Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня вищої освіти магістра

зі спеціальністю 121 – Інженерія програмного забезпечення

На тему: Розробка програмного забезпечення комплексу віртуальних лабораторій з фізики: дослідження впливу інтерактивних симуляцій на якість навчання

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.

Студент гр. ІПЗ-24 м
_____ /К. О. Павленко /

Керівник кваліфікаційної роботи _____ / А. М. Стрюк /

Економіко-організаційна частина _____ / _____ /

Нормоконтроль _____ / _____ /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«___» _____20_____р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-24 м Павленку Кирилу Олеговичу

1. Тема: Розробка програмного забезпечення комплексу віртуальних лабораторій з фізики: дослідження впливу інтерактивних симуляцій на якість навчання
   затверджено наказом по КНУ № __від «_» _____ 20____ р.
2. Термін подання студентом закінченої роботи: «__» _____ 20___ р.
3. Вихідні дані по роботі: розроблювана система повинна працювати з інтерактивними анімаціями, мати 2 профілі користувачів, мати можливість виконувати та перевіряти завдання з фізики.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз існуючих методів розв’язання задачі, вибрати метод відтворення анімації на мобільних пристроях, здійснити програмну реалізацію розробленої системи, провести тестування розробленої системи.
5. Перелік ілюстративного матеріалу: блок-схеми розроблених алгоритмів, схема взаємодії модулів системи, знімки екранних форм.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури за тематикою роботи	01.10.2024 – 15.10.2024
2	Проведення аналізу існуючих теоретичних методів дослідження і вирішення проблеми	16.10.2024 – 31.10.2024
3	Проведення критичного порівняльного аналізу існуючих аналогів програмних систем	01.11.2024– 30.11.2024
4	Формулювання актуальності і завдань роботи	01.12.2024 – 15.12.2024
5	Оформлення матеріалів першого розділу роботи	16.12.2024– 31.12.2024
6	Розробка структурних і функціональних моделей	01.01.2025 – 15.02.2025
7	Розробка структур даних та алгоритмів програмного комплексу	16.02.2025– 15.03.2025
8	Оформлення матеріалів другого розділу роботи	16.03.2025 – 31.03.2025
9	Розробка бази даних, програмних модулів, бібліотек та інтерфейсу програмного комплексу	01.04.2025 – 15.06.2025
10	Оформлення матеріалів третього розділу роботи	16.06.2025 – 30.06.2025
11	Дослідження та узагальнення результатів роботи з точки зору наукової та технічної цінності	01.07.2025 – 31.08.2025
12	Оформлення матеріалів четвертого розділу роботи	01.09.2025 – 15.09.2025
13	Аналіз економічної ефективності інновації	16.09.2025– 31.10.2025
14	Остаточне оформлення пояснювальної записки	11.11.2025 – 20.11.2025

Дата видачі завдання: «__» _____ 20____ р.
Студент _____/ К. О. Павленко /
Керівник роботи _____/ А. М. Стрюк /

РЕФЕРАТ

ІНТЕРАКТИВНА АНІМАЦІЯ, ЛАБОРАТОРНІ РОБОТИ З ФІЗИКИ, ШКІЛЬНА ПРОГРАМА, МОБІЛЬНІ ПРИСТРОЇ.

Пояснювальна записка: 73 с., 18 Рисунок , 2 дод., 10 джерел.

Метою кваліфікаційної роботи є підвищення ефективності виконання лабораторних робіт з фізики учнями 10–11 класів в умовах дистанційного навчання, покращення їхніх знань та практичних навичок у вивченні фізичних явищ за рахунок інтерактивних анімацій.

У процесі розробки сформовано дослідницьку методику, що включає моделювання фізичних процесів, обробку та інтерпретацію експериментальних даних. Для лабораторних робіт була створена інтерактивна анімаційна модель, яка відтворює перебіг фізичного експерименту, а також покрокову інструкцію щодо його виконання.

Реалізовано систему надсилання та перевірки робіт на 2 профілі користувачів. Додано механізм обліку виконаних завдань.

ABSTRACT

ANIMATION, PHYSICS LABORATORY WORK, SCHOOL CURRICULUM, MOBILE DEVICES.

Thesis in: 63 p., 18 fig., 2 app., 9 sources.

The aim of the qualification work is to improve the efficiency of laboratory work in physics by students in grades 10–11 in the context of distance learning, to improve their knowledge and practical skills in studying physical phenomena through interactive animations.

In the course of development, a research methodology was formed, which includes modelling physical processes, processing and interpreting experimental data. An interactive animated model was created for laboratory work, which reproduces the course of a physical experiment, as well as step-by-step instructions for its implementation

A system for submitting and checking work for two user profiles has been implemented. A mechanism for recording completed tasks has been added.

Зміст

1 ТЕОРЕТИЧНИЙ РОЗДІЛ

1.1 Актуальність роботи

Розробка мобільного застосунку зумовлена потребою в ефективному та доступному дистанційному навчанні. Пандемія коронавірусу, повномасштабна війна в Україні змінили не тільки повсякденне життя громадян, а й підходи до освіти. Система навчання змушена адаптуватися до нових умов, у яких традиційні та звичні форми викладання виявилися неефективними.

Якісна освіта повинна бути доступною незалежно від зовнішніх обставин – чи то карантин, переїзд, перебої з електропостачанням, обмежений доступ до навчальних закладів.

Традиційні форми навчання не мають інструментів для повноцінного відтворення експериментальної частини вдома, особливо в таких дисциплінах, як фізика. Експерименти є невід'ємною частиною цієї науки, тому лабораторні роботи мають важливе значення для формування повноцінного розуміння дисципліни і розвитку учнів. У реальних умовах важко організувати безпечні, якісні та доступні лабораторні заняття – через відсутність доступу до лабораторій, викликані воєнними діями.

Дистанційне навчання має багато переваг, але часто стикається з проблемами в організації саме практичної діяльності.

Розвиток сучасних цифрових технологій здатен вирішити ці виклики. Мобільні застосунки дають можливість створювати інтерактивне візуальне середовище для моделювання фізичних процесів, оцінювання віртуальних лабораторних робіт та закріплення теоретичних знань на практиці.

Завдяки тому, що застосунок розробляється на базі операційної системи Android, він є максимально доступним для великої кількості людей. Користувачі зможуть навчатися навіть за відсутності інтернету та електропостачання, що критично важливо в реаліях життя.

Таким чином, розробка застосунку відповідає вимогам часу та стає важливим кроком до забезпечення гнучкого, сучасного та якісного навчального процесу в умовах постійних змін.

1.2 Науковий апарат

Метою дослідження є підвищення ефективності виконання лабораторних робіт з фізики учнями 10-11 класів в умовах дистанційного навчання шляхом розробки і впровадження мобільного застосунку з інтерактивними анімаціями, що сприяють кращому засвоєнню матеріалу та роблять практичні заняття можливими в складних умовах.

Об’єктом дослідження є мобільні технології в освіті, які охоплюють сукупність програмних та апаратних засобів, що застосовуються для підтримки та організації навчального процесу з використанням пристроїв, таких як смартфони і планшети. Особлива увага приділяється тому, як ці технології можуть бути застосовані для викладання практичних занять з фізики у старшій школі.

Предметом дослідження є особливості використання мобільного застосунку для виконання лабораторних робіт з фізики учнями 10-11 класів. Фокус дослідження спрямований на те, як інструменти дистанційного навчання впливають на процес виконання лабораторних робіт.

Задачі дослідження

Для досягнення поставленої мети передбачено вирішення таких дослідницьких завдань:

– проаналізувати сучасний стан використання мобільних технологій у дистанційному навчанні, а саме викладанні фізики у старших класах;

– проаналізувати особливості навчальної програми та освітніх стандартів для 10-11 класів з фізики. Метою є досягнення відповідності між можливостями застосунку, що розробляється, та реальними, актуальними завданнями, які вирішують учні у процесі вивчення дисципліни;

– розробити структуру та функціональні компоненти мобільного застосунку, що забезпечать безпосередню можливість виконання робіт та широкий перелік підтримуваних пристроїв;

– використовуючи розроблену структуру та компоненти, реалізувати програмний прототип, який дасть змогу моделювати базові фізичні експерименти;

– здійснити експеримент, який спрямований на зіставлення результатів виконання завдань із застосуванням найближчого аналога та застосунком, що перебуває на етапі розробки;

– оцінити отримані результати та сформувати практичні рекомендації щодо удосконалення програмного забезпечення.

Методи дослідження

Для реалізації практичної частини дослідження будуть застосовуватися як кількісні, так і якісні методи.

Кількісні методи передбачають:

– тести – оцінка знань учнів до і після використання застосунку;

– опитування – анкети для отримання даних щодо покращення функціональності застосунку, зручності використання.

Якісні методи передбачають:

– інтерв'ю – для глибшого розуміння досвіду учнів;

– спостереження – аналіз поведінки учнів під час роботи з застосунком, їх зацікавленість та враження від використання цифрового інструменту.

Теоретична частина включає в себе аналіз наукової літератури, вивчення теоретичних основ використання мобільних технологій в освітньому процесі, а також вимог до проведення лабораторних робіт.

Описаний науковий апарат дозволить дослідити та виміряти ефективність використання нового цифрового інструменту в області фізики, а також надати практичні рекомендації щодо покращення якості освіти.

1.3 Дослідження особливості предметної галузі

Українська система освіти переживає далеко не найкращі часи. Через повномасштабне вторгнення значна частина шкільної інфраструктури пошкоджена або зруйнована, більшість навчальних закладів змушена перейти в дистанційний формат.

Внутрішні переміщення учнів, загроза обстрілів посилюють нерівність у доступі до навчальних матеріалів: учні, які знаходяться у місті мають більше можливостей: доступ до інтернету та комп'ютерів. Сільська школа, в свою чергу, відчуває нестачу технічних засобів та фахівців.

При переході на дистанційне навчання, вчителі з фізики стикаються зі значними проблемами. Більшість дослідів потребує спеціальної шкільної апаратури, яка є вдома далеко не у кожного здобувача освіти. Досвід показує, що деякі вчителі вдавалися до імпровізацій під час карантину COVID-19 та демонстрували дослідження дифракції світла за допомогою CD-диска, лінійки та лазерних указок [1]. Такий підхід можливий лише у випадку ретельної підготовки та віддалених консультацій вчителя з учнем. Таке навчання ускладнено контролем за процесом, без наочного спостереження за виконанням роботи важко перевірити самостійність виконання завдання і достовірність результатів. Зростає ризик нечесного виконання роботи.

Тому, зазвичай, дистанційний формат передбачає спрощення практичної роботи до презентацій і тестів, що негативно впливає на досвід учнів та засвоєння матеріалу.

Мобільні технології вирішують частину цих проблем та стають одним з ключових ресурсів для підтримки вивчення дисциплін. Головні переваги це безперервність і гнучкість.

У здобувачів освіти з'являється можливість працювати з контентом будь-де і будь-коли. Навіть у випадку тривалих відключень світла смартфон залишається в учня постійно. Сучасні застосунки розширюють можливості

для спостерігання і взаємодії зі складними фізичними процесами. Це збільшує залученість та сприяє кращому сприйняттю матеріалу.

Відкривається простір для розробки цікавих та інтерактивних інструкцій, які пояснюють специфіку роботи з віртуальними стендами в одному застосунку, що позитивно впливає на засвоєння матеріалу. У вже традиційному дистанційному навчанні, учитель зазвичай має декілька методичних вказівок щодо виконання лабораторної роботи і особливостями роботи з віртуальним середовищем.

Однак з’являються нові виклики, які потребують вирішення.

Сучасний школяр стикається з рядом психологічних труднощів. Діти, які пережили величезний стрес внаслідок переїзду сімей, окупації дому чи втрати рідних мають знижений рівень мотивації до навчання. Комплексний користувацький інтерфейс наявних віртуальних фізичних лабораторій може лише знизити бажання учня знову повертатися до повсякденного ритму життя.

Тому при розробці програмного забезпечення потрібно враховувати психоемоційний стан здобувачів освіти і завантаженість користувацького інтерфейсу. Потрібно дотримуватися балансу між простотою та інформативністю фізичних стендів, адаптовувати контент до специфіки навчального матеріалу, мобільних пристроїв і навчальних цілей.

1.4 Аналіз досліджень з обраної теми

Дослідження показують, що віртуальне навчання здебільшого не поступається традиційному, якщо звертати увагу на показники успішності учнів та засвоєння матеріалу.

Так, Антоніо та Кастро у своєму дослідженні “Effectiveness of Virtual Simulations in Improving Secondary Students’ Achievement in Physics: A Meta-Analysis” виявили значний позитивний ефект від використання симуляцій: зважений ефектний розмір Hedges`g=0.94 свідчить про зростання успішності

учнів. [2] У більшості досліджень застосовувалися симуляції PhET та Crocodile Physics.

Самі студенти також позитивно відносяться до віртуальних лабораторій. Опитування показало, що більше половини опитаних, а саме 83% заявила, що віртуальні експерименти допомагають сформувати глибші розуміння фізичних явищ, ніж традиційні. Учні відмітили, що завдяки симуляціям у них з'являється можливість багато разів повторювати дослід та отримувати додаткові дані (показники, графіки та ін.), чого не завжди можна досягти у традиційних заняттях.

Українські дослідження також дають схожі результати. Наприклад Ольга Федчишин у своїй роботі "Методичні основи використання Phet-симуляцій у процесі вивчення фізики" зазначає, що віртуальні фізичні експерименти формують дослідницькі компетентності та навички, також активізують пізнавальну діяльність учнів. [3]

Окрема увага була приділена мобільним технологіям у віртуальній реальності (VR) і доповненій (AR). Огляд Торреса Ансельмо та співавторів виділяють 6 ключових тем підвищення ефективності навчання за допомогою мобільних інструментів:

- глибше розуміння концепцій;
- збільшення зацікавленості;
- покращення успішності;
- розвиток вищих когнітивних умінь;
- розвиток практичних навичок;
- зниження когнітивного навантаження.

Відзначаються AR/VR системи, мобільні та навчальні застосунки. [4]

Дослідження з залученням AR - застосунку для вивчення Сонячної системи показало, що учні з AR мали вищий рівень досягнень у навчанні ніж ті, хто працював за підручниками. [5]

Однак впровадження AR/VR рішень мають технічні і методичні обмеження. Згадується необхідність сумісності пристроїв, нестабільність

програм, високої вартості обладнання у випадку використання VR, а також підготовки вчителів і інтеграції інструментів у навчальну програму.

Можна стверджувати, що віртуальне та інтерактивне навчання в цілому може позитивно впливати на результати роботи учнів. Уроки з застосуванням симуляцій не поступаються традиційним підходам.

