ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ STUDENT ASSISTANT ДЛЯ ПЛАНУВАННЯ НАВЧАННЯ НА МОВІ ПРОГРАМУВАННЯ JAVASCRIPT»**

Виконав: студент 4 курсу, групи ПД–42
спеціальності
121 Інженерії програмного забезпечення
(шифр і назва спеціальності)
Дехтяренко О.Р.
(прізвище та ініціали)
Керівник к.т.н., доцент Негоденко О.В.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)
Нормоконтроль _____
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Напрям підготовки - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення

О.В. Негоденко

«____» _____20__ року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Дехтяренку Олександру Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка мобільного додатку Student Assistant для планування навчання на мові програмування JavaScript»

Керівник роботи___Негоденко Олена Василівна, к.т.н., доцент, завідувач кафедри,

затверджені наказом вищого навчального закладу від — 12.03 2021 року №65.

2. Строк подання студентом роботи___01.06.2021_____

3. Вхідні дані до роботи:
   3.1. Додаток Visual Studio Code;
   3.2. Тестування програмного забезпечення;
   3.3. Проектування програмного забезпечення;
   3.4. Науково-технічна література пов'язана з розробкою мобільних застосунків;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
   4.1 Аналіз предметної області;
   4.2 Аналіз існуючих аналогів застосунку ;
   4.3 Визначення вимог до розроблюваного застосунку;
   4.4 Дослідження програмних засобів ;
   4.5 Моделювання застосунку Тестування застосунку

5. Перелік графічного матеріалу

5.1. Мета, об’єкт, дослідження;
5.2. Існуючі аналоги;
5.3. Таблиця переваг та недоліків;
5.4.Функціонал;
5.5. Операційні системи;
5.5. Програмні засоби реалізації;
5.6. Діаграма прецедентів;
5.7. Діаграми діяльності;
5.8. Схема бази даних застосунку;
5.9. Апробація результатів дослідження;
5.10. Висновки

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури	19.04.2021	
2.	Аналіз програмних засобів	20.04.2021 – 22.04. 2021	
3.	Проектування системи	22.04. 2021- 24.04. 2021	
4.	Програмна підготовка проекту	24.04. 2021	
5.	Програмна реалізація застосунку	24.04-2021- 28.04.2021	
6.	Оформлення пояснювальної записки	28.04.2021- 29.04.2021	
7.	Попередній захист роботи	11.05.2021	
8.	Подання роботи в деканат	1.06.2021	

Студент_____ __Дехтяренко О.Р_____
(підпис) (прізвище та ініціали)

Керівник Роботи_____ _Негоденко О.В___
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 49 с., 4 табл., 34 рис., 20 джерел.

VISUAL STUDIO CODE, REDUX, REACT NATIVE, REACT, JAVASCRIPT, EXPO, UML, FIREBASE, ANDROID

Об'єкт дослідження – покращення ефективності в навчальному процесі.

Предмет дослідження – додаток для планування навчання.

Мета роботи – розробка програмного забезпечення для студентів закладів вищої освіти та учнів шкіл на мові програмування JavaScript.

Наукова новизна роботи полягає в наступному:

1. Покращено алгоритм для планування навчання;
2. Встановлено, що JavaScript є досить перспективною мовою для розробки мобільного програмного забезпечення;
3. Розроблено програмне забезпечення для покращення ефективності в навчальному процесі;

Використання розроблюваного застосунку дозволить користувачам зберігати навчальну інформацію в компактному та чітко структурованому вигляді. Водночас застосунок зможе надавати користувачу інформацію про його успішність в навчанні та рекомендації щодо її покращення.

Проведено аналіз існуючих аналогів розроблюваного додатку. Визначено недоліки та переваги існуючих аналогів. На основі цього аналізу встановлено вимоги до програмного забезпечення.

Проведено аналіз середовищ розробки, встановлено їх переваги та недоліки.

Вибрано середовище розробки Visual Studio Code.

Сфера застосування – додаток використовується під час навчання та призначений для учнів шкіл та студентів закладів вищої освіти.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Прикладний програмний інтерфейс

DOM – Об'єктна модель документа

БД – База даних

USB – Універсальна послідовна шина

LAN – Локальна мережа

JSON – Текстовий формат обміну даних

ПЗ – Програмне забезпечення

ВСТУП

Актуальність теми: Невід'ємною частиною успішного навчання є аналіз власних слабких та сильних сторін в навчанні. Частково цей процес забезпечується навчальними паперовими носіями інформації такими як, щоденник, зошит. Але потреба зберігання інформації в паперовому форматі стає все меншою, оскільки розвиток інформаційних технологій надає набагато зручніший спосіб представлення інформації, зокрема і навчальної. Більш того, цифровий спосіб зберігання даних передбачає їх аналіз, чого не можуть робити паперові носії інформації. Цифрові системи надають широкий спектр можливостей, які значно спрощують життя. Якщо розглядати даний фактор в контексті навчання, то такий інструмент в цифровому форматі зміг би, наприклад аналізувати оцінки, які отримав користувач та надавати певні рекомендації, що до їх покращення, якщо вони погані. Такий інструмент значно би покращив орієнтацію в навчальному процесі.

Наразі не існує повноцінних цифрових інструментів для планування навчання. Більшість існуючих аналогів вирішують проблему планування навчання лише частково. Вони слугують лише, як спосіб зберігання навчальних даних, без будь-якого аналізу цих даних.

Тому створення цифрового інструменту для планування навчання є актуальною задачею, оскільки на ринку не існує жодного додатку для планування навчання, який би надавав рекомендації, щодо покращення успішності в навчанні. Також сучасний рівень розвитку інформаційних технологій сприяє створенню такого інструменту.

Джерела дослідження: Основними джерелами дослідження, є навчально-наукова література, яка пов'язана з розробкою мобільного програмного забезпечення на мові програмування JavaScript, та інтернет ресурси на тему програмування та розробки мобільних додатків.

Об'єктом дослідження є покращення ефективності в навчальному процесі.

Предмет дослідження – мобільний додаток для планування навчання.

Мета роботи – розробка програмного забезпечення для студентів закладів вищої освіти та учнів шкіл на мові програмування JavaScript.

Основними завданнями дослідження є:

1. Проаналізувати можливість покращення орієнтації в навчальному процесі за допомогою мобільних додатків;
2. Проаналізувати існуючі застосунки для планування навчання;
3. Визначити основні вимоги до застосунку;
4. Дослідити можливість використання мови JavaScript для розробки мобільних додатків;
5. Розробити мобільний додаток для покращення ефективності в навчальному процесі;

Наукова новизна роботи полягає в наступному:

1. Покращено алгоритм для планування навчання;
2. Встановлено, що JavaScript є досить перспективною мовою для розробки мобільного програмного забезпечення;
3. Розроблено програмне забезпечення для покращення ефективності в навчальному процесі;

Практична значущість полягає у вирішені практичної задачі, а саме створення мобільного застосунку для зберігання та аналізу навчальної інформації на мові програмування JavaScript, який надасть студентам вищих навчальних закладів та учням шкіл цифрові інструменти для моніторингу власної успішності в навчанні та покращить орієнтацію в навчальному процесі. Даний застосунок зможе чітко та структуровано відображати навчальну інформацію, та аналізувати її. На основі аналізу, надавати рекомендації щодо покращення успішності користувача в навчанні.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предмету дослідження та програм для планування навчання

Як було зазначено у вступі, яскравим прикладом інструмента для забезпечення орієнтації в навчальному процесі є шкільний щоденник, який дає змогу учню занотовувати необхідну для навчання інформацію, таку як: розклад, оцінки, домашнє завдання та інше. Але в еру інформаційних технологій, електронні носії інформації швидко витісняють паперові.Тому в наш час доцільно створити мобільний додаток, який не буде поступатися функціоналу паперового щоденника, а навпаки перевершувати його та виконувати набагато ширший спектр завдань.

Паперовий щоденник може тільки надати інформацію але не аналізувати її. Тому тут можна виділити суттєву перевагу електронного щоденника, який би зміг не тільки відображати введені користувачем дані, а ще й проаналізувати їх та видавати певний результат роботи з ними. Також суттєвою перевагою електронного щоденника є спосіб представлення інформації. Додаток зможе компактно та структуровано відобразити данні, полегшуючи тим самим пошук необхідної для користувача інформації. Ці фактори суттєво впливають на покращення орієнтації в навчальному процесі, що є головною метою мобільного додатку.

Тема створення мобільного додатку, який буде вирішувати вище вказані задачі є досить актуальною, оскільки в кожного, хто навчається в будь-якому навчальному закладі, виникають певні проблеми з організацією власного процесу навчання, а розвиток інформаційних технологій сприяє створенню подібних інструментів в електронному форматі. Також кількість якісних мобільних застосунків, які б відповідали вище зазначеним критеріям є вкрай малою. Більшість з них надають тільки мінімальний набір функцій.

Застосунок розраховується на учнів шкіл та студентів ВУЗів, які мають на меті перенести навчальну інформацію в електронний вигляд.

Тому проаналізувавши все вищесказане, мобільний додаток повинен надати користувачу наступні можливості:

- створення власного розкладу занять;
- мати вбудовані калькулятори, для обчислення різних навчальних задач;
- надавати користувачу можливість слідкувати за своїми оцінками;
- відстежувати кількість пропущених навчальних днів;
- аналізувати успішність навчання користувача;
- мати простий та зрозумілий інтерфейс;
- записувати домашні завдання;

Особливість додатку від інших аналогів полягає в тому, що застосунок, аналізуючи оцінки з предметів, які користувач зможе виставляти в окремому розділі, надає інформацію про те, з якими предметами користувач має проблеми. Також даний додаток на основі вищевказаної інформації зможе аналізувати склад розуму користувача та надати список найбільш підходящих професій.

Створюючи мобільний додаток, потрібно врахувати операційну систему на якій буде працювати додаток. Згідно з даними https://gs.statcounter.com [18], найпопулярнішими мобільними операційними системами є Android та iOS. Станом на жовтень 2020 року кількість користувачів Android – 72.92%, iOS – 26.53%. Інші операційні систем мають менш як 1% користувачів. Якщо взяти статистику Android та iOS по Україні, то кількість користувачів Android становить 77.16%, iOS – 22.41%. Проаналізувавши вище зазначену статистику, можна прийти до висновку, що на даний час серед користувачів мобільних гаджетів найбільший відсоток тих, хто використовує ОС Android, тому раціонально створювати додаток для операційної системи Android

1.2 Аналіз існуючих аналогів

Попри те, що проблема орієнтування в навчальному процесі є досить розповсюдженою, наразі існує не так багато додатків для усунення цієї проблеми. Для створення якісного мобільного додатку проведемо ретельний аналіз

існуючих аналогів, розміщених в Play Market, порівняємо їх функціонал, особливості використання та врахуємо всі їх недоліки та переваги.

School – популярний на просторах сервісу Play Market додаток, який надає велику кількість інструментів для вирішення повсякденних завдань школяра. В додатку, окрім розкладу є наступні функції: додавання домашнього завдання, достатньо великий збірник теорії з різних предметів, створення запланованих подій, функція перегляду часу до кінця уроку та навчального дня, широкий вибір налаштувань додатку та інше. Додаток має понад п’ять мільйонів завантажень. Додаток School зображений на рисунку 1.1.

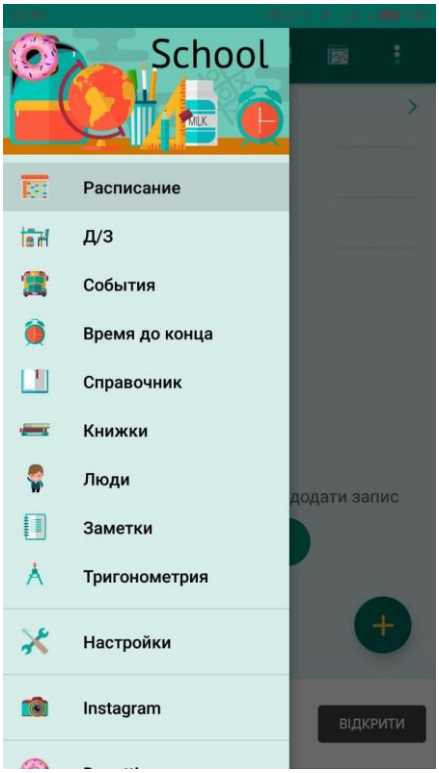


Рисунок 1.1 – Додаток School

Незважаючи на велику кількість завантажень, School має також багато недоліків. Найбільш неприємний недолік даного додатку – велика кількість реклами, яка з'являється майже при кожній взаємодії з інтерфейсом. Наприклад, якщо учень хоче швидко записати домашнє завдання, перехід на дану вкладку в додатку може супроводжуватись появою на екрані рекламного блоку, який триває певну кількість часу, що в даному випадку є неприпустимо.

Аналізуючи відгуки користувачів застосунку School в Play Market, можна прийти до висновку, що в додатку є проблема частих вильотів або помилок з базою даних додатку, які проявляються в тому, що останні збережені користувачем дані при наступному запуску додатку зникають з нього. На екрані розкладу в предметів немає таких полів як вчитель, час початку та кінця предмету, тому для отримання цієї інформації користувачу потрібно переходити на інші екрани.

Не дивлячись на недоліки, можна впевнено сказати, що School – один із найкращих додатків в своєму роді. Перш за все, великою перевагою School є велика кількість різних функцій, які майже повністю забезпечують потреби користувача. Додаток приваблює не тільки широкою кількістю інструментів, але й привабливим інтерфейсом. Великою перевагою є переклад додатку на більш ніж як 20 мов, що дає змогу користуватись додатком користувачам з різних країн світу.

Light School – мобільний додаток, який дає змогу користувачу створювати власний розклад занять та записувати домашні завдання. Додаток Light School зображено на рисунку 1.2.

Рисунок 1.2 – Додаток Light School

Програма має достатньо швидкий та простий інтерфейс, функцію вибору стартового екрана, та зміну теми інтерфейсу на свій смак. Є також екран, де можна встановити час початку та закінчення кожного заняття, є декілька вбудованих калькуляторів для вирішення деяких математичних задач. Незважаючи на те, що додаток виконує свої основні функції, він має певний перелік недоліків на який треба звернути увагу.

Перш за все потрібно зауважити те, що при створюванні нового заняття, користувач не може вказати його початок та кінець, а отже переглядаючи розклад, неможливо одразу дізнатися початок або кінець заняття, для цього потрібно перейти на інший екран додатку, що викликає певні незручності.

Іншим вагомим недоліком є те, що на екрані зміни мови представлено недостовірну інформацію (рис. 1.3), яка стосується української та білоруської мови, зокрема останні показані на екрані, як частини російської мови, а не як окремі мови. Також при зміні мови на українську або білоруську, мова інтерфейсу змінюється на російську, що не є коректним. В додатку присутній розділ з теоретичною інформацією таких предметів як: алгебра, геометрія, інформатика та російська мова, але кількість теорії, що надається користувачу є невеликою, або є такою, що не несе достатньої інформаційної цінності.

