МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Віртуальна лабораторна робота з дослідження амплітуд та часових параметрів електричних сигналів різної частоти»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-12 Радченко Денис Ярославович

Кваліфікаційну роботу
захищено на засіданні ЕК
з оцінкою _____ «___» грудня 2022 р.

Науковий керівник _____ к.т.н., доц., Баранова І.В.
(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ІТ

_____ С. М. Ващенко
«___» _____ 2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Радченко Денис Ярославович
(прізвище, ім'я, по батькові)

1 Тема проекту *Віртуальна лабораторна робота з дослідження амплітуд та часових параметрів електричних сигналів різної частоти*

затверджена наказом по університету від «_04_» _11__2022 р. № 01013-VI

2 Термін здачі студентом закінченого проекту «___» ___грудня___ 2022 р.

3 Вхідні дані до проекту технічне завдання на розробку додатку " Віртуальна лабораторна робота з дослідження амплітуд та часових параметрів електричних сигналів різної частоти "

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз програмних продуктів – аналогів, визначення технічних вимог додатку та засоби реалізації, створення моделей приладу та сцени, програмна реалізація симуляції сигналу, розробка клієнт-серверного взаємозв'язку, створення інтерфейсу для користувача

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, постановка задачі, аналіз програмних продуктів – аналогів, IDEF0-діаграма та її декомпозиція, діаграма варіантів використання, демонстрація додатку

. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____27.08.2022_____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз предметної області	01.08.2022 – 19.08.2022	
2	Планування ІТ-проекту	22.08.2022 – 26.08.2022	
3	Проектування робіт	29.08.2022 – 05.09.2022	
4	Створення 3D моделей	01.09.2022 – 16.09.2022	
5	Розробка симуляції осцилографу	19.09.2022 – 30.09.2022	
6	Розробка головного меню	03.10.2022 – 28.10.2022	
7	Розробка клієнт-серверної архітектури	31.10.2022 – 11.11.2022	
8	Тестування додатку	14.11.2022 – 18.11.2022	
9	Здача роботи	21.11.2022 – 11.12.2022	

Магістрант _____ Радченко Д.Я.

Керівник роботи _____ к.т.н., доц. Баранова І.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Віртуальна лабораторна робота з дослідженням амплітуд та часових параметрів електричних сигналів різної частоти на осцилографі».

Пояснювальна записка складається зі вступу, 4 розділів, висновка, списку використаних джерел із 37 найменувань, додатків. Загальний обсяг роботи – 134 сторінок, у тому числі 65 сторінок основного тексту, 5 сторінок списку використаних джерел, 56 сторінок додатку.

Кваліфікаційну роботу магістра присвячено розробці додатку для роботи з осцилографом та виконання лабораторних робіт з використанням клієнт-серверного взаємозв'язку.

В роботі проведено аналіз стану питання віртуальних лабораторних робіт, сформульовано мету та задачі проекту, обрано засоби реалізації.

Проведені етапи планування та проектування виконуваного проекту. Розроблено календарний графік виконуваних робіт та визначені ризики проекту. Розроблені діаграми у нотації IDEF0 і Use Case діаграму.

Виконано моделювання об'єктів сцен та імпорт до ігрового рушія, розробка програмної симуляції сигналів. Проведено налаштування сцен та розроблено клієнт-серверний взаємозв'язок додатку та його меню.

Результатом проведеної роботи є додаток для симуляції роботи з осцилографом, а також виконання лабораторних робіт на осцилографі.

Практичне значення роботи – створена віртуальна лабораторна робота може бути використана в навчальному процесі електро-технічних спеціальностей.

Ключові слова: VR-технологія, осцилограф, 3D модель, віртуальна лабораторна робота, електричний сигнал, симуляція, клієнт-сервер, Unity, Blender.

ЗМІСТ

ВСТУП

В сучасному світі, де майже всі люди мають телефон або комп’ютер, важливою є тема розвитку технологій віртуальної реальності (Virtual Reality – віртуальна реальність). VR-технології – це технології, які занурюють людину у комп'ютерно згенероване середовище з різними об'єктами, що впливають на органи чуттів та здаються реальними.

Хоча віртуальна реальність існує вже давно, вона наразі набирає популярність в сфері навчання. Наприклад, віртуальні симулятори в медичній галузі занурюють студентів у штучні, але справжні за відчуттями обставини, що робить їх доцільними для вивчення студентами різних сценаріїв, з якими вони можуть зіткнутися у своїй майбутній роботі. Звичайно, це не замінить реальні враження, але це великий розвиток у VR-технологіях.

Зараз VR-технології більше асоціюються з VR-окулярами, що дозволяють занурити людину майже в інший світ, але це просто одне з гарних рішень щодо покращення відчуття реальності середовища.

На початку розвитку VR-технології використовувались в основному для відеоігор, так як сама технологія була розвинута недостатньо для чогось більшого, а у відеоіграх цінувалась. Чим далі, тим віртуальні технології покращувалися, та їх використання розширялося, їх почали використовувати в фільмах, маркетингу, науці тощо (в науці і до цього використовувалося, але не так поширено).

Віртуальні технології можна використовувати не тільки для візуалізації реального світу, а й для симуляції різних явищ, таких наприклад як: землетруси та аналіз їх наслідків в окремих регіонах; проведення небезпечних вибухових або хімічних реакцій, симуляції електричного поля, стійкості різних конструкцій, проектування будівель і приладів тощо.

Особливо можна виділити симуляції фізичних явищ – в основному всі дослідження, наприклад з фізики, виконуються з використанням реальних

предметів або приладів, що були створені спеціально для цього. Проте не завжди у когось знайдеться під рукою транзистор чи осцилограф, тому і є різні заклади освіти, де можна їх знайти та навчитись з ними працювати. Якщо такої змоги немає – на допомогу приходять VR-технології.

Можна створити комп'ютерну копію даних приладів, огорнути це все в зручний та привабливий інтерфейс з корисними функціями для користувача, та надати доступ до них людям, хто хотів навчитися працювати з якимось приладом, але не мали можливості.

Тому розробка подібного застосунку наразі є актуальною. З його допомогою можна буде просто на своєму комп'ютері запустити додаток та без усякого ризику розвиватися в даному напрямі.

Таким чином, можна сформулювати мету та задачі роботи.

Метою роботи є створення додатку для симуляції роботи з осцилографом та виконання на ньому лабораторних робіт, які потім можуть бути зараховані викладачем з відповідної дисципліни.

Об'єктом дослідження є симуляція роботи з осцилографом, методом виконання лабораторних робіт по роботі з ним.

Предметом є віртуальна лабораторна робота, яка надає змогу користувачу виконувати лабораторні роботи на віртуальних приладах, максимально приближених до реальних, що потім використати ці навички в житті.

У ході реалізації даного проекту будуть вирішені такі задачі:

- аналіз предметної області та схожих рішень;
- визначення технічних аспектів створення додатку;
- створення необхідних моделей приладу та навколишньої сцени;
- програмна реалізація симуляції сигналу, що надходить до осцилографа;
- розробка клієнт-серверного взаємозв'язку;
- створення інтерфейсу для користувача.

Надалі додаток може бути інтегрований в навчальну програму з фахових предметів, як альтернатива виконанню лабораторних робіт для тих, хто не має можливості прийти в аудиторію та працювати з реальним осцилографом.

Практичне значення. Розроблений додаток дозволить користувачам працювати з моделлю осцилографа та вхідними сигналами, які він сам налаштовує, а також виконати лабораторні роботи по дослідженню характеристик різних видів сигналу.

1 АНАЛІЗ ПИТАННЯ РОЗРОБКИ ВІРТУАЛЬНИХ ЛАБОРАТОРНИХ РОБІТ

1.1 Огляд існуючого стану питання розробки віртуальних робіт

Представлених в інтернеті віртуальних лабораторних робіт не так і багато, в основному це програми, створені приблизно 10-15 років тому. Вони мають дуже простий, подекуди незрозумілий інтерфейс для користувача, який вперше запустив цей додаток, або просто не знайомий з направленням додатку.

В основному подібні додатки були створені науковцями, яким потрібен матеріал для аналізу, і це відображається в схематичності візуалізації рішень [1-3]. Наприклад, в додатках по симуляції електричних кіл [4], є робоча область, куди додаються електричні елементи в схематичному вигляді та їх параметри в окремих віконцях.

Такі додатки дають змогу побачити результат роботи електричного кола та проаналізувати його, проте ці програми не дають можливість користувачу повністю познайомитися с елементами кола, з їх повним налаштуванням, а тільки схематично, що не дозволить в реальному житті працювати з такими елементами в повному обсязі.

Щодо саме віртуальних додатків лабораторних робіт, то сучасні такі додатки розробляються у веб-вигляді, так як зараз дуже популярний напрямок веб-розробок [5-7]. В цілому, лабораторні веб-додатки мають в собі текстовий інтерфейс, де описані певні теоретичні відомості щодо устрою чи роботи приладу, інколи до лабораторної роботи додаються тести, – тобто вони не мають функціональної частини з симуляцією приладу [8]. Але є такі, що мають тільки симуляцію роботи якогось приладу, електричних кіл, але вже без самих лабораторних робіт з поясненнями та/або тестами. Щодо функціональної частини з симуляцією роботи приладу, то зазвичай, створюється простий інтерфейс з повзунками та перемикачами, що не відображають повною мірою

реальний прилад та його робочий простір, тобто симуляція приладу майже завжди наближена до реальності але як працювати з ним в реальному світі після додатку, незрозуміло.

Щодо саме віртуальних додатків лабораторних робіт, нинішні такі додатки розробляються у веб-вигляді, так як зараз дуже популярний напрямок веб розробок. Зазвичай лабораторні веб-додатки мають в собі текстовий інтерфейс, де описані певні теоретичні відомості щодо приладу, інколи додаються тести до лабораторної роботи, а в цілому вони не мають функціональної частини з симуляцією приладу. Трапляються й інші веб-додатки, що мають тільки симуляцію роботи якогось приладу, електричних кіл тощо, але вже без самих лабораторних робіт з поясненнями та/або тестами. Для симуляції роботи приладу зазвичай створюється простий інтерфейс з повзунками та перемикачами, який не відображає повною мірою реальний прилад та його робочий простір. Тобто функціонально симуляція приладу майже завжди наближена до реальності, але як працювати з ним в реальному світі після додатку – незрозуміло.

Стосовно десктоп додатків – вони в цілому такі ж, як і веб-додатки. Мають простий інтерфейс, де приблизно 15% виокремлено на інтерфейс, меню з базовими поясненнями функціоналу, налаштуваннями, меню приладу або меню вибору елементів, а інші 85% – це вікно з приладом або з полем для збору електричних кіл [9].

1.2 Дослідження предметної області моделювання електричних сигналів

Так як осцилограф аналізує електричні сигнали, що подаються на вхід, то потрібно у моделі реалізувати саме їх.

Електричний сигнал як матеріальний носій інформації, являє собою певну змінну фізичну величину (напругу, струм, заряд, магнітний потік), яку називають коливанням.

Розрізняють дві основні групи сигналів: детерміновані та випадкові

(стохастичні). Детермінованими сигналами називають сигнали, значення яких у будь-який момент часу є точно відоме. Випадковими називають такі сигнали, значення яких у будь-який момент часу неможливо передбачити абсолютно точно [10].

Крім того, сигнали розрізняються зміною значень в часі та на множині значень, тобто на неперервні та дискретні. На підставі цього, сигнали можна поділити на чотири типи сигналів (рис. 1.1):

– аналогові – неперервні в часі та на множині значень;
– дискретизовані – дискретні в часі неперервні на множині значень;
– квантовані – неперервні в часі та дискретні на множині значень;
– цифрові – дискретні одночасно в часі та на множині значень.

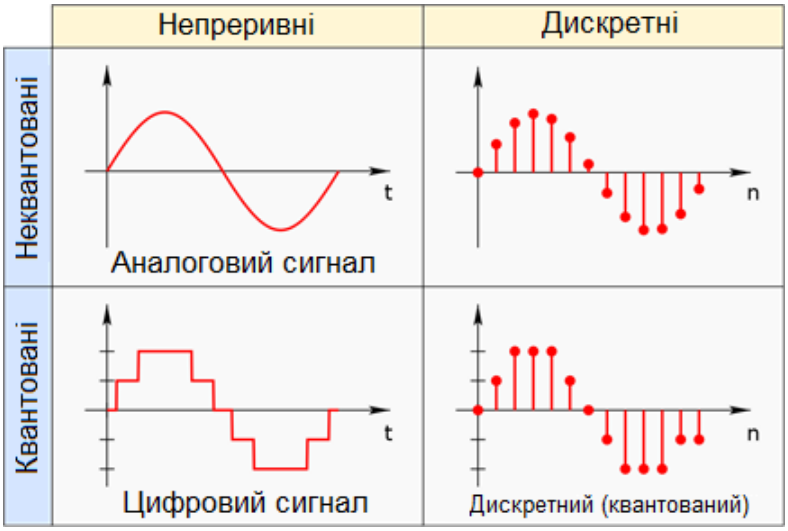

Рисунок 1.1 – Аналоговий, цифровий та дискретні сигнали

Всі реальні фізичні сигнали є дійсними функціями часу, але залежно від потреб методів аналізу найчастіше використовують такі способи їх математичного опису [11]:

– опис сигналу як функції часу - часовий опис;
– опис сигналу як деякої функції частоти - частотний (спектральний) опис;
– опис сигналу в операторній формі - операторний опис.

У багатьох випадках для спрощення аналізу проходження сигналів через електронні кола доцільно описати сигнал як сукупність вибраних певним чином

елементарних (найпростіших) сигналів. До елементарних сигналів належать: гармонічне коливання, одиничний стрибок (функція Хевісайда), дельта-імпульс (функція Дірака). Гармонічне коливання використовують при частотному (спектральному) описі сигналів, а одиничний стрибок та дельта-імпульс - при часовому описі, який інколи називають динамічним описом.

Так як в проєкті використовується модель осцилографа С1-83, що працює з аналоговими сигналами, то математичне представлення аналогового сигналу буде використовуватися для відображення сигнала користувачу у додатку.

Легше всього описати сигнал за допомогою частотного опису. Для частотного опису сигналу, використовують гармонічний сигнал, так як він є елементарним сигналом, що задається виразом:

$$\boldsymbol{s(t) = A_m \cos(wt + \varphi) = A_m \cos y(t)}, \quad (1)$$

де A_m – амплітуда (максимальне відхилення від нульового значення), розмірність якої збігається з розмірністю сигналу;

w – кутова швидкість, яку вимірюють у радіанах за секунду;

φ – початкова фаза, яку вимірюють в кутових одиницях (радіанах або градусах).

Аргумент гармонічного сигналу $y(t) = wt + j$ називають повною фазою.

Гармонічний сигнал належить до неперервних у часі сигналів, які теоретично існують на необмеженому часовому інтервалі, і задовольняють умову періодичності, тобто повторення миттєвих значень через певний проміжок часу, який називають періодом:

$$\boldsymbol{s(t) = s(t \pm nT)}, \quad (2)$$

де T – період сигналу, n – довільне ціле число.

Оскільки за період відбувається зміна повної фази на 2π радіанів, то звідси випливає відоме співвідношення:

$$\boldsymbol{T = \frac{2\pi}{w} = \frac{1}{f}}, \tag{3}$$

де f – частота коливань, яку вимірюють у герцах [Гц], і яка пов'язана з кутовою швидкістю w співвідношенням: $w = \frac{2\pi}{f}$.

Виходячи з цих параметрів, гармонічний сигнал можна подати у вигляді суми елементарних гармонічних коливань. Так як у нас сигнал буде незмінний протягом всього часу, то складову суми можна прибрати та описати за допомогою наступного співвідношення:

$$\boldsymbol{s(t) = A * \cos\left(\frac{2\pi}{f} t + \varphi\right)}, \tag{4}$$

де A – амплітуда, f – частота, φ – початкова фаза.

Формула 4 описує тільки сигнал, що подається на вхід до осцилографа, а сам осцилограф також має змогу змінювати його на константні значення, щоб потім вирахувати вхідні параметри сигналу. Тому потрібно до формули 4 додати додаткові параметри, після чого отримано співвідношення:

$$\boldsymbol{y(t) = ((V_s * A) * V_{ss}) * \cos\left(\frac{2\pi}{(f * t_S) * t_{ss}} x(t) + X\right) + Y}, \tag{5}$$

де V_s – показник дільника напруги $\frac{V}{\text{діл}}$;

V_{ss} – множник напруги;

t_s - показник дільника часу $\frac{t}{\text{діл}}$;

t_{ss} - множник напруги;

X,Y – положення сигналу на осі абсцис та ординат відповідно.

1.3 Дослідження існуючих рішень

Якщо аналізувати представлені аналоги з віртуальними роботами, то можна знайти декілька веб-додатків, які виконують симуляцію роботи осцилографа, наприклад, один із них – Virtual-oscilloscope з сайту Academo [12] (рис. 1.2).

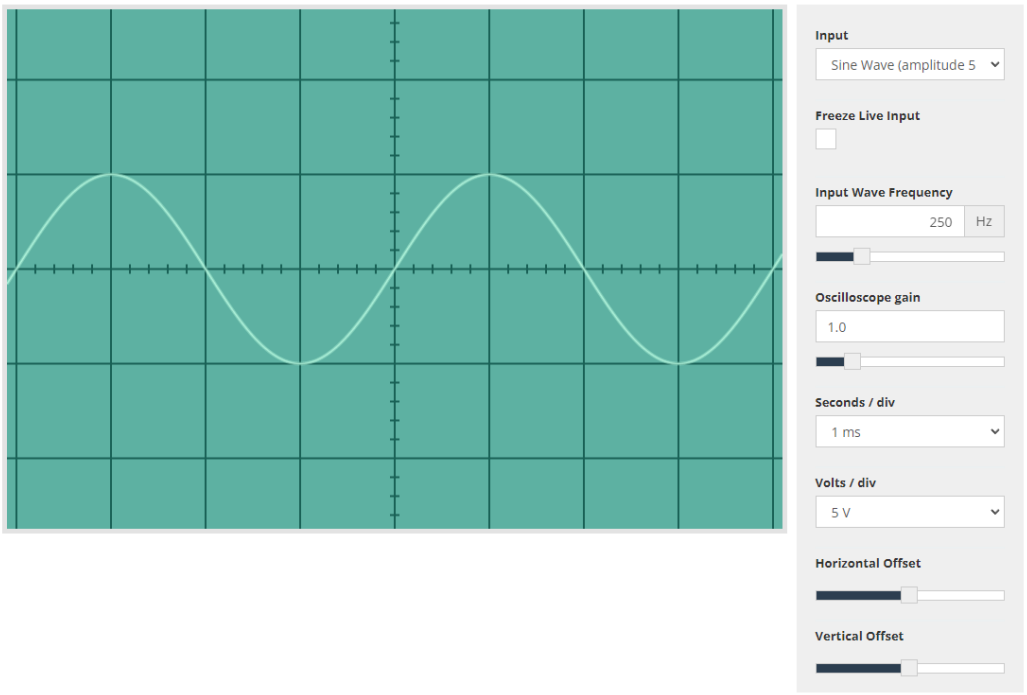

Рисунок 1.2 – Вигляд віртуального осцилографу з сайту Academo

Меню віртуального осцилографа дуже легке, але не дає уявлень про роботу зі справжнім осцилографом. Для прикладу на рисунку 1.3 показано осцилограф С1-83, який використовується в лабораторних кабінетах СумДУ.

Рисунок 1.3 – Вигляд осцилографа С1-83

Якщо порівняти інструменти налаштувань віртуального та реального осцилографа, то одразу не збагнути що й до чого – тільки після маніпуляцій з повзунками у віртуальному приладі стає зрозуміло, за що вони відповідають, та знайти їх на реальному осцилографі буде складно.

Також недолік даної реалізації – на вхід можна подати тільки синусоїдальний або імпульсний сигнал, немає додаткового каналу, на який можна подати ще один сигнал, щоб порівнювати їх, проводити додавання/віднімання сигналів або створювати сигнал за формою фігур Ліссажу (рис. 1.4) – замкнуті траєкторії, які прокреслюються точкою, що здійснює одночасно два гармонійних коливання у двох взаємно перпендикулярних напрямках [10].

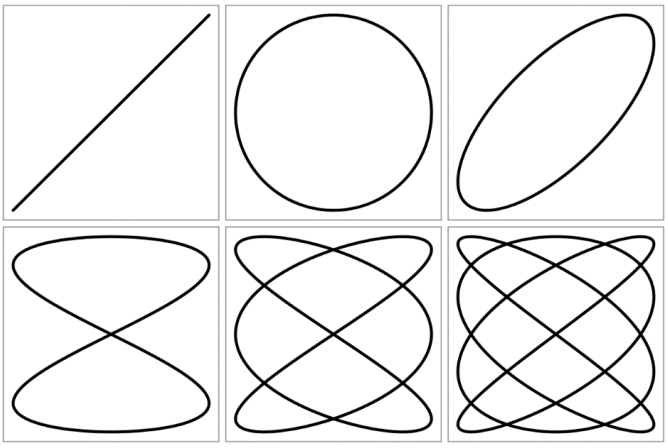

Рисунок 1.4 – Фігури Ліссажу

Інша реалізація віртуального осцилографа представлена на сайті physics-zone [13], де інтерфейс набагато краще і більше нагадує реальний (рис. 1.5). В інтерфейсі велика кількість різних перемикачів і спочатку складно їх зрозуміти. Для полегшення зверху є кнопка “How-To”, після натискання якої вмикається інструкція до кожної кнопки, як приклад на рисунку 1.6. На відміну від попереднього, він вже має два канали для вхідного сигналу та представлення двох генераторів сигналу – змінного та постійного.

Рисунок 1.5 – Віртуальний осцилограф з сайту physics-zone

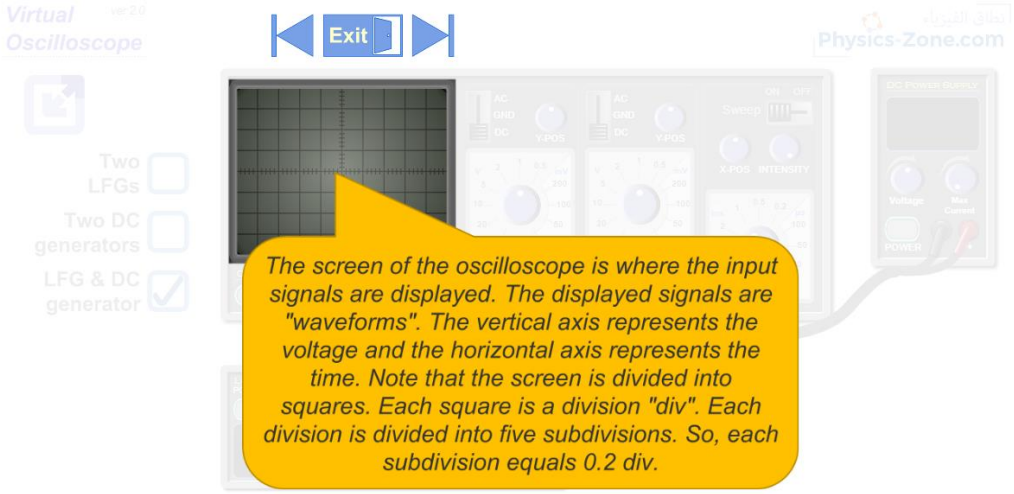

Рисунок 1.6 – Підказки по роботі з віртуальним осцилографом.

Щодо симуляції вхідного сигналу важко щось прокоментувати, так як коду реалізації не представлено в даних прикладах, скоріше за все, дані синусоїди мають звичайне математичне представлення.

Для аналізу порівняємо функціональні можливості розглянутих аналогів та власної розробки. Дані аналізу наведені у таблиці 1.1.

Таблиця 1.1 – Порівняння розроблюваного додатку та проаналізованих

Критерії \ Додаток		Розроблюваний додаток	Academo	Physics-zone
Інтерфейс	Зручність	+	+	-
	Простота	+	+	-
	Зрозумілість	+	+	-
Інструкція використання		+	-	+
Близкість робочої поверхні приладу до реального		+	-	+
Наявність теоретичних відомостей		+	-	+
Можливість виконання лабораторних робіт		+	-	-
Клієнт-серверна взаємодія (реєстрація, авторизація, зберігання даних/результатів)		+	-	-
Можливість працювати без інтернету		+	-	-

Як свідчать дані порівняльної таблиці 1.1, розроблюваний додаток для виконання лабораторних завдань по роботі з осцилографом враховуватиме

зазначені недоліки проаналізованих схожих рішень, а також матиме більш розширений функціонал, що надасть змогу користувачу вивчити прилад та роботу з ним більш досконало. Так як це десктопний додаток, то користувач зможе з ним працювати тільки після встановлення на комп'ютер, на відміну від веб-версій, які можна відкрити де завгодно в браузері. Проте дана властивість скоріше є перевагою в нинішніх умовах, бо даний додаток зможе працювати і без інтернету, а саме частина по роботі з самим осцилографом, теоретичні відомості до приладу та роботи з ним та лабораторні роботи.

1.4 Аналіз підходів до реалізації додатків

При виборі технології реалізації віртуальної лабораторної роботи потрібно розуміти, який в нього буде функціонал. По-перше, це сама робота з приладом, максимально наближеним до реального прототипу, тобто ідентична 3D модель приладу та повний функціонал його робочої панелі. Для цього підійдуть технологій з десктоп та мобільним додатками, так як для них вже створені рішення щодо 3D візуалізації та навігацій по типу ігор, 3D редакторів. Веб варіант наразі не розглядається, так як в ньому немає загального рішення по 3D візуалізації, а також тих ресурсів для обчислень, які мають комп’ютер або телефон.

Між вибором стаціонарного чи мобільного додатку потрібно враховувати клієнт-серверну взаємодію та кількість цих взаємодій. Оскільки додаток крім симуляції осцилографу буде мати меню, де користувач зможе зареєструватися/авторизуватися, обрати завдання лабораторної роботи та виконати її, продивитися результати тощо, можна зробити висновок, що меню користувача буде доволі громіздким по наявній інформації. Також потрібно врахувати, що меню не повинно перекривати сам прилад, на якому йде робота і тому маленького екрану телефону буде недостатньо для якісної роботи даного додатку, відповідно залишається тільки варіант з розробки десктоп додатку.

Так як додаток лабораторних робіт, що розглядається в цій роботі, спрямований на роботу користувача з інформацією, то користувач в клієнтській частині вводить деяку інформацію, або зберігає результат своєї роботи, і потім ця інформація передається на сервер, оброблюється та зберігається. Для цього додаток повинен реалізовувати клієнт-серверну архітектуру, що дасть змогу студентам працювати незалежно та розподілено, прямо в додатку виконувати лабораторну роботу від свого імені, а пізніше викладач зможе продивитись результати робіт студентів. Тому потрібно створити клієнт-серверні взаємовідносини, так щоб вирішити ці всі питання [14-15].

Клієнт-серверні додатки існують давно, відповідно вже було створено багато рішень. Для уникнення проблем втручання в серверні скрипти [16], виклику їх не з додатку, який тільки і повинен звертатися до них, до HTTP запиту, що відправляється на сервер з інформацією, можна додати складений пароль, який буде перевірятися на сервері. Це може бути дата, генерація випадкового ключа на основі одного ключа генерації, який теж можна генерувати кожен день новий.

Для запобігання втручання в запити баз даних [17-18], так званий Script Injection, кожне поле, куди користувач вводить інформацію, потрібно оброблювати на вірність введених даних в самому додатку. Тобто – якщо поле тільки для чисел – туди вводяться тільки числа. Якщо поля, де можна вводити різні символи (наприклад поля для пошти), – цю проблему можна вирішити через використання regular expression, що дозволяють перевіряти послідовність введених даних та звіряти їх з патерном того, що повинно там бути введено.

Але навіть зазначених перевірок іноді не вистачає. Тому на сервері вводиться додаткова перевірка інформації, що надходить, а саме перетворення всіх спеціальних символів типу “‘.,-_<>;:()”, які використовуються у формуванні SQL-запитів, та перекодування їх в код ASCII, тобто в запиті замість символу “–“ буде “&#8722”, при повторному запиті на інформацію, якщо не встановлювати перетворювачів інформації, воно буде відображено користувачу як “–”.

Щодо клієнт-серверної частини, то проаналізовані аналоги звісно ж її не мають (якщо виключити те, що вони розміщені на сайті), так як не орієнтовані на зберігання чи обробку якихось даних. Як і багато інших веб-додатків, майже всі вони були однакові, максимум відрізнялися самим виглядом та кількістю функціоналу в меню.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі дослідження

Метою роботи є створення унікального додатку для проведення віртуальної лабораторної роботи з моделлю осцилографа по дослідженню електричних сигналів різної частоти.

Ціль створення додатку – вдосконалення навичок роботи користувача з даним приладом і подальше застосування цих навичок в житті. Також можна додати полегшення доступу до виконання лабораторних робіт з осцилографом для користувачів/студентів, які не мають змогу з ним попрацювати віч-на-віч.

Для досягнення поставленої мети в даному додатку потрібно реалізувати передбачений функціонал, зокрема:

- зручний та зрозумілий інтерфейс з інструментами керування, які максимально наближені до реального приладу;
- зручне меню, яке не закриває вікно моделі;
- наявність інформації для виконання лабораторної роботи;
- наявність функції авторизації/реєстрації;
- наявність вибору завдань лабораторної роботи;
- збереження даних виконання роботи в клієнтській частині;
- надсилання інформації з даними авторизації користувача та результатів виконаних робіт на сервер;
- розподіл прав користувачів додатку;
- можливість користувача з правами викладача змінювати параметри лабораторних робіт (час, оцінка).