1.5 Визначення вимог

Перш за все, застосунок повинен забезпечувати можливість виконання лабораторних робіт з фізики, які відповідають чинній шкільній програмі. Кожна робота має бути побудована з урахуванням змісту відповідного навчального розділу, наприклад електрики, оптики, термодинаміки, тощо. Учень повинен мати змогу ознайомитися з обладнанням експерименту, метою, провести віртуальні вимірювання, виконати обчислення та заповнити розрахунки.

Користувацький інтерфейс має бути доступним для користувача, без перевантаження елементами. Повинна бути реалізована проста навігація між модулями, можливість збереження результатів та зручний доступ до вибору лабораторної роботи. Застосунок має містити графічні анімації, що імітують роботу реального фізичного обладнання.

У зв’язку з тим, що велика частина користувачів може мати труднощі зі зв’язком, застосунок не повинен вимагати постійного підключення до Інтернету.

Методичний зміст має бути структурований та поділений на маленькі секції з поясненням виконання експерименту, вступ з формулюванням мети, інтерактивні підказки, формули для обчислення.

Кожна лабораторна робота повинна підтримувати функцію оцінювання результатів обчислень, підкреслювати невідповідності знайдені у процесі перевірки. Додатково для вчителів має бути передбачена можливість перегляду

результатів роботи учня та винесення остаточної оцінки. Важливо реалізувати облік користувачів та ведення журналу.

Особисті дані користувачів повинні надійно зберігатися та мати можливість резервного копіювання.

Отже, майбутній застосунок повинен мати наступні можливості:

– знайомити користувача з методичними рекомендаціями виконання лабораторних робіт;

– мати різні інтерактивні анімації для різних видів робіт;

– надавати можливість користувачу переглядати його виконані роботи;

– зберігати дані користувача;

– ділитися результатами роботи;

– виконувати резервне збереження даних користувачів;

– створювати документи для друку, а саме звіти з результатами обчислень.

Додатково програма повинна володіти наступними функціональними можливостями: перевіряти вхідні дані на коректність та відповідність умовам, мати зручний та зрозумілий інтерфейс, мати підказки на відповідних кроках розрахунків.

Виділені завдання:

1. Розробити систему для організації дистанційного навчального процесу.
2. Розробити блок-схеми алгоритмів для операцій:
    2.1. Реєстрації нового учня.
    2.2. Реєстрації нового вчителя.
    2.3. Виконання та поширення лабораторних робіт.
    2.4. Перевірки лабораторних робіт.
    2.5. Резервне копіювання лабораторних робіт.
3. Розробити інтерфейс системи.
4. Розробити інструкцію користувача.

1.6 Моделювання, проєктування та прототипування розробки

Перед тим як приступити до повноцінної розробки програмного забезпечення слід спроектувати діаграми використання застосунку.

Представлено на рисунку 1.1 та 1.2

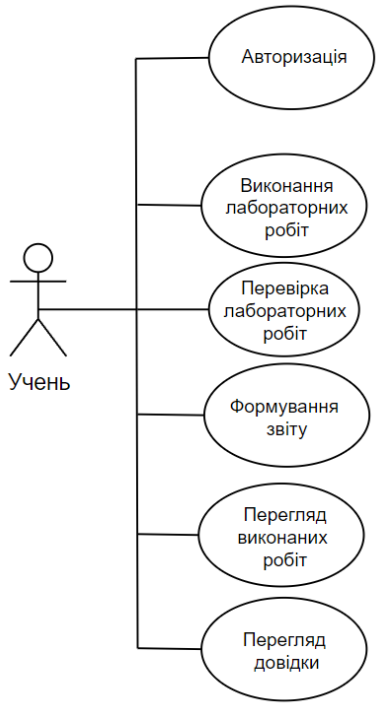

Рисунок 1.1 – Діаграма варіантів використання учня

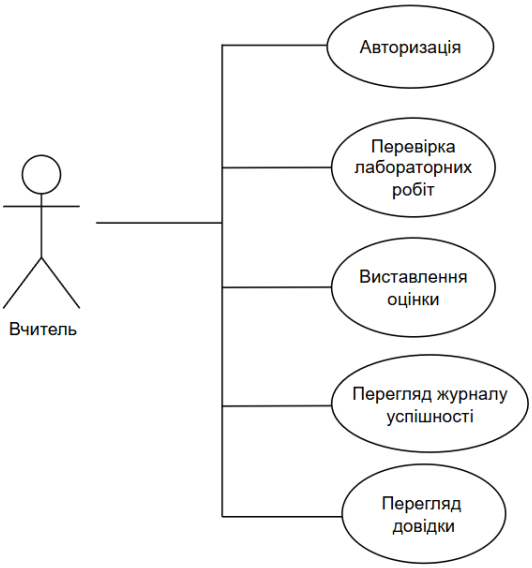

Рисунок 1.2 – Діаграма використання вчителя

Основними користувачами системи будуть вчитель та учень. Для них передбачена можливість авторизації у свої особисті кабінети в яких буде

доступний різний перелік функцій. Для учня важливим є виконання лабораторних робіт, для вчителя – їх перевірка.

Для подальшої розробки передбачається створення структурної та функціональної схеми програмного комплексу.

Представлено на рисунку 1.3 та 1.4

Рисунок 1.3 – Структурна схема програмного комплексу

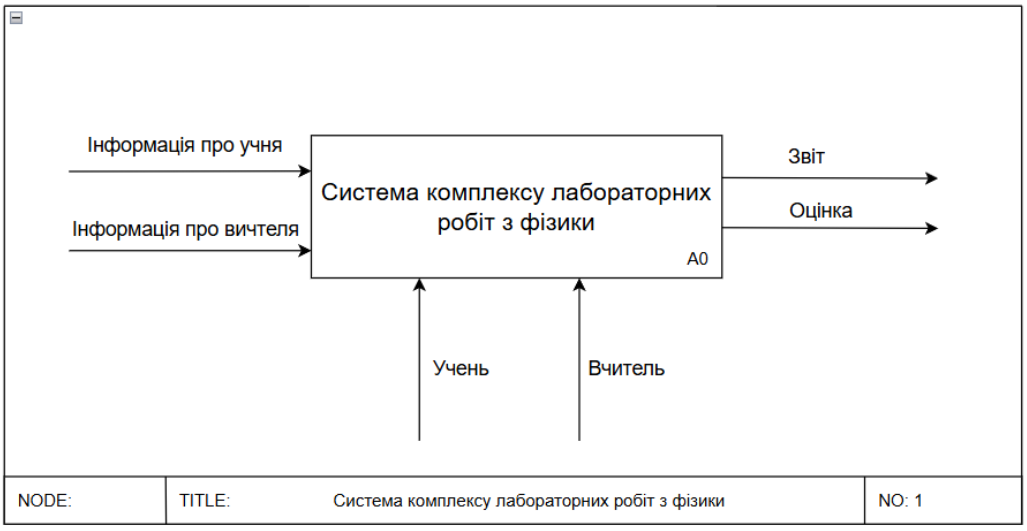

Рисунок 1.4 – Контекстна функціональна схема

Спочатку вводяться дані користувачів: інформація про учня та вчителя. Вихідні дані являють собою звіт, створений учнем та оцінку роботи після перевірки вчителем.

На рисунку 1.5 зображена декомпозиція контекстної функціональної схеми програмного комплексу, яка описує роботу програми більш детально.

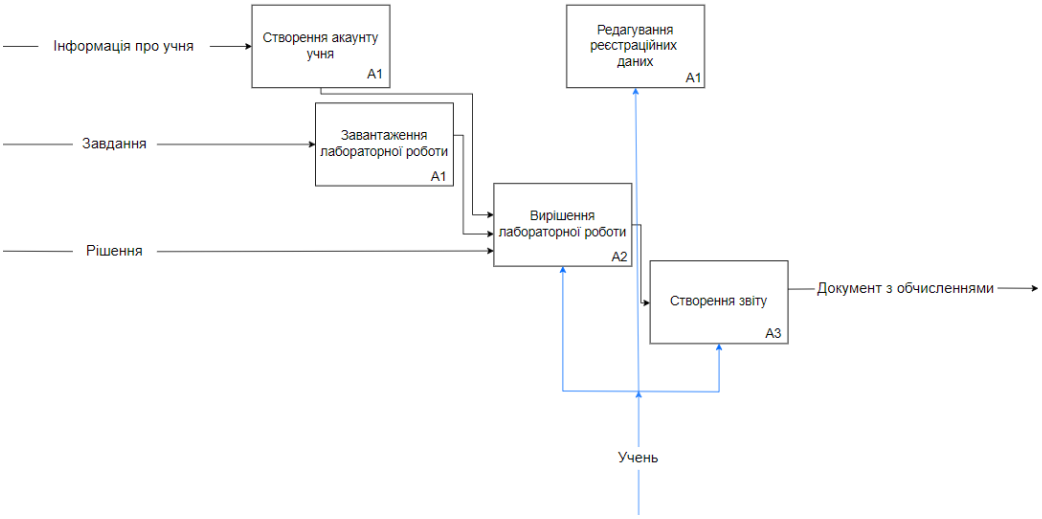

Рисунок 1.5 – Декомпозиція функціональної схеми

На вхід вводяться дані учня, після чого, він має змогу завантажити роботу та виконати обчислення завдання. Наступним кроком функціональної схеми є створення звіту, яким можна поділитися з вчителем для подальшої перевірки. Передбачена можливість зміни реєстраційних даних у випадку отримання нового прізвища дитини.

Для розробки майбутнього програмного забезпечення було створено 9 прототипів екранів інтерфейсу, які демонструють основний функціонал програми.

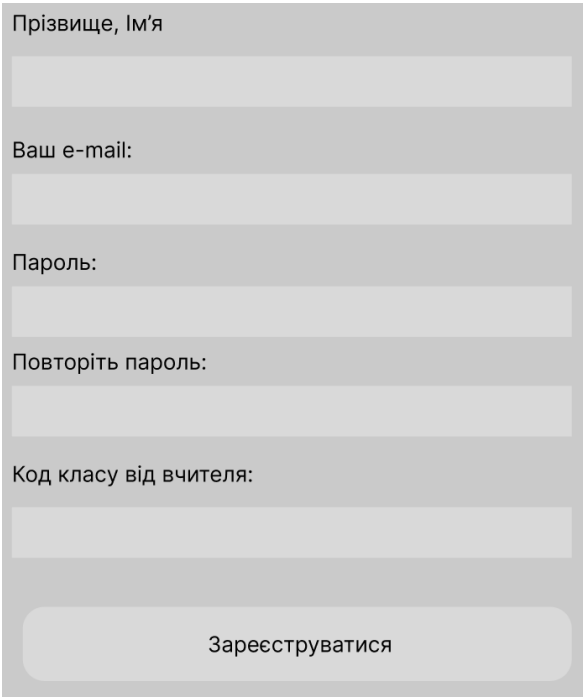

Рисунок 1.6 – Прототип екрану реєстрації учня

Передбачається, що учень вводить такі дані: прізвище, ім’я, електронну адресу, пароль та унікальний код класу, який генерує вчитель у своєму особистому кабінеті.

Представлено на рисунку 1.6

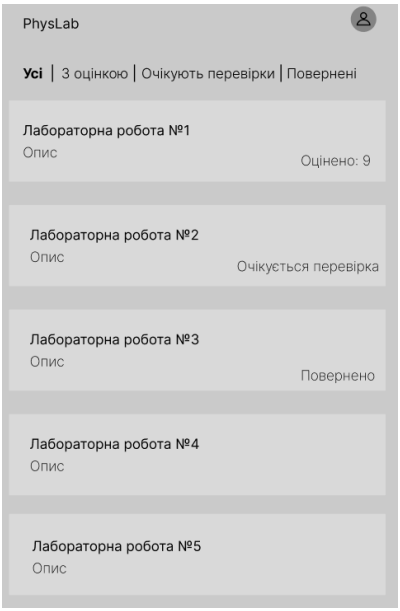

Рисунок 1.7 – Прототип головного екрану учня

Пройшовши процедуру реєстрації, користувачу стають доступні лабораторні роботи для виконання. Також присутня можливість відстеження стану перевірки, фільтри. Представлено на рисунку 1.7

Рисунок 1.8 – Прототип екрану виконання лабораторної роботи

Екран лабораторної роботи складається з таких елементів: тема, мета, інтерактивна анімація, кнопка для взаємодії з нею, поле для внесення розрахунків, кнопка для надсилання роботи вчителю та плаваючий калькулятор.

Переглянемо особистий кабінет вчителя.

