Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: Розроблення застосунку для проведення онлайн-курсів засобами Java

Виконав: студент **II** курсу групи **КНС - 21**
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Держило Роман Степанович
(прізвище та ініціали)

Керівник: Опришко Мар'ян Іванович
(прізвище та ініціали)

Керівник: Яцишин Світлана Іванівна
(прізвище та ініціали)

Рецензент Прусак Юрій Володимирович
(прізвище та ініціали)

Львів – 2025 року

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____ Бурецька І. Б.
"10" червня 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Держиду Роману Степановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для проведення онлайн-курсів засобами Java

керівник роботи *Опришко Мар'ян Іванович, асистент*

керівник роботи *Яцишин Світлана Іванівна, доцент, кт. наук*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "15" 11 2024 року № С-882

2. Термін подання студентом роботи 10 червня 2025 року

3. Вихідні дані до роботи Розроблення застосунку для проведення онлайн-курсів засобами Java

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Стан проблемної області, інформаційне та математичне забезпечення, програмне та технічне забезпечення, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Підготовка матеріалу до відповіді.

6. Дата видачі завдання 18.11.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел	18.11.24 – 06.12.24	Виконано
2	Аналіз проблемної області та постановка задачі	06.12.24 – 10.01.25	Виконано
3	Проектування інформаційного та математичного забезпечення	10.01.25 – 07.02.25	Виконано
4	Програмна реалізація проекту	07.02.25 – 02.05.25	Виконано
5	Відлагодження програмної реалізації	02.05.25 – 23.05.25	Виконано
6	Оформлення записки до дипломного проекту	23.05.25 – 08.06.25	Виконано

Студент _____ (підпис) **Держило Р. С.** (прізвище та ініціали)

Керівник роботи _____ (підпис) **Опришко М. І.** (прізвище та ініціали)

Керівник роботи _____ (підпис) **Яцишин С. І.** (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота містить 138 сторінок, 1 таблицю, 70 ілюстрацій, 5 додатків, 10 джерел та 10 формул.

Ця робота описує етапи розробки та впровадження ефективної програми для проведення онлайн-курсів. Такий застосунок покращить реалізацію онлайн-навчання, які проводять репетитори, оскільки вона містить всі необхідні функції у собі, що є зручним як для викладачів, так і для студентів. Результати цієї роботи можуть бути застосовані у різних освітніх закладах та безпосередньо самими репетиторами, які займаються навчанням студентів.

Ключові слова: Java, клієнт-серверна архітектура, онлайн-курси, користувацький інтерфейс, реляційна база даних MySQL, віддалений хостинг, FTP.

ABSTRACT

The Bachelor’s thesis consists of 138 pages, 1 table, 70 figures, 5 appendices, 10 references and 10 formulas.

This work describes the stages of designing and implementing an efficient application for delivering online courses.

The proposed solution streamlines private tutoring and distance-learning workflows by combining all essential features for both teachers and students in a single desktop tool.

The results of this project can be adopted by educational institutions as well as individual tutors who provide remote instruction.

Keywords: Java, client-server architecture, online courses, user interface, relational MySQL database, remote hosting, FTP.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити комп’ютерний застосунок для проведення онлайн-курсів. Використати такі технології: мова програмування Java, середовище SceneBuilder для побудови графічних вікон JavaFX, базу даних MySQL та віддалений хостинг.

Застосунок має виконувати наступні функції: реєстрацію та авторизацію репетиторів і студентів, з введенням даних у реєстраційних полях та вибором категорії користувача. У вікні авторизації реалізувати введення логіну, паролю та вибору категорії.

Після входу відкривається головна панель, де відображаються всі курси користувача. Додати кнопки: зміни мови інтерфейсу, створення курсу (для викладача) та реєстрації в курсі за кодом (для студента).

Вікно обраного курсу повинно містити вертикальне меню з кнопками для переходу на панель курсу, матеріалів, завдань, тестів, журналу успішності, панелі студентів курсу (для викладача), чатів (викладач має окремі чати зі студентами, студент – лише з викладачем), зустрічі в Google Meet, оголошення, рекомендації курсів з можливістю створення власної та повернення на головну панель.

У вмісті курсу реалізувати редагування матеріалів, завдань, тестів тощо, зміну доступу студентів, а також видалення компонентів, студентів і курсу.

Для завдань і тестів передбачити здачу робіт студентами, перегляд робіт обома категоріями, оцінювання – лише викладачем. У тестах реалізувати питання з однією правильною відповіддю, кількома, відповідністю, текстовою відповіддю, з можливістю вставки зображення.

Усі файли зберігати на FTP у межах вибраного віддаленого хостингу, а дані про курси й користувачів – у базі даних хостингу. Після завершення розробки підготувати докладний звіт, який охоплює як теоретичні аспекти проєкту, так і процес виконання кожного завдання, зазначеного в технічному завданні.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

FTP - протокол для передачі файлів по мережі.

PHP - мова сценаріїв, що виконується на стороні сервера.

MySQL - система керування базами даних на мові SQL.

JavaFX - платформа для створення графічних інтерфейсів користувача.

ER-діаграми - графічне зображення таблиць бази даних, їх атрибутів та зв’язків між ними.

MVC – архітектурна модель розробки застосунку, у якій відбувається поділ відповідальностей між компонентами всієї програми.

Fxml – файли у яких прописаний код реалізацій графічних вікон на JavaFX.

ВСТУП

Сьогодні розвиток цифрових технологій та їх зміни, викликані як глобальними подіями, так і зростанням популярності дистанційного навчання, спричинили дуже швидке поширення онлайн-освіти. Все більше репетиторів та приватних викладачів обирають організацію своїх навчальних курсів в онлайн-форматі, щоб не прив'язуватися до освітніх установ чи фізичного місця роботи. Такий підхід дозволяє максимально гнучко складати розклад занять і забезпечувати доступ до матеріалів у будь-який час. Велике значення цей формат отримав саме під час війни в Україні, коли велика кількість людей змушена була змінити місце проживання. Ця ситуація призвела до того, що очна освіта стала обмеженою та надзвичайно небезпечною як для викладачів, так і для студентів. У таких умовах виникла потреба у створенні універсального програмного рішення, що полегшить викладачам організацію навчального процесу, а студентам допоможе ефективно здобувати знання в онлайн-середовищі.

Об’єктом дослідження є процес організації онлайн-навчання за допомогою комп’ютерних засобів.

Предметом дослідження виступає програмний застосунок для проведення онлайн-курсів, зручний для використання викладачами та студентами.

Метою дипломного проєкту є розробка програмного застосунку "SparksNet", який забезпечить зручну організацію та проходження онлайн-курсів, враховуючи інтереси і викладачів, і студентів. Програма орієнтована переважно на репетиторів, які самостійно проводять заняття, працюють з кількома групами й потребують простого, але водночас функціонального інструменту для взаємодії зі своїми студентами.

Для досягнення мети передбачено виконання таких завдань:

- створення зручного графічного інтерфейсу з використанням SceneBuilder і JavaFX;
- реалізація функцій реєстрації та авторизації користувачів, із можливістю вибору ролі: студент або викладач;
- відображення курсів, до яких належить користувач, з можливістю

створювати нові курси або долучатися до існуючих;

- додавання та редагування навчальних матеріалів, завдань, тестів (викладачем);
- проходження тестів та здача завдань студентами;
- збереження результатів у базі даних, що дозволяє викладачам контролювати успішність;
- реалізація особистих чатів між студентами і викладачами, з можливістю збереження історії повідомлень;
- проведення онлайн-зустрічей через Google Meet;
- зберігання завантажених файлів репетиторів у файловій системі на віддаленому хостингу, а всіх даних програми — у базі MySQL.

Практичне значення роботи полягає у створенні комп’ютерного застосунку "SparksNet", який стане ефективним інструментом для організації дистанційного навчання в умовах обмеженого доступу до очної освіти.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Сучасний стан ринку онлайн-освіти та репетиторських послуг

Останніми роками онлайн-освіта стала затребуваною часткою навчання в усьому світі. Онлайн-навчання показує велике зростання завдяки широкій доступності Інтернету та наявності багатьох причин, які не дозволяють здійснювати навчання очно. Особливо великий її розвиток почався у 2020 році, коли з’явилася пандемія COVID-19,через яку багато освітніх закладів у всьому світі мали перейти на дистанційне навчання, ця ситуація не пройшла повз і самих репетиторів.

Згідно Tandfonline [1] у країнах Європи, США, Канаді та багатьох інших країнах онлайн-репетиторство стало дуже великою можливістю навчати людей, не прив’язуючись до освітнього закладу, оскільки не всі мають можливість або достатньо часу для навчання в традиційному вигляді, ця популярність на онлайн-репетиторство з року в рік все більше зростає (див. рис.1.1). Наприклад, за даними дослідження Sutton Trust, у Великій Британії близько 50% учнів користуються послугами онлайн-репетиторів.

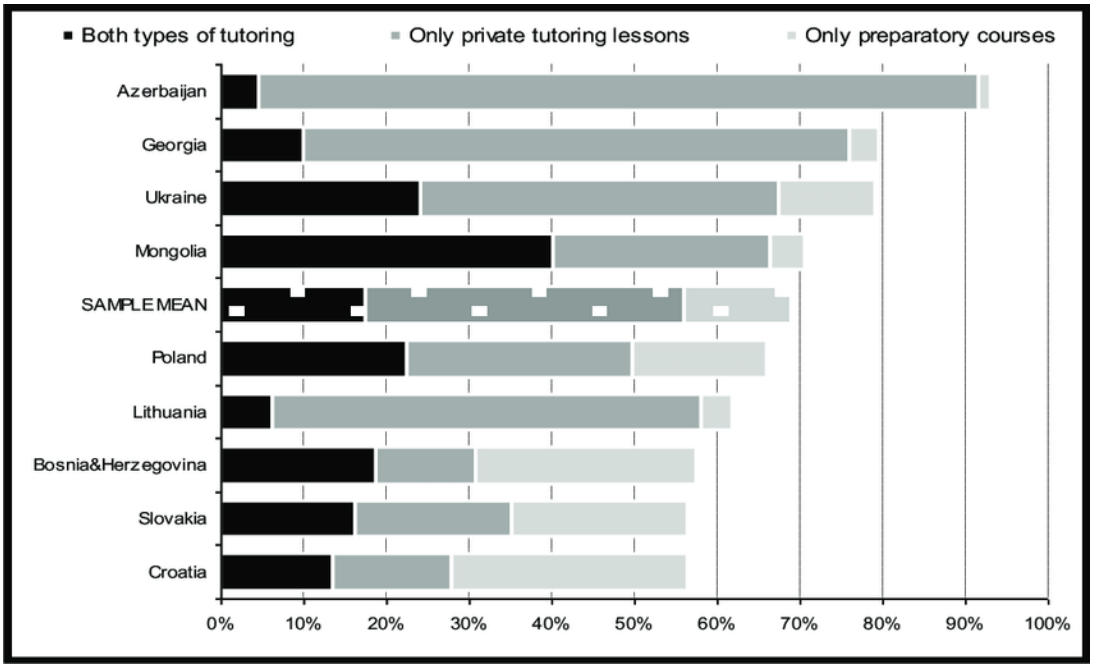

Рисунок 1.1 – Статистика застосування онлайн-репетитерства в деяких країнах світу

У Сполучених Штатах дуже активно розвиваються платформи на такого виду як: Chegg Tutors, Varsity Tutors, Wyzant, які надають викладачам можливість проводити свої дистанційні заняття з учнями різного рівня підготовки.

В Україні, згідно Uej [2] дистанційне навчання також швидко росте, особливо це відбулося у 2020 році, як і у всіх інших країнах, але ще більше зросло через повномасштабну війну в Україні. Українські викладачі зараз дуже активно використовують різні онлайн-сервіси для проведення своїх дистанційних занять (див. рис.1.2), щоб надати можливість своїм учням продовжувати навчання навіть при таких жахливих ситуаціях. Особливо актуальним онлайн-навчання стало для репетиторів, які прагнуть зберігати свою діяльність у важких для них умовах.

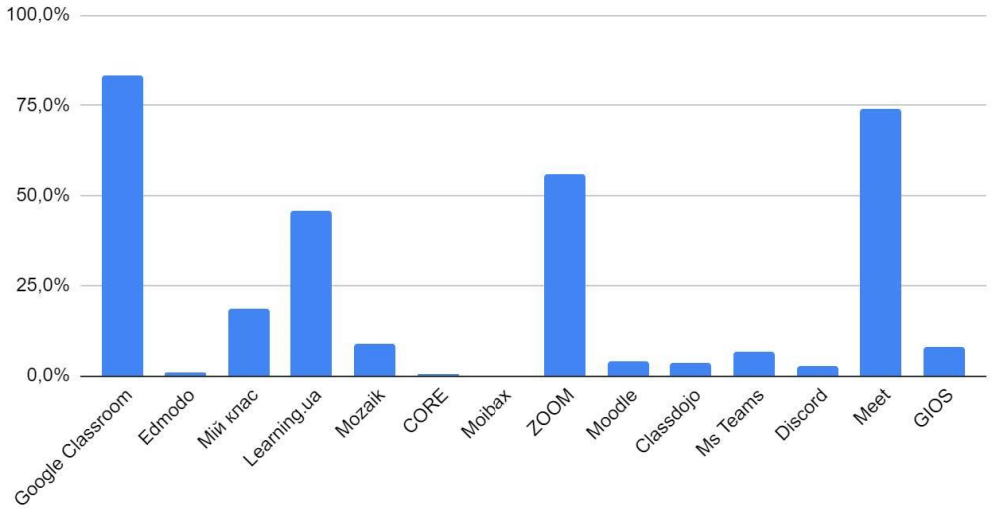

Рисунок 1.2 – Статистика застосування дистанційних освітніх платформ українськими викладачами

Репетиторські послуги дозволяють індивідуально підійти до навчання та швидко адаптувати процес навчання під конкретного учня. Все більше освітян переходять до онлайн-формату через його великі переваги.

Багато різних освітніх платформ є або занадто складними для індивідуальних викладачів, або орієнтованими на великі освітні заклади.

Репетитори потребують простого та функціонального рішення, яке дозволить:

- ♣ створювати їм свої власні онлайн-курси;
- ♣ створювати матеріали, завдання та тестування;

♣ підтримувати спілкування зі своїми студентами та стежити за їхніми успіхами у навчанні;

♣ зберігати всю інформацію курсу в одному місці.

Зараз викладачі часто використовують для онлайн-навчання— Zoom, Google Meet, Google Forms, Testportal, Google Drive. Це все надає великі труднощі викладачам та робить хаос в їхній організації навчання. Найбільше це погіршується для викладачів, коли вони вводять тривалі курси або здійснюють одночасне навчання з декількома групами.

Через це є велика потреба у створенні додатків, який зможе об'єднати ці всі дірки в організації онлайн-навчання.

Отже, онлайн-освіта розвивається на теперішній момент з дуже високою швидкістю, але проблема у відсутності додатку, який буде забезпечувати наявність всіх компонентів в одному місці, далі є невирішеною.

1.2. Використання комп’ютерних застосунків для онлайн-навчання

Зараз комп'ютерні програми стають важливою часткою навчального процесу. Вони допомагають проводити навчання на дистанції, забезпечуючи надзвичайно високу якість взаємодії викладача та учнів. Об'єднуваність онлайн-формату відкриває нові можливості викладання й навчання, зокрема — гнучкий графік, індивідуальний підхід, можливості доступу до матеріалів будь-коли, а також відсутність обмежень у своєму розташуванні.

Багато викладачів та репетиторів використовують різні онлайн-програми для проведення дистанційного навчання (див. рис.1.3), вони найчастіше застосовують Zoom, Google Meet для проведення відеозустрічей, згідно Jta [3]. Для організації матеріалів часто використовуються сервіси Google Drive, Dropbox, а для тестування — Google Forms, Testportal. В університетах популярною є навчальна платформа Moodle, а в школах та коледжах Google Classroom, але більшість з цих додатків не є доступними викладачам для репетиторства, бо вони зареєстровані конкретно для навчання у закладах освіти, і це стає проблемою для них.

Рисунок 1.3 – Програми, які застосовуються для онлайн-навчання

Зазвичай багато викладачів використовують одразу декілька інструментів, щоб забезпечити всі функції: проведення занять, перевірка домашніх завдань, тестування, обмін файлами, зворотний зв'язок, планування занять. Це допомагає їм впевнено викладати не переживаючи, що щось піде не так.

Тут наприклад: урок може відбуватися в Zoom, домашнє завдання — через електронні пошти або месенджерів, а тестування — в інших програмах. Це є надзвичайно незручно, бо виникають труднощі як для викладача, так і для учня а, коли ще одночасно працюється із декількома групами або студентами, то це вже стає катастрофою.

Хоч у цих додатках є недоліки, але вони мають багато переваг:

✓ Ті хто вже закінчив навчання у школі можуть навчатися з будь-де все що їм треба - мати комп'ютер або телефон з Інтернетом.

✓ Онлайн-лекції можна легко підібрати до певного розкладу учня чи вчителя. Це дозволяє поєднати навчання з роботою або іншими справами.

✓ Через ці додатки можна різний рівень складності та форму навчання конкретно для якогось учня.

✓ Багато додатків дозволяють використовувати діаграми, презентації, відео та щось подібне на це.

Введенням комп'ютерних додатків для навчання онлайн вже зробило щось

дивовижне, але більша частина наявних програм розробилася або на масові курси, або для широкого застосування. Репетитори мають велику потребу у виборі додатку для введення свого навчання, який дозволить їм так само проводити свої уроки не маючи обмежень, що економитиме як час репетитора, так і учня. Тому створення моєї програми має вирішити цю проблему для репетиторів.

1.3. Аналіз конкурентів

Найпопулярнішими платформами конкурентами є Google Classroom та Moodle.

1. Google Classroom

Це є застосунок, який дозволяє створювати класи, призначати завдання, перевіряти роботи та спілкуватися зі студентами. Базова версія цього додатку є доступна безкоштовно для всіх користувачів, але більші функції Classroom надаються тільки в платних версіях додатку, які закуповуються закладами освіти (див. рис.1.4). Це робить цю платформу дуже зручною як для шкіл, так і для вишів для дистанційного навчання, але є обмеженим для репетиторства. Платформа дозволяє працювати з Google Drive, Docs, Sheets, Meet та іншими додатками Google.

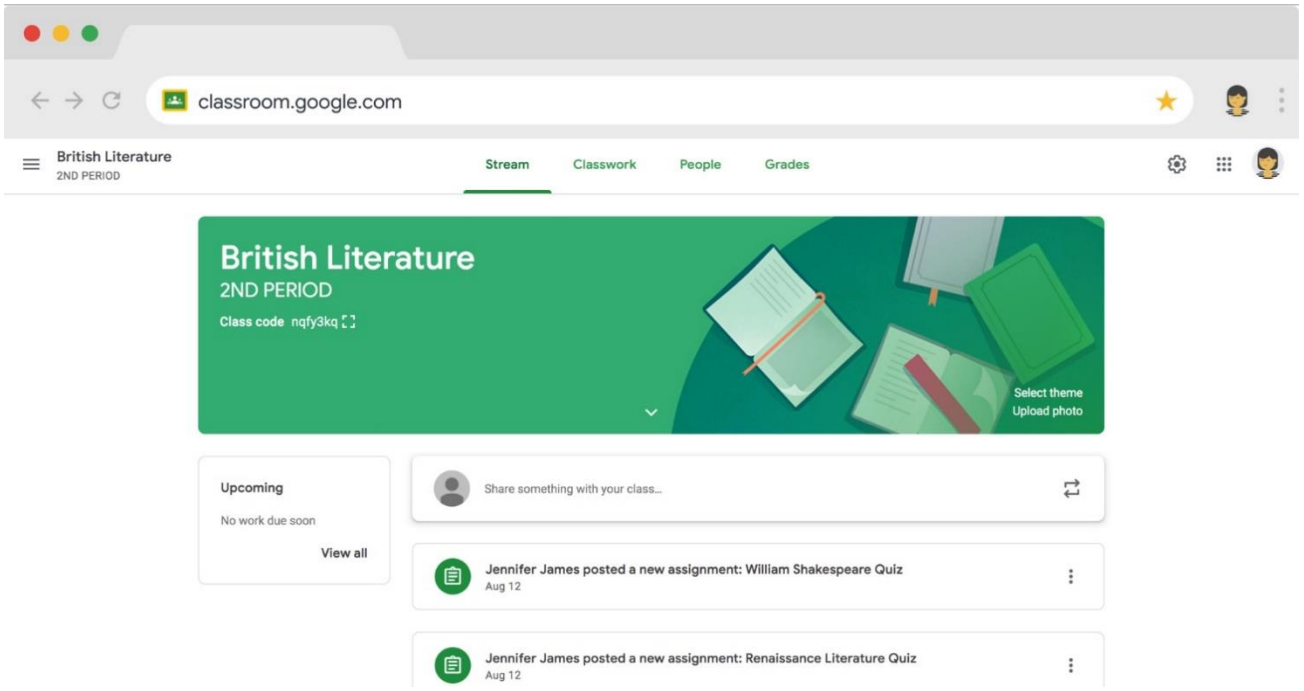

Рисунок 1.4 – Інтерфейс Google Classroom

Google Classroom є простим у використані, має зрозумілий інтерфейс, він швидко інтегрується з Google-документами та відеозв'язком через Meet, також цей застосунок містить безкоштовний пакет використання, але його не є достатньо для того, щоб повноцінно проводити навчання. Завдяки цьому застосунку викладачі можуть опубліковувати матеріали, завдання, виставляти оцінки та проводити онлайн-зустрічі та спілкуватися зі студентами, використовуючи коментарі до матеріалів та завдань.

Цей застосунок має і свої недоліки: він орієнтований на заклади освіти (школи, коледжі, університети) викладачі не можуть створювати свої власні курси для репетитерства, а можуть працювати лише в межах своїх закладів, де вони отримали доступ до цього додатку, вони можуть використати безкоштовну версію, але цього є мало, багато функцій додатку стають не доступними у цьому випадку. Також ще одним недоліком є те, що всі файли завантажені у класрум зберігаються на гугл диску облікового запису користувача, який використовує класрум як у викладачів всі їхні файли завантажені у цей застосунок зберігаються на диску, так і у самих студентів так само, тобто пам'ять на диску є обмеженою, і щоб збільшити її розміри треба купувати додаткове місце, відповідно, можна не помітити, як таким способом пам'ять може заповнитися повністю.

2. Moodle

Це є система управління навчанням з відкритим кодом, яка дозволяє створювати повноцінні онлайн-курси з великою кількістю налаштувань та можливостей (див. рис.1.5). Її часто використовують у вишах. Платформа є дуже функціональною, але є і дуже складною в налаштуванні.

Moodle має багато переваг: можна створювати матеріали, розділи для завантаження робіт студентами, створення тестування зразу є вбудованими у цей застосунок, також можна у журналі виставити підрахунок балів, так як потрібно саме викладачу.

Проте, тут є багато недоліків: немає комунікації між студентом та викладачем, всі матеріали, які завантажуються сюди забивають пам'ять на сервері

вишу, а виш має дуже багато спеціальностей в цей же час, також в ньому важко зразу зорієнтуватися, є деякі плутанини саме через оформлений графічний інтерфейс, ще одним недоліком є те, що тут так само викладачі, не можуть створювати свої курси в цілях репетитерства (виші цього не дозволять).

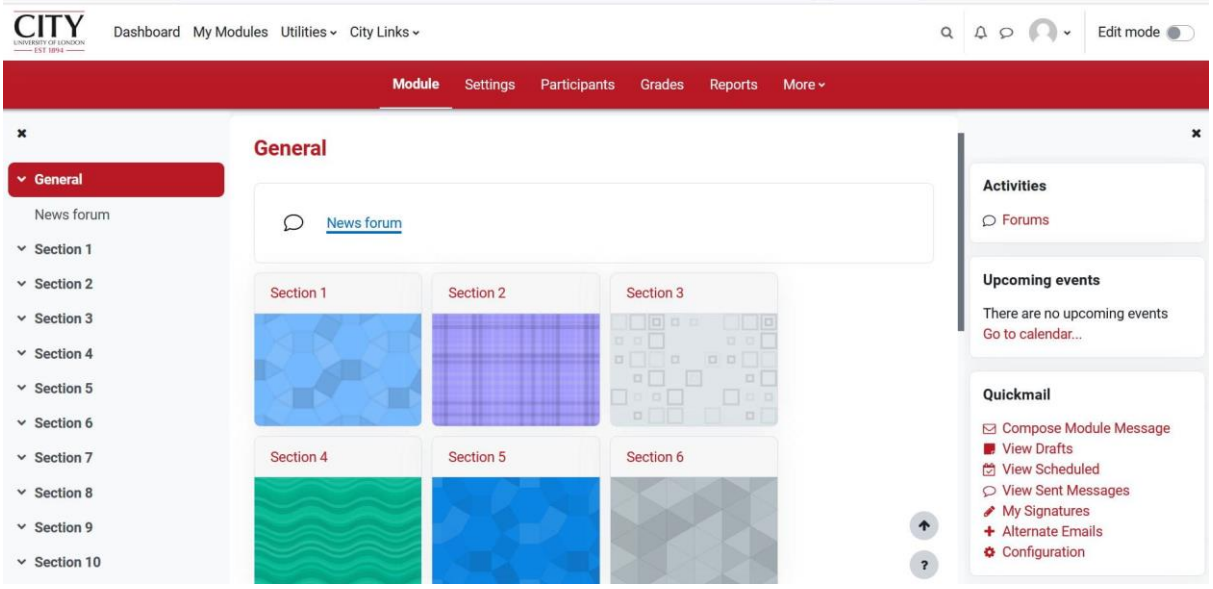

Рисунок 1.5 – Інтерфейс Moodle

Отже, обидва конкуренти добре працюють у сфері дистанційного навчання. Хоч вони і мають багато переваг, але в цей же час вони мають і свої недоліки, які є майже схожими в обидвох, але Google Classroom в порівнянні має менше недоліків, ніж Moodle, також ці два додатки не повністю відповідають потребам індивідуального репетитора.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Проектування бази даних

Для того, щоб забезпечити зберігання всіх важливих даних у новій програмі, які забезпечуватимуть функціональність програми та надаватимуть дані, які були внесені саме до конкретного елемента у програмі, саме в той момент, коли вони будуть необхідними, потрібно створити базу даних, яка реалізує ці всі етапи.

Найчастіше бази даних застосовують у комп'ютерних додатках, тоді, коли потрібно забезпечити зберігання певної інформації. Зберігання даних у базі даних спрощує структурування інформації та дозволяє здійснювати різноманітні операції з даними, такі як додавання, оновлення та вилучення. Крім того, бази даних забезпечують можливість виконувати складні запити до даних, фільтрувати та сортувати їх для отримання необхідної інформації.

Завдяки базам даних можна відтворювати збережену інформацію, працювати з нею та на основі неї робити якісь певні дослідження та на основі них робити певні статистики, наприклад: на основі збережених всіх оцінок користувачів у програмі можна порахувати їхній середній бал у будь-який момент часу, оскільки всі їхні бали за завдання були збережені у базі даних і зберігаються надалі аж допоки сам адміністратор бази даних не зітре їх звідти.

Для того, щоб реалізувати збереження та обробку даних у застосунку "SparksNet" було розглянуто одразу декілька можливих варіантів систем управління базами даних. Основна метою було — вибрати саме таку базу даних, яка буде в першу чергу надійною, швидкою, зручною у використанні разом з мовою програмування Java, а також буде підтримувати здійснення віддаленого підключення до неї через хостинг.

SQLite, згідно Freehost [4] є легкою вбудованою базою даних, якій не потрібно окремого сервера. Вона є ідеальним варіантом саме для невеликих проєктів або тих, які будуть працювати локально. Проте в цьому випадку програма "SparksNet" передбачає роботу з дуже великою кількістю користувачів, тестів, матеріалів, завдань, рекомендацій, оголошень, повідомлень і курсів, а також

потребує з’єднання з віддаленого серверу. Через це SQLite не вибралася для реалізації цього додатку.

PostgreSQL є потужною реляційною базою даних, яка підтримує складні запити, великі об’єми даних та високий рівень безпеки. Вона є чудовим вибором для корпоративних проєктів. Однак для простішої інтеграції з Java та через те, що PostgreSQL має трохи складніші налаштування для віддаленого доступу на хостингу, було вирішено не обирати її для реалізації проєкту.

MySQL є дуже популярна, стабільна, безкоштовна реляційна база даних. Вона зручно інтегрується з Java, підтримується на більшості хостингів і дозволяє працювати з великими об’ємами даних. Крім того, у MySQL можна керувати таблицями, зв’язками, індексацією, що є дуже важливим для ефективної роботи навчального застосунку.

Після вибору бази даних, далі потрібно побудувати таблиці бази даних, з якими працюватиме програма. Розпочнемо з побудови таблиці «Teachers» для зберігання даних про викладачів програми (див. рис.2.1).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id** 🔑	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**first_name**	varchar(60)	utf8mb4_unicode_ci		Ні	*Немає*		
3	**last_name**	varchar(60)	utf8mb4_unicode_ci		Ні	*Немає*		
4	**country**	varchar(55)	utf8mb4_unicode_ci		Ні	*Немає*		
5	**image**	longblob			Так	*NULL*		
6	**login**	varchar(65)	utf8mb4_unicode_ci		Ні	*Немає*		
7	**password**	varchar(65)	utf8mb4_unicode_ci		Ні	*Немає*		

Рисунок 2.1 – Структура таблиці «Teachers»

Teachers містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати конкретного викладача, воно є автоінкрементом (тобто число встановлюється саме по-порядку).
- **first_name** – текстове поле для зберігання ім’я викладача.
- **last_name** – текстове поле для зберігання прізвища викладача.
- **country** – текстове поле для зберігання країни викладача.
- **image** – поле для зберігання зображення викладача.