Цільовою аудиторією даного додатку є студенти університету електротехнічних спеціальностей, які вивчають професійну дисципліну «Фізика процесів електричних кіл» та виконують лабораторні роботи з осцилографом, і

викладачі відповідних дисциплін, які мають можливість перевірити та зарахувати виконані лабораторні роботи по окремій дисципліні.

2.2 Технології та інструменти реалізації

В результаті аналізу підходів до реалізації подібних програмних продуктів було обрано клієнт-серверну архітектуру додатка. Клієнтська частина буде реалізована у вигляді десктоп додатку з 3D сценою, приладом та меню користувача, серверна частина буде складатися з контейнеру, де знаходяться два елементи: база даних та сервер з скриптами доступу (до якого буде звертатися користувач щоб надати/отримати інформацію з БД).

Для вибору можливої реалізації у додатку було розглянуто найбільш відомі сервери, серед яких:

- Apache/Apache TomCat;
- GlassFish;
- JBoss EAP;
- WebLogic.

Одним з найуживаніших для домашнього використання є Apache/Apache TomCat [19], тому для реалізації проєкту було обрано саме його. Хоча WebLogic має більшу пропускну здатність, але знову ж його складніше встановити, налаштувати та підняти.

В цілому клієнт-серверну частину було проаналізовано на прикладах різних клієнт-серверних додатків, тому що розроблюваний додаток віртуальної лабораторної роботи повинен зберігати дані ідентифікації користувача та результати виконання робіт, а саме: оцінку, дату виконання та номер завдання. Потрібно прив'язати ці дані до унікальних параметрів користувача, тобто:

- ім'я та прізвище;
- група (так як даний додаток орієнтований на студентів);

– пошта;

– пароль.

Всі ці характеристики в цілому дають ідеальний ідентифікатор з дуже малою колізією (вірогідність створення ідентичних ідентифікаторів). Тому для реєстрації та пошуку даних, було обрано саме ці параметри, для авторизації тільки пошта та пароль.

Для розміщення бази даних користувачів найліпше підходять бази даних SQL. Серед багатьох БД SQL найчастіше використовують Oracle SQL, MySQL та PostgreSQL.

Краще всього та легше встановлюється саме СУБД MySQL. Хоча Oracle SQL нічим не уступає MySQL, але її складно встановити та підняти на сервер. Тому обрано для реалізації СУБД MySQL [20-21].

Для написання серверних скриптів використовують декілька мов програмування, й усі вони тою чи іншою мірою не поступаються одне одному. Серед подібних мов зазначимо:

– PHP;

– JavaScript;

– Python;

– Ruby;

– Java;

– Golang;

– C#.

Найпопулярнішими являються JavaScript, PHP та Python. Для розробки серверної частини (скрипти, що будуть пов’язувати додаток с БД) було обрано PHP [22], оскільки мається більший досвід його застосування.

Контейнер, в якому буде автоматично запускатися та налаштовуватися сервер та БД, може бути представлено в різних виглядах – через Docker [23], WAMP [24] або ж MAMP [25]. У випадку з докером, потрібно самому спочатку налаштувати БД та сервер, а вже потім встановлювати їх в докер, де вони будуть автоматично разом запускатися.

У випадку WAMP та MAMP все вже налаштовано, потрібно тільки обрати вільні порти для БД та серверу. З цих двох варіантів було обрано MAMP, так як він має в собі СУБД MySQL та сервер Apache зі спільним інтерфейсом phpMyAdmin, де можна дуже легко налаштовувати БД, створювати таблиці та редагувати їх.

Для створення 3D моделі приладу потрібна платформа для реалізації. Серед можливих варіантів розглядалися ігрові рушії Unity та Unreal Engine 4/5. Unreal Engine (UE) [26] в цілому дуже гарний ігровий рушій, що розробляється з початку 2000-х та пройшов дуже велику кількість доопрацювань та покращень. Компанія-розробник Epic Games має проривні розробки в 3D візуалізації, які доступні в UE безкоштовно. Проте з року в рік розробники додають нові функції, тому даний рушій дуже громіздкий та складний у вивченні. Якщо його використовувати лише для графічного представлення якихось 3D робіт, то він ідеальний і швидкий, але створення додатків з різними унікальними фічами реально хіба що в компанії професіоналів.

Окремо хочеться виділити програмування в UE. Хоча автори створили візуальний скриптинг BluePrint [27], що полегшує програмування і дає можливість не вчити мови програмування, але на весь функціонал недостатньо документації, тож потрібно довго розбирати, як що працює.

Щодо Unity [28] – він набагато легший у вивченні, є документація та відео як по самому редактору, так і по програмній частині. Але Unity не має такого реалістичного рендеру та його докладного налаштування. Саме тому інді-розробники (маленькі команди до 10 чоловік) обирають Unity для свої розробок, а коли набираються досвіду переходять на UE.

Так як додаток орієнтований на функціональну частину, а саме симуляція осцилографа та виконання робіт на ньому, нам достатньо звичайної реалізації рендеру, також потрібен зручний та зрозумілий програмний фунціонал, тому було обрано для розробки саме Unity.

Для створення 3D моделей сцени та осцилографу було обрано такий конвеєр реалізації:

– Blender [29-30] – для моделювання, рендерингу;

– Marmoset Tool Bag [31-32] – для запікання моделі з високо полігональної у низькополігональну;

– Substance Painter/Designer [33-34] – для текстуринга та створення текстур.

Такий набір було обрано, так як мається великий досвід зі створення 3D моделей саме по цьому набору.

3 ПРОЄКТУВАННЯ ДОДАТКУ ВІРТУАЛЬНОЇ ЛАБОРАТОРНОЇ РОБОТИ

3.1 Структурно-функціональне моделювання процесу роботи з додатком

Краще всього для опису структурно-функціональних процесів розроблюваного додатку підходить методологія SADT [35], яка надає повний, точний та адекватний опис системи. Дана методологія містить додаткові нотатки у вигляді діаграм опису процесів IDEF0 та IDEF3 [36], що описують структурно-функціональні процеси зі створення додатку.

SADT (Structured Analysis and Design Technique) – методологія аналізу та проектування систем. Модель SADT представляє собою серію діаграм з супровідною документацією, що розбивають складний об’єкт на складові, які представлені у вигляді блоків. Деталі кожного з основних блоків показані у вигляді блоків інших діаграм. Кожна детальна діаграма є декомпозицією блоку більш загальної діаграми.

Побудова SADT-моделі починається з представлення всієї системи у вигляді контекстної діаграми IDEF0, що складається з одного блоку та дуг, які показують інтерфейси з зовнішніми функціями системи. IDEF0 діаграма має найвищій рівень абстракції A0.

Діаграма містить функціональний блок “Дослідження параметрів сигналу на віртуальному осцилографі”, який обробляє вхідну інформацію та видає результат.

Кожна з сторін функціонального блоку має своє значення (роль):

– Верхня сторона має значення “Управління”: методи обчислення параметрів сигналу, математична модель сигналу;

– Ліва сторона має значення “Вхідні дані”: завдання лабораторної роботи;

– Права сторона має значення “Вихідні дані”: звіт з лабораторної роботи,

оцінка за роботу;

– Нижня сторона має значення “Механізми”: користувач, технічне забезпечення, програмне забезпечення.

За точку зору було взято користувача додатку, а також була визначена мета – виконання завдань лабораторної роботи за допомогою розробленого додатку.

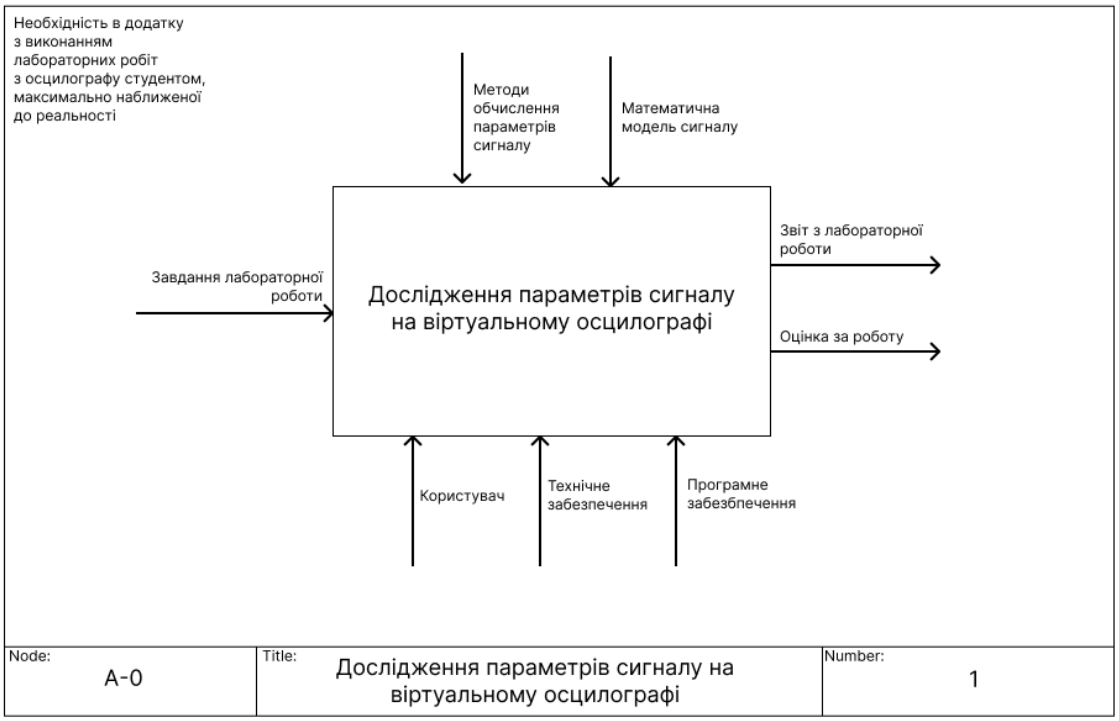

Рисунок 3.1 – Контекстна діаграма у нотації IDEF0

Після контекстного опису системи в цілому, потрібно провести розбиття її на складові для конкретизації процесів. В SADT-моделі це називається функціональною декомпозицією на діаграми, які описують кожен фрагмент і взаємодію фрагментів. Декомпозиція виконується за методологією послідовності процесів (IDEF0) та лінійності/паралельності робіт (IDEF3).

При декомпозиції контекстної діаграми було виділено такі головні блоки: “Авторизація/реєстрація”, “Вибір лабораторної роботи”, “Виконання лабораторної роботи”, “Розрахунок результатів”. Були визначені потоки даних між ними, а саме:

– “Дані користувача” між блоками “Авторизація/реєстрація” та “Вибір

лабораторної роботи”;

– “Завдання до лабораторної роботи” між блоками “Вибір лабораторної роботи” та “Виконання лабораторної роботи”;

– “Виконана лабораторна робота” між блоками “Виконання лабораторної роботи” та “Розрахунок результатів”.

Діаграма декомпозиції першого рівня наведена на рисунку 3.2.

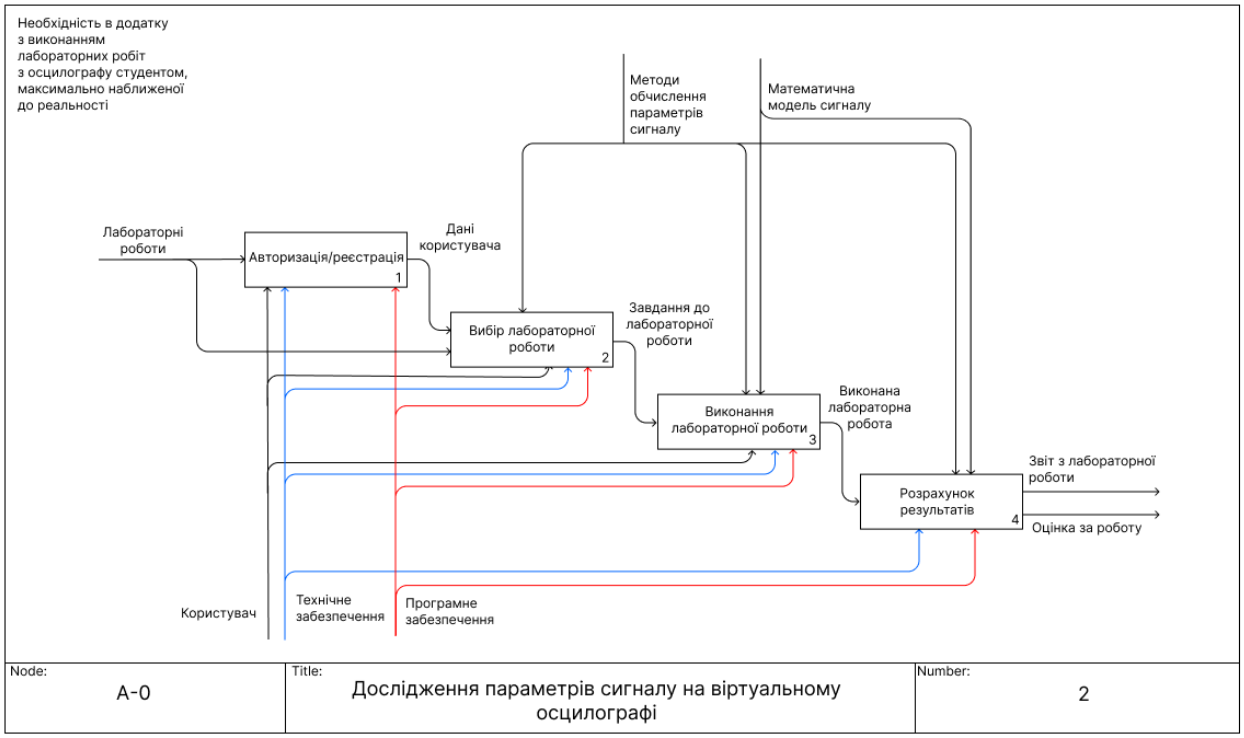

Рисунок 3.2 – Перший рівень декомпозиції IDEF0

За методологією IDEF0 була проведена декомпозиція другого рівня блоків “Авторизація/реєстрація”, “Вибір лабораторної роботи”, “Виконання лабораторної роботи”, “Розрахунок результатів”. Для першого блоку були виділені блоки: “Головне меню”, “Реєстрація”, “Перевірка реєстраційних даних”, “Авторизація”, “Перевірка даних авторизації”. Між ними встановлені наступні потоки даних:

– “Інтерфейс” між блоками “Головне меню”, “Реєстрація” і “Авторизація”;

– “Дані реєстрації” між блоками “Реєстрація” та “Перевірка реєстраційних

даних”;

– “Дані авторизації” між блоками “ Перевірка реєстраційних даних” та “Авторизація”;

– “Дані авторизації” між блоками “Авторизація” та “Перевірка даних авторизації”.

Діаграма декомпозиції другого рівня для блоку “Авторизація/реєстрація” наведена на рисунку 3.3.

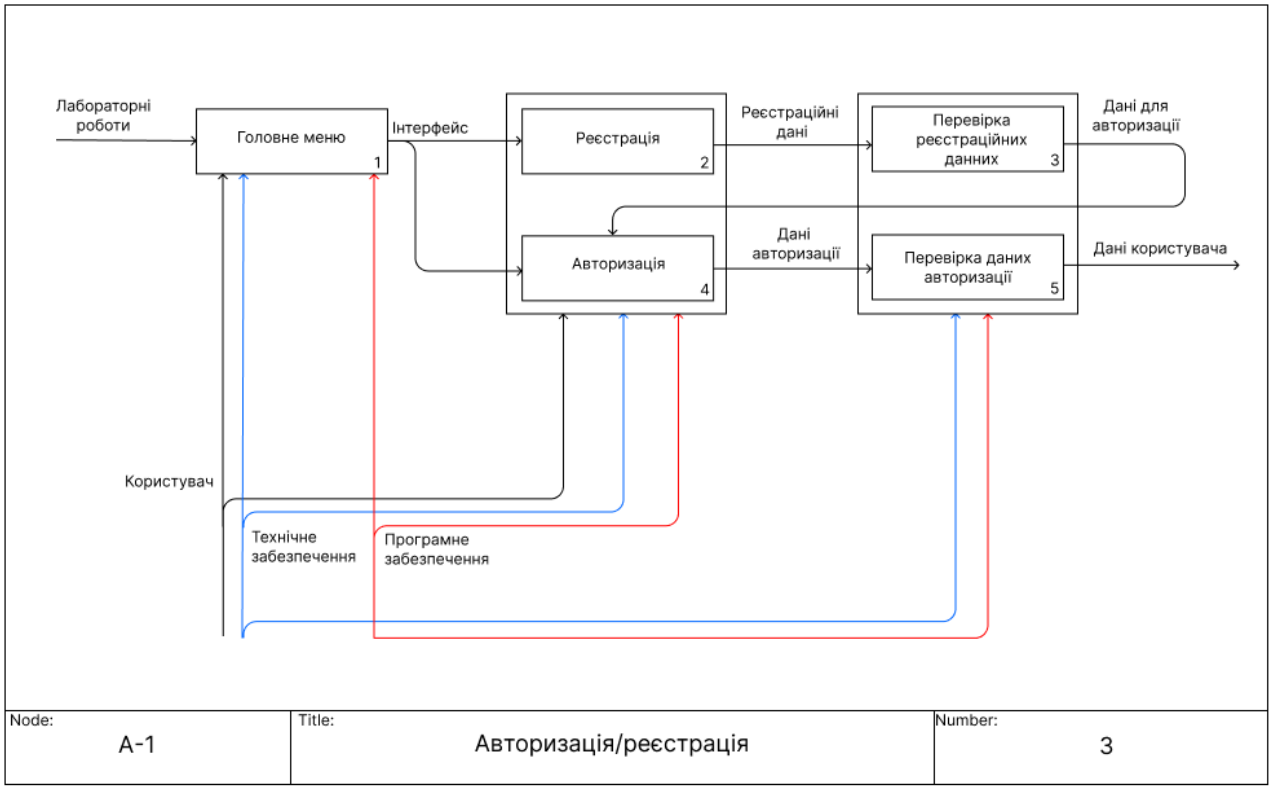

Рисунок 3.3 – Другий рівень декомпозиції IDEF0 блоку “Авторизація/реєстрація”

Для блоку “Виконання лабораторної роботи” були виділені блоки: “Вимірювання характеристик сигналу”, “Розрахунок часових параметрів сигналу”, “Розрахунок частотних параметрів сигналу”, “Введення даних в таблиці”. Між ними встановлені наступні потоки даних:

– “Характеристики сигналу” між блоками “Вимірювання характеристик сигналу” та “Розрахунок часових параметрів сигналу”;

– “Часові характеристики сигналу” між блоками “Розрахунок часових параметрів сигналу” та “Розрахунок частотних параметрів сигналу”;

– “Частотні характеристики сигналу” між блоками “Розрахунок частотних параметрів сигналу” та “Введення даних в таблиці”.

Діаграма декомпозиції другого рівня для блоку “Виконання лабораторної роботи” наведена на рисунку 3.4.

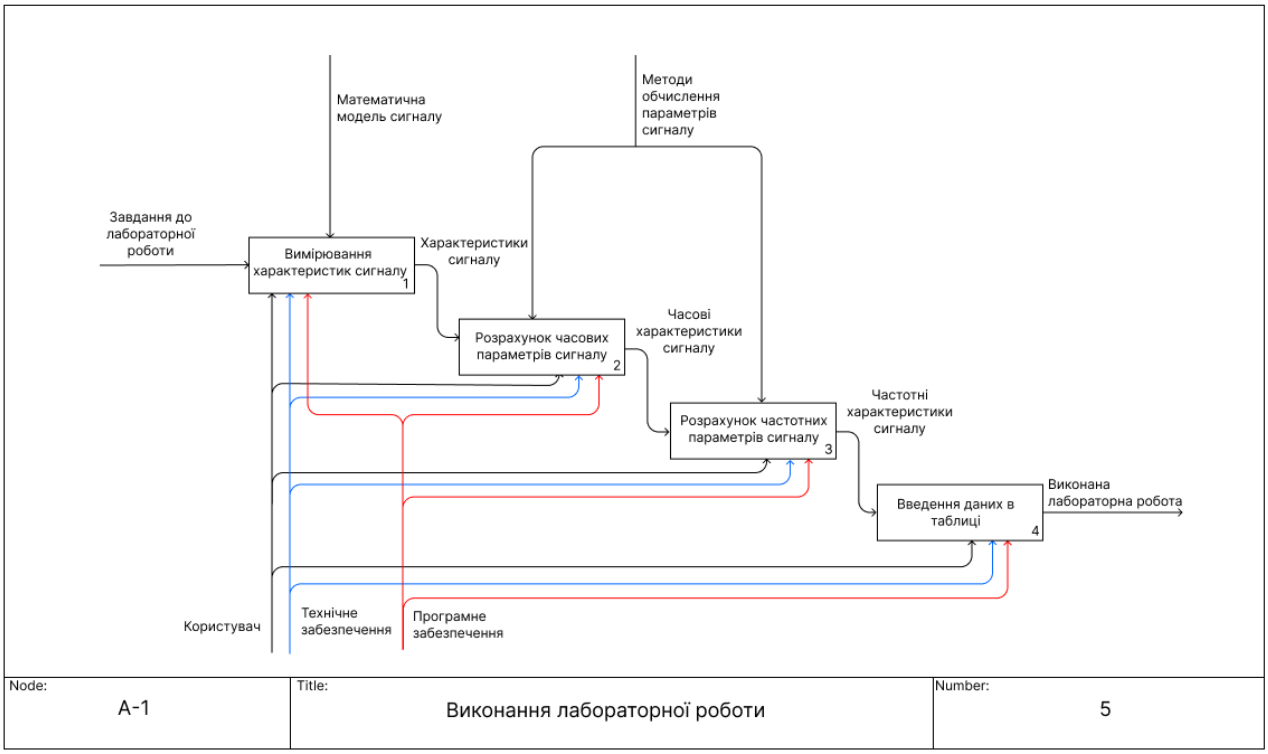

Рисунок 3.4 – Другий рівень декомпозиції IDEF0 блоку “Виконання лабораторної роботи”

Для декомпозиції другого рівня в нотації IDEF3 був обраний блок “Вибір лабораторної роботи”, так як даний блок потребує конкретизації процесів вибору лабораторної із врахуванням паралельності та синхронності виконання.

Для блоку “Вибір лабораторної роботи” були виділені наступні процеси:

– “Список лабораторних робіт”;

– “Лабораторна робота №1,2…”;

– “Початок роботи”.

Створена діаграма наведена на рисунку 3.5.

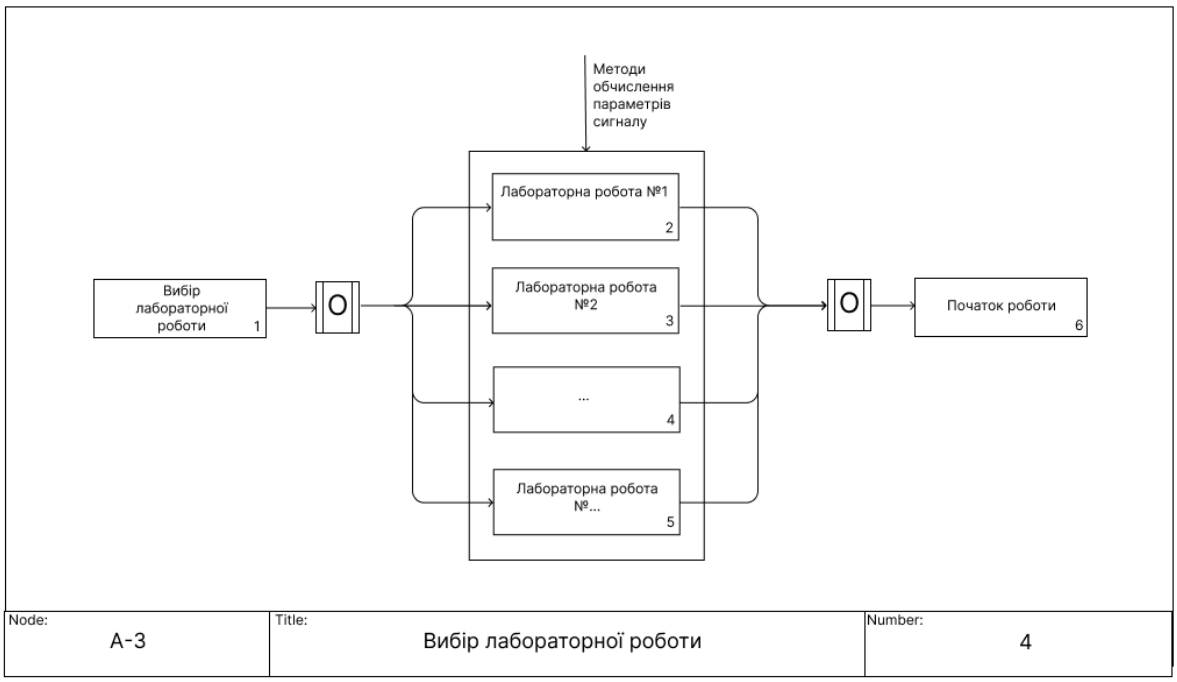

Рисунок 3.5 – Другий рівень декомпозиції IDEF3 блоку “Вибір лабораторної роботи”

Для блоку “Розрахунок результатів” були виділені блоки: “Перевірка введених даних”, “Розрахунок оцінки” та “Генерація звіту”. Між ними встановлені наступні потоки даних:

– “Кількість правильних даних” між блоками “Перевірка введених даних” та “Розрахунок оцінки”;

– “Оцінка за роботу” між блоками “Розрахунок оцінки” та “Генерація звіту”.

Створена діаграма наведена на рисунку 3.6.

На основі проведеної декомпозиції проекту за етапами можна переходити до безпосереднього виконання робіт зі створення інтерактивного додатку для роботи з лабораторними роботами на осцилографі, маючи достатнє уявлення про сторони проекту, процеси та потоки даних між ними.

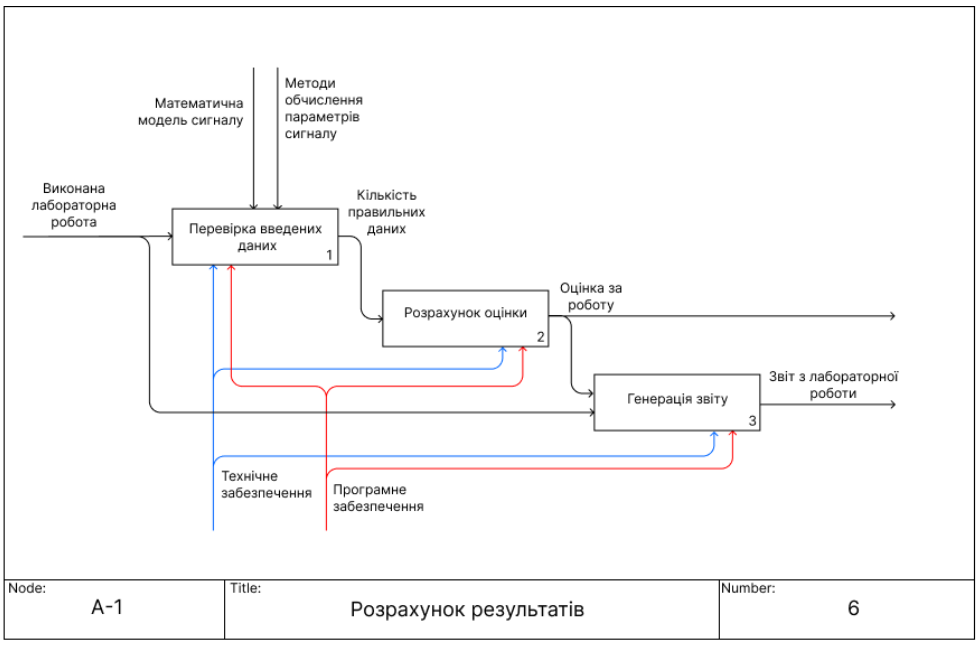

Рисунок 3.6 – Другий рівень декомпозиції IDEF0 блоку “Розрахунок результатів”

3.2 Моделювання Use-Case діаграми

Варіанти використання додатку зручніше всього показати за допомогою UML Use-Case діаграми [37]. UML Use-Case діаграма – це діаграма, що описує, який функціонал розроблюваного додатку доступний кожній групі користувачів.

Для початку потрібно визначити акторів (користувачів) розроблюваного додатку:

– Користувач (студент) – актор, який використовує додаток для роботи з осцилографом та виконання лабораторних робіт на ньому;

– Адміністратор (викладач) – актор, який може змінювати параметри лабораторних робіт (час виконання, максимальна оцінка) та надавати додаткові спроби користувачам;

– MS Word – актор, що відображає звіт з лабораторної роботи котру виконав користувач;

– MySQL – актор база даних, в яку зберігаються дані користувача.

Далі визначено основні прецеденти проекту:

– Авторизація та реєстрація (для користувача) та тільки авторизація для адміністратора;

– Робота з осцилографом (для користувача та адміністратора);

– Виконання лабораторних робіт, перегляд їх результатів та генерація звіту (для користувача та адміністратора);

– Зміна параметрів лабораторних робіт та додавання додаткових спроб для виконання лабораторних (для адміністратора).

Зв’язок акторів з варіантами використання показаний на діаграмі (рис. 3.7).