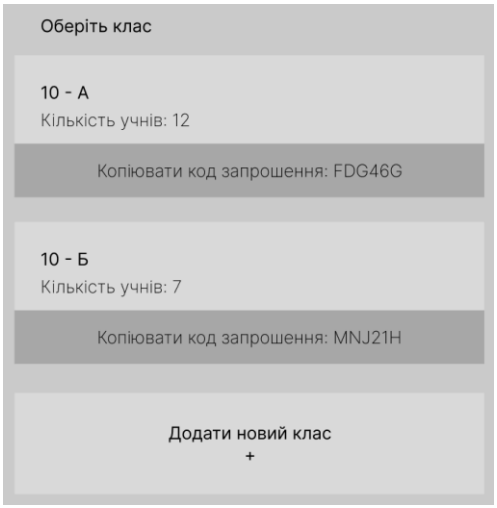

Рисунок 1.9 – Прототип екрану вибору класу

Після реєстрації, вчителю стає доступний екран з вибором класу для перевірки робіт. Є можливість створити декілька профілів та скопіювати код запрошення. Представлено на рисунку 1.9

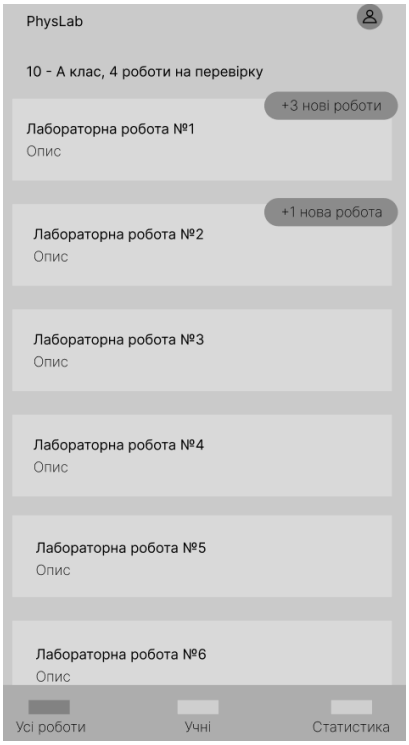

Рисунок 1.10 – Прототип головного екрану вчителя

Після вибору класу, вчитель може відслідковувати процес виконання лабораторних робіт на головному екрані застосунку. Коли учні здають свої розрахунки на перевірку, на картці з’являється сповіщення з кількістю нових зданих робіт. Представлено на рисунку 1.10

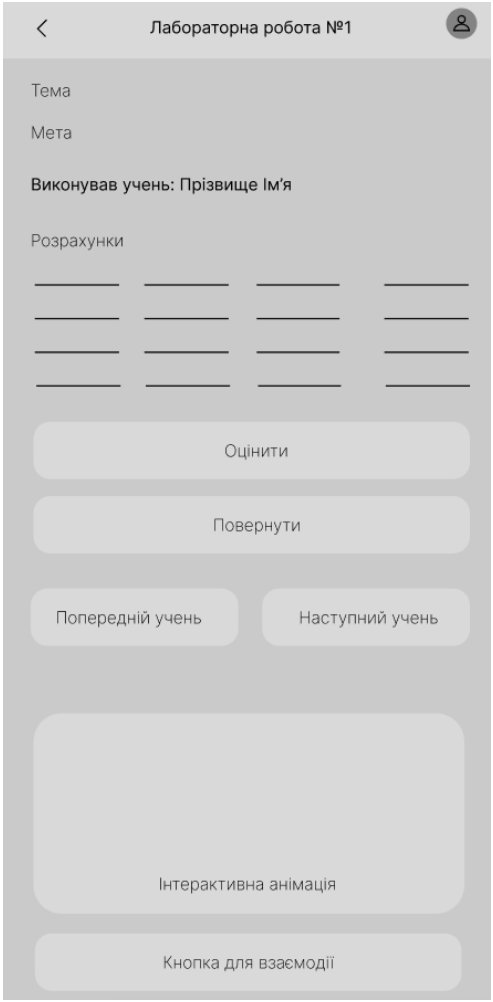

Рисунок 1.11 – Прототип екрану оцінювання роботи

На екран перевірки додалися необхідні для оцінювання елементи інтерфейсу: кнопка для оцінки або повернення роботи, ім’я та прізвище виконавця, кнопка переходу до наступного учня, що дозволяє прискорити процес перевірки. Представлено на рисунку 1.11

Вчитель може переглянути інтерактивну анімацію для кращого розуміння того, що бачить учень у своєму кабінеті.

Рисунок 1.12 – Прототип екрану «Учні»

Екран «Учні» показує детальну інформацію про створений клас. Міститься інформація про кількість учнів, їх виконану роботу та код запрошення. Представлено на рисунку 1.12

Рисунок 1.13 – Прототип екрану «Статистика»

Екран «Статистика» дозволяє переглянути успішність класу у звичному форматі, що має вигляд шкільного журналу. Вчителеві надається можливість експортувати таблицю для подальшого ведення звітності.

Представлено на рисунку 1.13

2 ПРОЄКТНИЙ РОЗДІЛ

2.1 Визначення апаратних і програмних вимог до ПЗ

Майбутній програмний застосунок повинен запускатися та належно працювати на широкому спектрі мобільних пристроїв.

Пристрій має бути обладнаний багатоядерним процесором із частотою не менше 1,8 ГГц, оперативною пам’яттю щонайменше 4 Гб, вільним простором у сховищі не менше 200 Мб. Також передбачається використання дисплеїв з роздільною здатністю від 720p, що дозволить коректно відображати інтерфейс користувача.

Застосунок вимагає періодичного доступу до мережі Інтернет для синхронізації даних хмарою Firestore. Мінімальна швидкість каналу має становити не менше 1 Мбіт/с для коректної роботи, включно з відправленням результатів лабораторних робіт і отриманням оновлень від викладача.

У випадку тимчасової відсутності підключення має бути забезпечений режим обмеженої автономної роботи з подальшою синхронізацією після відновлення зв’язку.

На рівні серверної частини застосунок використовує хмарну базу даних Firebase Firestore для зберігання інформації про користувачів, віртуальні класи, лабораторні роботи та результати обчислень.

Користувачами системи є студенти та викладачі, тому вимоги до інтерфейсу включають зручність навігації, підтримку автентифікації за допомогою електронної пошти та пароля, а також можливість обробки і відображення матриць обчислень.

Отже, було визначено наступні мінімальні апаратні і програмні вимоги:

- операційна система: Android 11;
- оперативна пам’ять: 4 Гб;
- процесор: не менше 1,8 ГГц;
- місце у сховищі: від 200 Мб;

– вихід в мережу Інтернет;

– роздільна здатність екрану 720р.

2.2 Моделювання бази даних

Для організації зберігання та обробки даних у мобільному застосунку було вирішено використати хмарний сервіс Google Firebase, а саме його документо-орієнтовану базу даних Cloud Firestore.

Firestore працює з документами, згрупованими у колекції, а формат цих документів близький до JSON. Це добре узгоджується з форматом даних, що використовується у React Native.

Архітектура БД будується навколо трьох ключових сутностей, а саме:

– користувачі;

– лабораторні роботи;

– результати виконання.

Для кожної сутності передбачена власна колекція. Окрім стандартної інформації про автентифікацію, документи містять дані про профілі користувачів.

Профілі вчителів містять списки класів, якими вони керують, а профілі учнів – посилання на групу, до якої вони належать. Такий підхід дозволяє швидко отримати всю потрібну інформацію без складних запитів, що є критичним для мобільних пристроїв.

Основний навчальний контент, а саме методичні вказівки до виконання лабораторних робіт зберігається у відповідній колекції.

Особливістю такого підходу є те, що методичні дані зберігаються у базі даних, а не в коді застосунку. Це дасть змогу змінювати умови завдань або виправляти неточності без збірки APK-файлу, що значно спрощує підтримку та оновлення навчального матеріалу

Взаємодія між учнями та викладачами забезпечується через колекцію результатів лабораторних робіт. Коли студент надсилає свою роботу на

перевірку, створюється окремий документ із введеними даними, розрахунками та часовою міткою. Оскільки Firestore підтримує механізм реального часу, інтерфейс вчителя миттєво отримує повідомлення про нову роботу.

Такий підхід дозволяє уникати конфліктів, забезпечити цілісність даних і гарантує, що кілька користувачів можуть працювати з журналом одночасно без помилок.

2.3 Основні алгоритми програмного комплексу

На рисунку 2.1 зображений алгоритм реєстрації нового користувача у програмному комплексі. Користувач, вчитель або учень, проходить процедуру реєстрації шляхом введення даних, система валідує їх та у разі правильності відкриває головну сторінку в залежності від профілю.

У іншому випадку система показує повідомлення про помилку та шляхи її вирішення.

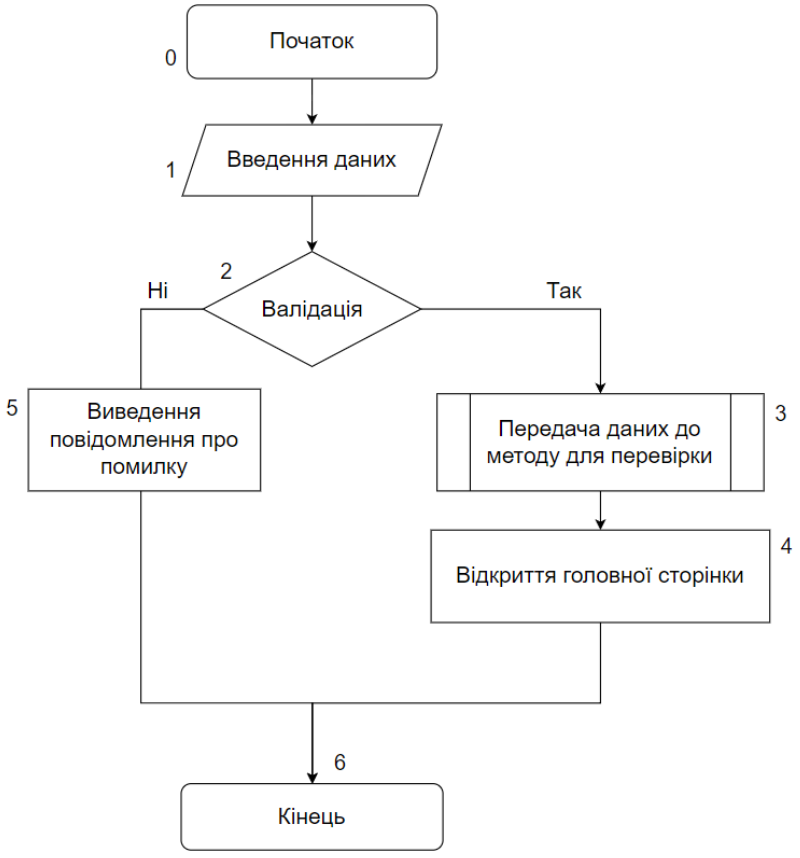

Рисунок 2.1 – Блок-схема реєстрації нового користувача

На рисунку 2.2 зображена блок-схема, яка показує етапи виконання лабораторної роботи з профілю учня. Користувач обирає лабораторну роботу із списку, вводить дані своїх обчислень та система надсилає дані вчителеві на перевірку.

Рисунок 2.2 – Блок-схема виконання лабораторної роботи

На рисунку 2.3 зображена блок-схема, яка показує етапи перевірки лабораторної роботи вчителем. Спочатку користувач обирає лабораторну роботу для перевірки та у разі правильного виконання виставляє оцінку. Повернення роботи передбачене у разі неправильного виконання.

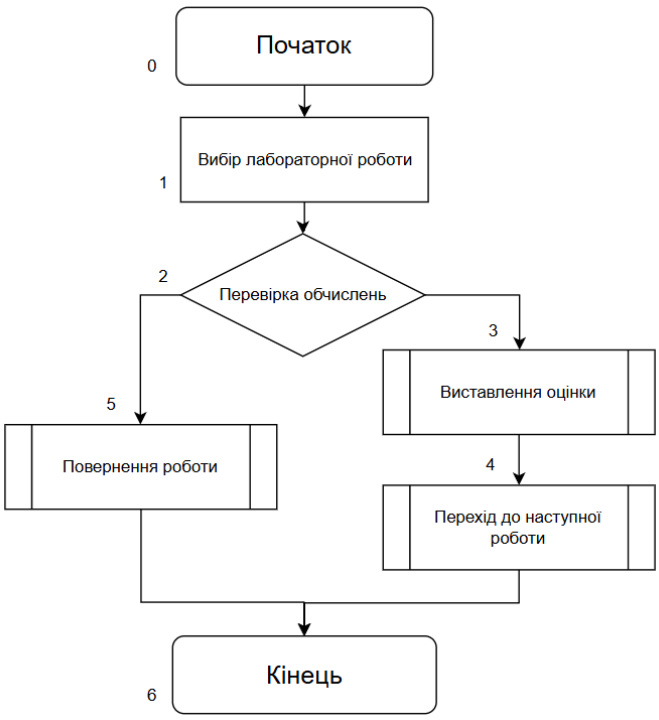

Рисунок 2.3 – Перевірка лабораторної роботи

На рисунку 2.4 зображене резервне копіювання звіту лабораторної роботи яке відбувається автоматично після згенерованого звіту.

Рисунок 2.4 – Резервне копіювання звіту

Отже, нами були розглянуті основні алгоритми програмного комплексу.

2.4 Аналіз існуючих рішень

Серед аналогічних рішень можна виділити такий інструмент як «PhEt Interactive Simulations» від університету Колорадо. Цей ресурс містить у собі набір інтерактивних віртуальних інструментів з фізики, які дають можливість учням проводити експерименти використовуючи візуальну складову.

Сильними сторонами можна відзначити простоту інтерфейсу, наочність та доступність, проте у цій системі не реалізовані функції для індивідуального обліку результатів учнів та оцінювання робіт викладачами, що обмежує її використання у освітньому процесі.

Інтерфейс програмного комплексу зображений на рисунку 2.5

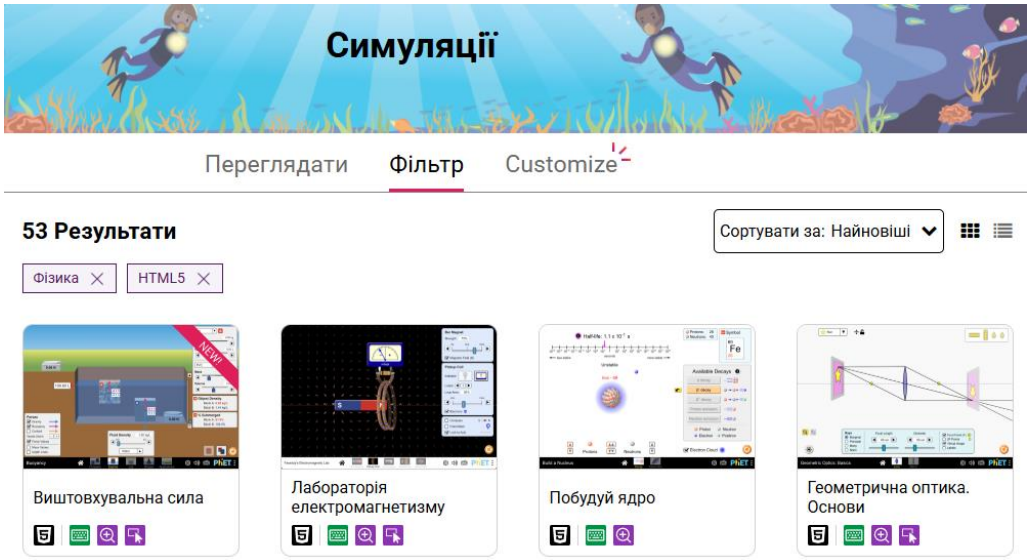

Рисунок 2.5 – Головна сторінка програмного комплексу

Наступне програмне забезпечення має назву «PhysicsApp», що має можливості для обчислення інтерактивних симуляцій.