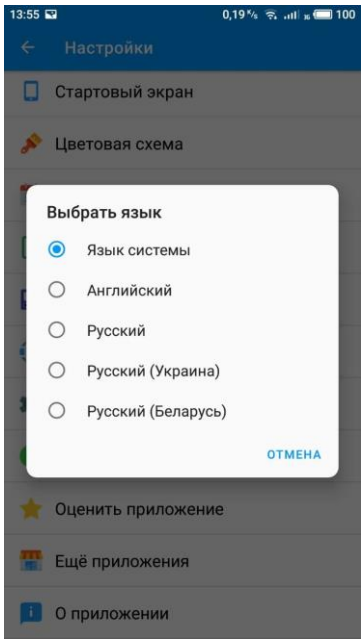


Рисунок 1.3 – Зміна мови інтерфейсу в LightSchool

До переваг LightSchool можна віднести повну відсутність реклами, стабільність, високу швидкодію, простий в освоєнні інтерфейс.

Class Timetable, як і вказані вище додатки, надає базовий функціонал, для оптимізації навчального процесу користувача, але порівнюючи його з розглянутими вище додатками, Class Timetable значно поступається своїм конкурентам.

На відміну від застосунку Light School користувач може вказувати час початку предмету та його кінець, а також задавати колір предмету, що є перевагою. Користувач може вказати декілька тижнів з різним розкладом, якщо розклад в навчальному закладі є не постійним. Плюсом також можна виділити відсутність реклами та підтримка як Android так і iOS.

До недоліків можна віднести: достатньо невеликий функціонал, який не забезпечує повністю сучасні потреби користувача, відсутність вибору іншої мови (присутня тільки російська). Додаток Class TimeTable зображений на рисунку 1.4.

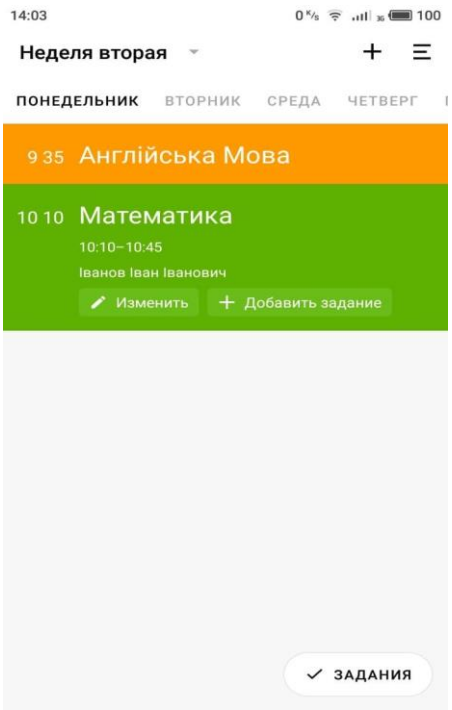


Рисунок 1.4 – Додаток Class TimeTable

Ще одним популярним застосунком даного класу додатків є School Planer (рис. 1.5).

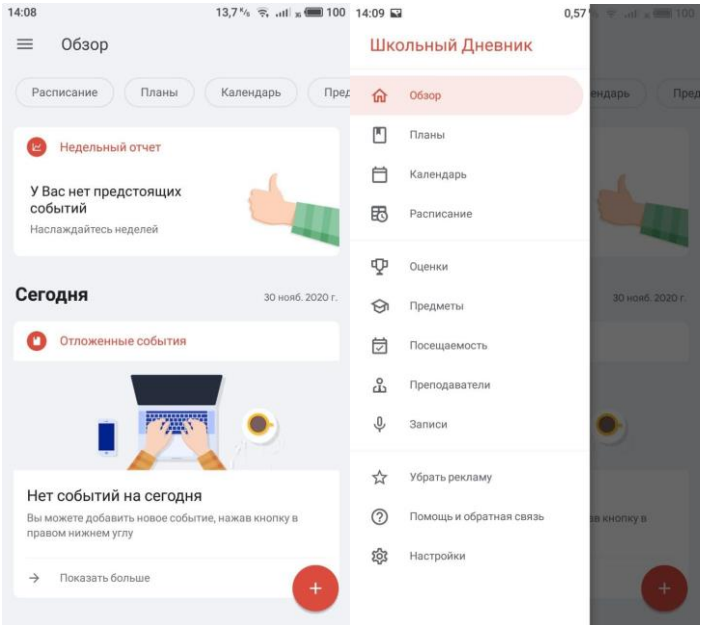


Рисунок 1.5 – Додаток SchoolPlanner

Застосунок постійно оновлюється, поповнюючись новим функціоналом. В додатку присутній розклад занять, можливість додавати пропущені навчальні дні, розділ з викладачами, де користувач зможе записати ім’я, прізвище, телефон та іншу інформацію. Широкий вибір налаштувань зможе налаштувати інструменти додатку під вимоги користувача. В застосунку реалізована функція push-повідомлень, які повідомляють про додані користувачем події у заданий час. Можливість резервного копіювання даних та їх відновлення. Застосунок добре оптимізований та не завантажений рекламою.

Недоліком можна вважати відсутність вибору інших мов інтерфейсу крім російської, що значно знижує коло користувачів. Інтерфейс перенасичений різними функціональними компонентами, які виконують однакову дію або мають низький рівень корисної дії. В додатку присутній розділ «Записи», основна функція якого – створювати голосовий запис. Даний інструмент є зайвим та не несе ніяку практичну значущість, адже в більшості сучасних смартфонів присутня функція диктофону, яка виконує аналогічну функцію. Крім того, в перевагах цього додатку було вказано, що додаток має функцію push-повідомлень. Недоліком тут можна вважати те, що push-повідомлення працюють тільки тоді, коли користувач знаходиться безпосередньо в додатку.

Крім вище проаналізованих аналогів, в Play Market можна знайти застосунки зі схожими функціями, але виділені вище додатки вирізняються набагато більшим функціоналом, та мають достатньо велику кількість завантажень. Основні недоліки та переваги існуючих аналогів наведено в таблиці 1.1.

Таблиця 1.1 - Переваги та недоліки існуючих аналогів

Назва додатку	Переваги	Недоліки
School	- таймер до початку та кінця уроку; - велика кількість теорії з різних предметів; - переклад інтерфейсу на понад 20 мов світу;	- часті вильоти додатку; - велика кількість реклами; - можлива втрата даних; - відсутність системи оцінок
Class Timetable	- відсутність реклами; - вибір кольору предмету в розкладі; - повна відсутність реклами; - вибір кольору предмету в розкладі; - повна відсутність реклами;	- невеликий функціонал; - мова інтерфейсу тільки російська;
School Planer	- можливість моніторингу відвідуваності школи/університету; - система оцінок; - широкий вибір налаштувань;	- відсутність інших мов крім російської; - перенасиченість інтерфейсу компонентами, які виконують одну і ту ж саму функцію;

Продовження таблиці 1.1

Назва додатку	Переваги	Недоліки
Light School	- повна відсутність реклами; - приємний та простий інтерфейс; - можливість встановити стартовий екран власноруч;	- відсутність можливості перегляду початку та кінця заняття на екрані розкладу; - при зміні мови на українську або білоруську, мова інтерфейсу змінюється на російську; - відсутність системи оцінок;

Враховуючи всі існуючі аналоги в сервісі Play Market можна прийти до висновку, що не зважаючи на велику кількість даного класу додатків, більшість з них надають тільки мінімальні базові можливості, такі як: створення розкладу занять, домашнього завдання. Але в наш час потреби тих, хто навчається стали набагато більшими та вийшли далеко за коло базових функцій. Більшість проаналізованих вище додатків частково вирішують цю проблему, значно розширюючи коло можливостей для користувача, але в ході аналізу їх функціоналу було виявлено, що жодний додаток не надає можливості моніторингу власної успішності в навчанні, яка могла би бути представлена у вигляді графіків або в якомусь іншому форматі, яка вказувала би користувачу на його слабкі та сильні сторони в навчанні. Також в додатках майже відсутні вбудовані калькулятори для обрахунку різних навчальних завдань, які б могли підтвердити правильність обрахунків користувача з різних навчальних предметів. Тому створюючи додаток, який буде покращувати навчальний процес користувача, потрібно врахувати всі вище зазначенні проблеми, недоліки та переваги існуючих аналогів.

2 ВИБІР МОВИ ТА ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір мови програмування

Сьогодні існує велика кількість мов програмування, які можуть використовуватись як платформа для розробки мобільних додатків. Найпопулярнішими на даний час можна вважати такі мови програмування: Java, Kotlin, Swift, JavaScript, Dart але враховуючи велику популярність та швидкий розвиток JavaScript було прийнято рішення, що мобільний додаток Student Assistant буде розроблений саме на цій мові програмування, а саме на бібліотеці React Native.

JavaScript – це динамічно типізована мова програмування, яка є імплементацією стандарту ECMAScript [3]. Розроблена Брендоном Ейхом в 1995 році. Приклад коду на JavaScript зображений на рисунку 2.1.

```
import {rewriteAsyncPeopleData} from "../../store/asyncMethods/methods"
export const peopleReducer = (state = initialState, action) => {
    switch (action.type) {
        case LOAD_PEOPLE:
            return {
                ...action.payload,
            }
        case ADD_PERSON:{
            const newItem = {
                id: new Date().getTime(),
                name: action.payload.name,
                subject: action.payload.subject,
                birthday: action.payload.birthday,
                phone: action.payload.phone,
                type: action.payload.type
            }
            if(action.payload.type == 'teacher'){
                const newData = {
                    ...state,
                    teachers:[...state.teachers, newItem],
                }
                rewriteAsyncPeopleData(newData)
                return {
                    ...newData
                }
```

Рисунок 2.1 –Код на JavaScript

В JavaScript поєднується як об'єктно-орієнтований так і функціональний підхід до програмування. Це забезпечується наявністю в мові класів та

наслідування, які є основою об'єктно-орієнтованих мов програмування а також наявність анонімних функцій, замикань, каррінгу, які властиві для функціонально-орієнтованих мов. JavaScript складається з трьох основних компонентів: ядро мови, об'єктна модель браузера, об'єктна модель документа.

Об'єктна модель браузера (англ. Document Object Browser) – це прошарок між ядром та об’єктною моделлю документа. Її основною функцією є забезпечення керування компонентами браузеру. Керованими компонентами є: вікна браузеру, діалогові повідомлення, робота з HTTP, фрейми.

Об'єктна модель документа (англ. Document Object Modal) – це API для роботи з XML та HTML документами [9]. HTML або XML документи являють собою набір взаємопов’язаних елементів, які будують каркас всього документу. Кожен елемент має певні властивості, за допомогою яких, можливо проводити різні операції над самим елементами, наприклад: додавання або видалення елементу з документу, отримання самих елементів, редагування, зміна взаємозв’язків та поведінки між ними .

JavaScript вийшов далеко за межі браузерів. Сьогодні на JavaScript розроблюють не лише веб-сайти, а й мобільні додатки, застосунки для Windows, Mac OS та Linux. Найпопулярнішою бібліотекою для розробки мобільних додатків на JavaScript є React Native.

2.2 Технологія React Native

React Native – це платформа для розробки кроссплатформенних мобільних додатків, яка побудована на базі бібліотеки React [5]. React Native має велике технологічне співтовариство а також велику кількість бібліотек від інших розробників, що дає змогу швидко написати достатньо складні мобільні додатки. Використовуючи готові бібліотеки для React Native, розробник створює додаток з готових блоків. Не зважаючи та те, що дана бібліотека є досить молодою, вона є одною з найпопулярніших платформ для розробки кроссплатформенних мобільних

додатків для Android та iOS. Серед переваг даної бібліотеки можна виділити наступне:

- кроссплатформеність: розробнику майже не потрібно звертати увагу на адаптацію під іншу мобільну ОС, оскільки, код який написаний на React native на 80-90%складається із загальних правил як для Android так і для iOS;

- велике співтовариство розробників: з кожним роком кількість перевірених бібліотек, розроблених для React native, стає все більшою. Співтовариство активно розвиває бібліотеки, вирішуючи ще більше проблем пов'язаних з розробкою мобільних додатків;

- підтримка TypeScript: оскільки JavaScript не є строго типізованою мовою програмування, TypeScript суттєво зменшує кількість імовірних помилок;

- швидке оновлення: швидке оновлення–це можливість React Native, яка при зміні коду виконує оновлення компоненту, який змінився, що значно пришвидшує розробку додатку та дає змогу швидко відловлювати імовірні помилки роботи додатку;

React Native використовує компонентний підхід до розробки мобільних додатків. Це означає що компоненти можуть працювати незалежно один від одного та використовуватись в додатку повторно. Компоненти можуть бути представлені у вигляді функцій, які завжди повинні повертати JSX розмітку [10].

JSX – це розширення JavaScript, яке дозволяє використовувати HTML-подібний синтаксис безпосередньо в JavaScript документі. JSX дає краще розуміння структури документу, а також значно покращує читабельність коду. На відміну від React, який для відображення графічних компонентів використовує HTML-теги, React Native в свою чергу використовує власні нативні компоненти, які інтегруються в середовище операційної системи для якої розроблюється додаток. Для виводу нативних компонентів, на екран та їх обробку React Native використовує власне ядро або так званий «Міст» (англ. Bridge), який встановлює з'єднання між двома потоками, які має кожен React Native-додаток. Основний потік відповідає за відображення графічних компонентів та обробку їх подій. Інший

потік відповідає за аналіз JavaScript коду, та логічних взаємодії між компонентами системи.

Для стилізації JSX-компонентів React Native використовує абстракцію схожу з каскадними таблицями стилів (CSS), яка являє собою JavaScript об’єкт з ключами (CSS-властивості) та їх значеннями. Слід зауважити, що на відміну від звичайного CSS, ключі повинні бути написані Camel Case стилем, оскільки тире, яке використовується в CSS не проходить перевірку синтаксису JavaScript.

Способи збільшення швидкодії додатку реалізовуються за допомогою внутрішніх механізмів React Native. React Native як і React використовує Virtual DOM.

Virtual DOM або VDOM – це набір алгоритмів для оптимізації побудови структури графічного інтерфейсу, який насправді являє собою спрощену копію звичайної об’єктної моделі документа (DOM) [8]. Оскільки кожен вузол звичайної DOM має десятки властивостей, а сучасні як веб-сайти так і мобільні додатки можуть містити тисячі таких вузлів, то виникає проблема швидкості відображення таких великих структур. VDOM створює копію структури звичайної DOM, елементи якої мають тільки основні властивості. Після зміни будь-якого елементу документу, VDOM синхронізується зі справжнім DOM та відображає обновленні компоненти. Але слід зауважити, що хоч React Native і використовує алгоритми побудови документу які притаманні VDOM, елементи які відображаються в документі – це нативні компоненти операційної системи, які не мають нічого спільного з елементами звичайної DOM. Механізм оптимізації роботи додатку полягає в тому, що VDOM обновлює тільки ті компоненти, в яких відбулись зміни, а не весь документ. Це забезпечується порівнянням попередньої та поточної структури документу, та змін що відбулись в документі.