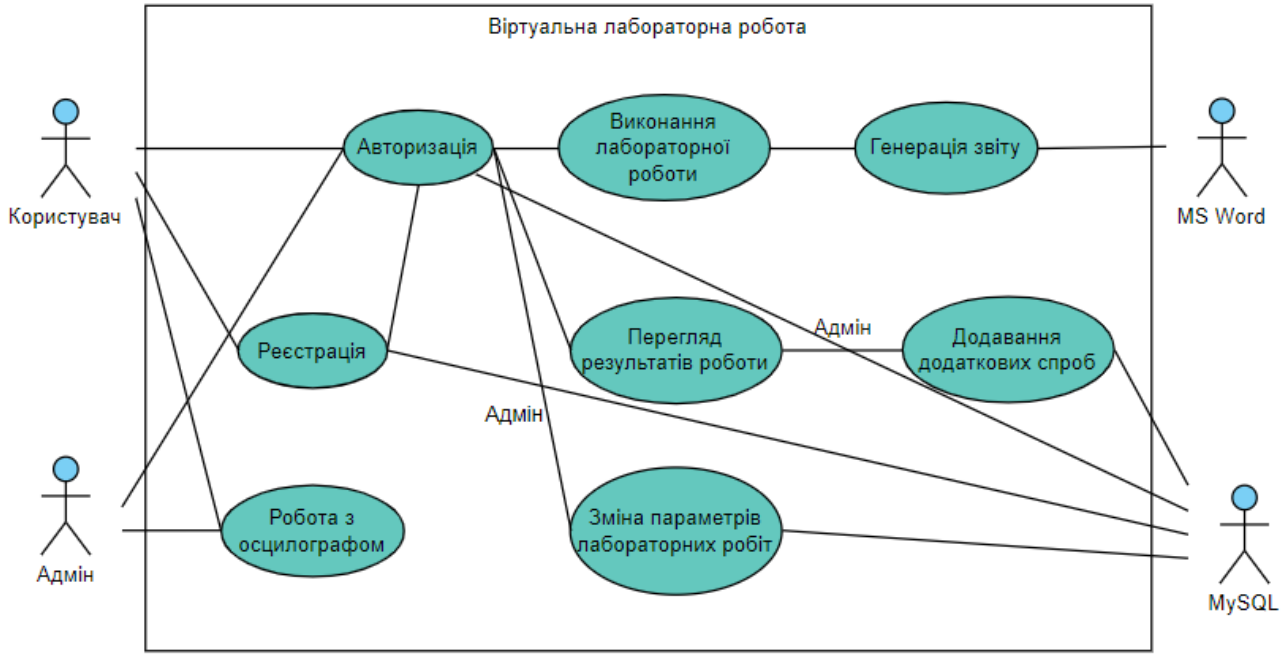

Рисунок 3.7 – Діаграма варіантів використання (Use Case)

3.3 Моделювання бази даних

Для розробки гарної схеми слід відповісти на декілька питань, які досить важливі для початку процесу проектування.

Почати слід з відповідей на питання щодо даних, так як неможливо

створити гарний дизайн бази даних без розуміння інформації, якою вона буде оперувати, і контексту, в якому буде використовуватися. Також потрібно розуміти вимоги користувача. Для більш зручного вигляду питання та відповіді представлені у таблиці 3.1.

Таблиця 3.1 – Питання та відповіді про дані, що будуть зберігатися

Питання	**Відповідь**
Які саме будуть дані?	Так як додаток спрямований на виконання лабораторних робіт користувачами, то буде зберігання інформації користувача, лабораторної яку він виконував та результати виконуваної лабораторної.
Які атрибути важливо записувати?	Головними атрибутами будуть дані користувача, його логін/пароль та особиста інформація. Також результати лабораторних робіт, а саме: оцінка, дата виконання, номер роботи та індекс користувача що виконував роботу. Кожна таблиця повинна мати унікальний індекс.
Наскільки великим буде об’єм даних?	З кількістю користувачів буде збільшуватися і об’єм даних.
Наскільки швидко система буде накопичувати нові дані?	Так як додаток орієнтований на студентів, то швидкість накопичення буде стабільною і не великою
Більшість запитів буде читання чи записом?	Майже порівну, тому що є авторизація/реєстрація, зберігання/перегляд результатів лабораторних що відповідно використовує читання/запис інформації.
Яка кількість одночасних користувачів буде?	Додаток орієнтований на роботу студентів академічної групи, тому одночасна кількість може бути 10+ користувачів.
Які дані будуть регулярно запитуватися?	Дані користувача для авторизації; дані по лабораторним, які є в додатку; назви груп, які є в університеті.
Більшість операцій націлено на окремі записи чи об’єднує більшість записів?	Більшість операцій буде окремими записами в спеціально визначені таблиці для цього.
Чи будуть надходження даних регулярними?	Ні, так як все залежить від користувача.

Проаналізувавши відповіді, можна підсумувати, що БД буде мати декілька об’єктів (таблиць) з унікальним атрибутом (полем), та атрибутом, що пов’язує дану таблицю з іншою. Одночасна кількість користувачів не є великою, і кількість записуваних даних теж. Кількість даних з часом буде збільшуватися.

Виходячи з визначення реляційної схеми БД, можна зробити висновок що нам потрібно використовувати саме її. Більшість з принципів побудови реляційних БД реалізовано в обраній СУБД MySQL, а саме там де говориться про СУБД та зберігання даних. Проте є певні пункти, які розробник БД повинен пропрацювати сам, їх можна виокремити:

– Інформація зберігається в таблицях;
– Кожна таблиця відображає якийсь один об’єкт;
– Таблиця складається з рядків та стовбців або запис та поле відповідно;
– Кожне поле (стовбець) має унікальне ім’я;
– Відсутність однакових записів (рядків).

З описаними вище пунктами можна приступити до створення БД додатку.

Як раніше сказано, додаток орієнтований на виконання лабораторних робіт користувачами, та збереження і перегляд результатів. Тому можна відразу виокремити декілька об’єктів (таблиць):

– Користувач (Студент);
– Лабораторна робота;
– Результати лабораторної;
– Групи.

Так як додаток орієнтований на студентів, також додається окремий об’єкт «Групи», що описує групи в університеті.

Для бази даних створено по кожному об’єкту таблицю з характеристиками. Для кожного об’єкту потрібно виділити основні характеристики, які дадуть змогу ідентифікувати запис (рядок) як унікальний, поля ідентифікації та поля, що зв’язують таблицю з іншими, тобто первинний та зв’язний ключ відповідно.

Також потрібно створити єдиний спільний стиль по іменах БД, таблиць,

стовпців, та для полегшення роботи при створенні запитів до БД потрібно дотримуватися правил щодо іменування БД, таблиць, стовпців, найменування яких повністю відображатиме інформацію, що зберігається в ній.

Дані зберігаються в БД, тому вони повинні мати один з типів даних, що є в MySQL, наприклад: varchar(?), int(?), float(?), date (календарна дата, може включати також час) тощо, де в дужках зазначають максимальну кількість допустимих символів. Їх також потрібно врахувати в таблиці.

Результат опису розробленої бази даних додатку, структура, імена (в дужках) та типи кожного з її об’єктів та характеристик об’єктів наведені у таблиці 3.2.

ER-діаграма бази даних наведена на рисунку 3.8.

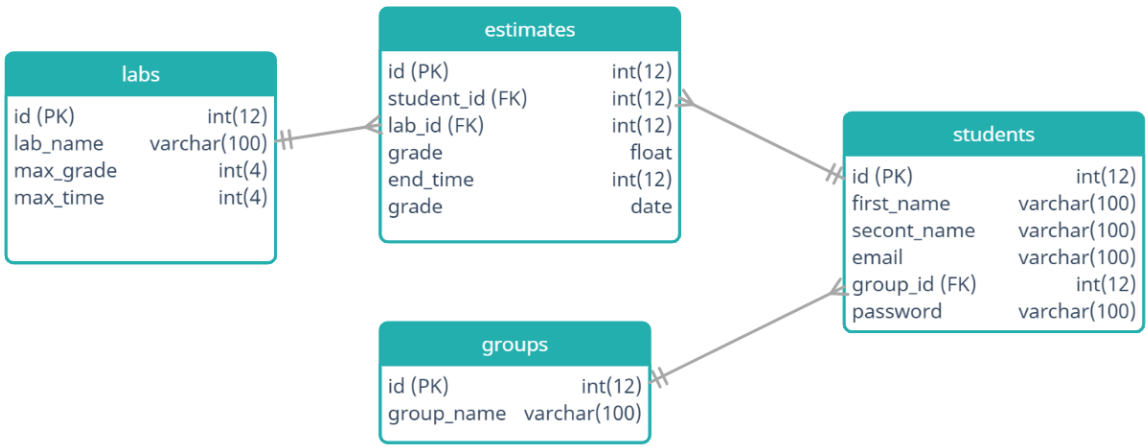

Рисунок 3.8 – ER-діаграма бази даних додатку

Таблиця 3.2 – Організація та опис об’єктів бази даних

База даних (desktop_virtual_labs)		
Об’єкт Користувач (students)		
Ім’я	**Опис**	**Тип**
id	Первинний ключ	int(12)
first_name	Ім’я	varchar(100)
second_name	Прізвище	varchar(100)
email	Пошта	varchar(100)
group_id	Зв’язний ключ з таблицею груп (groups)	int(12)
password	Пароль	varchar(100)
Об’єкт Лабораторні роботи (labs)		
Ім’я	**Опис**	**Тип**
id	первинний ключ	int(12)
lab_name	назва лабораторної роботи	varchar(100)
max_grade	максимальна оцінка за лабораторну роботу	int(4)
max_time	максимальний час виконання лабораторної роботи	int(4)
Об’єкт Результати лабораторної (estimates)		
Ім’я	**Опис**	**Тип**
id	первинний ключ	int(12)
student_id	зв’язний ключ з таблицею користувачів (students)	int(12)
lab_id	зв’язний ключ з таблицею лабораторних (labs)	int(12)
grade	результуюча оцінка за лабораторну	float
end_time	час, за який виконано лабораторну	int(10)
date	дата, коли виконано лабораторну	date
Об’єкт Університетські групи (groups)		
Ім’я	**Опис**	**Тип**
id	первинний ключ	int(12)
group_name	назва групи	varchar(100)

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1 Створення 3D моделі осцилографа та імпорт до Unity

4.1.1 Створення 3D моделі

Створення 3D моделей сцени та самого осцилографу реалізовано у 3D редакторі Blender.

Головні інструменти, з якими частіше всього працюють в ньому, це:

- **Select Box/Circle/Lasso** – виділення вершин/ребер/сторін;
- **Move** – зміна позиції обраних елементів;
- **Rotate** – поворот обраних елементів;
- **Scale** – зміна розміру обраних елементів;
- **Extrude** – витискання обраних елементів;
- **Bevel** – створення фаски (скосу різного діаметру) на ребрі;
- **Loop Cut/Knife** – розріз об’єкту (створення додаткових вершин).

За допомогою даних інструментів створюємо модель осцилографа на основі концепту (рис. 1.2). Спочатку створюємо основну форму з дотриманням розмірів моделі (рис. 4.1).

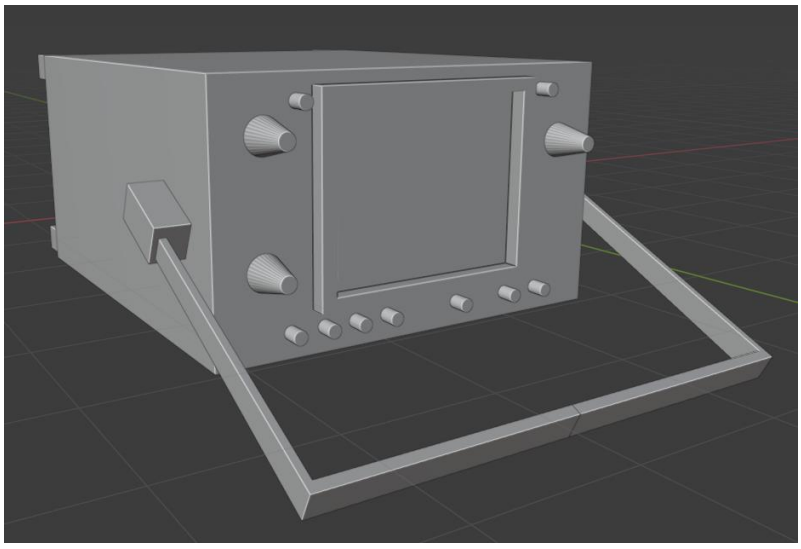

Рисунок 4.1 – Основа моделі осцилографа

Далі до створеної основи треба додати деталізацію, фаски та збільшувати кількість вершин (для більш гладкої форми моделі). В результаті буде отримано high-poly (високополігональну) деталізовану модель. Вона містить велику кількість вершин, відповідно потребує і більшу кількість ресурсів комп'ютеру для обробки моделі. Тому потрібно провести оптимізацію моделі за допомогою технології запікання. Спочатку переводимо high-poly модель в low-poly (низькополігональну). Low-poly модель має не всю кількість елементів high-poly моделі, так як більшість із них повторюються, і можна це використати для оптимізації. На рисунку 4.2 наведено високо- та низько-полігональні моде.лі осцилографа та відповідно кількість вершин у моделях після проведення оптимізації.

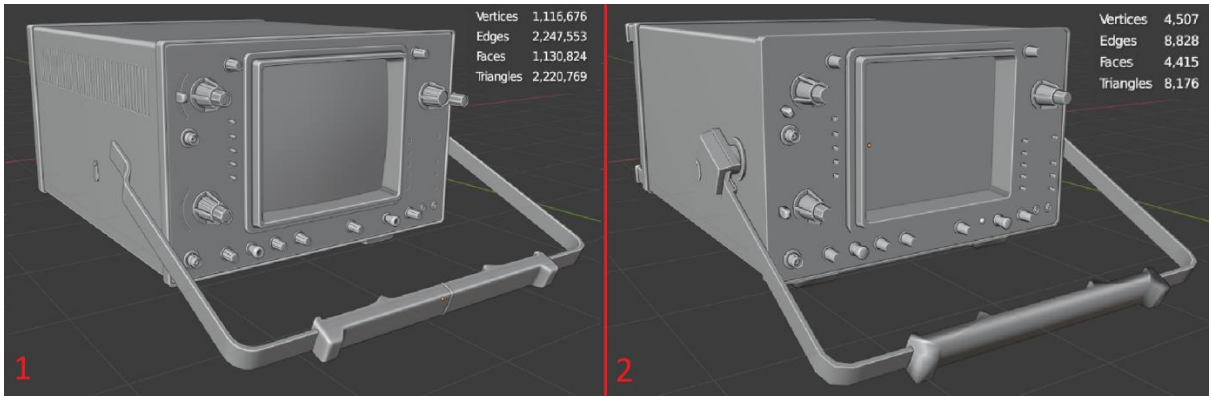

Рисунок 4.2 – Оптимізація моделі осцилографа

Передфінальним етапом є створення UV розгортки [29]. UV розгортка потрібна моделі для того, щоб 3D додатки, які працюють з моделлю, розуміли як саме потрібно накласти 2D зображення (текстури) на 3D об'єкт.

Blender представляє функціонал по автоматичній розгортці моделей. Отриману UV розгортку можна переглядати та редагувати у додатковому вікні (рис. 4.3).

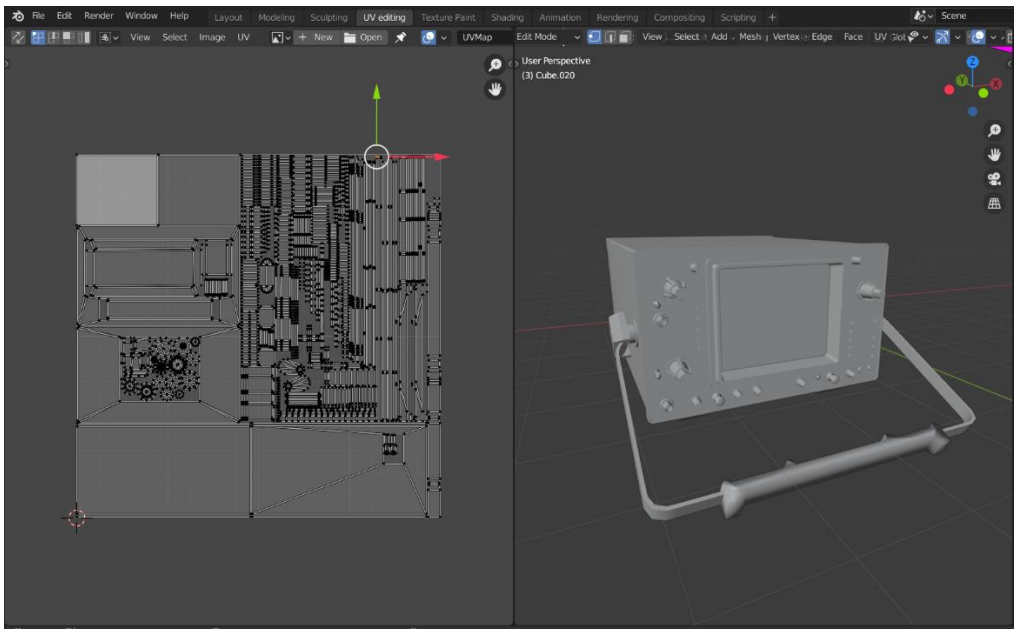

Рисунок 4.3 – UV розгортка (ліворуч) моделі осцилографа

Фінальний етап – експорт моделі в Marmoset, але перед цим потрібно підготувати моделі до експорту. Для цього high- та low-poly моделі потрібно розбити на унікальні елементи, які і будуть запікатися (рис. 4.4). Також потрібно правильно задати назву кожному елементу для полегшення імпорту, так як Marmoset автоматично розподіляє моделі за назвою, тобто якщо у нас є модель з назвою model_high та model_low, Marmoset сам додасть їх до відповідного файлу.

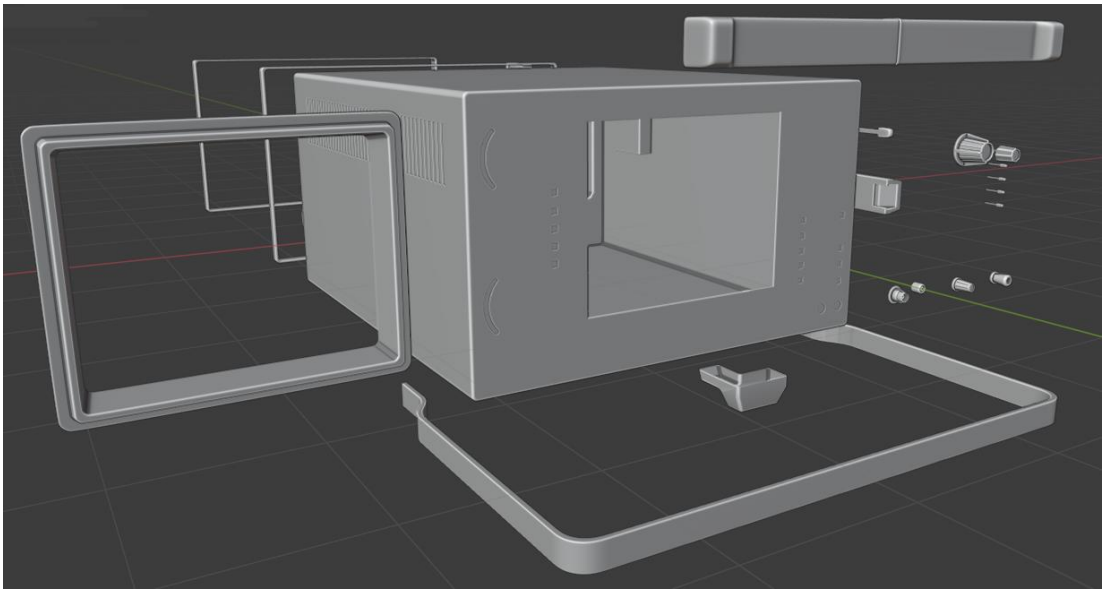

Рисунок 4.4 – Розбиття моделі на унікальні елементи

Для експорту потрібно обрати формат моделі fbx, який приймає Marmoset, та поставити відмітку на Selected Objects, щоб експортувати тільки виділені елементи (рис. 4.5).

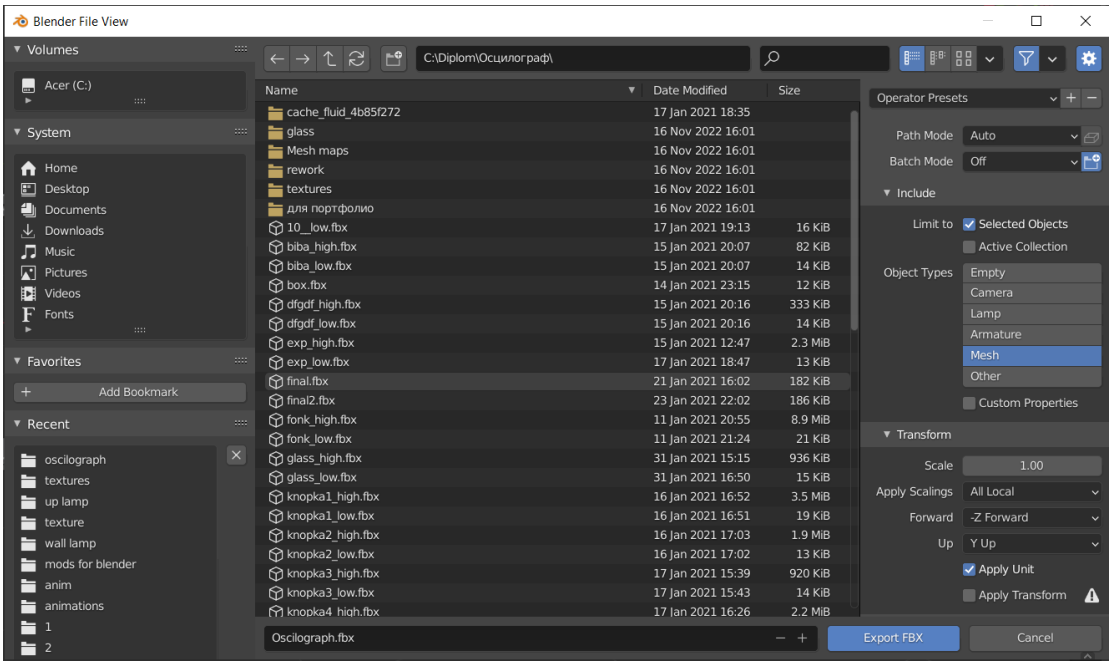

Рисунок 4.5 – Меню налаштувань експорту

4.1.2 Запікання карт моделі

Після експорту моделей в fbx формат, переходимо до додатку Marmoset Tool Bag, що дає можливість запікати, текстурувати та рендерити моделі. В даній роботі використовуватися буде тільки функція запікання карт.

Процес запікання, якщо пояснювати зрозумілими прикладами, – це як загортання подарунку, наприклад велосипеду, в обгортку – зникає більшість деталей, але все одно видно, що в обгортці велосипед.

Для запікання в діалоговому вікні зазначаємо модель та в налаштуваннях обираємо куди буде збережено карти запікання, якість запечених текстур, спосіб обрахунку геометрії (так як різні 3D редактори використовують різні напрямки осів), спосіб запікання та які саме карти будуть запечені (рис. 4.6).

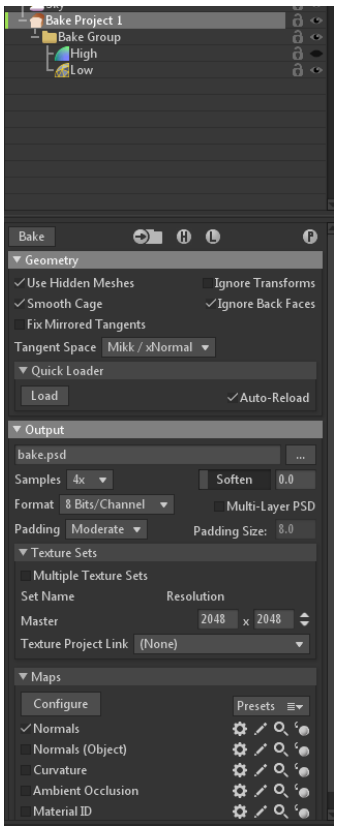

Рисунок 4.6 – Налаштування об’єкту запікання

Для додатку Substance painter, де будуть створювати текстури, потрібно декілька карт запікання, серед основних – Normals та Ambient Occlusion, та додаткові:

- **Curvature** – показує кривизну поверхні;
- **Thickness** – показує товщину поверхні (в випадку моделі, відстань від полігону до найближчого);
- **Position** – показує розміщення моделі в 3D просторі;
- **World space normal** – ще одна інтерпретація розміщення моделі в просторі але з іншими осями;
- **Cavity** – самозатемнення для дуже малих елементів, наприклад пори на шкірі.

Додаткові карти нададуть більше інформації про геометрію моделі, щоб додатку краще розуміти, як правильно накласти ту чи іншу маску з налаштуваннями. Фінальні карти результату запікання наведені на рисунку 4.7.

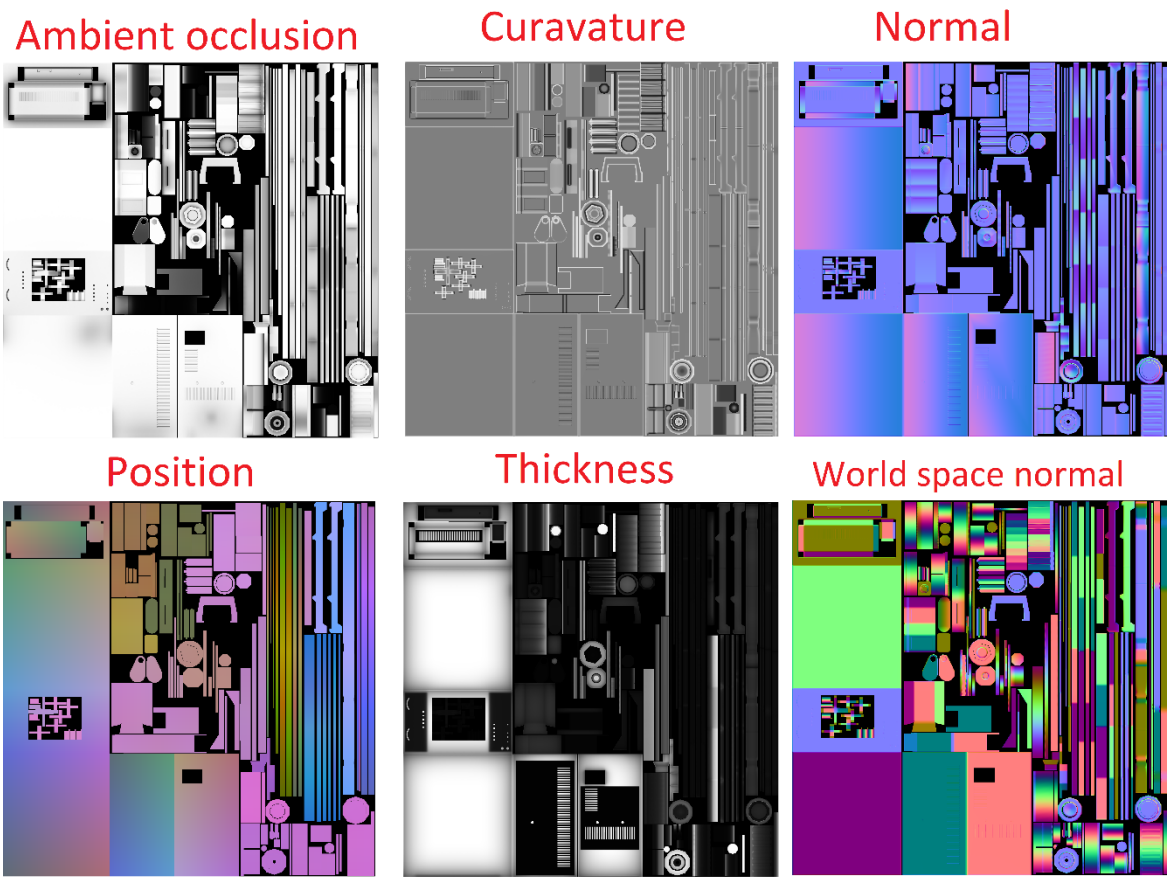

Рисунок 4.7 – Карти запікання моделі осцилографу

4.1.3 Текстурування моделі

Substance painter що дозволяє створювати дуже якісні текстури на моделі через малювання або використання «розумних» матеріалів, які на основі раніше створених карт вираховують позицію накладення різних текстур на моделі, по типу іржі на металі.

Substance painter використовує PBR (Physically Based Rendering) шаблон правил та технологій відображення матеріалів, тобто рендеринг матеріалів відбувається з урахування законів фізики.

В PBR є два шаблони, це Specular/Gloss та Metallic/Roughness. Перший тип частіше всього використовується в матеріалах живої природи, другий в матеріалах неживої природи. Metallic/Roughness використовує 6 основних мап для передачі матеріала:

– **Base Color (albedo) або колір** – колір матеріалу;

– **Roughness або шорсткість** – показує, наскільки матеріал шорсткий або

гладкий (від 0 до 1), тобто чим шорсткіше матеріал – тим більш розсіяним буде світло, що відбивається, і навпаки, чим гладше – тим більш сконцентрованим буде пляма світла на матеріалі;

– **Metalness або металічність** – показує до яких металів відноситься матеріал – до діелектриків (0) чи провідників (1);

– **Normal map або карта нормалі** – текстура, що використовується для технології фейкового відбивання світла від нерівних поверхонь, застосовується для малих деталей на моделі;

– **Occlusion map або карта самозатінення** – робить об’єкти з щілинами більш реалістичним, додаючи тіні до закритих ділянок;

– **Transparent або прозорість** – показує наскільки прозорий матеріал в діапазоні від 0 до 1.

Для імпорту моделі достатньо перенести fbx файл у вікно додатку, де відкриються додаткові налаштування імпорту (рисунок 4.8), і можна обрати шаблон матеріалу, розмір текстур моделі та формат обробки карт нормалі тощо. Розмір текстур було обрано 2048 на 2048, так як модель буде знаходитися поблизу користувача і неякісні текстури буде дуже помітно. Також у вікні General Properties було додано раніше створені карти запікання.