Ця версія програми надає користувачам доступ до різних розділів вивчення фізики від «Механіки» до «Квантової механіки»

Приклад інтерфейсу користувача зображений на рисунку 2.6

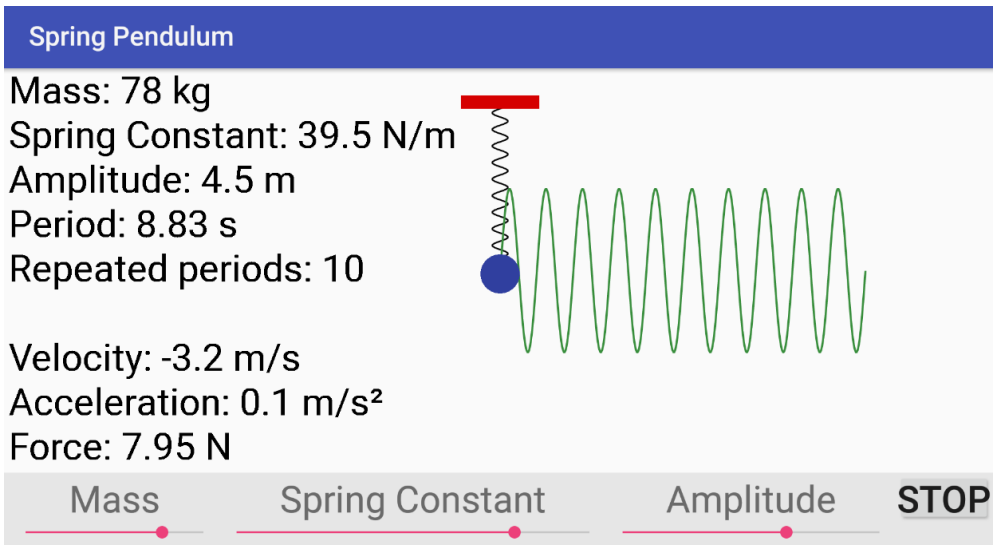

Рисунок 2.6 – Інтерфейс застосунку «PhysicsApp»

Серед переваг варто зазначити:

– зрозумілий інтерфейс;

– можливість автоматизації обчислень шляхом вбудованого калькулятора;

– анімації, якими можна керувати у режимі реального часу.

Серед недоліків варто зазначити:

– не має онлайн-можливостей для синхронізації даних;

– відсутність можливості оцінювання робіт;

– відсутність можливості для створення звітності по виконаним роботам.

Іншим прикладом є «Labster» - платформа з великою бібліотекою віртуальних стендів, орієнтованих на університети. ПЗ пропонує реалістичні тривимірні симуляції та включає елементи оцінювання.

Серед недоліків можна зазначити:

– проєкт є комерційним, що робить його недоступним для широкого кола користувачів;

– тривимірні симуляції звужують коло потенційних користувачів задаючи високі системні вимоги;

– відсутність кросплатформеності, проєкт доступний лише для персональних комп’ютерів.

Інтерфейс програми зображений на рисунку 2.7

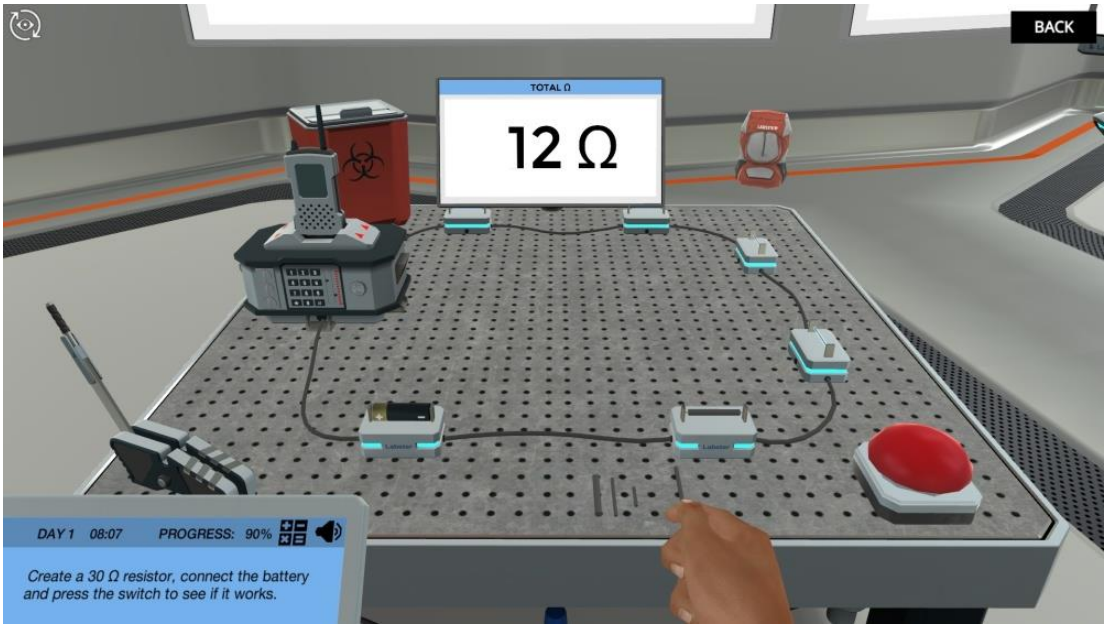

Рисунок 2.7 – Інтерфейс програми «Labster»

Існуючі рішення частково закривають потреби з вивчення фізики, але не поєднують у собі можливості проведення лабораторних експериментів, оцінювання, обліку студентів, внесення обчислень та звітів одночасно.

Майбутній програмний комплекс має перевагу в тому, що він об’єднує всі ці складові забезпечуючи зручний цикл взаємодії між учнем та викладачем у межах єдиної системи.

2.5 Програмна реалізація інтерактивних симуляцій

Інтерактивні симуляції базуються на фізичних анімаціях, створених у середовищі Adobe After Effects, які потім імпортуються у форматі JSON через спеціальний плагін Bodymovin.

Цей метод забезпечує найбільшу продуктивність, так як не потребує обчислень на стороні користувача, більшість симуляцій вже обчислена наперед.

Використання Lottie дозволяє зберігати невеликий розмір файлів анімацій, що є важливим для мобільних пристроїв з обмеженими ресурсами. Завдяки векторній графіці анімації масштабуються без втрати якості та відтворюються на будь-яких пристроях.

Інтерактивність досягається у поєднанні з логікою застосунку на основі React Native, режисовані анімації змінюють одне одну шляхом натискання кнопок користувачем чи перетягуванням елементів. Наприклад, користувач може змінювати кут нахилу математичного маятника чи змінювати силу струму у стенді шляхом натискання кнопки.

Основні методи для керування інтерактивним компонентом:

– play() – відповідає за старт програвання анімації;

– pause() – відповідає за призупинку анімації;

– reset() – дозволяє скинути анімацію;

– resolveKeyPath() - приймає ключовий шлях і повертає список з нуля або більше існуючих ключових шляхів.

Приклад програмної реалізації у середовищі React Native:

```
export default function PhysLabSimulation() {
  const animationRef = useRef(null);

  return (
    <View style={styles.container}>
      {/* Lottie-анімація */}
      <LottieView
    ref={animationRef}
    source={require("./assets/experiment.json")} // твоя анімація
з After Effects
    style={{ width: 300, height: 300 }}
    loop={false} // не зациклюється
  />
  {/* Кнопки для керування */}
  <View style={styles.buttons}>
    <Button       title="       Почати"       onPress={()       =>
  animationRef.current.play()} />
```

```
          <Button       title="      Зупинити"       onPress={()       =>
        animationRef.current.pause()} />
          <Button       title="       Скинути"       onPress={()       =>
        animationRef.current.reset()} />
          <Button
            title=" Виміряти"
            onPress={() => animationRef.current.play(50, 120)} // тільки
        частина анімації
          />
        </View>
      </View>
    );
  }
```

Adobe After Effects також має широкий інструментарій для імітації реалістичних анімацій шляхом математичних експресій:

– drop bounce expression – використовується для імітації інерції об’єктів;

– wiggle expression – використовується для імітації пружинних об’єктів;

Для їх програмування використовуються такі методи:

– velocityAtTime() – допомагає керувати швидкістю експресії;

– math.sin() – обчислення синусу для розрахунку поведінки експресії;

– math.exp() – повертає натуральний логарифм для обчислень положення;

– nearestKey() – повертає найближчий об'єкт ключового кадру.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування і вибір інструментарію розробки ПЗ

Перш ніж розпочати розробку програмного комплексу, потрібно визначити на яких технологіях він буде створений. Від цього залежить гнучкість розробки, швидкість роботи програми та можливість виходу застосунку на велику кількість наявних платформ.

Flutter – це фреймворк, який дозволяє створювати застосунки одразу для декількох платформ, використовуючи одну мову програмування Dart.

Серед переваг можна виділити:

– висока продуктивність завдяки компіляції у нативний код ARM, x86;

– має велику кількість бібліотек і компонентів для розробки інтерфейсу які можна легко налаштовувати;

– наявна можливість миттєво бачити зміни у коді без перезапуску програми.

Серед недоліків варто зазначити:

– застосунки займають більше місця у сховищі пристрою порівняно з нативними;

– частина бібліотек ще не мають повної підтримки або оновлюються повільніше, ніж у JavaScript-екосистемі.

React Native – фреймворк від компанії Meta, що базується на JavaScript і технології React. Має можливість створювати мобільні програми, використовуючи знайомі веб-розробникам інструменти.

Серед переваг можна виділити:

– обширна спільнота та велика кількість бібліотек;

– код можна використовувати для мобільного і веб-середовища;

– підтримується на Android та IOS.

До недоліків відносяться:

– продуктивність нижча, ніж у Flutter, адже частина логіки виконується на JavaScript рушії;

– виникають проблеми з сумісністю версій бібліотек.

Нативна розробка Java/Kotlin – написання коду спеціально для однієї платформи Android.

До переваг відносяться:

– найвища продуктивність, оскільки застосунок напряму працює з системними ресурсами;

– повний доступ до датчиків, камери, файлової системи пристрою;

– стабільність роботи.

Серед недоліків можна виділити:

– відсутність кросплатформеності, що критично для розроблювального проєкту;

– процес створення застосунків займає набагато більше часу.

Для зберігання даних користувачів, налаштувань на результатів роботи учнів та вчителів вирішено використовувати хмарну базу даних від Google, а саме Firestore.

Серед переваг варто зазначити, що дані використовуються в релаьному часі, має просту інтеграцію з React Native та Flutter, підтримує авторизацію користувачів через Firebase Authentication, також автоматично масштабується з ростом кількості користувачів.

NoSQL база даних дозволяє легко зберігати структуровану інформацію.

З огляду на поставлені завдання, найбільш доцільним рішенням стало використання React Native у поєданні з Firestore.

3.2 Розробка бази даних

База даних зберігає інформацію у вигляді колекцій і документів, що мають гнучку структуру та можуть містити як прості поля (рядки, числа, масиви), так і вкладені об’єкти чи підколекції.

Firestore забезпечує високу швидкість обробки запитів і можливість масштабування без необхідності адміністрування серверів, що особливо важливо для мобільних застосунків з потенційно великою кількістю користувачів.

База даних складається з основних колекцій:

– /students – учні;

– /teachers – викладачі;

– /classes – віртуальні класи;

– /assignments – лабораторні роботи (вкладені у класи);

– /submissions – робіт (вкладені в завдання);

– Матриці – містять відповіді учнів з розв'язанням завдання, функціонують у складі /submissions як вкладені масиви.

Колекція /students/{studentId} складається з таких полів:

– first_name – поле для зберігання імені учня;

– last_name – поле для зберігання прізвища учня;

– email – поле для зберігання електронної пошти учня;

– password_hash - поле для зберігання паролю у хешованому вигляді;

– class_id – поле для зберігання ключа класу.

Колекція /teachers /{teacherId} складається з таких полів:

– first_name – поле для зберігання імені вчителя;

– last_name – поле для зберігання прізвища вчителя;

– email – поле для зберігання електронної пошти вчителя;

– password_hash - поле для зберігання паролю у хешованому вигляді.

Колекція /classes /{classId} складається з таких полів:

– name – поле для назви віртуального класу;

– teacher_id – поле для зберігання id вчителя;

– invite_code – поле для зберігання унікального коду класу.

Шаблон /classes/{classId}/assignments/{assignmentId} складається з таких полів:

– title – поле для зберігання назви лабораторної роботи;

– description – поле для зберігання опису лабораторної роботи;

– due_date – поле для зберігання строку здачі лабораторної роботи.

Шаблон/classes/{classId}/assignments/{assignmentId}/submissions/{submissionId} складається з таких полів:

– student_id – поле для зберігання id учня;

– submission_date – поле для зберігання дати/часу виконання роботи;

– grade – поле для зберігання оцінки роботи;

– matrix – масив для зберігання відповідей учня.

Отже, нами була розглянута база даних програмного комплексу.

3.3 Розробка користувацького інтерфейсу

Головний екран профілю учня зображений на рисунку 3.1

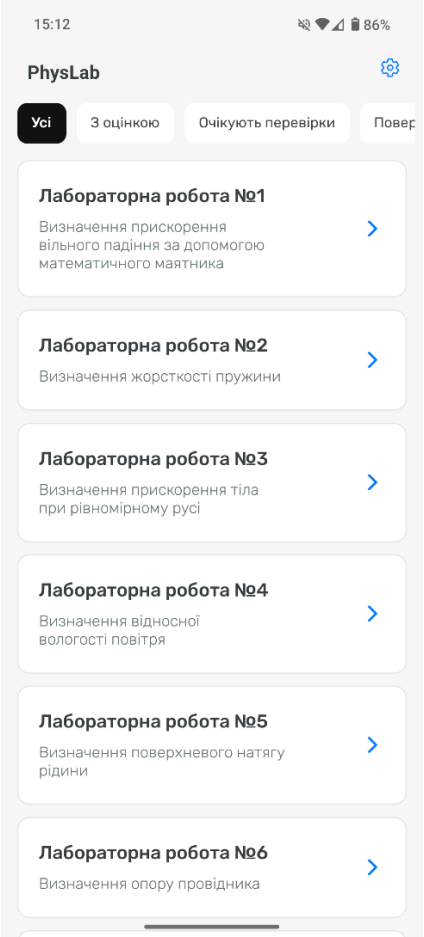

Рисунок 3.1 – Головний екран застосунку

На ньому розміщені лабораторні роботи, що доступні до виконання, фільтр та кнопка налаштувань.