2.3 Менеджер станів Redux

Кожен компонент може містити якісь данні, які зберігаються на стороні клієнта. Ці дані можуть бути відображені в документі, змінені та відправленні на

сервер або в локальну базу даних. Кожен компонент може приймати ці данні та передавати їх нижче по структурі, але чим складніше та глибше стає структура документу, тим складніше передавати дані між компонентами. Цю проблему вирішує технологія Redux.

Redux – це бібліотека, яка дозволяє зручно керувати даними з будь-якої точки застосунку [2]. Бібліотека була розроблена Данилом Абрамовим та Ендрю Кларком в 2015 році. Redux використовує шаблон проектування «Спостерігач» [4], який базується на спостережені об’єктом змін інших об’єктів батьківського класу. Схема роботи Redux зображена на рисунку 2.1. В якості глобальної сутності для зберігання даних Redux використовує так званий «стан» (англ. state), який представлений у вигляді звичайного JavaScript-об’єкту. В Redux стан є доступним для всіх компонентів застосунку, але для цього потрібно обернути всі елементи застосунку в компонент Provider, який приймає початковий стан додатку.

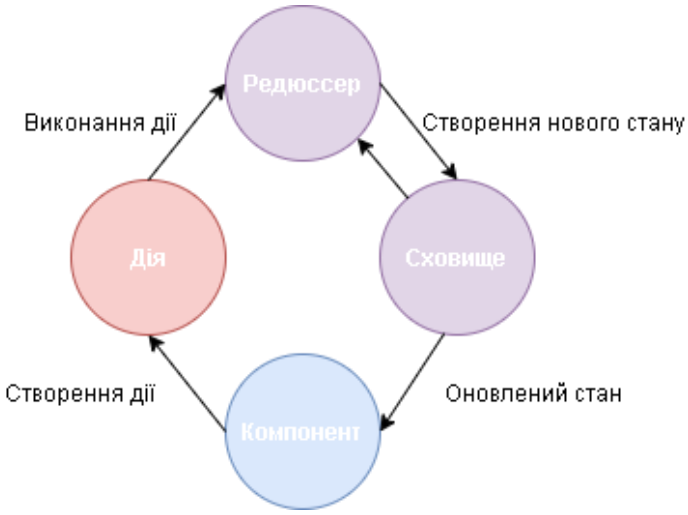


Рисунок 2.2 – Схема роботи Redux

Як в React так і в React Native неможливо прямо змінювати state, оскільки при звичайному присвоюванні нового значення об’єкту, ядро React Native не оновлює компоненти, які змінились. Це пояснюється тим, що React Native при кожній зміні вхідних даних порівнює попередній state та оновлений state, і лише після цього на основі порівняння двох станів оновлює ті компоненти, які використовують оновленні дані. При звичайному присвоюванні, перезаписується попереднє

значення state, а отже React Native не зможе вирішити, які компоненти слід оновлювати. Для зміни state можна використовувати вбудовані методи React Native, але їх доцільно використовувати тільки тоді, коли вони змінюють локальний state, який знаходиться в самому компоненті.

Використовуючи Redux, зміна state відбувається за допомогою так званих редюсерів (англ. Reducers).

Редюсер – це звичайна JavaScript функція, яка на вхід приймає попередній state, та дію (action) [13], яку потрібно провести над state. Action містить дві властивості: тип дії (action type), оновлені дані (payload). Action type вказує редюсеру, що конкретно він повинен зробити зі state. Payload – це оновленні дані які передаються в редюсер. Ці дані можуть бути представлені у вигляді будь-якого типу даних. Редюсер повинен завжди повертати state, навіть якщо ніяких змін з даними не відбулося.

Кількість дій над state зазвичай більше одного, тому для того щоб виконати певну дію зі state в залежності від action type, використовують логічний оператор switch. Кожен action type являє собою рядковий тип даних, який описує дію над state та зазвичай пишеться великими літерами. Кожний action type є константою, тому не може змінюватись в коді.

Для того щоб передати action type та payload в редюсер, використовують функцію, яка в контексті Redux має назву action. Дана функція являє собою звичайну JavaScript функцію, яка в якості аргументів приймає payload та повертає об’єкт з action type та payload. Action функція може бути викликана тільки за допомогою вбудованої в Redux функції dispatch, яка приймає саму action функцію та вказує редюсеру на необхідності зміни state.

Сам Redux не має вбудованих інструментів для роботи з мережевими або асинхронними запитами. Для цих операції існує спеціальна бібліотека під назвою Redux thunk, яка додає певні розширення до action-функцій. Бібліотека додає можливість виконувати асинхронні дії всередині звичайних функцій, такі як: запити на сервер, робота з базами даних, робота з зовнішнім API, виконання планувальних JavaScript-функцій. Redux thunk додає так званий middleware, який є

прошарком між state та функцією dispatch. Кожна дія, яка змінює state, проходить черезmiddleware.

2.4 Фреймворк Expo

Expo – це набір інструментів та рішень для розробки мобільних додатків на бібліотеці React Native, який значно спрощує процес розробки на React Native, та дає вже готовий набір компонентів для інтеграції в будь-який мобільний додаток [19]. Expo надає готовий набір компонентів графічного інтерфейсу, які зазвичай доступні тільки як окремі бібліотеки.

Expo також має власний додаток командної консолі, який має назву Expo CLI. Expo CLI – це програмний інтерфейс, який дає змогу розробнику керувати інструментами Expo, та виконувати такі функції як:

- запуск локального серверу проекту;
- запуск додатку на симуляторі чи прямо на мобільному телефоні;
- можливість компіляції проекту в файли типу apk або ipa;
- створення Expo – проектів;
- моніторинг помилок під час компіляції;

Expo CLI вимагає встановлену останню версію Node.js, оскільки вся робота з командами, а також керування залежностями відбувається через термінал за допомогою JavaScript. Expo CLI надає розробнику можливість спостерігати результат як на емуляторі та і прямо в смартфоні, використовуючи мобільний клієнт Expo.

З'єднання між локальним сервером та мобільними клієнтом Expo для передачі змін в коді та відображення їх на клієнті можливе за допомогою USB-з'єднання або бездротового з'єднання. USB-з'єднання встановлює дротовий зв'язок між запущеним локальним сервером та смартфоном. Бездротовий зв'язок встановлюється за допомогою сканування QR-коду серверу. Сервер генерує спеціальний QR-код, який насправді є посиланням на адрес локального серверу проекту, та який зчитується мобільним клієнтом Expo. Після скануванняQR-коду,

сервер генерує JavaScript-файл який містить мініфікований код додатку та передає його на клієнт. При кожній зміні коду, сервер посилає запит на відображення актуальних даних клієнту Expo. Основний недолік бездротової емуляції додатку на клієнті Expo є достатньо велика затримка в передачі даних, яка змушує розробника чекати обновлення результатів змін коду, які відображаються на клієнті.

2.5 База даних Firebase

Redux є тимчасовим сховищем даних. Це означає, що після виходу із застосунку всі дані будуть втрачені. Тому постає задача підключення бази даних до застосунку. На сьогодні існує достатньо великий вибір баз даних, які надають, широкі можливості для зберігання та керування даними. Однією із найпопулярніших БД є Firebase.

Firebase – це хмарна, NoSQL база даних, яка зберігає дані в текстовому форматі JSON [7]. NoSQL означає, що спосіб зберігання даних в БД такого типу кардинально відрізняється від звичної нам реляційної моделі зберігання даних. ПЗ. Сторінка з БД застосунку в Firebase зображена на рисунку 2.3.

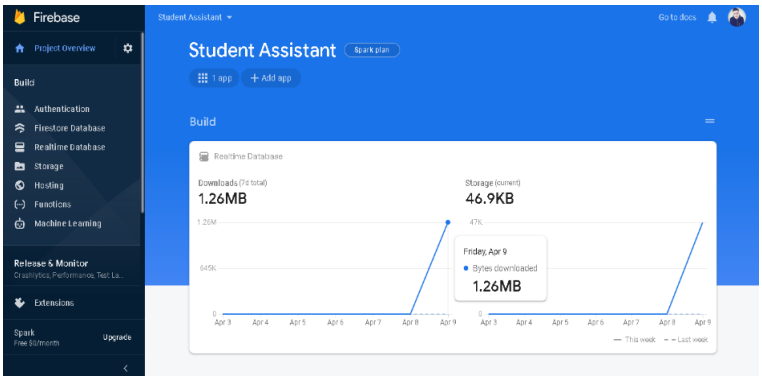


Рисунок 2.3 – Сторінка БД «Student Assistant» в Firebase

Якщо пояснити більш детальніше, то нереляційні бази даних використовують модель зберігання даних, яка оптимізується під конкретні вимоги той чи іншої системи, наприклад, пара “ключ-значення” або дані в текстовому форматі JSON або у вигляді графу. Також ще одною перевагою NoSQL баз даних є те, що розробник

взаємодіє з БД через API, тобто такий спосіб роботи з БД не вимагає знання мови SQL. Хоч Firebase і хмарна БД, вона може працювати і офлайн. При відсутності зв’язку з інтернетом, всі зміни в БД будуть закешовані на локальне сховище пристрою та будуть синхронізовані з хмарною БД при відновленні з інтернетом. Головною перевагою Firebase можна виділити швидкість розробки додатків та сайтів, тому що розробнику не потрібно звертати увагу на всі тонкощі роботи серверної частини, оскільки вони вже реалізовані в Firebase, і розробнику залишається лише інтегрувати їх до розроблюваного застосунку. Тому розробник може сконцентруватися на розробці інтерфейсу та бізнес логіки проекту. Всі зміни в БД Firebase відбуваються в режимі реального часу. Firebase має як платний функціонал та і безплатний.

2.6 Вибір інструментів розробки

Atom – один із популярних редакторів коду, розроблений GitHub Inc, який має підтримку Node.js-плагінів. Більшість плагінів розроблюється співтовариством, та мають статус вільного програмного забезпечення. Особливістю даного редактора коду є велика кількість налаштувань, що дає змогу користувачу налаштувати середовище розробки під власні потреби. Головним недоліком даного редактору коду є зниження продуктивності його роботи та збільшення потреби в ресурсах комп’ютера під час розробки великих проектів. Atom є повністю безкоштовним.

Visual Studio Code – безкоштовний редактор коду розроблений компанією Microsoft [14]. На відміну від Atom, Visual Studio Code має достатньо великий вбудований набір можливостей для розробки, без потреби встановлення додаткових плагінів. Редактор підтримує велику кількість мов програмування, такі як: C, C++, Java, Ruby, C# та інші [11]. Visual Studio Code володіє технологією автодоповнення IntelliSense, вбудовані засоби рефакторингу коду та можливість стилізації графічного інтерфейсу. Має вбудований термінал та консоль.

Налаштувати редактор можна, вносячи зміни в конфігураційні файли, які мають формат JSON. В редактор вбудована система контролю версій Git.

Microsoft використовує вбудовану в Visual Studio Code функцію телеметрії, яка аналізує та збирає персональні данні про використання додатку. Телеметрію можливо відключити в налаштуваннях редактору, але деякі можливості додатку стануть недоступними для використання. Перевагами додатку є інтуїтивно зрозумілий інтерфейс, велика кількість розширень, та налаштувань, його стабільність та швидкість при розробці великих проектів.

WebStorm – це платформа для розробки проектів на JavaScript, перший випуск якої відбувся в 2010 році. Платформа розроблена компанією JetBrains, яка є також розробником відомого середовища для розробки програмного забезпечення IntelliJ IDEA. В редакторі є вбудована технологія, яка обновлює результат в браузері під час редагування HTML, CSS або JavaScript коду. Редактор вміє працювати з платформою Node.js, та підтримує функції для редагування додатків на JavaScript такі як: рефакторинг коду, автодоповнення, перевірка на синтаксичні помилки. WebStorm підтримує всі сучасні фреймворки для розробки графічних інтерфейсів такі як: Angular, Vue, React, React Native. Але на відміну від Visual Studio Code та Atom, WebStorm є платним.

Отже, провівши аналіз можливостей вище перерахованих платформ для розробки мобільного додатку Student Assistant , був вибраний редактор Visual Studio Code, оскільки він безкоштовний, має широкий набір функцій для покращення процесу написання програмного коду, має хорошу швидкодію для роботи з великими проектами та має велике співтовариство розробників, які створюють велику кількість розширень для покращення роботи з даним додатком.

2.7 Інтерфейс та можливості Visual Studio Code

За замовчуванням вигляд інтерфейсу VS Code складається з таких компонентів як: вікно Explorer, область для роботи з кодом, консоль, термінал (рис. 2.4).

Вікно Explorer містить п’ять основних вкладок, та вікно для відображення їх вмісту. Кількість відображених вкладок може залежати від розширень, які були встановлені в середовище розробки. Іконки папок та файлів можуть бути стилізовані в залежності від їх вмісту. У вікні Explorer присутня вкладка як схожа з системою контролю версій Git. Вона відображає останні зміни внесені в файли проекту. Зеленим кольором заповнюються ті ділянки коду, які не були змінені. Відповідно красний колір відображає останні зміни в програмному коді. Вкладка «Run» відповідає за налагодження програми та перехоплення помилок.

