складалось з 20 питань про:

– Вік;
– Освітній рівень, який ви зараз здобуваєте;
– Основний напрям навчання;
– Заклад освіти;
– Спеціальність;
– Кількість годин приділених навчанню на тиждень;
– Причини відзначеної кількості та її комфортність;
– Продуктивність протягом дня;
– Методи організації навчання;
– Використання додатків та інструментів які допомагають у навчанні;
– Складнощі та перешкоди у навчанні;
– Бажаний формат навчання (очний, змішаний, дистанційний);
– Роль ментального здоров'я респондента у процесі навчання;
– Зацікавленість у додатку для допомоги організації навчання;
– Функції, які були б корисні респондентам;
– Важливість персоналізації додатку, адресування ментального здоров'я;
– Коментарі, оцінка опитування, пропозиції щодо співпраці та розробки;

В опитуванні взяло участь 35 респондентів з різних закладів освіти, в основному це були студенти ТНТУ, але також були студенти ЗУНУ, ЗДМФУ,

ТНПУ, КНУ імені Т. Шевченка та ЛНУ імені І. Франка. Віком від 18 до 23 років, в своїй більшості бакалаври, розподіл між технічними, медичними та гуманітарними спеціальностями був здебільшого рівномірний з перевагою в бік технічних. .

Аналізуючи результати опитування було визначено, що абсолютна більшість студентів присвячує навчанню не більше 12 годин на тиждень(рис. 1.4), відповіді щодо зручності вказаного часу були змішані, більшості студентів вказаний час є комфортним, але деяким не достатній, багато з них вказують відсутність мотивації, бажання навчатися та цікавого матеріалу як основні причини цього. Частині студентів вказаний час не є ні комфортним, ні достатнім, вони відзначають велику кількість нагрузки та дисциплін, нестабільну ситуацію в країні. Більшість респондентів відзначила, що вони найбільш продуктивні зранку або ввечері.

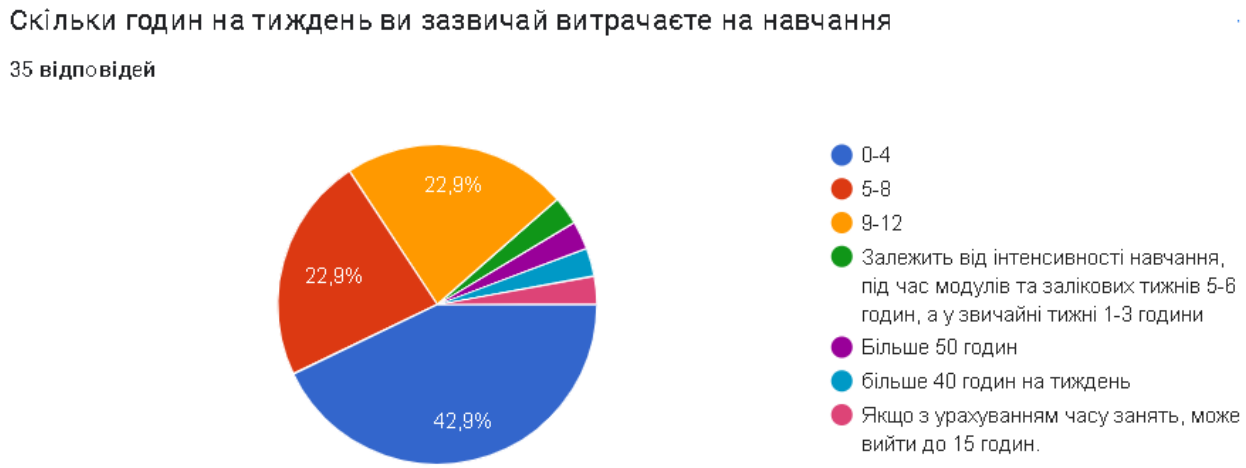

Рис. 1.4 — Статистика щодо присвячення часу на навчання

Переходячи до частини опитування про методи та додатки для допомоги в навчанні варто відмітити, що більшість опитаних використовує такі загальні методи як планувальники, списки задач та нагадування, будильники. Частина студентів відповіла, що в основному покладається на власну пам’ять. На питання,

про використання додатків і інструментів для допомоги в навчанні, відповіді майже порівну розділились між “Так” і “Ні”, де “Так” відповіли 57,1%(рис 1.5).

Рис. 1.5 — Співвідношення використання додатків та інструментів

Серед конкретно вказаних в наступних відповідях варто відзначити:

– Вузькоспеціалізовані додатки, інструменти та курси, за напрямом навчання респондентів;

– Платформи для проходження курсів, як Prometheus та Coursera;

– Таск менеджери, органайзери та платформи керування проєктами, колаборацією;

– Гіди на YouTube, PDF файли з матеріалом;

Серед унікальних відповідей варто зазначити додаток XMind, який надає користувачам можливість організовувати свої думки методами складання інтелект-мап, проведення мозкових штурмів та іншими способами. Також вартою уваги є вказана техніка Помодоро, хоч це і не конкретний інструмент чи додаток, це ефективна методика організації свого часу у виконанні завдань.

Під час проходження опитування, респонденти зазначили такі основні складнощі з навчанням як: відсутність мотивації, прокрастинацію, доступність та

зрозумілість матеріалу, кількість академічної нагрузки, проблеми з ментальним здоров’ям. Деякі студенти відмітили очне або дистанційне навчання як основну перепону з якою вони стикаються. На питання про вподобання формату навчання думки студентів розділились між очною, змішаною та дистанційною. Більшість студентів зазначила, що їх ментальне здоров'я відіграє важливу роль в процесі навчання, як основні проблеми були відмічені вигорання, тривога та депресивні стани. Абсолютна більшість студентів (85,7%) відзначила зацікавленість у потенційному додатку, який допоможе їм в організації навчання (рис. 1.6).

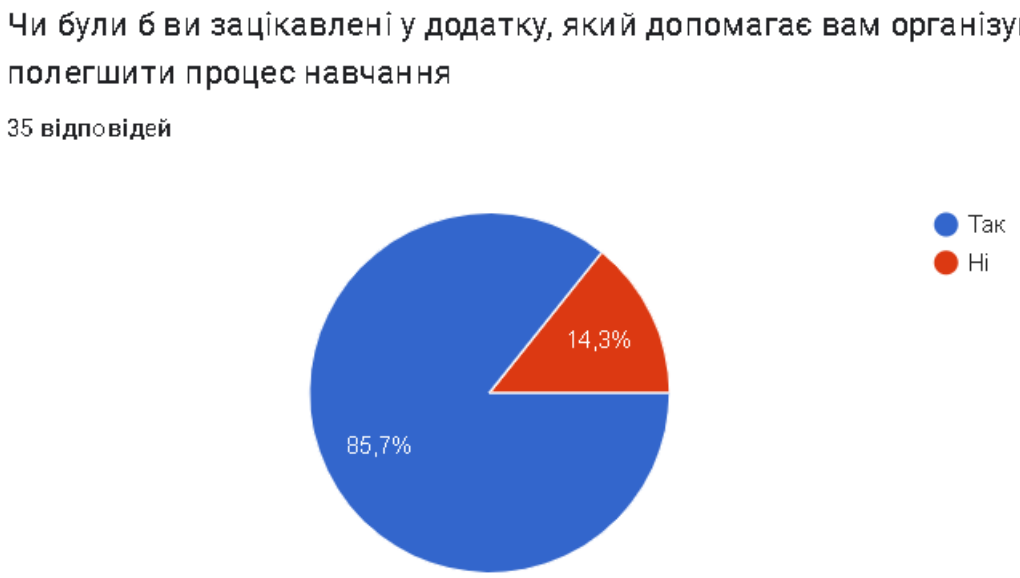

Рис. 1.6 — Відображення зацікавленості респондентів у предметі проєкту

Поміж запропонованих респондентам функцій додатку найбільшого попиту мали:

– Трекер прогресу;
– Менеджер завдань;
– Хаб ресурсів;
– Планувальник;
– Система нагадувань;

Серед цікавих пропозицій наданих респондентами варто зазначити функції створення ігрового процесу схожого з Quizlet, вимкнення сповіщень інших додатків для зниження відволікання на них, інтеграцію з сервісом курсів Prometheus. Усі респонденти відзначили, що можливість персоналізації додатку щодо їх вподобань, звичок та ментального здоров'я є принаймні мінімально важлива, більшість відзначила, що дуже важлива. Відгуки щодо опитування в основному є позитивними, респонденти в своїй більшості оцінили його від 7 до 10 балів по десятибальній шкалі.

Як висновок цього опитування зроблено підсумки, що через проблеми з якими студенти стикаються у академічному житті, в них є фактичний попит на користувач-дружній додаток, який був би їх помічником в навчанні та містив зручний інструментарій функцій для допомоги у навчанні. В них є потреба у підтримці та піклуванні про їх ментальне здоров’я, з чим міг би помогти такий додаток, способом надання ресурсів підтримки та контактів служб, заохоченням турботи про своє ментальне здоров’я.

1.4 Технічний аспект проблеми

Було розглянуто технічний аспект проблеми розробки додатку для допомоги студентам у організації їх навчання [4]. Як основну платформу для додатку було обрано мобільні телефони з огляду на такі плюси цієї платформи як:

– Доступність та зручність. Мобільний додаток дає можливість користувачу мати доступ до нього будь де, за умови наявності при собі смартфону, яка, враховуючи сучасність, справджується в більшості випадків. Ця зручність може призвести до частішого та більшого залучення до використання додатку.

– Push сповіщення. Вони є корисним інструментом для використання додатком розробки, ця функція буде мати використання у системі нагадувань щодо

майбутніх завдань, важливих дат та пропозиції ресурсів, які можуть зацікавити користувача. Це є ефективним інструментом підтримки організації та менеджменту часу студента.

– Персоналізація. Мобільні додатки надають користувачам легку та зручну можливість налаштування під свої вподобання та потреби. Враховуючи, визначену опитуванням, потребу студентів у цьому вибір мобільної платформи є очевидним, це дасть їм змогу отримати індивідуальний підхід використання додатку.

– Ріст популярності мобільної освіти [5]. Можливість вчитися “на ходу” та у власному зручному темпі, нині високо цінується серед здобувачів освіти. Беручи до уваги, що користувачі мобільних телефонів проводять більшість(88%) свого часу саме у мобільних застосунках, а не веб сторінках [6], пропозиція такого рішення зустрічає усі вимоги сучасного користувача.

Розглядаючи технічний аспект розробки мобільного додатку варто визначити такі ключові моменти як: інтерфейс користувача (UI), досвід користування (UX), керування даними, безпека, функціональність на різних платформах, робота у режимі офлайн, тестування.

Інтерфейс користувача має бути інтуїтивним та легким в користуванні, мінімізуючи час на його опанування користувачем. Дизайн повинен брати до уваги потреби користувача з пріоритетом на зручності та доступності. На додаток до інтуїтивного інтерфейсу, досвід користування додатком має бути однорідним та легкий. Це залежить від ефективності та логіки процесу роботи застосунку, швидкості роботи та його приємності використання. Враховуючи ці фактори в процесі проєктування та розробки варто взяти до уваги яким буде досвід користувача та дизайн інтерактивних елементів.