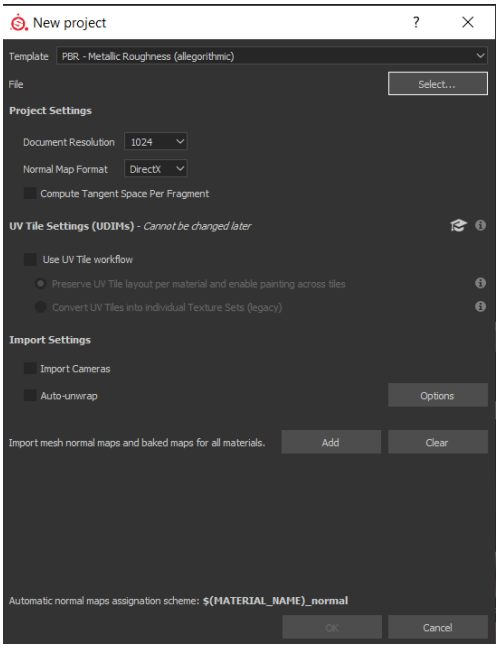

Рисунок 4.8 – Налаштування імпорту моделі в Substance painter

Текстурування в Substance painter відбувається за допомогою додавання шарів до моделі. В шарах можна налаштувати всі зазначені в PBR параметри, щоб створити матеріал по типу металу, пластику, шкіри тощо. До шарів можна додавати чорно-білі маски, які показують де саме матеріал буде відображатися, та ступінь її прозорості. В результаті створено текстури для моделі осцилографа на основі раніше обраних концептів (рис. 1.2).

Результат текстурування моделі показаний на рисунку 4.9, де можна бачити велику кількість шарів матеріалів та саму модель, яка майже не відрізняється від реальної.

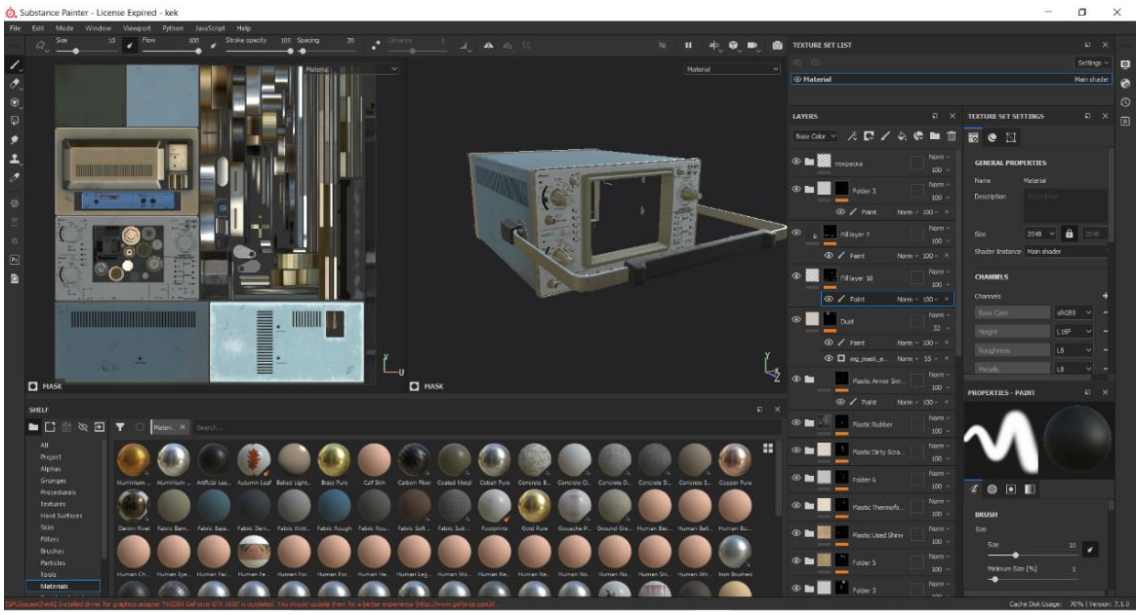

Рисунок 4.9 – Фінальний варіант текстур осцилографа

Останнім етапом є експорт текстур. В налаштуваннях експорту зазначається шлях збереження текстури, шаблон та формат збереження текстур. При виборі PBR формату Substance сам експортує потрібні для цього карти. Набір PBR карт, які було створено, показано на рисунку 4.10.

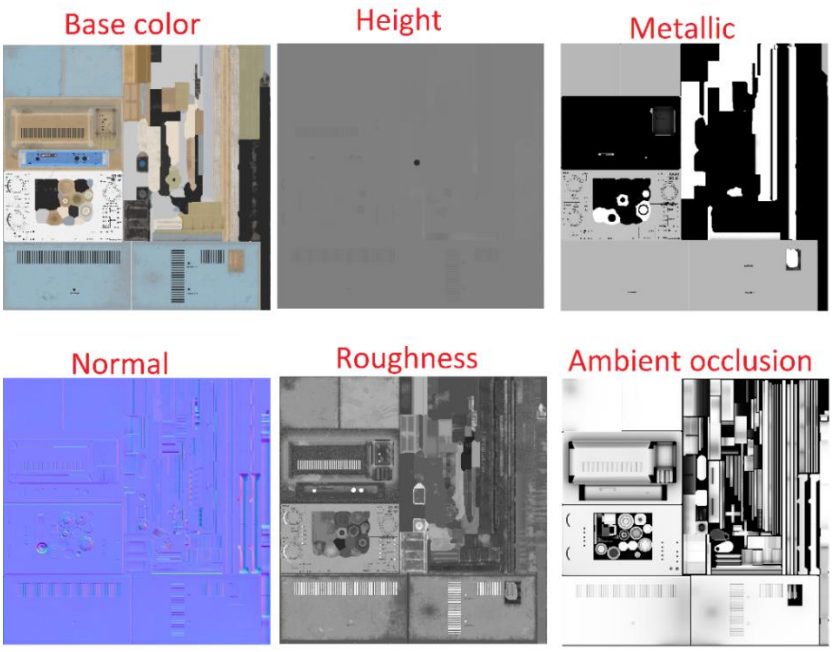

Рисунок 4.10 – Набір PBR карт матеріалу осцилографа

4.1.4 Імпорт текстур та моделі до Unity

Експорт моделі осцилографу з Blender до Unity відбувається в тому ж форматі fbx. Так як Blender використовує інше представлення координатної системи, де Z направлена вгору, а в Unity – вісь Y, то потрібно в налаштуваннях експорту (рис. 4.11) зазначити вірний напрям осі.

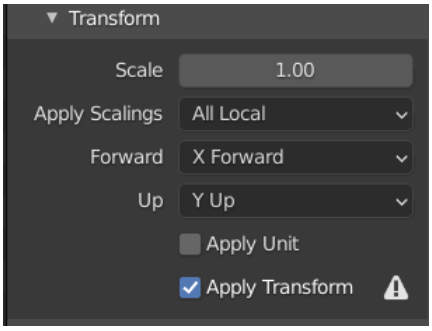

Рисунок 4.11 – Налаштування трансформу при експорті до Unity

Для імпорту моделей до Unity потрібно лише перенести їх у вікно проекту, де вони відразу будуть відображені в 3D вигляді (рис. 4.12). Сама модель осцилографа розбита на два типи компонентів: статичні та динамічні. До динамічних відносяться об’єкти, з якими користувач буде взаємодіяти, тобто кнопки, все інше статичні. Для додавання об’єктів в сцену потрібно перемістити

їх до сцени.

Рисунок 4.12 – Відображення 3D моделей в проекті Unity.

Для налаштувань матеріалів з текстурами, реалізованими в попередньому розділі, в Unity потрібно створити порожній матеріал та додати ті всі карти відповідно за назвою, і, за потреби, додатково налаштувати (рис. 4.13). Щоб додати матеріал до моделі, достатньо перенести його з проекту на модель в сцені. Фінальний вигляд моделі осцилографа наведено на рисунку 4.14.

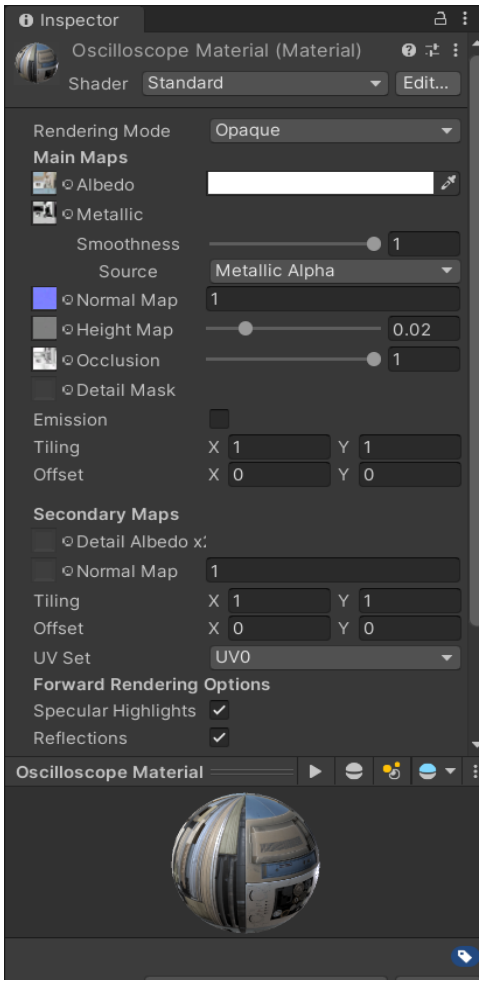

Рисунок 4.13 – Налаштування матеріалу

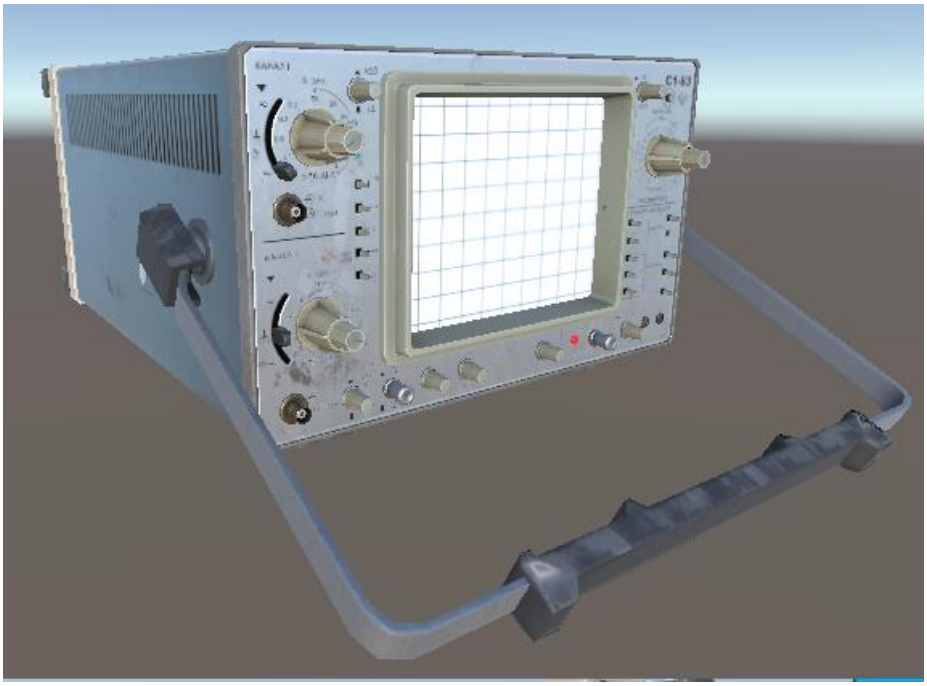

Рисунок 4.14 – Фінальний вигляд осцилографу в Unity

4.2 Архітектура додатка

Додаток складається з клієнту та серверу, клієнт має 3 основні модулі, в які надається інформація введена користувачем в конкретні поля для введення:

– **Модуль обробки інформації** – збирає інформацію з полів для введення та оброблює її;

– **Модуль генерації звіту** – приймає зібрану інформацію з модулю обробки інформації та генерую з неї звіт для користувача;

– **Модуль відправки/прийому даних** – приймає зібрану інформацію з модулю обробки інформації, запаковує її та відправляє до серверу в вигляді HTTP запиту. Сервер в свою чергу відправляє відповідь на запит в вигляді JSON, яка оброблюється.

Сервер має в собі:

– **Скрипти** – набір серверних скриптів що приймають інформацію, оброблюють її та формують запит до бази даних який потім відправляють клієнту (додатку).

– **БД (База даних)** – зберігає в собі інформацію додатку що була надана серверними скриптами.

Архітектура додатку з модулями та їх взаємозв’язками зображена на рисунку 4.15.

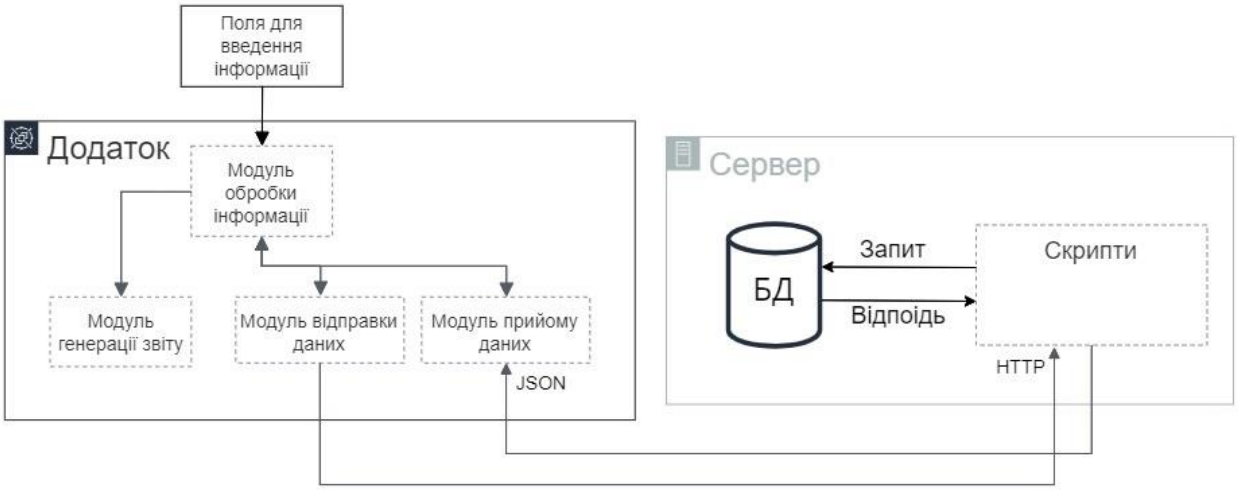

Рисунок 4.15 – Архітектура додатка

4.3 Програмна реалізація симуляції роботи осцилографу

Для реалізації показу сигналу на екрані, в Unity існує компонент відображення прямої за координатами X та Y – LineRenderer. Компонент бере масив з двох або більше точок в тривимірному просторі та малює пряму лінію між цих точок. Для LineRenderer в сцені задамо налаштування лінії, її колір, товщину, форму, матеріал та реакції світла на лінію (рис.4.16).

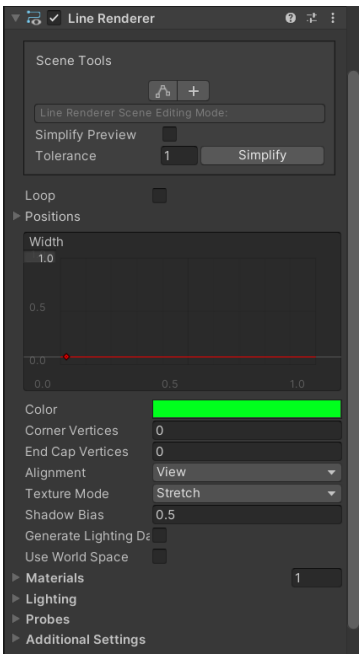

Рисунок 4.16 – Налаштування компоненту LineRenderer

Управління компонентом LineRenderer можна отримати з коду за допомогою однойменного об’єкту, тому було створено скрипт Sinwave.cs. Щоб не шукати об’єкт в сцені, Unity надає змогу створювати публічні змінні в коді, в які користувач сам може додати потрібний об’єкт, який буде ініціалізовано в цю змінну. Тому було створено публічну змінну LineRenderer та додано раніше створений об’єкт з сцени.

Для відображення точок кожної секунди, в Unity існує метод Update, який викликає закладений в нього код один раз за один кадр. Тому для динамічної зміни кривої в сцені потрібно реалізовувати логіку, що додає x та y до масиву LineRenderer. Так як LineRenderer тільки малює лінію між двома точками, то потрібно динамічно перебирати масив з точками, брати поточну точку та попередню, і з’єднувати їх за допомогою створення нового компоненту LineRenderer (рис. 4.17).

```
void Draw()
{
    float xStart = xLimits.x;
    float xEnd = xLimits.y;

    myLineRenderer.positionCount = points;
    for (int currentPoint = 0; currentPoint < points; currentPoint++)
    {
        float progress = (float)currentPoint / (points - 1);
        this.x = Mathf.Lerp(xStart, xEnd, progress);
        this.y = calculateY();
        myLineRenderer.SetPosition(currentPoint, new Vector3(this.x, this.y, 0));
    }
}
```

Рисунок 4.17 – Код малювання кривої в Update

Щоб користувач зміг взаємодіяти з сигналом, будуть використовуватися різного роду перемикачі. Для реалізації цього потрібно в скрипті Sinwave.cs, що реалізує відображення сигналу, створити публічні методи для зміни кожного з показників, щоб потім інший метод іншого класу зміг звертатися до нього та змінювати параметри кривої.

В моделі осцилографа є 3 види перемикачів, що:

– повертаються;

– натискаються;

– повертаються та натискаються.

Для перемикачів, що натискаються, створено скрипт Buttons.cs. Для обробки натискання на об’єкт, на якому знаходиться скрипт, меню чи то сцени, використано методи OnMouseUp, OnMouseDown, OnMouseHold (рис. 4.18).

```
void OnMouseDown()
{
    this.isPressed = true;
    StartCoroutine(DoButtonsCheck());
}

Unity Message | 0 references
private void OnMouseUp()
{
    this.isPressed = false;
    time = 0.1f;
    timeScale = 1.0f;
}
```

Рисунок 4.18 – Реалізація методів OnMouseDown, OnMouseUp

Для повороту перемикачів використовуються кватерніони [38], які є комбінацією тривимірного вектору та скаляру для представлення повороту чи орієнтації об’єкту. Структура кватерніону виглядає наступним чином:

$$(x_i,\ y_j,\ z_k, \text{w}),$$

де $(x_i,\ y_j,\ z_k)$ – одиничний вектор, що представляє кут між орієнтацією та кожною окремою віссю;

w представляє собою кут повороту вздовж одиничного вектору $(x_i,\ y_j,\ z_k)$.

Для повороту береться початковий вектор орієнтації об’єкта та помножується на кут, на який потрібно повернути – в результаті отримаємо кватерніон з потрібним нам направленням, що встановлюється як початковий для об’єкту. Для “анімації” повороту береться час повороту, що передається як закінчення циклу до самого циклу, та в ньому за допомогою методу Quaternion.Lerp, що згладжує зміну значення від початкового до кінцевого, передається початковий та кінцевий вектор. Поки не закінчиться цикл, об’єкт буде змінювати свій поворот. Код реалізації поворота наведено на рисунку 4.19).

```
Quaternion startRotation = transform.rotation;
Quaternion targetRotation = transform.rotation * Quaternion.Euler(localRotateAngle, 0f, 0f);

float angle = Quaternion.Angle(startPos, targetRotation);

if (angle < maxRotate) {
    while (timeElapsed < lerpTime)
    {
        transform.rotation = Quaternion.Lerp(startRotation, targetRotation, curve.Evaluate(timeElapsed/lerpTime));
        timeElapsed += Time.deltaTime;
        yield return null;
    }

    transform.rotation = targetRotation;
    isRotate = false;

    sendScaler(sigh);
}
else {
    yield return null;
}
```

Рисунок 4.19 – Код повороту об'єкту.

В результаті створено відображення вхідного сигналу зі змінними параметрами сигналу та два скрипти: Rotator.cs та Buttons.cs, що надають змогу натискати й повертати об’єкти, на яких знаходяться дані скрипти. Повний код Sinwave.cs, Rotator.cs, Buttons.cs наведений в додатку Б.

4.4 Клієнт-серверна реалізація

Перед початком роботи з клієнт-серверною частиною, потрібно встановити і налаштувати веб сервер MAMP та базу даних.

Для налаштування серверу треба обрати параметри запуска сервера зі стартом додатку; порти, на яких буде розміщено сервер та БД; версію PHP та її тип хешування; один з двох серверів та шлях до серверних скриптів (рис. 4.20). Для більш детального налаштування, по типу зміни логіну та паролю БД, треба змінити параметри в конфігураційних файлах сервера.

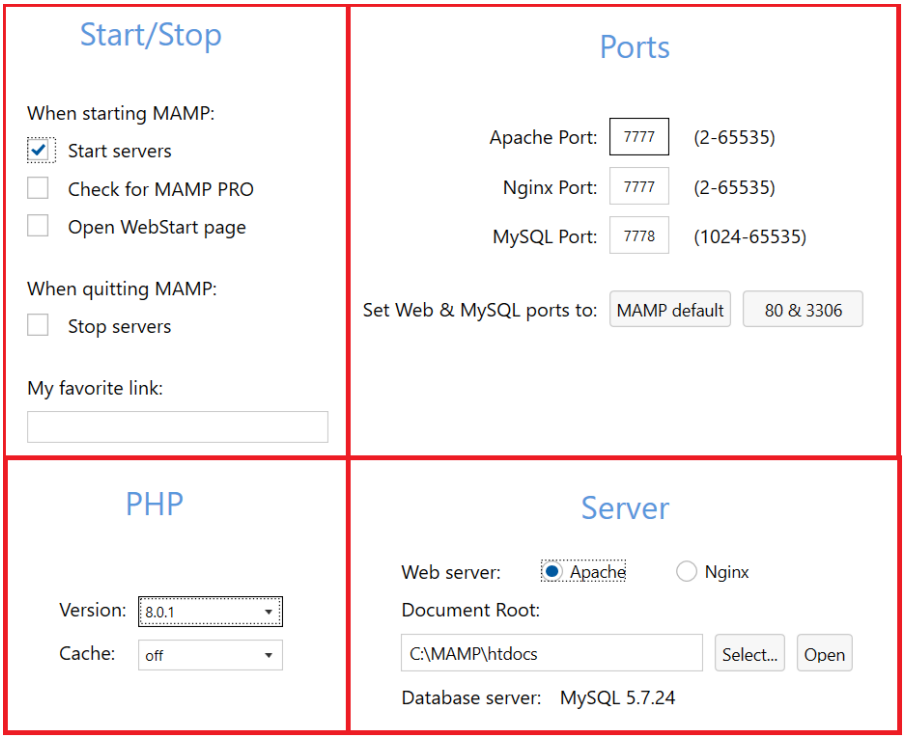

Рисунок 4.20 – Налаштування серверу в MAMP

4.4.1 Клієнт частина (меню та взаємодія з сервером)

Клієнтську частину додатку слід розпочати з розробки головного меню додатку, через яке і буде відбуватися взаємодія клієнта з сервером та додатком. На основі проаналізованих аналогів було сформовано перелік вимог до меню:

– меню не повинно заважати користувачу виконувати лабораторну на пристрої;

– меню повинно мати: реєстрацію, авторизацію, змогу виконувати лабораторні роботи, перегляд результатів;

– зрозумілість меню, щоб користувач з першого погляду міг зрозуміти що йому робити.

Щодо першого пункту, то в мобільних додатках або інтернет сторінках існує так зване висувне меню (Slide menu), яке відкривається коли користувач тягне його або натискає на кнопку меню. Таке меню надасть змогу користувачу одночасно дивитися на зміст лабораторної роботи, слідкувати за пунктами виконання, вносити зміни та працювати на приладі, порівнювати результати вимірів.

По двом іншим пунктам – кожна функція розбита на окремі сторінки в меню, доступ до яких буде відбуватися через кнопки з підписом. По даним аналізу створено макет меню (рис. 4.21).

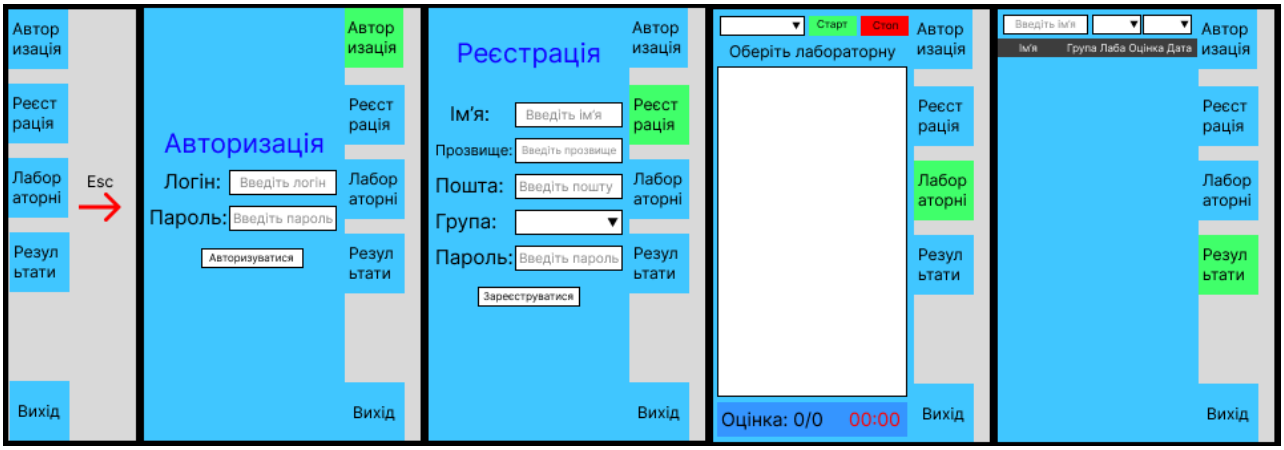

Рисунок 4.21 – Макет меню додатку

Для створення меню в Unity потрібно додати в сцену новий об'єкт Canvas, що використовується як контейнер для елементів меню (кнопки, списки, поля для введення). До нього прикріплюється EventSystem, який оброблює натискання на кожен елемент меню. В налаштуваннях Canvas потрібно вказати, як саме буде чи не буде розширюватися меню залежно від різних екранів та початкового розширення, яке було обрано 1920 на 1080, що є стандартом для більшості комп'ютерів.

В Unity існують такі елементи меню:

– **Button** – кнопка з обробкою натиску;

– **Text** – відображає заданий текст;

– **Input Field** – поле для введення інформації користувачем;

– **Dropdown –** спадний список з обробкою натискання;

– **Image** – відображає поміщене в нього зображення або фон для елементів меню.

Кожен елемент представляє собою об’єкт з відповідними компонентами, для яких можна налаштувати потрібний різноманітний функціонал .

Також існує додатковий модуль для меню Text Mesh Pro (TMP), що додає більше налаштувань тексту до елементів.

За допомогою вищеописаних елементів та створеного макету, було розроблене меню (рис. 4.22). Ліворуч показано список всіх головних елементів меню, праворуч сторінка авторизації користувача (інші сторінки вимкнені так як вони накладаються друг на друга, і вмикаються при натисканні на відповідну кнопку).

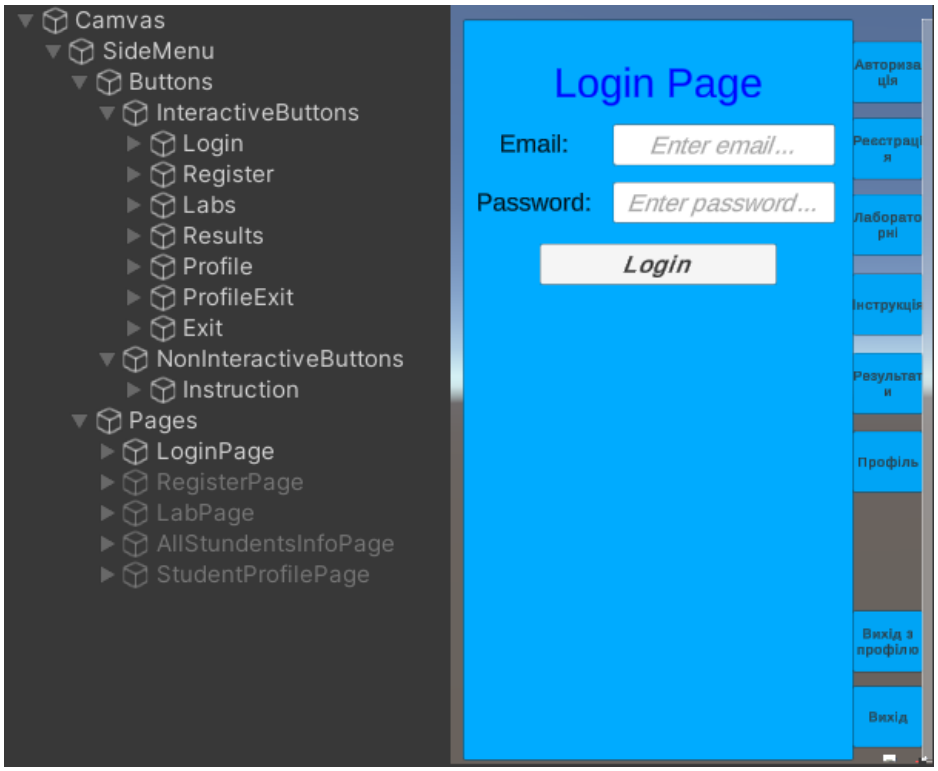

Рисунок 4.22 – Список елементів меню та сторінка авторизації

Для використання елементів меню в коді, динамічного зчитування/запису інформації, потрібно в коді створити публічну/серріалізовану (ініціалізується один раз) змінну такого ж типу, як і потрібний елемент меню, що потрібно зв’язати з кодом. Зазвичай це шаблонна назва самого елементу меню (Button, InputField і т.д.), але так як буде використано бібліотеку TMP, то до типу змінної додається префікс TMP_ (TMP_Button, TMP_InputField і т.д). Через інтерфейс Unity, до створеної змінної можна додати потрібні елементи меню до скрипту (рис. 4.23).