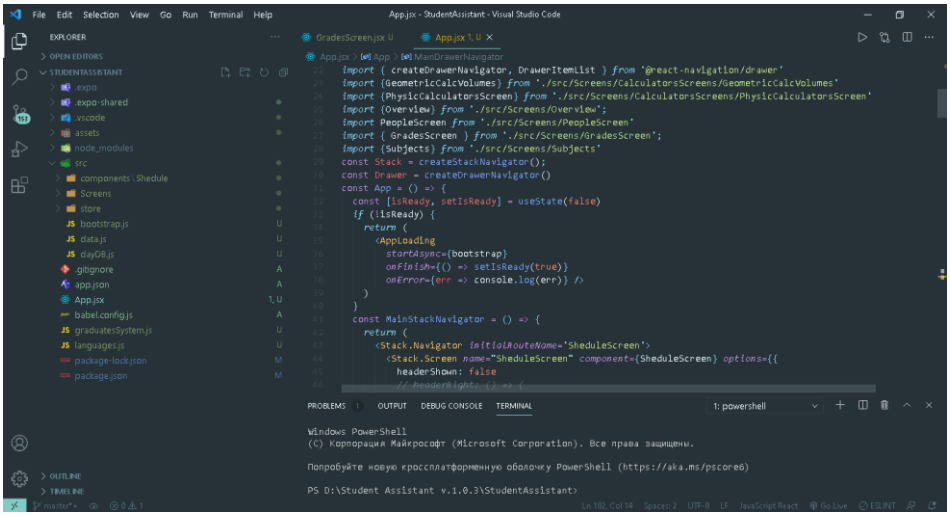


Рисунок 2.4 – Інтерфейс VS Code

Основний принцип роботи даного інструменту полягає в тому, що розробник може ставити в коді так звані точки зупину, які призупиняють виконання коду в тому місці де була поставлена точка. Під час зупинення процесу виконання програми, розробник може переміщатись послідовно по коду та відслідковувати послідовність та правильність виконання дій в коді. Це полегшує процес знаходження імовірних помилок в коді. Остання вкладка відображає плагіни, які можна встановити та які вже встановленні. Плагіни можуть бути відключенні, або налаштовані під потреби розробника. Вікно для роботи з кодом має достатньо широкий вибір функцій. Зверху на робочою областю розташовуються вкладки, за допомогою яких можна переміщатись між файлами проекту. Робоча область може бути поділена між файлами, що збільшує швидкість розробки. Знизу від робочої

області розташовується термінал та консоль, які дають змогу розробнику працювати з пакетними менеджерами, файловою системою або запускатиNode.js. Присутня можливість клонувати термінали для одночасного виконання різних задач, пов’язаних із запуском проекту або його збірки.

2.8 Vs Code плагіни

Visual Studio Code надає можливість розширювати власний функціонал, встановлюючи додаткові плагіни. Плагіни є абсолютно безкоштовними та завжди доступними для використання. Найпопулярнішими плагінами для розробки як веб сайтів так і мобільних додатків є: ESLint, Live Server, Prettier, Сніппети.

ESLint – це інструмент, який аналізує якість коду написаного на JavaScript, та виправляє більшість синтаксичних помилок. Розробник може задавати власний стиль написання коду. ESLint перевіряє синтаксичні конструкції в режимі реального часу, тобто під час написання коду. Стилі та правила повинні бути вказані в спеціальному конфігураційному файлі з розширенням eslintrs.

Плагін Live Server дозволяє автоматично оновлювати документ при будь-яких змінах в коді. Це дозволяє значно прискорити процес розробки на JavaScript. Live Server також має набір конфігураційних параметрів, за допомогою яких, можливо налаштувати даний плагін під власні вимоги, наприклад виставити власний інтервал між оновленням документу.

Prettier – один із найпопулярніших інструментів для форматування коду. Він задає чітко фіксовані правила для оформлення коду. Особливо корисно використовувати даний плагін в командній роботі, оскільки він приводить код до одного стилю, однакого для всіх розробників в команді [15]. Як і вище зазначенні плагіни, Prettier може бути налаштований через конфігураційні файли. Конфігурація задається JavaScript-об’єктом, в якому ключі – це конфігураційні параметри.

Сніппети (англ. Snippets) – це інструменти для швидкого генерування коду. Даний вид плагінів містить в собі готові шаблони для побудови різних

синтаксичних структур, наприклад: функцій, класів, логічних конструкцій, тощо. Сніппети існують майже для всіх мов програмування, зокрема і для JavaScript та бібліотеки React Native. Згенерувати будь-яку синтаксичну конструкцію можна, вводячи спеціальні скорочення в коді. Знайти ці скорочення можна в документації, яка знаходиться в описовому вікні плагіну.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1 Проектування системи

Проектування в контексті розробки програмного забезпечення необхідне для формування чіткого плану дій на самому початку розробки ПЗ. Цей процес мінімізує імовірні втрати часу на неочікувані доопрацювання та дозволяє передбачити інші тонкощі під час розробки. Показниками вдало проведеного проектування є: відповідність заданим функціональним вимогам, узгодженість з певними обмеженнями, які накладаються обладнанням, ясність та простота архітектури системи. Одним із інструментів для забезпечення цього процесу є мова UML.

UML – це уніфікована мова для візуального моделювання компонентів інформаційної системи, бізнес-процесів та інших систем [6]. Моделювання в UML включає всі рівні проектування інформаційної системи, починаючи від загальної концепції, і закінчуючи фізичною моделлю. UML-моделювання реалізовується за допомогою побудови UML-діаграм, кожна з яких дає уявлення про архітектурні рішення та опис моделі в рамках заданої бізнес задачі.

Для опису концептуальної моделі в UML використовується діаграма прецедентів (англ. use-case diagram). Діаграма прецедентів візуалізує взаємозв’язок між діючими сторонами та прецедентами. Діючими сторонами виступають люди або інші системи, які взаємодіють з функціями системи. Прецедентами називають сутність, яка описує певний фрагмент поведінки системи з якою користувач може вступити у взаємодію. Іншими словами прецедент – це частина функціоналу застосунку або іншої інформаційної системи, з якою користувач може взаємодіяти.

Діаграма прецедентів описує загальну схему опису роботи системи. Важливо зберігати стислість при розробці даної діаграми, оскільки велика кількість даних зображених на діаграмі ускладнює розуміння роботи системи. Діаграма прецедентів не спрямована на те, щоб відобразити всі тонкощі роботи застосунку, а лише відобразити основні кроки роботи системи.

При проектуванні застосунку «Student Assistant», було розроблено діаграму прецедентів, яка ілюструє взаємодію компонентів додатку. Діаграма прецедентів зображена на рисунку 3.1.

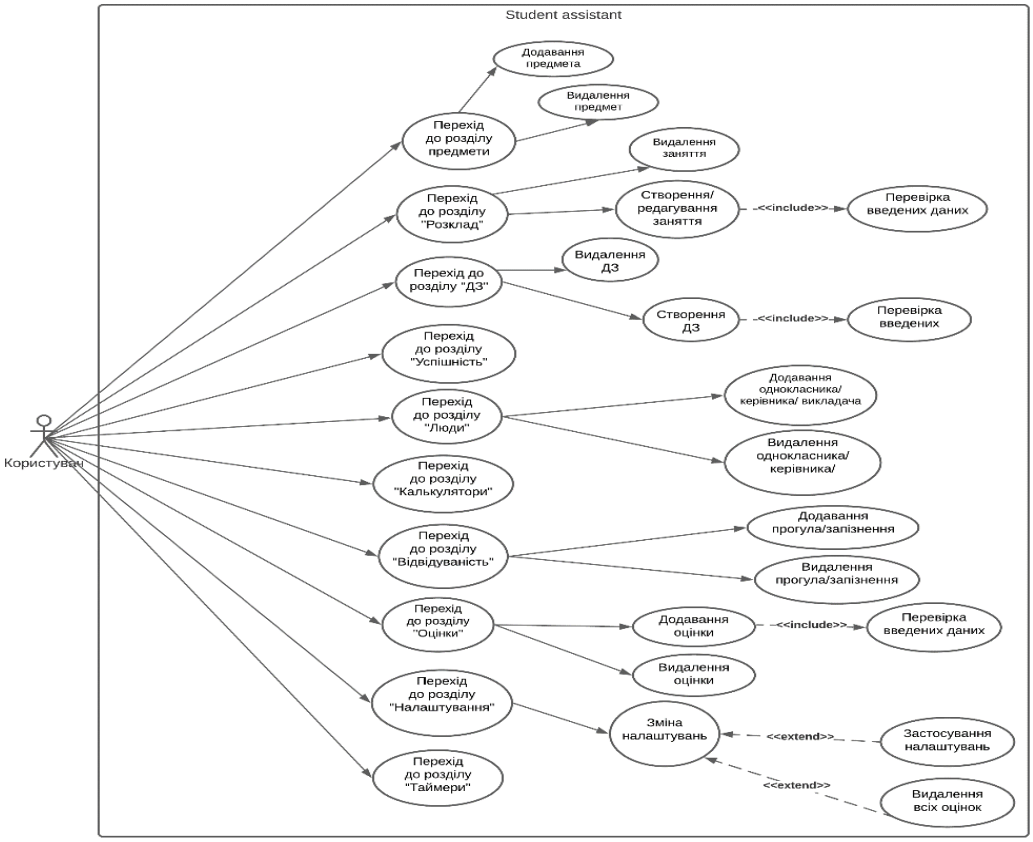


Рисунок 3.1 – UML діаграма прецедентів

Крім діаграми, яка відображає загальну ідею, UML включає діаграму, яка описує логічну модель системи. Для ілюстрації логіки, поведінки та послідовності виконання програми використовується діаграма діяльності.

Діаграма діяльності – це UML діаграма, яка відображає динамічні механізми поведінки системи в цілому або її окремих компонентів. На вигляд діаграма виглядає як звичайна блок схема, яка показує, яким чином та за яких умов, потік виконання програми переходить від одної діяльності до іншої. Іншим словами діаграма діяльності ілюструє, яким чином користувач може взаємодіяти з той чи іншою функцією системи та до яких результатів такі взаємодії можуть призвести. Діаграми діяльності окремих компонентів застосунку Student Assistant зображені на рисунках 3.2 та 3.3.

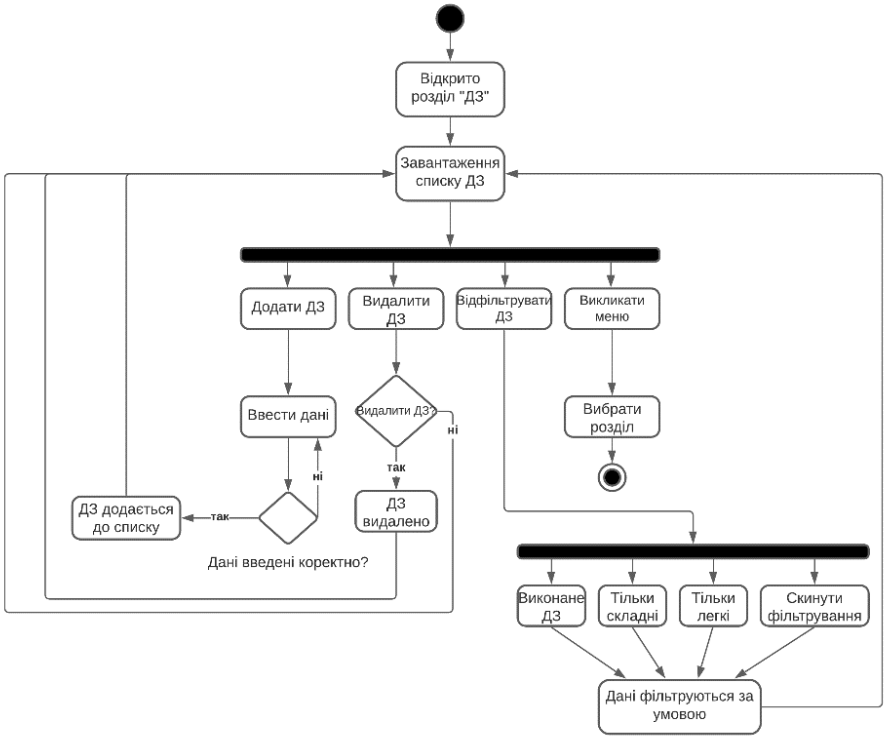


Рисунок 3.2 – Діаграма діяльності розділу «ДЗ»

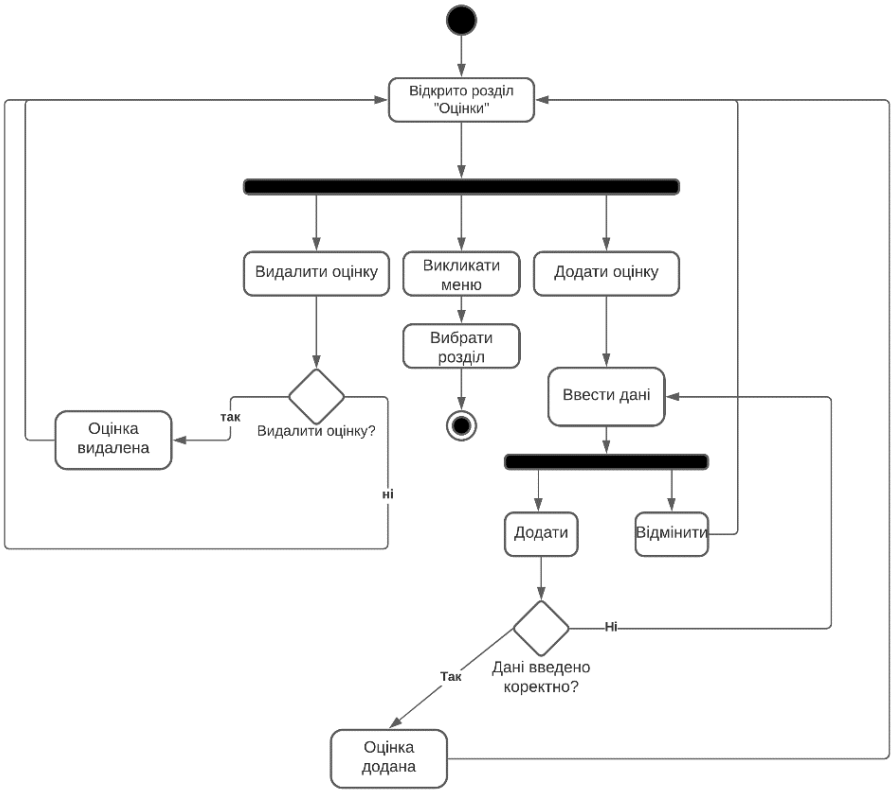


Рисунок 3.3 – Діаграма діяльності розділу «Оцінки»

3.2 Опис призначення додатку та його функціоналу

Призначення мобільного додатку Student Assistant полягає в тому, щоб забезпечити користувача цифровим інструментом для зберігання та аналізу навчальної інформації. Додаток повинен чітко та структуровано відображати інформацію введену користувачем.

Враховуючи те, що користуватись додатком можуть також школярі початкових класів, постає питання створення максимально простого та інтуїтивно зрозумілого інтерфейсу. Додаток не повинен бути перенасичений графічними компонентами та зайвим функціоналом.

Розглянемо більш детальніше функціонал застосунку та особливості його роботи. Додаток має десять розділів, кожен з яких має свій окремий набір функцій. Всі розділи представлені в меню, яке можна визвати двома способами. Перший – натискаючи на кнопку, яка присутня в кожному розділі застосунку. Другий – провести пальцем від лівого краю екрана до середини екрану. Застосунок містить наступні розділи: «розклад», «ДЗ», «оцінки», «відвідуваність», «калькулятори», «успішність», «таймери», «люди», «налаштування», «предмети».

У розділі «Розклад» користувач може створювати навчальні заняття в кожен день неділі, переглядати їх, редагувати або видаляти. Навчальні заняття представлені у вигляді блока, в якому є наступна інформація: предмет, викладач, номер аудиторії, час початку та кінець заняття. Цю інформацію користувач вказує у вікні створення заняття.