Керування даних включає в себе зберігання та обробку великої кількості інформації: профілів користувачів та їх налаштування, їх прогрес та інші метрики, потенційно велику кількість навчальних ресурсів, оптимальним рішенням цьому буде хмарне сховище, що дозволить забезпечити швидкий доступ до даних.

Забезпечення роботи у офлайн режимі за відсутності зв’язку з мережею, включає в себе локальне зберігання даних на пристрої, з наступною синхронізацією у хмарних сервісах, при підключенні до мережі. Це дасть користувачам працювати в додатку без втрати прогресу та доступних інструментів. Безпека даних користувачів має бути відповідно забезпечена за допомогою відповідних сервісів, з метою уникнення витоку даних та інших негативних наслідків. Проведене тестування повинно забезпечити функціональність та надійність усіх елементів застосунку, з максимальних охопленням якомога більше можливих сценаріїв використання та сценаріїв роботи, ситуацій не визначених під час розробки. Функціонування на всіх платформах дасть змогу охопити всю цільову аудиторію додатку, для цього варто використовувати крос-платформові фреймворки та додані інтерфейси програмування, це дасть змогу реалізувати додаток на всіх платформах без потреби повторення процесу розробки.

Не менш важливим аспектом є вибір відповідної моделі розробки програмного забезпечення, це дозволить максимально ефективно використати наявні ресурси та можливості розробки програмного продукту.

Адресування цих аспектів під час проєктування та розробки застосунку запевнить визначений пріоритет на забезпеченні користувача ефективним, дружнім та зручним рішенням для допомоги у організації навчання.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Обґрунтування вибору технологій розробки

За результатом аналізу предметної області та технічних аспектів проєкту було обрано простий та зручний набір інструментів та методів розробки, які дозволять забезпечити швидку та якісну розробку продукту. Основний стек технологій складається з: JavaScript, ReactNative, Firebase, JetBrains WebStorm, Jest, Expo. Методологією розробки був обраний Kanban.

ReactNative це фреймворк мобільних додатків, він має можливість реалізації на усіх платформах, що дає змогу написати код один раз та розгорнути проєкт на всіх системах, що в свою чергу суттєво зменшує час та витрати на розробку. Його структура, що базується на компонентах, дає можливість розробникам створювати гнучкі застосунки у веб-стилі, не використовуючи веб технології, на відміну від інших гібридних фреймворків.

Firebase це платформа бекенду-як-сервісу (BaaS). Вона надає сервіси готових базових рішень як автентифікація користувачів та зберігання даних. Це забирає потребу створення з нуля інфраструктури, серверів та їх програмного забезпечення, що дає можливість розробнику зосередитись на створенні унікальних та функціональних елементів додатку та прискорити його розгортання. Використання сервісу Firestore, NoSQL бази даних на основі моделі документів, дає змогу з легкістю зберігати, синхронізувати та шукати дані додатку на глобальному рівні, як сервіс Firebase він легко інтегрується з у систему та обновляється в реальному часі. Сервіс Firebase Authentication надає сервіс аутентифікації користувачів за допомогою паролю, номеру телефону і популярних соціальних мереж як Google та Facebook. Він також надає функції створення облікового запису, скидання паролю та підтвердження електронної пошти та інші. Це забезпечує облегшення процесу розробки та гарантує надійність збереження даних користувача. Також сервіси Firebase надають рішення для роботи

застосунку у режимі офлайн, що також усуває цю проблему з боку розробки.

Тестування за допомогою фреймворку Jest, популярного серед ReactNative проєктів, надає змогу проводити швидкі паралельні тести, “знімки” тестування, що дозволяє порівнювати зміни графічного інтерфейсу протягом часу. Також його зручними рисами є мінімальна кількість попередньої, до використання, установки, його доданий інтерфейс програмування є повноцінним та легким в використанні, що робить його оптимальним рішенням для цього проєкту.

Аналізувавши наявні методології розробки програмного забезпечення та визначивши загальні вимоги до додатку, було прийнято рішення використати методологію з сімейства Agile, як таку, що бере до уваги швидку та непередбачувану зміну умов та ситуацій. Ставлячи в пріоритет користувача та його взаємодію та робоче програмне забезпечення. Agile є відповідним сімейством методологій, яка забезпечить швидку та ефективну розробку застосунку. Як конкретну методологію обрано Канбан, це обґрунтовано такими його рисами як:

– Гнучкість. На відміну від інших методологій сімейства, Канбан не вимагає розбиття роботи на спринти, в результаті чого, якщо виникне нагальна потреба виконання якогось завдання, до його виконання можливо приступити одразу, не чекаючи наступного спринту.

– Візуалізація робочого процесу. Дошка Канбан це зручний інструмент візуалізації робочого процесу. Це дозволяє розробникам мати чітке розуміння поточного стану проєкту та потенційні проблеми в майбутньому. Це дозволяє бачити стани як і всього проєкту, так і окремих його частин. Візуалізація також часто допомагає з кращим розумінням проєкту в цілому.

– Обмеження кількості роботи. Це дозволяє концентруватись на поточних завданнях, завіряючи їх швидке виконання та уникаючи одночасної роботи над декількома завданнями.

– Зменшення кількості планування, оцінки та звітності. Основним пріоритетом є кінцеве виконання завдань.

– Динамічна пріотеризація. Завдяки постійному робочому процесу можливо переглянути пріоритет та важливість завдань у реальному часі.

Як інтегроване середовище розробки було обрано JetBrains WebStorm, з огляду на такі його риси як:

– Ефективність та продуктивність. WebStorm розроблений з метою покращення продуктивності розробника. Завдяки функціям закінчення коду, глибокого його аналізу, швидкої навігації та допомоги з аналізом та вирішенням помилок.

– Інтегровані інструменти. Середовище містить в собі широкий набір інструментів як вбудований термінал, бази даних, системи контролю версій та інші інструменти, які усувають потребу в перемиканні між додатками.

– Плагіни. JetBrains WebStorm підтримує велике різноманіття плагінів, що дозволяє добавляти додатковий функціонал, опираючись на вимоги проєкту.

В контексті розробки застосунку для допомоги студентам у організації навчання обрані технології при застосуванні разом дають змогу побудувати надійний, масштабований та безпечний мобільний застосунок. У якому ReactNative буде відповідальний за створення безшовного інтерфейсу користувача. Firebase, разом з Firestore та Firebase Authentication будуть проводити операції з серверного боку, зберігати дані і аутентифікувати користувачів. Jest забезпечить тестування додатку, зменшуючи кількість помилок. Використання методології Канбан збігається з потребами проєкту у гнучкості, ефективному робочому процесі та адаптивністю до змін, що знижує ризик затримок та зайвої роботи. Середовище розробки WebStorm надасть великий набір інструментів для ефективної реалізації проєкту. Ця комбінація рішень, технологій та інструментів дає змогу концентруватися на побудові та дизайні застосунку та його основних функціоналі, уникаючи довгого та затяжного процесу постанови та менеджменту власної інфраструктури.

2.2 Розробка моделі предметної області

Під час процесу моделювання системи було визначено основний функціонал системи, який буде слугувати підґрунтям для створення варіантів використання та їх діаграми. Цей функціонал визначено як:

– Можливість реєстрації та входження до облікового запису. Першою взаємодією користувача з додатком буде процес реєстрації та/або входу, Щоб отримати доступ до інших функцій додатку користувач має створити обліковий запис. Для реєстрації користувачу буде потрібно вписати таку інформацію як ім’я, електронну пошту, надійний пароль, або просто зареєструватись за допомогою наявного облікового запису інших довірених ресурсів таких як Google. Зареєстровані користувачі будуть мати змогу увійти в додаток використовуючи свої облікові дані.

– Панель користувача. Увійшовши в додаток користувач буде направлений до панелі користувача, яка буде слугувати центральною точкою для навігації до інших частин додатку. Панель користувача буде відображати важливу інформацію з першого погляду, таку як завдання, що наближаються у часі, звіт прогресу та інше.

– Доступ до навчальних матеріалів. Користувачі матимуть змогу направлятися до секції, де вони зможуть переглядати або шукати доступні навчальні матеріали. Вони зможуть читати ці матеріали у самому додатку або будуть перенаправлені на зовнішнє джерело (веб переглядач, інший додаток, тощо).

– Менеджер завдань. Користувачі матимуть можливість створювати, переглядати, доповнювати та видаляти завдання, які пов’язані з їх навчанням і не тільки. Вони зможуть ставити нагадування для цих завдань, які будуть повідомляти їх у відповідний час.

– Трекер прогресу. По мірі того як користувачі завершують завдання і

просуваються у вивченні обраного матеріалу, додаток буде автоматично відстежувати їх прогрес. Це може бути зображено у вигляді графіків або інших візуальних елементів, надаючи користувачу можливість легко побачити кількість опрацьованого матеріалу та скільки ще залишилось.

– Сповіщення. Користувачі зможуть отримувати сповіщення про важливі події як: завдання, що наближаються у часі, появу нових навчальних матеріалів у доступі, тощо. Ці сповіщення можуть з’являтися у додатку, або ж бути інтегрованими у рідну систему сповіщень пристрою, щоб оповіщати користувачів, коли вони не перебувають у додатку.

– Налаштування додатку та профілю користувача. Користувачі будуть здатні керувати налаштуваннями додатку та власного профілю. Наприклад змінювати своє ім’я, електронну пошту, пароль, налаштування сповіщень, змінювати тему додатку та інше.

Варто зазначити, що цей функціонал повинен мати дружній, до користувача, інтерфейс, забезпечуючи користувачів всіх рівнів технічної грамотності можливістю використовувати додаток ефективно. Головним завданням є вироблення процесів навчання і відстеження академічного успіху у максимально безшовні та організовані, для користувачів.

Важливим елементом моделювання програмного забезпечення є відображення специфікацій та високорівневої архітектури додатку за допомогою UML діаграм. Ці діаграми дають змогу більш чітко та детально описати модель програмного забезпечення та взаємодію елементів системи [7]. Для цього спершу було створену діаграму варіантів використання (рис.2.1).

Ця діаграма містить таких акторів як “Користувач” та “Адміністратор”. Адміністратор стоїть за створенням та підтримкою системи у робочому стані, він має необмежений доступ до елементів системи та може змінювати їх стан, модифікувати їх та добавляти нові. Користувач системи взаємодіє з нею через графічний інтерфейс користувача, входить в систему за допомогою аутентифікації.

Відображені на діаграмі варіанти використання як “Навчальні ресурси”,

“Менеджер завдань”, “Трекер прогресу” та “Аутентифікація користувача”. Вони відображають відповідні елементи системи та взаємодію акторів та елементів системи на найвищому абстрактному рівні.