- **login** – текстове поле для зберігання логіну викладача.
- **password** – текстове поле для зберігання ім’я викладача.

Перед тим як формувати таблиці завжди потірбно знати, яку структуру даних буде мати таблиця (табл. 2.1).

Таблиця 2.1 – Приклад структури даних

Id	Name_course	Code	Color	Author_id
1	Java	3ZLJPOZA12	#804D80	1
2	C#	11X3IDO824	#FFCCE6	3
3	C++	Y5WP3YE815	#4D1A4D	5

Таблиця «Student» має таку саму структуру, як і в таблиці «Teachers», тут розділення виконується для того, щоб не було хаосу між даними студентів та викладачів у програмі.

Далі побудуємо таблицю «Courses» для зберігання даних про всі курси викладачів у програмі (див. рис.2.2).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**name**	varchar(70)	utf8mb4_unicode_ci		Ні	*Немає*		
3	**image**	longblob			Ні	*Немає*		
4	**code**	varchar(70)	utf8mb4_unicode_ci		Ні	*Немає*		
5	**color**	varchar(70)	utf8mb4_unicode_ci		Ні	*Немає*		
6	**teacher_id**	int(11)			Ні	*Немає*		

Рисунок 2.2 – Структура таблиці «Courses»

Courses містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати курс серед всіх інших, воно є автоінкрементом.
- **name** – текстове поле для зберігання назви курсу.
- **image** – поле для зберігання зображення курсу.
- **code** – текстове поле для зберігання коду курсу.
- **color** – текстове поле для зберігання кольору панелі курсу.
- **teacher_id** – зовнішній ключ (показуватиме які курси є пов’язані з певним id викладачем).

Таблиця «Studentcourses» містить у собі id, та два зовнішні ключі, які зв’язують цю таблицю з таблицями course та student. Ця таблиця потрібна для того, щоб можна було побачити в яких курсах зареєстрований певний студент, щоб потім можна було відображати всі курси, у яких зареєстрований певний студент, на його панелі відображення всіх курсів.

Наступною побудуємо таблицю «Materials», у якій зберігатимуться дані про матеріали курсів (див. рис.2.3).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**name**	text	utf8mb4_unicode_ci		Ні	*Немає*		
3	**text**	text	utf8mb4_unicode_ci		Так	*NULL*		
4	**course_id**	int(11)			Ні	*Немає*		

Рисунок 2.3 – Структура таблиці «Materials»

Materials містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати матеріал від інших, воно є автоінкрементом.
- **name** – текстове поле для зберігання назви матеріалу.
- **text** – текстове поле для зберігання тексту матеріалу (воно може бути NULL, тобто зовсім не мати його).
- **course_id** – зовнішній ключ (з’єднює записи цієї таблиці з id таблиці курсів, щоб ідентифікувати до якого курсу належить певний матеріал).

Далі створимо таблицю «Assignments» для зберігання інформації про завдання, які будуть надавати викладачі своїм студентам (див. рис.2.4).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**name**	text	utf8mb4_unicode_ci		Ні	*Немає*		
3	**text**	text	utf8mb4_unicode_ci		Ні	*Немає*		
4	**deadline**	datetime			Так	*NULL*		
5	**score**	double			Ні	0		
6	**course_id**	int(11)			Ні	*Немає*		

Рисунок 2.4 – Структура таблиці «Assignments»

Assignments містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна

було ідентифікувати завдання від інших, воно є автоінкрементом.

- **name** – текстове поле для зберігання назви завдання.
- **text** – текстове поле для зберігання тексту завдання (воно може бути NULL, тобто може зовсім не бути).
- **deadline –** поле для зберігання кінцевої дати здачі завдання (може бути NULL, тобто завдання може бути без терміну здачі).
- **score –** поле для зберігання максимального балу за завдання
- **course_id** – зовнішній ключ (з'єднює записи цієї таблиці з id таблиці курсів, щоб ідентифікувати до якого курсу належить певне завдання).

Тепер створимо таблицю «Works», у якій зберігатимуться дані роботи, які здавали студенти, і також вона зберігатиме оцінки, які виставить викладач після перевірки певного завдання (див. рис.2.5).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**assignment_id**	int(11)			Ні	*Немає*		
3	**student_id**	int(11)			Ні	*Немає*		
4	**link**	text	utf8mb4_unicode_ci		Ні	*Немає*		
5	**date**	datetime			Ні	*Немає*		
6	**score**	double			Так	-1		

Рисунок 2.5 – Структура таблиці «Works»

Works містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати здану роботу від інших, воно є автоінкрементом.
- **assignment_id** – зовнішній ключ (зв'язок з таблицею assignments, щоб можна була побачити до якого саме завдання було здана робота).
- **student_id –** зовнішній ключ (зв'язок з таблицею student, щоб ідентифікувати студента, який здав завдання).
- **link –** текстове поле для того, щоб зберегти посилання на здане завдання студентом(студенти надсилають посилання на папку, де вони розміщують всі файли виконаного завдання).
- **date –** поле для зберігання дати здачі завдання певним студентом.
- **score –** поле для зберігання оцінки за здане завдання.

Наступною створимо таблицю «Tests» для зберігання інформації про тестування (див. рис.2.6).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**name**	text	utf8mb4_unicode_ci		Ні	*Немає*		
3	**start**	datetime			Так	current_timestamp()		
4	**end**	datetime			Так	*NULL*		
5	**attempts**	int(11)			Ні	*Немає*		
6	**max_questions**	int(11)			Ні	*Немає*		
7	**max_score**	int(11)			Так	*NULL*		
8	**time**	time			Так	*NULL*		
9	**viewing**	int(11)			Ні	0		
10	**course_id**	int(11)			Ні	*Немає*		

Рисунок 2.6 – Структура таблиці «Tests»

Tests містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати тестування від інших, воно є автоінкрементом.
- **name** – текстове поле для зберігання назви тесту.
- **start –** поле для зберігання початкової дати початку тестування.
- **end –** поле для зберігання кінцевої дати початку тестування.
- **attempts –** поле для зберігання кількості дозволених спроб на тест.
- **max_questions –** поле для зберігання максимальної кількості питань у тесті
- **max_score –** поле для зберігання максимальної оцінки за тест.
- **time –** поле для зберігання часу який виведений на проходження тесту.
- **viewing** – поле для вказування чи студенти можуть переглядати свої результати тестування.
- **course_id –** зовнішній ключ.

Тепер створимо таблицю «Test_questions», який зберігатиме інформацію про питання до певного тесту (див. рис.2.7).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**test_id**	int(11)			Ні	*Немає*		
3	**question_text**	text	utf8mb4_unicode_ci		Ні	*Немає*		
4	**running_number**	int(11)			Так	*NULL*		
5	**max_score**	int(11)			Ні	*Немає*		
6	**type**	text	utf8mb4_unicode_ci		Ні	*Немає*		
7	**is_mandatory**	tinyint(4)			Ні	*Немає*		

Рисунок 2.7 – Структура таблиці «Test_questions»

Test_questions містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати питання від інших, воно є автоінкрементом.
- **test_id** – зовнішній ключ (зв'язок з таблицею tests, щоб можна була побачити до якого саме тесту належить запитання).
- **question_text –** поле для зберіганння тексту питання.
- **running_number –** поле для зерігання порядку питання у тесті
- **max_score –** поле для зберігання максимального балу за поточне питання.
- **type –** поле для зберігання типу запитання
- **is_mandatory –** поле для того, щоб позначати чи питання є обов'язковим.

Визначимо структуру для таблиці «Test_answers», у якій зберігатимуться всі можливі відповіді на запитання та позначатимуться які з них є правильними, а які ні (див. рис.2.8).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**question_id**	int(11)			Ні	*Немає*		
3	**answer_text**	text	utf8mb4_unicode_ci		Так	*NULL*		
4	**match_pair**	text	utf8mb4_unicode_ci		Так	*NULL*		
5	**is_correct**	tinyint(1)			Так	*NULL*		

Рисунок 2.8 – Структура таблиці «Test_answers»

Test_answers містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати відповідь від інших, воно є автоінкрементом.
- **question_id** – зовнішній ключ (зв'язок з таблицею tests_questions, щоб можна була побачити до якого саме питання тесту належить відповідь).
- **answer_text –** поле тексту відповіді (може бути NULL).
- **match_pair –** поле тексту для питання типу відповіді (може бути NULL).
- **is_correct –** поле для позначення чи відповідь є правильною.

Тепер створимо таблицю «Tests_results», у якій зберігатимуться результати за тестування кожним студентом, який пройшов певний тест (див. рис.2.9).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**test_id**	int(11)			Ні	*Немає*		
3	**student_id**	int(11)			Ні	*Немає*		
4	**score**	int(11)			Ні	*Немає*		
5	**max_score**	int(11)			Ні	*Немає*		

Рисунок 2.9 – Структура таблиці «Tests_results»

Tests_results містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати відповідь студента від інших, воно є автоінкрементом.
- **test_id** – зовнішній ключ (зв'язок з таблицею tests, щоб можна було побачити до якого саме тесту належить результат студента).
- **student_id** – зовнішній ключ (зв'язок з таблицею student, щоб можна було побачити до якого саме тесту належить результат студента).
- **score –** поле, яке показуватиме бал який отримав студент за весь тест.
- **max_score –** поле, яке показує, який бал є максимальним за весь тест.

Далі створимо таблицю «Test_student_results», у якій зберігатимуться відповіді на тестування, так як кожен студент відповідав на кожне запитання, для того, щоб потім можна було реалізувати можливість перегляду того, як студент пройшов тестування (див. рис.2.10).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**question_id**	int(11)			Ні	*Немає*		
3	**answer**	text	utf8mb4_unicode_ci		Ні	*Немає*		
4	**pair**	text	utf8mb4_unicode_ci		Так	*NULL*		
5	**score**	int(11)			Ні	*Немає*		
6	**student_id**	int(11)			Ні	*Немає*		
7	**tests_results_id**	int(11)			Ні	*Немає*		

Рисунок 2.10 – Структура таблиці «Test_student_results»

Test_student_results містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати відповідь студента від інших, воно є автоінкрементом.
- **question_id** – зовнішній ключ (зв'язок з таблицею tests_questions, щоб можна була побачити до якого саме питання тесту належить відповідь студента).

- **answer -** поле для збереження відповідей, які надав студент до поточного питання тесту.
- **pair –** поле з відображенням того, яку пару обрав студент до питання з відповідністю.
- **score –** поле для збереження балу, який отримав студент за питання.
- **student_id** – зовнішній ключ (зв’язок з таблицею student, щоб можна було побачити до якого саме відповідь на питання тесту належить студенту).
- **tests_results_id** – зовнішній ключ (зв’язок з таблицею tests_results, щоб можна було побачити до якого відповідь на питання тесту належить до результату студента).

Таблиці «Accessassignments», «Accessmaterial», «Accesstest» мають однакову структуру, тільки у них міняються зовнішні ключі з різними таблицями. Вони мають первинний ключ id та два зовнішні ключі, один це є зв’язок з таблицею студентів, щоб можна було вкзати які саме студенти не будуть мати доступ, і другий зовнішній ключ є зв’язок з таблицею компонента(завдання, матеріал, тест), який забезпечить ідентифікацію до якого саме компонента студент немає доступу.

Тепер створимо таблицю «Announcements», щоб зберігати інформацію про оголошення в курсі, які робить викладач (див. рис.2.11).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**course_id**	int(11)			Ні	*Немає*		
3	**text**	text	utf8mb4_unicode_ci		Ні	*Немає*		
4	**time**	timestamp			Ні	current_timestamp()		

Рисунок 2.11 – Структура таблиці «Announcements»

Announcements містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати оголошення курсу від інших, воно є автоінкрементом.
- **course_id** – зовнішній ключ (зв’язок з таблицею courses, щоб можна була побачити до якого саме курсу належить оголошення).
- **text –** поле для збереження тексту оголошення.
- **time –** поле для збереження часу оприлюднення оголошення.

Таблиця «Meetings» призначена для того, щоб зберегти посилання на зустріч, яке додаватиме викладач. Ця таблиця містить id первинний ключ, поле для збереження посилання та зовнішній ключ з таблицею courses, щоб можна було ідентифікувати до якого курсу належить посилання.

Наступною побудуємо таблицю «Messages», за допомогою якої буде відбуватися збереження повідомлень з чатів між студентом та викладачем та саме завдяки цій таблиці буде відображатися переписка між ними (див. рис.2.12).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**teachers_id**	int(11)			Ні	*Немає*		
3	**student_id**	int(11)			Ні	*Немає*		
4	**course_id**	int(11)			Ні	*Немає*		
5	**type_sender**	varchar(50)	utf8mb4_unicode_ci		Ні	*Немає*		
6	**message**	varchar(80)	utf8mb4_unicode_ci		Ні	*Немає*		
7	**sent_at**	timestamp			Ні	current_timestamp()		
8	**is_read**	tinyint(1)			Ні	0		

Рисунок 2.12 – Структура таблиці «Messages»

Message містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати повідомлення від інших, воно є автоінкрементом.
- **teachers_id** – зовнішній ключ (зв’язок з таблицею teachers, щоб можна було побачити до якого саме викладача належить повідомлення).
- **student_id** – зовнішній ключ (зв’язок з таблицею student, щоб можна було побачити до якого саме студента належить повідомлення).
- **course_id** – зовнішній ключ (зв’язок з таблицею courses, щоб можна була побачити до якого саме курсу належить повідомлення).
- **type_sender** – поле для збереження типу користувача, який надіслав повідомлення.
- **message –** поле для збереження тексту повідомлення.
- **sent_at –** поле для збереження часу, коли було надіслано повідомлення.
- **is_read –** поле для показування того чи повідомлення прочиталося.

Останньою створимо таблицю «Recommendation», яка зберігатиме інформацію про рекомендацію всіх курсів (див. рис.2.13).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	**course_id**	int(11)			Ні	*Немає*		
3	**text**	text	utf8mb4_unicode_ci		Ні	*Немає*		
4	**link**	text	utf8mb4_unicode_ci		Так	*NULL*		

Рисунок 2.13 – Структура таблиці «Recommendation»

Recommendation містить наступні поля:

- **id** – поле використовується для первинного ключа таблиці, щоб можна було ідентифікувати рекомендацію від інших, воно є автоінкрементом.
- **course_id** – зовнішній ключ (зв’язок з таблицею courses, щоб можна була побачити до якого саме курсу належить рекомендація).
- **text –** поле для зберігання тексту рекомендації.
- **link –** поле для зберігання посилання на детальні дані курсу.

Загальний вигляд ER-діаграми з відображенням всіх таблиць та зв’язків між ними у базі даних цієї програми наведено у додатку А.

2.2. Структура програмних компонентів

2.2.1. Діаграма класів

Діаграма класів, згідно з Mermaid [5] — це один із типів діаграм, що найчастіше використовується в об'єктно-орієнтованому проектуванні певного програмного забезпечення. Вона призначена для графічного зображення структури системи класів додатку.

На діаграмі класів відображаються: поля, конструктори, методи та відображаються зв’язки між класами. На діаграмі класів відображаються всі класи проєкту для того, щоб показати абсолютно всі класи які використовуються у програмних застосунках.

Це дозволяє побачити, як побудована система зсередини, які класи взаємодіють між собою, а також яка між ними ієрархія. Таке візуальне представлення сильно спрощує саму розробку, обслуговування та навіть у майбутньому дозволить модернізувати розроблений програмний продукту.

Так, як застосунок створюється за допомогою JavaFX, більша частина класів

є контролерами, тобто тут кожен з цих контролерів відповідає за працездатність якогось окремого вікна графічного інтерфейсу користувача. Всі вони містять певну свою логіку для обробки подій, перевірки введених даних, взаємодії з базою даних.

Створення графічних застосунків з використанням JavaFX потребує наявності класів- контролерів для кожного вікна програми, тобто кожне вікно – це окремий клас-контролер, у якому вказаний весь алгоритм дій цього вікна, який працюватиме, тоді, коли відкриється це вікно у програмному застосунку.

Окрім самих класів-контролерів, у проєкті є допоміжні класи (наприклад, клас Tools), що реалізовує загальні методи для повторного використання, а також кілька об'єктно-орієнтованих класів, які використовуються для роботи з результатами студентів по тестах чи іншими функціями.

На рисунку нижче (див. рис.2.14) представлено загальний вигляд діаграми класів із прихованими деталями, тобто без показування полів, конструкторів, методів, щоб уникнути перевантаження зображення та забезпечити початкову зрозумілість того які саме класи використовуються для роботи програми та які класи пов’язані між собою. Повний варіант з розгорнутими класами наведено у додатку Б.

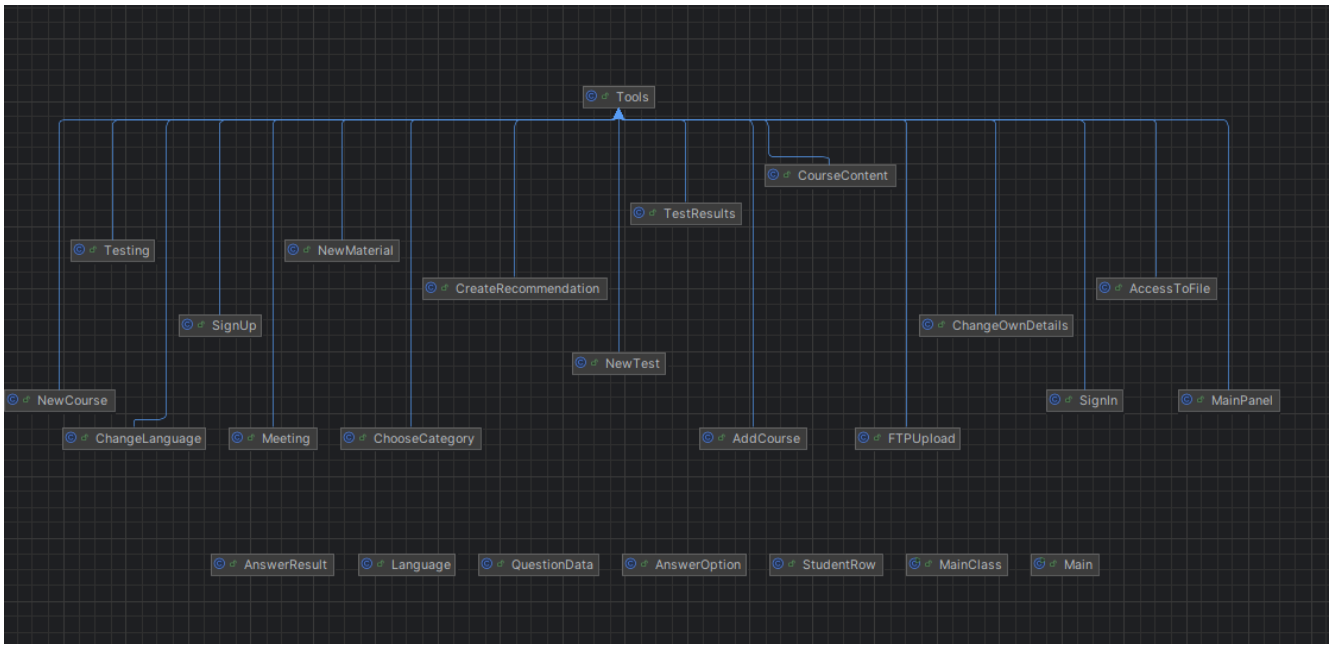

Рисунок 2.14 – Загальний вигляд діаграми класів програми з прихованими деталями

2.2.2. Use case діаграма

Use Case діаграма варіантів використання є однією з дуже важливих інструментів для здійснення комунікації між розробниками та зацікавленими особами. Вона дозволяє дуже швидко отримати загальне уявлення про поведінку застосунку, не дивлячись у технічні деталі розробки програми. Її використання допомагає виявити відсутні або лишні функції програми, що дозволяє спростити наш проєкт і покращити план майбутніх етапів розробки програми. Вона описує зовнішніх акторів (викладач і студент) та варіанти дій (див. рис.2.15, додаток В).

Викладач застосунку може: зареєструватись у застосунку, створити курс, редагувати чи видаляти курс, видаляти студентів з курсу, також може переглядати учасників курсу, публікувати матеріали, завдання та тестування, оцінювати та переглядати результати учасників, ще може надсилати особисті повідомлення студенту, публікувати новини та рекомендацію свого курсу.

Студент у цій програмі може: зареєструватися у застосунку, добавитися до курсів викладачів, видаляти себе з курсу, також може переглядати матеріали, надсилати виконане завдання та проходити тестування, може переглядати свою оцінку у курсу, ще може надсилати особисті повідомлення викладачу курсу, переглядати новини та рекомендації курсів.

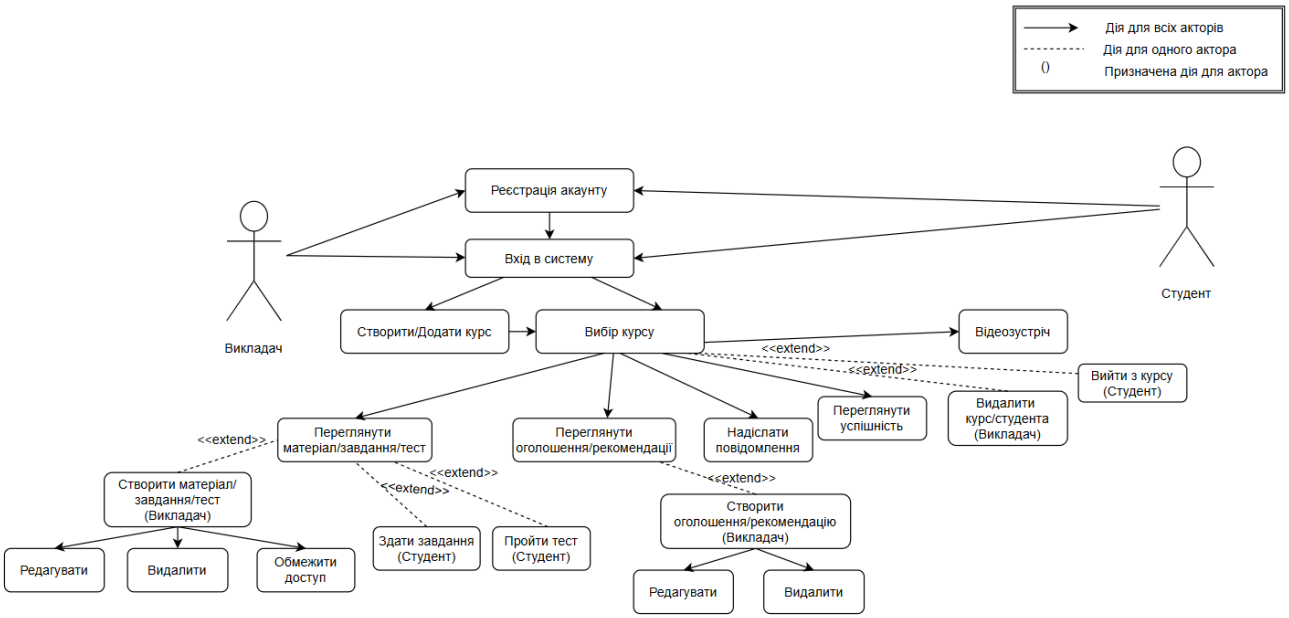

Рисунок 2.15 – Use case діаграма

2.2.3. Архітектура застосунку

Застосунок реалізований з використанням архітектурної моделі MVC (Model-View-Controller), згідно Javarush [6] за допомогою нього відбувається поділ відповідальностей між компонентами всієї програми. Така модель допомагає спрощувати процес розробки самого застосунку та його тестування.

Тут у цій архітектурі є Model (Модель), яка відповідає за доступ до даних та бізнес-логіку програми. У програмі тут частково представлена допоміжними класами, такими як клас, який формує об’єкт відповіді на тест, також клас `Tools`, який містить всі загальні методи, що використовуються практично у всіх класах програми.

Компонент View (Уявлення), він забезпечує тут у програмі графічний інтерфейс користувача. Для цього використовується JavaFX, яка дозволяє створювати графічний інтерфейс для взаємодії користувача з програмою за допомогою Fxml файлів.

Компонент Controller (Контролер) тут реалізований у вигляді окремих класів-контролерів, які містять у собі алгоритм за допомогою якого здійснює свою роботу певне вікно, тобто у цих класах описується все що може відбутися у вікні при роботі користувача з ним. Кожна сцена в JavaFX має свій окремий контролер, який відповідає за її логіку.

Застосунок має клієнт-серверну архітектуру, програма взаємодіє з базою даних, що розміщена на віддаленому сервері. Для обміну інформацією використовується api.php, у якому мовою PHP реалізовано підключення до бази даних та дії з нею. Ще додатково у програмі використовується FTP-сервер для збереження файлів, які викладач додає в курс.

Ця архітектура дозволяє досягти масштабованості та дозволяє розділити відповідальності між модулями.

Графічне представлення цієї використаної архітектури у цій програмі подано у додатку Г.

2.3. Вибір віддалених хостингів

Для того, щоб забезпечити роботу застосунку у віддаленому режимі, а також для організації доступу до файлів та бази даних, використовуються хостингові сервіси. Це дозволяє розгорнути програму та надає гнучкість, масштабованість та доступність до програми з будь-якого комп'ютера, все що тільки потрібно це наявність інтернету.

Файли які завантажують викладачі у програмі у матеріалах, завданнях та тестуваннях зберігаються за допомогою хостингу zzz.com.ua, який надає можливість користувачам мати FTP-доступ до файлового сховища (див. рис.2.16).

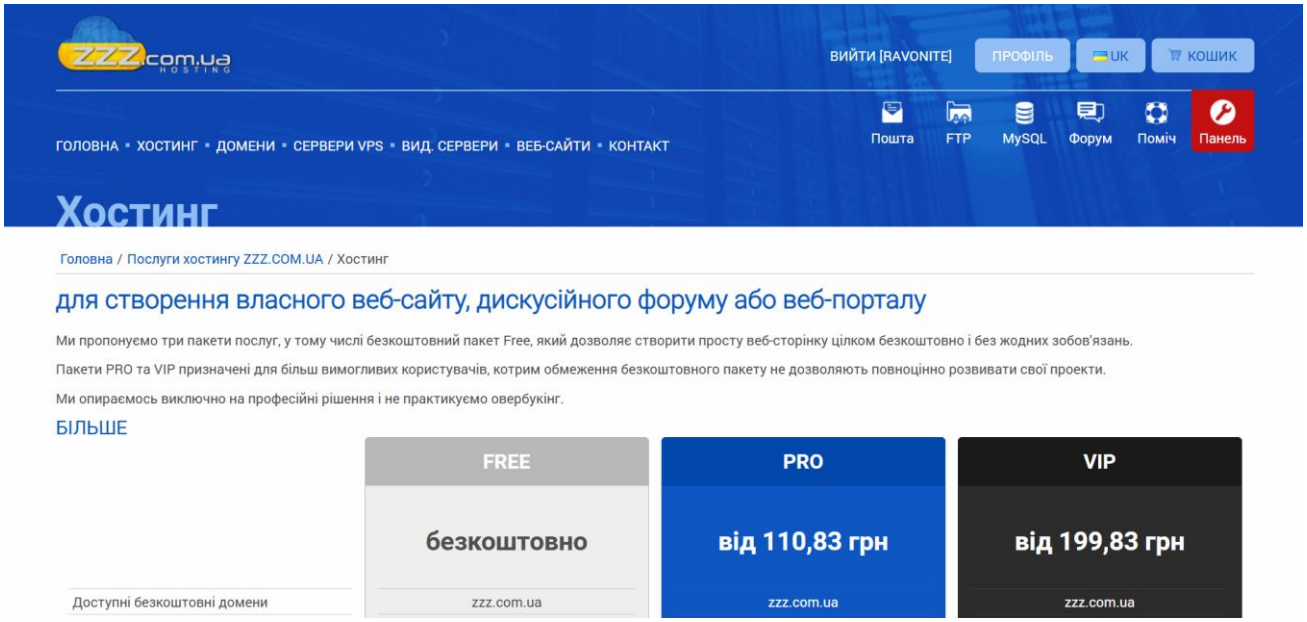

Рисунок 2.16 – Хостинг zzz.com.ua

Він має багато переваг:

- можна зручно керувати файлами через самий FTP-клієнт;
- має безкоштовні та платні тарифи, навіть платні є не дуже дорогими;
- він підтримує PHP, MySQL та надає доступ до панелі управління;
- він надає технічну підтримку та надає можливості докуповувати ресурси, якщо проєк буде зростати.

Для розміщення бази даних використовується хостинг-платформа Хостинг Україна (ukraine.com.ua), яка пропонує дуже великий набір інструментів для роботи з реляційними базами даних (див. рис.2.17), наприклад: він має MySQL, яка і використовується у цьому випадку.