У розділі з «ДЗ» користувач може створювати домашні завдання видаляти їх та переглядати інформацію про них. Кожне домашнє завдання являє собою інформацію про предмет, з якого було задане завдання, опис завдання, термін до якого треба здати завдання та рівень складності. Також користувач може відмітити завдання, як виконане. Користувач може відфільтрувати усі завдання за певними умовами. Домашні завдання можуть бути відфільтровані або за рівнем складності або за статусом виконання. Створюються завдання в окремому модальному вікні з веденням необхідної інформації для створення завдання.

У розділі «Таймери» представлені два таймера. Перший таймер вказує на час до початку або кінця заняття. Таймер бере поточний час та знаходить заняття, яке попадає в поточний часовий діапазон, після цього виводить інформацію про предмет, який відповідає вищевказаній умові в окремому блоці. Якщо навчальний день закінчився, таймер виведе відповідний текст. Другий таймер вказує на відлік до кінця поточного навчального року. Для роботи останнього, необхідно власноруч вказати кінець навчального року.

Розділ «Оцінки» дає можливість виставляти оцінки. Розділ представлений у вигляді календаря та поля з оцінками. Календар слугує для вибору дати, на яку необхідно виставити оцінку. Поле з оцінками представлене у вигляді окремих блоків, де в кожному блоці є наступна інформація: опис оцінки (за що була виставлена оцінка), дата виставлення оцінки, бал. Також слід зауважити, що користувач може виставляти оцінки в розділі з домашніми завданнями, у вікні інформації про завдання.

Розділ «Люди» надає інформацією про викладачів та одногрупників або однокласників. В цьому розділі представлено чотири групи людей: класний керівник або староста, викладачі, одногрупники або однокласники, та інші. Інформація, яка подається , представлена наступними полями: ФІО людини, предмет(якщо людина - викладач) , номер телефону, день народження.

Розділ «Калькулятори» надає користувачеві набір інструментів для обрахунку різних математичних або фізичних. Калькулятори не показують докладний алгоритм обчислення, а лише дають кінцевий результат. Даний розділ створений для того, щоб користувач зміг перевірити правильність своїх розрахунків. Калькулятори представлені з таких предметів: алгебра, геометрія, фізика.

В розділі «Відвідуваність» користувач може вказувати всі випадки запізнення або пропуску навчальних днів. В розділі є дві шкали кількості запізнень та пропусків. Шкала з прогулами заповнюється в залежності від поточної кількості прогулів та максимальної кількості прогулів. Шкала із запізненнями працює аналогічним способом. Кожен прогул або кожне запізнення відображається в

окремих блоках, які розташовуються нижче шкал. В блоках розміщується наступна інформація: дата прогулу або запізнення, та причина.

Розділ «Успішність» містить три підрозділи: оцінки, предмети, склад розуму.

В підрозділі «Оцінки» представлені рекомендації про те, які оцінки слід негайно виправити, які оцінки можливо виправити, але не обов’язково, а також кругова діаграма, яка вказує числову пропорцію кількості оцінок. Дана діаграма явно вказує, які оцінки переважають в користувача.

Підрозділ «Предмети» вказує користувачу його слабкі та сильні сторони в навчанні у вигляді пелюсткової діаграми. Пелюсткова діаграма – зручний спосіб відображення даних з цілю встановлення та порівняння статистичних значень трьох та більше рядків даних [20]. В даному підрозділі в якості рядків даних виступають навчальні предмети, а в якості статистичних даних – сума всіх оцінок з кожного навчального предмету. Пелюсткова діаграма зображена на рисунку 3.4. Даний вид діаграм добре ілюструє користувачу, в яких предметах користувач має успіхи, а на які слід звернути увагу. Слід зауважити, що назви предметів на діаграмі скорочені. Скорочення для назви предмета користувач вказує при створені предмета в розділі «Предмети», який буде описаний далі. Також в даному підрозділі присутня інформація про топ три предмета з найбільшим середнім арифметичним значенням та топ три з найменшим. Середнє арифметичне значення обчислюється на основі оцінок введених користувачем з відповідних предметів.

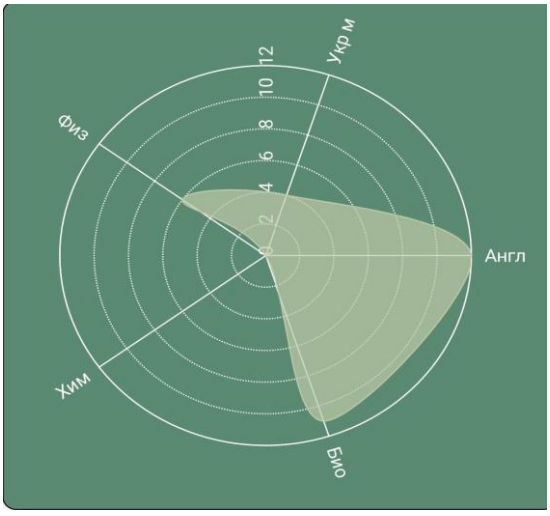


Рисунок 3.4 – Пелюсткова діаграма

В підрозділі «Склад розуму» представлена інформація про склад розуму користувача на основі його оцінок та середнього арифметичного з предметів.

Підрозділ містить кругову діаграму, яка вказує на співвідношення аналітичного складу розуму до гуманітарного у відсотковому форматі. На вищевказаних даних формується блок з описом переваг домінуючого складу розуму в користувача. В залежності від аналізу складу розуму, та оцінок з предметів користувача, застосунок надає список найбільш підходящих професій, якщо користувач не може визначитись зі своєю майбутньою професією .

В розділі «Налаштування» користувач може змінити мову інтерфейсу, вибрати систему оцінок та встановити максимальну кількість прогулів або запізнень. В додатку встановлено дві мови інтерфейсу: українська, англійська. Користувач має змогу вибрати одну із систем оцінок. Цей параметр дозволяє вибрати метод оцінювання, який використовується в навчальній установі користувача. В застосунку представлено три найбільш розповсюджені системи оцінок: оцінки від 1 до 12, від 1 до 5, від 0 до 100. Слід зауважити, що при зміні параметра додаток видаляє всі оцінки, записані користувачем.

З кожного розділу користувач може визвати бічне меню та перейти до іншого розділу.

3.3 Створення екранів за допомогою React Navigation

Для навігації між екранами додатку зазвичай використовується бібліотека React Navigation [17]. React Navigation створює оточення, в межах якого відбувається навігація між екранами. Слід зазначити, що бібліотека надає не тільки механізм навігації але й готові компоненти такі як: бічне меню, шапка екрану, кнопки для навігації. За механізмом роботи React Navigation дуже схожий зі звичайним React Router, який використовується для навігації між сторінками веб-сайту, побудованого на React. Весь застосунок являє собою деревоподібно структуру із React Navigation-компонентів, екранів та зв’язків між ними. Діаграма структури навігації застосунку зображена на рисунку 3.5.

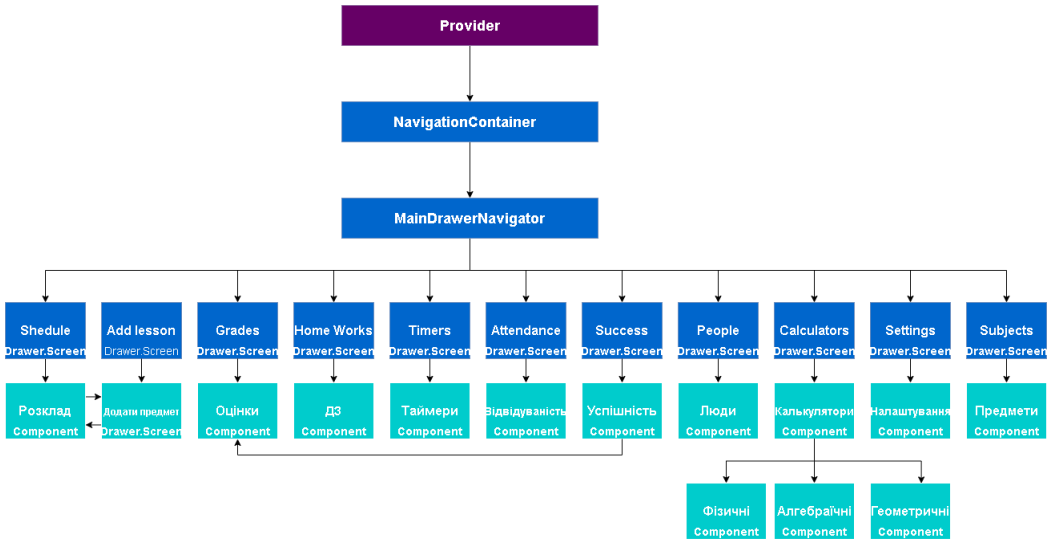


Рисунок 3.5 – Діаграма структури навігації застосунку

Provider – компонент бібліотеки Redux, який надає вкладеним вниз по дереву компонентам доступ до стану застосунку. В цей компонент обертається увесь застосунок. Компонент NavigationContainer являє собою оточення в межах якого відбувається переміщення між екранами застосунку. Компонент MainDrawerNavigator – функція, яка повертає компонент Drawer Navigator. Drawer Navigator – бічне меню, яке містить посилання на екрани.

Drawer Navigator складається з компонентів Drawer Screen. В компонент Drawer Screen передається певний перелік властивостей, наприклад властивість name задає назву екрану в рядковому форматі, яка є своєрідним посиланням на цей екран. Властивість componentприймає сам компонент екрану. В options задається стилізація посилання в Drawer Navigator (рис. 3.6).

Розглянемо приклад створення екрану з домашніми завданнями та прив’язки його до навігації застосунку.

Для початку створюється файл в корні проекту з назвою, яка повинна відповідати за зміст компоненту. Файл може бути або з розширенням jsx або js. Розширення js та jsx інтерпретуються компілятором однаково, але неявним правилом є те, що файли з розширенням jsx повинні містити тільки графічні компоненти та без імплементації будь якої бізнес-логіки. Далі у файлі

імпортуються бібліотека React, та компоненти для розмітки із бібліотеки React Native. Після цього об’являється сам компонент екрану, який має назву HomeWork. Компонент представляє собою функцію, яка повинна повертати jsx-розмітку.

```
const MainDrawerNavigator = () => {
  return (
    <Drawer.Navigator
      drawerStyle={{
        backgroundColor: 'white',
        borderBottomRightRadius: 20,
      }}
      drawerContent={props => CustomDrawer(props)}
      drawerContentOptions={{
        activeTintColor: 'white',
        activeBackgroundColor: '#adb5bd',
        inactiveTintColor: '#6c757d',
        labelStyle: {
          fontSize: 16
        }
      }}>
```

Рисунок 3.6 –Компонент MainDrawerNavigator

Для того, щоб додати компонент HomeWork до навігаційного оточення React Navigation, необхідно імпортувати даний компонент до головного компоненту App, яке коренем застосунку. Компонент HomeWork зображений на рисунку 3.7.

```
import React from 'react';
import {View, Text} from 'react-native'
export const HomeWork  = (props) => {
    return (
        <View>
            <Text>Home work screen</Text>
        </View>
    );
}
```

Рисунок 3.7 – Компонент HomeWork

Компонент HomeWork додається всередину MainDrawerNavigator, як компонент Drawer Screen (рис. 3.8) з наступними атрибутами:

- name являє собою назву компоненту, яке використовується при навігації, як посилання;
- component містить компонент, який необхідно завантажувати;
- options містить об’єкт з стилями для відображення в бічному меню;

Drawer Screen – компонент, який знаходиться в бічному меню застосунку, по кліку на який відбувається перехід на екран, який вказаний в атрибуті name.

```
const MainDrawerNavigator = () => {
  return (
    <Drawer.Navigator
      drawerStyle={{
        backgroundColor: 'white',
        borderBottomRightRadius: 20,
      }}
      drawerContent={props => CustomDrawer(props)}
      drawerContentOptions={{
        activeTintColor: 'white',
        activeBackgroundColor: '#adb5bd',
        inactiveTintColor: '#6c757d',
        labelStyle: {
          fontSize: 16
        }
      }}>

        <Drawer.Screen name='HomeWork' component={HomeWork} options={{
          title: 'Домашні завдання',
          drawerIcon: ({ focused, size }) => (
            <Image style={{ width: 20, height: 20 }} source={require('./assets/DrawerSheduleIcon.png')} />
          )
        }} />
```

Рисунок 3.8 – Додавання HomeWork до MainDrawerNavigator

Після всіх проведених вище маніпуляцій, можна перейти на сторінку з домашніми завданнями, визвавши бічне меню, та вибравши відповідний пункт в ньому.

3.4 Запуск додатку

Expo надає зручний спосіб для запуску застосунку як на віртуальному пристрої так і на реальному. Запуск застосунку безпосередньо на реальному мобільному девайсі зменшує кількість імовірних помилок при розробці та збірці проекту. Запуск додатку відбувається на мобільному клієнті Expo. Мобільний додаток Expo зображений на рисунку… Для запуску необхідно в терміналі проекту ввести команду expo start --localhost --android. Ця команда вказує Expo, запустити локальний сервер, який відкриє застосунок на смартфоні. Перед запуском серверу необхідно підключити смартфон до комп’ютеру по USB та увімкнути

налагодження по USB на самому смартфоні. Після вище описаних дій в браузері відкриється вкладка з панеллю керування (рис. 3.9).

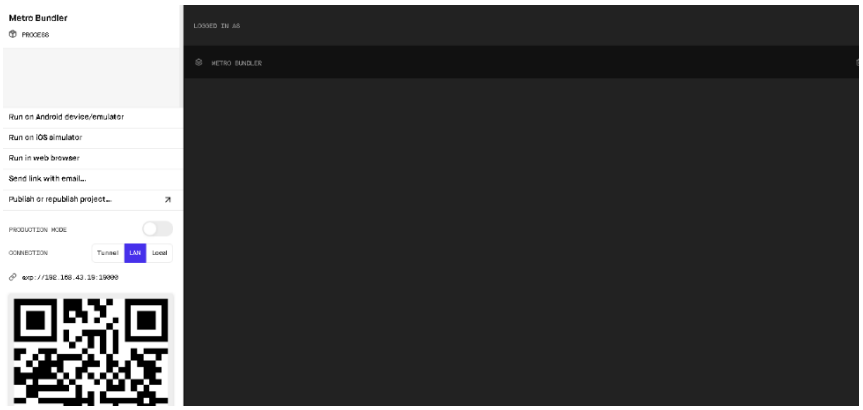


Рисунок 3.9 – Панель керування Expo.