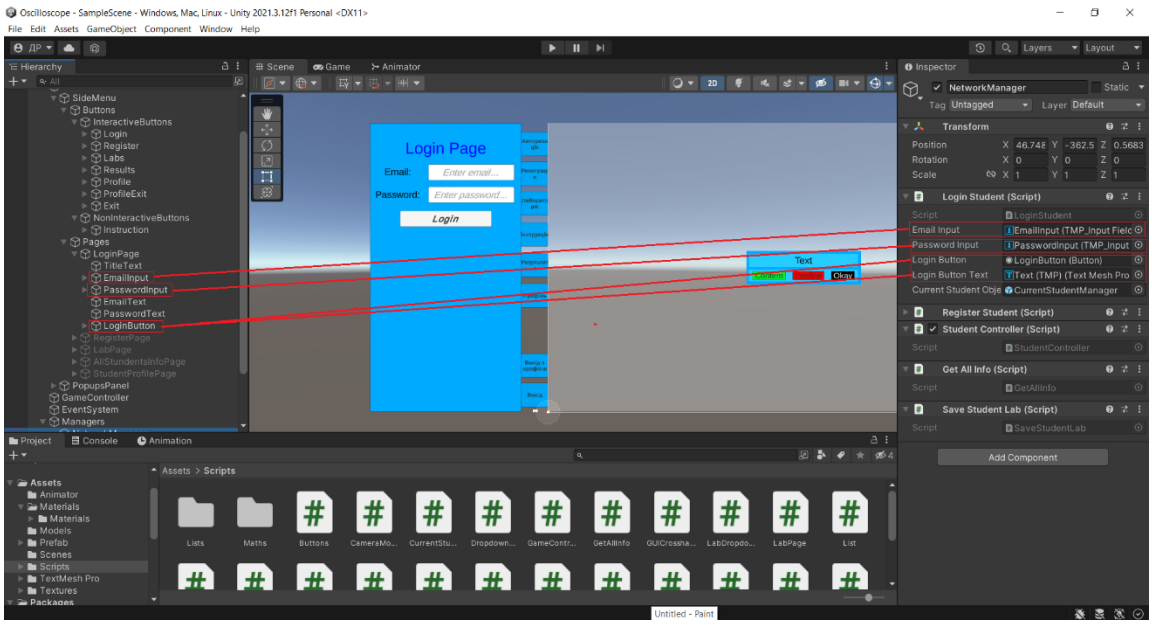

Рисунок 4.23 – Додавання елементів меню до змінних в коді

Щоб реалізувати функцію появи меню, створено скрипт SideMenu.cs (рис. 4.24), куди надається об'єкт з усіма елементами меню та створюється метод ShowHide, який за допомогою об’єкту Animator викликає анімацію переміщення меню до видимої області користувача. В Update створено перевірку на натиск кнопки Escape на клавіатурі – якщо кнопка була натиснута, то викликається метод ShowHide, що відкриває або закриває меню.

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        FindObjectOfType<SideMenu>().ShowHide();
        gameIsPaused = !gameIsPaused;
        PauseGame();
    }
}
public void ShowHide()
{
    if (panelMenu != null)
    {
        Animator animator = panelMenu.GetComponent<Animator>();
        if (animator != null)
        {
            bool isOpen = animator.GetBool("Show");
            animator.SetBool("Show", !isOpen);
        }
    }
}
```

Рисунок 4.24 – Реалізація зсуву меню по натисканню кнопки

Для переключення сторінок меню по натисканню на відповідні кнопки створено скрипт TabGroup.cs, що доданий до елементу меню Buttons, в якому знаходяться всі кнопки. Також створено модель (представлення) кнопки TabButton.cs, в якій знаходяться об’єкт класу TabGroup, зображення кнопки, показник блокування кнопки та перевизначені методи OnPointerClick, OnPointerEnter, OnPointerExit інтерфейсів IPointerEnterHandler, IPointerClickHandler, IPointerExitHandler що відповідають на реакцію кнопки на курсор.

В скрипті TabGroup є публічний лист ObjectsToSwap, в який додано всі сторінки меню (рис. 4.25), і при натисканні на кнопку викликається метод OnPointerEnter. Він в свою чергу викликає методи TabGroup, що активують потрібну сторінку зі списку та деактивують інші, тим самим перемикаючи їх. Прив’язка кнопки до сторінки відбувається за розташуванням кнопки в списку елементів, тобто 1 кнопка відповідає за 1 сторінку в листі TabGroup.

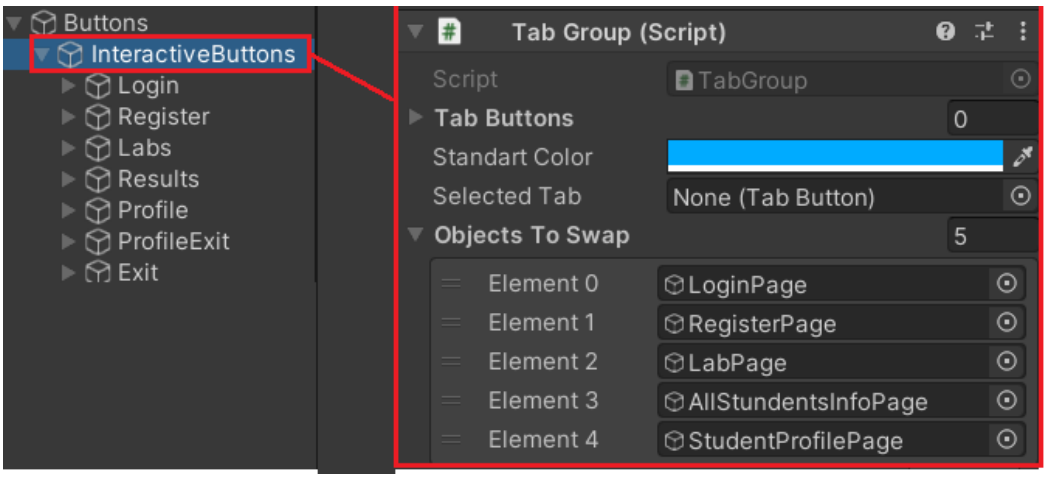

Рисунок 4.25 – Розміщення скрипту TabGroup та список із сторінками меню

Для відправки інформації, яка введена користувачем, використано бібліотеку Unity.Networking, що дозволяє створювати запити та відправку інформації у вигляді HTTP форми на вказану адресу. В об’єкт бібліотеки WWWForm додано поле зі значенням та ключем доступу до значення.

Для відправки інформації створено об’єкт UnityWebRequest, на якому викликається UnityWebRequest.Post(string url, WWWForm formData) з адресою відправки та формою з інформацією. Потім на створеному об’єкті потрібно визвати SendWebRequest(), що відправить інформацію на раніше введену адресу.

Якщо буде відповідь з серверу, то за допомогою downloadHandler.text її можна отримати з UnityWebRequest об’єкту. Так як UnityWebRequest завжди очікує на відповідь від серверу, то метод повинен повертати об’єкт IEnumerator, що дозволяє паралельно очікувати на інформацію, яка може повернутися через деякий час, за допомогою yield return і не блокувати методи, що йдуть далі по виконанню. Для відправки інформації на раніше встановлений сервер потрібно вказати IP комп’ютера, на якому встановлений сервер, порт, на якому встановлений веб сервер, та шлях до скрипту, що буде приймати інформацію.

Приклад реалізації відправки інформації авторизації користувача, для перевірки існування користувача, наведений на рисунку 4.26.

```
IEnumerator SendLoginForm()
{
    WWWForm loginForm = new WWWForm();
    loginForm.AddField("apppassword", "password");
    loginForm.AddField("email", emailInput.text);
    loginForm.AddField("password", passwordInput.text);
    using UnityWebRequest loginRequest = UnityWebRequest
        .Post("http://localhost:7777/oscilloscope/cruds/loginstudent.php", loginForm);
    yield return loginRequest.SendWebRequest();
    if (loginRequest.error == null)
    {
```

Рисунок 4.26 – Реалізація методу SendLoginForm класу LoginStudent для перевірки існування користувача в БД

Для виконання лабораторних робіт створено Prefab (заготовка) лабораторної (на рисунку 4.27 показаний Prefab для першої лабораторної), який буде створюватися при старті лабораторної в меню. Після виконання роботи, натискання кнопки «стоп» або після закінчення часу, на даному Prefab буде викликатися метод збору інформації з InputField (рис. 4.28), що розподілить інформацію до словника (Dictionary або словник – лист де у кожного значення є ключ за яким відбувається доступ до нього) та передасть до класу з методами обрахунку правильності введених даних.

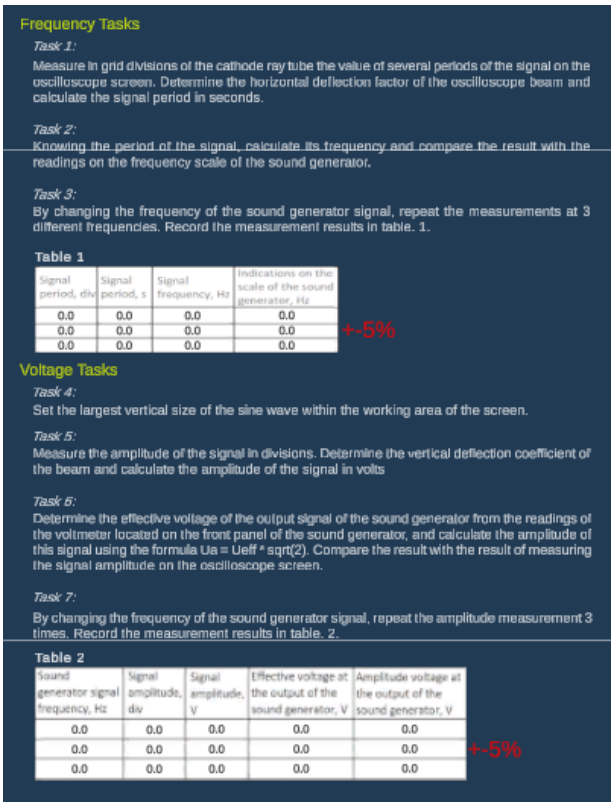

Рисунок 4.27 – Prefab до лабораторної №1

```
public static Dictionary<string, float> CollectInputInfo(GameObject gameObject)
{
    float currentValue = 0f;
    var labResultList = new Dictionary<string, float>();

    foreach (TMP_InputField inputField in gameObject.GetComponentsInChildren<TMP_InputField>())
    {
        if (float.TryParse(inputField.text, out float holder))
        {
            currentValue = holder;
        }

        labResultList.Add(inputField.gameObject.name, currentValue);
    }

    return labResultList;
}
```

Рисунок 4.28 – Реалізація збору інформації з Input Field в Prefab

Також за допомогою Prefab реалізовано відображення результатів лабораторних робіт користувача. Було створено окремий Prefab у вигляді UI елементу з текстовими полями для інформації та скриптом, що мав доступ до цих полів (рисунок 4.29).

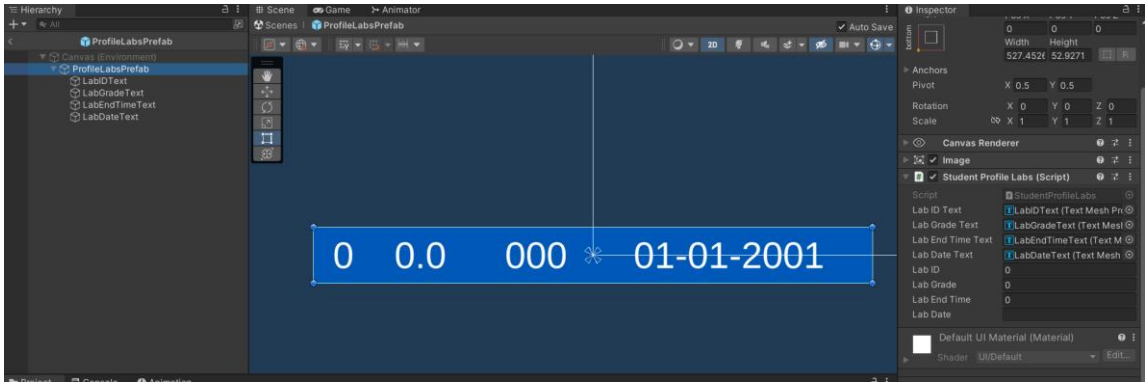

Рисунок 4.29 – Префаб результатів лабораторної роботи користувача

Даний Prefab передано до скрипту, що відповідає за заповнення профілю користувача ProfileScrollGroup.cs (рисунок 4.30), де за допомогою методу Instantiate створюється копія даного Prefab'у яка потім заповнюється інформацією про лабораторні користувача. Створена копія Prefab'у передається до елемента меню Scroll Group, що надає можливість переглядати багато елементів як список, що можна гортати (рис. 4.31).

```
var profileLab = Instantiate(profileLabsPrefab, new Vector3(0, 0, 0), Quaternion.identity);
profileLab.transform.SetParent(profilePanel.transform);
profileLab.GetComponent<StudentProfileLabs>().labID = item.labID;
profileLab.GetComponent<StudentProfileLabs>().labGrade = item.grade;
profileLab.GetComponent<StudentProfileLabs>().labEndTime = item.endTime;
profileLab.GetComponent<StudentProfileLabs>().labDate = item.date;
```

Рисунок 4.30 – Створення Prefab результатів лабораторної роботи та його заповнення інформацією

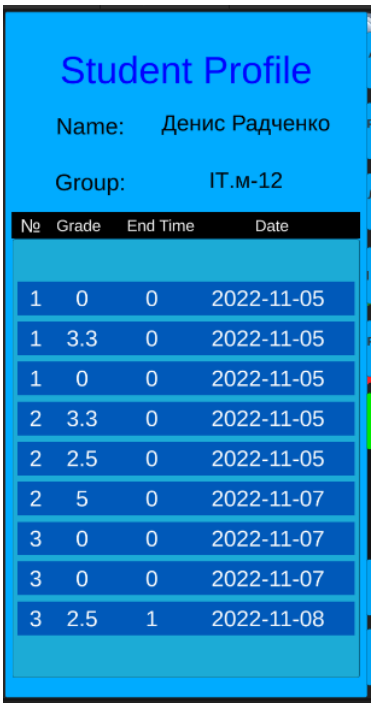

Рисунок 4.31 – Профіль користувача з інформацією та результатами робіт

Для збереження результатів роботи користувача використано стандартну бібліотеку C# IO (Input-Output), що надає змогу відкривати та зчитувати/записувати текстові файли, а також створювати їх.

Для цього використовується метод File.WriteAllText(path, text), куди надається шлях до файлу, та текст, який потрібно записати. Якщо такого файлу не існує, то він його створює.

Шаблон звіту однієї з лабораторних робіт наведено на рисунку 4.32.

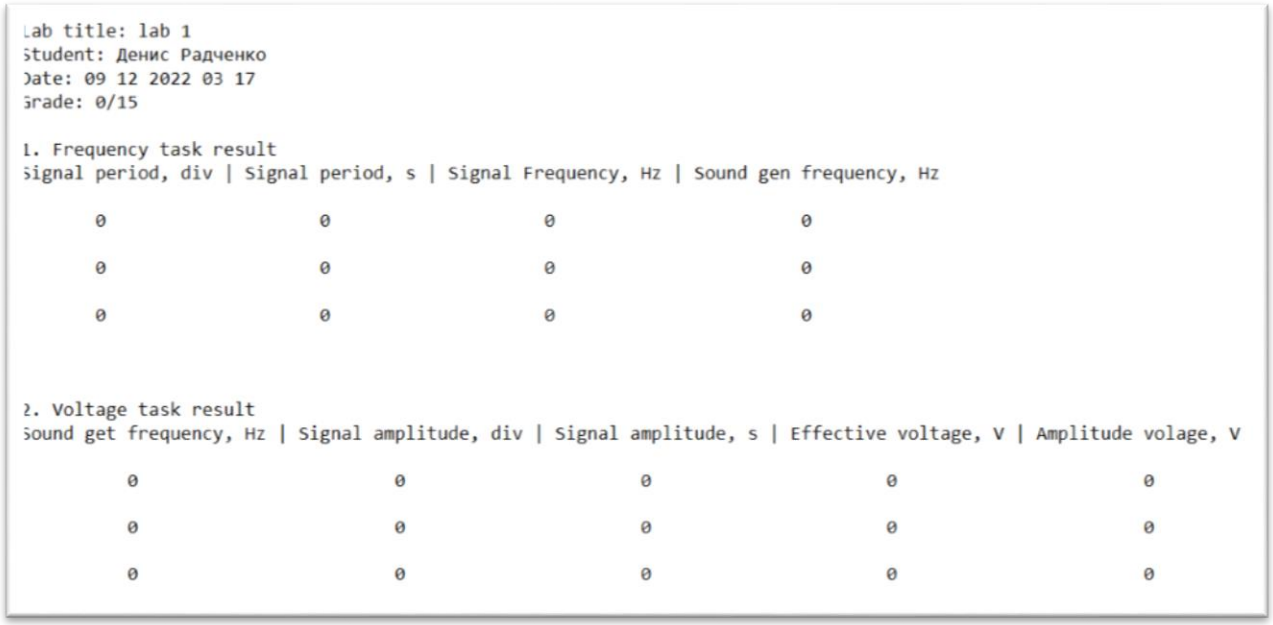

Рисунок 4.32 – Шаблон звіту з лабораторної роботи

4.4.2 Серверна частина (зв’язок з БД та обробка інформації)

Щоб додаток міг звернутися до скриптів на сервері, потрібно розмістити їх в шляху, який було вказано в налаштуваннях серверу MAMP. За замовчуванням це C:/MAMP/htdocs, де і будуть знаходитися скрипти, створені мовою PHP.

В кореневому файлі веб-серверу знаходитися папка з CRUD (Create, Update, Delete) операціями з БД, де в свою чергу знаходяться скрипти розширення .php, а саме:

– GetAllinfo.php – повертає всю статичну (що не зазнає змін) інформацію;

– LoginStudent.php – приймає дані для перевірки на існування користувача в БД;

– RegisterStudent.php – приймає, оброблює та зберігає дані нового користувача в БД;

– SaveStudentLab.php – приймає, оброблює та зберігає результати виконаної лабораторної роботи користувача;

– AddNewGroup.php – приймає назву групи та додає її до існуючих;

– ChangeGroupName.php – приймає та змінює назву існуючої групи;

– ChangeLabParams.php – приймає та замінює параметри лабораторних

робіт на нові;

– DeleteStudentLab.php – видаляє результати існуючої роботи, конкретного користувача.

Для підключення до БД з веб-серверних скриптів, в веб інтерфейсі MAMP є приклади з кодом для різних платформ, які підтримує веб-сервер MAMP.

Для прийому даних, які передаються з додатку, в php є асоціативна змінна $_POST[“string”] з ключем, яка може прийти з HTTP запитом. Тобто при кожному зверненні до скрипту, змінна буде перевіряти наявність такого ключа в HTTP формі, та забирати, якщо є.

Також потрібно подбати про безпеку БД, а саме про Script Injection. Script Injection це небажане втручання до запиту через конкатенацію рядків. Так як дані, що приходять скрипту, використовуються як параметр для SQL запиту, то цей запит можна розбити на два, де другий запит може бути яким завгодно, чи то для видалення всієї бази, чи то для доступу до приватної інформації. Для цього в php є метод filter_vat($var, FILTER) куди передається змінна на перевірку, та один з фільтрів, що замінює всі спец символи на код ASCII, який не інтерпретується компілятором.

На рисунку 4.33 зображена реалізація прийому даних користувача для авторизації з HTTP форми та створення за допомогою конкатенації SQL запиту, який перевіряє існування даного користувача та правильність паролю, з використанням отриманих даних.

Для захисту пароля користувача використано вбудовану функцію php кодування з різними ключами кодування password_hash, куди передається пароль та тип кодування. Так як пароль зберігається в закодованому вигляді в БД, для його перевірки на правильність із паролем, введеним користувачем, використовується функція password_verify, куди передається закодований пароль та пароль для перевірки.

```
$email = $_POST["email"];
$emailClean = filter_var($email, FILTER_SANITIZE_EMAIL);
$password = $_POST["password"];

$emailcheckquerry = "SELECT password
                     FROM students
                     WHERE email = '".$emailClean."';";
$emailcheckresult = mysqli_query($link, $emailcheckquerry) or die("2");

if ($emailcheckresult->num_rows != 1) {
    echo("3");
    exit();
}
else {
    $fetchedpassword = mysqli_fetch_assoc($emailcheckresult)["password"];
    if (password_verify(($password), $fetchedpassword)) {
```

Рисунок 4.33 – Отримання даних авторизації користувача та їх перевірка

За допомогою змінної $link, в яку поміщено з'єднання з БД, можна створювати SQL запити до бази даних за допомогою виклику вбудованого методу mysqli_query, куди передається підключення до БД та string змінна з SQL запитом. Якщо запит щось повертає, тобто використовує SELECT, то метод mysqli_query() потрібно викликати з присвоюванням до змінної, в яку потім і будуть додані дані. Якщо виникне помилка в SQL запиті, то для цього існує оператор or, який при помилці викличе функцію, вказану після оператора, в даному випадку це die($message), що просто завершить виконання скрипту. На рисунку 4.34 показана реалізація запиту з php коду до таблиці groups, виконання цього запиту та запис результатів запиту в змінну.

```
$getallgroups = "SELECT group_name FROM groups ORDER BY id ASC;";
$groupsresult = mysqli_query($link, $getallgroups) or die("2");
```

Рисунок 4.34 – Створення запиту з php коду до БД

Так як результати, що повертаються з запиту, мають свою структуру, її потрібно змінити в читабельний для php вигляд. Для цього є вбудований метод mysqli_fetch_assoc($result), куди передається результат запиту, дані розбиваються на масив рядків, який можна легко зчитати через цикл або за допомогою назви колонки в БД.

Для відправки даних до клієнта достатньо запакувати їх та використати звичайну функцію echo(), в яку передати запаковані дані. В даному випадку дані будуть запаковуватися в JSON формат за допомогою функції json_encode($array), куди передається масив даних. На рисунку 4.35 показане зчитування кожного рядку з запиту та додавання його як масиву до спільного масиву, який потім перетворюється в JSON формат та відправляється як відповідь клієнту.

```
$group_array = array();

while($group = mysqli_fetch_assoc($groupsresult)) {
    $group_array[] = $group;
}

$json_array = array('group_info' => $group_array);
echo json_encode($json_array);
```

Рисунок 4.35 – Обробка отриманих даних запиту та відправка їх клієнту

Дані, що були відправлені сервером до клієнту, в додатку отримуються за допомогою UnityWebRequest об’єкту та виклику на ньому downloadHandler.text, який повертає всю надану інформацію у вигляді рядка, що не дуже зручно для обробки даних. Так як сервер переформатує це все в JSON форму, що робить дані структурованими, їх можна зчитати за допомого різних бібліотек.

В даному випадку використовується стороння бібліотека simpleJSON, яка надає змогу парсувати рядок в JSON формат. Для цього створюється змінна JSONNode, на якій викликається JSON.Parse(text), куди передається рядок для парсування. Потім зі змінної JSONNode можна за допомогою foreach зчитати дані як з масиву (рисунок 4.36).

Також сервер може не виконати запит, якщо наприклад такого користувача не існує чи запит до БД викликав помилку. Для такого в метод die() передано числа, де кожне число відповідає за якусь помилку в скрипті. Після кожного запиту до серверу стоїть перевірка на всі ці помилки, і у випадку помилки вона буде оброблена та відображена користувачу. Код помилки та його пояснення для деяких скриптів наведені в таблиці 4.1.

```
JSONNode studentInfo = JSON.Parse(loginRequest.downloadHandler.text);
foreach (JSONNode labs in studentInfo["labs"])
{
    foreach (JSONNode lab in labs)
    {
        currentStudent.GetComponent<CurrentStudent>().AddLab(new StudentLabs(short.Parse(lab["lab_id"]),
                                                                              float.Parse(lab["grade"]),
                                                                              int.Parse(lab["end_time"]),
                                                                              lab["date"]));