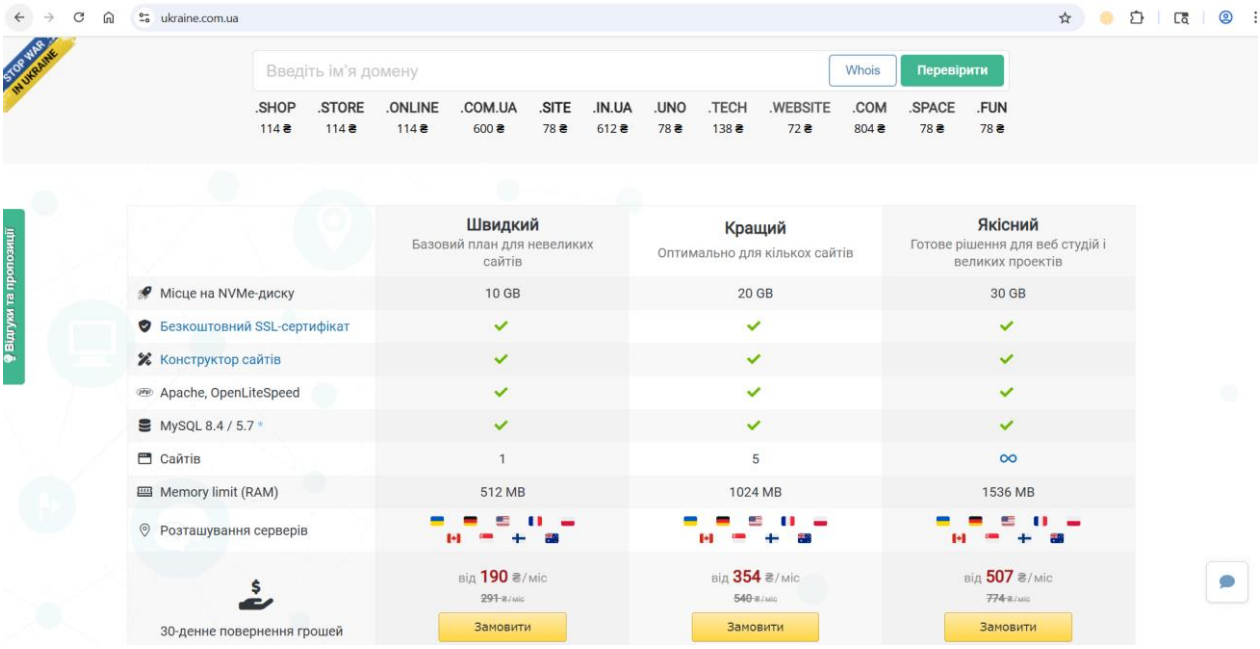

Рисунок 2.17 – Хостинг ukraine.com.ua

Хостинг Україна (ukraine.com.ua) має свої переваги:

✓ він підтримує створення та адміністрування баз даних через phpMyAdmin;

✓ він надає можливість створювати віддалене з'єднання з базою даних із Java-додатку;

✓ цей хостинг має дуже потужний сервер, що впливатиме на продуктивність системи, забезпечуючи велику швидкість у програмі.

Використання одразу двох хостингів дозволяють не перевантажувати один і той самий сервер, що є певною страховкою для працездатності програми.

2.4. Алгоритмічне забезпечення

Одним із важливих елементів алгоритмічного забезпечення є захист паролів користувачів програми. Для цього використовується клас BCryptPasswordEncoder, з бібліотеки Spring Security. Згідно Spring [7] його використання дозволяє безпечно зберігати паролі у базі даних у вигляді хешів, а не у відкритому вигляді, що може спричинити викрадення паролів, бо вони всі навиду. Хешування виконується з використанням криптографічного алгоритму BCrypt.

Важливою частиною інтерфейсу є алгоритм визначення розмірів вікна застосунку. Ширина вікна встановлюється рівною ширині екрана користувача, а висота обчислюється за формулою (2.1), що враховує необхідність залишити

вільним простір для панелі завдань операційної системи. Таким чином, висота вікна зменшується приблизно на 7% від повної висоти екрана.

$$H_w = H_s \times 0.93, \quad (2.1)$$

де H_w — визначена висота екрану, H_s – загальна висота екрану операційної системи комп'ютера.

Унікальні коди для кожного окремого курсу формуються за формулою (2.2), щоб мінімізувати ймовірність повторення коду, оскільки кожен викладач і кожен курс мають унікальні ідентифікатори.

$$C = R_8 + TID + CID, \quad (2.2)$$

де R_8— випадковий рядок з 8 символів (великі літери англійського алфавіту та цифри 0–9), TID — id викладача у застосунку, CID — id курсу в таблиці Courses.

Для визначення кількості студентів у курсі потрібно підрахувати усіх студентів, які є в курсі за формулою (2.3).

$$N = \sum_{i=1}^{n} 1, \quad (2.3)$$

де N — загальна кількість студентів у курсі, n — кількість студентів у курсі, 1 — означає, що до загальної кількості додається по 1 за кожного студента.

Після обчислення кількості студентів у кожному курсі значення сортуються за спаданням за формулою (2.4) для визначення позиції поточного курсу в рейтингу курсів.

$$N1, N2, \dots, Nm \Rightarrow N[1] \geq N[2] \geq \cdots \geq N[m], \quad (2.4)$$

де m — загальна кількість курсів, $N_1,\dots,N_m$ — кількість студентів у кожному курсі, N[i] — і-те число після сортування (найбільше — перше, найменше — останнє).

Визначення місця курсу у рейтингу за кількістю студентів відбувається за формулою (2.5).

$$R_k = 1+count(N_j>N_k), \text{ для } j=1..m, j\neq k \quad (2.5)$$

де R_k — місце курсу k у рейтингу, N_k — кількість студентів у курсі k, N_j — кількість студентів у курсі j, $count(N_j>N_k)$— підраховує, скільки курсів мають більше студентів, ніж курс k, додається 1, щоб перше місце відповідало найвищій кількості студентів.

Середній результат кожного студента за курс визначається у кілька етапів. На першому етапі розраховується загальна кількість балів, які студент набрав за всі завдання. Це визначається за формулою (2.6).

$$S = \sum_{i=1}^{n} S_i, \tag{2.6}$$

де S — загальна кількість балів, які набрав студент, Si — оцінка студента за і-те завдання, n — загальна кількість завдань.

Далі визначається сума максимально можливих балів, які студент міг би набрати за всі завдання. Цей розрахунок виконується за формулою (2.7)

$$M = \sum_{i=1}^{n} M_i, \tag{2.7}$$

де M — сума максимальних балів за всі завдання, Mi — максимальний бал за і-те завдання, n — кількість завдань.

Після цього середній результат студента визначається як відсоткове відношення набраних балів до максимально можливих. Остаточне значення обчислюється за формулою (2.8).

$$A = \left(\frac{S}{M}\right) \times 100, \tag{2.8}$$

де A — середній бал студента у відсотках, S — загальна кількість балів, які набрав студент, M— сума максимальних балів за всі завдання.

Для зручного відображення середніх балів використовується округлення значення до двох знаків після коми. Цей процес виконується у два етапи: спочатку значення множиться на 100 і округлюється до цілого числа за формулою (2.9).

$$B = round(A \cdot 100), \tag{2.9}$$

де A — середній бал студента у відсотках, B — середній бал, округлений до цілого після множення на 100, round() - функція округлення до найближчого цілого числа.

Після округлення числа, воно ділиться на 100 за формулою (2.10), щоб отримати середній бал з двома знаками після коми.

$$A_{\text{окр}} = \frac{B}{100}, \tag{2.10}$$

де $A_{\text{окр}}$ — середній бал, округлений до двох знаків після коми, B — середній бал, округлений до цілого після множення на 100.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

3.1.1. Мова програмування Java

Це одна з найпопулярніших мов програмування серед розробників у багатьох країнах світу, вона дуже швидко розвивається і зараз є однією з найтоповішою серед інших мов програмування.

Java, згідно О.М. Васильєва [8] — це мова програмування загального призначення, яка була створена компанією Sun Microsystems ще у 1995 року. Її використовують для розробки різних програм, вебсервісів, ігор. На цій мові програмування можна створити як консольні застосунки, так і графічні, графічні у свою чергу можна створювати за допомогою Java Swing або JavaFX.

Java підходить для початківців, зараз існує дуже багато навчальних матеріалів за допомогою яких можна реально вивчити її, і вже буквально за декілька місяців можна створювати свої перші консольні програми. Крім того, вона точно не втратить актуальність ще тривалий час, бо багато програм не є тільки написаними на мові Java, а вони ще й є підв'язані під платформу для запуску Java Virtual Machine і набір для розробки Java Development Kit.

Переваги Java:

- є популярною у багатьох країнах світу;
- має велику спільноту;
- має дуже розвинену екосистему;
- є об'єктно-орієнтованою і суворо типізована мовою
- не є складною у під дач навчання;
- перевірена часом.

Недоліки Java:

- є популярною тільки у бекенд-розробці;
- синтаксис мови є несучасним;
- сильна мова завдяки своїм бібліотекам;
- дуже велика кількість програм використовує стару версію мови.

Зараз Java містить нескінченну кількість бібліотек і фреймворків для виконання різних завдань у програмуванні як складних, так і легких. На Java вже написано дуже багато великих програм. Java постійно розвивається, щоб бути актуальною та тримати свою популярність, що і змушує розробників постійно вчитися та розвиватися у цій мові програмування дізнаючись щось нове для себе.

3.1.2. Мова структурованих запитів SQL

Мова структурованих запитів SQL, згідно aCode [9] - це мова за допомогою якої відбувається збереження, отримання, видалення та оновлення у таблицях бази даних. Тут ця мова відображає дані в таблиці, які складаються з рядків та стовпців.

Використовувавши цю мову можна: створити таблиці та у її структурі задати: назву стовппів, тип даних, опис, значення за замовчуванням та обмеження. Ще можна встановлювати, оновлювати і видаляти дані з бази даних. У ній можна шукати дані в таблицях за допомогою різних методів. Можна дозволяти доступ до бази даних певним учасникам або забороняти доступ, також ця мова дозволяє генерувати індекси.

3.1.3. Середовище розробки IntelliJ IDEA

IntelliJ IDEA - це інтегрованим середовищем розробки мовою програмування Java. Він є зручним середовищем розробки для Java. Це середовище має зручні інструменти для автодоповнення коду(див. рис.3.1).

Переваги IntelliJ IDEA:

✓ велика кількість функцій.

✓ підтримує різні інші системи (Git, Maven).

✓ можна налаштувати середовище розробки під як захочеться.

✓ можна автоматично виправляти помилки.

Недоліки IntelliJ IDEA:

- вимагає багато ресурсів системи.
- повністю всі функції доступні тільки у платній версії.
- може бути складним через велику кількість функцій та опцій.

IntelliJ IDEA є одним з дуже популярних середовищ розробки, у якій можна створити свою програму на мові Java як консольну версію так і графічну версію програми.

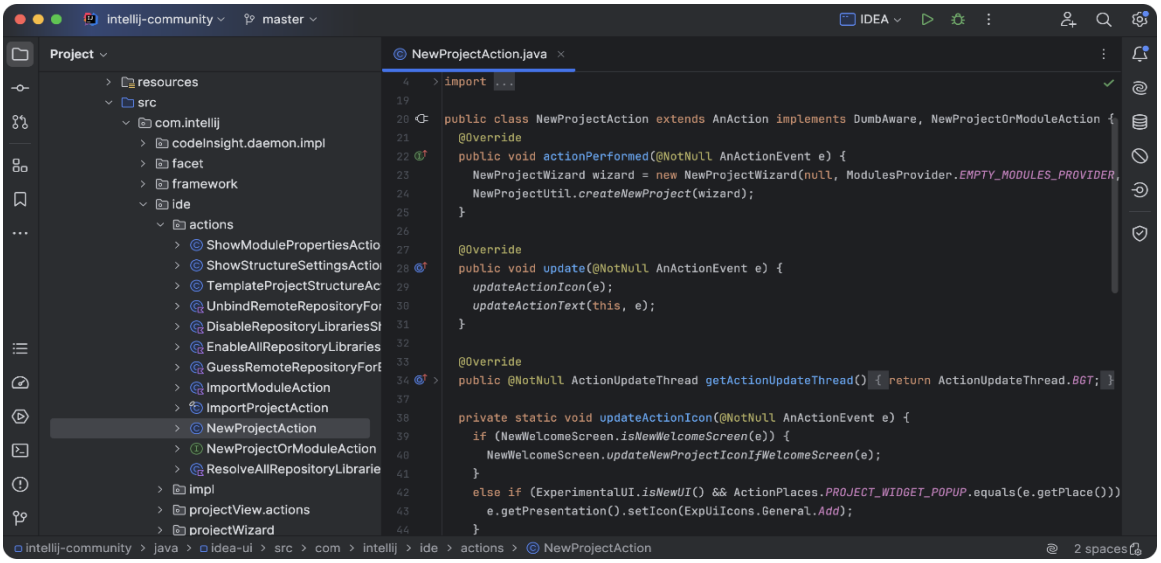

Рисунок 3.1 – Інтерфейс IntelliJ IDEA

3.1.4. Візуальний редактор Scene Builder

Scene Builder — це візуальний редактор, який можна використовувати для створення графічних інтерфейсів у JavaFX. Завдяки ньому можна створювати вікна програм перетягуючи компоненти на вікно за допомогою властивостей, які мають ці компоненти можна встановлювати шрифт, розмір, фон (див. рис.3.2). Тут, коли використовується він, то вже не потрібно писати код вручну.

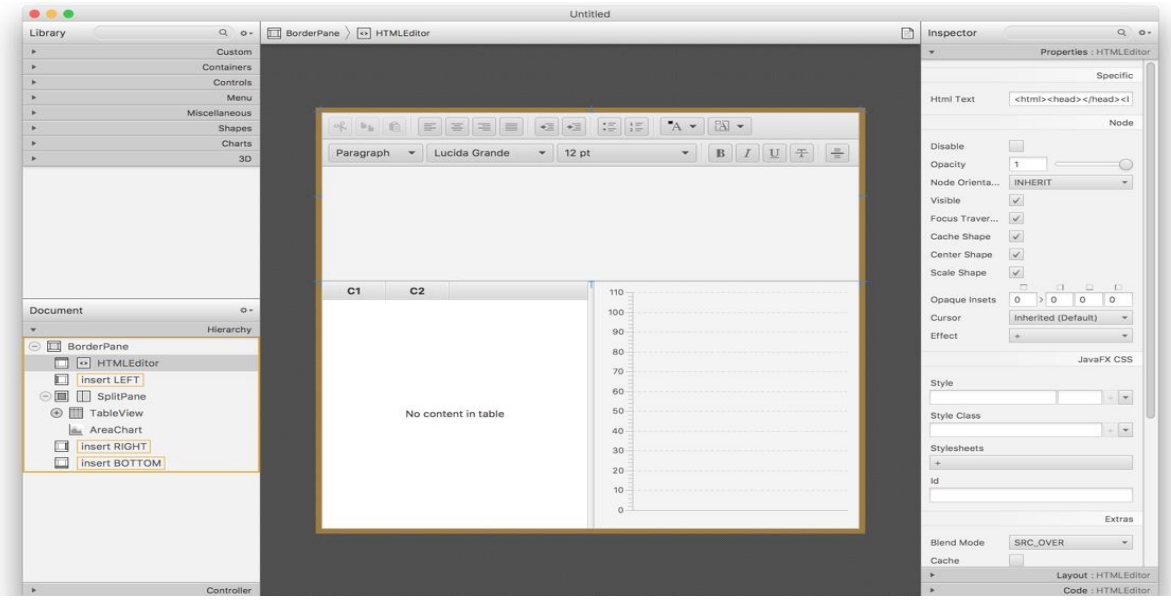

Рисунок 3.2 – Інтерфейс Scene Builder

Тут у ньому можна автоматично створювати FXML-код вікна, змінювати стилі вибраних компонентів за допомогою CSS і потім надати певний алгоритм , який працюватиме через контролери Java.

3.1.5. Система управління базами даних phpMyAdmin

phpMyAdmin — це безкоштовний програмний інструмент за допомогою якого можна працювати з базою даних MySQL, він має дуже зручний графічний інтерфейс за допомогою якого можна легко та швидко встановити, оновити дані або здійснити їх пошук (див. рис.3.3).

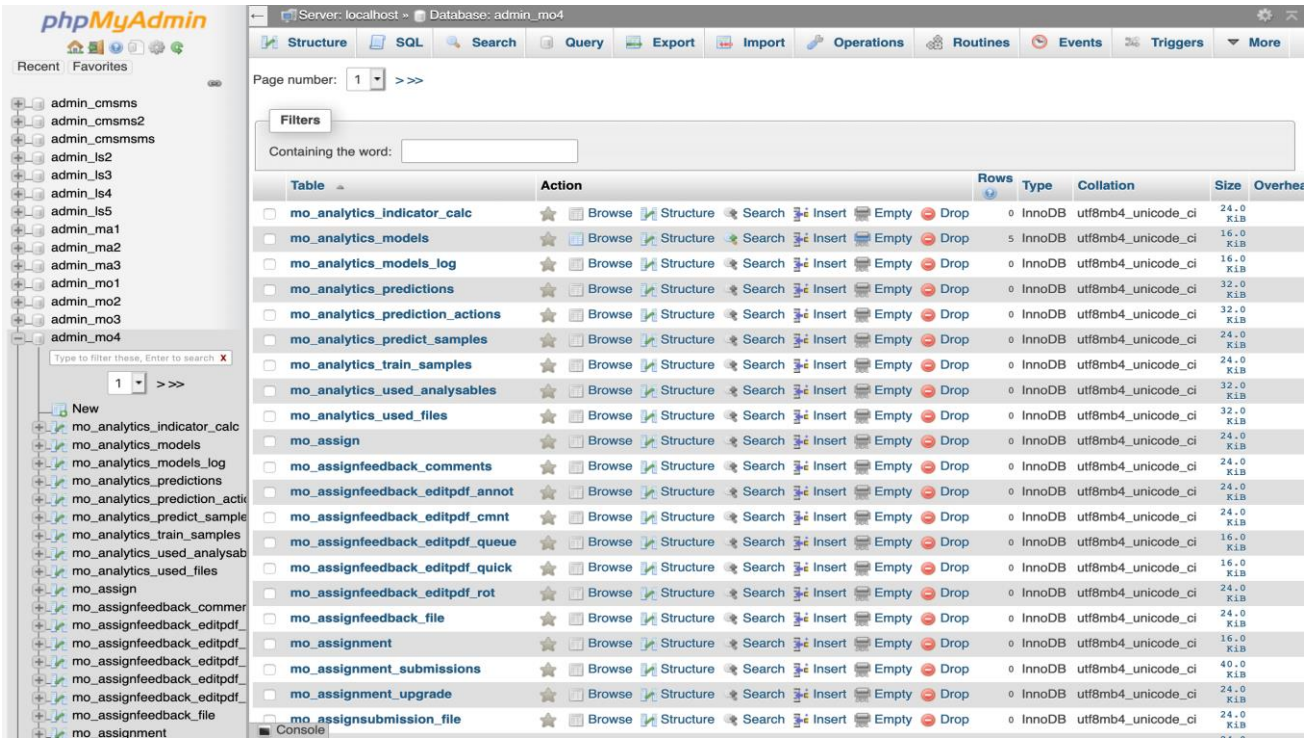

Рисунок 3.3 – Інтерфейс phpMyAdmin

Він є легкий у використанні, він підтримує всі функції MySQL, можна також створювати процедурами та тригери та керувати ними в базі даних, також ще можна здійснювати імпорт та експорт даних з бази даних, він може бути легко налаштований і розгорнутий на різних серверах та віддалених хостингах.

3.2. Вимоги до технічного та програмного забезпечення

Для реалізації програми використати JavaFx у якому будувати вікна за документацією згідно Openjfx [10].

Для успішної роботи з нашою програмою необхідно врахувати наступні вимоги до технічного та програмного забезпечення:

Вимоги до пристрою:

- Операційна система: Windows 10 / 11, Linux, macOS;
- Підключення до інтернету: має бути стійким, щоб все працювало добре.

Вимоги до розробника:

- Операційна система: Windows, MacOS, Linux;
- Програмне забезпечення: IntelliJ IDEA, Scene Builder, phpMyAdmin, веб хостинги;
- Доступ до інтернету.

Застосунок повинен включати такі можливості:

- Реєстрацію та авторизацію викладача/студента;
- Відображення всіх курсів до яких залучений користувач;
- Створювати/Додавати курси;
- Переглядати вміст курсів;
- Додавати/Редагувати матеріали, завдання, тести або проходити їх;
- Відображати успішність студентів;
- Спілкуватися з викладачем та студентом через чат;
- Створювати/Переглядати оголошення та рекомендації;
- Видаляти дані курсу, там де є потреба;
- Змінювати мову інтерфейсу;
- Переглядати/Видаляти зареєстрованих студентів у курсі;
- Обмежувати доступ до елементів курсу студентам;
- Організовувати відео зустріч;
- Відображати загальну статистику скільки студентів є у курсі та яке це місце за кількістю студентів за всіма наявними іншими курсами у програмі.

3.3. Опис програмної реалізації

3.3.1. Основні класи та методи застосунку

Одним з основних класів програми для проведення онлайн-курсів «SparksNet» є клас «Tools», він містить в собі методи, які використовують інші класи для забезпечення функціональності програми. Одним з таких методів є метод «setScreenSize», який визначає розміри екрану для того, щоб потім підлаштувати розміри вікна під розміри екрану комп'ютера користувача:

```
private void setScreenSize(){
    if(widthScreen==0) {
        Rectangle2D screenSize = Screen.getPrimary().getBounds();
        widthScreen = screenSize.getWidth();
        heightScreen = screenSize.getHeight() * 0.93;
    }
}
```

Ще одним з таких методів цього класу є метод newWindow(), який забезпечує відкриття вікон програми:

```
public void newWindow(String title, String fxml,Button btn){
    try {
        setScreenSize();
        FXMLLoader loader = new FXMLLoader(Main.class.getResource(fxml));
        Scene scene = new Scene(loader.load(), widthScreen, heightScreen);
        Stage stage = new Stage();
        stage.setTitle(title);
        stage.setScene(scene);
        stage.getIcons().add(new
Image(getClass().getResource("logo_icon.png").toExternalForm()));
        stage.show();
    }catch (Exception e){
        alertWindow("Error","ERROR!", Alert.AlertType.ERROR);
    };
}
```

У цьому ж класі метод «alertWindow» виконує важливу роль, а саме відображення повідомлення у програмі у вигляді діалогових вікон:

```
public void alertWindow(String title, String text, Alert.AlertType alertType){
    Alert alert= new Alert(alertType);
    alert.setHeaderText(null);
    alert.setContentText(text);
    alert.setTitle(title);
    alert.getDialogPane().setStyle(
            "-fx-font-family: 'Cambria';" +
                    "-fx-font-size: 16px;" +
                    "-fx-background-color: linear-gradient(to bottom right,
```

```
#2c142d, #231025);"
    );
    alert.getDialogPane().lookup(".content").setStyle("-fx-text-fill: white;");
    alert.showAndWait();
}
```

Метод «isFieldEmpty» перевіряє чи всі необхідні поля вікна, де користувач має ввести якісь дані, не були пустими

```
public boolean isFieldEmpty(TextField... fields) {
    for (TextField field : fields) {
        if (field.getText().trim().isEmpty()) {
            return true;
        }
    }
    return false;
}
```

Для того, щоб з'єднатися з базою даних, у моїй програмі використовується api, який створений у файлі api.php в ньому вказано всі команди для того, щоб з’єднатися з базою даних і вказані так як має оброблятися запити, єдине, що надсилається у цей файл з програми це значення за яким відбувається запити і надсилається також його назва цього запиту, який потім за цієї назвою шукається у цьому файлі. З’єднання з базою даних через через api.php відбувається за допомогою методу «callApi»:

```
public String callApi(Map<String, String> params) throws IOException {
        StringBuilder postData = new StringBuilder();
        for (Map.Entry<String, String> param : params.entrySet()) {
            if (postData.length() != 0) postData.append('&');
            postData.append(URLEncoder.encode(param.getKey(), "UTF-8"));
            postData.append('=');
            postData.append(URLEncoder.encode(param.getValue(), "UTF-8"));
        }
        byte[] postDataBytes =
postData.toString().getBytes(StandardCharsets.UTF_8);
        URL url = new URL(API_URL);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
        conn.setDoOutput(true);
        conn.getOutputStream().write(postDataBytes);
        BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream(), "UTF-8"));
```

```
        StringBuilder response = new StringBuilder();
        String line;
        while ((line = in.readLine()) != null) {
            response.append(line);
        }
        in.close();
        return response.toString();
}
```

Наступним одним з головних класів програми є клас «SignUp» за допомогою якого забезпечується реєстрація викладача або студента у програмі. Він має методи обробник події натискання на кнопки реєстрації, і цей обробник викликає у собі інші два класи: progressCreate() та insertUser(). Тут перший з цих методів забезпечує спочатку перевірку чи часом вже не існує у програмі користувача з тим самими логіном, у випадку якщо не існує, то викликається insertUser() метод, який забезпечує реєстрацію користувача у програмі:

```
@FXML
void onSignInClick(ActionEvent event) {
        if(isFieldEmpty(txtFirstName, txtLastName, txtCountry, txtLogin,
txtPasword) || cmbCategoria.getValue()==null){
            alertWindow("Error", Language.get("Message.emptyFields"),
Alert.AlertType.ERROR);
        }else {
            new Thread(()->progressCreate()).start();
        }
}

private void progressCreate(){
        startBlocker(timeline,prBar,blockPane);
        String categoria= cmbCategoria.getValue().equals("Teacher")
?"teachers":"student";
        Map<String, String> params = new HashMap<>();
        params.put("action", "check_user_exists");
        params.put("login", txtLogin.getText());
        params.put("category", categoria);
        params.put("token", "sparksnet253");
        try {
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            if (json.getBoolean("success") && json.getInt("rowCount") > 0) {
                Platform.runLater(() -> {
                    stopBlocker(timeline, prBar, blockPane);
                    alertWindow("Error", Language.get("Message.loginExists"),
Alert.AlertType.ERROR);
```

```
                });
            } else {insertUser(categoria);}
        } catch (Exception e) {
            connectionGettingDataError(timeline, prBar, blockPane);
        }
}

private void insertUser(String categoria) throws IOException {
        String imageBase64 = (imgForDatabase != null) ?
imageToBase64(imgForDatabase) : "";
        Map<String, String> params = new HashMap<>();
        params.put("action", "register_user");
        params.put("first_name", txtFirstName.getText());
        params.put("last_name", txtLastName.getText());
        params.put("country", txtCountry.getText());
        params.put("login", txtLogin.getText());
        params.put("password", hashCodePassword(txtPasword.getText()));
        params.put("image", imageBase64);
        params.put("category", categoria);
        params.put("token", "sparksnet253");
        callApi(params);
        Tools.rememberLanguage(cmbLanguage);
        Platform.runLater(() -> {
            stopBlocker(timeline, prBar, blockPane);
            animationSubButton("SparksNet", "signIn.fxml", btnSignUp, player);
        });
}
```

Ще одним з важливих класів програми є клас «SignIn», який забезпечує авторизація викладача/студента в програмі. У цьому класі метод «handlerSignInData()» забезпечує перевірку чи правильно користувач ввів авторизаційні дані, саме тут відбувається надсилання запиту у api бази даних. Результати цього запиту обробляються в методі «getDataUserProcess()», а перевірка на правильність паролю та відкриття наступного вікна, у випадку успішної авторизації, відбувається у методі «processLoginResult()»:

```
private void handlerSignInData(){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "login_user");
            params.put("login", txtLogin.getText());
            params.put("category", categoria);
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
```

```
            getDataUserProcess(json);
        }catch (Exception e){
            connectionGettingDataError(timeline,prBar,blockPane);
        }
}

private void getDataUserProcess(JSONObject json){
        if (json.getBoolean("success") && json.getInt("rowCount") > 0) {
            JSONObject user = json.getJSONArray("rows").getJSONObject(0);
            String storedPassword = user.getString("password");
            int id = user.getInt("id");
            byte[] image = user.has("image") &&
!user.getString("image").isEmpty()?Base64.getDecoder().decode(user.getString("im
age")) : null;
            Platform.runLater(() -> processLoginResult(storedPassword, id,
image));
        } else {
            Platform.runLater(() -> {
                stopBlocker(timeline, prBar, blockPane);
                alertWindow("Error", Language.get("Message.NoLogin"),
Alert.AlertType.ERROR);
            });
        }
}

private void processLoginResult(String storedPassword, int idRes, byte[] image)
{
        if (checkPassword(txtPasword.getText(), storedPassword)) {
            id = idRes;
            CourseContent.imgIcon = image;
            openWindow();
        } else {
            stopBlocker(timeline,prBar,blockPane);
            alertWindow("Error", Language.get("Message.wrongPassword"),
Alert.AlertType.ERROR);
        }
}
```

У класі «MainPanel», який забезпечує відображення у панелі користувача всі його курси до якого він належить, міститься метод «fetchCoursesFromDatabase()», який поверне інформацію про всі курси до якого належить користувач, а метод «loadCourses()» забезпечить виклик цього попереднього методу та, отримавши ці курси, він викличе метод «createCoursePane()», який відобразить у панелі всі курси у яких зарахований користувач.

```
private void fetchCoursesFromDatabase() throws SQLException, IOException {
        Map<String, String> params = new HashMap<>();
        params.put("action", "get_user_courses");
        params.put("user_id", String.valueOf(id));
        params.put("category", category);
        params.put("token", "sparksnet253");
        String response = callApi(params);
        JSONObject json = new JSONObject(response);
            if (json.getBoolean("success")) {
                JSONArray rows = json.getJSONArray("rows");
                for (int i = 0; i < rows.length(); i++) {
                    JSONObject rowJson = rows.getJSONObject(i);
                    Object[] row = new Object[4];
                    row[0] = rowJson.getInt("id");
                    row[1] = rowJson.getString("name");
                    String base64Image = rowJson.optString("image", null);
                    row[2] = base64Image != null ?
Base64.getDecoder().decode(base64Image) : null;
                    row[3] = rowJson.getString("color");
                    courses.add(row);
                }
            } else {
                connectionGettingDataError(null, null, null);
        }
}
private void loadCourses() {
        new Thread(() -> {
            try {
               fetchCoursesFromDatabase();
                Platform.runLater(() -> {
                    for (Object[] row : courses) {
                        int courseId = (int) row[0];
                        String courseName = (String) row[1];
                        byte[] imageData = (byte[]) row[2];
                        String color= (String) row[3];
                        StackPane pane = createCoursePane(courseId, courseName,
imageData, color);
                        flowPane.getChildren().add(pane);
                        animationCourse(pane);
                    }
                });
            } catch (Exception e) {
                connectionGettingDataError(null, null, null);
            }
        }).start();
}
```

```
private StackPane createCoursePane(int course_id,String name,byte[]
imageData,String color){
        StackPane pane = new StackPane();
        pane.setPrefSize(350, 200);
        pane.setStyle("-fx-border-radius: 50; -fx-background-radius: 50; -fx-
overflow: hidden;");
        ImageView imageView = createImage(imageData);
        Label label = createLabel(name);
        VBox vBox = new VBox(0, imageView, label);
        pane.getChildren().add(vBox);
        pane.setOnMouseClicked(e->{
       transferCourseInfo(course_id,imageView.getImage(),label.getText(),color);
            backgroundMusic("components/Choose_course.mp3");
            animationOpenCourse(pane);
        });
        pane.setOnMouseEntered(e->{
            pane.setEffect(new Glow(0.15));
        });
        pane.setOnMouseExited(e->{
            pane.setEffect(null);
        });
        return pane;
}
```