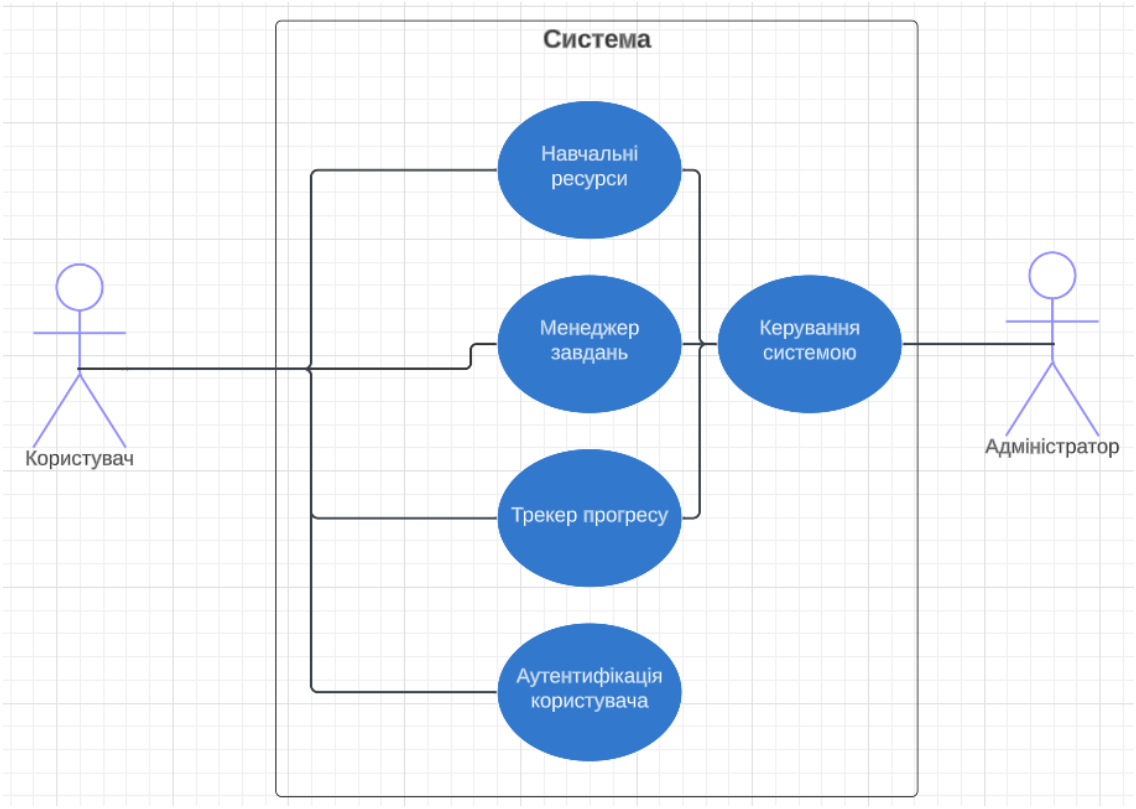

Рис. 2.1 — Діаграма варіантів використання системи

Прикладом використання додатку можна описати сценарій створення та керування завданням.

Сценарій “Створення та керування завданням”:

– Користувач входить в систему за допомогою автентифікації користувача;

– Користувача переносить на його основну панель керування;

– Користувач переходить до секції “Завдання”;

– Користувач обирає опцію “Створити нове завдання”;

– Перед користувачем відкривається форма заповнення деталей

завдання (назва, опис, дедлайн);

– Користувач заповнює форму і натискає “Створити”;

– Менеджер завдань записує завдання у базу даних та відповідно відображає завдання у графічному інтерфейсі, система встановлює нагадування щодо дедлайну;

– По мірі просування по завданню, користувач може змінювати його статус (не почате, в процесі, завершене) та деталі завдання за потреби;

– По завершенню завдання, воно отримує позначку “Завершене” в листі завдань і прогрес відображається в графіку прогресу на панелі користувача;

Цей сценарій відображає як і взаємодію користувача і системи так і самих елементів системи між собою.

Задля створення діаграми станів описано сценарій взаємодії користувача і додатку “Авторизація користувача”, де описується взаємодія користувача та системи під час його авторизації до системи. Сценарій складається з:

– Користувач відкриває додаток;

– Він бачить вікно привітання користувача та опції входу до системи або реєстрації облікового запису;

– Якщо користувач вже зареєстрований:

– Він вводить свої облікові дані в відповідні рядки;

– Система перевіряє відповідність введених даних;

– Якщо дані введені некоректно то користувачу пропонується ще раз вписати дані;

– Якщо дані введені коректно то користувач успішно авторизується до системи;

– Якщо користувач не зареєстрований:

– Користувача переносить до вікна реєстрації де пропонується створити обліковий запис;

– Користувач вводить дані, потрібні для реєстрації;

– Система перевіряє правильність введення даних;

– Якщо дані введені неправильно то користувачу пропонується ще раз вписати дані;

– Якщо дані введені правильно то система створює новий обліковий запис користувача і повертає його на сторінку входу до системи;

Для візуалізації було створено діаграму станів для цього сценарію (рис. 2.2). У ній зображено стани через, які проходить система під час авторизації користувача, як “Вікно авторизації”, “Введення даних користувача”, “Авторизація в систему”, тригери перевірки правильності введення даних та інше. Це дозволяє наочно побачити цей процес з боку взаємодії системи та користувача.

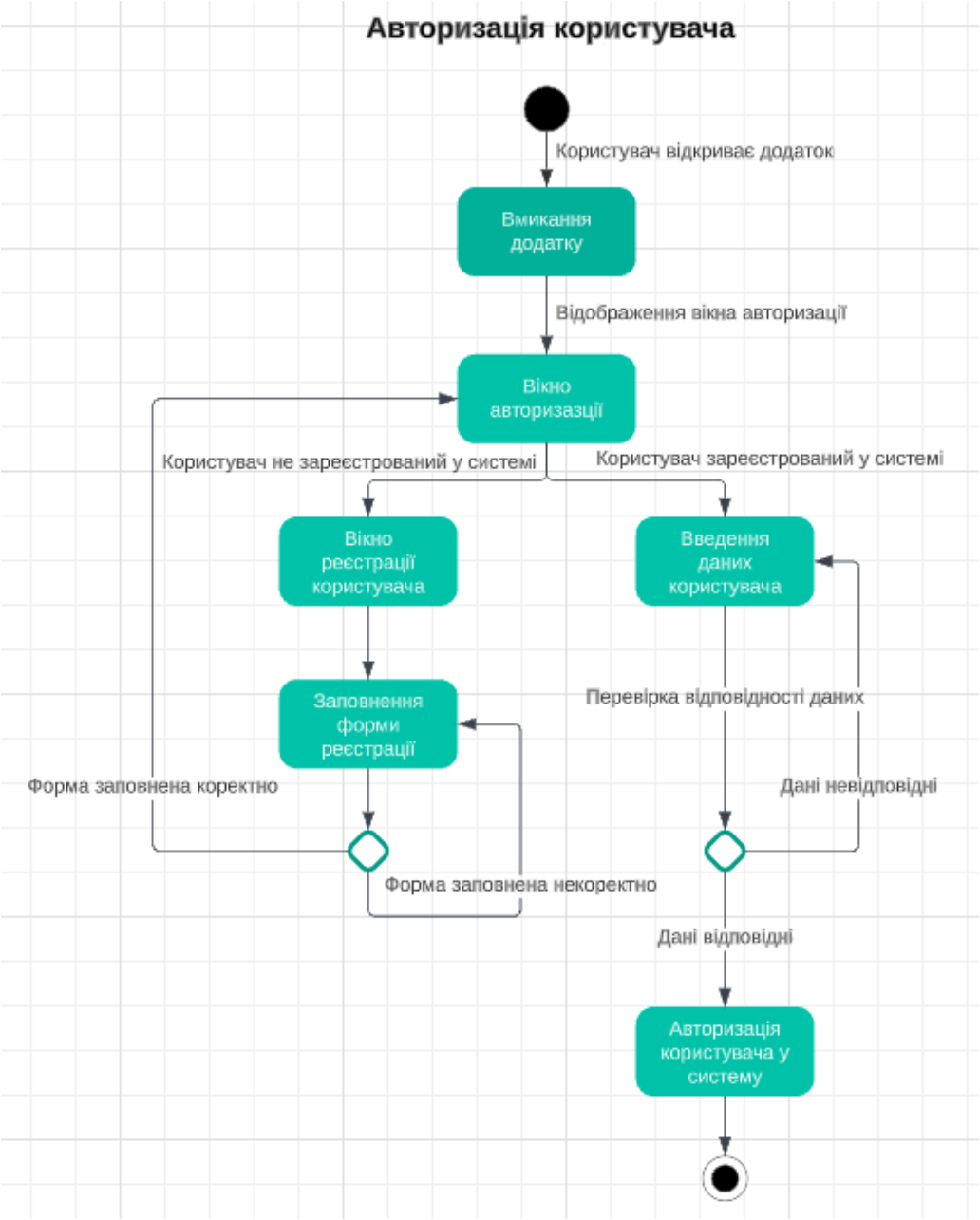

Рис. 2.2 — Діаграма станів системи авторизації користувача

Для того, щоб відобразити потік повідомлень між об’єктами системи та їх життєвий цикл було створено діаграму послідовностей для сценарію “Взаємодія користувача зі сповіщеннями та менеджером завдань” (рис. 2.3). Це дасть змогу краще зрозуміти та імплементувати функціонал системи методом графічного відображення взаємодії компонентів системи між собою під час виконання конкретного завдання. Це також дає змогу визначити роль елементів в різних умовах та визначити контроль потоку та логіки системи.

Сценарій “Взаємодія користувача з сповіщеннями і менеджером завдань”:

– Додаток, в постійному режимі, отримує від пристрою дані про поточну системну дату та час.

– Додаток надає цю інформацію менеджеру завдань, який визначає, що в завдання підходить строк дедлайну та ініціює сповіщення користувача.

– Додаток передає це повідомлення пристрою;

– Пристрій ініціює звукове та візуальне сповіщення;

– Користувач входить в додаток та авторизується;

– Користувач вибирає вкладку “Менеджер завдань”;

– Користувач змінює статус завдання на “у виконанні”;

– Система опрацьовує ці зміни та відображає їх користувачу;

Цей сценарій вносить чіткості в розуміння співпраці як і всередині системи та і поза нею з пристроєм та користувачем.