    }
}
```

Рисунок 4.36 – Парсування даних лабораторних робіт в JSON формат з серверу та створення з них об’єкту лабораторної роботи

Таблиця 4.1 – Розшифрування коду помилки для серверних скриптів

	Пояснення помилки				
Скрипт / Код	GetAllInfo.php	LoginStudent.php	RegisterStudent.php	SaveStudentLab.php	GetAllStudents.php
1	Помилка при підключенні до БД				
2	Помилка при запиті до таблиці groups	Помилка при запиті до таблиці students	Помилка при запиті до таблиці students	Помилка додавання лабораторної до labs	Помилка при запиті до таблиці students
3	Помилка при запиті до таблиці labs	Такого студента не існує або їх більше ніж 1	Така пошта вже існує	-	В результаті запиту не має ніякої інформації
4	В результаті запиту не має ніякої інформації	Пароль не співпадають	Помилка при додаванні користувача до таблиці students	-	-
5	-	Помилка при запиті до таблиці students	-	-	-

4.4.3 Створення бази даних

Для створення бази даних використано вбудований в MAMP інтерфейс phpMyAdmin. Після запуску MAMP, відкриваємо myPhpAdmin в браузері, для цього в пошуку браузеру потрібно ввести:

IP:**Port**/phpMyAdmin/index.php

де IP та Port – IP та Port на якому розміщено сервер. В нашому випадку сервер розміщено на локальній машині, тобто IP - localhost, а порт було вказано вільний в MAMP – 7777.

При переході за посиланням відкривається головна сторінка phpMyAdmin. В ній потрібно ввести назву БД та обрати формат кодування. Кодування було обране стандартне, так як воно покриває розпізнавання майже всіх символів ASCII.

Назва бази даних обрана desktop_virtual_labs. Після створення бази даних відкривається вікно по створенню таблиць в ній (рис. 4.37), куди потрібно ввести ім’я таблиці та кількість стовбців (далі також можна буде або додати або видалити їх).

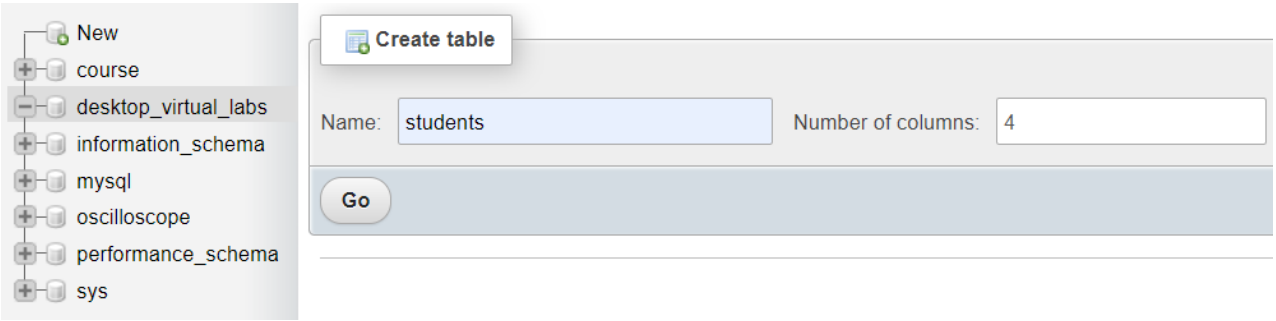

Рисунок 4.37 – Сторінка створення таблиці в обраній БД

Для початку створено головну таблицю користувачів (students). Після введення назви таблиці, кількості стовбців, відкриється сторінка створення/редагування характеристик (стовбців) таблиці (рис. 4.38).

На сторінці є можливість додавати нові стовбці, також можна вказати кількість доданих стовбців. Нижче знаходяться налаштування характеристик (стовпців), вони мають такі параметри:

– **Name** - ім’я;

– **Type** - тип;

– **Length/Values** - максимально допустима кількість символів;

– **Default** - стандартне значення при створенні (якщо воно не було вказано):

- Обране користувачем;
- Ніяке;
- Null;
- Дата, коли було додано дані.

– **Collation** - тип кодування;

– **Attributes** – атрибут, що відбудеться з рядком після зміни;

– **Null** – чи дозволений Null (поле без значення);

– **Index** – чи являється дане поле ключем:

- Primary – унікальний ключ, що не повторюється;
- Unique – унікальне значення;
- Index – значення, яке може мати дані тільки ті, які є стовпці іншої таблиці (тобто зв’язний ключ).

– **A_I** (Auto incrementing) – автоматичне збільшення на +1, доступне тільки для числових типів, можна не вказувати при додаванні рядку.

– **Comments** – коментар.

В таблиці 3.2 вже було визначено, які саме характеристики (стовбці) будуть первинним ключем, які зв’язним ключем, а які просто даними, і відповідно до цього в Index обрано відповідний параметр. При виставленні Primary, A_I автоматично відмічається. Таким чином створено й інші необхідні об'єкти (таблиці).

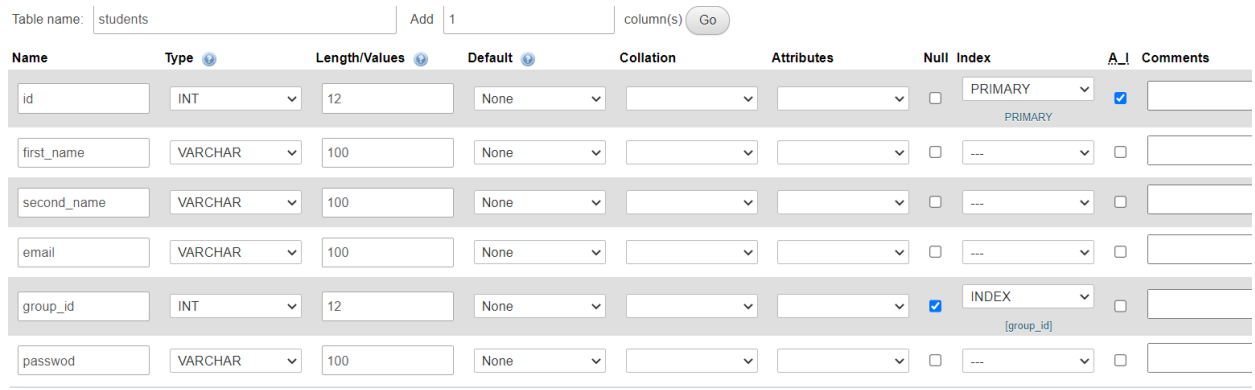

Рисунок 4.38 – Сторінка додавання/редагування характеристик до таблиці

Залишилося тільки зв'язати раніше створені ключі з відповідними таблицями за допомого вбудованого дизайнеру, де обирається ключ, який потрібно зв'язати, та поле, з яким буде зв'язано.

4.5 Тестування додатку

При тестуванні було перевірено два види функціоналу, для звичайного користувача та для адміністратора.

Перевірка функціоналу користувача:

- можливість роботи з осцилографом: зміна частоти та амплітуди, якість відображення сигналу;
- можливість реєстрації та подальша авторизація користувача;
- можливість вибору та виконання лабораторних робіт;
- коректне відображення даних користувача та результатів його робіт;
- коректна генерація звіту з роботи.

Перевірка функціоналу адміністратора:

- можливість авторизації як адміністратор;
- можливість додавання нових груп та зміни вже існуючих;
- можливість зміни параметрів лабораторних робіт (максимальний час на виконання, максимальна оцінка);

- коректне відображення результатів робіт всіх користувачів;
- можливість видалення результатів лабораторних робіт користувачів.

Під час перевірки, некоректної роботи виявлено не було. Все працює згідно функціональних вимог.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи проаналізовані питання використання віртуальних лабораторних робіт для покращення навчання студентів. Зазначено, що на сьогодні актуальність застосування тривимірних технологій в будь-якій зі сфер діяльності людей є високою. Для підвищення навичок студентів технічних спеціальностей буде доречним створення віртуальної лабораторної роботи з різними приладами.

Розглянуто варіанти реалізації та досліджено схожі аналоги віртуальних приладів, а саме осцилографа, щоб мати більше уявлення щодо правильності розробки віртуальної лабораторної роботи з осцилографом. В результаті сформульовані постановлення мети та задачі проєкту.

Визначені та обрані засоби і методи реалізації проекту. Ключовими програмними продуктами є Unity та Blender. На етапі проекту візуалізації осцилографа та сцени навколо було використано для запікання карт моделі та створення матеріалів Marmoset Tool та Substance painter відповідно.

Проведені етапи планування проєкту, деталізовано мету проєкту методом SMART. Також створено ієрархічну структуру робіт та організаційну схему проєкту. Розроблено календарний графік виконуваних робіт і досліджено ризики проєкту.

Етап проектування потребував проведення структурно-функціонального аналізу та створення діаграми сценаріїв використання. В результаті чого були розроблені діаграми: контекстна, першого рівня декомпозиції та Use Case.

Створено оптимізовану 3D модель осцилографу, максимально наближено де реального приладу. Програмно створено симуляцію роботи осцилографу з відображенням вхідного сигналу різної частоти та амплітуди зі змогою змінювати розгортку сигналу. Розроблено меню додатку зі сторінками авторизації, реєстрації, лабораторних робіт та результатів робіт.

Реалізовано клієнт-серверну архітектуру додатку, де дані з додатку

передаються до серверу, обробляються та зберігаються і навпаки.

У результаті створено додаток для виконання лабораторних робіт на віртуальному осцилографі.

Практичне значення додатку – підвищення якості навчання та закріплення практичних навичок у студентів фізико-електричних спеціальностей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лабораторія електрики: Постійний струм [Електронний ресурс] – Режим доступу до ресурсу: https://phet.colorado.edu/uk/simulations/circuit-construction-kit-dc

2. Жалдак М.І. Комп’ютер на уроках фізики: [навч.посіб.] / М.І. Жалдак, Д.К. Наборук, І.Л. Семещук. – Рівне: Тетіс, 2004. – 130 с.

3. Активізація пізнавального інтересу учнів за допомогою фізичного експерименту [Електронний ресурс] – Режим доступу до ресурсу : http://ur.co.ua/102/724-3-ativizaciya-poznavatel-noiy-deyatel-nosti-uchashihsya-posredstvom-fizicheskogo-eksperimenta.html

4. Єфименко Ю.О Комп'ютерний лабораторний практикум з цифрової техніки / Ю.О. Єфименко // Наукові записки Бердянського державного педагогічного університету. - 2014. - Вип. 3. - С. 85-90. - Режим доступу : http//bdpu.org/pedagogy/ua/files/2014/3/17.pdf

5. Electricity AR [Електронний ресурс] - Режим доступу до ресурсу : https://www.researchgate.net/publication/330511184_Electricity_AR

6. Інтерактивний сайт "Інтерактивні симуляції" PhET (Physucs Education Technology) [Електронний ресурс] - Режим доступу до ресурсу : https://phet.colorado.edu/uk/

7. Д. В. Мацокін, І. М. Пахомова. Платформи й мобільні додатки для створення та використання контенту із технологією доповненої реальності в освітньому процесі - Режим доступу до ресурсу : https://periodicals.karazin.ua/issuesedu/article/download/17672/16216/&usg=AOvVaw362QZuRFug2ps8WG3BA8po

8. Воронкін О.С. Віртуальний лабораторний практикум «Електричний струм». Наукові записки Малої академії наук України. Серія «Педагогічні науки» : [зб. наук. праць; редкол. : С.О. Довгий (голова), О.Є. Стрижак, О.В. Лісовий, І.М. Савченко та ін.]. – К. : 2018. – Вип. 13. – С. 38–59.

9. Мястковська М. О. Використання Phet-симуляцій для виконання домашніх завдань з фізики / М. О. МястковськаІ. М. Пшембаєв // Зб. наукових праць Кам'янець-Подільського національного університету ім. Івана Огієнка. - 2016. - Вип. 22. - С. 204-207. - Режим доступу до ресурсу : http://nbuv.gov.ua/UJRN/znpkp_ped_2016_22_66.

10. Гумен М. Б. Основи теорії процесів в інформаційних системах: підручник (у 2-х кн.). Кн.1. Аналіз детермінованих процесів / М. Б. Гумен, В. М. Співак, С. К. Мещанінов, Г. Г. Власюк, Т. Ф. Гумен. – 2-е вид., зі змінами і доповн. – Київ: Кафедра, 2017. − 281 с.

11. Теорія сигналів [Електронний ресурс] : навч. посіб. для студ. спеціальності 153 «Мікро- та наносистемна техніка» / КПІ ім. Ігоря Сікорського ; уклад.: А.О.Попов. – Електронні текстові данні (1 файл: 7399 Кбайт). – Київ : КПІ ім. ІгоряСікорського, 2019. – 268 с.

12. Осцилограф з сайту Academo [Електронний ресурс] - Режим доступу до ресурсу : https://academo.org/demos/virtual-oscilloscope/

13. Осцилограф з сайту physics-zone [Електронний ресурс] - Режим доступу до ресурсу : https://physics-zone.com/sim/virtual-oscilloscope/

14. Sulyman, Shakirat. (2014). Client-Server Model. IOSR Journal of Computer Engineering. 16. 57-71. 10.9790/0661-16195771. Режим доступу : https://www.researchgate.net/publication/271295146_Client-Server_Model

15. Dimitrios Serpanos, Tilman Wolf, In The Morgan Kaufmann Series in Computer Architecture and Design, Architecture of Network Systems, Morgan Kaufmann, 2011, Pages xvii-xviii, ISSN 15459888, ISBN 9780123744944. Режим доступу : https://doi.org/10.1016/B978-0-12-374494-4.00023-2. ((https://www.sciencedirect.com/science/article/pii/B9780123744944000244))

16. Chad Hollman. (2017). Modern Web Security Patterns [Електронний ресурс] - Режим доступу до ресурсу: https://owasp.org/www-chapter-sacramento/assets/slides/20200117-modern-web-security-patterns.pdf

17. Jan L. Harrington, Relational Database Design and Implementation (Fourth Edition), Morgan Kaufmann, 2016, Page iv, ISBN 9780128043998. Режим доступу :

https://doi.org/10.1016/B978-0-12-804399-8.00042-9. (https://www.sciencedirect.com/science/article/pii/B9780128043998000429)

18. IBM Cloud Education. (2019). Database Security. [Електронний ресурс] - Режим доступу до ресурсу : https://www.ibm.com/cloud/learn/database-security

19. Apache HTTP Server Documentation. [Електронний ресурс] - Режим доступу до ресурсу : https://httpd.apache.org/docs/

20. MySQL Documentation. [Електронний ресурс] - Режим доступу до ресурсу : https://dev.mysql.com/doc/

21. Suehring, Steve. PHP 6 and MySQL 6 Bible. — Wiley Publishing, 2010. — 912 с. — ISBN 978-5-8459-1640-2.

22. Jon Duckett, PHP & MySQL: Server-side Web Development, 2022, Pages 672. ISBN: 978-1-119-14921-7.

23. David Joseph Katz, 2022, Docker - Introducing Docker Essentials, Containers, and more. [Електронний ресурс] - Режим доступу до ресурсу : https://www.udemy.com/course/docker-containers/?utm_source=adwords& utm_medium=udemyads&utm_campaign=LongTail_la.EN_cc.ROW&utm_content=deal4584&utm_term=_._ag_77879424134_._ad_535397279649_._kw__._de_c_._dm__._pl__._ti_dsa-1007766171312_._li_1012864_._pd__._&matchtype=&gclid=Cj0KCQiAj4ecBhD3ARIsAM4Q_jE2177yfblEX8DrkW7wPRswnHH-v2bjX0ZSvlqux-fpP1err8nvHGYaAv-HEALw_wcB

24. Налаштування Apache, PHP, MySQL на WampServer. [Електронний ресурс] - Режим доступу до ресурсу : https://1cloud.ru/help/windows/ustanovka-nastroika-wampserver

25. Austin Patkos, 2021, Unity + SQL Databasesd Player Management Leaderboards + More! [Електронний ресурс] - Режим доступу до ресурсу : https://www.udemy.com/course/unity-sql-database/

26. Unreal Engine 5.1 Documentation. [Електронний ресурс] - Режим доступу до ресурсу : https://docs.unrealengine.com/5.1/en-US/

27. Unreal Engine 5.1 Documentation, Blueprints Visual Scripting. [Електронний ресурс] - Режим доступу до ресурсу :

https://docs.unrealengine.com/5.0/en-US/blueprints-visual-scripting-in-unreal-engine/

28. Unity Documentation, Unity User Manual 2021.3 (LTS). [Електронний ресурс] - Режим доступу до ресурсу : https://docs.unity3d.com/Manual/index.html

29. Blender Documentation. [Електронний ресурс] - Режим доступу до ресурсу : https://docs.blender.org/

30. Jan van den Hemel, 2020, Blender Secrets.

31. Marmoset Toolbag 4, Tutorials & Resources. [Електронний ресурс] – Режим доступу до ресурсу : https://marmoset.co/resources/

32. Nexttut Education Pvt.Ltd., 2021, Complete Guide to Marmoset 4. [Електронний ресурс] - Режим доступу до ресурсу : https://www.udemy.com/course/complete-guide-to-marmoset-toolbag-4/?utm_ source =adwords&utm_medium=udemyads&utm_campaign=LongTail_la.EN_cc.ROW&ut m_content=deal4584&utm_term=_._ag_77879424374_._ad_535397245869_._kw__. _de_c_._dm__._pl__._ti_dsa-1007766171592_._li_1012864_._pd__._ &matchtype =&gclid=Cj0KCQiAj4ecBhD3ARIsAM4Q_jHmWWy_5zp05THAy4vewp7LNtQlk3 vG8WCfxGOWadfNRp09hBzOx0caArGoEALw_wcB

33. Adobe, Substance 3D Painter (Documentation). [Електронний ресурс] - Режим доступу до ресурсу : https://substance3d.adobe.com/documentation/ spdoc/substance-3d-painter-20316164.html

34. Sean Fowler, 2022, Learn 3D Texturing in Substance Painter 2022 All Levels! [Електронний ресурс] - Режим доступу до ресурсу : https://www.udemy.com/course/learn-3d-texturing-in-substance-painter-2022-all-levels/

35. Ahmed, Fahim & Robinson, Stewart & Tako, A.A.. (2015). Using the structred analysis and design technique (SADT) in simulation conceptual modeling. Proceedings - Winter Simulation Conference. 2015. 1038-1049. 10.1109/WSC.2014.7019963.

36. IDEF0 [Електронний ресурс] – Режим доступу до ресурсу : https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6#p62

37. Діаграма варіантів використання (use case diagram) [Електронний

ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema12/tema12_2

38. Martin Spicuzza, 2020, Quaternions in Unity. [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@spicuzza157/quatern-b5cf7b83b1d1.

ДОДАТОК А ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація мети ІТ-проекту

Для визначення мети проекту, буде використано метод SMART (Specific, Measurable, Achievable, Relevant, Time-framed), що розбиває мету на головні складові проекту.

Таблиця А.1 – Ідентифікація мети методом SMART

Specific	Створити додаток для роботи з віртуальним осцилографом та виконанням лабораторних робіт на ньому
Measurable	Створити симуляцію виконання лабораторних робіт на осцилографі
Achievable	Створити інтерактивний додаток основі реально доступних ресурсних та функціональних можливостей
Relevant	Створити віртуальний осцилограф та змогу виконання та перегляду лабораторних робіт до нього
Time-Specific	Створення додатку у часі на основі сформованого календарного плану

А.2 Планування змісту структури робіт інформаційної системи

Визначимо перелік робіт для 4-рівневої декомпозиції проекту створення інтерактивного додатку віртуальної лабораторної роботи для роботи на осцилографі.

1. Створення додатку для виконання віртуальних лабораторних робіт на осцилографі

1.1. Аналіз предметної області.

1.1.1. Огляд проблеми віртуальних лабораторних робіт.

1.1.2. Дослідження існуючих рішень.

1.1.3. Постановка мети та задачі.

1.1.4. Вибір методів дослідження та інструментів реалізації.

1.2. Планування IT проекту.

1.2.1. Постановка задачі методом SMART.

1.2.2. Декомпозиція роботи.

1.2.2.1. Створення структурної діаграми робіт WBS.

1.2.2.2. Створення організаційної діаграми робіт OBS.

1.2.2.3. Створення матриці відповідальності.

1.2.2.4. Створення діаграми Ганта.

1.2.3. Управління ризиками.

1.3. Проектування робіт

1.3.1. Моделювання варіантів використання.

1.3.2. Документування проекту у нотаціях IDEF0 та IDEF3.

1.4. Моделювання бази даних.

1.4.1. Вибір схеми.

1.4.2. Побудова реляційної бази даних.

1.4.3. Створення бази даних.

1.5. Практична реалізація.

1.5.1. Створення 3D моделей.

1.5.1.1. Створення моделей приладів.

1.5.1.2. Запікання моделей.

1.5.1.3. Текстурування моделей.

1.5.1.4. Підготовка моделей до експорту в Unity

1.5.2. Програмна реалізація симуляції роботи осцилографу.

1.5.3. Реалізація архітектури клієнт-сервер.

1.5.3.1. Клієнтська частина (меню та взаємодія з сервером).

1.5.3.2. Серверна частина (зв’язок з БД та обробка інформації).

1.5.4. Тестування додатку

1.6. Здача роботи.

1.6.1. Представлення електронних файлів додатку.

1.6.2. Створення файлів документування.

1.6.3. Захист роботи.

WBS (Work Breakdown Structure) представлена на рисунку А.1.

OBS (Organization Breakdown Structure) – це ієрархічна структура управління проектом, що показує відношення між учасниками проекту.

Для побудови OBS потрібно визначити виконавців роботи та інших зацікавлених сторін проекту:

1. Виконавець (Радченко Д.Я.).
2. Керівник (Баранова І.В.).

OBS-структура організації робіт проекту наведена на рисунку А.2.

На основі WBS та OBS структур робіт було побудовано матрицю відповідальності, що показує виконавців елементів декомпозиції роботи.

Матрицю відповідальності проекту наведено у таблиці А.1.

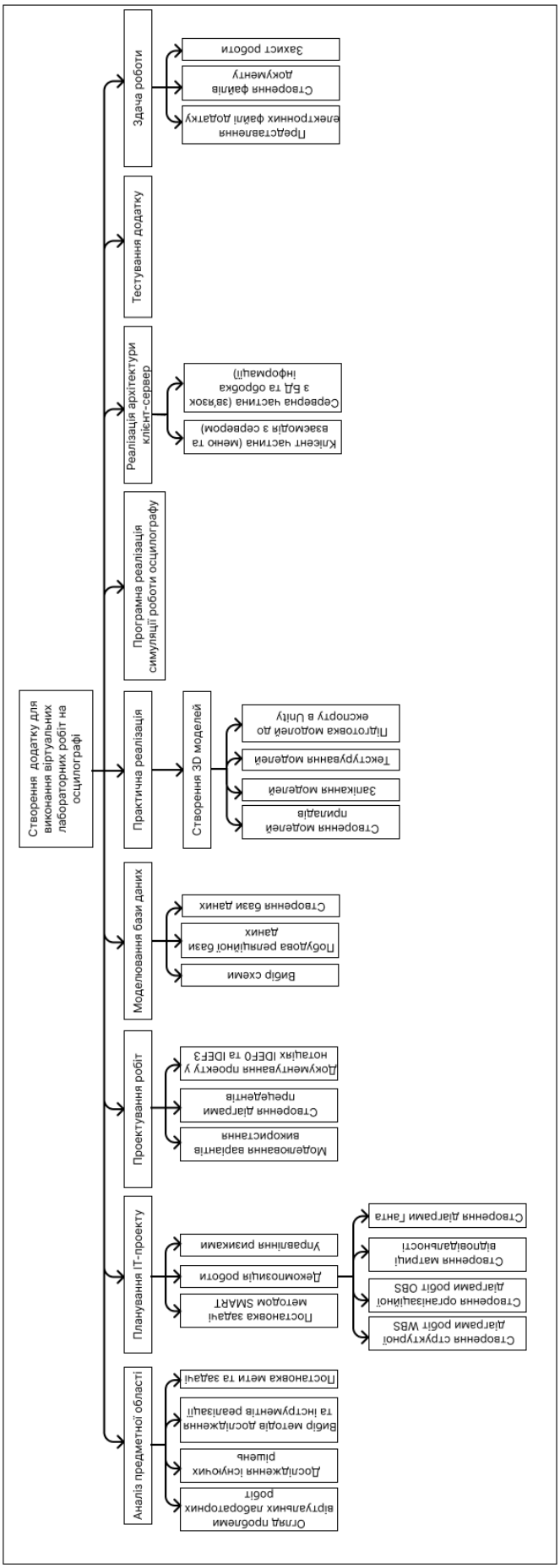

Рисунок А.1 – Діаграма декомпозиції робіт проекту (WBS)

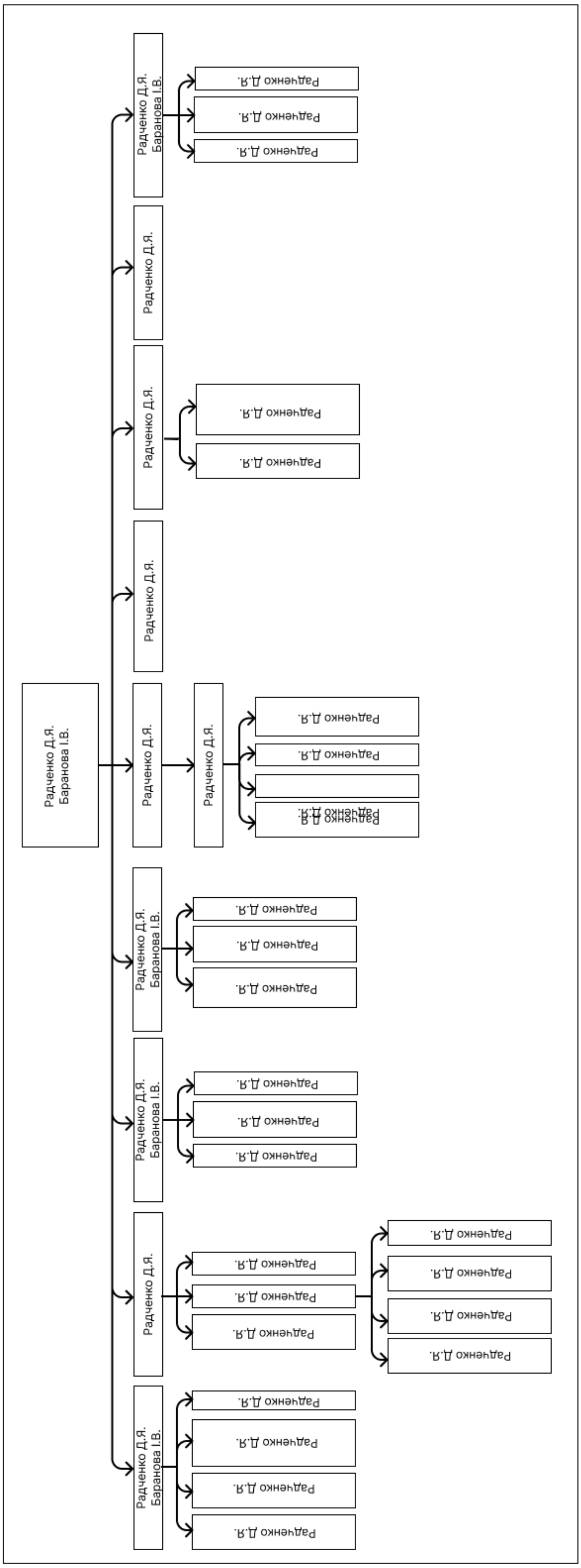

Рисунок А.2 – Діаграма організації робіт проекту (OBS)

Таблиця А.2 – Матриця відповідальності

Роль	**Виконавець**	**Керівник**
WBS/OBS	Радченко Д.Я.	Баранова І.В.
Створення додатку для виконання віртуальних лабораторних робіт на осцилографі		
Аналіз предметної області.		
Огляд проблеми віртуальних лабораторних робіт.		
Дослідження існуючих рішень.		
Постановка мети та задачі.		
Вибір методів дослідження та інструментів реалізації.		
Планування IT проекту.		
Постановка задачі методом SMART.		
Створення структурної діаграми робіт WBS.		
Створення організаційної діаграми робіт OBS.		
Створення матриці відповідальності.		
Створення діаграми Ганта.		
Управління ризиками.		
Проектування робіт		
Моделювання варіантів використання.		
Документування проекту у нотаціях IDEF0 та IDEF3.		
Моделювання бази даних.		
Вибір схеми. Побудова реляційної бази даних.		
Створення бази даних.		
Підготовка сцени.		
Створення 3D моделей.		
Створення моделей приладів.		
Запікання моделей.		
Текстурування моделей.		
Підготовка моделей до експорту в Unity		
Програмна реалізація симуляції роботи осцилографу.		
Клієнт-серверна реалізація.		
Клієнтська частина (меню та взаємодія з сервером).		
Серверна частина (зв’язок з БД та обробка інформації).		
Задача роботи.		
Представлення електронних файлів додатку.		
Створення файлів документування.		
Захист роботи.		

А.3 Побудова календарного графіку виконання інформаційної системи

Для розуміння тривалості виконання роботи з урахуванням обмеженості у використанні часових ресурсів, було створено на основі ієрархії проекту календарний графік робіт. Для цього було використано онлайн інструмент для побудови діаграми Ганта, що допомагає візуально відслідковувати виконання робіт.

Діаграма Ганта – горизонтальна лінійна діаграма, на якій задачі проекту представляються протяжними в часі відрізками, що характеризуються датами початку та закінчення, послідовністю, затримками та іншими часовими параметрами.

Діаграму Ганта для даного проекту на основі декомпозиції робіт та встановлених часових термінів представлено на рисунку А.3.

А.4 Планування ризиків проекту

Управління ризиками проекту – це процес виявлення, аналізу та реагування на будь-який ризик, який виникає протягом життєвого циклу проекту, щоб допомогти проекту залишатися на правильному шляху та досягти своєї мети. Управління ризиками є не лише реактивним; частиною процесу планування має бути визначення ризику, який може виникнути в проекті, і того, як контролювати цей ризик, якщо він справді має місце.

Для управління ризиками, дуже важливо почати з чіткого та точного визначення ризиків проекту, виходячи з уже готової декомпозиції проекту. Для цього побудуємо матрицю декомпозиції ризиків RBS (Risk Breakdown Structure) для нашого проекту (рисунок А.4).

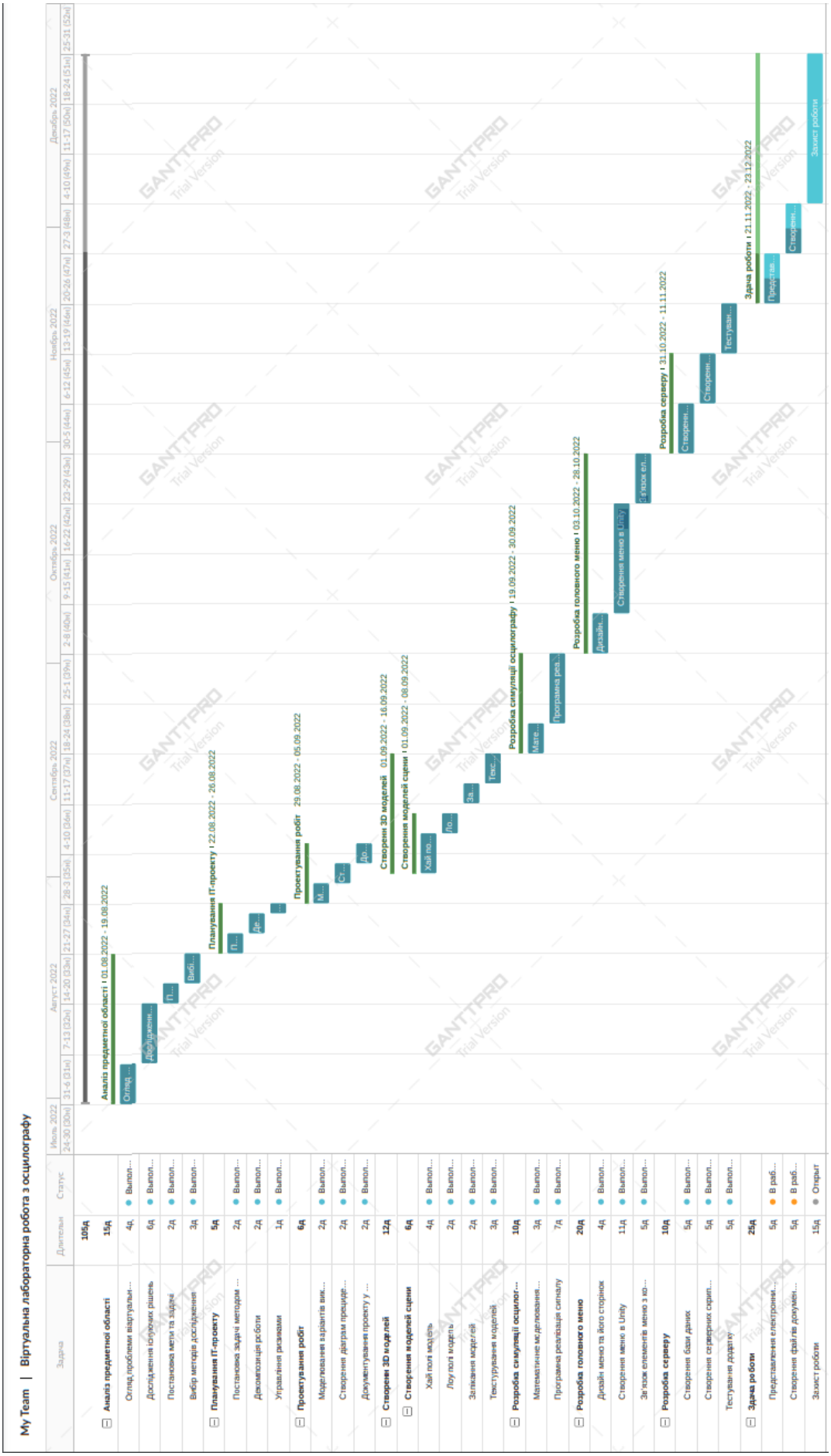

Рисунок А.3 – Діаграма Ганта

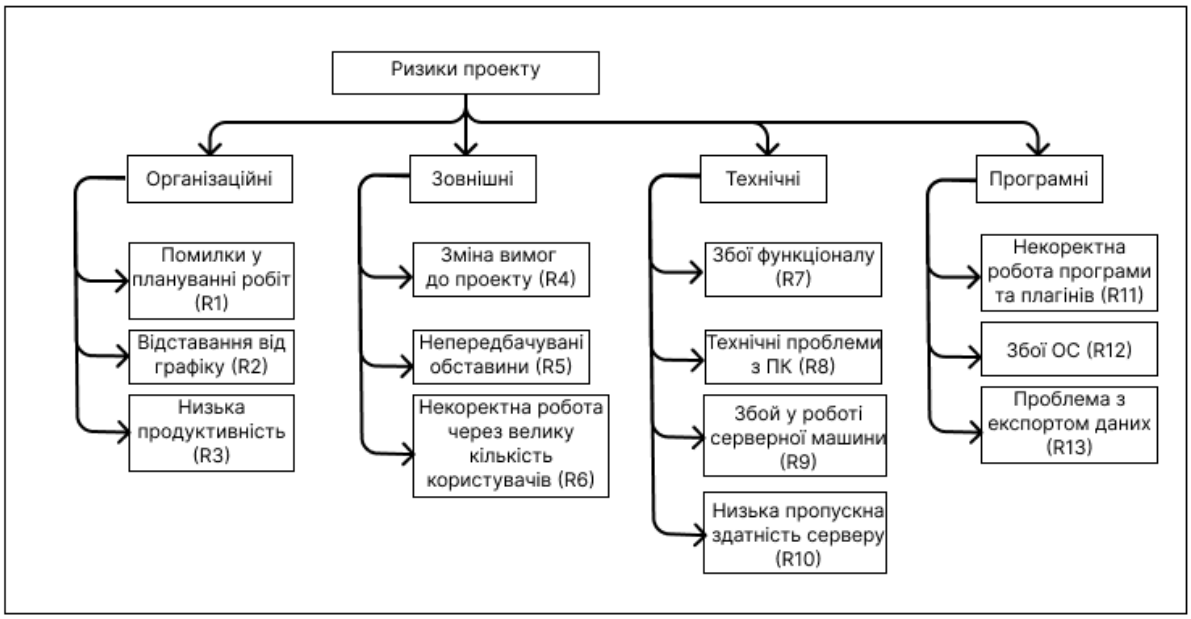

Рисунок А.4 – Матриця декомпозиції RBS

Для більш точного дослідження впливу ризиків на проект, можна охарактеризувати ризики за показниками ймовірності виникнення (таблиця А.3) та ступенями втрат (таблиця А.4).

Таблиця А.3 – Ймовірність виникнення ризиків проекту

Ступінь ймовірності		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
1	Слабо ймовірно				■				■		■	■	■	■
2	Мало ймовірно	■	■				■			■				
3	Ймовірно			■				■						
4	Доволі ймовірно					■								
5	Майже ймовірно													

Таблиця А.4 – Ступінь втрат ризиків проекту

Ступінь ймовірності		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
1	Мінімальна													
2	Низька													
3	Середня													
4	Висока													
5	Максимальна													

ДОДАТОК Б КОДИ МОДУЛІВ ДОДАТКА

Файл Sinwave.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using TMPro;

public class Sinwave : MonoBehaviour
{

    private const float TAU = 2 * Mathf.PI;
    private float x;
    private float y;
    private short isLine = 0;
    public LineRenderer myLineRenderer;
    public TMP_Text frequencyText;
    public TMP_Text amplitudeText;
    public int points;

    private float amplitude = 1;
    private float frequency = 1;
    private float timeSacaler = 1;
    private float voltageSacaler = 1;
    private float transformY = 0;
    private float transformX = 0;
    private float scalerTimeSacaler = 1;
    private float scalerVoltageSacaler = 1;
    private Vector2 xLimits = new Vector2(-5.5f,5.5f);

    // Start is called before the first frame update
    void Start()
    {
        myLineRenderer = GetComponent<LineRenderer>();
        updateText();
    }

    void Draw()
    {
        float xStart = xLimits.x;
        float xEnd = xLimits.y;

        myLineRenderer.positionCount = points;
        for (int currentPoint = 0; currentPoint < points; currentPoint++)
        {
            float progress = (float)currentPoint / (points - 1);
            this.x = Mathf.Lerp(xStart, xEnd, progress);
            this.y = calculateY();
            myLineRenderer.SetPosition(currentPoint, new Vector3(this.x, this.y, 0));
```

```
        }
    }

    private float calculateY()
    {
        if (this.isLine == 0)
        {
            //this.points = 2;
            return this.transformY;
        }
        else
        {
            //this.points = 500;
            return (((this.voltageSacaler * this.amplitude) * this.scalerVoltageSacaler) * Mathf.Cos(((TAU /
((this.frequency * this.timeSacaler) * this.scalerTimeSacaler)) * this.x) + this.transformX)) + this.transformY;
        }
    }
    public void setY(short sigh)
    {
        if (sigh < 0)
        {
            this.isLine -= 1;
        }
        else if (sigh > 0)
        {
            this.isLine += 1;
        }
    }

    public void plusAmplitude()
    {
        this.amplitude = RoundNum(this.amplitude + (float) 0.1);
        //Debug.Log("amplitude " + amplitude);
    }
    public void plusFrequency()
    {
        this.frequency = RoundNum(this.frequency + (float) 0.01);
        //Debug.Log("frequency " + frequency);
    }

    public void minusAmplitude()
    {
        this.amplitude = RoundNum(this.amplitude - (float) 0.1);
        //Debug.Log("amplitude " + amplitude);
    }

    public void minusFrequency()
    {
        this.frequency = RoundNum(this.frequency - (float) 0.01);
        //Debug.Log("frequency " + frequency);
    }

    public void updateText()
    {
```

```
    frequencyText.text = frequency.ToString();
    amplitudeText.text = amplitude.ToString();
}

public void setTimeScaler(float scaler)
{
    this.timeSacaler = scaler;
    //Debug.Log("timeSacaler " + timeSacaler);
}

public void setVoltageScaler(float scaler)
{
    this.voltageSacaler = scaler;
    //Debug.Log("voltageSacaler " + voltageSacaler);
}

private float RoundNum(float number)
{
    return Mathf.Round(number * 100f) / 100f;
}

public void setYTransform(float transform)
{
    this.transformY += transform;
    this.transformY = Mathf.Clamp(this.transformY, -5, 5);
    //Debug.Log("transformY " + this.transformY);
}

public void setXTransform(float transform)
{
    this.transformX += transform;
    //ebug.Log("transformX " + this.transformX);
}

public void multiplyTime(bool isMultiplied)
{
    if (isMultiplied)
    {
        this.scalerTimeSacaler *= (float) 0.2;
    }
    else
    {
        this.scalerTimeSacaler /= (float) 0.2;
    }
    //Debug.Log("timeSacaler " + this.timeSacaler);
}

public void multipleVoltage(bool isMultiplied)
{
    if (isMultiplied)
    {
        this.scalerVoltageSacaler *= (float) 10;
    }
    else
```

```
        {
            this.scalerVoltageSacaler /= (float) 10;
        }
        //Debug.Log("voltageSacaler " + this.voltageSacaler);
    }

    // Update is called once per frame
    void Update()
    {
        //Debug.Log();
        Draw();
    }
}
```