Клас «NewCourse» забезпечує створення курсу, він містить метод «addingCourseData()», який забезпечує формування запиту до файлу api.php для роботи з базою даних. Метод «startExecuteInsert()» вже забезпечує створення самого курсу:

```
private void addingCourseData(){
        if (teacherNewIdCourse != 0) {
            String code = generateCourseCode() + MainPanel.id +
teacherNewIdCourse;
            String color = colorToHex(selectColor.getValue());
            String imageBase64 =
Base64.getEncoder().encodeToString(imgForDatabase);
            Map<String, String> params = new HashMap<>();
            params.put("action", "add_course");
            params.put("name", txtName.getText());
            params.put("image", imageBase64);
            params.put("code", code);
            params.put("color", color);
            params.put("teacher_id", String.valueOf(MainPanel.id));
            params.put("token", "sparksnet253");
            startExecuteInsert(params);
```

```
        }else {
            Platform.runLater(()->stopBlocker(timeline,prBar,blockPane));
        }
}

private void startExecuteInsert(Map<String, String> params){
        try {
            callApi(params);
            Platform.runLater(() -> {
                stopBlocker(timeline, prBar, blockPane);
                animationSubButton("SparksNet", "mainPanel.fxml", btnCreate,
player);
            });
        } catch (Exception e) {
            connectionGettingDataError(timeline, prBar, blockPane);
        }
}
```

Клас «FTPUpload» забезпечує завантаження файлів, які додаються викладачами у програмі, на хостинг в FTP. Він містить метод «uploadFile», який встановлює з'єднання з ftp хостинга, і у своєму ж тілі він викликає метод «uploading», який здійснює завантаження файлів у ftp хостингу:

```
public void uploadFile(File localFile,int courseId,int idMaterial,String letter)
throws IOException {
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect(SERVER, PORT);
            ftpClient.login(USERNAME, PASSWORD);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
            uploading(localFile,ftpClient,courseId,idMaterial,letter);
        } catch (IOException ex) {
            alertWindow("Error", "Can't connect with server!",
Alert.AlertType.ERROR);
        } finally {
            ftpClient.logout();
            ftpClient.disconnect();
        }
}

public void uploading(File localFile,FTPClient ftpClient, int courseId, int
idMaterial,String letter) throws IOException {
        FileInputStream inputStream = new FileInputStream(localFile);
        String fileName = localFile.getName();
        int dotIndex = fileName.lastIndexOf('.');
        String name = fileName.substring(0, dotIndex);
```

```
        String ext = fileName.substring(dotIndex + 1);
        ftpClient.storeFile(REMOTE_DIR + "/" +courseId+ "/"+
name+"_"+letter+idMaterial+"."+ext, inputStream);
}
```

Ще одним з важливих класів програми є клас «CourseContent», у якому здійснюється всі дії над курсом: перегляд, створення матеріалів/завдань/тестувань, перегляд успішності студентів, здійснення комунікації між студентом та викладачем через чат, перегляд, створення оголошень та інше. Тут, наприклад, показ всіх чатів відбувається за допомогою методу «gettingChatsInfo()», який задає запит на отримання даних з бази даних, викликаючи у себе метод «createChat()» відбувається додавання чату у панель за отриманою інформацією з бази даних, метод «buildChats()» будує зам вигляд чату і додає його на панель:

```
private void gettingChatsInfo(){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_chats");
            params.put("user_type",MainPanel.category);
            params.put("user_id", String.valueOf(MainPanel.id));
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            createChat(json.getJSONArray("rows"));
            Platform.runLater(() ->stopBlocker(timelinePrBar,prBar,blockPane));
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
}

private void createChat(JSONArray rows){
        for (int i = 0; i < rows.length(); i++) {
            JSONObject row = rows.getJSONObject(i);
            byte[] image = Base64.getDecoder().decode(row.getString("image"));
            String name=row.getString("first_name")+"
"+row.getString("last_name");
            int userId = row.getInt("user_id");
            int status = row.getInt("status");
            Platform.runLater(() -> buildChats(image, name, userId, status));
        }
}
```

```
private void buildChats(byte [] image,String name,int id,int status){
        Button button= new Button(name);
        ImageView imageView=createImage(image);
        button.setGraphic(imageView);
        button.setPrefSize(700,30);
        button.setFont(Font.font("Cambria", FontWeight.BOLD,18));
        button.setStyle("-fx-text-fill: white; -fx-background-color: linear-
gradient(to bottom right, #231025, #be823a, #231025);" +
                " -fx-background-radius:10; -fx-border-radius:10;");
        button.setUserData(id);
        button.setAlignment(Pos.CENTER_LEFT);
        button.setOnAction(e->{
            txtName.setText("");
            ImageView view= (ImageView) button.getGraphic();
            showChosenConversation(view.getImage(),button.getText(), (int)
button.getUserData());
        });
        if(status==1){
            button.setEffect(new Glow(0.5));
        }
        pnlAllChats.getChildren().add(button);
        allButtons.add(button);
}
```

Ще у класі є метод «createAnnouncementsFields()» за допомогою, якого відбувається відображення всіх оголошень курсу, які додає викладач. Після того, як дані про оголошення отрималися з бази даних, викликається метод «setAllAnnouncements()», який встановить у панель всі оголошення, а інший метод створює вигляд оголошення, додаючи до нього дані отриманні з бази даних:

```
private void createAnnouncementsFields(){
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            try{
                Map<String, String> params = new HashMap<>();
                params.put("action", "get_announcements");
                params.put("course_id", String.valueOf(idCourse));
                params.put("token", "sparksnet253");
                String response = callApi(params);
                JSONObject json = new JSONObject(response);
            setAllAnnouncements(setAllAnnouncements(json.getJSONArray("rows")));
            }catch (Exception e){
                connectionGettingDataError(timelinePrBar,prBar,blockPane);}
        }).start();
}
```

```
private List<Node> setAllAnnouncements(JSONArray rows){
        List<Node> announcementNodes = new ArrayList<>();
        for (int i = 0; i < rows.length(); i++) {
            JSONObject obj = rows.getJSONObject(i);
            String text = obj.getString("text");
            int id = obj.getInt("id");
            Timestamp time = Timestamp.valueOf(obj.getString("time"));
            Node node = createAnnouncementLabel(text, id, time);
            announcementNodes.add(node);
        }
        return announcementNodes;
}

private void setAllAnnouncements(List<Node> announcementNodes){
        Platform.runLater(()->{
            vBoxAnnouncements.getChildren().clear();
            vBoxAnnouncements.getChildren().addAll(announcementNodes);
            stopBlocker(timelinePrBar,prBar,blockPane);
        });
}
```

Далі одним з важливих методів є відображення матеріалів, завдань та тестів курсу. Це відбувається за допомогою методу «getMaterials()», який викликає метод «allDetailsElement()», що витягує дані про компонент з бази даних, і результати передаються у метод «addAllElement()», який додає всі наявні елементи певного компонента(матеріали, завдання, тести) у панель відображення всіх елементів, а метод «createMaterialPanel()» створює вигляд елементу у панелі, метод «getTableConfig()» визначає дані в залежності від того, чи має показатися матеріали чи завдання чи тести:

```
private void getMaterials(String tableName){
        Object[] config = getTableConfig(tableName);
        String table = (String) config[0];
        VBox vBox = (VBox) config[1];
        String nameColumn = (String) config[2];
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            allDetailsElement(tableName,table,vBox,nameColumn);
        }).start();
}

private Object[] getTableConfig(String tableName) {
        String table;
        VBox vBox;
        String nameColumn;
```

```
        if(tableName.equals("materials")){
            table = "accessmaterial";
            vBox = pnlAllMaterials;
            nameColumn = "material_id";
        } else if(tableName.equals("assignments")){
            table = "accessassignments";
            vBox = pnlAllAssignments;
            nameColumn = "assignment_id";
        } else {
            table = "accesstest";
            vBox = pnlAllTests;
            nameColumn = "test_id";
        }
        return new Object[]{table, vBox, nameColumn};
}

private void allDetailsElement(String tableName,String table, VBox vBox, String
nameColumn){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_all_details");
            params.put("course_id", String.valueOf(idCourse));
            params.put("table_name", tableName);
            params.put("table", table);
            params.put("name_column", nameColumn);
            params.put("user_id", String.valueOf(MainPanel.id));
            params.put("category", MainPanel.category);
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONArray array = new JSONArray(response);
            addAllElement(array,tableName,vBox);
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
}

private void addAllElement(JSONArray array,String tableName, VBox vBox){
        List<Node>nodes=new ArrayList<>();
        for (int i = 0; i < array.length(); i++) {
            JSONObject obj = array.getJSONObject(i);
            int materialId = obj.getInt("id");
            String name = obj.getString("name");
            nodes.add(createMaterialPanel(name, materialId, tableName));
        }
        Platform.runLater(()->{
            vBox.getChildren().addAll(nodes);
```

```
            stopBlocker(timelinePrBar, prBar, blockPane);
        });
}

private HBox createMaterialPanel(String name, int id, String tableName){
        ImageView imageView= new ImageView(new
Image(getClass().getResource(getPhotoToPanel(tableName)).toExternalForm()));
        imageView.setFitWidth(100);
        imageView.setFitHeight(75);
        Label label= new Label(name);
        label.setWrapText(true);
        label.setFont(Font.font("Cambria", FontWeight.BOLD, 20));
        label.setStyle("-fx-text-fill: white; ");
        HBox hBox=new HBox(2,imageView,label);
        hBox.setStyle("-fx-background-color: linear-gradient(to bottom right,
#be823a, #231025);" +
                "-fx-background-radius:20; -fx-border-radius:20");
        hBox.setAlignment(Pos.CENTER_LEFT);
        hBox.setPrefHeight(80);
        hBox.setUserData(id);
        hBox.setOnMouseClicked(e->{
            if(tableName.equals("materials")){
                showContentMaterial(name,id);

            }
            else if(tableName.equals("assignments")){
                showContentAssignment(name,id);
            }
            else {
                showContentTest(name,id);
            }
        });
        return hBox;
}
```

Відображення вмісту матеріалів, завдання та тестування відбувається за допомогою наступних методів, які забезпечують цю функцію у програмі, це є методи «showContentMaterial()», «showContentAssignment()», «showContentTest()»:

```
private void showContentMaterial(String name,int id){
        turnOffAllPanels();
        vBoxMaterialContent.setVisible(true);
        lblMaterialTitle.setText(name);
        flowPaneMaterial.getChildren().clear();
        currentMaterial=id;
        setTextMaterial(id);
        allMaterials(id,flowPaneMaterial,"");
}
```

```
private void showContentAssignment(String name,int id){
        turnOffAllPanels();
        panelDoublePanelAssignment.setVisible(true);
        lblAssignmentTitle.setText(name);
        flowPaneAssignment.getChildren().clear();
        currentMaterial=id;
        setTextAssignment(id);
        setPropertiesToContentAssignment();
        allMaterials(id,flowPaneAssignment,"a");
}

private void showContentTest(String name,int id){
        turnOffAllPanels();
        lblStartDate.setText(Language.get("CourseContent.lblStartDate"));
        lblEndDate.setText(Language.get("CourseContent.lblEndDate"));
        lblMaxAttempts.setText(Language.get("CourseContent.lblMaxAttempts"));
        lblTestLimit.setText(Language.get("CourseContent.lblTestLimit"));
        vBoxTestContent.setVisible(true);
        lblTestTitle.setText(name);
        pnlAllTestAttempts.getChildren().clear();
        currentMaterial=id;
        setTestProperties();
        if(MainPanel.category.equals("student")){
            btnStartTest.setDisable(true);
            new Thread(()->{
                isShowStartButton();
                setTextButtonViewing(false);
            }).start();
        }else {
            new Thread(()->setTextButtonViewing(true)).start();
        }
}
```

Створення матеріалів та завдань відбувається за допомогою класу «New Material», а створення тестів відбувається за допомогою класу «NewTest». Деталі коду цих класів та деталі решту коду класу «CourseContent» та інших класів знаходяться у додатку Д.

3.3.2. Перевірка працездатності застосунку

Для того, щоб упевнитися, що застосунок працює та все працює у програмі, так як планувалося, запускаємо виконання програми та здійснюємо її перевірку.

При запуску програми відривається початкове вікно з можливістю вибору дії у застосунку: зареєструватися або увійти (див. рис.3.4).

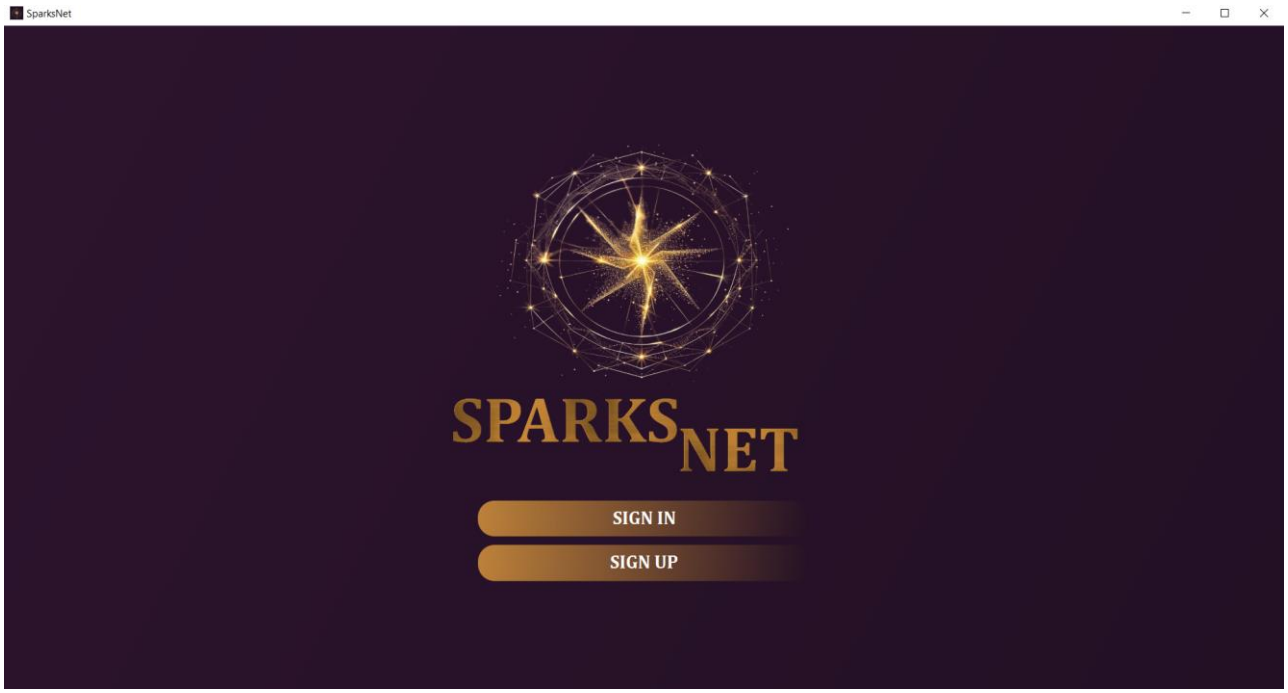

Рисунок 3.4 – Початкове вікно програми

При виборі реєстрації, відкривається вікно реєстрації, у яке вводимо дані про користувача (див. рис.3.5).

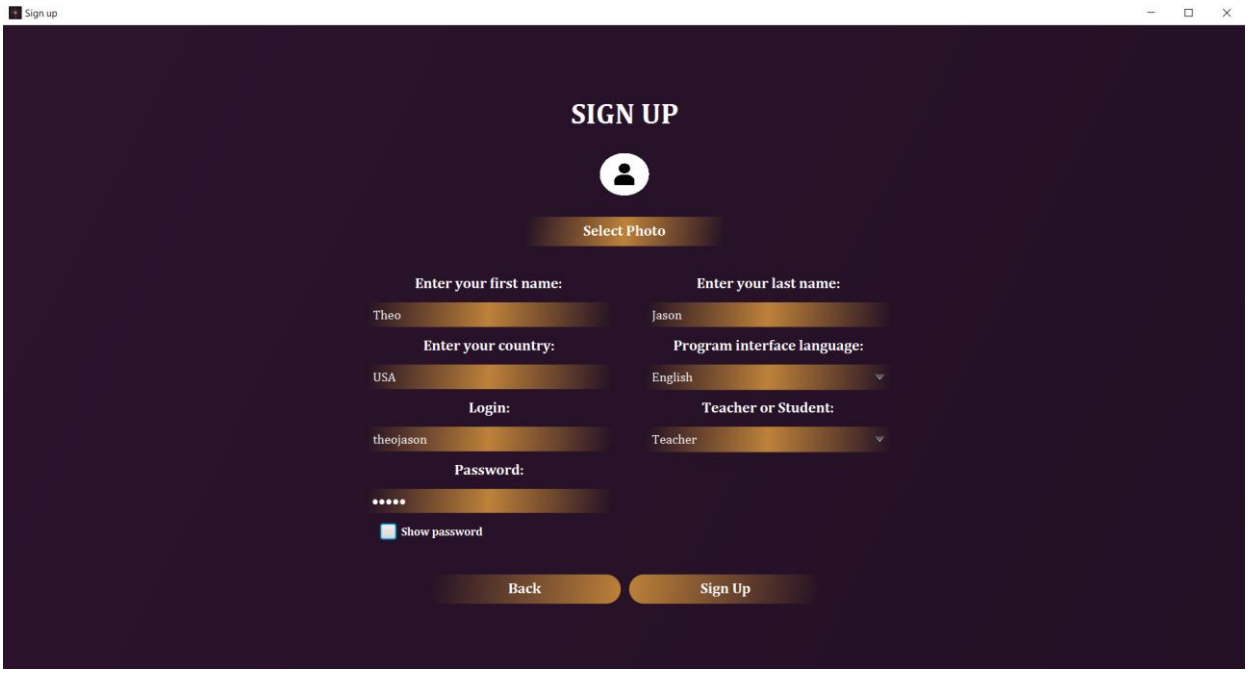

Рисунок 3.5 – Вікно реєстрації у застосунку

Тепер, після здійсненої реєстрації користувача відкриваємо вікно авторизації у застосунку (див. рис.3.6).

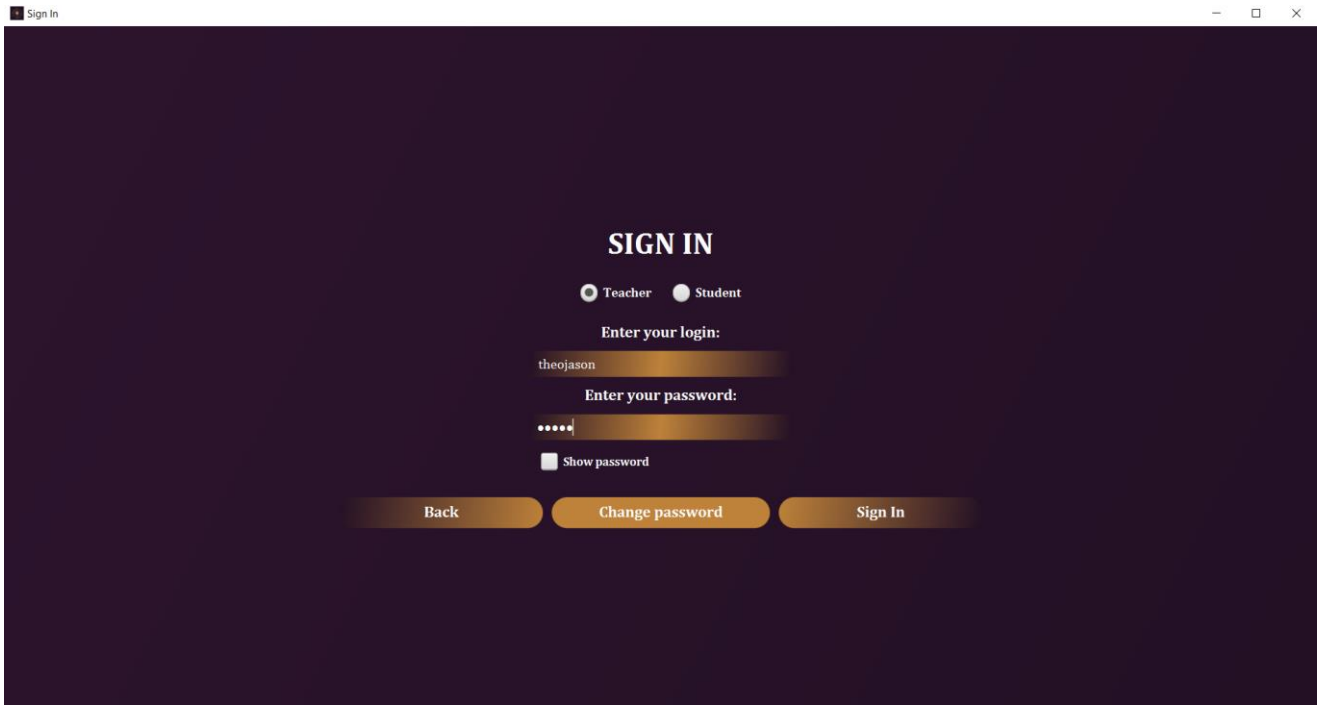

Рисунок 3.6 – Вікно реєстрації у застосунку

Тепер, після здійснення успішної авторизації змінюємо мову інтерфейсу програми (див. рис.3.7).

Рисунок 3.7 – Вікно зміни мови інтерфейсу застосунку

Далі створимо новий курс, ввівши всі необхідні дані про нього у відведенні поля вікна (див. рис.3.8).

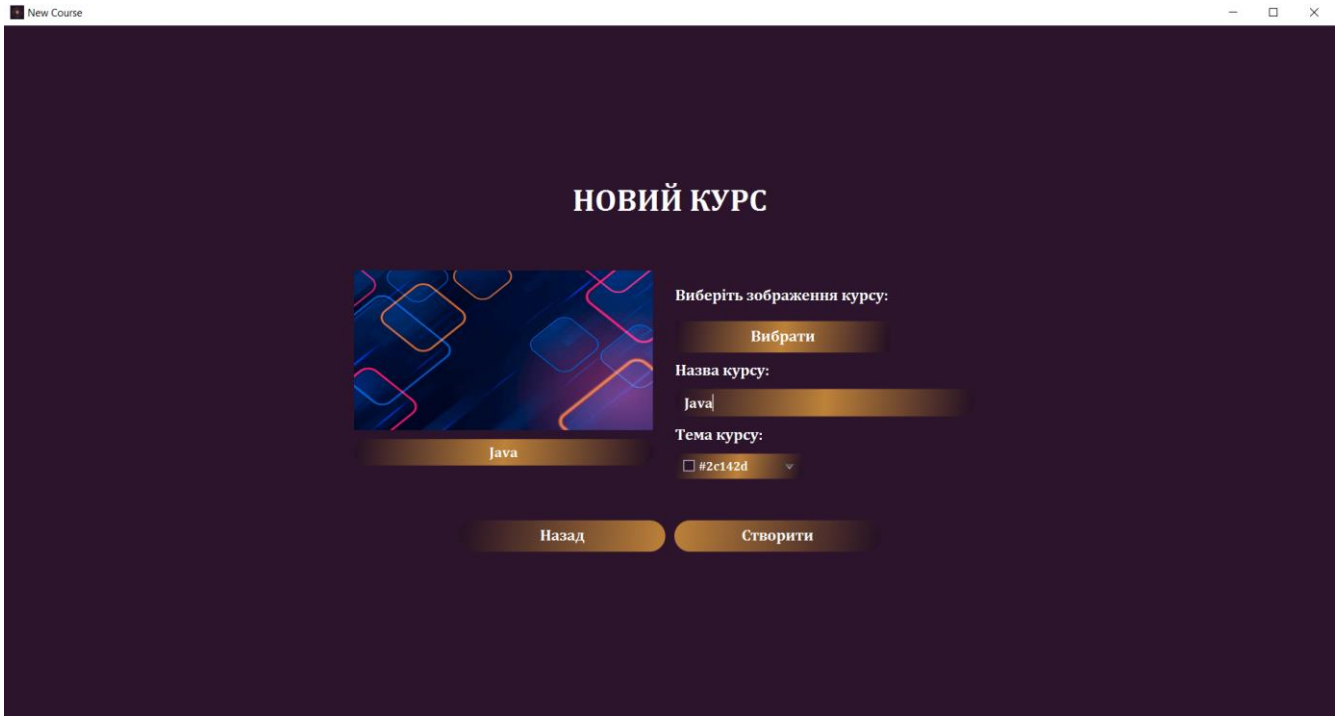

Рисунок 3.8 – Вікно зміни мови інтерфейсу застосунку

Після створення, у панелі всіх курсів відображається наш курс (див. рис.3.9).

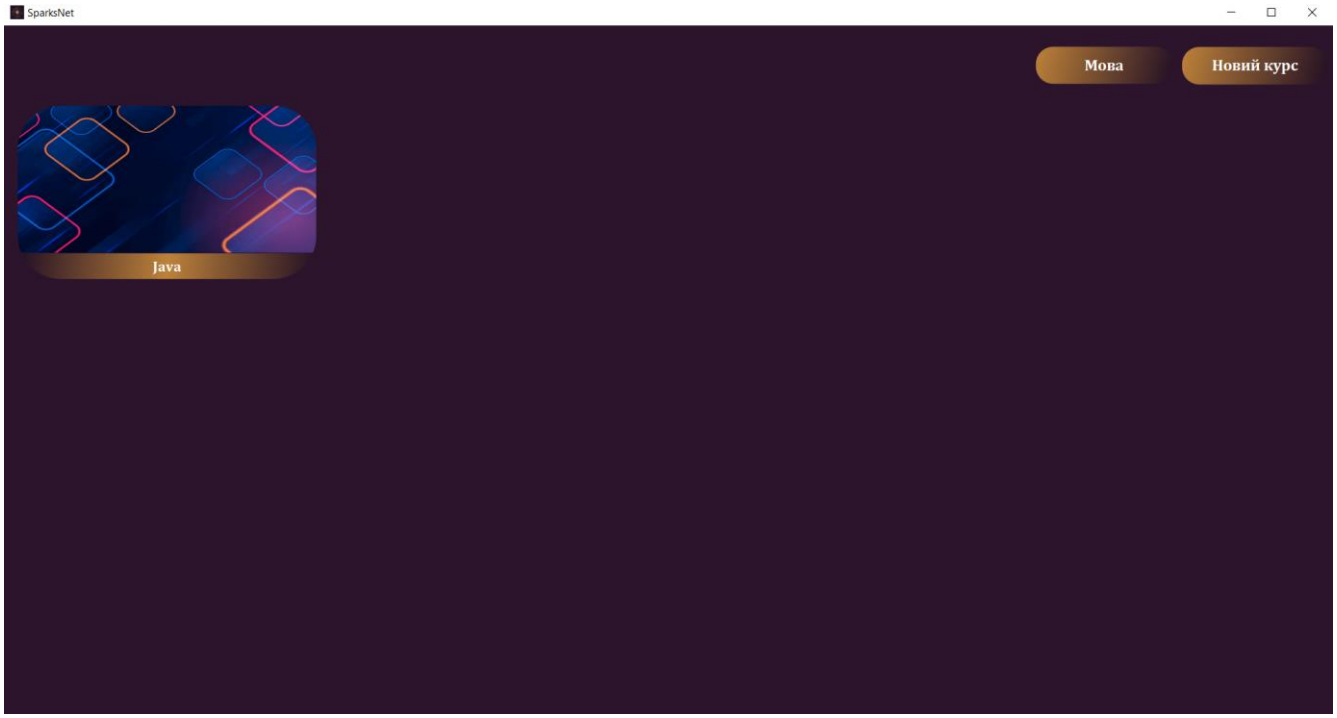

Рисунок 3.9 – Панель відображення всіх курсів користувача у застосунку

Натискаючи на відображений курс у панелі програми, відкривається вікно з відображенням всього вмісту курсу, і тепер там можна здійснювати різні функції у створеному курсі (див. рис.3.10).