Список реалізовано через компонент бібліотеки React Native, має назву «ReportItem» та пропси. Приклад коду:

```
// Тип для пропсів (вхідних даних) компонента ReportItem
type ReportItemProps = {
title: string; // Заголовок елемента
subtitle: string; // Підзаголовок елемента
onPress: () => void;};
```

Налаштування компоненту:

```
const ReportItem: React.FC<ReportItemProps> = ({ title,
subtitle, onPress }) => (
// TouchableOpacity - View на який можна натискати
<TouchableOpacity style={styles.itemContainer} onPress={onPress}
activeOpacity={0.7}>
{/* Контейнер для тексту, щоб він займав увесь доступний простір
зліва */}
<View style={styles.textContainer}>
<Text style={styles.itemTitle}>{title}</Text>
<Text style={styles.itemSubtitle}>{subtitle}</Text>
</View>
{/* Іконка стрілки праворуч */}
<Ionicons name="chevron-forward" size={24} color="#007AFF"
style={styles.chevronIcon} />
</TouchableOpacity>
);
```

Фільтр складається з масиву, що описує доступні опції:

```
// Використовується для побудови компонента FilterBar.
const FILTERS = [
{ key: 'all', label: 'Усі' },
{ key: 'graded', label: 'З оцінкою' },
{ key: 'pending', label: 'Очікують перевірки' },
{ key: 'returned', label: 'Повернені' },
];
```

Налаштування компоненту FilterBar для функціонування фільтру:

```
const FilterBar: React.FC<{ active: string; onChange: (k:
string) => void }> = ({ active, onChange }) => {
return (
<View style={styles.filterWrap}>
{/* ScrollView (horizontal) потрібен, щоб фільтри можна було
прокручувати вбік */}
<ScrollView horizontal showsHorizontalScrollIndicator={false}
contentContainerStyle={styles.filterScroll}>
{/* Динамічно створюємо кнопки фільтрів з масиву FILTERS */}
{FILTERS.map((f) => {
// Чи активний цей фільтр
const isActive = f.key === active;
```

```
return (
<TouchableOpacity
key={f.key} // Ключ обов'язковий для .map()
onPress={() => onChange(f.key)} // При натисканні викликаємо
функцію onChange з ключем
// Динамічно застосовуємо стилі: звичайний + активний (якщо
isActive)
style={[styles.filterPill, isActive && styles.filterPillActive]}
activeOpacity={0.8}
>
{/* Текст фільтра також з динамічними стилями */}
<Text style={[styles.filterText, isActive &&
styles.filterTextActive]}>{f.label}</Text>
</TouchableOpacity>);})}
```

На рисунку 3.2 зображений детальний перегляд роботи, що містить тему, мету та вказівки до виконання роботи.

Рисунок 3.2 – Детальний перегляд

На рисунку 3.3 зображена інтерактивна анімація та поля для внесення результатів обчислень.

Рисунок 3.3 – Інтерактивна анімація

Поля для вводу описує інтерфейс під назвою FormData:

```
interface FormData {
t: string[];
l: string[];
N: string;
T: string[];
g: string[];
g_avg: string;
epsilon: string;
}
```

Компонент :

```
const PendulumLabForm: React.FC<PendulumLabFormProps> = ({
setIsScrollEnabled,
onSubmit,
}) => {
// 1. Use useState to hold all form data
const [formData, setFormData] = useState<FormData>({
t: ['', '', '', ''],
l: ['', '', '', ''],
N: '',
T: ['', '', '', ''],
```

```
g: ['', '', '', ''],
g_avg: '',
epsilon: '',});
```

На рисунку 3.4 зображений екран вибору та створення класу у кабінеті вчителя.

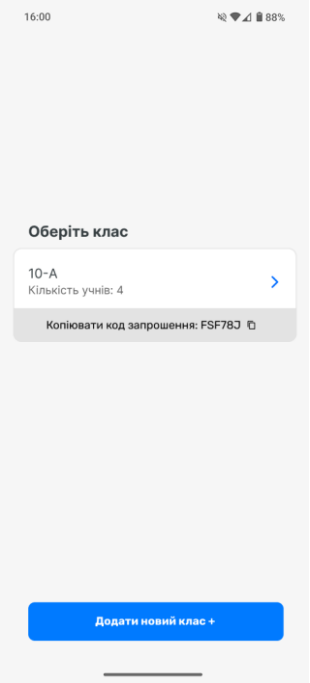

Рисунок 3.4 – Вибір класу для роботи

На рисунку 3.5 зображений головний екран вчителя, на якому можна обрати лабораторну роботу для перевірки.

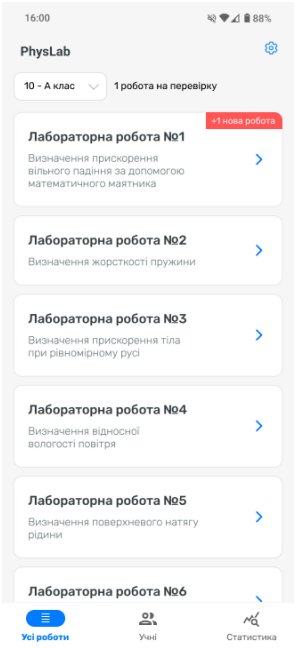

Рисунок 3.5 – Головний екран вчителя

На рисунку 3.6 зображений екран для ведення обліку виконаних робіт учня.

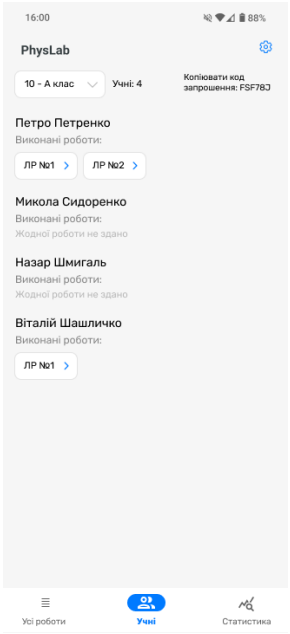

Рисунок 3.6 – Облік учнів

На рисунку 3.7 зображений екран для перевірки роботи учнів.

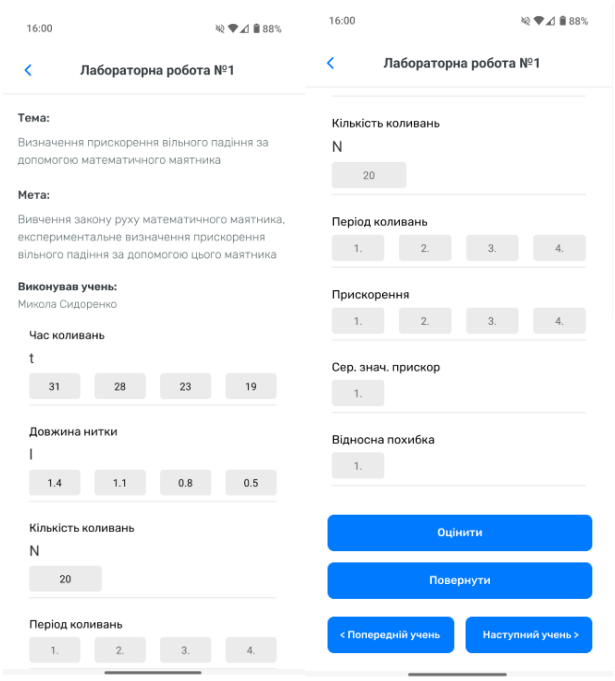

Рисунок 3.7 – Екран перевірки роботи учнів

3.4 Інструкція користувачів

Для користування програмним забезпеченням у режимі учня, потрібно пройти процедуру реєстрації акаунту:

– ввести електронну пошту, пароль, прізвище, ініціали та код класу від вчителя у відповідні поля та натиснути кнопку «Зареєструватися».

Для виконання лабораторної роботи потрібно:

– обрати завдання зі списку на головній сторінці застосунку;

– заповнити поля з обчисленнями та натиснути кнопку «Відправити».

Поля для обчислень супроводжуються формулою для обчислень.

Для перегляду статусу перевірки роботи, потрібно перейти на головну сторінку застосунку та обрати один з 4 пунктів фільтрації:

– всі роботи;

– очікують перевірки;

– оцінені;

– повернені.

У разі переведення учня до іншої школи, потрібно ввести новий код класу через меню «Профіль»:

– натиснути на кнопку «Профіль» у правому верхньому куту головної сторінки;

– обрати пункт «Змінити код класу»;

– ввести новий код у поле для вводу та натиснути кнопку «Змінити».

Для користування інтерактивною анімацією потрібно:

– прочитати інструкцію до виконання роботи;

– керувати віртуальним стендом через відповідну кнопку.

Процес реєстрації акаунту вчителя:

– ввести email, пароль, ініціали та прізвище у відповідні поля;

– натиснути на кнопку «Зареєструвтаися».

Для створення нового класу потрібно:

– перейти на головну сторінку застосунку;

– натиснути на кнопку «Новий клас»;

– ввести ім'я класу у відповідне поле;

– натиснути на кнопку «Створити клас»;

– натиснути на кнопку «Скопіювати код класу» та передати його учням.

Для перегляду списку учнів у класі потрібно:

– натиснути на кнопку «Список» у нижній частині екрану.

Для зміни класу на інший потрібно:

– перейти на головний екран застосунку;

– натиснути на ім'я класу та у випадаючому меню обрати інший клас або створити новий за потребою.

Для перегляду статистики успішності класу потрібно натиснути на відповідну кнопку, яка розміщена у нижній частині екрану та натиснути на кнопку «Оновити таблицю».

Для перевірки роботи потрібно:

– перейти на головний екран застосунку;

– обрати лабораторну роботу зі списку;

– обрати оцінку та натиснути кнопку «Оцінити»;

– у разі відправки роботи на доопрацювання натиснути на кнопку «Повернути», після чого можна перейти до оцінки роботи іншого учня.

Для видалення учня з класу потрібно:

– натиснути на кнопку «Список» у нижній частині екрану;

– натиснути на кнопку «Видалити», що знаходиться навпроти прізвища учня;

– підтвердити свій вибір у спливаючому вікні.

4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ, АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

4.1 Розробка методики проведення дослідження

Для оцінювання практичної ефективності розробленого програмного застосунку, була побудована спеціальна методика дослідження. Вона передбачала порівняння результатів виконання однакових навчальних завдань двома групами учасників, які використовували різні інструменти онлайн навчання.

Група 1 – Виконання завдань за допомогою онлайн-симуляцій сервісу PhET.

Група 2 – Виконання тих самих завдань, але за допомогою розробленого застосунку.

Кожна група містить у собі по 2 учасника.

Методика включала в себе:

- підготовку однакового навчального завдання;
- визначення початкового рівню знань учасників шляхом тестування;
- виконання завдання за допомогою відповідного інструменту.

Були виміряні такі параметри:

- час виконання завдання (хвилини);
- результат попереднього тестування (за 11-бальною шкалою);
- кількість допущених помилок;
- результат виконання завдання (за 11-бальною шкалою);
- відгуки користувачів.

Така методика дозволяє оцінити одразу кілька аспектів ефективності: навчальну результативність, швидкість виконання робіт та відгуки користувачів.

4.2 Формулювання залежностей між параметрами об’єкта дослідження

На основі отриманих даних була сформована таблиця 4.1, в якій описуються:

Учасник	Група	Попер. тест	Оцінка лр (11-б.)	Час виконання (хв)	Помилок
A	PhET	8	9	39	3
B	PhET	7	7	42	5
C	PhysLab	9	8	30	4
D	PhysLab	8	8	34	4

Таблиця 4.1 – Результати дослідження

У відгуках групи неодноразово відзначали простоту та легкість використання віртуальних стендів PhysLab в порвінянні з PhET.

Виходячи з даних дослідження, можна сформулювати залежність між інструментом та часом виконання. Використання нового застосунку призводило до зменшення часу на лабораторну роботу (у середньому 32 хв), ніж використання PhET (40,5 хв).

Також простежується кореляція між інструментом та суб’єктивним сприйняттям. Відгуки PhysLab суттєво позитивніші, а PhET містять вказівки на труднощі у навігації та високий поріг входження.

Це вказує на кращу адаптованість під навчальний процес та зрозумілість інтерфейсу для користувачів, що і відзначалося у відгуках.

4.3 Аналіз адекватності моделей, що були розроблені

Незважаючи на малу вибірку (n=4), модель дослідження демонструє відповідність параметрам реального освітнього процесу, усі вимірювані показники відображають ефективність нового інструменту.

Також розроблена методика може бути легко масштабована на більшу вибірку учнів.

Завдання, як і інструкції були однаковими для всіх, умови дослідження направлені на мінімізацію зовнішніх чинників.

Серед обмеження моделі варто зазначити надто малу вибірку для статистичних тестів, деякі суб'єктивні фактори учнів як мотивація, базові навички не враховані.

4.4 Практичне значення результатів роботи

Результати роботи мають важливе теоретичне значення для розвитку цифрової освіти та моделювання фізичних процесів.

Робота містить у собі:

– обґрунтування підходів до створення лабораторних експериментів для мобільних платформ;

– показано можливість коректного відтворення фізичних явищ у віртуальному середовищі;

– демонстрацію ефективності взаємодії користувача з моделлю;

Отримані результати можуть бути використані для подальших досліджень у сферах побудови навчальних систем, цифрового моделювання фізичних процесів.

Розроблений програмний комплекс має прикладне значення, оскільки дозволяє проводити лабораторні роботи з фізики без використання матеріального обладнання. Також забезпечує доступність експериментів для студентів у будь який час використовуючи лише мобільний пристрій.

Результати дослідження демонструють, що використання адаптованого під тонкощі навчального процесу та дружелюбного до користувачів застосунку сприяють зниженню часу роботи на виконання завдання та підвищують суб'єктивну оцінку зручності користування.

4.5 Економічна ефективність інновації

Розрахуємо трудомісткість і основну заробітна плату

Тривалість розробки проєкту: 4 місяці, 20 робочих днів на місяць.

Робочий час у місяці: 160 годин.

Погодинна ставка: 42,60 грн.

Основна зарплата розраховується:

$$З_{осн} = l_{год} * T_{год}$$

Де:

$l_{год} = 42{,}60$ грн

$T_{год} = 4$ місяці $\times$ 160 год/місяць $= 640$ годин

$$З_{осн} = 42{,}60 \times 640 = 27\,264 \text{грн.}$$

2. Додаткова зарплата

Додаткова зарплата становить 10% від основної:

$$З_{дод} = \frac{З_{осн} * Д\%}{100} = 27264 \times 0{,}1 = 2\,726{,}4 \text{ грн.}$$

3. Відрахування у соціальні фонди

Відрахування становлять 15% від суми основної та додаткової зарплати:

$$З_{соц} = \frac{(З_{осн} + З_{доп}) * С\%}{100} = (27264 + 2726{,}4) \times 0{,}15 = 4497{,}96 \text{грн.}$$

4. Загальновиробничі витрати

Приймається, що загальновиробничі витрати становлять 100% від основної зарплати:

$$З_{заг} = З_{осн} = 27264 \text{грн.}$$

5. Витрати на електроенергію

Середня споживана потужність Wi: орієнтовно 200 Вт (робота сервера/обладнання), час роботи — 640 годин:

$$B_E = P_E \sum Wi * t_{шт\,i} = 4{,}32 \sum 0{,}2 \times 640 = 552{,}96 \text{ грн.}$$

6. Виробнича собівартість

Виробнича собівартість:

$$S_{\text{виробнича}} = З_{\text{осн}} + З_{\text{дод}} + З_{\text{соц}} + З_{\text{заг}} + Р_{\text{електроенергії}} =$$

$$27\,264 + 2\,726{,}4 + 4\,497{,}96 + 27\,264 + 552{,}96 = 62\,305{,}32 \text{ грн.}$$

7. Оцінка терміну окупності

Економічний ефект (наприклад, заміна фізичних лабораторних робіт для 200 студентів на сервері за 4 місяці, з економією 200 грн/студента):

$$Е_{\text{ефект}} = 200 \times 200 = 40\,000 \text{ грн.}$$

Термін окупності:

$$T = \frac{S_{\text{виробнича}}}{Е_{\text{ефект}}} = \frac{62\,305{,}32}{40\,000} \approx 1{,}55 \text{ періоду} = Т * 4 \text{ місяці} \approx 6{,}2 \text{ місяця}$$

Собівартість мобільного застосунку PhysLab складає 62 305,32 грн, а термін окупності — приблизно 6,2 місяця, якщо поширювати застосунок за платною моделлю.