На панелі представлено консоль розробника, та способи запуску застосунку, які знаходяться в бічному меню панелі. Розробник може запустити LAN сервер, який дозволяє запустити безпровідний зв'язок між клієнтом та сервером. Для безпровідного зв’язку у вкладці Connection треба увімкнути LAN та за допомогою камери яка вбудована в клієнт Expo сканувати QR-код.

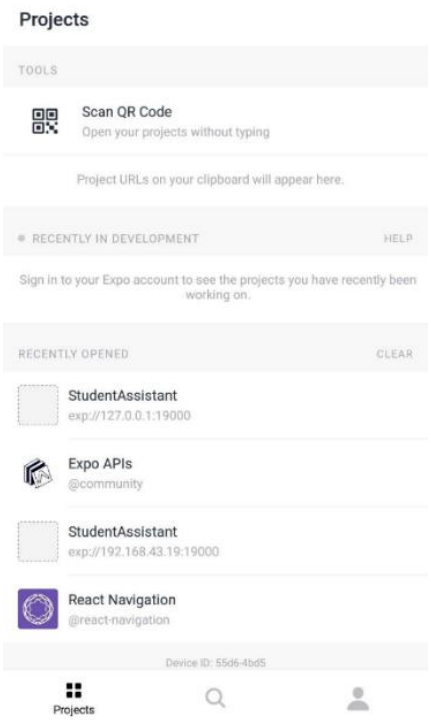


Рисунок 3.10 – Мобільний клієнт Expo

3.5 Створення бази даних Firebase

Створення бази даних відбувається безпосередньо на сайті Firebase [16]. Для початку необхідно вказати назву проекту(рис.3.11).

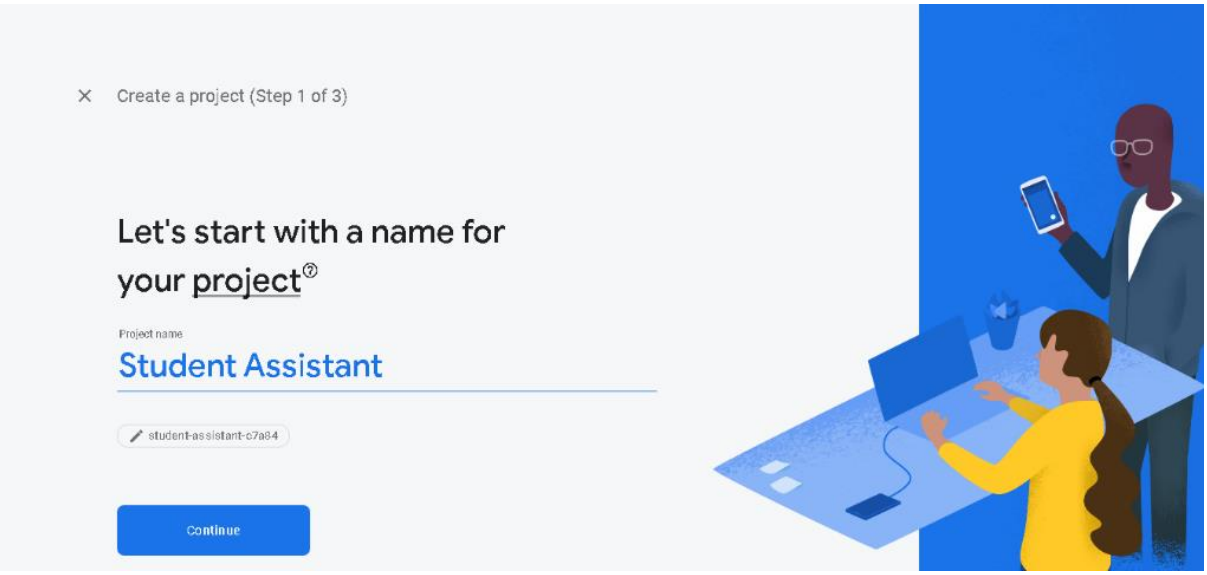


Рисунок 3.11 – Ініціалізація проекту в Firebase

Після ініціалізації проекту, відкриється головне меню проекту в якому необхідно перейти на вкладку «Realtime database», та позначити застосунок як «Web». Слід звернути увагу, що застосунок створюється, як веб додаток, тому що Firebase не підтримує пряму інтеграцію з React Native. Після виконання всіх вищеописаних дій Firebase згенерує конфігураційний файл для доступу до API БД. Ініціалізація БД зображена на рисунку 3.12.

```
const app = Firebase.initializeApp(firebaseConfig);
export const db = app.database();
```

Рисунок 3.12 – Ініціалізація БД

Зміна app містить результат виклику функції initializeApp в об’єкті Firebase, яка на вхід приймає конфігураційний файл, який згенерувала Firebase. Зміна db

отримує результат виклику функції database яка викликається із створеної раніше змінної app. Зміна app містить всю інформацію про те, звідки змінній db необхідно брати дані в залежності від налаштувань, які вказані в конфігураційному файлі. Виклик функції database повертає безпосередньо БД. Для використання змінної db в інших частинах проекту необхідно експортувати її. Розглянемо приклад завантаження даних з БД в розділі з налаштуваннями застосунку (рис. 3.13). Метод useEffect викликається при завантажені компоненту або при оновлені даних, які надходять до компоненту. В методі useEffect в змінній db викликається метод ref, який вказує з якого розділу в БД необхідно завантажити дані. В свою чергу в методу ref викликається метод on. Метод on приймає функцію, яка в якості аргументу приймає результат запиту до бази даних, тобто JSON-файл.

```
useEffect(()=>{
    db.ref('/state').on('value', querySnapShot => {
        let data = querySnapShot.val() ? querySnapShot.val() : {};
        let Items = {...data};
        setState(Items.gradeSystem)
        setIsLoad(false)
    });
}, [])
```

Рисунок 3.13 – Завантаження даних з БД

При кожній зміні параметрів в розділі налаштувань, буде викликатись метод ChangeDB (рис. 3.14).

```
const ChangeDB = () =>{
    db.ref('/state').set(state);
}
```

Рисунок 3.14 – метод ChangeDB

В методі ChangeDB, йде запит до БД зі описаним раніше методом ref. Метод set перезаписує попередній JSON-документ в новий зі зміненими даними. Загальна структура БД зображена на рисунку 3.15.

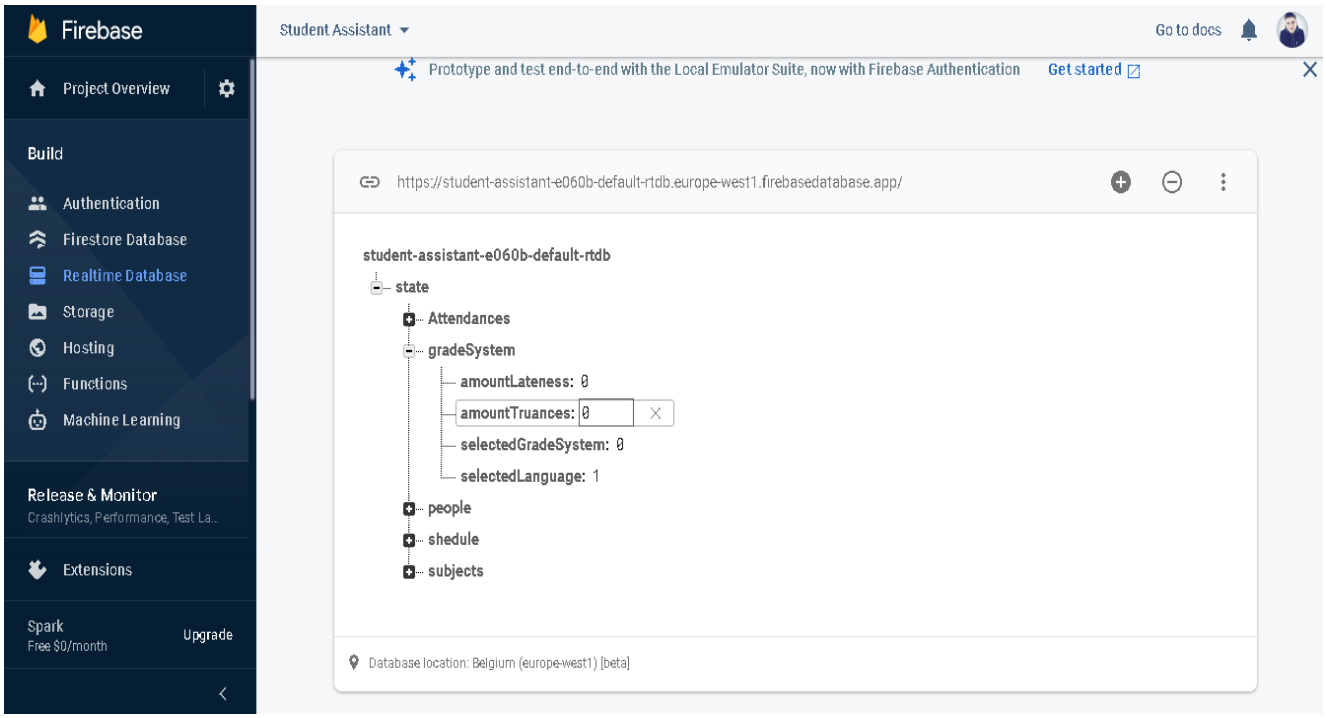


Рисунок 3.15 – Структура БД застосунку

3.6 Тестування додатку

Тестування є невід'ємною частиною розробки програмного забезпечення, оскільки поведінка програми в процесі розробки ПЗ може відрізнятись від кінцевого результату. Тому розробку будь-якого ПЗ слід супроводжувати тестуванням функціоналу додатку та проводити аналіз його поведінки. Мета тестування – пошук збоїв в роботі програм та алгоритмів з подальшим їх усуненням [12]. Звісно, тестування не гарантує те, що всі помилки в роботі ПЗ будуть знайдені, але їх мінімізація значно покращує якість ПЗ. Одною із фундаментальних сутностей в тестуванні виступають тест-кейси.

Тест-кейс – сутність, яка описує послідовне виконання певних дій або умов, для перевірки поведінки певного функціоналу ПЗ [1]. Тест-кейси поділяються на позитивні та негативні. Позитивні тест-кейси – це тест-кейси, в яких фактичний результат виконання дорівнює очікуваному. Негативні тест-кейси – це тест-кейси, в яких фактичний результат не дорівнює очікуваному, тобто знайдена помилка.

Далі в таблицях 3.1, 3.2, 3.3 представлено позитивні тест-кейси для окремого функціоналу застосунку.

Таблиця 3.1 – Тест-кейси для тестування функціоналу розкладу

№	Модуль	Підмодуль	Опис	Очікуваний результат
1	Розклад	Додавання заняття	- вибрати день неділі; - перейти на екран додавання заняття; - вибрати предмет; - вказати викладача; - вказати номер аудиторії; - вказати час початку заняття; - вказати час кінця заняття; - натиснути кнопку «Додати»;	- створюється нове заняття ; - відкривається розклад;
2	Розклад	Редагування заняття	- натиснути на предмет, - вказати нові данні або залишити поточні натиснути на кнопку «Перезаписати»;	- заняття перезаписується; - відкривається розклад зі зміненим предметом;
3	Розклад	Видалення заняття	- натиснути на заняття, яке необхідно видалити - натиснути кнопку «Видалити»;	- заняття видаляється; - відкривається розклад;

Таблиця 3.2 – Тест-кейси для тестування функціоналу розділу «ДЗ»

Модуль	Підмодуль	Опис	Очікуваний результат
Домашні завдання	Додавання домашнього завдання	- перейти в розділ «ДЗ»; - натиснути на кнопку додавання предмету; - ввести назву ДЗ; - ввести опис ДЗ; - вибрати предмет ; - вибрати складність ДЗ; - натиснути кнопку «Додати»;	- ДЗ додається до списку; - модальне вікно додавання ДЗ закривається;
Домашні завдання	Фільтрування домашніх завдань	- натиснути на кнопку фільтру домашніх завдань; - вибрати умову фільтрування;	- завдання фільтруються в залежності від умови;
Домашні завдання	Статус виконання ДЗ	- створити домашнє завдання ; - натиснути на кнопку розгортання інформації про ДЗ; - натиснути на кнопку «Виконати завдання»;	- шрифт назви ДЗ стає перекресленим; - завдання помічається як виконане ;

Таблиця 3.3 – Тест-кейси для тестування функціоналу розділу «ДЗ»

Модуль	Підмодуль	Опис	Очікуваний результат
Налаштування	Система оцінок	- вибрати одну із систем оцінок;	- система оцінок застосовується в додатку;
Налаштування	Зміна мови інтерфейсу	- вибрати одну із мов інтерфейсу;	- мова інтерфейсу змінюється;
Налаштування	Відвідуваність	- ввести максимальну кількість прогулів або запізнень; - натиснути кнопку «Застосувати»;	- максимальна кількість прогулів або запізнень змінюється;

3.7 Користування застосунком

Після запуску застосунку відкриється розділ з розкладом (рис. 3.16).

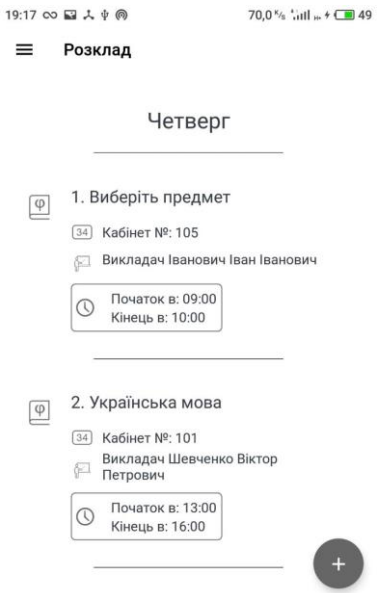


Рисунок 3.16 – Розділ «Розклад»

Для того, щоб створити заняття, треба перейти на день в який необхідно створити заняття, та натиснути на кнопку яка знаходиться справа знизу . Після цього відриється екран де користувачу необхідно ввести дані про предмет та натиснути кнопку «Додати». Для видалення або редагування заняття необхідно натиснути на заняття в розкладі. Для видалення необхідно натиснути кнопку «Видалити». Для редагування слід ввести нові дані для заняття на натиснути кнопку «Редагувати».

Для того щоб перейти в інші розділи застосунку, необхідно визвати бічне меню, провівши пальцем від правого краю екрану вліво. Бічне меню зображено на рисунку 3.17.