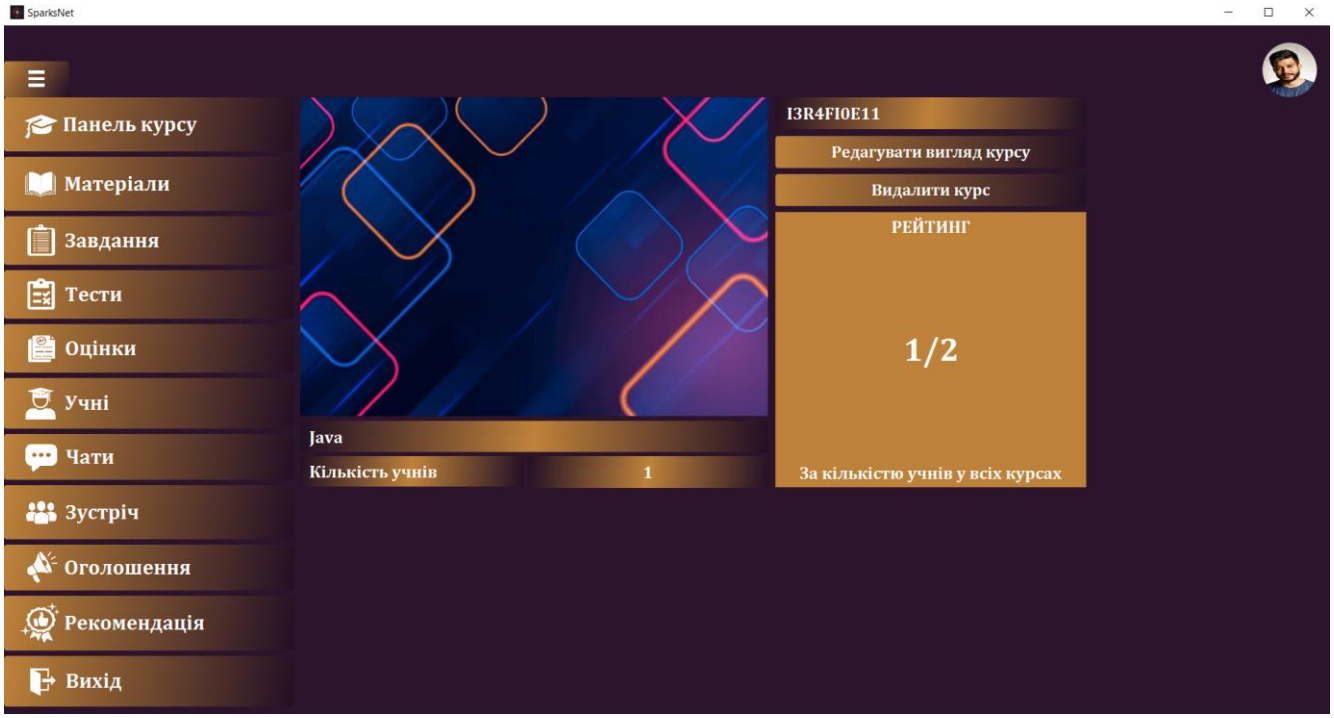

Рисунок 3.10 – Вікно вмісту вибраного курсу

Створимо матеріал заповнивши всі необхідні поля інформацією про матеріал та додамо його (див. рис.3.11).

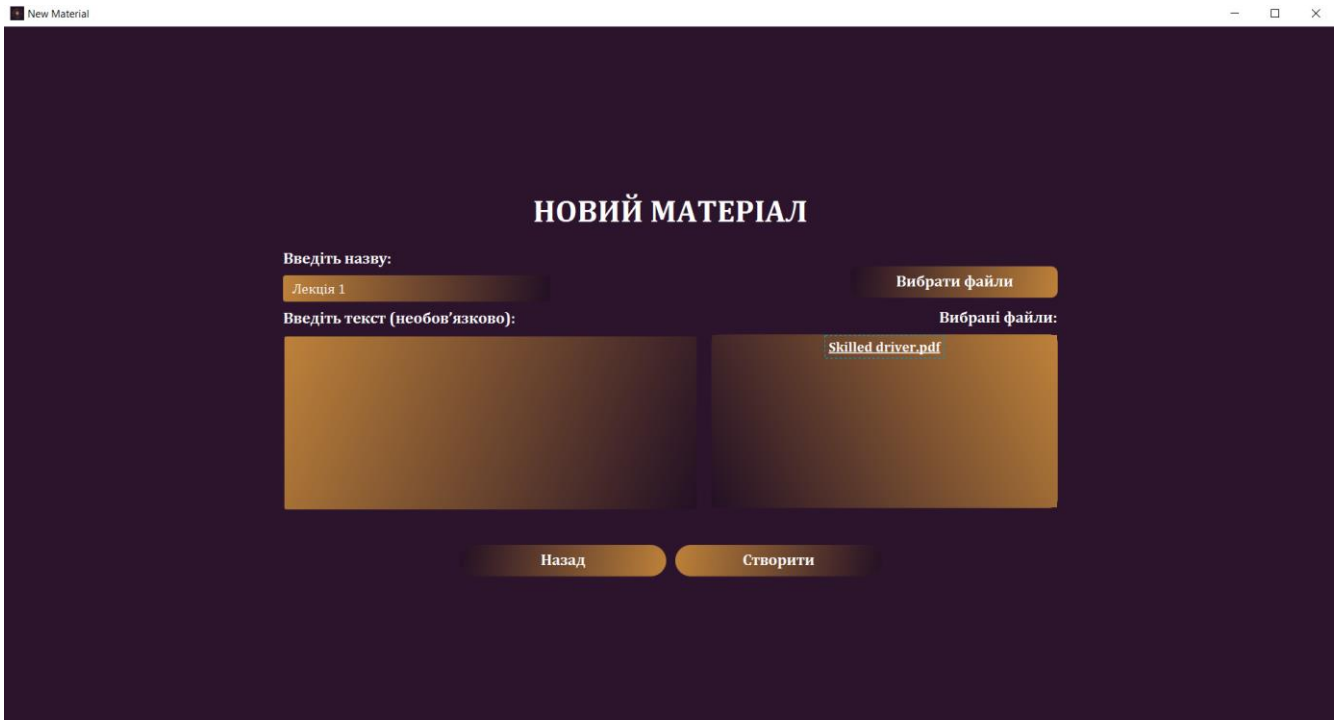

Рисунок 3.11 – Вікно створення матеріалу

Тепер у панелі відображення всіх матеріалів можна побачити створений матеріал (див. рис.3.12).

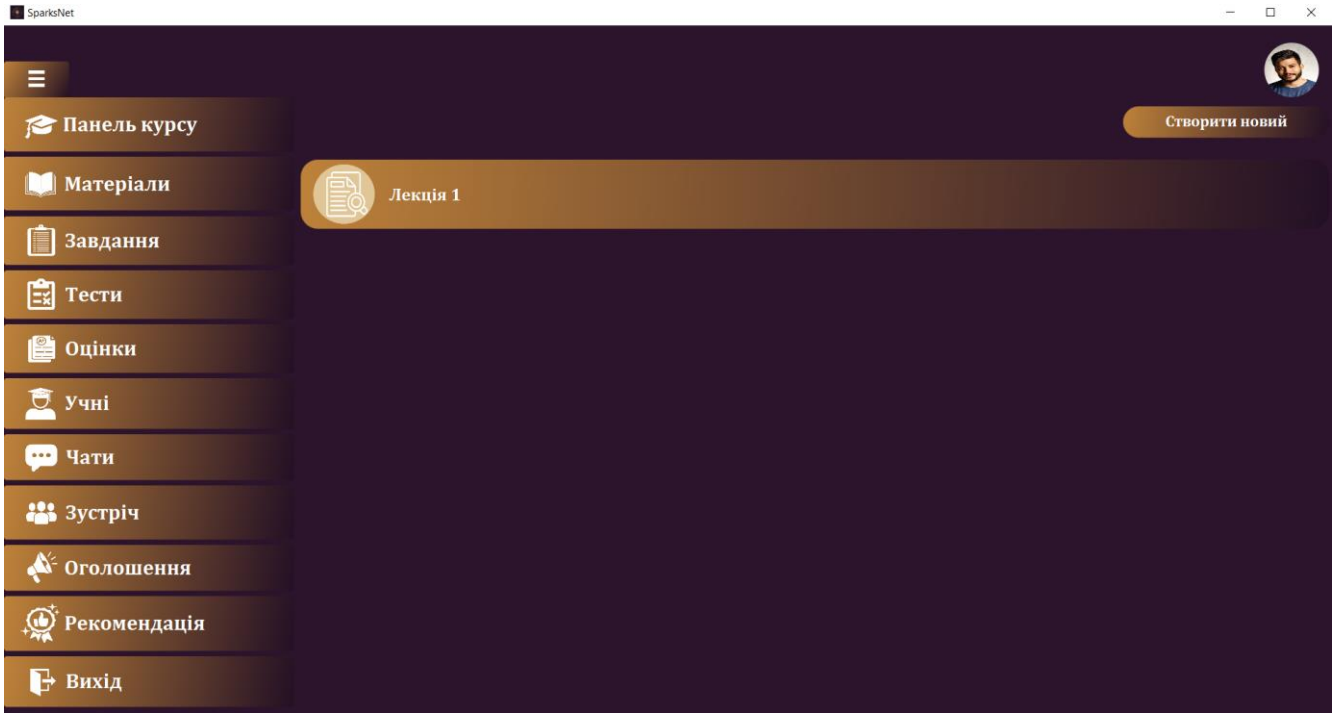

Рисунок 3.12 – Відображення всіх матеріалів курсу

Щоб переглянути вміст матеріалу, натискаємо на нього і тоді в нас відобразиться цей матеріал у панелі (див. рис.3.13).

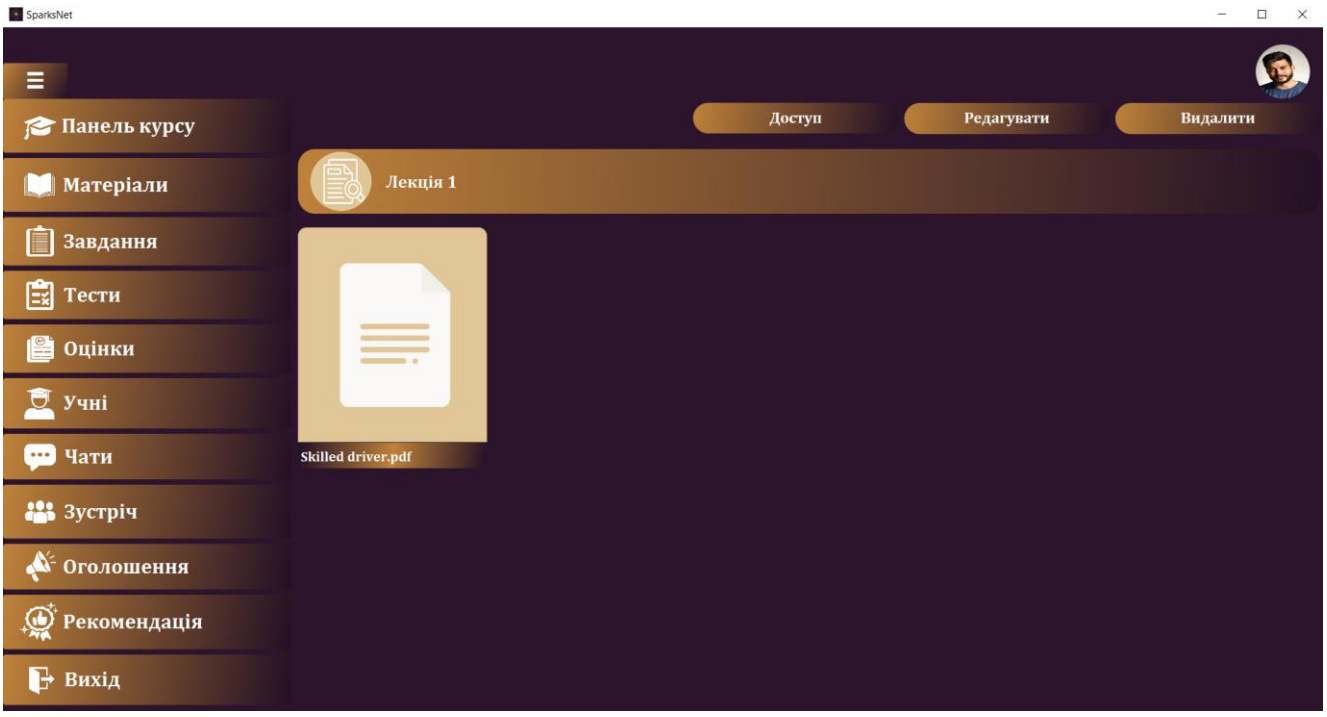

Рисунок 3.13 – Відображення вмісту матеріалу

Виконаємо редагування матеріалу (див. рис.3.14). Після редагування можна побачити, що вміст матеріалу змінився (див. рис.3.15).

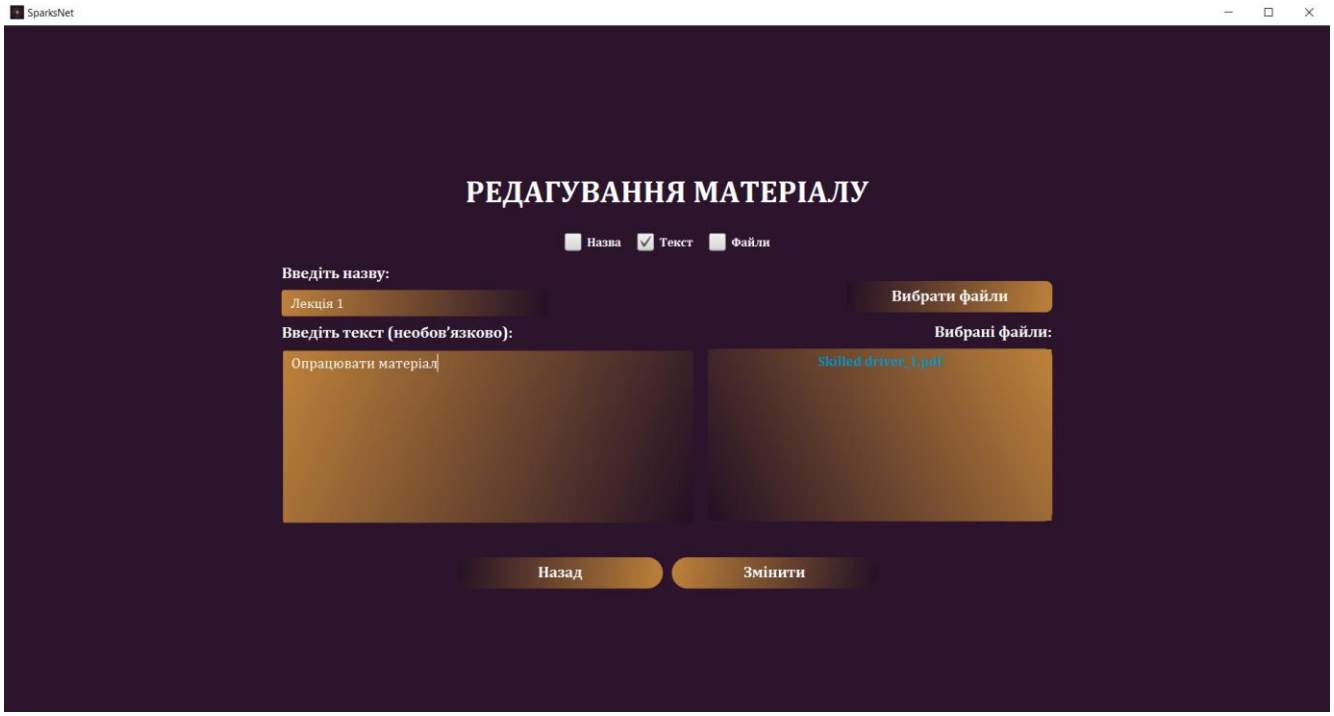

Рисунок 3.14 – Редагування матеріалу

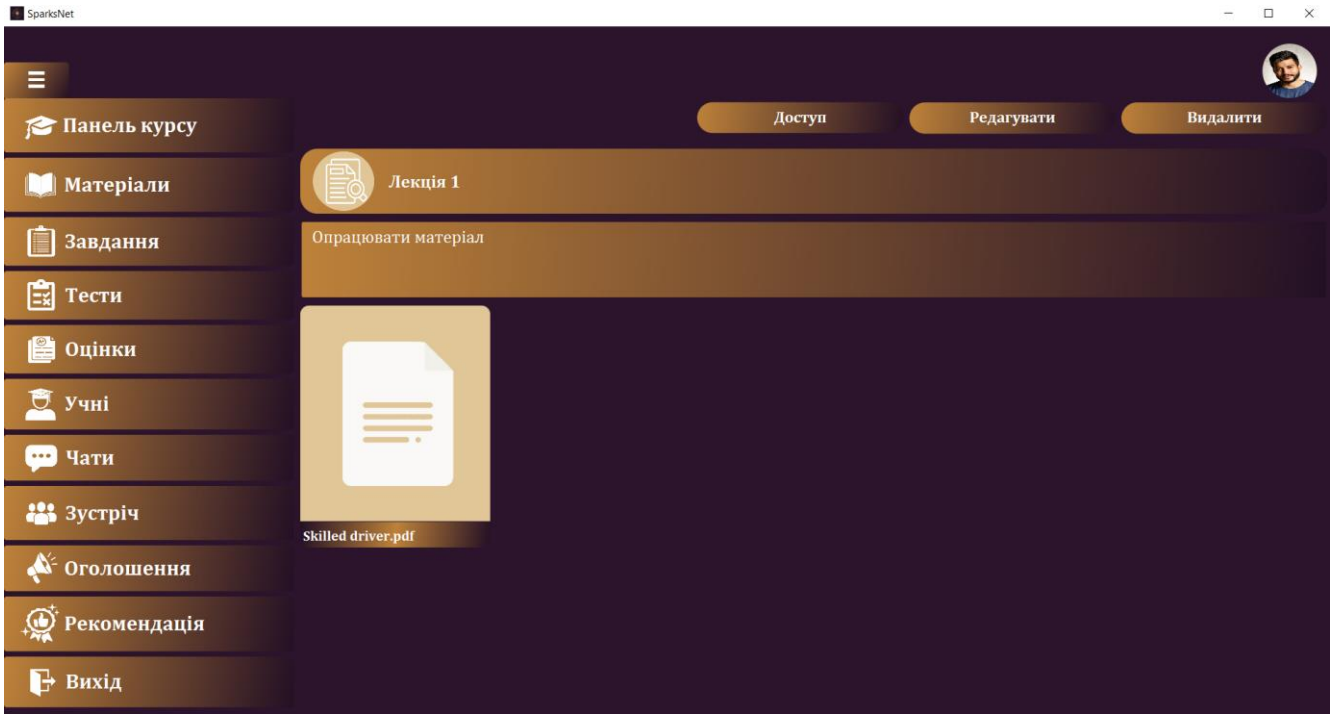

Рисунок 3.15 – Результат редагування матеріалу

Тепер створимо завдання, для цього впишемо всі необхідні дані у поля вікна створення завдання (див. рис.3.16). Після цього, натискаємо на назві завдання та побачимо вміст створеного завдання (див. рис.3.17).

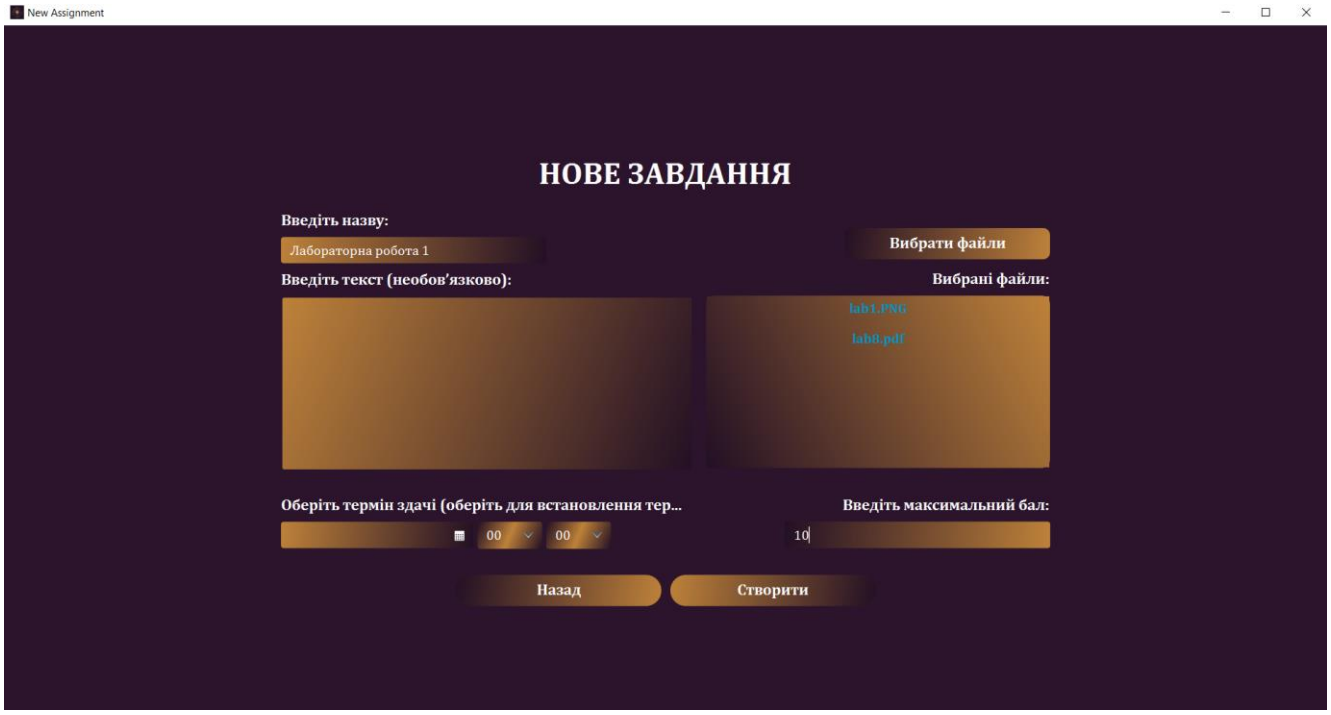

Рисунок 3.16 – Створення завдання курсу

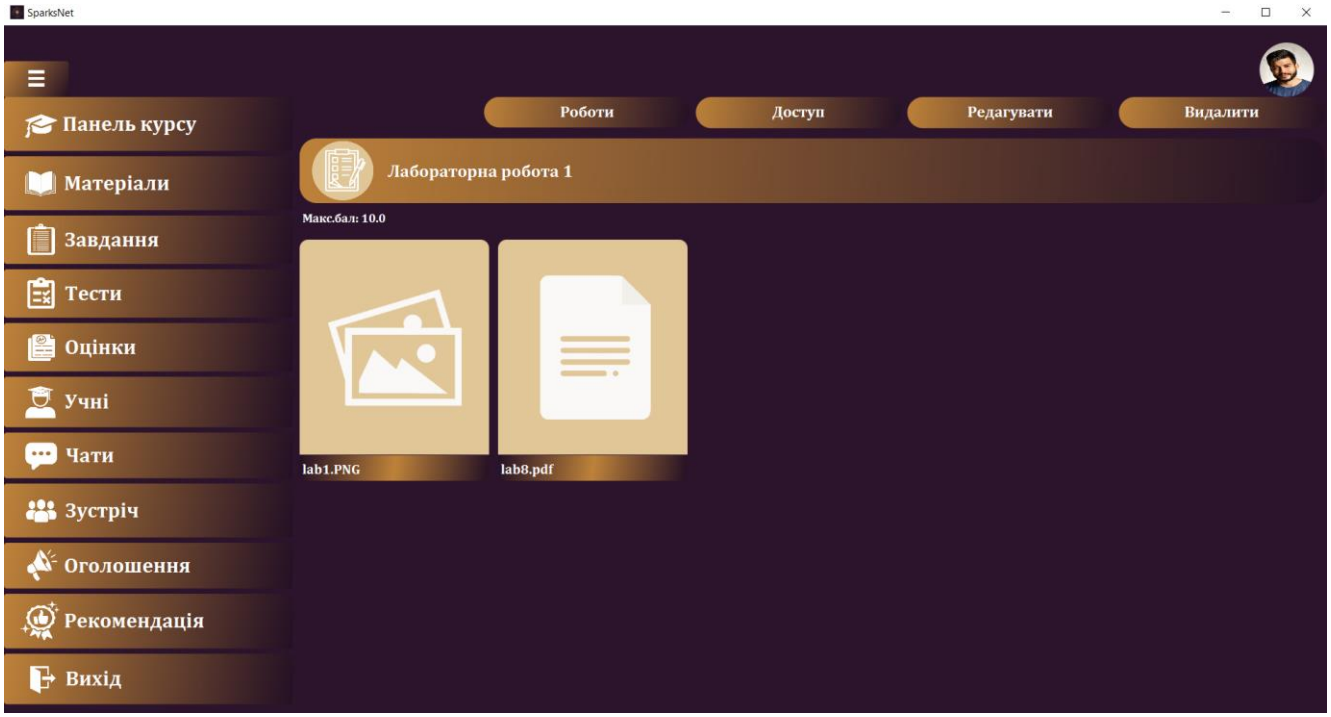

Рисунок 3.17 – Вміст створеного завдання

Створимо тест для цього впишемо інформацію про нього (див. рис.3.18) та сформуємо питання тестування (див. рис.3.19).

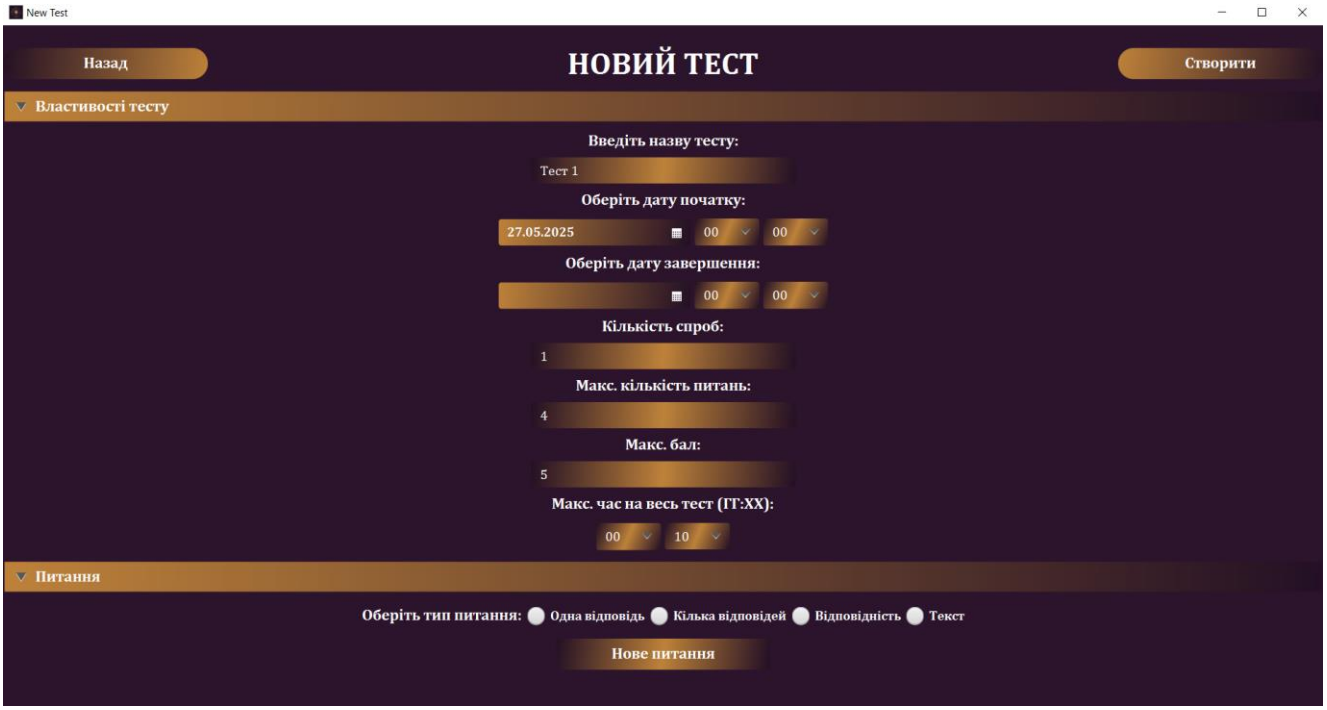

Рисунок 3.18 – Надана інформація про тест

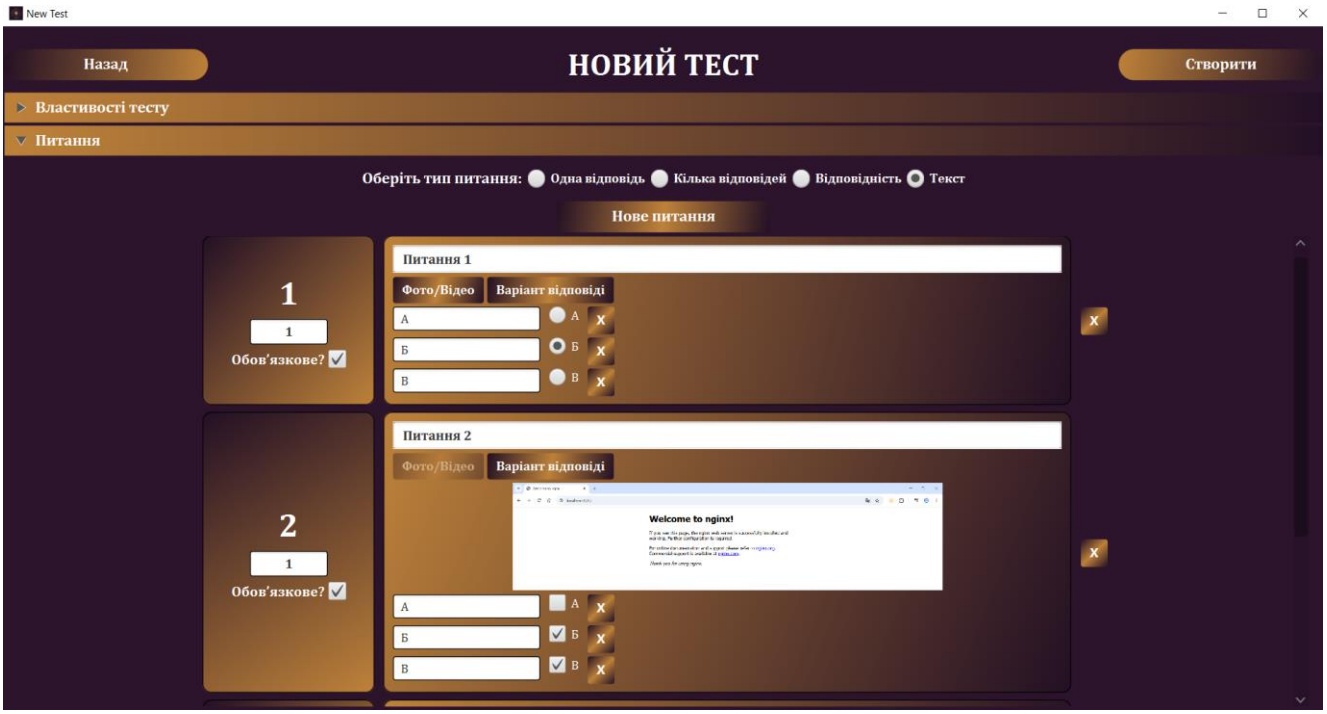

Рисунок 3.19 – Сформовані питання тесту

Результат створеного тесту можна тепер побачити після натискання на назві тесту (див. рис.3.20).