Файл LoginStudent.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;
using TMPro;
using SimpleJSON;


public class LoginStudent : MonoBehaviour
{
    public TMP_InputField emailInput;
    public TMP_InputField passwordInput;

    public Button loginButton;
    public TMP_Text loginButtonText;

    public GameObject currentStudentObject;
    //public GameObject currentStudentLabObject;

    private void Awake()
    {
        var currentStudent = GameObject.FindGameObjectsWithTag("CurrentStudent");
        foreach (var item in currentStudent)
        {
            Destroy(item);
        }
    }
    public void Login()
    {
        loginButton.interactable = false;
        loginButtonText.text = "Sending...";
        if (emailInput.text.Length < 3)
        {
            ErrorOnLoginMessage("Check Username");
        }
```

```
    else if (passwordInput.text.Length < 3)
    {
        ErrorOnLoginMessage("Check Password");
    }
    else
    {
        StartCoroutine(SendLoginForm());
    }
}

public void ErrorOnLoginMessage(string message)
{
    PopupWindow.Instance.SetContent(message)
        .AddOkayAction()
        .Show();
}

public void ResetLoginButton()
{
    loginButton.GetComponent<Image>().color = Color.white;
    loginButtonText.text = "Login";
    loginButtonText.fontSize = 35;
    loginButton.interactable = true;
}

public void SetButtonSuccess()
{
    emailInput.interactable = false;
    passwordInput.interactable = false;
    loginButton.GetComponent<Image>().color = Color.green;
    loginButtonText.text = "Logged in";
    loginButtonText.fontSize = 35;
    loginButton.interactable = false;
}

public void ResetInputFields()
{
    emailInput.interactable = true;
    passwordInput.interactable = true;
    emailInput.text = "";
    passwordInput.text = "";
}

IEnumerator SendLoginForm()
{
    WWWForm loginForm = new WWWForm();
    loginForm.AddField("apppassword", "password");
    loginForm.AddField("email", emailInput.text);
    loginForm.AddField("password", passwordInput.text);
    using UnityWebRequest loginRequest = UnityWebRequest
        .Post("http://localhost:7777/oscilloscope/cruds/loginstudent.php", loginForm);
    yield return loginRequest.SendWebRequest();
    if (loginRequest.error == null)
    {
```

```
        string response = loginRequest.downloadHandler.text;
        //Debug.Log(response);
        if (response == "1" || response == "2" || response == "5" || response == "6")
        {
            ErrorOnLoginMessage("Server Error");
        }
        else if (response == "3")
        {
            ErrorOnLoginMessage("Email didnt exist");
        }
        else if (response == "4")
        {
            ErrorOnLoginMessage("Wrong Password");
        }
        else
        {
            var currentStudent = Instantiate(currentStudentObject, new Vector3(0, 0, 0), Quaternion.identity);

            JSONNode studentInfo = JSON.Parse(loginRequest.downloadHandler.text);

currentStudent.GetComponent<CurrentStudent>().AssigneInfo(studentInfo["student_info"]["first_name"],
                                                studentInfo["student_info"]["second_name"],
                                                studentInfo["student_info"]["email"],
                                                studentInfo["student_info"]["group_name"]);
            if (!studentInfo["labs"].IsNull)
            {
                //Debug.Log(studentInfo["labs"]);
                foreach (JSONNode labs in studentInfo["labs"])
                {
                    foreach (JSONNode lab in labs)
                    {
                        currentStudent.GetComponent<CurrentStudent>().AddLab(new
StudentLabs(short.Parse(lab["lab_id"]),
                                                                float.Parse(lab["grade"]),
                                                                int.Parse(lab["end_time"]),
                                                                lab["date"]));
                    }
                }
            }
            FindObjectOfType<ProfileScrollGroup>().CreateProfile();
            FindObjectOfType<TabGroup>().UnblockPagesButtons();

            if (currentStudent.GetComponent<CurrentStudent>().email.Equals("admin"))
            {
                FindObjectOfType<AdminFunctionality>().ActivateAdminFunc();
                FindObjectOfType<StudentInfoScrollGroup>().ActivateAdminFunc();
            }

            FindObjectOfType<StudentInfoScrollGroup>().GetAllStudentsInfo();

            SetButtonSuccess();
        }
    }
    else
```

```
        {
            Debug.Log(loginRequest.error);
        }
    }
}
```

Файл RegisterStudent.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;
using TMPro;
using SimpleJSON;
using System.Text.RegularExpressions;

public class RegisterStudent : MonoBehaviour
{
    public TMP_InputField firstnameInput;
    public TMP_InputField secondnameInput;
    public TMP_InputField emailInput;
    public TMP_InputField passwordInput;
    private string groupName;

    public Button registerButton;
    public TMP_Text registerButtonText;

    public TMP_Dropdown groupDropdown;

    private string namePattern = @"^[\p{L} ,.'-]+$";
    private string emailPattern = @"^(?(""")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-
!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
                          @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-9]{2,17}))$";

    public void RegisterNewStudent()
    {
        registerButton.interactable = false;
        if ((firstnameInput.text.Length < 2 || secondnameInput.text.Length < 2)
            || !Regex.IsMatch(firstnameInput.text, namePattern, RegexOptions.IgnoreCase))
        {
            ErrorMessage("Check name fields");
        }
        else if (emailInput.text.Length < 5
            || !Regex.IsMatch(emailInput.text, emailPattern, RegexOptions.IgnoreCase))
        {
            ErrorMessage("Check email field");
        }
        else if (passwordInput.text.Length < 5)
        {
            ErrorMessage("Password too short");
        }
        else
        {
```

```
            SetButtonToSending();
            StartCoroutine(CreatePlayerPostRequest());
        }
    }

    public void ErrorMessage(string message)
    {
        PopupWindow.Instance.SetContent(message)
            .AddOkayAction()
            .Show();
    }

    public void ResetRegisterButton()
    {
        registerButton.GetComponent<Image>().color = Color.white;
        registerButtonText.text = "Register";
        registerButtonText.fontSize = 35;
        registerButton.interactable = true;
    }

    public void SetButtonToSending()
    {
        registerButton.GetComponent<Image>().color = Color.gray;
        registerButtonText.text = "Sending...";
        registerButtonText.fontSize = 35;
    }

    public void SetButtonSuccess()
    {
        registerButton.GetComponent<Image>().color = Color.green;
        registerButtonText.text = "Success";
        registerButtonText.fontSize = 35;
    }

    IEnumerator CreatePlayerPostRequest()
    {
        groupName = groupDropdown.options[groupDropdown.value].text;

        WWWForm registerForm = new WWWForm();
        registerForm.AddField("apppassword", "password");
        registerForm.AddField("firstname", firstnameInput.text);
        registerForm.AddField("secondname", secondnameInput.text);
        registerForm.AddField("email", emailInput.text);
        registerForm.AddField("group", groupName);
        registerForm.AddField("password", passwordInput.text);
        using UnityWebRequest registerRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/registerstudent.php", registerForm);
        yield return registerRequest.SendWebRequest();
        if (registerRequest.error == null)
        {
            Debug.Log(registerRequest.downloadHandler.text);
            string response = registerRequest.downloadHandler.text;

            if (response == "1" || response == "2" || response == "4")
```

```
            {
                ErrorMessage("Server Error");
            }
            else if (response == "3")
            {
                ErrorMessage("Email is already exist");
            }
            else
            {
                SetButtonSuccess();
            }
        }
        else
        {
            Debug.Log(registerRequest.error);
        }
    }
}
```

Файл SaveStudentLab.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

public class SaveStudentLab : MonoBehaviour
{
    private StudentLabs studentLab;
    private string email;

    public void Save(StudentLabs studentLab, string email)
    {
        this.studentLab = studentLab;
        this.email = email;

        StartCoroutine(SendSaveForm());
    }

    IEnumerator SendSaveForm()
    {
        WWWForm saveForm = new WWWForm();
        saveForm.AddField("apppassword", "password");
        saveForm.AddField("email", email);
        saveForm.AddField("labID", studentLab.labID.ToString());
        saveForm.AddField("grade", studentLab.grade.ToString());
        saveForm.AddField("endTime", studentLab.endTime.ToString());
        saveForm.AddField("date", studentLab.date.ToString());
        using UnityWebRequest saveRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/savestudentlab.php", saveForm);
        yield return saveRequest.SendWebRequest();
        if (saveRequest.error == null)
        {
            string response = saveRequest.downloadHandler.text;
```

```
            Debug.Log(response);
            if (response == "1" || response == "2")
            {
                Debug.Log(response);
            }
            else
            {
                Debug.Log("OK");
            }
        }
    }
}
```

Файл AdminFunctionality.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.Networking;
using UnityEngine.UI;
using Unity.VisualScripting;

public class AdminFunctionality : MonoBehaviour
{
    public TMP_InputField newGradeInput;
    public TMP_InputField newTimeInput;
    public TMP_InputField newGroupNameInput;
    public Button labChangeApply;
    public Button groupChangeApply;
    public TMP_Dropdown groupDropdown;
    public TMP_Dropdown labDropdown;

    public GameObject adminPanel;

    private void Start()
    {
        adminPanel.SetActive(false);
        labChangeApply.interactable = false;
        groupDropdown.onValueChanged.AddListener(delegate { OnGroupDropdownChange(); });
        labDropdown.onValueChanged.AddListener(delegate { OnLabDropdownChange(); });
    }

    public void ActivateAdminFunc()
    {
        adminPanel.SetActive(true);
    }

    public void OnLabDropdownChange()
    {
        labChangeApply.interactable = true;
        if (labDropdown.value == 0)
        {
            labChangeApply.interactable = false;
```

```
            newGradeInput.text = "";
            newTimeInput.text = "";
        }

        LabInfo labInfo =
FindObjectOfType<GetAllInfo>().getLabList()[labDropdown.options[labDropdown.value].text];
        newGradeInput.text = labInfo.maxGrade.ToString();
        newTimeInput.text = labInfo.maxTime.ToString();

    }
    public void OnGroupDropdownChange()
    {
        if (groupDropdown.value == 0)
        {
            newGroupNameInput.text = "";
        }
        else
        {
            newGroupNameInput.text = groupDropdown.options[groupDropdown.value].text;
        }
    }

    public void DeleteStudentLab(int labID)
    {
        StartCoroutine(SendDeleteLabRequest(labID));
    }
    public void ChangeLabParams()
    {
        bool gradeParse = int.TryParse(newGradeInput.text, out int newGrade);
        bool timeParse = int.TryParse(newTimeInput.text, out int newTime);

        if (gradeParse && timeParse)
        {
            if (newGrade > 1 && newTime > 1)
            {
                Debug.Log("test");
                StartCoroutine(SendChangeLabParamsRequest());
            }
        }
    }

    public void AddOrChangeGroup()
    {
        if (groupDropdown.value == 0)
        {
            StartCoroutine(SendAddGroupRequest());
        }
        else
        {
            StartCoroutine(SendChangeGroupRequest());
        }
    }

    IEnumerator SendDeleteLabRequest(int labID)
```

```
    {
        WWWForm deleteForm = new WWWForm();
        deleteForm.AddField("apppassword", "password");
        deleteForm.AddField("labID", labID);
        using UnityWebRequest deleteRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/deletestudentlab.php", deleteForm);
        yield return deleteRequest.SendWebRequest();
        if (deleteRequest.error == null)
        {
            string response = deleteRequest.downloadHandler.text;

            if (response == "1" || response == "2")
            {
                Debug.Log("Server Error");
            }
            else if (response == "0")
            {
                Debug.Log("Ok");
            }
        }
    }

    IEnumerator SendChangeLabParamsRequest()
    {
        WWWForm labForm = new WWWForm();
        labForm.AddField("apppassword", "password");
        labForm.AddField("labName", labDropdown.options[labDropdown.value].text);
        labForm.AddField("newGrade", newGradeInput.text);
        labForm.AddField("newTime", newTimeInput.text);
        using UnityWebRequest labRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/changelabparams.php", labForm);
        yield return labRequest.SendWebRequest();
        if (labRequest.error == null)
        {
            string response = labRequest.downloadHandler.text;

            if (response == "1" || response == "2")
            {
                Debug.Log("Server Error");
            }
            else if (response == "0")
            {
                Debug.Log("Ok");
            }
        }
    }

    IEnumerator SendAddGroupRequest()
    {
        WWWForm gruopForm = new WWWForm();
        gruopForm.AddField("apppassword", "password");
        gruopForm.AddField("groupName", newGroupNameInput.text);
        using UnityWebRequest groupRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/addnewgroup.php", gruopForm);
```

```
        yield return groupRequest.SendWebRequest();
        if (groupRequest.error == null)
        {
            string response = groupRequest.downloadHandler.text;

            if (response == "1" || response == "2" || response == "4")
            {
                Debug.Log("Server Error");
            }
            else if (response == "4")
            {
                Debug.Log("This group is allready exist");
            }
            else if (response == "0")
            {
                Debug.Log("Ok");
            }
        }
    }

    IEnumerator SendChangeGroupRequest()
    {
        WWWForm gruopForm = new WWWForm();
        gruopForm.AddField("apppassword", "password");
        gruopForm.AddField("groupName", groupDropdown.options[groupDropdown.value].text);
        gruopForm.AddField("newGroupName", newGroupNameInput.text);
        using UnityWebRequest groupRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/changegroupname.php", gruopForm);
        yield return groupRequest.SendWebRequest();
        if (groupRequest.error == null)
        {
            string response = groupRequest.downloadHandler.text;

            if (response == "1" || response == "2" || response == "4")
            {
                Debug.Log("Server Error");
            }
            else if (response == "4")
            {
                Debug.Log("This group didnt exist");
            }
            else if (response == "0")
            {
                Debug.Log("Ok");
            }
        }
    }
}
```

Файл Buttons.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using System;

public class Buttons : MonoBehaviour
{
    private bool isPressed;
    private float timeScale = 1.0f;
    private float time = 0.1f;

    // private int currentTimePos = 2; //Start position of swither is 1 ms. 1 period per cell = 1000 Hz (1 kHz)

    public int buttonId;

    private Sinwave sinwave;
    [SerializeField] LineRenderer lineRenderer;

    private void Awake()
    {
        sinwave = lineRenderer.GetComponent<Sinwave>();
    }

    void OnMouseDown()
    {
        this.isPressed = true;
        StartCoroutine(DoButtonsCheck());
    }

    private void OnMouseUp()
    {
        this.isPressed = false;
        time = 0.1f;
        timeScale = 1.0f;
    }

    IEnumerator DoButtonsCheck()
    {
        if (this.isPressed)
        {
            while (this.isPressed)
            {
                if (timeScale <= 2)
                {
                    timeScale += 0.0025f;
                    time /= timeScale;
                }

                if (buttonId == 1)
                {
                    sinwave.plusAmplitude();
                    sinwave.updateText();
                }
                else if (buttonId == 2)
                {
                    sinwave.minusAmplitude();
                    sinwave.updateText();
```

```
            }
            else if (buttonId == 3)
            {
                sinwave.plusFrequency();
                sinwave.updateText();
            }
            else if (buttonId == 4)
            {
                sinwave.minusFrequency();
                sinwave.updateText();
            }
            yield return new WaitForSeconds(time);
        }
    }
    yield return new WaitForSeconds(0.0f);
  }
}
```

Файл CameraMovement.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraMovement : MonoBehaviour
{
    private Vector2 turn;
    public float sensitivity = .5f;
    // Start is called before the first frame update
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
    }

    // Update is called once per frame
    void Update()
    {
        turn.x += Input.GetAxis("Mouse X") * sensitivity;
        turn.y += Input.GetAxis("Mouse Y") * sensitivity;
        transform.localRotation = Quaternion.Euler(-turn.y, turn.x, 0);
    }

    public void Lock()
    {
        this.sensitivity = 0;
    }

    public void Unlock()
    {
        this.sensitivity = 1.5f;
    }
}
```

Файл CurrentStudent.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CurrentStudent : MonoBehaviour
{
    public string firstName;
    public string secondName;
    public string email;
    public string group;

    private List<StudentLabs> studentLabs = new List<StudentLabs>();

    public void AssigneInfo(string firstName, string secondName, string email, string group)
    {
        this.firstName = firstName;
        this.secondName = secondName;
        this.email = email;
        this.group = group;
    }

    private void Awake()
    {
        var students = FindObjectsOfType<CurrentStudent>();
        if (students.Length > 1)
        {
            Destroy(gameObject);
        }
        DontDestroyOnLoad(gameObject);
    }

    public void AddLab(StudentLabs studentLab)
    {
        studentLabs.Add(studentLab);
    }

    public List<StudentLabs> GetStudentLabs()
    {
        return studentLabs;
    }

    public bool IsEnoughTries(int labID)
    {
        int tries = 1;

        foreach (var item in studentLabs)
        {
            if (tries == 3)
            {
                return false;
            }
```

```
            if (item.labID == labID)
            {
                tries++;
            }
            //Debug.Log(tries);
        }

        return true;
    }
}
```

Файл DropdownManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using static UnityEditor.Progress;
using UnityEngine.Events;

public class DropdownManager : MonoBehaviour
{
    private TMP_Dropdown dropdown;
    private List<string> infoList = new List<string>();
    private string currentDropdownItem;
    private int currentDropdownIndex;

    public bool isLabsOrGroup;
    public bool includeNone;
    public string noneText = "None";

    void Start()
    {
        dropdown = GetComponentInChildren<TMP_Dropdown>();

        if (isLabsOrGroup)
        {
            Dictionary<string, LabInfo> labList = FindObjectOfType<GetAllInfo>().getLabList();

            foreach (var labInfo in labList)
            {
                infoList.Add(labInfo.Value.labName);
            }
        }
        else
        {
            infoList = FindObjectOfType<GetAllInfo>().getGroupList();
        }

        dropdown.options.Clear();

        if (includeNone)
        {
            dropdown.options.Add(new TMP_Dropdown.OptionData() { text = noneText });
```

```
        }

        foreach (string item in infoList)
        {
            dropdown.options.Add(new TMP_Dropdown.OptionData() { text = item });
        }

        DropdownItemSelected(dropdown);

        dropdown.onValueChanged.AddListener(delegate { DropdownItemSelected(dropdown); });
    }

    void DropdownItemSelected(TMP_Dropdown dropdown)
    {
        int index = dropdown.value;
        currentDropdownItem = dropdown.options[index].text;
        //Debug.Log(index);
        //Debug.Log(currentDropdownItem);
    }
}
```

Файл GameControls.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class GameControls : MonoBehaviour
{
    public bool gameIsPaused;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            FindObjectOfType<SideMenu>().ShowHide();
            gameIsPaused = !gameIsPaused;
            PauseGame();
        }
    }
    public void PauseGame()
    {
        if (gameIsPaused)
        {
            FindObjectOfType<CameraMovement>().Lock();
            Cursor.lockState = CursorLockMode.Confined;
            //Time.timeScale = 0f;
        }
        else
        {
            FindObjectOfType<CameraMovement>().Unlock();
```

```
            Cursor.lockState = CursorLockMode.Locked;
            //Time.timeScale = 1;
        }
    }

    public void doExitGame()
    {
        PopupWindow.Instance.SetContent("Do you really want to exit?")
            .AddAcceptDeclineAction(Application.Quit)
            .SetAcceptDeclineButtonText("Yes", "No")
            .Show();
    }

    public void Test()
    {
        Debug.Log("Exit");
    }
}
```

Файл GetAllInfo.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;
using TMPro;
using SimpleJSON;
using UnityEngine.EventSystems;

public class GetAllInfo : MonoBehaviour
{
    private List<string> groupList = new();
    private Dictionary<string, LabInfo> labList = new();

    void Awake()
    {
        StartCoroutine(GetAllInfoRequest());
    }

    public List<string> getGroupList()
    {
        return groupList;
    }

    public Dictionary<string, LabInfo> getLabList()
    {
        return labList;
    }

    IEnumerator GetAllInfoRequest()
    {
        WWWForm wWWForm = new WWWForm();
        wWWForm.AddField("apppassword", "password");
```

```
        using UnityWebRequest getGroupsRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/cruds/getallinfo.php", wWWForm);
        yield return getGroupsRequest.SendWebRequest();
        if (getGroupsRequest.error == null)
        {
            //Debug.Log(getGroupsRequest.downloadHandler.text);
            string response = getGroupsRequest.downloadHandler.text;

            if (response == "1" || response == "2" || response == "3" || response == "4")
            {
                Debug.Log("Server Error");
            }
            else
            {
                groupList.Clear();
                labList.Clear();

                JSONNode allInfo = JSON.Parse(getGroupsRequest.downloadHandler.text);

                foreach (JSONNode group in allInfo["group_info"])
                {
                    groupList.Add(group[0]);
                }

                foreach (JSONNode lab in allInfo["labs_info"])
                {
                    labList.Add(lab["lab_name"], new LabInfo(lab["lab_name"], int.Parse(lab["max_time"]),
int.Parse(lab["max_grade"])));
                }
            }
        }
        else
        {
            Debug.Log(getGroupsRequest.error);
        }
    }
}
```

Файл GUICrosshair.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GUICrosshair : MonoBehaviour
{
    public Texture2D crosshairTexture;
    public float crosshairScale = 1;
    void OnGUI()
    {
        //if not paused
        if (!FindObjectOfType<GameControls>().gameIsPaused)
        {
            if (crosshairTexture != null)
```