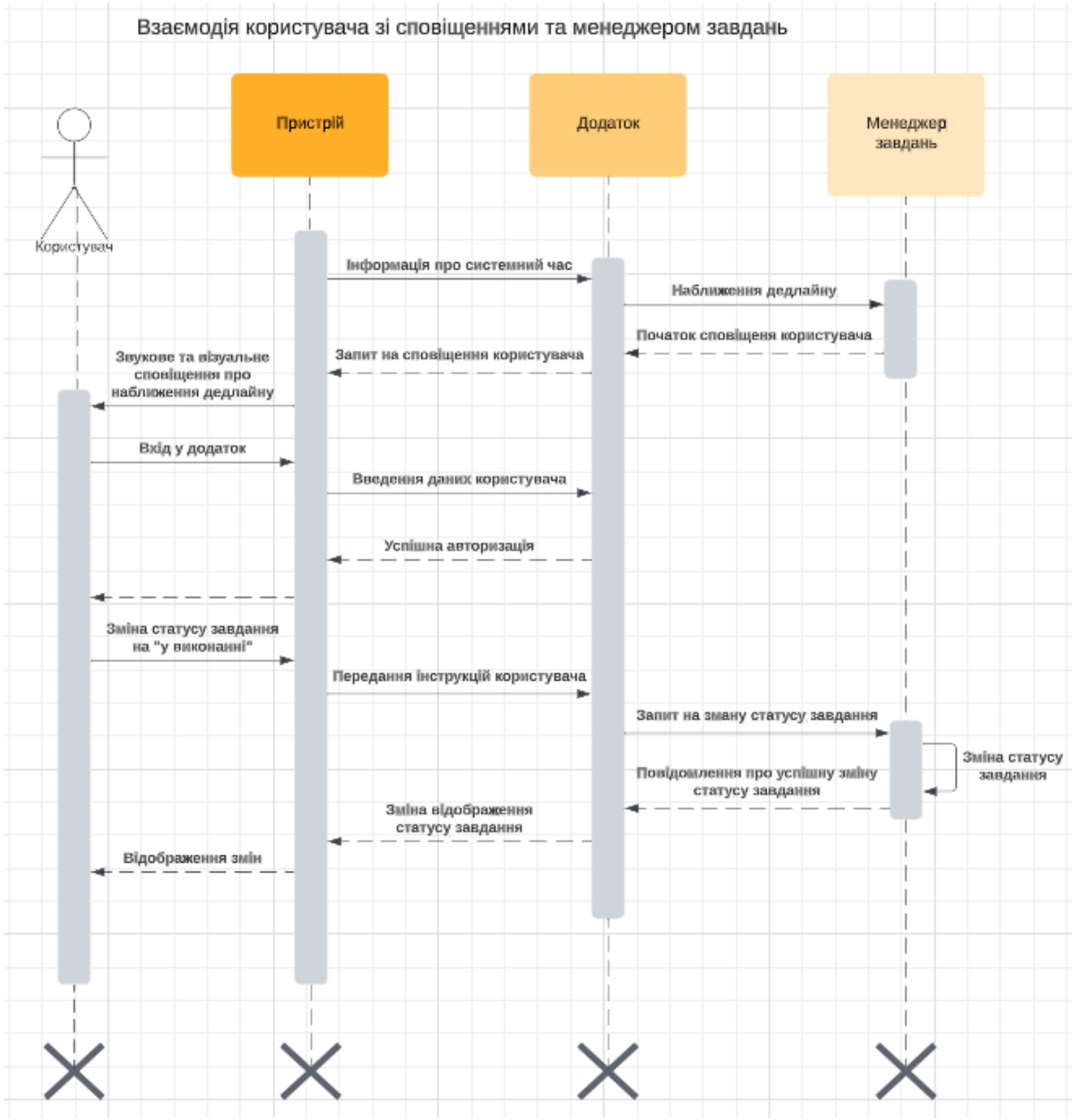

Рис. 2.3 — Діаграма послідовностей сценарію взаємодії користувача зі сповіщеннями та менеджером завдань

2.3 Проєктування архітектури

Для ефективної реалізації архітектури додатку було створено діаграму архітектури системи (рис. 2.4). Вона відображає основні елементи структури та їх взаємодію між собою. Собою вона представляє варіант архітектури мікросервісів.

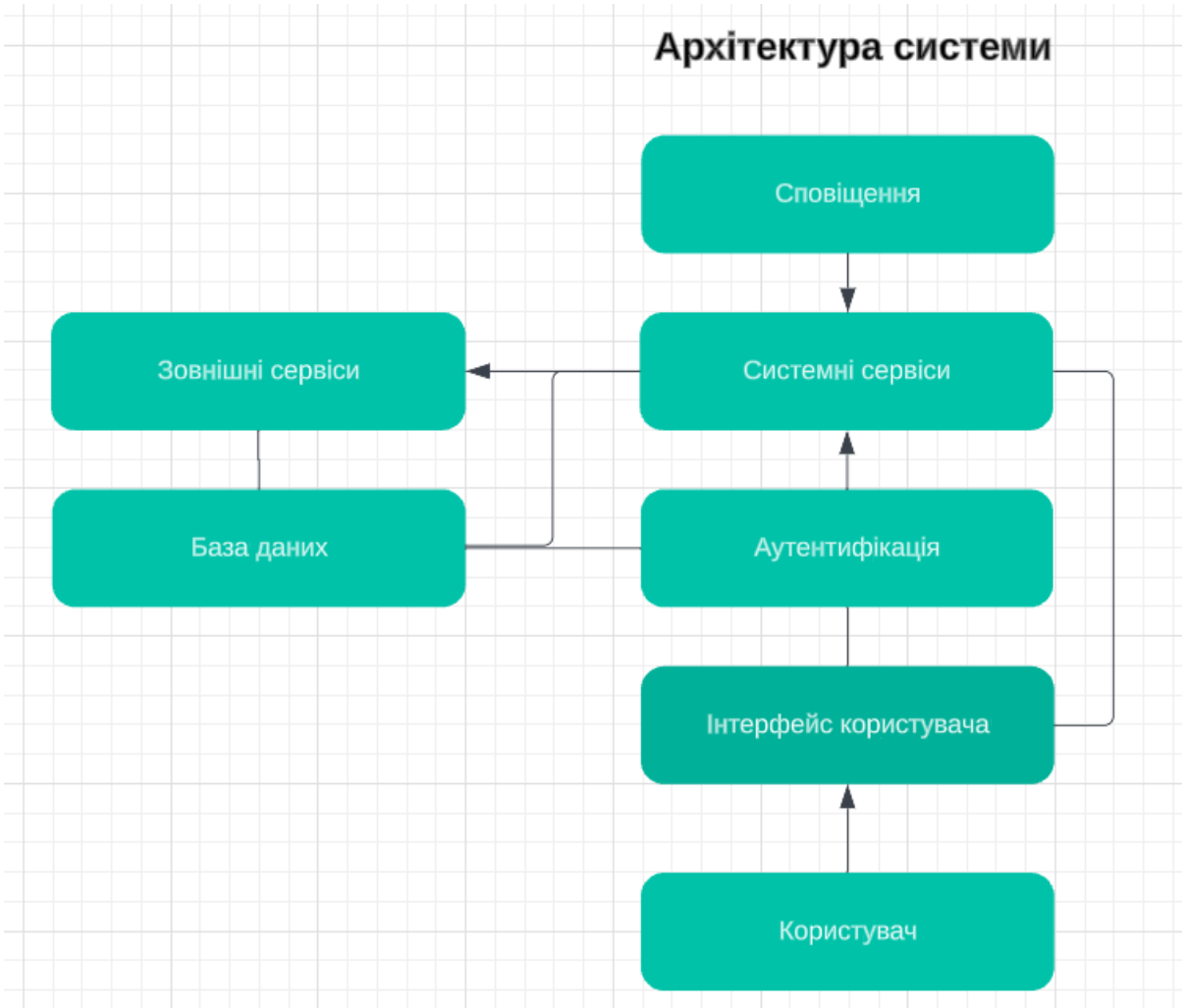

Рис. 2.4 — Діаграма архітектури системи

В цій діаграмі зображено основні елементи архітектури системи та як вони взаємодіють між собою, користувачем та зовнішніми сервісами.

– Інтерфейс користувача: Це те, з чим будуть взаємодіяти користувачі.

Він складається з екранів, кнопок, полів введення та всіх інших компонентів інтерфейсу. Він буде побудований за допомогою React Native і буде взаємодіяти з системними сервісами для отримання даних або виконання дій.

– Системні сервіси слугують мостом між рівнем інтерфейсу користувача та рівнем даних. Він взаємодіє з інтерфейсом користувача для отримання користувацьких запитів, обробляє ці запити, а потім взаємодіє з базою даних для отримання або маніпулювання даними. Результати потім надсилаються назад на рівень інтерфейсу користувача для відображення користувачеві.

– База даних. Це місце, де зберігаються та отримуються дані. Цей рівень буде побудований з використанням бази даних Firebase Firestore. Він взаємодіє з системними сервісами, виконуючи запити для отримання даних або оновлення даних на основі отриманих запитів.

– Аутентифікація. Це сервіс, де відбувається аутентифікація користувача. Зазвичай він є частиною системного рівня, але через його важливість його зображено як окремий рівень. Для цієї частини буде використовуватися аутентифікація Firebase Authentication.

– Служба сповіщень. Це ще одна сервіс, який може бути частиною системного рівня, але через свою особливу роль вона часто представлена як окремий компонент. Саме тут здійснюється управління push-сповіщеннями. Для цього можна використовувати Firebase Cloud Messaging (FCM).

– Зовнішні сервіси: Це інші сервіси, з якими може взаємодіяти додаток. Для отримання даних від стороннього постачальника піл час використання доданих програмних інтерфейсів.

3 КОНСТРУЮВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Реалізація ключових елементів

Першим етапом розробки було налаштування середовища розробки(рис 3.1). WebStorm був обраним інтегрованим середовищем розробки. Для мобільної розробки ми використовували фреймворк Expo у поєднанні з ReactNative. Ця комбінація надає потужну та гнучку платформу для створення додатку, що дозволило скористатися перевагами різних бібліотек та інструментів для оптимізації процесу розробки.

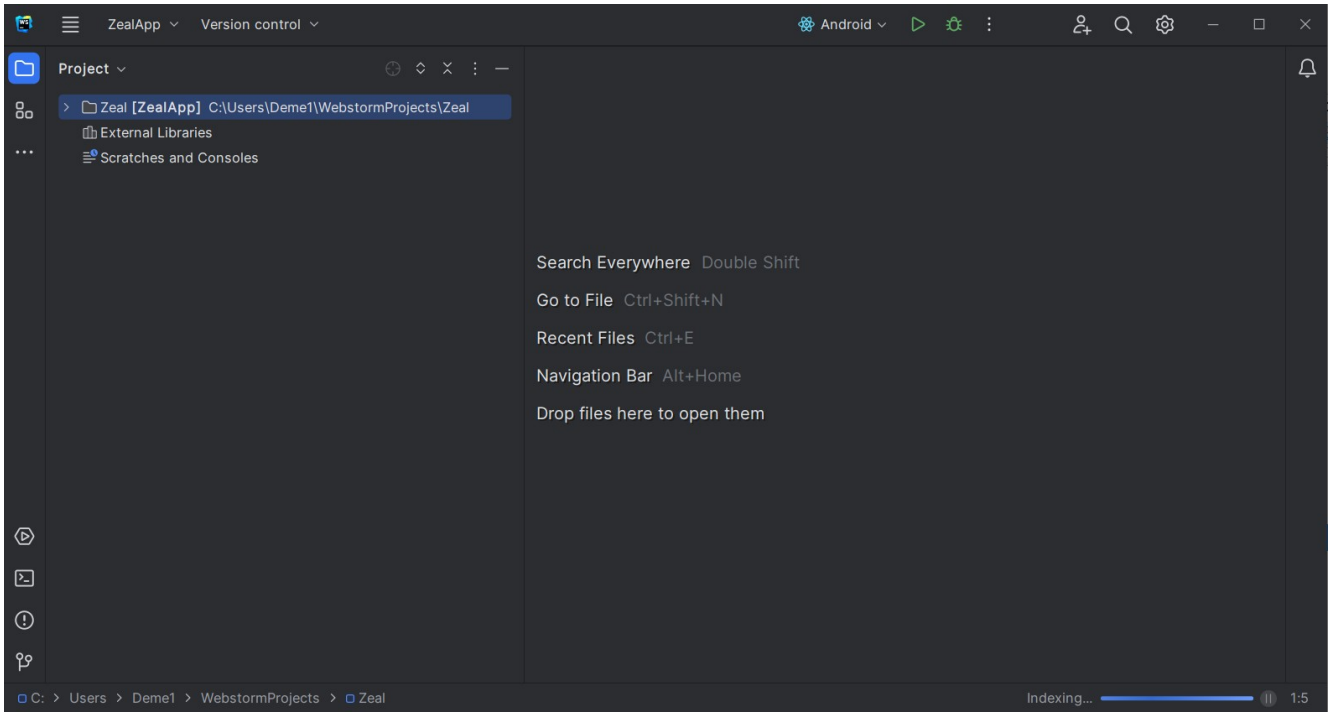

Рис. 3.1 — Середовище розробки WebStorm

Після налаштування середовища, було ініціалізовано проєкт ReactNative за допомогою Expo. Це автоматично створило базову структуру проекту та встановило необхідні бібліотеки для початку розробки.