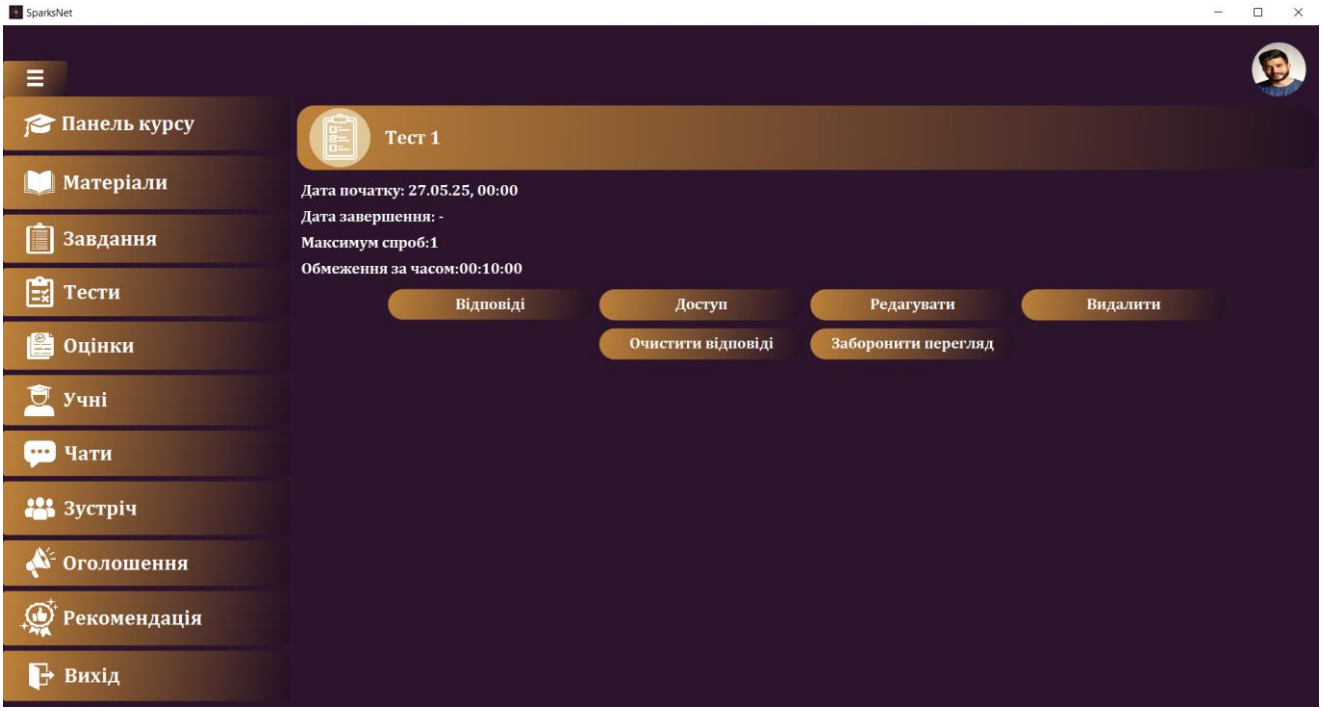

Рисунок 3.20 – Вміст створеного тесту

Тепер, зареєструвавши обліковий запис студента, додамо йому створений курс (див. рис.3.21).

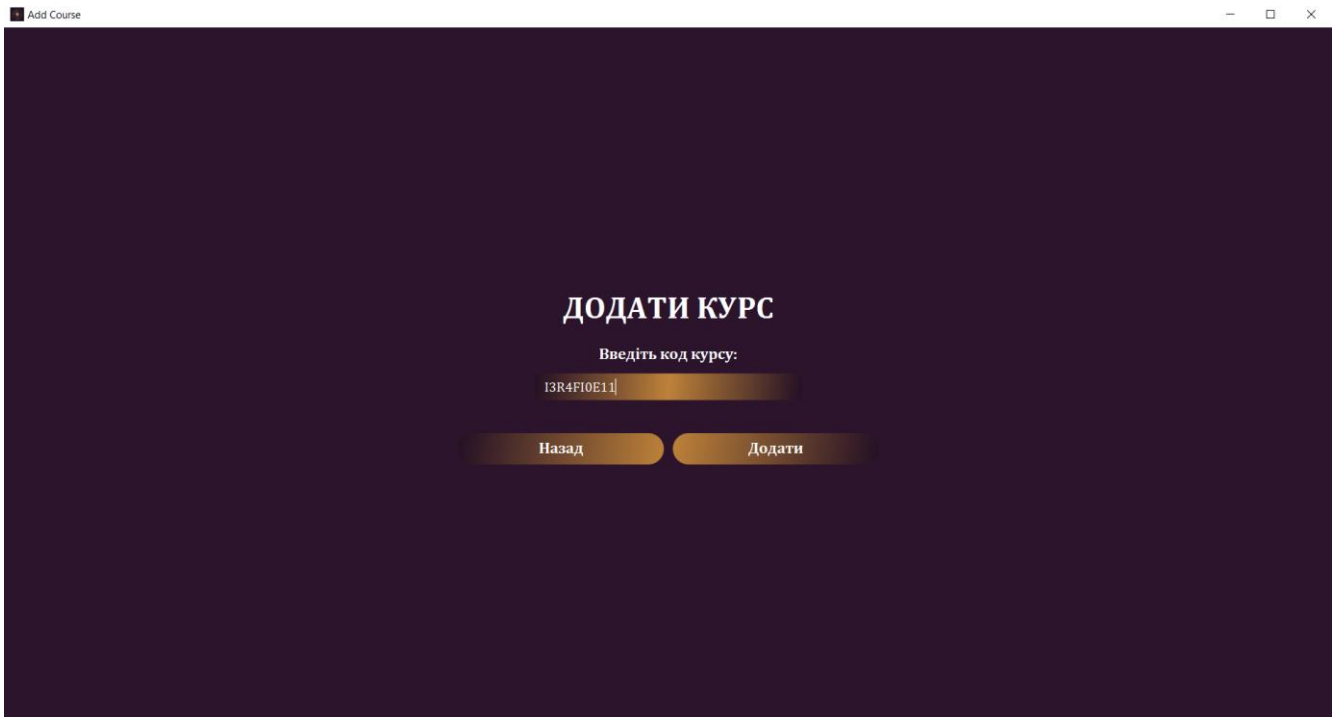

Рисунок 3.21 – Додавання курсу студентом

Після введення коду курсу, студент буде зареєстрований у курсі та йому буде відображати цей курс у його панелі курсів (див. рис.3.22).

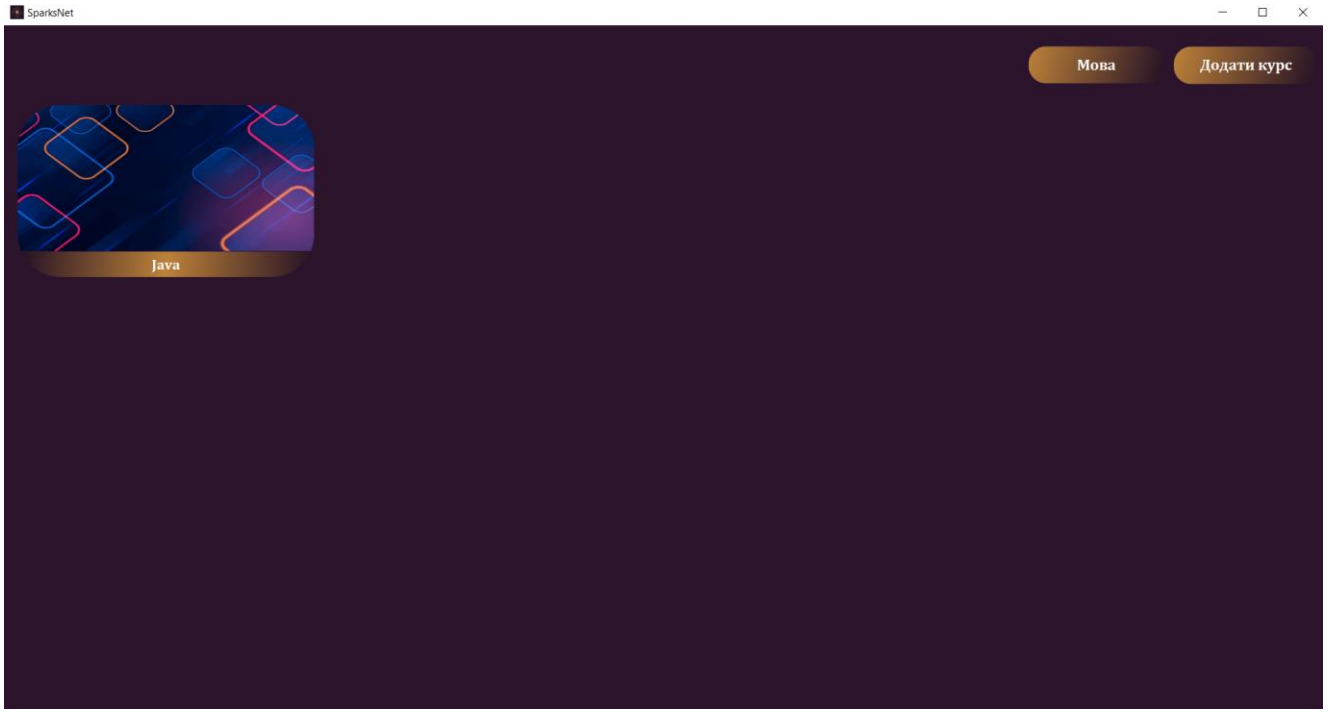

Рисунок 3.22 – Відображення всіх курсів де зареєстрований студент

При натисканні на цьому курсі, відобразиться панель вмісту курсу, де студент може здійснювати певні дії у курсі (див. рис.3.23).

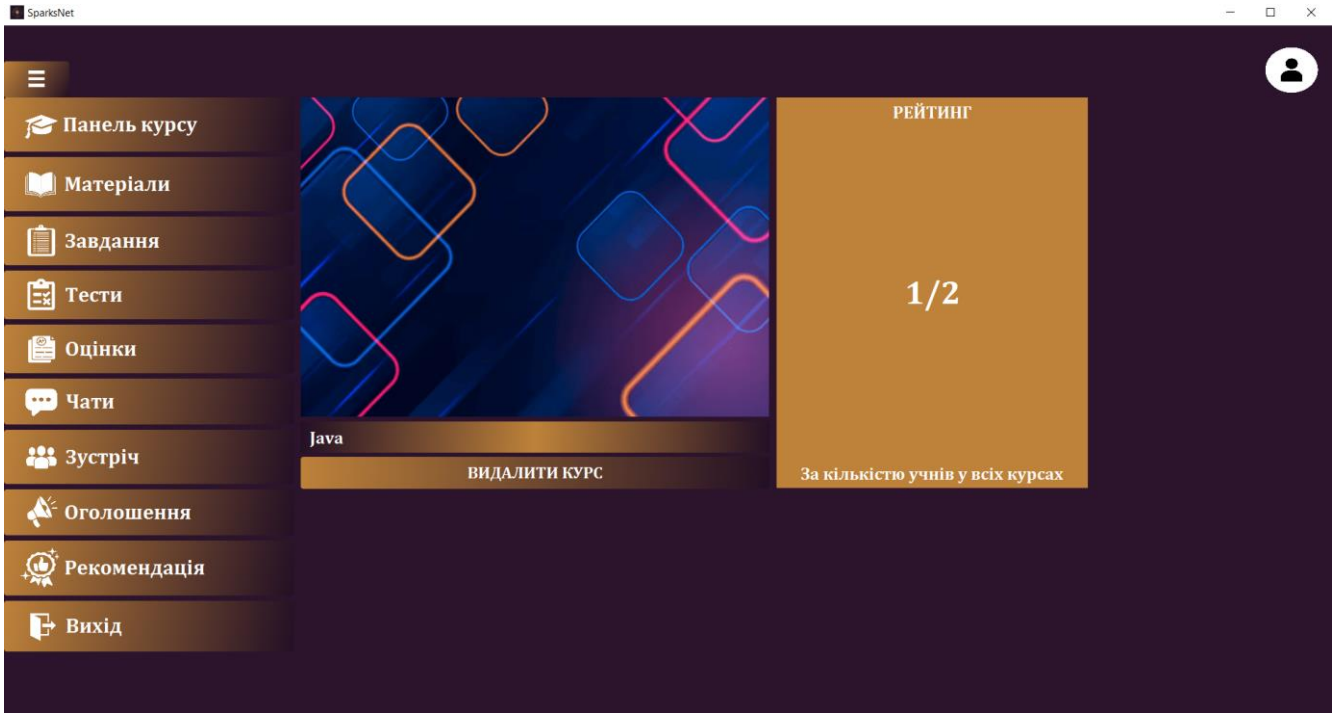

Рисунок 3.23 – Панель вмісту курсу для студента

Тепер студент може видаляти себе з курсу, переглядати матеріали, здавати завдання, проходити тести та робити інші можливі дії у застосунку.

Здамо роботу у завдання курсу (див. рис.3.24), щоб здати його прикріпляємо посилання на папку з файлами, які демонструють виконану роботу.

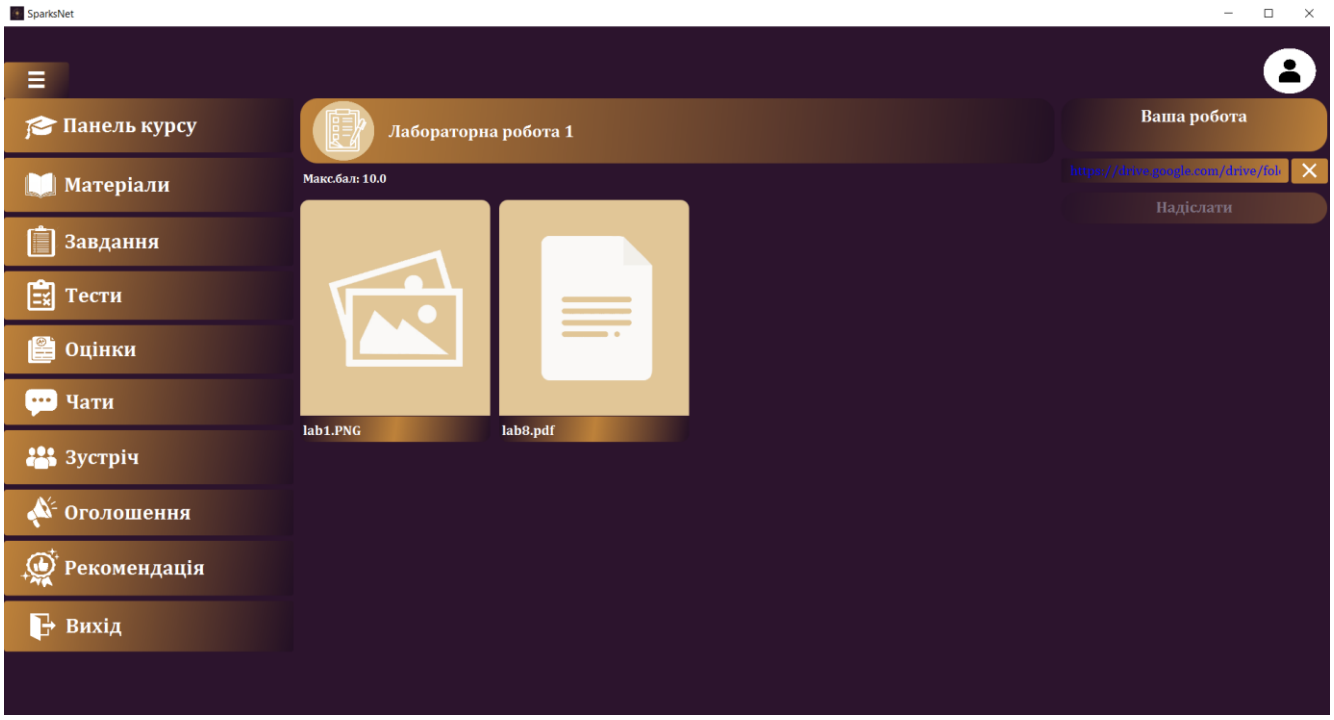

Рисунок 3.24 – Здача виконаного завдання студентом

Пройдемо тестування студентом, для цього обираємо опцію тести та обраємо тест, який є доступний, потім ми бачимо опис тесту та кнопку для початку тестування (див. рис.3.25).

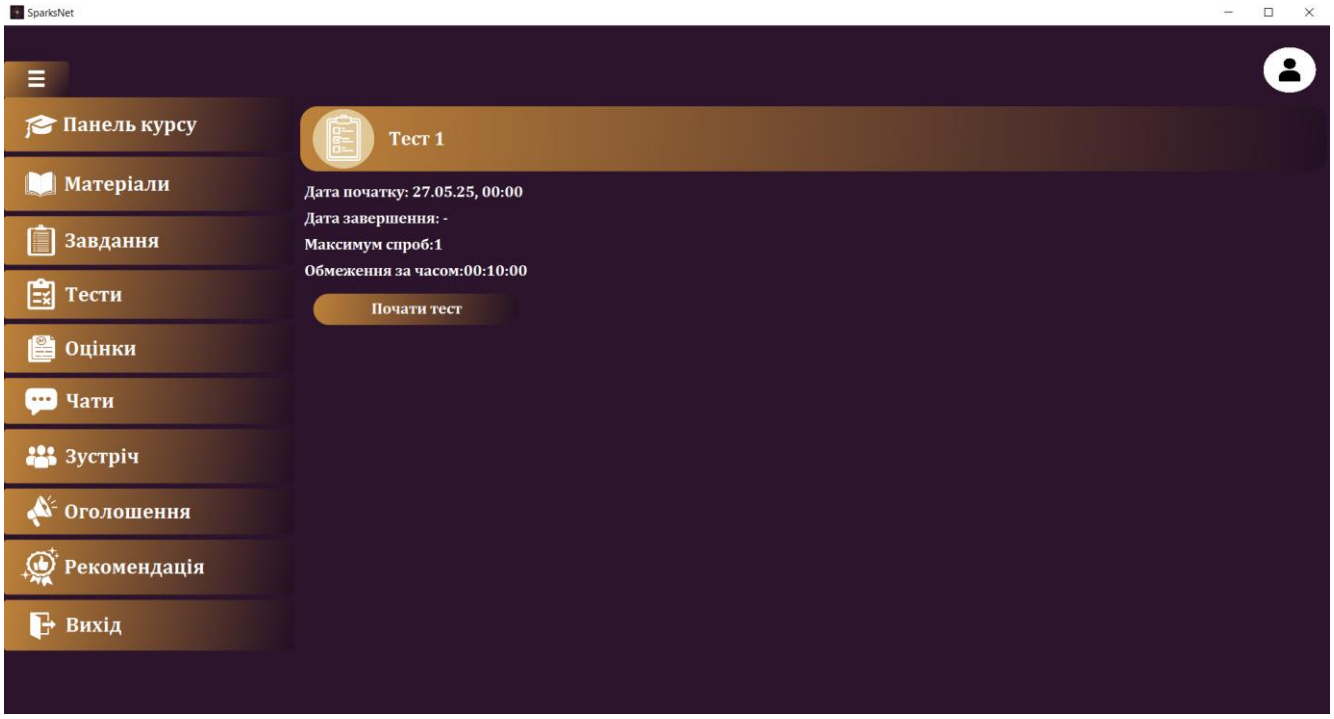

Рисунок 3.25 – Вигляд вмісту тесту у студентській версії програми

При натисканні на кнопку початку тестування, відкриється вікно тестування, якщо у тесті є обмеження в часі, то почнеться зворотній відлік (див. рис.3.26), і студент може завершити тест при натисканні на кнопку завершення тесту, після чого всі його відповіді будуть перевірені на правильність.

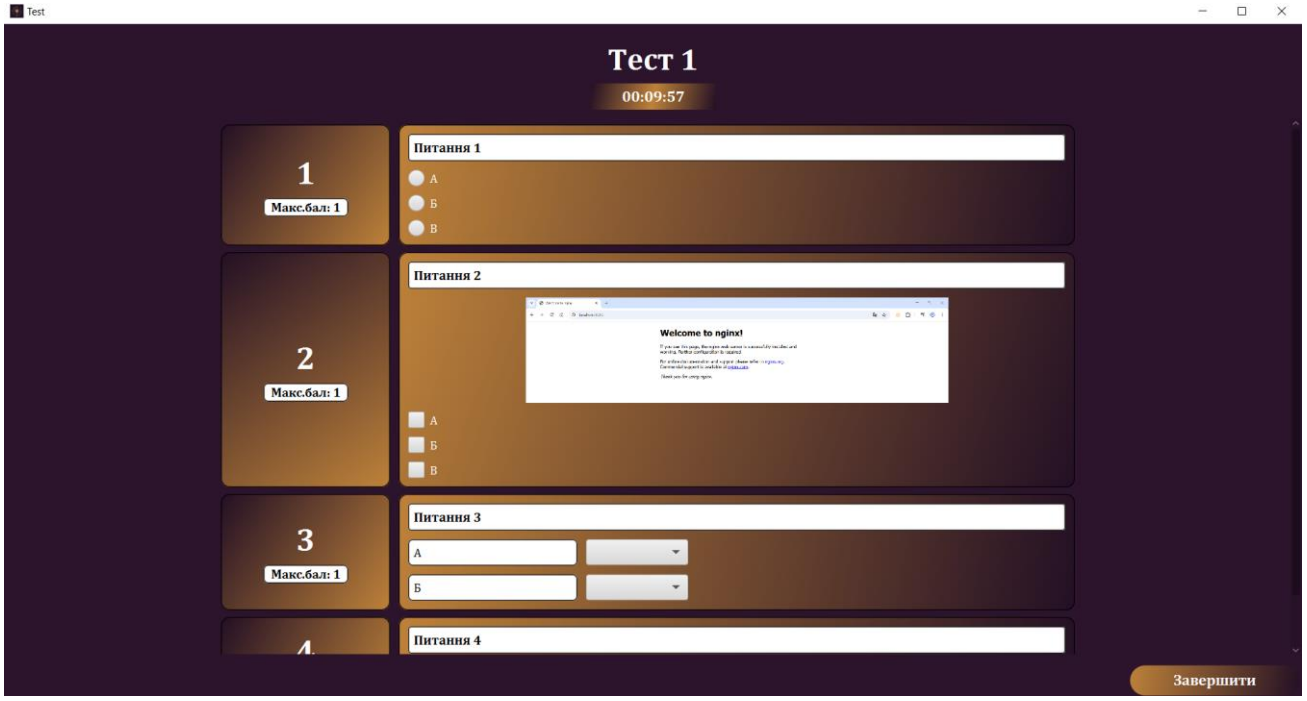

Рисунок 3.26 – Процес проходження тестування

Якщо викладач відрив дозвіл на перегляд результатів тестування, то студент після проходження тесту може побачити свій результат (див. рис.3.27).

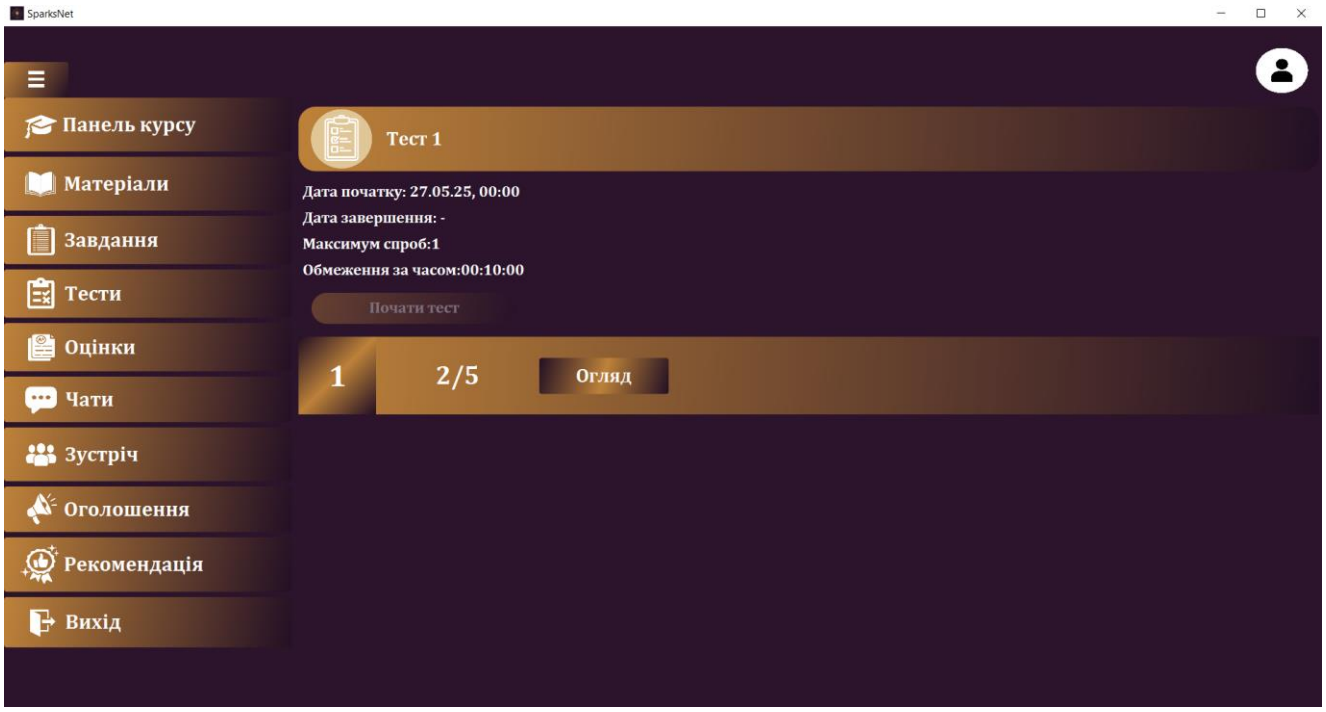

Рисунок 3.27 – Відображення результату тестування

Якщо тест містив питання з текстовою відповідю студента, то це буде оцінка без цього питання, бо викладач має його перевірити власноруч та поставити кількість балів за це питання. Натискаючи на кнопку огляду, можна побачити результати по кожному питанню (див. рис.3.28).

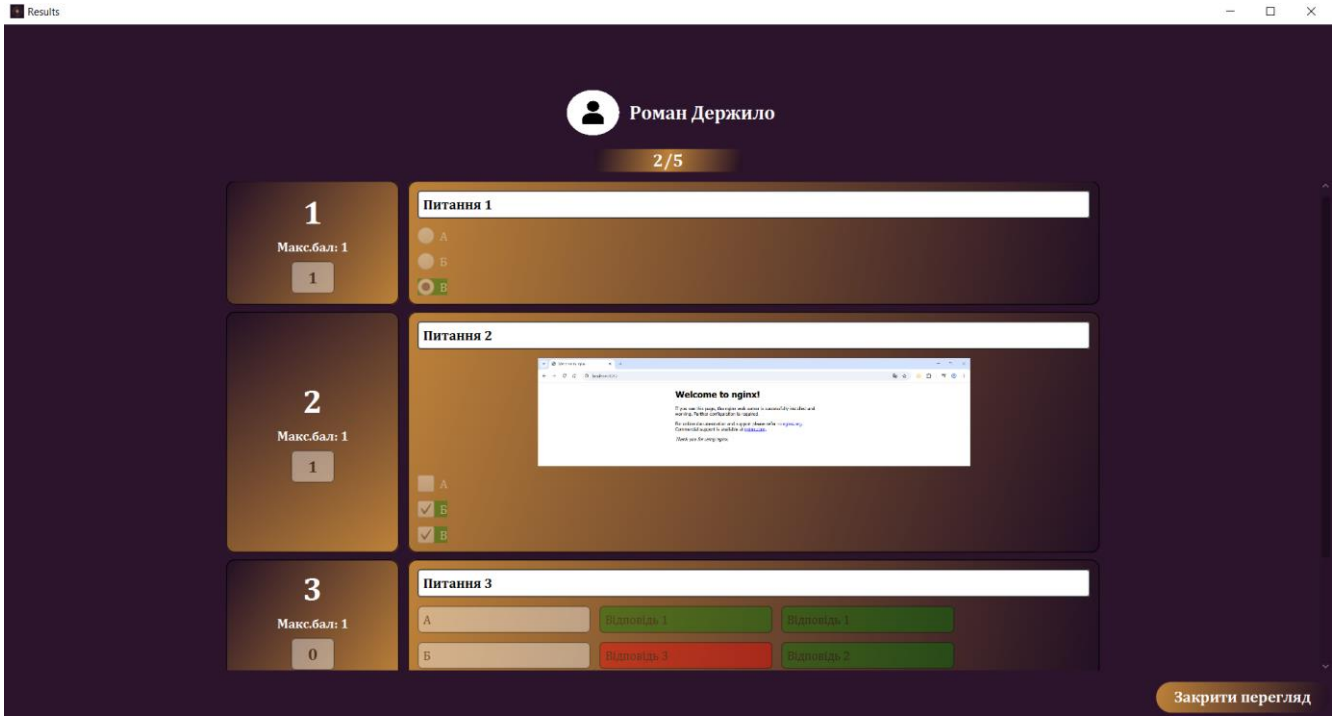

Рисунок 3.28 – Детальний огляд результатів тесту

Викладач ще може оцінювати роботи, які здають студенти у завдання курсу, для цього йому потрібно натиснути на кнопку «Роботи» у панелі відображення вмісту завдання, тоді йому відобразяться всі роботи які здавали студенти, у спеціально відведеному полі компонента зданого завдання, викладач може поставити бал за завдання (див. рис.3.29).