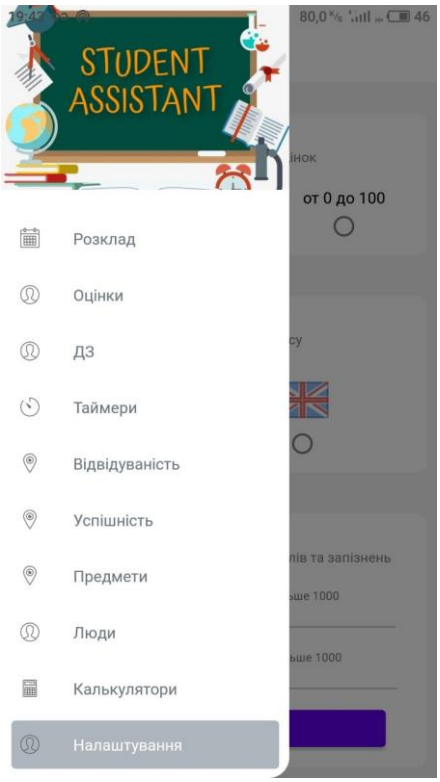


Рисунок 3.17 – Бічне меню застосунку

У розділі «Таймери» користувач може знайти таймер до початку заняття, якщо заняття ще не почалось, або час до кінця заняття, якщо воно вже почалось. Також в розділі представлений інший таймер який вказує на час до кінця заняття. Користувач може змінити дату до кінця навчального року, натиснувши на кнопку «Змінити дату». Розділ «Таймери» зображений на рисунку 3.18.

Рисунок 3.18 – Розділ «Таймери»

У розділі «Оцінки» користувач може виставляти оцінки (рис. 3.19). Для того, щоб виставити оцінку необхідно натиснути на кнопку, яка знаходиться справа зверху. Після цього з’явиться модальне вікно у якому необхідно вказати дані про оцінку та натиснути кнопку «Додати». В розділі представлений календар, в якому користувач може переглянути оцінки за конкретний день неділі, натиснувши на нього на календарі. Для видалення оцінки необхідно вибрати оцінку, яку необхідно видалити натиснути кнопку «Видалити».

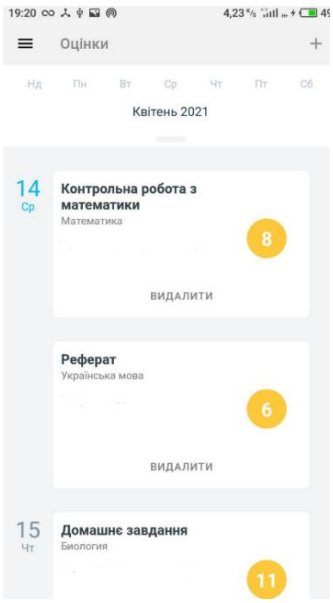


Рисунок 3.19 –Розділ «Оцінки»

Для того, щоб додати домашнє завдання, необхідно перейти в розділ «ДЗ» (рис 3.20), натиснути на кнопку зі знаком плюсу та в модальному вікні ввести наступні дані: назву домашнього завдання, опис домашнього завдання, предмет, складність, та термін здачі. Після цього необхідно натиснути кнопку «Додати». Для того, щоб видалити завдання необхідно натиснути на кнопку із символом корзини. При натисканні на завдання розгортається блок з докладним описом завдання. Для того, що відмітити домашнє завдання, як виконане, необхідно розкрити меню з докладним описом завдання та натиснути на кнопку «виконати завдання». Завдання буде візуально помічено, як виконане.

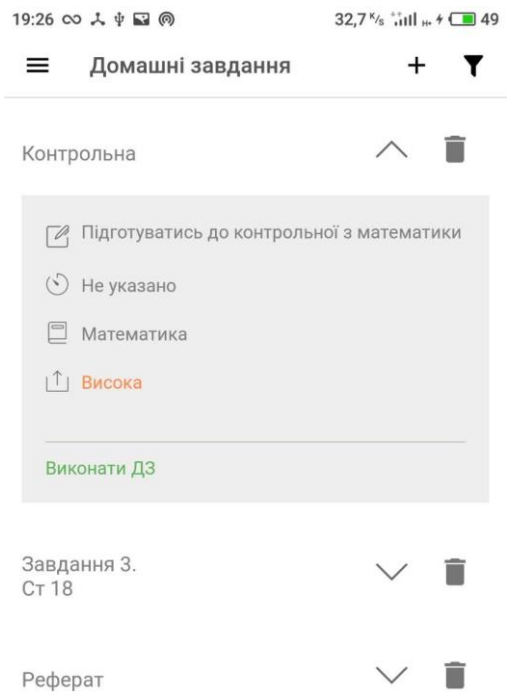


Рисунок 3.20 – Розділ «ДЗ»

Щоб відфільтрувати список домашніх завдання, необхідно натиснути на кнопку поряд з кнопкою додавання завдання. Після цього відкриється меню з вибором методу фільтрування. Представлені наступні методи фільтрування: виконані домашні завдання, спочатку легкі, спочатку складні. При виборі першого

методу, будуть виведені тільки ті завдання, які помічені, як виконані. При виборі другого та третього методу будуть виведені тільки легкі домашні завдання або тільки складні.

Застосунок надає можливість слідкувати за власною відвідуваністю. Користувач може додавати прогули та запізнення. Додати прогул або запізнення можна в розділі «Відвідуваність», натиснувши на плаваючу кнопку в нижньому правому куті екрану, та вибравши «Додати прогул» або «Додати запізнення». Після цього відкриється модальне вікно, в якому необхідно вибрати дату прогулу або запізнення та ввести причину. Після введення даних треба натиснути кнопку «Додати». Розділ відвідуваність зображений на рисунку 3.21.

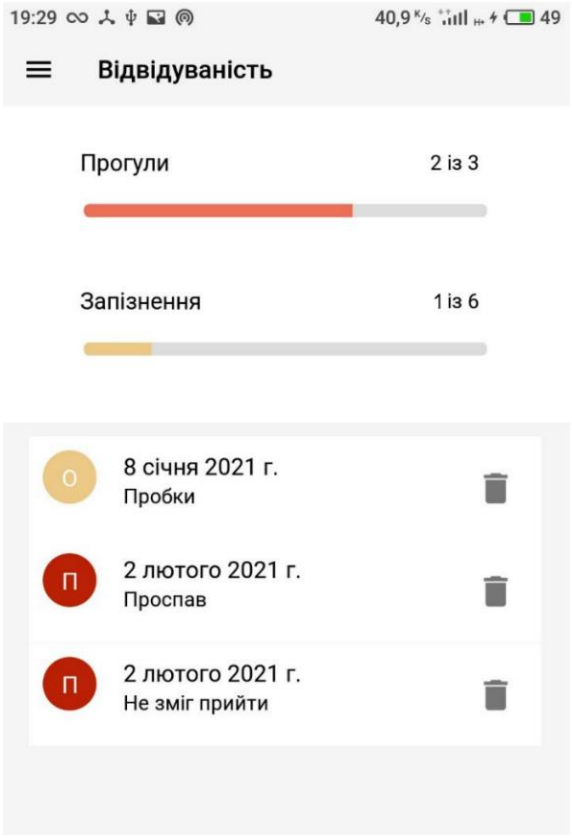


Рисунок 3.21 – Розділ «Відвідуваність»

Аналіз успішності та надання рекомендацій відбувається в розділі «Успішність». В розділі представлено три підрозділу.

В розділі «Оцінки» представлений список оцінок, які можливо виправити та які необхідно виправити. А також кругова діаграма, яка відображає співвідношення

оцінок. Для відображення цих даних необхідно додати хоча б одну оцінку в розділі «Оцінки».

Наступний підрозділ – «Предмети». В даному розділі представлено пелюсткову діаграму та два списки із трьох предметів, в яких середнє арифметичне найбільше та найменше відповідно. Даний розділ також вимагає наявність не менше трьох оцінок з різних предметів. Чим більше буде оцінок з різних предметів, тим більше буде точність аналізу. Для відображення даних на діаграмі треба вибрати не менше трьох предметів, з яких є оцінки. Для вибору предметів необхідно в полі «Виберіть предмети для відображення» розкрити меню з предметами та вибрати предмети, дані яких необхідно відобразити.

Останній розділ – «Склад розуму». Даний розділ на основі аналізу попередніх розділів показує інформацію, яким складом розуму володіє користувач та найбільш підходящі професії для користувача в залежності від предметів, в яких користувач має успіхи. Тому для відображення результатів аналізу даного підрозділу необхідно мати виставленні оцінки з предметів. На рисунку 3.22 зліва на право зображено: підрозділ «Оцінки», підрозділ «Предмети», підрозділ «Склад розуму».

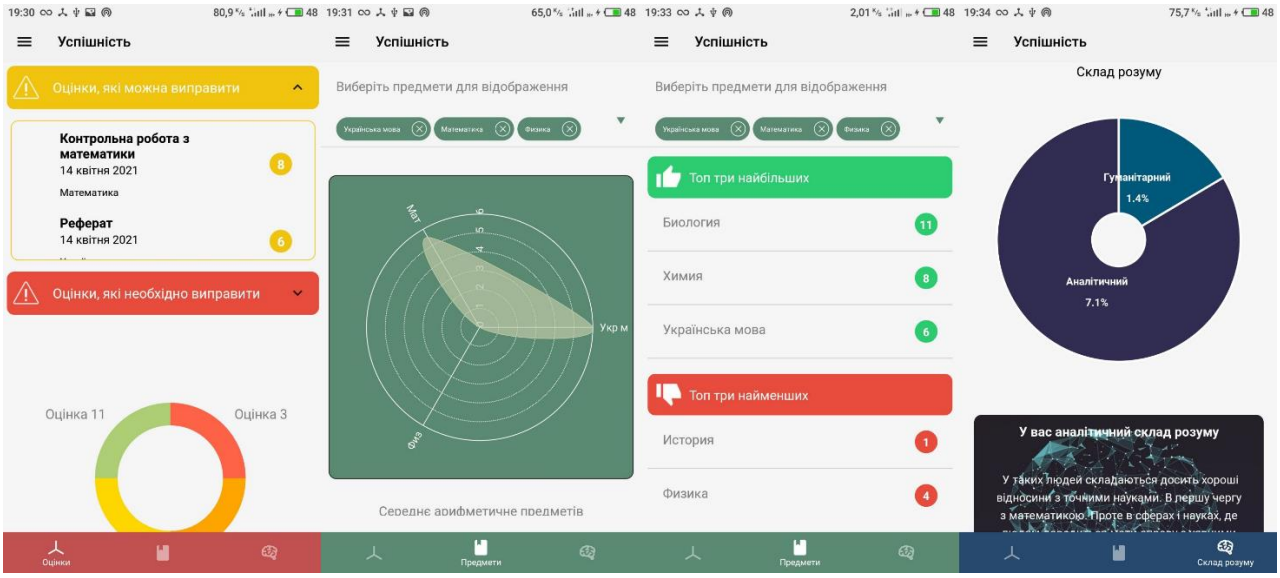


Рисунок 3.22 – Підрозділи розділу «Успішність»

Щоб додати предмети для подальшого використання їх в аналізі успішності та в інших розділах застосунку, необхідно перейти в розділ «Предмети» та

натиснути на кнопку у верхньому правому кутку екрану. Після цього відкриється модальне вікно, в якому необхідно ввести назву предмету, скорочену назву предмету для відображення в пелюстковій діаграмі в розділі «Успішність», та вибрати тип предмету(гуманітарний або точний). Після всіх вище описаних дій натиснути кнопку «Додати». В розділі присутні два списки, які можна розкрити. Перший список містить предмети, які є у застосунку за замовчуванням. Другий містить предмети, які додає користувач. Розділ предмети зображений на рисунку 3.23.

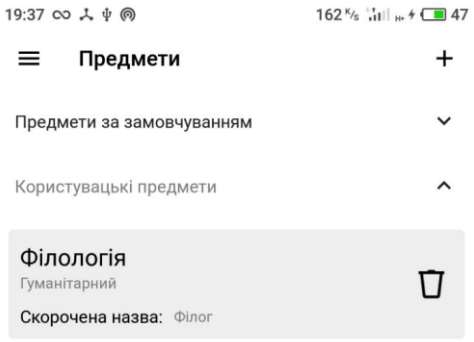


Рисунок 3.23 – Розділ «Предмети»

Якщо користувач хоче додати інформацію про одногрупників, однокласників, викладачів, то слід використовувати розділ «Люди» (рис. 3.24). В розділі представлено три списки: «Викладачі», «Однокласники/Одногрупники», «Інші». Крім того, користувач може додати інформацію про класного керівника. Для того щоб додати інформацію про людину, необхідно натиснути на кнопку в

правому верхньому кутку екрану. Відкриється модальне вікно, в якому користувач повинен ввести наступні дані: вибрати, кого необхідно додати (викладач, однокласник або одногрупник, класний керівник або інші), ввести ФІО людини, назву предмета(якщо людина – викладач або класний керівник), номер телефону, день народження. Після введення даних, натиснути «Додати». Видалити інформацію про людину можна, натиснувши на кнопку «Видалити», яка є в кожному блоці з інформацією про людину. Також, якщо список людей достатньо великий, можна скористатись пошуком, вводячи прізвище або ім’я користувача в поле пошуку, яке розміщене зверху екрану. Керівника можна змінити, натиснувши на кнопку «Змінити керівника».

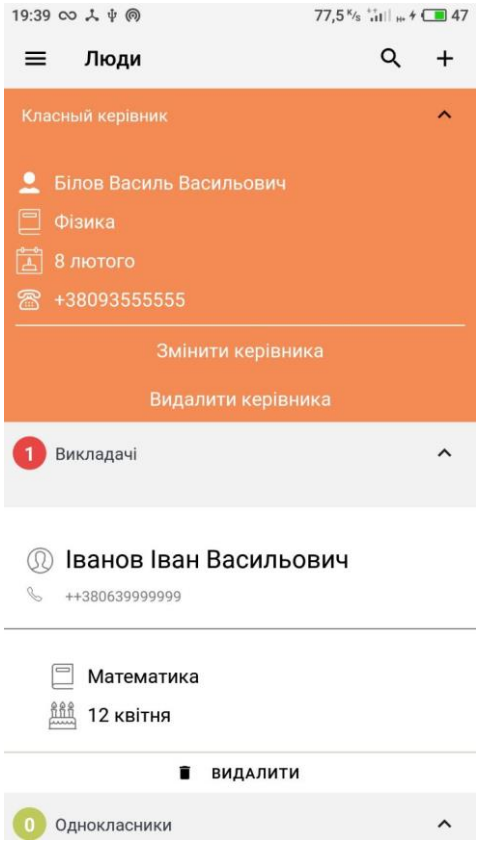


Рисунок 3.24 – Розділ «Люди»

Розділ «Калькулятори» вміщує 17 калькуляторів з таких предметів: алгебра, геометрія, фізика. Для того щоб почати роботу з калькуляторами, треба перейти в сам розділ, та вибрати один із підрозділів. Вибравши необхідний калькулятор, слід вказати дані, які необхідні для обрахунку та натиснути кнопку «Обрахувати».

Обрахунок виведеться в блоці з калькулятором. На рисунку 3.25 зображено один із калькуляторів підрозділу «Фізика».