Це було зроблено за наступних кроків:

– Встановлено Expo CLI. Expo CLI - це інструмент командного рядка, який використано для створення ReactNative проєкту. Його встановлено за допомогою npm (який встановлюється разом з Node.js).

– Створено новий нативний проект React: Після завершення встановлення створено новий проект React Native.

– За допомогою Expo запущено сервер розробки на якому можна запустити додаток.

– Створено файл App.js, який буде відправною точною створення проєкту.

Далі було створено базовий навігатор додатку за допомогою React Navigation. React Navigation - це широко використовувана бібліотека в React Native для реалізації навігації. Вона підтримує різні типи навігації, включаючи навігацію стеком та навігацію вкладками, які є найпоширенішими. Реалізація цього елементу наведена у лістингу 3.1.

Лістинг 3.1

```
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

const App = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Details" component={DetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
export default App;
```

Наступним кроком була розробка та реалізація окремих екранів додатку. На головному екрані відображається список курсів або предметів. Це прокручуваний список, кожен елемент якого показує деякі ключові деталі про курс/предмет. Вигляд цього компоненту зображено у Лістингу 3.2.

Лістинг 3.2

```
import React, { useEffect, useState } from 'react';
import { View, Text, FlatList } from 'react-native';
const HomeScreen = () => {
  const [courses, setCourses] = useState([]);
  useEffect(() => {
      setCourses([{ id: 1, title: 'Course 1' }, { id: 2, title: 'Course
2' }]);
  }, []);
  return (
    <View>
      <FlatList
        data={courses}
        keyExtractor={(item) => item.id.toString()}
        renderItem={({ item }) => (
          <View>
            <Text>{item.title}</Text>
          </View>
        )}
      />
    </View>
  );
};
export default HomeScreen;
```

Кожен інший екран реалізовано подібним чином, з макетом і функціональністю, що залежать від завдань окремого екрану. Кожен екран був розроблений з урахуванням потреб користувача, забезпечуючи інтуїтивно зрозумілий інтерфейс, який дозволяє легко виконати поставлені завдання.

Після цього було реалізовано функціональні модулі додатку.

Першою функціональною частиною додатку, є модуль аутентифікації. Використовуючи Firebase Authentication, було забезпечено безпечну реєстрацію користувачів, вхід та скидання паролю. Для спрощення процесу та збільшення кількості варіантів авторизації було використано React Native SDK від Firebase. Його реалізація зображена у лістингу 3.3.

Лістинг 3.3

```
import React, { useState } from 'react';
import { View, TextInput, Button, Alert } from 'react-native';
import auth from '@react-native-firebase/auth';
const SignIn = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const signIn = async () => {
    try {
      await auth().signInWithEmailAndPassword(email, password);
    } catch (error) {
      Alert.alert('Sign In Error', error.message);
    }
  };
  return (
    <View>
      <TextInput
        value={email}
        onChangeText={setEmail}
        placeholder="Email"
        autoCompleteType="email"
        keyboardType="email-address"
        textContentType="emailAddress"
      />
      <TextInput
        value={password}
        onChangeText={setPassword}
        placeholder="Password"
        secureTextEntry
        textContentType="password"
      />
      <Button title="Sign In" onPress={signIn} />
    </View>
  );
};
export default SignIn;
```

У цьому лістингу signInWithEmailAndPassword використовується для входу користувачів, які були створені за допомогою відповідного методу createUserWithEmailAndPassword.

Наступним було розроблено модуль завдань, щоб користувачі могли додавати та керувати своїми навчальними завданнями. Де кожне завдання є об'єктом з певними властивостями і зберігається у Firestore, лістинг коду вказаний у додатку Б. Модуль нагадувань було створено для надсилання сповіщень користувачам про майбутні навчальні сесії або завдання. Було використано бібліотеку react-native-push-notification для планування та управління локальними сповіщеннями. Реалізація цього модуля вказана у лістингу 3.4

Лістинг 3.4

```
import React, { useEffect } from 'react';
import { View, Text, Button } from 'react-native';
import NotifService from './NotifService';
const Task = ({ task, toggleCompletion }) => {
  const notif = new NotifService();
  useEffect(() => {
    notif.configure();
  }, []);
  const scheduleNotification = () => {
    const date = new Date(Date.now() + DefinedDate);
      notif.scheduleNotif('Task Reminder', 'You have a task coming up!',
date);
  };
  return (
    <View>
      <Text>{task.text}</Text>
      <Text>{task.completed ? 'Completed' : 'Not Completed'}</Text>
                  <Button title="Toggle Completion" onPress={() =>
toggleCompletion(task)} />
      <Button title="Remind Me" onPress={scheduleNotification} />
    </View>
  );
};

export default Task
```

Всі ці модулі було розроблено ітеративно, щоб переконатися, що вони функціонують належним чином, перш ніж переходити до наступного модуля. Для цього було використано функцію гарячого перезавантаження ReactNative.

Під час розробки було використано такі патерни як:

– Модель-Вид-Контролер (MVC): Цей патерн розділяє додаток на три взаємопов'язані компоненти, що дозволяє ефективно організувати код і розділити завдання.

– Паттерн "Спостерігач": Цей патерн було використано для створення моделі підписки на повідомлення про зміни в стані додатку. Використано бібліотеку Redux в екосистемі React Native.

3.2 Розробка графічного інтерфейсу користувача

Загальний дизайн додатку повинен бути мінімалістичним і зручним для навігації. Для цього було використано чистий і простий макет з достатньою кількістю пробілів. Обрано плаский дизайн, який включає прості елементи, яскраві кольори. Використано бібліотеки ReactNative React Native Elements та NativeBase. Задля простоти у використанні додатку було реалізовано покроковий процес адаптації, який допоможе новим користувачам ознайомитися з основними функціями додатку. Для цього використано бібліотеку ReactNative "react-native-onboarding-swiper". Створено центральний хаб для ресурсів за допомогою макету на основі вкладок, де кожна вкладка представляє окремий тип ресурсу (наприклад, відео, статті, книги і т.д.). Використано такий компонент, як "react-navigation-tabs". Для відстеження користувачами прогресу використано візуальні представлення, такі як індикатори прогресу та діаграми. Для цього використано бібліотеки "react-native-progress" тп "react-native-svg-charts". За допомогою бібліотеки ‘react-native-push-notification' організовано локальні сповіщення. Ця бібліотека дозволяє планувати сповіщення, які можна використовувати для нагадувань. Використано "react-native-dark-mode" для реалізації темної теми у додатку. Ця бібліотека надає хуки та компоненти, які допоможуть керувати темним режимом у ReactNative додатку.

Рис. 3.2 — Зображення прототипу користувацького інтерфейсу

3.3 Тестування програмного забезпечення та оцінка якості

Тестування додатку було проведено за допомогою фреймворку Jest. Jest - це фреймворк для тестування JavaScript, який має хорошу сумісність з ReactNative. Було встановлено Jest в інтегроване середовище програмування та визначення його типів, також було встановлено React Testing Library, що дозволяє рендерити компоненти React в об’єкти JavaScript без використання нативного мобільного середовища.

Було створено файл jest.config.js в кореневій директорії проєкту, у лістингу 3.5 наведено конфігурацію цього файлу.

Лістинг 3.5

```
module.exports = {
  preset: 'react-native',
  setupFilesAfterEnv: ['@testing-library/jest-native/extend-expect'],
};
```

Після цього було проведено різноманітні юніт-тести для компонентів додатку з визначенням проблемних зон та виправленням помилок в коді. Розширені результати тестування наведено у додатках.

3.4 Результати розробки, оцінка перспективності додатку

Результати розробки показали що додаток є багатофункціональним інструментом для самоорганізації та самонавчання, який забезпечує користувачам гнучкий і персоналізований досвід. Застосунок допомагає користувачам планувати свій навчальний процес, відстежувати свій прогрес, а також знаходити натхнення та мотивацію для подальшого навчання.

Додаток був розроблений з використанням сучасних технологій, що забезпечують гнучкість, масштабованість та високу продуктивність. Додаток базується на архітектурі microservices з використанням технологій ReactNative та Firebase. Це дозволило нам створити надійний, масштабований додаток з гнучкими налаштуваннями та можливістю легко додавати нові функції.

З огляду на розвиток ситуації з самоосвітою в світі, що перспективи цього проєкту є багатообіцяними. У даний час все більше людей віддають перевагу самостійному навчанню, що включає онлайн-курси, вебінари, тощо. Додаток допомагає користувачам упорядкувати цей процес, створюючи індивідуальний навчальний план, відстежуючи прогрес і допомагаючи знаходити однодумців для

колективного навчання.

Планується подальша підтримка та вдосконалення додатку. Серед можливих напрямків розвитку - інтеграція з іншими платформами для онлайн-навчання, введення функції гейміфікації для стимулювання мотивації користувачів, розробка додаткових інструментів для роботи в команді і додання ресурсів з підтримки ментального здоров’я, контактів відповідних служб допомоги, методик та ресурсів щодо пізнання себе та своїх потреб.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Соціальні та психологічні фактори ризику

Однією з головних причин створення цього проєкту є проблема масового погіршення ментального здоров'я здобувачів освіти, що в свою чергу зменшує їх академічну продуктивність. Тому варто взяти до уваги та розглянути соціальні та психологічні фактори ризику, з якими стикаються ті, хто здобуває освіту, з боку охорони праці та безпеки життєдіяльності, нинішніх норм, практик та регулювальних документів. З огляду на те, що інститут ментального здоров'я з боку охорони праці та безпеки життєдіяльності в Україні швидше розвивається на локальному рівні ніж на систематичному, дослідження базується на особистому вивченні цієї теми та проведеному раніше опитуванню серед студентів [3].

Для початку варто розглянути, що таке соціальні та психологічні фактори ризику, які з них найпоширеніші та які загрози вони несуть студентам навчальних закладів

Соціальні та психологічні фактори ризику відносять до таких елементів середовища людини, які несуть загрозу її ментальному, емоційному та фізичному здоров'ю. Зв'язок між ментальним здоров'ям та соціальними обставинами відображає вплив численних особистих, соціальних та середовищних умов на благополуччя та безпеку людини. Основним чинником цих ризиків є стрес – фізіологічна та психологічна реакція на ситуації та події, які перевищують здатність людини з ними справитись. Ці фактори зазвичай відносяться до таких сфер життя людини як: праця, сім'я, соціальні стосунки, фінансовий стан, суттєві життєві події і т.д. Загалом ці фактори ризику містять в собі широку різноманітність ситуацій, які можуть впливати на рівень стресу людини та нести загрозу її життю та здоров'ю. Розуміючи та адресуючи їх можна покращити підтримку студентів, працівників та інших людей, які є об'єктами охорони праці. Тому розглянемо детальніше найпоширеніші види соціальних та психологічних

факторів ризику та загрози безпеці життєдіяльності людини, які вони несуть, фактори, що їх окреслюють.