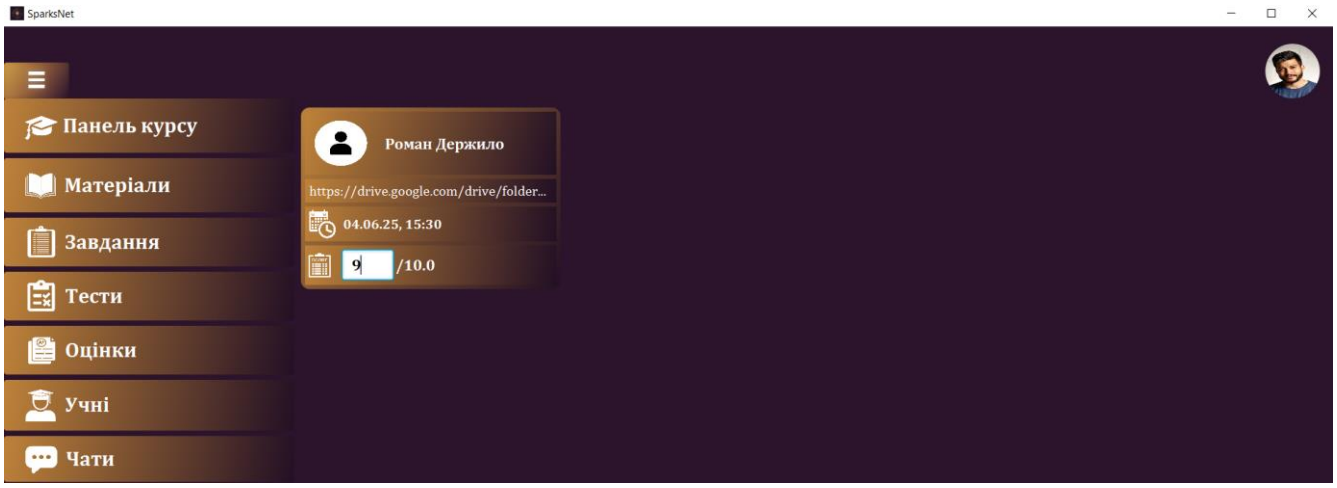

Рисунок 3.29 – Панель відображення всіх зданих робіт завдання

Так само, викладач може переглянути відповіді на тестування, натискаючи на кнопку відповіді у панелі вмісту тестування. Тут викладач може переглянути, які відповіді надав студент, і оцінити питання з розгорнутою відповідю, які він і сам має перевірити. У нижче наведеному зображені відображений результат студента вже після того, як викладач оцінив питання з розгорнуто відповіді (див. рис.3.30).

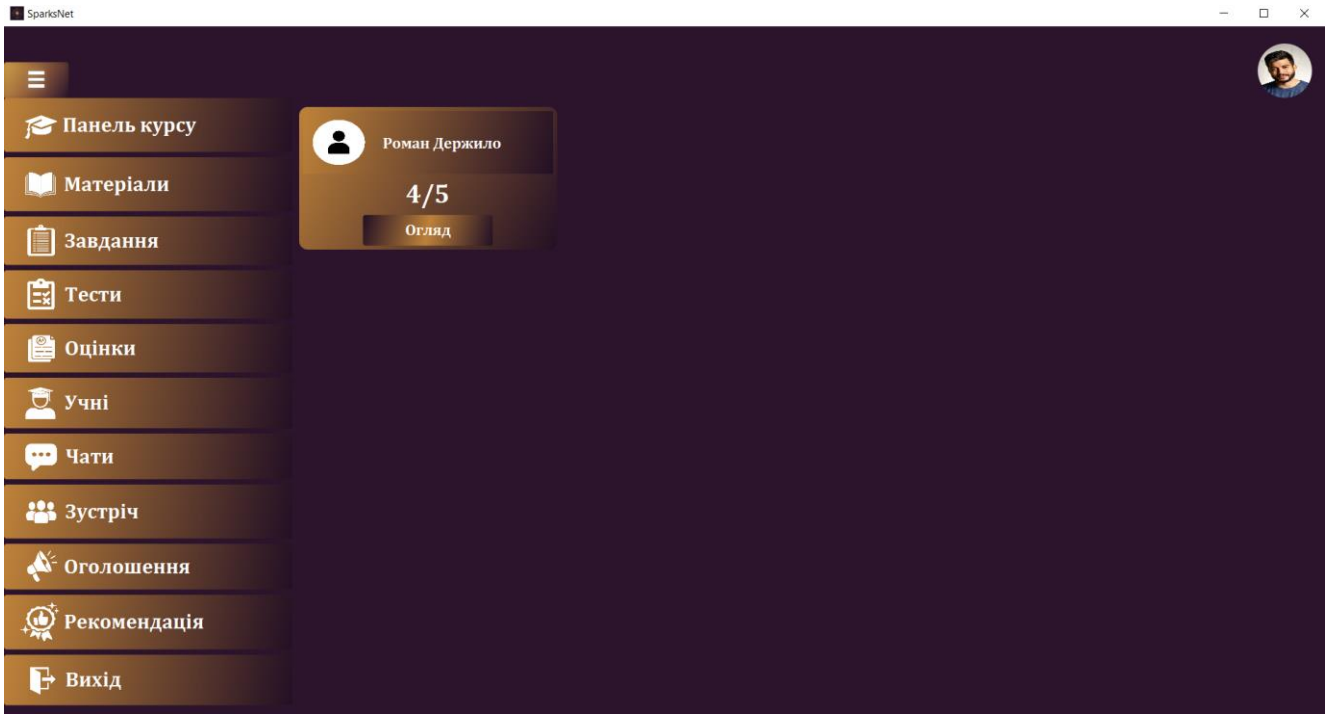

Рисунок 3.30 – Панель відображення результатів тесту

Ще можна переглянути успішність всіх студентів у курсі (див. рис.3.31).

Рисунок 3.31 – Панель відображення успішності студентів

Викладач може побачити всіх студентів зареєстрованих у курсі, для цього йому потрібно натиснути кнопку учнів, тоді у панелі відобразяться всі студенти, де викладач може видаляти їх, якщо потрібно (див. рис.3.32).

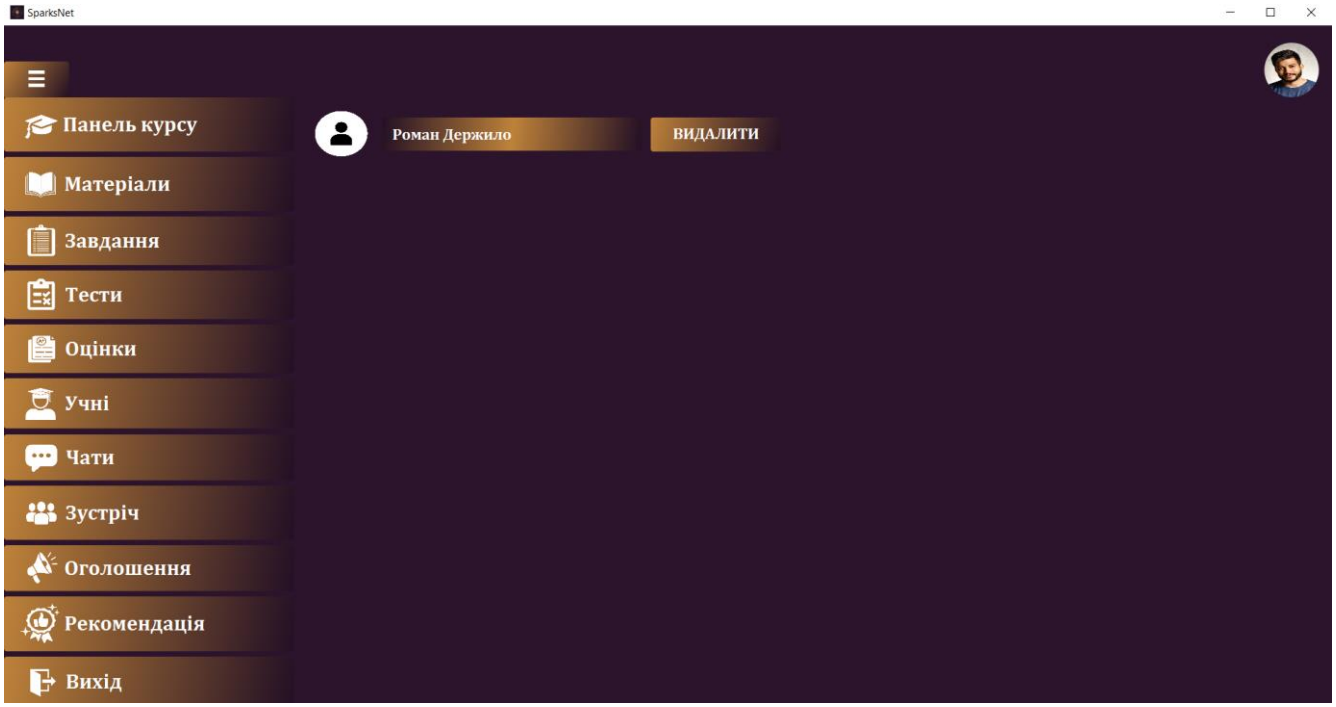

Рисунок 3.32 – Панель відображення всіх студентів

Також викладачі та студенти у цій програмі можуть спілкуватися за допомогою чатів (див. рис.3.33). Тут викладач має окремий чат з кожним студентом, а студент має лише один чат з викладачем курсу.

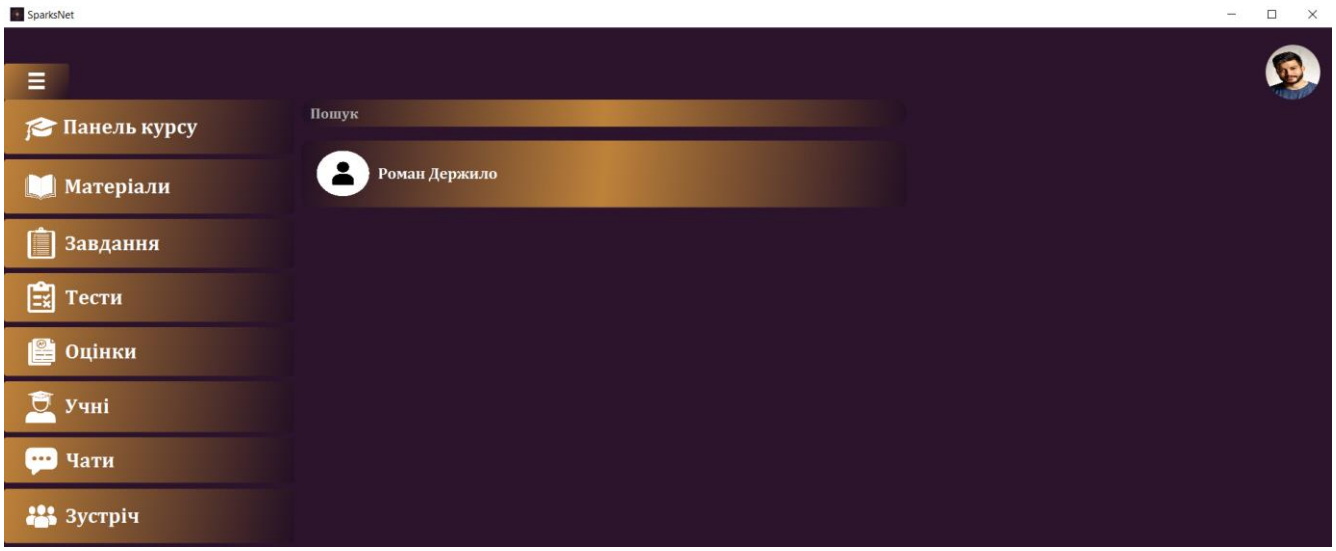

Рисунок 3.33 – Панель відображення чатів викладача

Щоб побачити повідомлення або написати нове потрібно натиснути на чаті, і тоді відобразиться панель у якій відобразиться усі повідомлення між користувачами та

можна буде надіслати нове (див. рис.3.34). Видалити вміст всієї переписки може тільки викладач натискаючи на кнопку очистити.

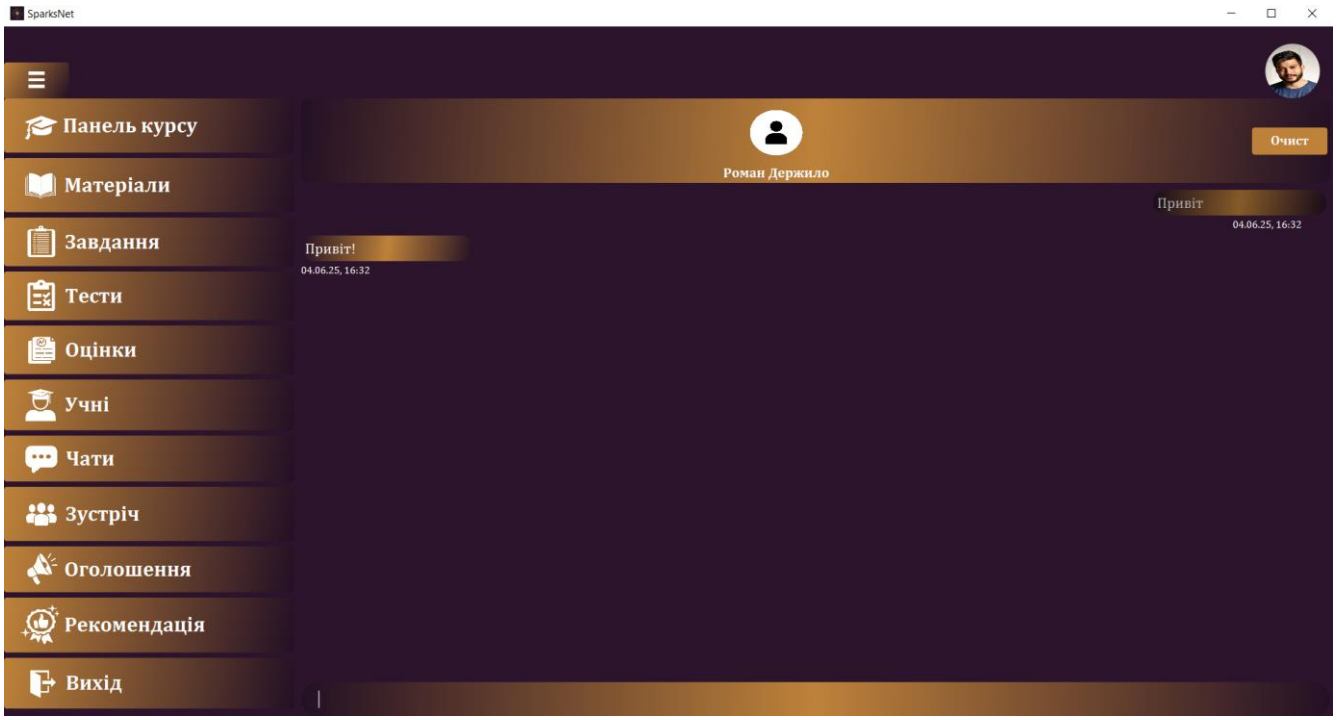

Рисунок 3.34 – Переписка між викладачем та студентом у курсі

Викладач може розпочати відео зустріч зі студентами курсу, для цього йому потрібно натиснути на кнопку зустріч та погодитися на створення зустрічі, після цього відкриється нова зустріч у Google Meet (див. рис.3.35).

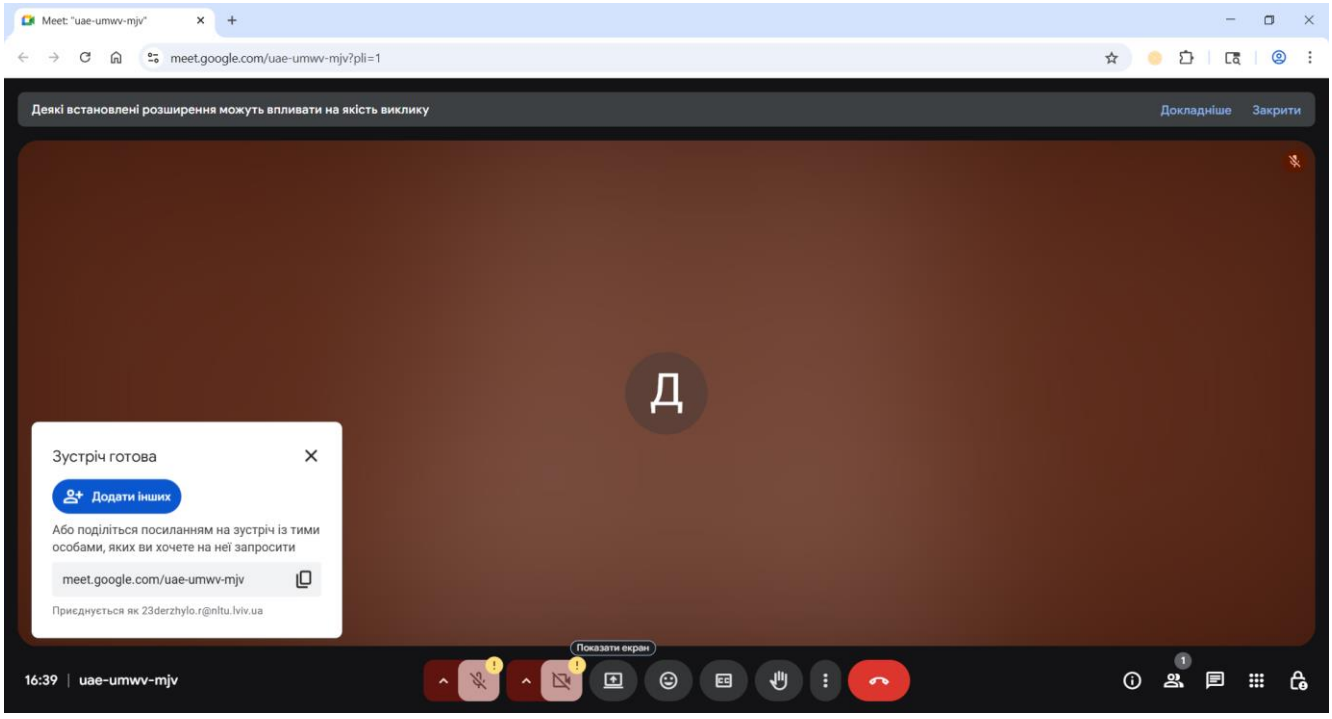

Рисунок 3.35 – Створена відео зустріч

Щоб студенти могли приєднуватися до зустріч, викладачу потрібно вставити посилання у вікно яке відривається разом із зустрічю. Він може потім деактивувати цей лінк (див. рис.3.36), і тоді студенти більше не зможуть приєднатися до зустрічі через кнопку зустріч у курсі.

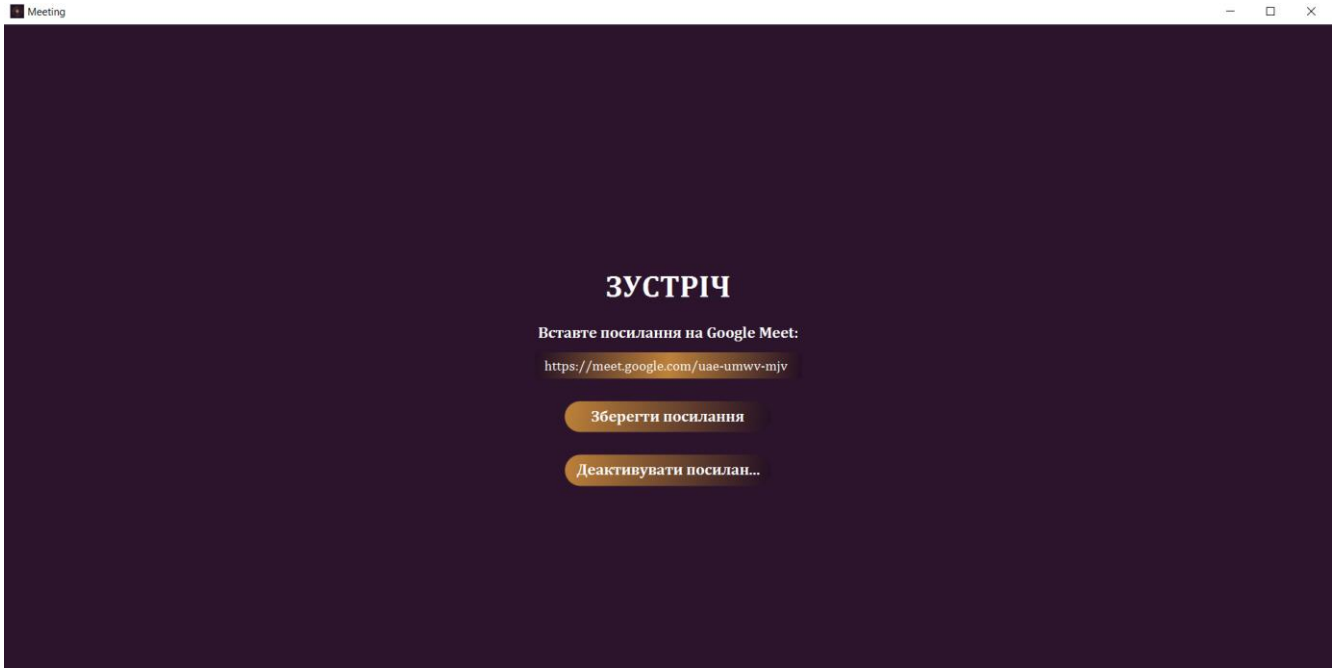

Рисунок 3.36 – Вікно дій з посиланням зустрічі

Ще можна створювати оголошення у курсі, для цього потрібно натиснути на кнопку оголошення, і тоді можна написати нове оголошення (див. рис.3.37).

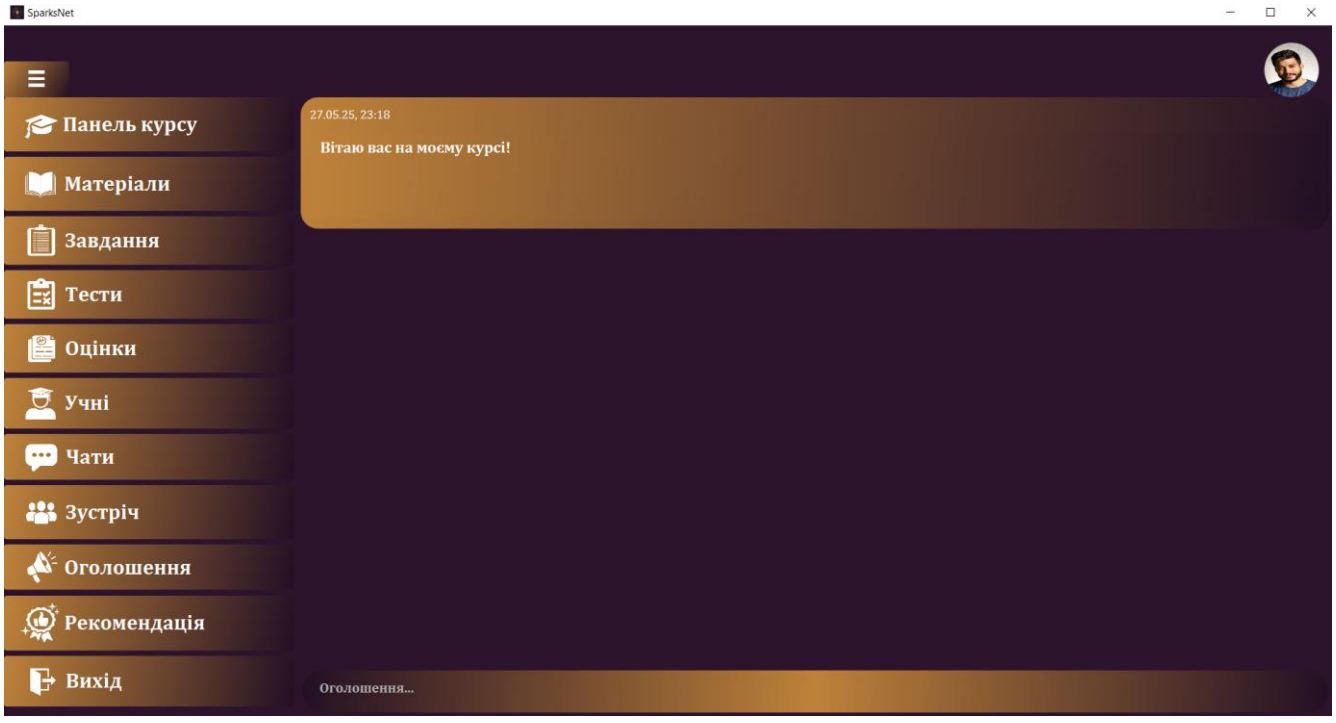

Рисунок 3.37 – Відображення оголошень курсу

Викладач може створити рекомендацію свого курсу, який буде показуватися у панелі рекомендацій абсолютно всім користувачам програми, для цього йому потірбно натиснути на кнопку створення у панелі рекомендацій, яка відображається після вибору кнопки рекомендація (див. рис.3.38).

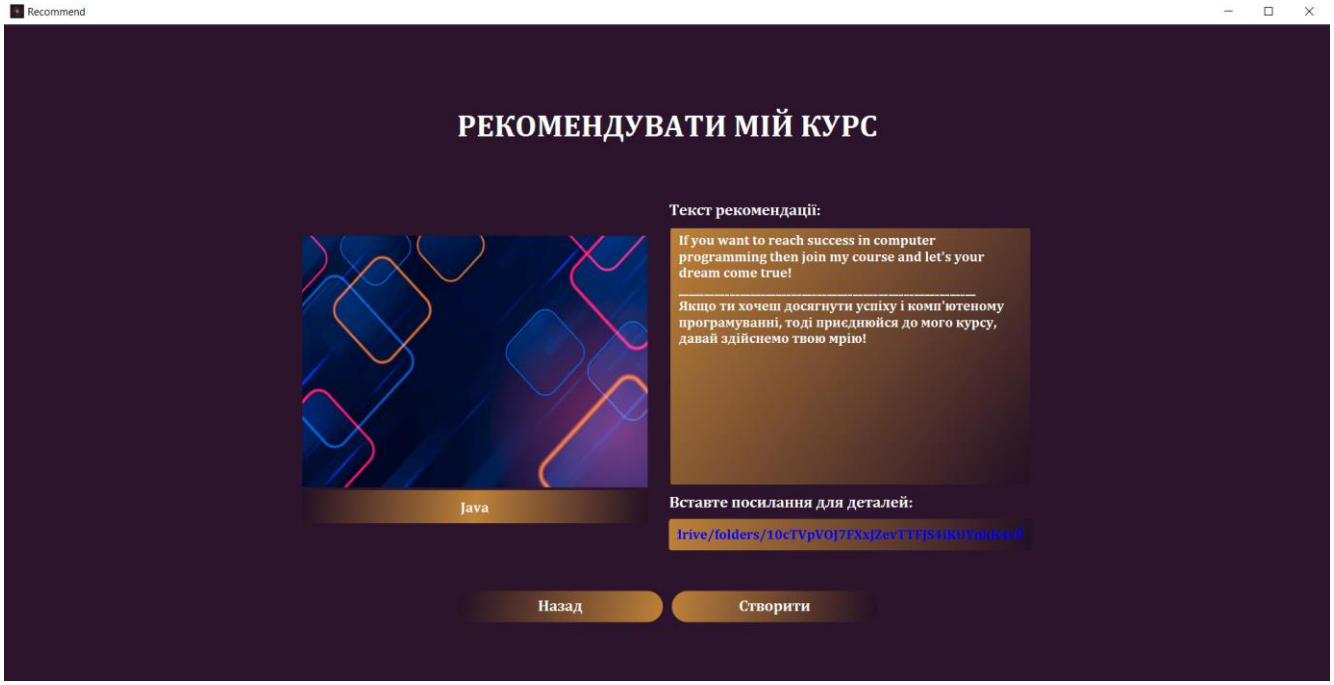

Рисунок 3.38 – Створення рекомендації курсу

Після створення ця рекомендація додається до усіх інших і відображається у панелі (див. рис.3.39).