ВИСНОВОК

У результаті виконання кваліфікаційної роботи було створено програмний комплекс віртуальних лабораторних робіт з фізики для мобільних пристроїв під керуванням Android. У ході дослідження проведено аналіз предметної області, визначено вимоги до функціональності системи та встановлено особливості адаптації шкільних фізичних експериментів до цифрового формату.

Сформовано систему здачі та перевірки звітів. Структура програмного комплексу забезпечує можливість подальшого розширення функціоналу та інтеграції нових лабораторних робіт.

Створене програмне забезпечення підвищує доступність фізичних експериментів та дозволяє проводити лабораторні роботи у дистанційному або змішаному навчанні та сприяє швидшому виконанню завдань. Результати роботи демонструють практичну ефективність застосування віртуальних лабораторій у освітньому процесі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Як проводити лабораторні роботи на дистанційному навчанні? – URL: https://osvitoria.media/

2. Effectiveness of Virtual Simulations in Improving Secondary Students, – URL: https://www.e-iji.net/

3. Методичні основи використання Phet-симуляцій у процесі вивчення фізики, – URL: https://journals.tnpu.ternopil.ua/

4. A Systematic Review of the Effectiveness of Mobile Learning Tools in Enhancing Physics Education, – URL: https://ijlter.org/

5. Implementation of mobile augmented reality on physics learning in junior high school students, – URL: https://files.eric.ed.gov/

6. Eloquent JavaScript, 4th Edition, – URL: https://eloquentjavascript.net/

7. Learning React Native, – URL: https://pepa.holla.cz

8. React Native documentation, – URL: https://reactnative.dev

9. JavaScript Design Patterns, – URL: https://patterns.addy.ie/

10. React Native navigation, – URL: https://reactnavigation.org/

ДОДАТОК А

Код програми

Код головної сторінки учня:

```
// Тип для пропсів (вхідних даних) компонента ReportItem
type ReportItemProps = {
    title: string; // Заголовок елемента
    subtitle: string; // Підзаголовок елемента
    onPress: () => void; // Функція яка буде викликана при
натисканні
};
//Компонент "ReportItem":
const ReportItem: React.FC<ReportItemProps> = ({ title,
subtitle, onPress }) => (
    <TouchableOpacity style={styles.itemContainer}
onPress={onPress} activeOpacity={0.7}>
        {/* Контейнер для тексту, щоб він займав увесь
доступний простір зліва */}
        <View style={styles.textContainer}>
            <Text style={styles.itemTitle}>{title}</Text>
            <Text
style={styles.itemSubtitle}>{subtitle}</Text>
        </View>
        {/* Іконка стрілки праворуч */}
        <Ionicons name="chevron-forward" size={24}
color="#007AFF" style={styles.chevronIcon} />
    </TouchableOpacity>);
//Компонент AppBar:
const AppBar: React.FC<{ title: string; onRightPress?: ()
=> void }> = ({ title, onRightPress }) => {
    // Отримуємо об'єкт з відступами
    const insets = useSafeAreaInsets();
    return (
        // Застосовуємо відступ зверху
        <View style={[styles.appBar, { paddingTop:
insets.top }]}>
            <Text style={styles.appBarTitle}>{title}</Text>
            {/* Кнопка налаштувань*/}
            <TouchableOpacity onPress={onRightPress}
style={styles.appBarAction} activeOpacity={0.7}>
                <Ionicons name="settings-outline" size={20}
color="#007AFF" style={{ marginRight: 2 }} />
            </TouchableOpacity>
        </View>);};
//Визначення фільтрів
const FILTERS = [
    { key: 'all', label: 'Усі' },
    { key: 'graded', label: 'З оцінкою' },
    { key: 'pending', label: 'Очікують перевірки' },
    { key: 'returned', label: 'Повернені' },
];
```

```
      //Компонент "FilterBar":
      const FilterBar: React.FC<{ active: string; onChange: (k:
string) => void }> = ({ active, onChange }) => {
          return (
              <View style={styles.filterWrap}>
                  {/* щоб фільтри можна було прокручувати вбік
*/}
                  <ScrollView horizontal
showsHorizontalScrollIndicator={false}
contentContainerStyle={styles.filterScroll}>
                      {/* Динамічно створюємо кнопки фільтрів*/}
                      {FILTERS.map((f) => {
                          // Визначаємо чи активний цей фільтр
                          const isActive = f.key === active;
                          return (
                              <TouchableOpacity
                                  key={f.key}
                                  onPress={() => onChange(f.key)}
                                  style={[styles.filterPill,
isActive && styles.filterPillActive]}
                                  activeOpacity={0.8}>
                                  <Text
style={[styles.filterText, isActive &&
styles.filterTextActive]}>{f.label}</Text>
                              </TouchableOpacity>);})}
                  </ScrollView>
              </View>);};
      //Головний компонент "Index"
      export default function Index() {
          const [selectedLab, setSelectedLab] =
useState<LabReport | null>(null);
          // Стан для збереження ключа активного фільтра
          const [activeFilter, setActiveFilter] =
useState<string>('all');
          // --- Обробники подій (Handlers) ---
          const handleSelectLab = (lab: LabReport) => {
              if (lab.details) {
                  // Якщо деталі є оновлюємо стан, що спричинить
перерендер
                  // і показ LabDetailScreen
                  setSelectedLab(lab);
              } else {
                  alert('Детальна інформація для цієї
лабораторної роботи ще недоступна.');}};
          const filteredReports = useMemo(() => {
      // Використовуємо switch для перевірки активного фільтра
              switch (activeFilter) {
                  case 'graded':
                      return labReports.filter(r => (r as
any).grade != null);
                  case 'pending':
                      return labReports.filter(r => (r as
any).status === 'pending');
```

```
                    case 'returned':
                        return labReports.filter(r => (r as
any).status === 'returned');
                    default: // 'all'
                        // За замовчуванням (фільтр 'all')
повертаємо всі звіти
                        return labReports;
                }}, [activeFilter]); // Масив залежностей
            if (selectedLab) {
                return (
                    <LabDetailScreen
                        lab={selectedLab} // Передаємо дані про
обрану роботу
                        onBack={() => setSelectedLab(null)} //
Передаємо функцію для повернення назад
                    />);}
            return (
                <SafeAreaView style={styles.container}>
                    <StatusBar barStyle="dark-content" />
                    {/* Верхня панель*/}
                    <AppBar title="PhysLab" onRightPress={() =>
alert('Відкрити налаштування')} />
                    {/* Панель фільтрів */}
                    <FilterBar
                        active={activeFilter} // Передаємо поточний
активний фільтр
                        onChange={setActiveFilter}/>
          {/*Список лабораторних*/}
                    <FlatList
                        // Дані беремо з відфільтрованого масиву
                        data={filteredReports}
                        renderItem={({ item }) => (
                            <ReportItem
                                title={item.title}
                                subtitle={item.subtitle}
                                onPress={() =>
handleSelectLab(item)} // Натискання викликає обробник/>)}
keyExtractor={(item) => item.id}
contentContainerStyle={styles.listContentContainer}/></SafeAreaV
iew>);}
          //Всі стилі що використовуються в компоненті
          const styles = StyleSheet.create({
              // Стилі для головного контейнера (SafeAreaView)
              container: {
                  flex: 1, // Займати 100% висоти
                  backgroundColor: '#F6F6F6', // Світлий фон},
              //Стилі для відступів всередині FlatList
              listContentContainer: {
                  padding: 16, // Відступи з боків
                  paddingTop: 8, // Менший відступ зверху},
          //Стилі для FilterBar
              filterWrap: {
                  paddingVertical: 8,
```

```
            paddingHorizontal: 8,
            backgroundColor: '#F6F6F6',
            borderBottomWidth: 0,},
        filterScroll: {
            paddingLeft: 8,
            paddingRight: 16,
            alignItems: 'center',},
        // Стиль кнопки фільтра
        filterPill: {
            backgroundColor: '#FFFFFF',
            paddingVertical: 8,
            paddingHorizontal: 14,
            borderRadius: 10,
            marginRight: 10,
            minHeight: 40,
            justifyContent: 'center',},
        // Стиль активної пігулки
        filterPillActive: {
            backgroundColor: '#111111', // Темний фон},
        // Стиль тексту в пігулці
        filterText: {
            fontSize: 14,
            color: '#444',
            fontFamily: 'Rubik_400Regular',},
        // Стиль тексту в активній пігулці
        filterTextActive: {
            color: '#ffffffff', // Білий колір
            fontFamily: 'Rubik_500Medium',},
        badge: {
            position: 'absolute',
            right: -6,
            top: -6,
            backgroundColor: '#FF3B30',
            paddingHorizontal: 6,
            height: 20,
            borderRadius: 10,
            justifyContent: 'center',
            alignItems: 'center',},
        badgeText: { color: '#fff', fontSize: 12 },
        itemContainer: {
            backgroundColor: '#FFFFFF',
            borderRadius: 12,
            padding: 20,
            marginBottom: 12,
            borderColor: '#E4E4E4',
            borderWidth: 1,
            flexDirection: 'row',
            alignItems: 'center',
            justifyContent: 'space-between', // Розтягнути по
горизонталі},
        textContainer: {
            flex: 1, // Займати весь доступний простір
            marginRight: 16, // Відступ від іконки},
```

```
          itemTitle: {
              fontSize: 18,
              fontFamily: 'Rubik_500Medium',
              color: '#363E41',
              marginBottom: 4,},
          itemSubtitle: {
              fontSize: 15,
              color: '#626F71',
              fontFamily: 'Rubik_300Light',},
          chevronIcon: {
              marginLeft: 6, // Невеликий відступ зліва},
          appBar: {
              paddingHorizontal: 16,
              fontFamily: 'Rubik_700Bold',
              flexDirection: 'row', // Елементи в рядок
              alignItems: 'center',
              justifyContent: 'center',
              backgroundColor: '#F6F6F6',},
          appBarTitle: {
              flex: 1, // Займати весь простір
              fontFamily: 'Rubik_700Bold',
              paddingVertical: 10,
              textAlign: 'left', // Текст зліва
              marginLeft: 10,
              fontSize: 18,
              fontWeight: '700',
              color: '#363E41',},
          appBarAction: {
              position: 'absolute', // Позиціонувати відносно
appBar
              right: 12,
              top: 48,
              padding: 8,},
          chevron: {},
          appBarActionText: {},});
```