Беручи до уваги соціальні фактори ризику слід розуміти, що вони є відображенням того, як соціальне оточення формує звички та поведінку як і особистостей так і груп [8]. Розглянемо ключові аспекти цих ризиків:

– Міжособистісні стосунки. Вони вносять до себе стосунки з сім'єю, друзями, колегами та іншими. Якість цих стосунків, відчуття ізоляції та вигнання можуть впливати на здоров'я людини. Негативні чинники такі як соціальне відчуження або цькування можуть призвести до стресу, тривоги та депресії.

– Соціальна підтримка. Важливим фактором є ступінь емоційної, інформаційної та практичної підтримки, яку отримує людина від свого оточення. Їх нестача може призвести до зростання вразливості до стресу та негативних наслідків для здоров'я.

– Соціоекономічний стан. Такі фактори як дохід, освіта та працевлаштування суттєво впливають на доступ людини до ресурсів та можливостей покращити своє ментальне та фізичне благополуччя. Економічна нестабільність або бідність ведуть до хронічного стресу, що несе шкоду фізичному та ментальному здоров'ю, це також збільшує шанси антисоціальної та маргінальної поведінки окремих груп та особистостей.

– Культурні норми та цінності. Суспільні правила та очікування, які впливають на поведінку окремого соціуму. Його ставлення щодо ментального здоров'я, статі, раси та інших рис особистості може вплинути на утворення стигми, дискримінації та інших форм соціальної нерівності, які шкодять здоров'ю людини.

Адресування цих факторів ризику включає до себе ширше втручання на соціальному та державному рівнях. Прикладами таких дій можуть бути покращення та збільшення доступності охорони здоров'я, ініціативи розвитку громад, програми з надання робочих місць, спрямовані на покращення соціальних умов та справедливішого доступу до ресурсів та можливостей.

Щодо психологічних факторів ризику варто наголосити, що вони тісно пов’язані з ментальним та емоційним станом особистості, процесами, які впливають на здоров’я та благополуччя людини, це взаємодія між її особистими думками, почуттями та поведінкою [9]. Основними такими факторами є:

– Когнітивні процеси. Вони містять в собі моделі мислення та системи цінностей особистості. Негативні або викривлені моделі призводять до підвищення стресу та можуть призвести до таких розладів як тривожність та депресія.

– Емоційний стан. Стійкі та довготривалі негативні емоції шкідливо впливають на функціональність та працездатність людини, потенційно призводячи до емоційних розладів особистості. З іншого боку позитивні емоції та емоційна стійкість є факторами, що зменшують ці ризики.

– Стрес-менеджмент та механізми подолання складностей. Те як людина справляється зі стресом та її механізми подолання труднощів суттєво впливають на стан її ментального здоров’я. Неадаптивні механізми та стратегії такі як уникання та зловживання речовинами можуть призвести до подальшого погіршення психологічного стану особистості та подальших ускладнень.

– Моделі поведінки. Вони відносять до себе звички та рутину особистості. Шкідливі звички як фізична неактивність, нездорове харчування та недостатній або неякісний сон можуть призвести до виникання широкого спектру проблем зі здоров’ям, як фізичним, так і ментальним.

4.2 Нормативна база охорони праці

Враховуючи, що основною цільовою аудиторією цього проєкту є здобувачі освіти, в тому числі студенти спеціальності 121 “Інженерія програмного

забезпечення”, розглянемо як соціальні та психологічні фактори ризику впливають конкретно на них та які заходи приймаються для запобігання їм.

Законодавство України про охорону праці являє собою систему взаємозв'язаних нормативно-правових актів, що регулюють відносини у галузі реалізації державної політики щодо правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці.

Законодавство про охорону праці складається з (відповідно до статті 3 Закону України «Про охорону праці» [10]) :

– Закону України «Про охорону праці»;

– Кодексу законів про працю України;

– Закону України «Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності»;

– Прийнятих відповідно до них нормативно-правових актів.

Закон України «Про охорону праці» визначає основні положення з реалізації конституційного права громадян на охорону їхнього життя й здоров'я у процесі трудової діяльності, регулює за участю відповідних державних органів відносини між власником підприємства і працівником з питань безпеки праці, виробничої санітарії, встановлює єдиний порядок організації охорони праці у виробничій сфері в Україні. Чинність закону України «Про охорону праці» поширюється на всі підприємства, установи, організації незалежно від форм власності й видів діяльності, що використовують найману працю, і на всіх, хто працює. Закон визначає основні принципи державної політики в області охорони праці, серед яких, відповідно до статті 4 Закону України «Про охорону праці», чільне місце займають:

– Пріоритет життя й здоров'я працівників стосовно результатів виробничої діяльності підприємства;

– Повна відповідальність власника підприємства за створення безпечних і нешкідливих умов праці;

– Соціальний захист працівників;

– Повне відшкодування шкоди особам, які потерпіли на виробництві від нещасних випадків або професійних захворювань.

– Окремо виділені статті Закону присвячені регулюванню охорони праці жінок, неповнолітніх, інвалідів, видам відповідальності за порушення законодавства і нормативних актів про охорону праці, за створення перешкод для діяльності посадових осіб органів державного нагляду за охороною праці і представників профспілок.

Нормативно-правові акти з охорони праці (НПАОП) (стаття 27 Закону України «Про охорону праці») – це правила, норми, регламенти, положення, інструкції та інші документи, які є обов'язковими до виконання. Сьогодні до Реєстру НПАОП включено нормативні акти з охорони праці, затверджені відповідними органами нагляду протягом останніх років, внесено офіційні зміни і доповнення, що містяться в інформаційних повідомленнях, враховано зауваження міністерств і відомств щодо уточнення назв нормативних актів та дат їх затвердження. У версії Покажчика НПАОП станом на 22.09.2022 р. міститься 432 чинних НПАОП [11].

Розглядаючи соціальні та психологічні фактори ризику в студентів ІПЗ, варто розуміти, що попри те, що процеси праці та навчання інженерії програмного забезпечення схожі в своїй суті, досвід студента та працівника відрізняється, як і проблеми з якими вони стикаються. Враховуючи результати проведеного опитування серед студентів ІПЗ та додаткового дослідження цієї теми, було сформовано основні соціальні та психологічні проблеми з якими стикаються студенти — інженери програмного забезпечення.

Основним, визначеним нормативами, суб’єктом запобігання соціальним та психологічним ризикам є психологічна служба закладу освіти. На сьогодні основними документами, які регламентують роботу, структуру та організацію

охорони праці у вигляді психологічної служби закладу освіти, можна вважати наступні підзаконні акти:

– Закон України «Про освіту» від 05.09.2017р. Стаття 76. «Психологічна служба та соціально-педагогічний патронаж у системі освіти»;

– Положення про психологічну службу системи освіти України. Наказ МОН України № 509 від 22.05.2018;

– Лист МОН від 02.08.2022 № 1/8794-22 “Щодо діяльності психологічної служби у системі освіти в 2022/2023 навчальному році”;

– Лист МОН від 30.11.2020 № 6/1427-20 «Про затвердження професійного стандарту «Практичний психолог закладу освіти»;

– Лист МОН від 06.06.13 № 1/9-413 «Про впровадження факультативних курсів працівниками психологічної служби системи освіти»;

– Постанова кабінету міністрів від 21.08.2019р № 800 «Деякі питання кваліфікації педагогічних і науково-педагогічних працівників»;

Згідно з положенням про психологічну службу системи освіти України [12], вона має відповідну структуру:

– Державна наукова установа «Інститут модернізації змісту освіти»;

– Український науково-методичний центр практичної психології і соціальної роботи Національної академії педагогічних наук України;

– Навчально-методичні центри/кабінети/лабораторії психологічної служби Автономної Республіки Крим, обласні, Київський(а) і Севастопольський(а) міські;

– Міські та місцеві навчально-методичні центри/кабінети/лабораторії психологічної служби, її підрозділи у закладах освіти, їх практичні психологи та соціальні педагоги;

У цьому положенні також визначено завдання, принципи та фукнції психологічної служби.

Завдання психологічної служби:

– Збереження та зміцнення психічного та соціального здоров’я, сприяння особистісному, інтелектуальному, фізичному і соціальному розвитку здобувачів освіти шляхом доповнення сучасних методів навчання та виховання ефективними психолого-педагогічними технологіями;

– Сприяння забезпеченню психологічної безпеки, надання психологічної і соціально-педагогічної допомоги всім учасникам освітнього процесу.

Принципами діяльності психологічної служби є:

– Науковість, цілісність і наступність, професійна компетентність та відповідальність;

– Індивідуальний підхід;

– Доступність соціально-педагогічних та психологічних послуг (допомоги);

– Міждисциплінарність, комплексність і системність у здійсненн професійної діяльності;

– Добровільність;

– Людиноцентризм та партнерство;

– Конфіденційність;

Функції психологічної служби:

– Діагностично-прогностична – психолого-педагогічне вивчення чинників становлення особистості, її індивідуального розвитку;

– Організаційно-методична – визначення стратегії, мети, завдань, планування діяльності психологічної служби та координація взаємодії учасників освітнього процесу;

– Корекційно-розвиткова – надання психолого-педагогічної допомоги здобувачам освіти з метою адаптації до умов навчання і життєдіяльності;

– Консультативна – допомога у вирішенні проблем щодо розвитку, виховання, навчання та формування психологічної і соціальної компетентності учасників освітнього процесу;

– Соціально-захисна – здійснення соціально-педагогічного супроводу учасників освітнього процесу, які опинилися у складних життєвих обставинах, перебувають у кризових ситуаціях (постраждали від соціальних, техногенних, природних катастроф, перенесли тяжкі хвороби, стреси, переселення, зазнали насильства тощо); захист конституційних прав і статусу, законних інтересів здобувачів освіти.

За умов їх реалізації, ці заходи повинні зменшувати соціальні та психологічні ризики з якими стикаються студенти — інженери програмного забезпечення, створювати комфортне навчальне середовище та надавати доступ до ресурсів підтримки ментального здоров’я.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено додаток для допомоги студентам у покращенні їх організації навчання та підходу до виконання академічних завдань. Для цього було проведено ретельний аналіз предметної області: проведено аналіз наявних додатків на цьому ринку, предметної області мобільних додатків для допомоги у навчанні, проведено та проаналізовано опитування серед студентів з метою визначення їх потреб та попиту у таких додатках. Опитування показало, що студенти мають велику потребу у підтримці та організації їх навчального процесу через проблеми з якими вони стикаються. Проведено аналіз технічного аспекту розробки мобільного застосунку такого типу.

Наступним був етап моделювання та проєктування застосунку. У ньому було визначено та обґрунтовано використання технологій розробки, це ReactNative, Firebase, Jest та інші. Після того було визначено модель предметної області розробки за допомогою визначення сценаріїв використання додатку та UML діаграм, які дали краще розуміння взаємодії елементів додатку між собою та користувачем. В кінці було спроєктовано модель архітектури системи за допомогою діаграми взаємодії її елементів, та описано їх функціонал та взаємодію.