```
            GUI.DrawTexture(new Rect((Screen.width - crosshairTexture.width * crosshairScale) / 2,
                (Screen.height - crosshairTexture.height * crosshairScale) / 2,
                crosshairTexture.width * crosshairScale, crosshairTexture.height * crosshairScale),
                crosshairTexture);
        else
            Debug.Log("No crosshair texture set in the Inspector");
    }
  }
}
```

Файл LabDropdownPicker.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;

public class LabDropdownPicker : MonoBehaviour
{
    public List<GameObject> labPages;
    public List<RectTransform> labPrefabs;

    private TMP_Dropdown labDropdown;
    private GameObject currentLabPage;
    private int currentIndex;

    public Button startButton;
    public Button stopButton;

    public TMP_Text gradeText;

    private TimerScript timer;

    private LabInfo currentLabInfo;

    private string currentTitle;

    private void Start()
    {
        timer = FindObjectOfType<TimerScript>();
        labDropdown = GetComponentInChildren<TMP_Dropdown>();
        stopButton.interactable = false;

        StartCoroutine(OnStart());
    }

    IEnumerator OnStart()
    {
        yield return new WaitForSeconds(0.01f);

        GetLabIndex(labDropdown);

        labDropdown.onValueChanged.AddListener(delegate { GetLabIndex(labDropdown); });
```

```
    }

    private void GetLabIndex(TMP_Dropdown dropdown)
    {
        if (dropdown.options.Count != 0)
        {
            currentIndex = dropdown.value;
            currentTitle = dropdown.options[currentIndex].text;

            currentLabInfo = FindObjectOfType<GetAllInfo>().getLabList()[currentTitle];

            for (int i = 0; i < labPages.Count; i++)
            {
                if (i == currentIndex)
                {
                    currentLabPage = labPages[i];
                    currentLabPage.GetComponent<LabPage>().SetTitle(currentTitle);
                    timer.SetTimerText(currentLabInfo.maxTime);
                    ResetGradeText(currentLabInfo.maxGrade);
                    currentLabPage.SetActive(true);
                }
                else
                {
                    labPages[i].SetActive(false);
                }
            }
        }
    }

    public void onStart()
    {
        PopupWindow.Instance.SetContent($"Do you really want to start lab#{currentIndex + 1}")
            .AddAcceptDeclineAction(StartLab)
            .SetAcceptDeclineButtonText("Yes", "No")
            .Show();
    }

    public void onStop()
    {
        PopupWindow.Instance.SetContent($"Do you really want to stop lab#{currentIndex + 1}")
            .AddAcceptDeclineAction(StopLab)
            .SetAcceptDeclineButtonText("Yes", "No")
            .Show();
    }

    public void StartLab()
    {
        var currentStudent =
GameObject.FindGameObjectWithTag("CurrentStudent").GetComponent<CurrentStudent>();

        if (currentStudent.IsEnoughTries(currentIndex + 1))
        {
            timer.timeLeft = currentLabInfo.maxTime * 60;
            timer.TimerOn();
```

```
            stopButton.interactable = true;
            startButton.interactable = false;
            labDropdown.interactable = false;

            ResetGradeText(currentLabInfo.maxGrade);
            currentLabPage.GetComponent<LabPage>().ActivatePage();
            currentLabPage.GetComponent<LabPage>().CreateLab(labPrefabs[currentIndex]);
        }
        else
        {
            PopupWindow.Instance.SetContent("You spent 3 attempts for this lab")
                .AddOkayAction()
                .Show();
        }
    }

    public void StopLab()
    {
        timer.TimerOff();
        float grade =
currentLabPage.GetComponent<LabPage>().CalculateResultGrade(currentLabInfo.maxGrade);
        UpdateGradeText(grade, currentLabInfo.maxGrade);
        currentLabPage.GetComponent<LabPage>().DisablePage();
        SaveNewLab(grade);
        currentLabPage.GetComponent<LabPage>().DestroyLab();

        stopButton.interactable = false;
        startButton.interactable = true;
        labDropdown.interactable = true;
    }

    private void SaveNewLab(float grade)
    {
        var currentStudent =
GameObject.FindGameObjectWithTag("CurrentStudent").GetComponent<CurrentStudent>();
        short labID = (short) (currentIndex + 1);
        int endTime = FindObjectOfType<TimerScript>().GetEndTime();
        string date = System.DateTime.Now.ToString("yyyy-MM-dd");

        StudentLabs studentLab = new StudentLabs(labID, grade, endTime, date);

        currentStudent.GetComponent<CurrentStudent>().AddLab(studentLab);

        FindObjectOfType<SaveStudentLab>().Save(studentLab, currentStudent.email);

        FindObjectOfType<ProfileScrollGroup>().RecreateProfile();

        FindObjectOfType<StudentInfoScrollGroup>().RecreateStudentsInfo();
    }

    private void UpdateGradeText(float grade, int maxGrade)
    {
        gradeText.text = grade.ToString() + "/" + maxGrade.ToString();
        gradeText.color = Math.CalculateGradeColor(grade);
```

```
    }

    private void ResetGradeText(int maxGrade)
    {
        gradeText.text = $"0/{maxGrade}";
        gradeText.color = Color.black;
    }
}
```

Файл LabPage.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;

public class LabPage : MonoBehaviour
{
    [SerializeField] GameObject explanationPage;
    public TMP_Text titleText;
    public ScrollRect scrollRect;
    private RectTransform currentLab;

    public void ActivatePage()
    {
        explanationPage.SetActive(true);
    }
    public void DisablePage()
    {
        explanationPage.SetActive(false);
    }

    public void SetTitle(string titleText)
    {
        this.titleText.text = titleText;
    }

    public void CreateLab(RectTransform labPrefab)
    {
        currentLab = Instantiate(labPrefab, new Vector3(0, 0, 0), Quaternion.identity);
        currentLab.transform.SetParent(explanationPage.transform);
        scrollRect.content = currentLab;
    }

    public void DestroyLab()
    {
        Destroy(currentLab.gameObject);
    }

    public float CalculateResultGrade(float maxGrade)
    {
```

```
        return currentLab.gameObject.GetComponent<Math>().CalculateGrade(maxGrade);
    }
}
```

Файл PopupWindow.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using UnityEngine.Events;
using static System.Net.Mime.MediaTypeNames;
using Unity.VisualScripting;

public class PopupWindow : MonoBehaviour
{

    [SerializeField]
    private Transform windowPanel;

    [SerializeField]
    private Transform contentArea;
    [SerializeField]
    private TMP_Text contentText;

    [SerializeField]
    private Transform footerArea;
    [SerializeField]
    private Button acceptButton;
    [SerializeField]
    private Button declineButton;
    [SerializeField]
    private Button alternateButton;

    [SerializeField]
    private TMP_Text acceptButtonText;
    [SerializeField]
    private TMP_Text declineButtonText;
    [SerializeField]
    private TMP_Text alternateButtonText;

    public static PopupWindow Instance;

    private void Awake()
    {
        Hide();
        Instance = this;
    }

    public PopupWindow SetContent(string content)
    {
        contentText.text = content;
        return Instance;
```

```
    }

    public void Show()
    {
        windowPanel.gameObject.SetActive(true);
    }

    public void Hide()
    {
        windowPanel.gameObject.SetActive(false);
    }

    public PopupWindow SetAcceptDeclineButtonText(string acceptText, string declineText)
    {
        declineButtonText.text = declineText;
        acceptButtonText.text = acceptText;
        return Instance;
    }
    public PopupWindow SetAlternateButtonText(string text)
    {
        alternateButtonText.text = text;
        return Instance;
    }

    public PopupWindow AddAcceptDeclineAction(UnityAction acceptAction, UnityAction declineAction =
null)
    {
        alternateButton.gameObject.SetActive(false);
        acceptButton.gameObject.SetActive(true);
        declineButton.gameObject.SetActive(true);

        acceptButton.onClick.RemoveAllListeners();
        declineButton.onClick.RemoveAllListeners();
        acceptButton.onClick.AddListener(delegate { Hide(); acceptAction(); });
        declineButton.onClick.AddListener(Hide);

        if (!declineAction.IsUnityNull())
        {
            declineButton.onClick.AddListener(declineAction);
        }

        return Instance;
    }

    public PopupWindow AddOkayAction(UnityAction alterAction = null)
    {
        alternateButton.gameObject.SetActive(true);
        acceptButton.gameObject.SetActive(false);
        declineButton.gameObject.SetActive(false);

        alternateButton.onClick.RemoveAllListeners();
        alternateButton.onClick.AddListener(Hide);

        if (!alterAction.IsUnityNull())
```

```
        {
            alternateButton.onClick.AddListener(alterAction);
        }

        return Instance;
    }
}
```

Файл ProfileScrollGroup.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class ProfileScrollGroup : MonoBehaviour
{
    public GameObject profileLabsPrefab;
    public GameObject profilePanel;

    public TMP_Text username;
    public TMP_Text group;
    public TMP_Text zeroLabsText;

    public void CreateProfile()
    {
        zeroLabsText.text = "";
        var currentStudent = GameObject.FindGameObjectWithTag("CurrentStudent");
        var labs = currentStudent.GetComponent<CurrentStudent>().GetStudentLabs();

        username.text = currentStudent.GetComponent<CurrentStudent>().firstName + " " +
currentStudent.GetComponent<CurrentStudent>().secondName;
        group.text = currentStudent.GetComponent<CurrentStudent>().group;

        if (labs.Count != 0)
        {
            foreach (var item in labs)
            {
                var profileLab = Instantiate(profileLabsPrefab, new Vector3(0, 0, 0), Quaternion.identity);
                profileLab.transform.SetParent(profilePanel.transform);
                profileLab.GetComponent<StudentProfileLabs>().labID = item.labID;
                profileLab.GetComponent<StudentProfileLabs>().labGrade = item.grade;
                profileLab.GetComponent<StudentProfileLabs>().labEndTime = item.endTime;
                profileLab.GetComponent<StudentProfileLabs>().labDate = item.date;
            }
        }
        else
        {
            zeroLabsText.text = "There is no labs";
        }
    }

    public void DestroyPrefabs()
    {
```

```
        username.text = "";
        group.text = "";

        FindObjectOfType<TabGroup>().BadThing(true);
        var prefabs = GameObject.FindGameObjectsWithTag("ProfileLabsPrefab");

        foreach (var item in prefabs)
        {
            Destroy(item);
            //Debug.Log("Destroying...");
        }
        FindObjectOfType<TabGroup>().BadThing(false);
        //Debug.Log("DestroyPrefabs");
    }

    public void RecreateProfile()
    {
        DestroyPrefabs();
        CreateProfile();
    }
}
```

Файл Rotator.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotator : MonoBehaviour
{
    [SerializeField] LineRenderer lineRenderer;
    [SerializeField] List numberList;
    [SerializeField] [Range(0.01f, 1f)] float lerpTime = 0.1f;
    [SerializeField] float rotateAngle = 16.3f;
    [SerializeField] int posHolder;
    [SerializeField] bool isTime;
    [SerializeField] bool isVoltage;
    [SerializeField] bool isY;
    [SerializeField] bool isX;
    [SerializeField] bool isSwitcher;
    [SerializeField] AnimationCurve curve;

    [SerializeField] [Range(1, 181)] int maxRotate = 1;

    private float startTime, endTime;
    private Quaternion startPos;

    private bool isMultiplied;
    private bool isPressed;
    private bool isRotate;
    private Sinwave sinwave;

    private void Awake()
    {
```

```
    startPos = transform.rotation;
    sinwave = lineRenderer.GetComponent<Sinwave>();
}

void OnMouseDown()
{
    FindObjectOfType<CameraMovement>().Lock();
    this.isPressed = true;
    startTime = Time.time;
}

private void OnMouseUp()
{
    if (!FindObjectOfType<GameControls>().gameIsPaused)
    {
        FindObjectOfType<CameraMovement>().Unlock();
    }
    this.isPressed = false;
    endTime = Time.time;

    if (endTime - startTime < 0.1f)
    {
        sendMultiplayer();
    }
}

// Update is called once per frame
void Update()
{
    //transform.localRotation *= Quaternion.AngleAxis(30 * Time.deltaTime, Vector3.left);

    if (isPressed && !isRotate)
    {
        StartCoroutine(Rotate());
    }
}

public IEnumerator Rotate()
{
    while(isPressed)
    {
        isRotate = true;
        float localRotateAngle = 0;
        float timeElapsed = 0;
        short sigh = 0;

        if (Input.GetAxis("Mouse X") > 0 || Input.GetAxis("Mouse Y") > 0)
        {
            sigh = 1;
            localRotateAngle += rotateAngle;
        }
        else if (Input.GetAxis("Mouse X") < 0 || Input.GetAxis("Mouse Y") < 0)
        {
            sigh = -1;
```

```
            localRotateAngle -= rotateAngle;
        }
        else
        {
            sigh = 0;
            localRotateAngle = 0;
        }

        Quaternion startRotation = transform.rotation;
        Quaternion targetRotation = transform.rotation * Quaternion.Euler(localRotateAngle, 0f, 0f);

        float angle = Quaternion.Angle(startPos, targetRotation);

        if (angle < maxRotate) {
            while (timeElapsed < lerpTime)
            {
                transform.rotation = Quaternion.Lerp(startRotation, targetRotation,
curve.Evaluate(timeElapsed/lerpTime));
                timeElapsed += Time.deltaTime;
                yield return null;
            }

            transform.rotation = targetRotation;
            isRotate = false;

            sendScaler(sigh);
        }
        else {
            yield return null;
        }

        angle = Mathf.Clamp(angle, 0, maxRotate);

        //Debug.Log(angle);
    }
}

private void sendMultiplayer()
{
    if (!this.isMultiplied)
        {
            if(isX && !isVoltage && !isTime && !isY)
            {
                sinwave.multiplyTime(isMultiplied);
            }
            else if(isY && !isVoltage && !isX && !isTime)
            {
                sinwave.multipleVoltage(isMultiplied);
            }
            this.isMultiplied = true;
        }
        else
        {
            if(isX && !isVoltage && !isTime && !isY)
```

```
            {
                sinwave.multiplyTime(isMultiplied);
            }
            else if(isY && !isVoltage && !isX && !isTime)
            {
                sinwave.multipleVoltage(isMultiplied);
            }
            this.isMultiplied = false;
        }
    }

    private void sendScaler(short sign)
    {
        if(isTime && !isVoltage && !isX && !isY && !isSwitcher)
        {
            sinwave.setTimeScaler(getNumber(sign));
        }
        else if(isVoltage && !isTime && !isX && !isY && !isSwitcher)
        {
            sinwave.setVoltageScaler(getNumber(sign));
        }
        else if(isX && !isVoltage && !isTime && !isY && !isSwitcher)
        {
            sinwave.setXTransform(Input.GetAxis("Mouse X"));
        }
        else if(isY && !isVoltage && !isX && !isTime && !isSwitcher)
        {
            sinwave.setYTransform(Input.GetAxis("Mouse X"));
        }
        else if(isSwitcher && !isVoltage && !isX && !isTime && !isY)
        {
            sinwave.setY(sign);
        }
    }

    private float getNumber(short sign)
    {
        //Debug.Log(posHolder);
        if (sign == 1)
        {
            this.posHolder++;
            posHolder = Mathf.Clamp(posHolder, 0, numberList.list.Count - 1);
            return numberList.list[posHolder];
        }
        else if (sign == -1)
        {
            this.posHolder--;
            posHolder = Mathf.Clamp(posHolder, 0, numberList.list.Count - 1);
            return numberList.list[posHolder];
        }
        else
        {
            return numberList.list[posHolder];
        }
```

```
    }
}
```

Файл SideMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SideMenu : MonoBehaviour
{
    public GameObject panelMenu;

    public void ShowHide()
    {
        if (panelMenu != null)
        {
            Animator animator = panelMenu.GetComponent<Animator>();
            if (animator != null)
            {
                bool isOpen = animator.GetBool("Show");
                animator.SetBool("Show", !isOpen);
            }
        }
    }
}
```

Файл StudentController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class StudentController : MonoBehaviour
{
    private void Start()
    {
        FindObjectOfType<TabGroup>().BlockPagesButtons();
    }

    public void onSignOut()
    {
        PopupWindow.Instance.SetContent("Do you really want to unlogin?")
            .AddAcceptDeclineAction(SignOut)
            .SetAcceptDeclineButtonText("Yes", "No")
            .Show();
    }

    public void SignOut()
    {
        var currentStudent = GameObject.FindGameObjectsWithTag("CurrentStudent");
```

```
        foreach(var item in currentStudent)
        {
            Destroy(item);
        }
        //Debug.Log("SignOut");
        FindObjectOfType<TabGroup>().BlockPagesButtons();
        FindObjectOfType<ProfileScrollGroup>().DestroyPrefabs();
        FindObjectOfType<StudentInfoScrollGroup>().DesctroyStudentsInfoPrefabs();
    }
}
```

Файл StudentInfo.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.UI;

public class StudentInfo : MonoBehaviour
{
    public TMP_Text studentNameText;
    public TMP_Text studentGroupText;
    public TMP_Text labIDText;
    public TMP_Text labGradeText;
    public TMP_Text labDateText;
    public GameObject delete;
    public Button deleteButton;

    public string studentName;
    public string studentGroup;
    public short labTaskID;
    public float labGrade;
    public string labDate;
    public int labID;


    public void AssignInfo(int labID, string name, string group, short labTaskID, float grade, string date, bool
isAdmin)
    {
        delete.SetActive(isAdmin);
        this.labID = labID;
        studentName = name;
        studentGroup = group;
        this.labTaskID = labTaskID;
        labGrade = grade;
        labDate = date;
    }

    // Start is called before the first frame update
    void Start()
    {
```

```
        deleteButton.onClick.AddListener(delegate { DeleteStudentLabInfo(); });
        studentNameText.text = studentName;
        studentGroupText.text = studentGroup;
        labIDText.text = labTaskID.ToString();
        labGradeText.text = labGrade.ToString();
        labDateText.text = labDate;
    }

    private void DeleteStudentLabInfo()
    {
        FindObjectOfType<AdminFunctionality>().DeleteStudentLab(labID);
        Destroy(gameObject);
    }
}
```

Файл StudentInfoScrollGroup.cs

```
using SimpleJSON;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using TMPro;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.Networking;

public class StudentInfoScrollGroup : MonoBehaviour
{
    public GameObject studentInfoPrefab;
    public GameObject studentInfoPanel;

    public TMP_InputField nameInput;
    public TMP_Dropdown groupDropdown;
    public TMP_Dropdown labDropdown;
    public TMP_Text zeroInfoText;

    private bool isAdmin;

    public void GetAllStudentsInfo()
    {
        StartCoroutine(SendGetAllStudentsForm());
    }

    IEnumerator SendGetAllStudentsForm()
    {
        zeroInfoText.text = "";

        string group = groupDropdown.options[groupDropdown.value].text;
        string lab = labDropdown.options[labDropdown.value].text;

        //Debug.Log($"Group {group} Lab {lab} Input {nameInput.text}");

        WWWForm allStudentsForm = new WWWForm();
        allStudentsForm.AddField("apppassword", "password");
```

```
        allStudentsForm.AddField("name", nameInput.text);
        allStudentsForm.AddField("group", group);
        allStudentsForm.AddField("lab", lab);
        using UnityWebRequest allStudentsRequest =
UnityWebRequest.Post("http://localhost:7777/oscilloscope/leaderboard/getallstudents.php",
allStudentsForm);
        yield return allStudentsRequest.SendWebRequest();
        if (allStudentsRequest.error == null)
        {
            DesctroyStudentsInfoPrefabs();
            string response = allStudentsRequest.downloadHandler.text;
            //Debug.Log(response);

            if (response == "1" || response == "2")
            {
                Debug.Log("Server Error");
            }
            else if (response == "3")
            {
                zeroInfoText.text = "Ther is no info with those params";
            }
            else
            {
                JSONNode studentsInfo = JSON.Parse(allStudentsRequest.downloadHandler.text);

                foreach (JSONNode item in studentsInfo)
                {
                    var studentInfo = Instantiate(studentInfoPrefab, new Vector3(0, 0, 0), Quaternion.identity);

                    studentInfo.GetComponent<StudentInfo>().AssignInfo(item["id"],
                        item["student_name"],
                        item["group_name"],
                        short.Parse(item["lab_id"]),
                        float.Parse(item["grade"]),
                        item["date"],
                        isAdmin);

                    studentInfo.transform.SetParent(studentInfoPanel.transform);
                }
            }
        }
    }

    public void ActivateAdminFunc()
    {
        isAdmin = true;
    }

    public void DesctroyStudentsInfoPrefabs()
    {
        var prefabs = GameObject.FindGameObjectsWithTag("StudentInfoPrefab");

        foreach (var item in prefabs)
        {
```

```
            Destroy(item);
        }
    }

    public void RecreateStudentsInfo()
    {
        DesctroyStudentsInfoPrefabs();
        GetAllStudentsInfo();
    }
}
```

Файл TimerScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class TimerScript : MonoBehaviour
{
    public float timeLeft;
    private bool isTimerOn;
    private float timeHolder;

    public TMP_Text timerText;
    private void Start()
    {
        ResetTimer();
    }

    void Update()
    {
        if (isTimerOn)
        {
            if (timeLeft > 0)
            {
                timeLeft -= Time.deltaTime;
                UpdateTimer(timeLeft);
            }
            else
            {
                timeLeft = 0;
                TimerOff();
            }
        }
    }

    public void TimerOn()
    {
        timeHolder = timeLeft;
        this.isTimerOn = true;
    }

    public void TimerOff()
```

```
    {
        this.isTimerOn = false;
        ResetTimer();
    }

    private void UpdateTimer(float currentTime)
    {
        currentTime += 1;

        float minutes = Mathf.FloorToInt(currentTime / 60);
        float seconds = Mathf.FloorToInt(currentTime % 60);

        timerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
    }

    private void ResetTimer()
    {
        float minutes = Mathf.FloorToInt(timeHolder / 60);
        float seconds = Mathf.FloorToInt(timeHolder % 60);

        timerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
    }

    public int GetEndTime()
    {
        int minutes = Mathf.RoundToInt(timeHolder / 60);
        int minutesLeft = Mathf.RoundToInt(timeLeft / 60);

        return (minutes - minutesLeft);
    }

    public void SetTimerText(int text)
    {
        timerText.text = $"{text}:00";
    }
}
```

Файл AddNewGroup.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
}
```

```
$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}

$groupname = $_POST["groupName"];
$groupnameClean = filter_var($groupname, FILTER_SANITIZE_STRING);

$checkgroupquery = "SELECT * FROM groups WHERE group_name = '".$groupnameClean."';";
$checkgroupresult = mysqli_query($link, $checkgroupquery) or die("2");

if(mysqli_num_rows($checkgroupresult) > 0) {
    echo("3");
    exit();
}

$insertnewgroupquery = "INSERT INTO groups(group_name) VALUES ('".$groupnameClean."');";
mysqli_query($link, $insertnewgroupquery) or die("4");
echo("0");

$link->close();

//Errors codes
//0 - All is ok
//1 - Data base connection error
//2 - Check query failed
//3 - This group is allready exist
//4 - Insert query failed

?>
```

Файл ChangeGroupName.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
}
```

```
$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}

$groupname = $_POST["groupName"];
$newgroupname = $_POST["newGroupName"];
$newgroupnameClean = filter_var($newgroupname, FILTER_SANITIZE_STRING);

$checkgroupquery = "SELECT * FROM groups WHERE group_name = '".$groupname."';";
$checkgroupresult = mysqli_query($link, $checkgroupquery) or die("2");

if(mysqli_num_rows($checkgroupresult) < 1) {
    echo("3");
    exit();
}

$updategroupquery = "UPDATE groups
                SET group_name = '".$newgroupnameClean."'
                WHERE group_name = '".$groupname."';";
mysqli_query($link, $updategroupquery) or die("4");
echo("0");

$link->close();

//Errors codes
//0 - All is ok
//1 - Data base connection error
//2 - Check query failed
//3 - This group didnt exist
//4 - Update query failed

?>
```

Файл ChangeLabParams.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
```

```
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}

$labname = $_POST["labName"];
$newgrade = $_POST["newGrade"];
$newtime = $_POST["newTime"];

$alterlabparamsquery = "UPDATE labs
                        SET max_time = '".$newtime."',
                            max_grade = '".$newgrade."'
                        WHERE lab_name = '".$labname."';";
mysqli_query($link, $alterlabparamsquery) or die("2");
echo("0");

$link->close();

//Errors codes
//0 - All is ok
//1 - Data base connection error
//2 - Alter query failed

?>
```

Файл DeleteStudentLab.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}
```

```
$labID = $_POST["labID"];

$deletestudentlabquery = "DELETE FROM estimates WHERE id = '".$labID."';";
mysqli_query($link, $deletestudentlabquery) or die("2");
echo("0");

$link->close();

//Errors codes
//0 - All is ok
//1 - Data base connection error
//2 - Delete query failed
```

Файл GetAllInfo.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}

$getallgroups = "SELECT group_name FROM groups ORDER BY id ASC;";
$groupsresult = mysqli_query($link, $getallgroups) or die("2");

$getalllabs = "SELECT * FROM labs ORDER BY id ASC;";
$labsresult = mysqli_query($link, $getalllabs) or die("3");

if ($groupsresult->num_rows > 0 && $labsresult->num_rows > 0) {
    $group_array = array();
    $labs_array = array();

    while($group = mysqli_fetch_assoc($groupsresult)) {
        $group_array[] = $group;
    }

    while($lab = mysqli_fetch_assoc($labsresult)) {
```

```
        $labs_array[] = $lab;
    }

    $json_array = array('group_info' => $group_array, 'labs_info' => $labs_array);
    echo json_encode($json_array);
} else {
    echo("4");
    exit();
}

$link->close();

//Error Codes
//1 - Database connection error
//2 - Group query error
//3 - Labs query error
//4 - Result didnt have any info


?>
```

Файл LoginStudent.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}

$email = $_POST["email"];
$emailClean = filter_var($email, FILTER_SANITIZE_EMAIL);
$password = $_POST["password"];

$emailcheckquerry = "SELECT password
                    FROM students
```

```
                WHERE email = '".$emailClean."';";
$emailcheckresult = mysqli_query($link, $emailcheckquerry) or die("2");

if ($emailcheckresult->num_rows != 1) {
   echo("3");
   exit();
}
else {
   $fetchedpassword = mysqli_fetch_assoc($emailcheckresult)["password"];
   if (password_verify(($password), $fetchedpassword)) {
      $studentinfo = "SELECT s.id, s.first_name, s.second_name, s.email, g.group_name
                  FROM students s
                  LEFT JOIN groups g
                  ON s.group_id = g.id
                  WHERE s.email = '".$emailClean."';";
      $studentinforesult = mysqli_query($link, $studentinfo) or die("5");
      $existingstudentinfo = mysqli_fetch_assoc($studentinforesult);

      $studentid = $existingstudentinfo["id"];
      $studentlabsinfo = "SELECT e.lab_id, e.grade, e.end_time, e.date
                     FROM estimates e
                     WHERE student_id = '".$studentid."'
                     ORDER BY e.lab_id ASC;";
      $studentlabinforesult = mysqli_query($link, $studentlabsinfo) or die("6");

      $student_info_array = array('first_name' => $existingstudentinfo["first_name"],
                           'second_name' => $existingstudentinfo["second_name"],
                           'email' => $existingstudentinfo["email"],
                           'group_name' => $existingstudentinfo["group_name"]);

      $student_lab_info_array = array();

      if (!$studentlabinforesult->num_rows < 1) {
         while ($row = mysqli_fetch_assoc($studentlabinforesult)) {
            #$labid = $row["lab_id"];
            $lab_array = array($row);
            $student_lab_info_array[] = $lab_array;
         }
      }
      else {
         $student_lab_info_array = null;
      }

      $json_array = array('student_info' => $student_info_array, 'labs' => $student_lab_info_array);

      echo json_encode($json_array);

   }
   else {
      echo("4");
      exit();
```

```
    }
}

$link->close();

//Errors codes
//1 - Data base connection error
//2 - Username querry error
//3 - Username not exist or there is more that 1
//4 - Password was not able to be verified
//5 - Playerinfo query failed
//6 - Labs query failed



?>
```

Файл RegisterStudent.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
    echo("1");
    exit();
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
    exit();
}

//wwwform from Unity
$firstname = $_POST["firstname"];
$secondname = $_POST["secondname"];
$email = $_POST["email"];
$emailClean = filter_var($email, FILTER_SANITIZE_EMAIL);
$group = $_POST["group"];
$password = $_POST["password"];
$passwordhash = password_hash($password, PASSWORD_DEFAULT);

$emailcheckquerry = "SELECT email
```

```
                FROM students
                WHERE email = '".$emailClean."';";
$emailcheck = mysqli_query($link, $emailcheckquerry) or die("2");

if(mysqli_num_rows($emailcheck) > 0) {
   echo("3");
   exit();
}

$insertstudentquerry = "INSERT INTO students(first_name, second_name, email, group_id, password)
                VALUES('".$firstname."', '".$secondname."', '".$emailClean."', (SELECT id FROM groups WHERE
group_name = '".$group."'), '".$passwordhash."');";
mysqli_query($link, $insertstudentquerry) or die("4");
echo("0");

$link->close();

//Errors codes
//1 - Data base connection error
//2 - Email check querry failed
//3 - Email already exist
//4 - Insert user failed

?>
```

Файл SaveStudentLab.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);

if (mysqli_connect_errno()) {
   echo("1");
   exit();
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
   exit();
}

$email = $_POST["email"];
$emailClean = filter_var($email, FILTER_SANITIZE_EMAIL);
```

```
$labID = $_POST["labID"];
$grade = $_POST["grade"];
$endtime = $_POST["endTime"];
$date = $_POST["date"];

$insertnewlab = "INSERT INTO estimates(student_id, lab_id, grade, end_time, date)
            VALUES((SELECT id FROM students WHERE email = '".$emailClean."'), '".$labID."', '".$grade."',
'".$endtime."', '".$date."');";
mysqli_query($link, $insertnewlab) or die("2");
echo("0");

$link->close();

//Errors codes
//1 - Data base connection error
//2 - Insert lab failed

?>
```

Файл GetAllStudents.php

```
<?php

$user = 'root';
$password = 'root';
$db = 'oscilloscope';
$host = 'localhost';
$port = 7778;

$link = mysqli_connect($host, $user, $password, $db, $port);
if (mysqli_connect_errno()) {
   echo("1");
   exit();
}

$apppassword = $_POST["apppassword"];

if($apppassword != "password") {
   exit();
}
$studentname = $_POST["name"];
$group = $_POST["group"];
$lab = $_POST["lab"];

$studentnamecheck = isEmpty($studentname);
$groupcheck = isEmpty($group);
$labcheck = isEmpty($lab);

$getallstudentsinfoquerry = "SELECT
   CONCAT(s.first_name, \" \", s.second_name) AS student_name,
```

```
        e.id,
        g.group_name,
        e.lab_id,
        e.grade,
        e.date
        FROM
        students s
    LEFT JOIN estimates e ON
        e.student_id = s.id
    LEFT JOIN groups g ON
        g.id = s.group_id
    LEFT JOIN labs l ON
        l.id = e.lab_id
    WHERE
        CONCAT(s.first_name, \" \", s.second_name) LIKE IF(".$studentnamecheck." IS NULL,
CONCAT(s.first_name, \" \", s.second_name), ".$studentnamecheck.")
        AND g.group_name LIKE IF(".$groupcheck." IS NULL, g.group_name, ".$groupcheck.")
        AND l.lab_name LIKE IF(".$labcheck." IS NULL, l.lab_name, ".$labcheck.");";
$getallstudentcheckresult = mysqli_query($link, $getallstudentsinfoquerry) or die("2");

if ($getallstudentcheckresult->num_rows < 1) {
    echo("3");
    exit();
}
else
{
    $json_array = array();

    while($row = mysqli_fetch_assoc($getallstudentcheckresult)) {
        $json_array[] = $row;
    }

    echo json_encode($json_array);
}
$link->close();

function isEmpty($text)
{
    if ($text == "" || $text == "None")
    {
        return "NULL";
    }
    return "'%$text%'";
}

//Error Codes
//1 - Database connection error
//2 - All info query error
//3 - There is no info

?>
```