Код сторінки лабораторної роботи:

```
      type LabDetailScreenProps = {
        lab: LabReport;
        onBack: () => void;};
      // Панель заголовка AppBar
      const AppBar: React.FC<{ title: string; onBack: () => void
}> = ({ title, onBack }) => {
        const insets = useSafeAreaInsets();
        return (
          <View style={[styles.appBar, { paddingTop: insets.top
}]}>
            <TouchableOpacity onPress={onBack}
style={styles.appBarLeft} activeOpacity={0.7}>
              <Ionicons name="chevron-back" size={24}
color="#007AFF" style={styles.backIcon} />
            </TouchableOpacity>
```

```
            <Text style={styles.appBarTitle}>{title}</Text>
            <View style={styles.appBarActionPlaceholder} />
          </View>);};
        // Описуємо як мають виглядати дані нашої форми
        interface FormData {
          t: string[];
          l: string[];
          N: string;
          T: string[];
          g: string[];
          g_avg: string;
          epsilon: string;}
        // Описуємо пропси для нашого компонента
        type PendulumLabFormProps = {
          setIsScrollEnabled: (enabled: boolean) => void;
          onSubmit: (data: FormData) => void;};
        const PendulumLabForm: React.FC<PendulumLabFormProps> = ({
          setIsScrollEnabled,
          onSubmit,}) => {
          //useState, щоб зберігати всі дані форми
          const [formData, setFormData] = useState<FormData>({
            t: ['', '', '', ''],
            l: ['', '', '', ''],
            N: '',
            T: ['', '', '', ''],
            g: ['', '', '', ''],
            g_avg: '',
            epsilon: '',});
          // функції для оновлення стану
          const handleArrayInputChange = (
            field: 't' | 'l' | 'T' | 'g',
            index: number,
            value: string) => {
            setFormData((prev) => {
              // копія конкретного масиву
              const newArray = [...prev[field]];
              newArray[index] = value;
              // Повертаємо новий об'єкт стану
              return { ...prev, [field]: newArray };});};
          const handleInputChange = (
            field: 'N' | 'g_avg' | 'epsilon',
            value: string) => {
            setFormData((prev) => ({
              ...prev,
              [field]: value,}));};
          const handleFocus = () => {
            setIsScrollEnabled(true);};
          const handleFormSubmit = () => {
            // Передаємо заповнені дані форми батьківському
компоненту
            onSubmit(formData);};
          return (
            <View style={styles.formContainer}>
```

```
            {/* Час коливань */}
            <View style={styles.section}>
              <Text style={styles.label}>Час коливань</Text>
              <Text style={styles.symbol}>t</Text>
              <View style={styles.inputsRow}>
                {[0, 1, 2, 3].map((n) => (
                  <TextInput
                    key={n}
                    style={styles.input}
                    placeholder={`${n + 1}.`}
                    keyboardType="numeric"
                    onFocus={handleFocus}
                    //Підключаємо стан до поля вводу
                    value={formData.t[n]}
                    onChangeText={(text) =>
handleArrayInputChange('t', n, text)}
                  />
                ))}
              </View>
            </View>
            {/* Довжина нитки */}
            <View style={styles.section}>
              <Text style={styles.label}>Довжина нитки</Text>
              <Text style={styles.symbol}>l</Text>
              <View style={styles.inputsRow}>
                {[0, 1, 2, 3].map((n) => (
                  <TextInput
                    key={n}
                    style={styles.input}
                    placeholder={`${n + 1}.`}
                    keyboardType="numeric"
                    onFocus={handleFocus}
                    value={formData.l[n]}
                    onChangeText={(text) =>
handleArrayInputChange('l', n, text)}
                  />))}</View></View>
            {/* Кількість коливань */}
            <View style={styles.section}>
              <Text style={styles.label}>Кількість
коливань</Text>
              <Text style={styles.symbol}>N</Text>
              <TextInput
                style={[styles.input, { width: 100 }]}
                placeholder="20"
                keyboardType="numeric"
                onFocus={handleFocus}
                value={formData.N}
                onChangeText={(text) => handleInputChange('N',
text)}/>
            </View>
            {/* Період коливань */}
            <View style={styles.section}>
              <Text style={styles.label}>Період коливань</Text>
```

```
              <View style={styles.inputsRow}>
                {[0, 1, 2, 3].map((n) => (
                  <TextInput
                    key={n}
                    style={styles.input}
                    placeholder={`${n + 1}.`}
                    keyboardType="numeric"
                    onFocus={handleFocus}
                    value={formData.T[n]}
                    onChangeText={(text) =>
handleArrayInputChange('T', n, text)}
      />))}
              </View>
            </View>
            {/* Прискорення */}
            <View style={styles.section}>
              <Text style={styles.label}>Прискорення</Text>
              <View style={styles.inputsRow}>
                {[0, 1, 2, 3].map((n) => (
                  <TextInput
                    key={n}
                    style={styles.input}
                    placeholder={`${n + 1}.`}
                    keyboardType="numeric"
                    onFocus={handleFocus}
                    value={formData.g[n]}
                    onChangeText={(text) =>
handleArrayInputChange('g', n, text)}
      />))}
              </View>
            </View>
            {/* Сер. знач. прискор */}
            <View style={styles.section}>
              <Text style={styles.label}>Сер. знач.
прискор</Text>
              <TextInput
                style={styles.input}
                placeholder="1."
                keyboardType="numeric"
                onFocus={handleFocus}
                value={formData.g_avg}
                onChangeText={(text) =>
handleInputChange('g_avg', text)}
      /></View>
            {/* Відносна похибка */}
            <View style={styles.section}>
              <Text style={styles.label}>Відносна похибка</Text>
              <TextInput
                style={styles.input}
                placeholder="1."
                keyboardType="numeric"
                onFocus={handleFocus}
                value={formData.epsilon}
```

```
                onChangeText={(text) =>
handleInputChange('epsilon', text)}
              /></View>
            {/*кнопка "Зберегти" */}
            <TouchableOpacity style={styles.submitButton}
onPress={handleFormSubmit}>
              <Text
style={styles.submitButtonText}>Зберегти</Text>
            </TouchableOpacity>
          </View>);};
      //Основний екран
      const LabDetailScreen: React.FC<LabDetailScreenProps> = ({
lab, onBack }) => {
        const [isScrollEnabled, setIsScrollEnabled] =
useState(true);
        const handleFormSubmit = async (formData: FormData) => {
          console.log('Отримані дані форми:', formData);
          const currentStudentId = 'temp_student_123';
          try { //Додаємо новий документ у колекцію 'lr'
            const docRef = await addDoc(collection(db, 'lr'), {
              studentId: currentStudentId,
              labName: lab.title,
              labId: lab.id,
              data: formData,
              createdAt: serverTimestamp(), // Додає мітку часу з
боку сервера});
            console.log('Документ записано з ID: ', docRef.id);
            Alert.alert('Успіх', 'Лабораторну роботу
збережено.');} catch (e) {
            console.error('Помилка при додаванні документа: ',
e);
            Alert.alert('Помилка', 'Не вдалося зберегти
роботу.');}};return (
          <SafeAreaView style={styles.container}>
            <StatusBar barStyle="dark-content" />
            <AppBar title={lab.title} onBack={onBack} />
            <ScrollView
contentContainerStyle={styles.detailContentContainer}
scrollEnabled={isScrollEnabled}>
              <View style={styles.detailSection}>
                <Text style={styles.detailLabel}>Тема:</Text>
                <Text
style={styles.detailText}>{lab.details?.theme}</Text>
              </View>
              <View style={styles.detailSection}>
                <Text style={styles.detailLabel}>Мета:</Text>
                <Text
style={styles.detailText}>{lab.details?.goal}</Text>
              </View>
              <View style={styles.detailSection}>
                <Text style={styles.procedureTitle}>Хід
роботи</Text>
                {lab.details?.procedure.map((step, index) => (
```

```
                  <Text key={index}
style={styles.procedureStep}>{`${index + 1})
${step}`}</Text>))}</View>
              {/* WebView для симуляції */}
              <View style={styles.webViewContainer}>
                <WebView
                  source={require('../assets/pendul.html')}
                  scrollEnabled={false}
                  javaScriptEnabled={true}
                  originWhitelist={['*']}
                  onTouchStart={() => setIsScrollEnabled(false)}
                  onTouchEnd={() => setIsScrollEnabled(true)}/>
              </View> <PendulumLabForm
                setIsScrollEnabled={setIsScrollEnabled}
                // Передаємо функцію яка має логіку збереження в
Firestore
                onSubmit={handleFormSubmit}/>
            </ScrollView>
          </SafeAreaView>);};
```

Код реєстрації учня:

```
const RegistrationScreen = ({ navigation }) => {
  // Стани для полів вводу
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [classCode, setClassCode] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  // Стейт для помилки
  const [error, setError] = useState("");
  // Функція реєстрації учня
  const registerStudent = async () => {
    setError("");
    try {
      // Створення акаунту в Firebase Auth
      const userCredential = await
createUserWithEmailAndPassword(auth, email, password);
      // Отримуємо id користувача
      const userId = userCredential.user.uid;
      // Зберігаємо деталі учня в Firestore
      await setDoc(doc(db, "students", userId), {
        firstName,
        lastName,
        classCode,
        email,
        createdAt: new Date()
      });
      // Перехід на сторінку входу після успішної
реєстрації
      navigation.navigate("Login");} catch (err: any) {
      // Якщо Firebase повернув помилку — показуємо її
```

```
            setError(err.message);}};
        return (
          <View style={styles.container}>
            <Text style={styles.title}>Реєстрація учня</Text>
            {/* Поле: Ім'я */}
            <TextInput
              placeholder="Ім'я"
              value={firstName}
              onChangeText={setFirstName}
              style={styles.input}/>
            {/* Поле: Прізвище */}
            <TextInput
              placeholder="Прізвище"
              value={lastName}
              onChangeText={setLastName}
              style={styles.input}/>
            {/* Поле: Код класу, який надає вчитель */}
            <TextInput
              placeholder="Код класу від вчителя"
              value={classCode}
              onChangeText={setClassCode}
              style={styles.input}/>
            {/* Email */}
            <TextInput
              placeholder="Електронна пошта"
              value={email}
              onChangeText={setEmail}
              style={styles.input}
              keyboardType="email-address"/>
            {/* Пароль */}
            <TextInput
              placeholder="Пароль"
              value={password}
              onChangeText={setPassword}
              style={styles.input}
              secureTextEntry/>
            {/* Вивід помилки*/}
            {error ? <Text style={styles.error}>{error}</Text> :
null}
            {/* Кнопка реєстрації */}
            <TouchableOpacity style={styles.button}
onPress={registerStudent}>
              <Text
style={styles.buttonText}>Зареєструватися</Text>
            </TouchableOpacity>
            {/* Навігація до входу */}
            <TouchableOpacity onPress={() =>
navigation.navigate("Login")}>
              <Text style={styles.link}>Вже є акаунт?
Увійти</Text></TouchableOpacity></View>)};
```

Код реєстрації вчителя:

```
      const TeacherRegistrationScreen = ({ navigation }) => {
        // Стани для збереження введених полів
        const [firstName, setFirstName] = useState("");
        const [lastName, setLastName] = useState("");
        const [email, setEmail] = useState("");
        const [password, setPassword] = useState("");
        // Стейт для помилок
        const [error, setError] = useState("");
        // Реєстрація вчителя
        const registerTeacher = async () => {
          setError("");
          try {
            // Створення акаунту у Firebase Auth
            const userCredential = await
createUserWithEmailAndPassword(auth, email, password);
            const teacherId = userCredential.user.uid;
            // Генеруємо код класу
            // Додаємо вчителя в Firestore
            await setDoc(doc(db, "teachers", teacherId), {
              firstName,
              lastName,
              email,
              createdAt: new Date()});
            // Після успішної реєстрації ведемо на логін
            navigation.navigate("TeacherLogin");
          } catch (err: any) {
            // Виводимо помилку Firebase
            setError(err.message);}};
        return (
          <View style={styles.container}>
            <Text style={styles.title}>Реєстрація вчителя</Text>
            {/* Ім'я */}
            <TextInput
              placeholder="Ім'я"
              value={firstName}
              onChangeText={setFirstName}
              style={styles.input}
            />
            {/* Прізвище */}
            <TextInput
              placeholder="Прізвище"
              value={lastName}
              onChangeText={setLastName}
              style={styles.input}
            />
            {/* Email */}
            <TextInput
              placeholder="Електронна пошта"
              value={email}
              onChangeText={setEmail}
              keyboardType="email-address"
              style={styles.input}
            />
```

```
            {/* Пароль */}
            <TextInput
              placeholder="Пароль"
              value={password}
              onChangeText={setPassword}
              secureTextEntry
              style={styles.input}
            />
      {error ? <Text style={styles.error}>{error}</Text> : null}
            {/* Кнопка реєстрації */}
            <TouchableOpacity style={styles.button}
onPress={registerTeacher}>
              <Text
style={styles.buttonText}>Зареєструватися</Text>
            </TouchableOpacity>
            {/* Перехід на логін */}
            <TouchableOpacity onPress={() =>
navigation.navigate("TeacherLogin")}>
              <Text style={styles.link}>Вже маєте акаунт?
Увійти</Text>
            </TouchableOpacity>
          </View>);};
```

Код головної сторінки вчителя:

```
      import { Ionicons } from "@expo/vector-icons";
      // Тимчасові дані
      const LABS = [{
          id: "1",
          title: "Лабораторна робота №1",
          subtitle: "Визначення прискорення вільного падіння за
допомогою математичного маятника",
          new: true, // позначає що є нова робота від учня},{
          id: "2",
          title: "Лабораторна робота №2",
          subtitle: "Визначення жорсткості пружини",
          new: false,},{
          id: "3",
          title: "Лабораторна робота №3",
          subtitle: "Визначення прискорення тіла при рівномірному
русі",
          new: false,},{
          id: "4",
          title: "Лабораторна робота №4",
          subtitle: "Визначення відносної вологості повітря",
          new: false,},];
      const TeacherHomeScreen = () => {
        // Обраний клас у випадаючому списку
        const [selectedClass, setSelectedClass] = useState("10-А
клас");
        const [showDropdown, setShowDropdown] = useState(false);
        return (
```

```
          <View style={styles.container}>
            {/* Верхня панель */}
            <View style={styles.topBar}>
              <Text style={styles.appTitle}>PhysLab</Text>
              {/* Іконки у правому верхньому куті */}
              <View style={styles.iconsRow}>
                <Ionicons name="notifications-outline" size={24}
color="#000" style={{ marginRight: 12 }} />
                <Ionicons name="settings-outline" size={24}
color="#000" />
              </View>
            </View>
            {/* Вибір класу + кількість робіт */}
            <View style={styles.classRow}>
              {/* Випадаючий список класів */}
              <TouchableOpacity
                style={styles.classSelector}
                onPress={() => setShowDropdown(!showDropdown)}>
                <Text
style={styles.classText}>{selectedClass}</Text>
                <Ionicons name="chevron-down" size={18}
color="#000" />
              </TouchableOpacity>
              {/* Плашка кількості нових робіт */}
              <View style={styles.pendingBadge}>
                <Text style={styles.pendingText}>1 робота на
перевірку</Text>
              </View>
            </View>
            {/* Меню випадаючого списку */}
            {showDropdown && (
              <View style={styles.dropdownBox}>
                {["10-А клас", "10-Б клас", "11-А
клас"].map((cls) => (
                  <TouchableOpacity
                    key={cls}
                    style={styles.dropdownItem}
                    onPress={() => {
                      setSelectedClass(cls);
                      setShowDropdown(false);
                    }}>
                    <Text
style={styles.dropdownItemText}>{cls}</Text>
                  </TouchableOpacity>))}
              </View>
            )}
            {/* Список лабораторних робіт */}
            <FlatList
              data={LABS}
              keyExtractor={(item) => item.id}
              contentContainerStyle={{ paddingTop: 10,
paddingBottom: 80 }}
              renderItem={({ item }) => (
```

```
                <TouchableOpacity style={styles.labCard}>
                  {item.new && (
                    <View style={styles.newBadge}>
                      <Text style={styles.newBadgeText}>+1 нова
робота</Text>
                    </View>
                  )}
                  <Text
style={styles.labTitle}>{item.title}</Text>
                  <Text
style={styles.labSubtitle}>{item.subtitle}</Text>
                  <Ionicons name="chevron-forward" size={20}
color="#3A7BFF" style={styles.arrow} />
                </TouchableOpacity>
              )}/>
            {/* Нижня панель навігації */}
            <View style={styles.bottomNav}>
              <View style={styles.navItemActive}>
                <Ionicons name="book-outline" size={22}
color="#3A7BFF" />
                <Text style={styles.navTextActive}>Усі
роботи</Text>
              </View>
              <View style={styles.navItem}>
                <Ionicons name="people-outline" size={22}
color="#777" />
                <Text style={styles.navText}>Учні</Text>
              </View>
              <View style={styles.navItem}>
                <Ionicons name="stats-chart-outline" size={22}
color="#777" />
                <Text style={styles.navText}>Статистика</Text>
              </View></View></View>);};
```