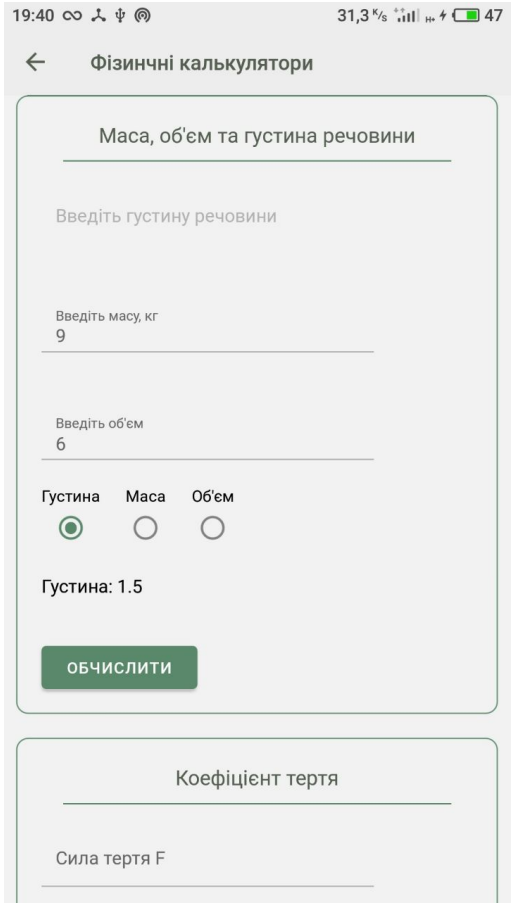


Рисунок 3.25 – Один із калькуляторів у підрозділі «Фізика»

ВИСНОВКИ

Робота проведена з метою створення мобільного додатку для планування навчання. Головна мета застосунку – надати користувачеві цифровий інструмент для зберігання навчальної інформації та її аналізу з метою покращення успішності користувача в навчанні.

Наведено результати дослідження, які підтверджують актуальність роботи та її наукову новизну. Результати дослідження полягають в наступному:

- проаналізовано можливість перенесення необхідної для користувача навчальної інформації в цифровий формат;
- проведено аналіз недоліків та переваг існуючих аналогів;
- визначено основні вимоги до застосунку;
- досліджено можливість використання мови JavaScript для розробки мобільних додатків;
- розроблено мобільний додаток для покращення ефективності в навчальному процесі.

Встановлено, що мова програмування JavaScript має великий потенціал до розробки мобільного програмного забезпечення. Було проаналізовано можливості середовищ для розробки ПЗ та вибрано програму Visual Studio Code, як основне середовище розробки.

Проведено проектування системи за допомогою UML діаграм. В результаті проектування було розроблено діаграми прецедентів та діяльностей.

Максимальну користь від додатку слід очікувати в тому випадку, якщо користувач сам бажає покращити успішність в навчанні або зберігати навчальну інформацію в даному додатку. В застосунку присутній функціонал, який буде корисний, як для учня школи так і для студента.

ПЕРЕЛІК ПОСИЛАНЬ

1. Paul C. Jorgensen. Software Testing: A Craftsman’s Approach, Fourth Edition 4th Edition — Boca Raton: Auerbach Publications, 2013. – 86 с.
2. Friedmann. J., Masiello.E., Mastering React Native Kindle Edition. — Birmingham: Packt Publishing, 2017.— 190 с.
3. Haverbeke. M. Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming. — San Francisco: No starch Press, 2018.— 23 с.
4. Фрімен. Е., Робсон. Е. Патерни проектування — Харків: Фабула, 2020.— 85 с.
5. Eisenman.B. Learning React Native, 2nd edition — Sebastopol: O'Reilly Media, 2017.— 1 с.
6. Постіл С. Д. UML. Уніфікована мова моделювання інформаційних систем : навч. посіб. / С. Д. Постіл. - Ірпінь : Університет державної фіскальної служби України, 2019. — 18с.
7. Ashok Kumar S. Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase, — Birmingham: Packt Publishing, 2018. — 8 с.
8. Akshat.P., Abhishek.W. React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications, — New-York: Apress, 2019. — 3 с.
9. Antani.V. Mastering JavaScript: Birmingham: Packt Publishing, 2016. — 181 с.
10. Scott.A. JavaScript Everywhere: Building Cross-Platform Applications with GraphQL, React, React Native, and Electron — Sebastopol: O'Reilly Media, 2020. — 235 с.
11. Del Sole. A. Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux, – New-York: Apress, 2018 – 40 с.
12. Вступ до інженерії програмного забезпечення: навч. посібник / Є.В. Левус, Н.Б. Мельник – Львів: Видавництво Львівської політехніки, 2017. — 91 с.

13. Як інтегрувати Redux у свій додаток з React Native та Expo [електронний ресурс]: [Веб-сайт]. — Режим доступу: https://hackit-ukraine.com/2786-how-to-integrate-redux-into-your-application-with-react-native-and-expo (дата звернення 21.11.2020) – Назва з екрана.

14. Getting Started [електронний ресурс]: [Веб-сайт]. — Режим доступа: https://code.visualstudio.com/docs (дата звернення 18.10.2020) – Назва з екрана.

15. Не просто накладіть свій код – виправте його за допомогою Prettier [електронний ресурс]: [Веб-сайт]. — Режим доступу: https://hackit-ukraine.com/1631-dont-just-lint-your-code-fix-it-with-prettier (дата звернення 07.01.2021) – Назва з екрана.

16. Firebase helps you build and run successful apps [електронний ресурс]: [Веб-сайт]. — Режим доступу: https://firebase.google.com/ (дата звернення 18.03.2021) – Назва з екрана.

17. Hello React Navigation [електронний ресурс]: [Веб-сайт]. — Режим доступу: https://reactnavigation.org/docs/hello-react-navigation (дата звернення 03.11.2020) – Назва з екрана.

18. Mobile Operating System Market Share Worldwide [електронний ресурс]: [Веб-сайт]. — Режим доступу: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення 18.10.2020) – Назва з екрана.

19. Introduction to Expo [електронний ресурс]: [Веб-сайт]. —Режим доступу: https://docs.expo.io/(дата звернення 21.10.2020) – Назва з екрана.

20. Top 10 Types of Charts and Graphs for Data Visualization [електронний ресурс]: [Веб-сайт]. —Режим доступу: https://www.edrawsoft.com/chart-types-uses.html (дата звернення 03.02.2021) – Назва з екрана.

Додаток

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

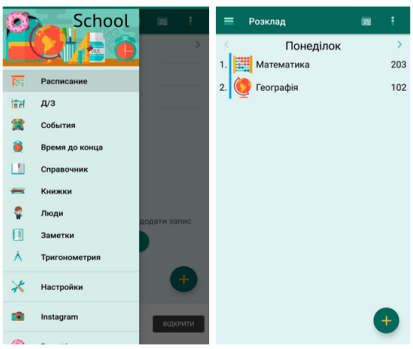
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ STUDENT ASSISTANT ДЛЯ ПЛАНУВАННЯ НАВЧАННЯ НА МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

Виконав студент 4 курсу
групи ПД-42
Дехтяренко Олександр Русланович
Керівник роботи
к.т.н., доцент Негоденко Олена Василівна

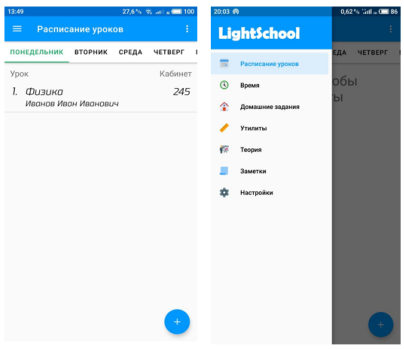
МЕТА,ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Мета роботи - розробка програмного забезпечення для студентів закладів вищої освіти та учнів шкіл на мові програмування JavaScript
- Об'єкт дослідження - покращення ефективності в навчальному процесі.
- Предмет дослідження - додаток для планування навчання.

ІСНУЮЧІ АНАЛОГИ

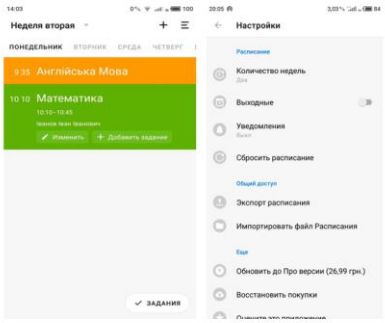


Додаток School

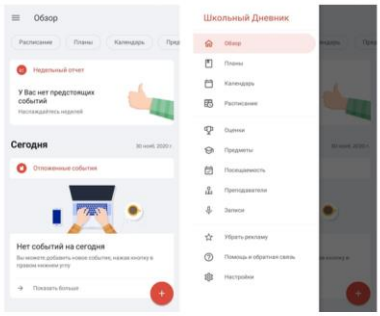


Додаток LightSchool

ІСНУЮЧІ АНАЛОГИ



Додаток Class TimeTable



Додаток School Planer

ТАБЛИЦЯ ПЕРЕВАГ ТА НЕДОЛІКІВ

Назва застосунку	Переваги	Недоліки
School	• таймер до початку та кінця уроку; • велика кількість теорії з різних предметів; • переклад інтерфейсу на понад 20 мов світу;	• часті вильоти додатку; • велика кількість реклами; • можлива втрата даних; • відсутність системи оцінок;
Class Timetable	• відсутність реклами; • вибір кольору предмету в розкладі;	• невеликий функціонал; • мова інтерфейсу тільки російська;
School Planer	• можливість моніторингу відвідуваності • система оцінок;	• відсутність інших мов крім російської; • перенасиченість інтерфейсу компонентами, які виконують одну і ту ж саму функцію;
Light School	• повна відсутність реклами; • можливість встановити стартовий екран власноруч;	• відсутність можливості перегляду початку та кінця заняття на екрані розкладу; • відсутність системи оцінок;

ФУНКЦІОНАЛ

Базовий

- Розклад
- Домашні завдання
- Відвідуваність
- Люди
- Оцінки
- Таймери

Покращено

- Домашні завдання
- Люди
- Таймери

Нове

- Успішність
- Склад розуму
- Підходящі професії
- Калькулятори

Вилучено

- Теорія
- Записи

ОПЕРАЦІЙНІ СИСТЕМИ

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

JavaScript – це динамічно типізована мова програмування, яка є імплементацією стандарту ECMAScript.

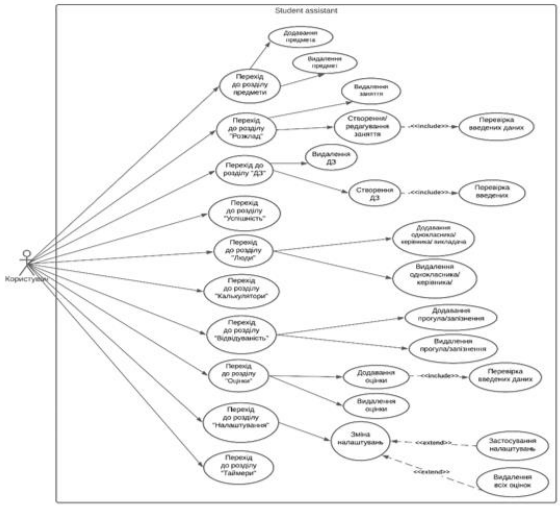
React Native – це JavaScript-бібліотека для розробки кроссплатформенних мобільних додатків, яка побудована на базі бібліотеки React.

Redux – це бібліотека, яка дозволяє зручно керувати даними з будь-якої точки застосунку побудованого на React або React Native. Redux використовує шаблон проектування «Спостерігач», який базується на спостережені об'єктом змін інших об'єктів батьківського класу.

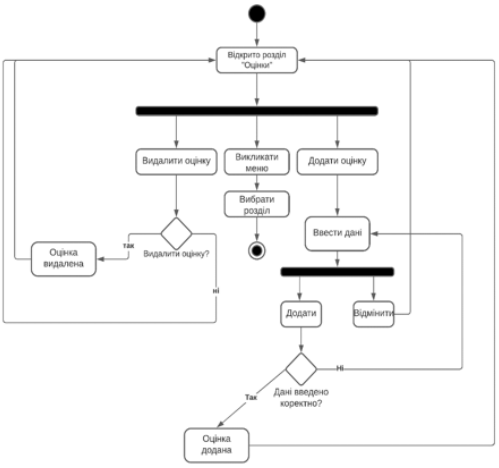
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Firebase – це хмарна, NoSQL база даних, яка зберігає дані в текстовому форматі JSON.

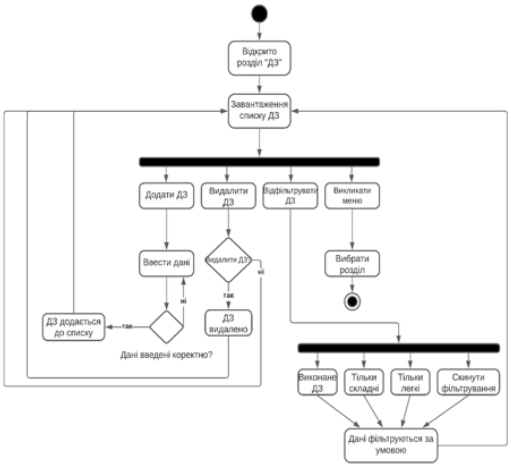
Visual Studio Code – безкоштовний редактор коду розроблений компанією Microsoft. Visual Studio Code має достатньо великий вбудований набір можливостей для розробки, без потреби встановлення додаткових плагінів. Редактор підтримує велику кількість мов програмування, такі як: C, C++, Java, Ruby, C# та інші



Діаграма прецедентів



Діаграма діяльності розділу «Оцінки»



Діаграма діяльності розділу «ДЗ»

Схема бази даних застосунку

ВИСНОВКИ

Робота проведена з метою створення мобільного додатку для планування навчання.

- наведено результати дослідження, які підтверджують актуальність роботи та її наукову новизну.
- проаналізовано можливість перенесення необхідної для користувача навчальної інформації в цифровий формат;
- проведено аналіз недоліків та переваг існуючих аналогів;
- визначено основні вимоги до застосунку;
- досліджено можливість використання мови JavaScript для розробки мобільних додатків;
- встановлено, що мова програмування JavaScript має великий потенціал до розробки мобільного програмного забезпечення.
- проведено проектування системи за допомогою UML діаграм. В результаті проектування було розроблено діаграми прецедентів та діяльностей.
- розроблено мобільний додаток для покращення ефективності в навчальному процесі.

Апробація результатів дослідження

Дехтяренко О.Р. Мобільний додаток для планування навчання на мові програмування JavaScript // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали всеукраїнської науково-технічної конференції. Збірник тез. – К.: ДУТ, 2021. — С. 36 – 37.

Дякую за увагу!