Останнім етапом розробки було конструювання та тестування самого додатку. Завдяки визначеним технологіям, створеній моделі та спроєктованій архітектурі було швидко та ефективно створено основні робочі модулі системи та графічний дизайн додатку. Тестування було проведено в фреймворку Jest та показало достатню якість додатку відповідно до вимог та потреб користувачів, було закладено подальшу стратегію підтримки та розвитку додатку у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. EdTech Market - Global Outlook & Forecast 2022-2027, MarketResearch.com: Market Research Reports and Industry Analysis, 2022. [Online]. Available: https://www.marketresearch.com/Arizton-v4150/EdTech-Global-Outlook-Forecast-30630646/

2. Víctor J. García-Morales, Aurora Garrido-Moreno and Rodrigo Martín-Rojas, "The Transformation of Higher Education After the COVID Disruption: Emerging Challenges in an Online Learning Scenario", Frontiers, vol. 12, February 2021, [Online], Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2021.616059/full

3. Д. В. Романюк, Опитування "Дослідження навчальних звичок студентів та їх потреб", Google Docs, 2023, [Онлайн], Доступно: https://forms.gle/Aswc5w9WxHzgDkLG8

4. S. Dekhane, X. Xu, M. Yin Tsoy, "Mobile App Development to Increase Student Engagement and Problem Solving Skills", Journal of Information Systems Education, vol. 24(4), 2013, [Online], Available: https://core.ac.uk/download/pdf/301384564.pdf

5. Z. L. Berge та L. Muilenburg, Ред., Handbook of Mobile Learning. Routledge, 2013. [Онлайн]. Доступно: https://doi.org/10.4324/9780203118764

6. Y. Wurmser. "Apps Far Outpace Browsers in Adults' Mobile Time Spent". Insider Intelligence. 2020, [Онлайн], Доступно: https://www.insiderintelligence.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps

7. M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language. Pearson Education, Limited, 2018.

8. E. Goffman, The presentation of self in everyday life. Edinburgh: Edinburgh University Social Sciences Research Centre, 1956.

9. "Mental health". World Health Organization (WHO). 2014, [Онлайн], Доступно: https://www.who.int/news-room/fact-sheets/detail/mental-health-strengthening-our-response

10. Україна, Верховна Рада України. (1992, 14 жовт.). Закон України №2694-XII, Про охорону праці. [Онлайн]. Доступно: https://zakon.rada.gov.ua/laws/show/2694-12#Text

11. “Покажчик нормативно-правових актів з охорони праці - Державна служба України з питань праці”. Державна служба України з питань праці, 2022, [Онлайн], Доступно: https://dsp.gov.ua/pokazhchyk-normatyvno-pravovykh-aktiv-z/.

12. “Про затвердження Положення про психологічну службу у системі освіти України”. Офіційний вебпортал парламенту України, 2018, [Онлайн], Доступно: https://zakon.rada.gov.ua/laws/show/z0885-18#Text.

13. D. M. West. "Mobile Learning: Transforming Education, Engaging Students, and Improving Outcomes". Brookings, 2013, [Онлайн], Доступно: https://www.brookings.edu/research/mobile-learning-transforming-education-engaging-students-and-improving-outcomes/

14. B. Eisenman, Learning React Native: Building Native Mobile Apps with JavaScript. O'Reilly Media, 2017.

15. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Програмне забезпечення систем», Освітньо-наукова програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М. Р. Петрик, Д. М. Михалик, О. Ю. Петрик, Г. Б. Цуприк – Тернопіль: ТНТУ, 2020- 51с.

ДОДАТКИ

Додаток А

Лістинг коду застосунку

TaskItem.js

```
import React from 'react';
import { View, Text, StyleSheet, TouchableOpacity } from 'react-native';

const TaskItem = ({ task, onDelete }) => {

  // This function will be called when the user presses the task
  const handlePress = () => {
    onDelete(task.id);
  };

  return (
    <TouchableOpacity onPress={handlePress}>
      <View style={styles.item}>
        <Text style={styles.title}>{task.title}</Text>
        <Text style={styles.description}>{task.description}</Text>
      </View>
    </TouchableOpacity>
  );
};

const styles = StyleSheet.create({
  item: {
    backgroundColor: '#f9c2ff',
    padding: 20,
    marginVertical: 8,
    marginHorizontal: 16,
    borderRadius: 10,
  },
  title: {
    fontSize: 24,
  },
  description: {
    fontSize: 18,
  }
});

export default TaskItem;
```

ScheduleItem.js

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, Modal, Button } from 'react-
native';
import DateTimePicker from '@react-native-community/datetimepicker';

const ScheduleItem = ({ schedule, onDelete, onEdit }) => {
  const [modalVisible, setModalVisible] = useState(false);
  const [date, setDate] = useState(new Date());

  const handlePress = () => {
    onDelete(schedule.id);
  };

  const handleEdit = () => {
    setModalVisible(true);
```

```
  };

  const handleConfirm = () => {
    onEdit(schedule.id, date);
    setModalVisible(false);
  };

  const handleChange = (event, selectedDate) => {
    const currentDate = selectedDate || date;
    setDate(currentDate);
  };

  return (
    <TouchableOpacity onPress={handlePress} onLongPress={handleEdit}>
      <View style={styles.item}>
        <Text style={styles.title}>{schedule.subject}</Text>
        <Text style={styles.description}>{schedule.location}</Text>
        <Text style={styles.description}>{schedule.description}</Text>
                        <Text   style={styles.description}>{schedule.startTime}   -
{schedule.endTime}</Text>
      </View>
      <Modal
        animationType="slide"
        transparent={true}
        visible={modalVisible}
      >
        <View style={styles.centeredView}>
          <View style={styles.modalView}>
            <DateTimePicker
              testID="dateTimePicker"
              value={date}
              mode="date"
              is24Hour={true}
              display="default"
              onChange={handleChange}
            />
            <Button title="Confirm" onPress={handleConfirm} />
          </View>
        </View>
      </Modal>
    </TouchableOpacity>
  );
};

const styles = StyleSheet.create({
  centeredView: {
    flex: 1,
    justifyContent: "center",
    alignItems: "center",
    marginTop: 22
  },
  modalView: {
    margin: 20,
    backgroundColor: "white",
    borderRadius: 20,
    padding: 35,
    alignItems: "center",
    shadowColor: "#000",
    shadowOffset: {
      width: 0,
      height: 2
```

```
    },
    shadowOpacity: 0.25,
    shadowRadius: 4,
    elevation: 5
  },
  item: {
    backgroundColor: '#f9c2ff',
    padding: 20,
    marginVertical: 8,
    marginHorizontal: 16,
    borderRadius: 10,
  },
  title: {
    fontSize: 24,
  },
  description: {
    fontSize: 18,
  }
});

export default ScheduleItem;
```

SettingItem.js

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, Switch, Button, Slider } from 'react-native';

const DEFAULT_FREQUENCY = 15;

const SettingItem = ({ setting, onToggle, onSliderChange, onReset }) => {
  const [isEnabled, setIsEnabled] = useState(setting.isEnabled);
  const [frequency, setFrequency] = useState(setting.frequency);

  const toggleSwitch = () => {
    setIsEnabled(previousState => !previousState);
    onToggle(setting.id, !isEnabled);
  };

  const sliderChange = (value) => {
    setFrequency(value);
    onSliderChange(setting.id, value);
  };

  const resetSetting = () => {
    setIsEnabled(true);
    setFrequency(DEFAULT_FREQUENCY);
    onReset(setting.id);
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>{setting.name}</Text>
      <Switch
        trackColor={{ false: "#767577", true: "#81b0ff" }}
        thumbColor={isEnabled ? "#f5dd4b" : "#f4f3f4"}
        ios_backgroundColor="#3e3e3e"
        onValueChange={toggleSwitch}
        value={isEnabled}
      />
      <Slider
        style={{width: 200, height: 40}}
```

```
        minimumValue={1}
        maximumValue={60}
        value={frequency}
        onValueChange={sliderChange}
        maximumTrackTintColor="#000000"
        minimumTrackTintColor="#0000ff"
      />
      <Text style={styles.frequency}>{frequency} min</Text>
      <Button title="Reset" onPress={resetSetting} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
    padding: 10,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    padding: 10,
  },
  frequency: {
    fontSize: 18,
    padding: 10,
  }
});

export default SettingItem;
```

MainTabNavigator.js

```
import React from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { MaterialCommunityIcons } from '@expo/vector-icons';

import HomeScreen from '../screens/HomeScreen';
import ScheduleScreen from '../screens/ScheduleScreen';
import TasksScreen from '../screens/TasksScreen';
import SettingsScreen from '../screens/SettingsScreen';

const Tab = createBottomTabNavigator();

function MainTabNavigator() {
  return (
    <Tab.Navigator
      initialRouteName="Home"
      tabBarOptions={{
        activeTintColor: '#2f95dc',
      }}
    >
      <Tab.Screen
        name="Home"
        component={HomeScreen}
        options={{
          tabBarLabel: 'Home',
          tabBarIcon: ({ color, size }) => (
            <MaterialCommunityIcons name="home" color={color} size={size} />
```

```
          ),
        }}
      />

      <Tab.Screen
        name="Schedule"
        component={ScheduleScreen}
        options={{
          tabBarLabel: 'Schedule',
          tabBarIcon: ({ color, size }) => (
            <MaterialCommunityIcons name="calendar" color={color} size={size} />
          ),
        }}
      />

      <Tab.Screen
        name="Tasks"
        component={TasksScreen}
        options={{
          tabBarLabel: 'Tasks',
          tabBarIcon: ({ color, size }) => (
            <MaterialCommunityIcons name="check" color={color} size={size} />
          ),
        }}
      />

      <Tab.Screen
        name="Settings"
        component={SettingsScreen}
        options={{
          tabBarLabel: 'Settings',
          tabBarIcon: ({ color, size }) => (
            <MaterialCommunityIcons name="cog" color={color} size={size} />
          ),
        }}
      />
    </Tab.Navigator>
  );
}

export default MainTabNavigator;
```

AppNavigator.js

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

import MainTabNavigator from './MainTabNavigator';
import AuthScreen from '../screens/AuthScreen';
import SplashScreen from '../screens/SplashScreen';

const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        screenOptions={{
          headerShown: false,
        }}
```

```
      >
        <Stack.Screen name="Splash" component={SplashScreen} />

        <Stack.Screen name="Auth" component={AuthScreen} />

        <Stack.Screen
          name="Main"
          component={MainTabNavigator}
          options={{
              animationEnabled: false,  // You might want to disable animation for
the Main Tab Navigator
          }}
        />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default AppNavigator;
```

HomeScreen.js

```
import React, { useEffect, useState } from 'react';
import { View, Text, FlatList, ActivityIndicator } from 'react-native';
import { TouchableOpacity } from 'react-native-gesture-handler';
import firebase from 'firebase';
import TaskItem from '../components/TaskItem';

export default function HomeScreen({ navigation }) {
  const [loading, setLoading] = useState(true);
  const [tasks, setTasks] = useState([]);

  useEffect(() => {
    const subscriber = firebase.firestore()
      .collection('tasks')
      .onSnapshot(querySnapshot => {
        const tasks = [];

        querySnapshot.forEach(documentSnapshot => {
          tasks.push({
            ...documentSnapshot.data(),
            key: documentSnapshot.id,
          });
        });

        setTasks(tasks);
        setLoading(false);
      });

    // Unsubscribe from events when no longer in use
    return () => subscriber();
  }, []);

  if (loading) {
    return <ActivityIndicator />;
  }

  return (
    <View style={{ flex: 1, width: '100%' }}>
      <FlatList
        data={tasks}
```