Код сторінки з учнями:

```
      import { db } from '../firebaseConfig';
      import { collection, query, where, getDocs, DocumentData }
from 'firebase/firestore';
      interface Student {
        id: string; // ID документа учня
        fullName: string;
        studentId: string; // ID для зв'язку з Lab Reports
        completedLabs: string[]; }
      // Тип для сирого об'єкта звіту ЛР
      interface RawLabReport {
          studentId: string;
          labName: string;}
      const CURRENT_CLASS = '10 - А клас';
      const INVITATION_CODE = 'FSF78J';
      // Індикатор
```

```
      const LabChip: React.FC<{ labName: string }> = ({ labName
}) => {
        return (
          <TouchableOpacity style={styles.labChip}
activeOpacity={0.7}>
            <Text style={styles.labChipText}>{labName}</Text>
            <Ionicons name="chevron-forward" size={16}
color="#007AFF" />
          </TouchableOpacity>);};
      // Рядок учня
      const StudentRow: React.FC<{ student: Student }> = ({
student }) => {
        const hasCompletedLabs = student.completedLabs.length >
0;
        return (
          <View style={styles.studentRow}>
            <Text
style={styles.studentName}>{student.fullName}</Text>
            <Text style={styles.completedLabel}>Виконані
роботи:</Text>
            {!hasCompletedLabs ? (
              <Text style={styles.noLabsText}>Жодної роботи не
здано</Text>
            ) : (
              <View style={styles.labsContainer}>
                {student.completedLabs.map((lab, index) => (
                  <LabChip key={index} labName={lab} />))}
              </View>)}
          </View>);};
      // Header
      const StudentsHeader: React.FC<{ studentCount: number }> =
({ studentCount }) => (
        <View style={styles.header}>
          <View style={styles.classPicker}>
              <Text
style={styles.classText}>{CURRENT_CLASS}</Text>
              <Ionicons name="chevron-down" size={18}
color="#000" />
          </View>
          <View style={styles.infoRow}>
            <Text style={styles.studentCount}>Учні:
{studentCount}</Text>
            <View style={styles.inviteContainer}>
              <Text style={styles.inviteLabel}>Копіювати код
запрошення:</Text>
              <TouchableOpacity onPress={() => Alert.alert('Код
скопійовано', INVITATION_CODE)}>
                  <Text
style={styles.inviteCode}>{INVITATION_CODE}</Text>
              </TouchableOpacity>
            </View>
          </View>
        </View>
```

```
      );
      const StudentsScreen: React.FC = () => {
        const insets = useSafeAreaInsets();
        const [students, setStudents] = useState<Student[]>([]);
        const [isLoading, setIsLoading] = useState(true);
        useEffect(() => {
          const fetchStudentsData = async () => {
            try {
              setIsLoading(true);
              const studentsRef = collection(db, 'students');
              const studentsQuery = query(studentsRef,
where('class', '==', CURRENT_CLASS));
              const studentsSnapshot = await
getDocs(studentsQuery);
              const labsRef = collection(db, 'lab_reports');
              const labsSnapshot = await getDocs(labsRef);
              const labsData: RawLabReport[] =
labsSnapshot.docs.map(doc => ({
                  studentId: doc.data().studentId as string,
                  labName: doc.data().labName as string,
              }));
              const labsByStudentId: { [key: string]: string[] }
= labsData.reduce((acc, lab) => {
                const id = lab.studentId;
                if (!acc[id]) {
                  acc[id] = [];
                }
                acc[id].push(lab.labName);
                return acc;
              }, {});
        const loadedStudents: Student[] =
studentsSnapshot.docs.map(doc => {
                const data = doc.data() as DocumentData;
                const studentId = data.studentId as string;
                return {
                  id: doc.id,
                  fullName: `${data.name} ${data.surname}`,
                  studentId: studentId,
                  completedLabs: labsByStudentId[studentId] ||
[],};});
              setStudents(loadedStudents);
            } catch (error) {
              console.error('Помилка при завантаженні даних з
Firestore:', error);
              Alert.alert('Помилка', 'Не вдалося завантажити
список учнів.');
            } finally {
              setIsLoading(false);
            }
          };
          fetchStudentsData();
        }, []);
        return (
```

```
          <View style={styles.container}>
            <StatusBar barStyle="dark-content" />
            {/* Верхній AppBar */}
            <View style={[styles.appBar, { paddingTop: insets.top
}]}>
              <Text style={styles.appBarTitle}>PhysLab</Text>
              <TouchableOpacity style={styles.settingsButton}
activeOpacity={0.7}>
                <Ionicons name="settings-outline" size={24}
color="#000" />
              </TouchableOpacity>
            </View>
            <ScrollView
contentContainerStyle={styles.contentContainer}>
              {/* Хедер з вибором класу та кодом */}
              <StudentsHeader studentCount={students.length} />
              {/* Індикатор завантаження*/}
              {isLoading ? (
                  <View style={styles.loadingContainer}>
                      <ActivityIndicator size="large"
color="#007AFF" />
                      <Text
style={styles.loadingText}>Завантаження учнів...</Text>
                  </View>) : (
                  students.map((student) => (
                      <StudentRow key={student.id}
student={student} />))))}
            </ScrollView>
            {/* Нижній Tab Bar*/}
            <View style={[styles.tabBar, { paddingBottom:
insets.bottom }]}>
              <TouchableOpacity style={styles.tabItem}>
                <Ionicons name="list-outline" size={24}
color="#363E41" />
                <Text style={styles.tabText}>Усі роботи</Text>
              </TouchableOpacity>
              <TouchableOpacity style={styles.tabItemActive}>
                <Ionicons name="people" size={24} color="#007AFF"
/>
                <Text style={styles.tabTextActive}>Учні</Text>
              </TouchableOpacity>
              <TouchableOpacity style={styles.tabItem}>
                <Ionicons name="stats-chart-outline" size={24}
color="#363E41" />
                <Text style={styles.tabText}>Статистика</Text>
              </TouchableOpacity>
            </View>
          </View>);
      };
```

Код сторінки для перевірки лр:

```
import { Ionicons } from '@expo/vector-icons';
import React, { useEffect, useState } from 'react';
import {
  ActivityIndicator,
  Alert,
  ScrollView,
  StatusBar,
  Text,
  TextInput,
  TouchableOpacity,
  View
} from 'react-native';
import { useSafeAreaInsets } from 'react-native-safe-area-
context';
import { doc, DocumentData, getDoc } from
'firebase/firestore';
import { db } from '../firebaseConfig';
interface CalculationItem {
    label: string;
    value: string;}
interface LabReportData {
  labTitle: string;
  theme: string;
  goal: string;
  studentName: string;
  calculations: CalculationItem[]; }
type LabReviewScreenProps = {
  reportId: string; // ID документа звіту
  onNextStudent: () => void;
  onPrevStudent: () => void;
  onBack: () => void; // Функція для повернення
};
//Блок розрахунків
const CalculationBlock: React.FC<{ data:
LabReportData['calculations'] }> = ({ data }) => {
  if (!data || data.length === 0) {
      return <Text style={styles.noDataText}>Дані
розрахунків відсутні.</Text>;
  }
  // Розбиваємо дані на 4 колонки
  const columns = [[], [], [], []] as CalculationItem[][];
  data.forEach((item, index) => {
    columns[index % 4].push(item);
  });
  return (
    <View style={styles.calculationsContainer}>
      <Text
style={styles.calculationTitle}>Розрахунки</Text>
      <View style={styles.calculationsGrid}>
        {columns.map((column, colIndex) => (
```

```
                <View key={colIndex}
style={styles.calculationColumn}>
                  {column.map((item, itemIndex) => (
                    <View key={itemIndex}
style={styles.calculationItem}>
                      <Text
style={styles.calculationLabel}>{item.label}</Text>
                      <TextInput
                          style={styles.calculationInput}
                          value={item.value}
                          editable={false}
                          placeholder="—"/>
                    </View>
                  ))}
                </View>
              ))}
            </View>
          </View>);};
      const LabReviewScreen: React.FC<LabReviewScreenProps> = ({
          reportId,
          onNextStudent,
          onPrevStudent,
          onBack,
      }) => {
        const insets = useSafeAreaInsets();
        const [reportData, setReportData] =
useState<LabReportData | null>(null);
        const [isLoading, setIsLoading] = useState(true);

        useEffect(() => {
          const fetchReport = async () => {
            if (!reportId) {
              setIsLoading(false);
              return;
            }
            try {
              setIsLoading(true);
        //Створюємо посилання на документ
              const docRef = doc(db, 'lab_reports', reportId);
              // Отримуємо знімок документа
              const docSnap = await getDoc(docRef);
              if (docSnap.exists()) {
                const rawData = docSnap.data() as DocumentData;
                // Адаптація даних
                const labData: LabReportData = {
                  labTitle: rawData.labTitle || 'Лабораторна
робота',
                  theme: rawData.theme ||
rawData.lab.details.theme || 'Не вказано',
                  goal: rawData.goal || rawData.lab.details.goal
|| 'Не вказано',
                  studentName: rawData.studentName || 'Прізвище
Ім’я', // Потрібно вирішити, звідки береться ПІБ
```

```
            calculations: [
                        { label: 'Час (t, с)', value:
rawData.data?.t?.[0] || '—' },
                        { label: 'Довжина (l, м)', value:
rawData.data?.l?.[0] || '—' },
                        { label: 'Кількість (N)', value:
rawData.data?.N || '—' },
                        { label: 'Період (T, с)', value:
rawData.data?.T?.[0] || '—' },
                        { label: 'Прискорення (g, м/с²)', value:
rawData.data?.g?.[0] || '—' },
                        { label: 'Сер. g (м/с²)', value:
rawData.data?.g_avg || '—' },
                        { label: 'Похибка (ε, %)', value:
rawData.data?.epsilon || '—' },
                    ].filter(item => item.value !== '—' &&
item.value !== ''), // Фільтруємо порожні};
                  setReportData(labData);
                } else {
                  Alert.alert('Помилка', `Звіт з ID: ${reportId} не
знайдено.`);
                  setReportData(null);}
              } catch (error) {
                console.error('Помилка при завантаженні звіту з
Firestore:', error);
                Alert.alert('Помилка', 'Не вдалося завантажити дані
звіту.');
              } finally {
                setIsLoading(false);
              }
            };
            fetchReport();
          }, [reportId]);
          const labTitle = reportData?.labTitle || 'Лабораторна
робота';
          // AppBar
          const AppBar: React.FC<{ title: string }> = ({ title })
=> (
            <View style={[styles.appBar, { paddingTop: insets.top
}]}>
              <TouchableOpacity onPress={onBack}
activeOpacity={0.7}>
                <Ionicons name="chevron-back" size={24}
color="#000" />
              </TouchableOpacity>
              <Text style={styles.appBarTitle}>{title}</Text>
              <TouchableOpacity onPress={() =>
Alert.alert('Профіль', 'Налаштування профілю вчителя')}
style={styles.profileIcon} activeOpacity={0.7}>
                <Ionicons name="person-circle-outline" size={30}
color="#000" />
              </TouchableOpacity>
            </View>);
```

```
        if (isLoading) {
          return (
            <View style={styles.loadingContainer}>
              <StatusBar barStyle="dark-content" />
              <AppBar title={labTitle} />
              <ActivityIndicator size="large" color="#007AFF"
style={{marginTop: 50}} />
              <Text style={styles.loadingText}>Завантаження
звіту...</Text>
            </View>);}
        if (!reportData) {
          return (
            <View style={styles.loadingContainer}>
              <StatusBar barStyle="dark-content" />
              <AppBar title={labTitle} />
              <Text style={styles.errorText}>Дані відсутні або
звіт не знайдено.</Text>
            </View>);}
        return (
          <View style={styles.container}>
            <StatusBar barStyle="dark-content" />
            <AppBar title={labTitle} />

            <ScrollView
contentContainerStyle={styles.contentContainer}>
              {/* Тема та Мета */}
              <View style={styles.section}>
                  <Text style={styles.label}>Тема</Text>
                  <Text
style={styles.detailText}>{reportData.theme}</Text>
              </View>
              <View style={styles.section}>
                  <Text style={styles.label}>Мета</Text>
                  <Text
style={styles.detailText}>{reportData.goal}</Text>
              </View>
              {/* Виконував учень */}
              <View style={styles.studentInfoSection}>
                  <Text style={styles.studentLabel}>Виконував
учень:</Text>
                  <Text
style={styles.studentNameText}>{reportData.studentName}</Text>
              </View>
              {/* Розрахунки*/}
              <CalculationBlock data={reportData.calculations} />
              {/* Кнопки оцінювання */}
              <TouchableOpacity style={styles.buttonPrimary}
onPress={() => Alert.alert('Оцінити', 'Відкрити форму оцінки')}>
                <Text
style={styles.buttonTextPrimary}>Оцінити</Text>
              </TouchableOpacity>
```

```
              <TouchableOpacity style={styles.buttonSecondary}
onPress={() => Alert.alert('Повернути', 'Відправити на
доопрацювання')}>
                <Text
style={styles.buttonTextSecondary}>Повернути</Text>
              </TouchableOpacity>
              {/* Кнопки навігації */}
              <View style={styles.navigationRow}>
                <TouchableOpacity style={styles.buttonNav}
onPress={onPrevStudent}>
                  <Text style={styles.buttonTextNav}>Попередній
учень</Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.buttonNav}
onPress={onNextStudent}>
                  <Text style={styles.buttonTextNav}>Наступний
учень</Text>
                </TouchableOpacity>
              </View>

              {/* Інтерактивна анімація */}
              <View style={styles.interactiveBlock}>
                <Text style={styles.interactiveText}>Інтерактивна
анімація</Text>
              </View>
              {/* Кнопка для взаємодії */}
              <TouchableOpacity
style={styles.buttonSecondaryBottom} onPress={() =>
Alert.alert('Взаємодія', 'Дія інтерактивної анімації')}>
                <Text style={styles.buttonTextSecondary}>Кнопка
для взаємодії</Text>
              </TouchableOpacity>
            </ScrollView>
          </View>);};
```

ДОДАТОК Б

Диск з програмою

ДОДАТОК В

Заява студента про оригінальність роботи

Павленко Кирило Олегович
ПІБ студента

Номер залікової книжки: _____

Засвідчую, що кваліфікаційна робота на тему

Розробка програмного забезпечення комплексу віртуальних лабораторій з фізики: дослідження впливу інтерактивних симуляцій на якість навчання
зі спеціальності 121 – Інженерія програмного забезпечення
була підготовлена виключно мною і не порушує авторські права третіх осіб відповідно до закону про авторське право; повністю або частково не була використана як основа для отримання диплома про вищу освіту або науковий ступінь мною або іншою особою. Представлена мною для перевірки електронна версія роботи збігається з друкованим примірником.

Підтверджую, що був(ла) проінформований(на) про права та обов'язки здобувача вищої освіти Університету та правила, що стосуються перевірки оригінальності наукових робіт. Згоден(на) на обробку моєї письмової роботи й архівування цієї роботи в базі даних відповідно антиплагіатних правил і процедур Університету.

З Положенням про академічну доброчесність у Криворізькому національному університеті ознайомлений(на)

_____	_____	_____
(дата)	(підпис)	(ініціали та прізвище автора)