```
        renderItem={({ item }) => (
              <TouchableOpacity onPress={() => navigation.navigate('TaskDetails',
{item})}>
            <TaskItem {...item} />
          </TouchableOpacity>
        )}
      />
      <TouchableOpacity
        style={{margin: 20, padding: 20, backgroundColor: '#f5f5f5'}}
        onPress={() => navigation.navigate('AddTask')}>
        <Text>Add Task</Text>
      </TouchableOpacity>
    </View>
  );
}
```

TaskScreen.js

```
import React, { useEffect, useState } from 'react';
import { View, Text, Button, ActivityIndicator } from 'react-native';
import firebase from 'firebase';

export default function TaskScreen({ route, navigation }) {
  const { item } = route.params;
  const [task, setTask] = useState(null);
  const [loading, setLoading] = useState(true);

  const taskRef = firebase.firestore().collection('tasks').doc(item.key);

  useEffect(() => {
    return taskRef.onSnapshot((doc) => {
      setTask({ ...doc.data(), key: doc.id });
      setLoading(false);
    });
  }, []);

  const onMarkComplete = () => {
    setLoading(true);
    taskRef
      .update({
        complete: true,
      })
      .then(() => setLoading(false))
      .catch((error) => {
        setLoading(false);
        console.error("Error updating document: ", error);
      });
  };

  if (loading) {
    return <ActivityIndicator />;
  }

  return (
    <View style={{ flex: 1, padding: 20 }}>
      <Text>{task.title}</Text>
      <Text>{task.description}</Text>
      <Text>{task.complete ? 'Complete' : 'Incomplete'}</Text>
      {!task.complete && (
        <Button title="Mark as complete" onPress={onMarkComplete} />
      )}
```

```
    </View>
  );
}
```

ScheduleScreen.js

```
import React, { useState, useEffect } from 'react';
import { View, Text, Button, FlatList, ActivityIndicator } from 'react-native';
import firebase from 'firebase';
import ScheduleItem from './ScheduleItem';

export default function ScheduleScreen({ navigation }) {
  const [schedules, setSchedules] = useState([]);
  const [loading, setLoading] = useState(true);

  const scheduleRef = firebase.firestore().collection('schedules');

  useEffect(() => {
    return scheduleRef.onSnapshot(querySnapshot => {
      const list = [];
      querySnapshot.forEach(doc => {
        const { title, details, date } = doc.data();
        list.push({
          id: doc.id,
          title,
          details,
          date,
        });
      });
      setSchedules(list);
      if (loading) {
        setLoading(false);
      }
    });
  }, []);

  if (loading) {
    return <ActivityIndicator />;
  }

  return (
    <View style={{ flex: 1, padding: 20 }}>
                          <Button    title="Add    Schedule"    onPress={()    =>
navigation.navigate('AddSchedule')} />
      <FlatList
        data={schedules}
        keyExtractor={item => item.id}
        renderItem={({ item }) => <ScheduleItem {...item} />}
      />
    </View>
  );
}
```

SettingScreen.js

```
import React, { useState } from 'react';
import { View, Text, Button, Switch, StyleSheet } from 'react-native';

export default function SettingsScreen({ navigation }) {
  const [isNotificationsEnabled, setIsNotificationsEnabled] = useState(false);
```

```
   const toggleNotifications = () => setIsNotificationsEnabled(previousState => !
previousState);

  const [isDarkModeEnabled, setIsDarkModeEnabled] = useState(false);
     const   toggleDarkMode   =   ()   =>   setIsDarkModeEnabled(previousState   =>   !
previousState);

  return (
    <View style={styles.container}>
      <Text style={styles.header}>Profile</Text>
                          <Button    title="Edit    Profile"    onPress={()    =>
navigation.navigate('EditProfile')} />

      <Text style={styles.header}>Notifications</Text>
      <View style={styles.row}>
        <Text>Enable Notifications</Text>
        <Switch
          trackColor={{ false: "#767577", true: "#81b0ff" }}
          thumbColor={isNotificationsEnabled ? "#f5dd4b" : "#f4f3f4"}
          ios_backgroundColor="#3e3e3e"
          onValueChange={toggleNotifications}
          value={isNotificationsEnabled}
        />
      </View>

      <Text style={styles.header}>Appearance</Text>
      <View style={styles.row}>
        <Text>Dark Mode</Text>
        <Switch
          trackColor={{ false: "#767577", true: "#81b0ff" }}
          thumbColor={isDarkModeEnabled ? "#f5dd4b" : "#f4f3f4"}
          ios_backgroundColor="#3e3e3e"
          onValueChange={toggleDarkMode}
          value={isDarkModeEnabled}
        />
      </View>

      <Text style={styles.header}>Other</Text>
      <Button title="About App" onPress={() => navigation.navigate('About')} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
  },
  header: {
    fontSize: 18,
    fontWeight: 'bold',
    marginTop: 20,
  },
  row: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    paddingVertical: 10,
  },
});
```

authService.js

```
import { firebase } from './firebaseConfig';

class AuthService {
    // Sign up new user
    signUp(email, password) {
        return firebase
            .auth()
            .createUserWithEmailAndPassword(email, password);
    }

    // Sign in existing user
    signIn(email, password) {
        return firebase
            .auth()
            .signInWithEmailAndPassword(email, password);
    }

    // Sign out user
    signOut() {
        return firebase.auth().signOut();
    }

    // Check auth status
    authStateChange(onChange) {
        firebase.auth().onAuthStateChanged(onChange);
    }
}

export default new AuthService();
```

TaskService.js

```
import { firebase } from './firebaseConfig';

class TaskService {
    constructor() {
        this.db = firebase.firestore();
        this.tasks = this.db.collection('tasks');
    }

    // Add a new task
    addTask(userId, task) {
        return this.tasks.add({
            userId: userId,
            title: task.title,
            description: task.description,
            dueDate: task.dueDate,
            completed: task.completed || false
        });
    }

    // Get tasks of a user
    getUserTasks(userId) {
        return this.tasks
            .where('userId', '==', userId)
            .orderBy('dueDate')
            .get();
    }
```

```
    // Update a task
    updateTask(taskId, updatedTask) {
        return this.tasks.doc(taskId).update(updatedTask);
    }

    // Delete a task
    deleteTask(taskId) {
        return this.tasks.doc(taskId).delete();
    }
}

export default new TaskService();
```

ScheduleService.js

```
import { firebase } from './firebaseConfig';

class ScheduleService {
    constructor() {
        this.db = firebase.firestore();
        this.schedule = this.db.collection('schedule');
    }

    // Add a new schedule item
    addScheduleItem(userId, item) {
        return this.schedule.add({
            userId: userId,
            title: item.title,
            start: item.start,
            end: item.end
        });
    }

    // Get all schedule items of a user
    getUserScheduleItems(userId) {
        return this.schedule
            .where('userId', '==', userId)
            .orderBy('start')
            .get();
    }

    // Update a schedule item
    updateScheduleItem(itemId, updatedItem) {
        return this.schedule.doc(itemId).update(updatedItem);
    }

    // Delete a schedule item
    deleteScheduleItem(itemId) {
        return this.schedule.doc(itemId).delete();
    }
}

export default new ScheduleService();
```

SettingService.js

```
import { firebase } from './firebaseConfig';

class SettingsService {
    constructor() {
        this.db = firebase.firestore();
```

```
        this.settings = this.db.collection('settings');
    }

    // Get user settings
    getSettings(userId) {
        return this.settings
            .doc(userId)
            .get()
            .then(doc => {
                if (doc.exists) {
                    return doc.data();
                } else {
                    // If no settings exist for this user, initialize them
                    return this.setSettings(userId, this.defaultSettings);
                }
            });
    }

    // Set user settings
    setSettings(userId, settings) {
        return this.settings
            .doc(userId)
            .set(settings);
    }

    // Default settings to be applied when a new user is created or when settings
are reset
    defaultSettings = {
        theme: 'light', // or 'dark'
        notificationsEnabled: true,
        scheduleView: 'week', // or 'month', 'day'
        // Add more as required
    };
}

export default new SettingsService();
```

helpers.js

```
import { format } from 'date-fns';

// Function to format dates consistently
export function formatDate(date, dateFormat = 'yyyy-MM-dd') {
  return format(date, dateFormat);
}

// Function to format time consistently
export function formatTime(date, timeFormat = 'HH:mm') {
  return format(date, timeFormat);
}

// Function to validate email format
export function validateEmail(email) {
  const re = /\S+@\S+\.\S+/;
  return re.test(email);
}

// Function to capitalize the first letter of a string
export function capitalizeFirstLetter(string) {
  return string.charAt(0).toUpperCase() + string.slice(1);
```

```
}

// Function to limit the number of characters in a string (for display purposes)
export function truncateString(string, num) {
  return string.length > num ? string.slice(0, num) + '...' : string;
}
```

App.tsx

```
import React from 'react';
import { StatusBar } from 'react-native';
import { Provider as PaperProvider } from 'react-native-paper';
import { NavigationContainer } from '@react-navigation/native';

import { AuthProvider } from './src/contexts/authContext';
import AppNavigator from './src/navigation/AppNavigator';

const App = () => {
  return (
    <PaperProvider>
      <AuthProvider>
        <StatusBar barStyle="dark-content" />
        <NavigationContainer>
          <AppNavigator />
        </NavigationContainer>
      </AuthProvider>
    </PaperProvider>
  );
};

export default App;
```

index.js

```
import { AppRegistry } from 'react-native';
import App from './App';
import { name as appName } from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

package.json

```
  "name": "Zeal",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "lint": "eslint .",
    "start": "react-native start",
    "test": "jest"
  },
  "dependencies": {
    "react": "18.2.0",
    "react-native": "0.71.11"
  },
  "devDependencies": {
    "@babel/core": "^7.20.0",
    "@babel/preset-env": "^7.20.0",
    "@babel/runtime": "^7.20.0",
```

```
    "@react-native-community/eslint-config": "^3.2.0",
    "@tsconfig/react-native": "^2.0.2",
    "@types/jest": "^29.2.1",
    "@types/react": "^18.0.24",
    "@types/react-test-renderer": "^18.0.0",
    "babel-jest": "^29.2.1",
    "eslint": "^8.19.0",
    "jest": "^29.2.1",
    "metro-react-native-babel-preset": "0.73.10",
    "prettier": "^2.4.1",
    "react-test-renderer": "18.2.0",
    "typescript": "4.8.4"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

App-test.tsx

```
import React from 'react';
import { render, cleanup, fireEvent } from '@testing-library/react-native';

import App from '../App';

// Clean up on exiting
afterEach(cleanup);

describe('App', () => {
    it('renders successfully', () => {
        const { toJSON } = render(<App />);

        expect(toJSON()).toBeTruthy();
    });

    it('renders the home screen', () => {
        const { getByTestId } = render(<App />);
        const homeScreen = getByTestId('home-screen');

        expect(homeScreen).toBeTruthy();
    });

    it('navigates to the calendar screen', () => {
        const { getByTestId, getByText } = render(<App />);
        const navButton = getByText('Go to Calendar');

        fireEvent.press(navButton);
        const calendarScreen = getByTestId('calendar-screen');

        expect(calendarScreen).toBeTruthy();
    });

});
```

Додаток Б

Диск