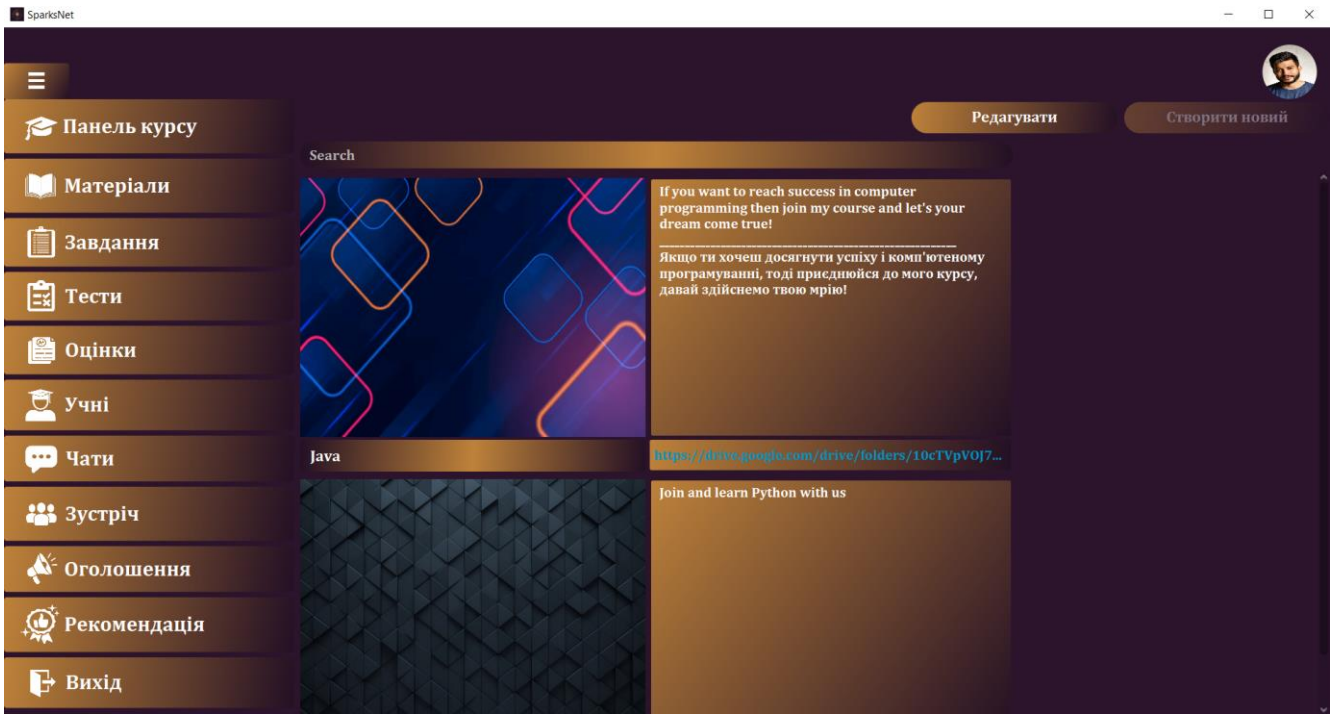

Рисунок 3.39 – Відображення всіх рекомендацій курсів

Свою рекомендацію можна відредагувати, натискаючи на кнопку «Редагувати», де може буде змінити текст, посилання або видалити рекомендацію (див. рис.3.40). Першими у панелі рекомендацій відображаються найновіші з них.

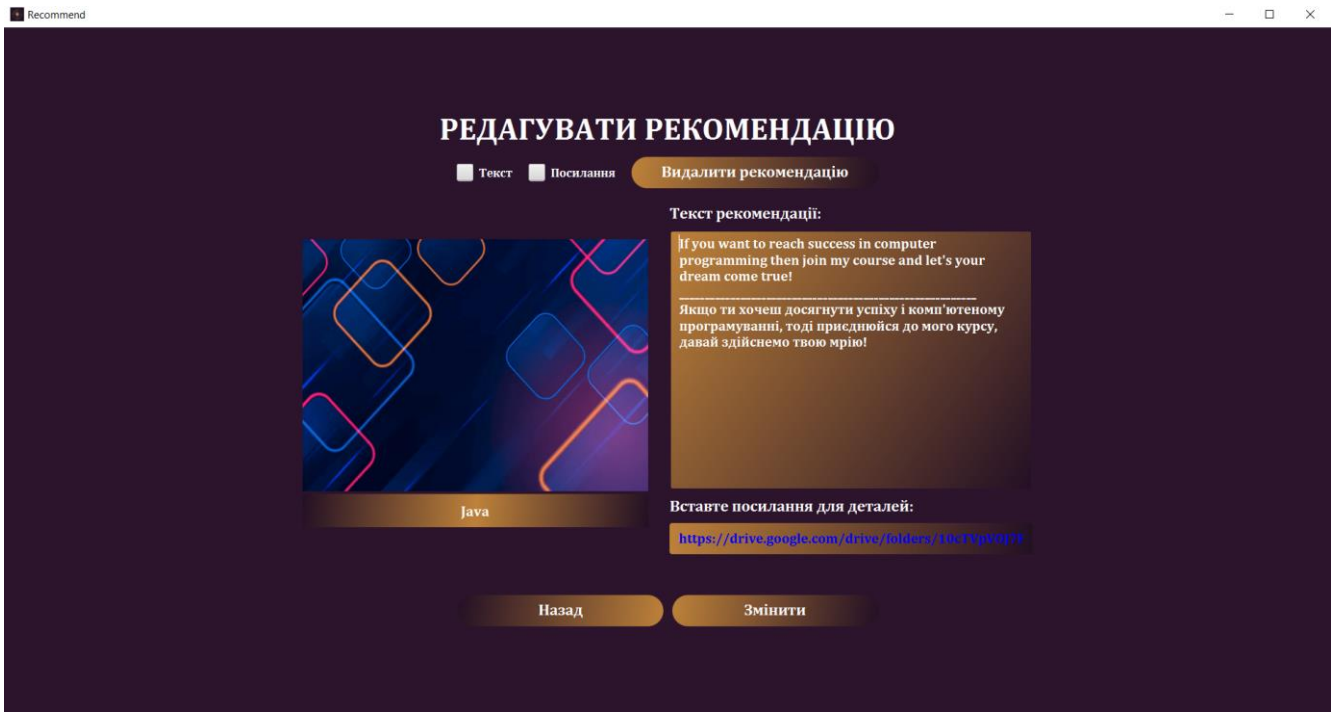

Рисунок 3.40 – Редагування власної рекомендації

Студенти можуть переглядати ці оголошення та рекомендації, які створює викладач.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено комп'ютерну програму, яка має хороший візуальний інтерфейс та допомагає створювати курси та виконувати різні дії над ними за допомогою функціоналу програми.

У ході розробки програми було використано середовище розробки IntelliJ IDEA Scene Builder та інше програмне забезпечення, за допомогою якого було створено якісний читабельний код для реалізації всіх функцій програми для здійснення ефективного онлайн-навчання та створення чудового графічного оформлення програми. Також для того, щоб впевнитися, що все працює як і мало працювати за завданням, було проведено ретельне тестування програми з використанням всіх можливих випадків, які могли викликати будь-які помилки під час роботи програми. Результати тестування підтвердили те, що застосунок є працездатним та виконує всі вимоги, які були передбаченні, помилок не було.

Під час розробки самої програми було розроблено алгоритми таким чином, щоб забезпечити швидке виконання дій у програмі, щоб вона не тормозила і все виконувала правильно, це особливо потрібно було під час роботи програми з базою даних, тому там використовувалися фонові потоки, щоб не блокувати графічний інтерфейс програми та всі дії з базою даних зберігаються у apі файлі, що пришвидшить виконання запитів та надасть можливість в один і той самий час з’єднюватися з базою даних декільком користувачам.

Отримані результати показують, що розроблена комп'ютерна програма є ефективною і вона дуже добре працює з користувачами. Вона дозволяє зменшити проблеми як викладачам так і студентам, бо їм вже не доведеться здійснювати онлайн-навчання використовуючи багато програм, що робить деякий хаос, тут ця програма містить все в собі, також викладачам не потрібно буде переживати за заповнення місця вивантаження своїх файлів у курсі, бо вони всі зберігаються на сервері застосунку.

Дипломну роботу виконано успішно, і у результаті створено комп'ютерну програму, яка відповідає всім поставленим вимогам, завдяки їй можна здійснювати онлайн-навчання репетиторам у легший спосіб для них.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Приватне репетиторство у Східній Європі та Центральній Азії [Електронний ресурс] // Tandfonline. – 2022. – Режим доступу до ресурсу: https://www.tandfonline.com/doi/full/10.1080/03057920903361926

2. Використання освітніх платформ в освітньому середовищі в Україні [Електронний ресурс] // Uej. – 2022. – Режим доступу до ресурсу: https://uej.undip.org.ua/index.php/journal/article/download/619/592?inline=1

3. Топ інструментів онлайн-навчання [Електронний ресурс] // Jta. – 2022. – Режим доступу до ресурсу: https://www.jta.com.ua/knowledge-base/top-15-onlayn-instrumentiv-iaki-znadobliatsia-kozhnomu-vykladachu-na-dystantsiytsi/

4. Характеристики SQLite [Електронний ресурс] // Freehost. – 2020.– Режим доступу до ресурсу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/

5. Class diagrams [Електронний ресурс] // Mermaid. – 2024. – Режим доступу до ресурсу: https://mermaid.js.org/syntax/classDiagram.html.

6. Патерн Model View Controller [Електронний ресурс] // Javarush. – 2023. – Режим доступу до ресурсу: https://javarush.com/ua/groups/posts/uk.2536.chastina-7-oznayomlennja-z-paternom-mvc-model-view-controller

7. Клас BCryptPasswordEncoder [Електронний ресурс] // Spring. – 2010. – Режим доступу до ресурсу: https://docs.spring.io/spring-security/reference/api/java/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html

8. О.М. Васильєв . Програмування мовою Java / О.М. Васильєв. – Тернопіль: Навчальна книга – Богдан, 2021. -696 с.

9. Мова структурованих запитів [Електронний ресурс] // aCode – 2024. – Режим доступу до ресурсу: https://acode.com.ua/sql-intro/

10. Створення Java-програм з графічним інтерфейсом за допомогою JavaFX [Електронний ресурс]// Openjfx – 2024. – Режим доступу до ресурсу: https://uk.myservername.com/java-development-using-eclipse-ide

ДОДАТКИ

ДОДАТОК А

ER-діаграма бази даних застосунку

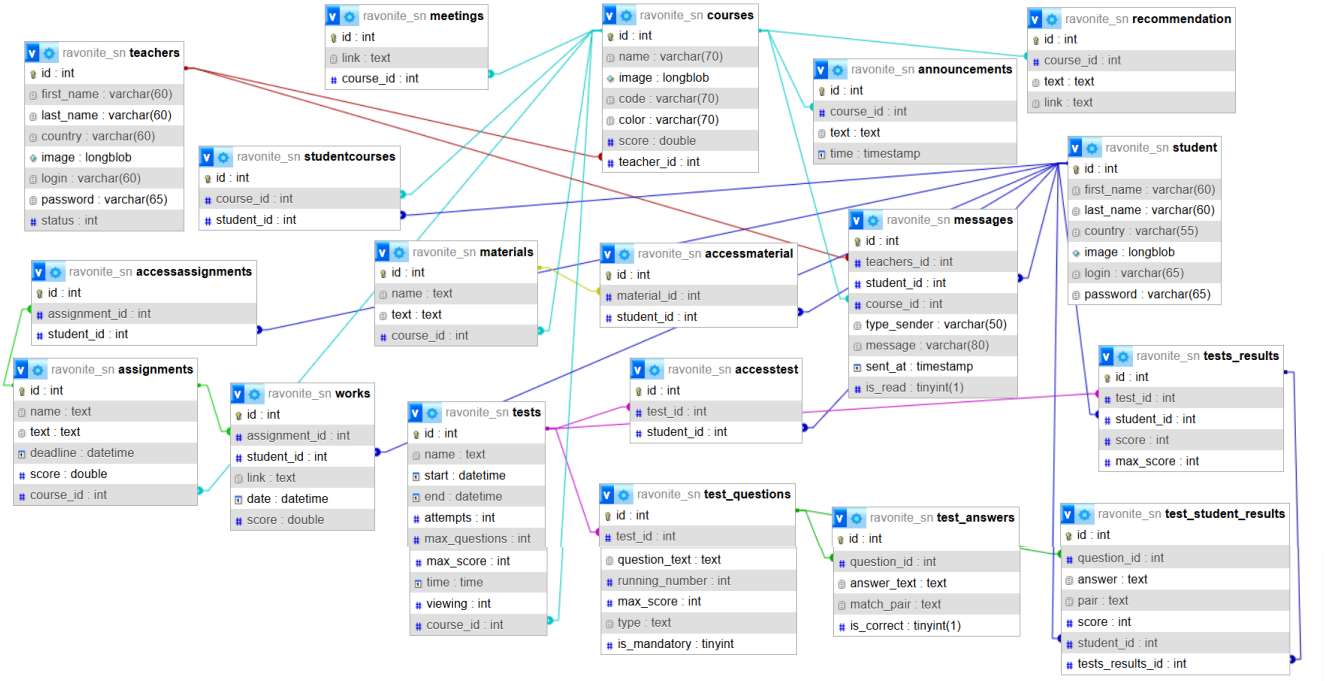

Рисунок А.1 – ER- діаграма бази даних

ДОДАТОК Б

Діаграми класів

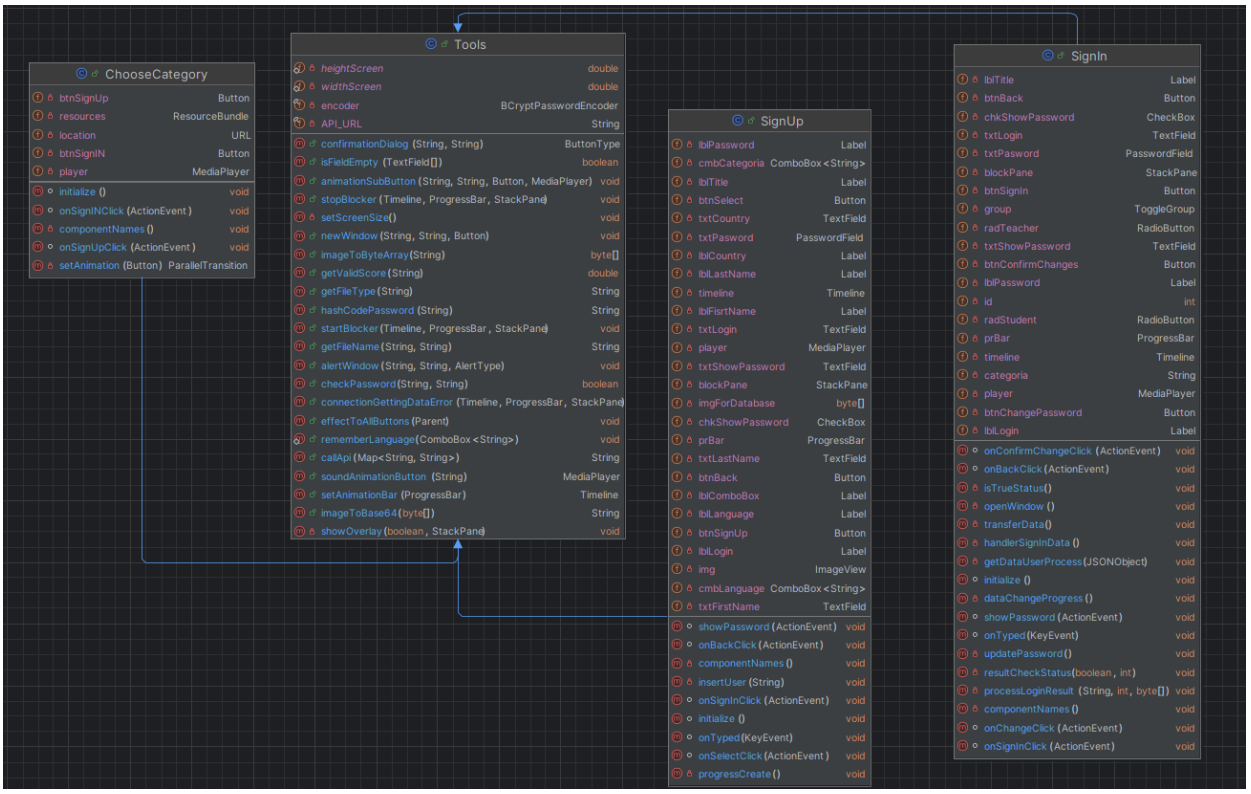

Рисунок Б.1 – Діаграми класів реєстрації та авторизації

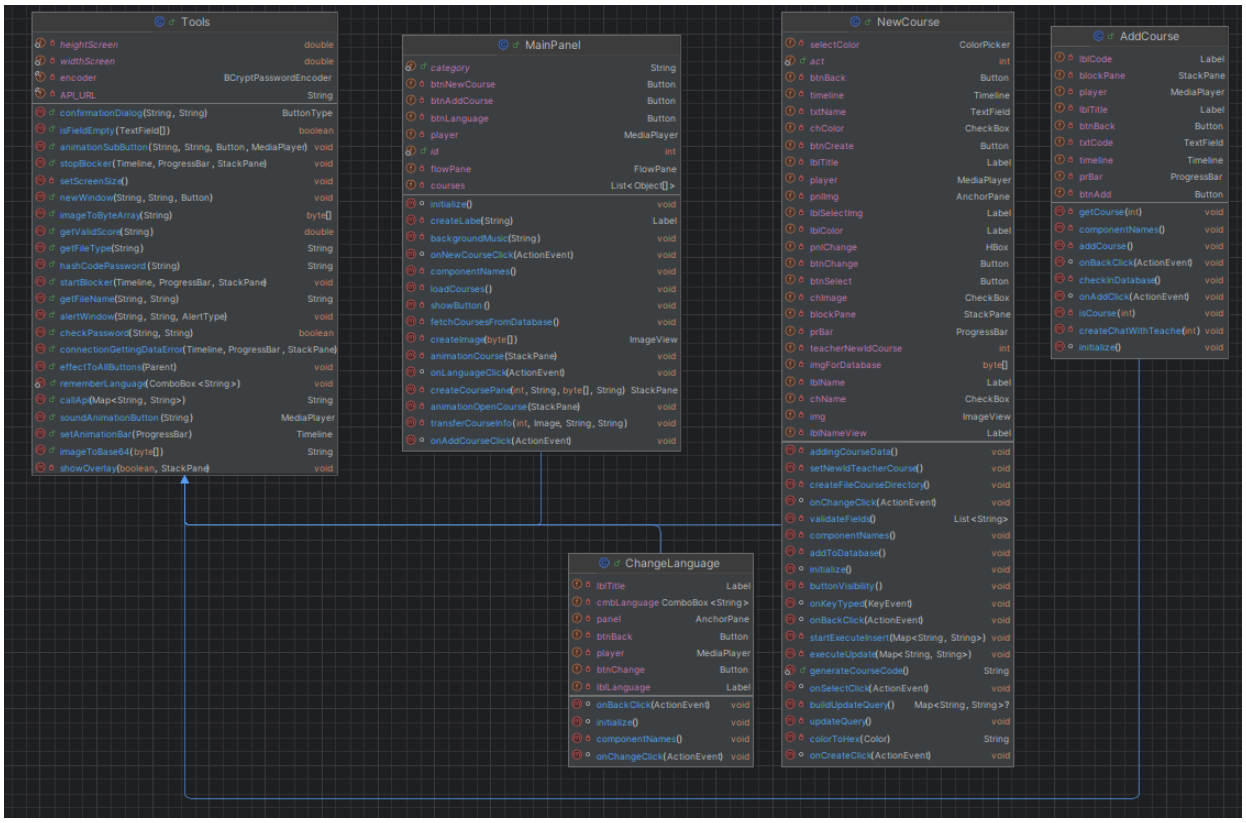

Рисунок Б.2 – Діаграми класів відображення всіх курсів їх створення або додавання та зміну мови інтерфейсу

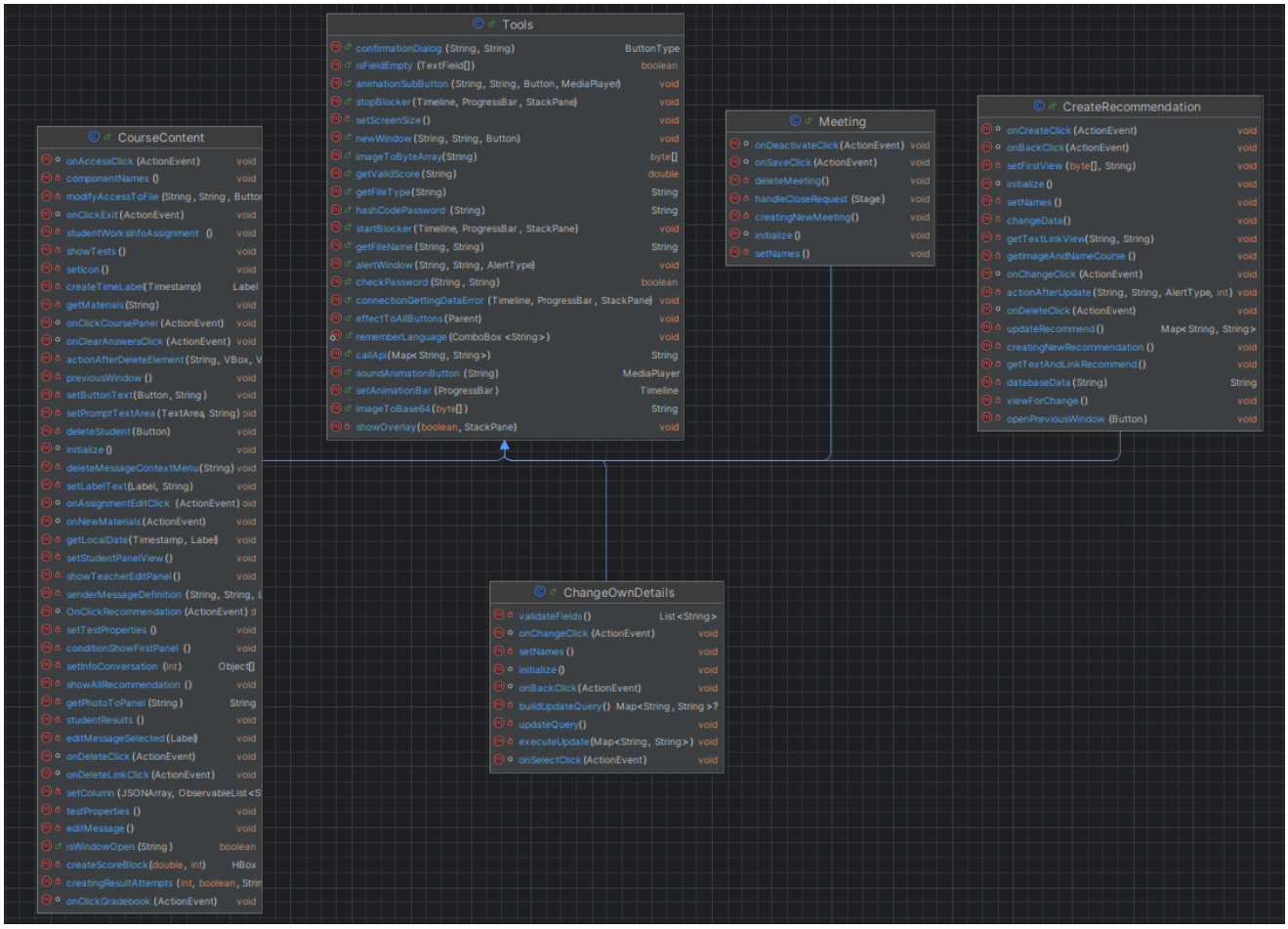

Рисунок Б.3 – Діаграми класів відображення вмісту курсу, зміни деталей свого акаунту, створення рекомендації курсу та відео зустрічі

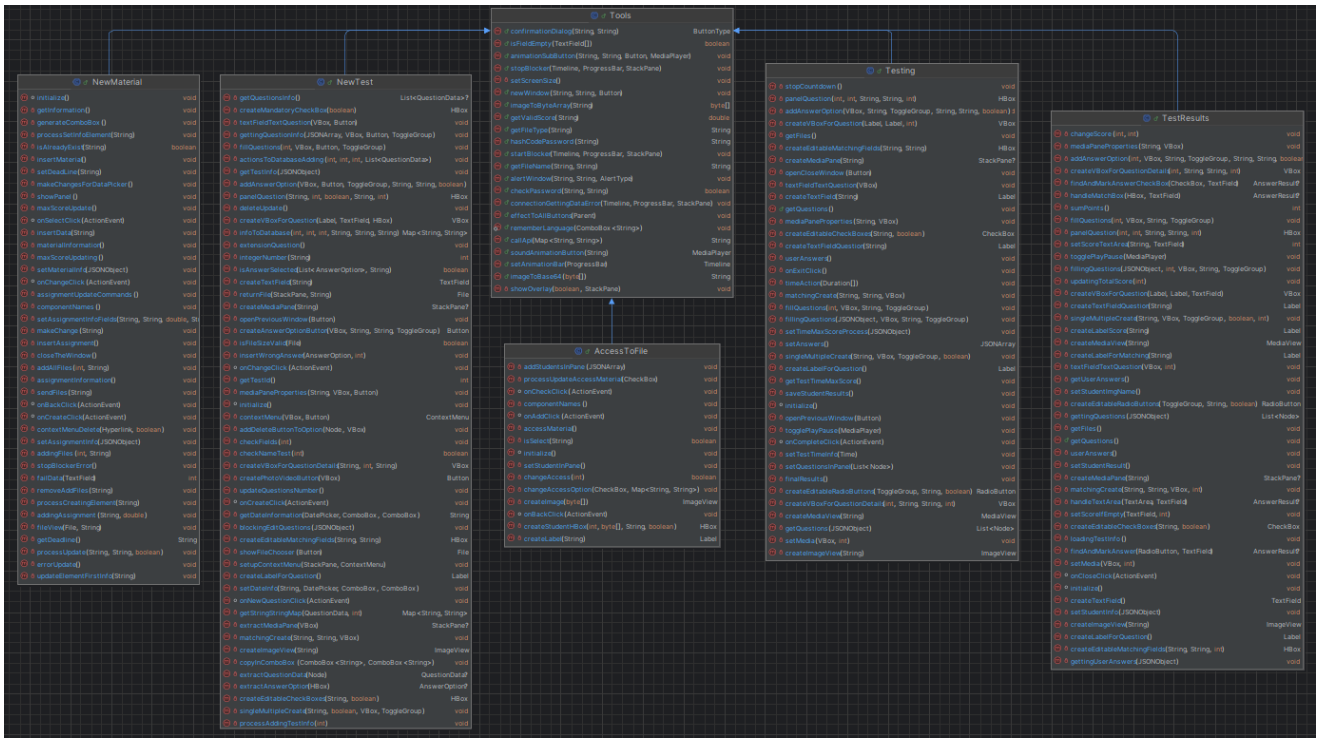

Рисунок Б.4 – Діаграми класів створення матеріалів та завдань, тестів, обмеження студентам на елементи курс, тестування, результати тестування

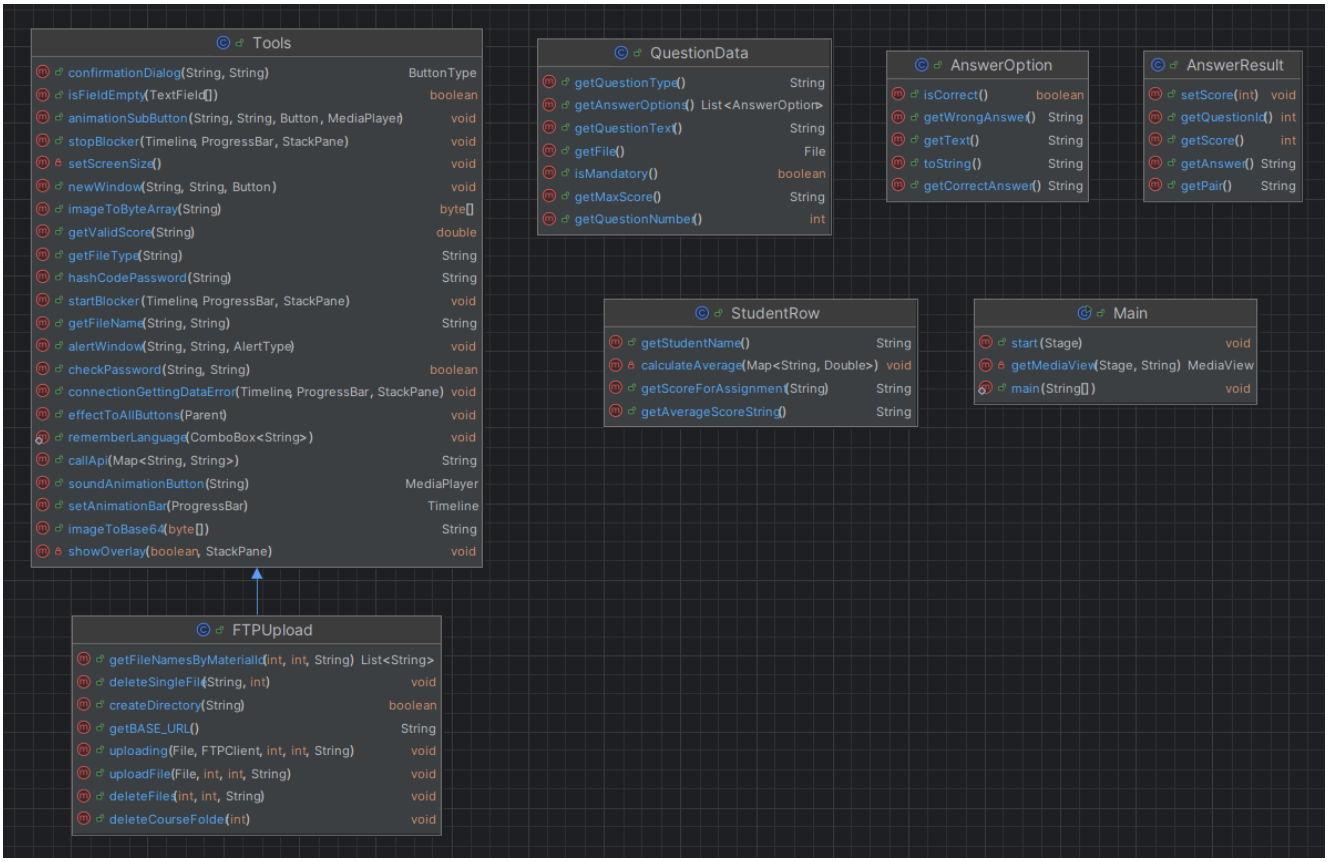

Рисунок Б.5 – Діаграми класів для завантаження файлів курсів на хостинг, формування питань, варіантів та відповідей на питання тесту, формування таблиці успішності студентів та головний метод програми

ДОДАТОК В

Use case діаграма застосунку

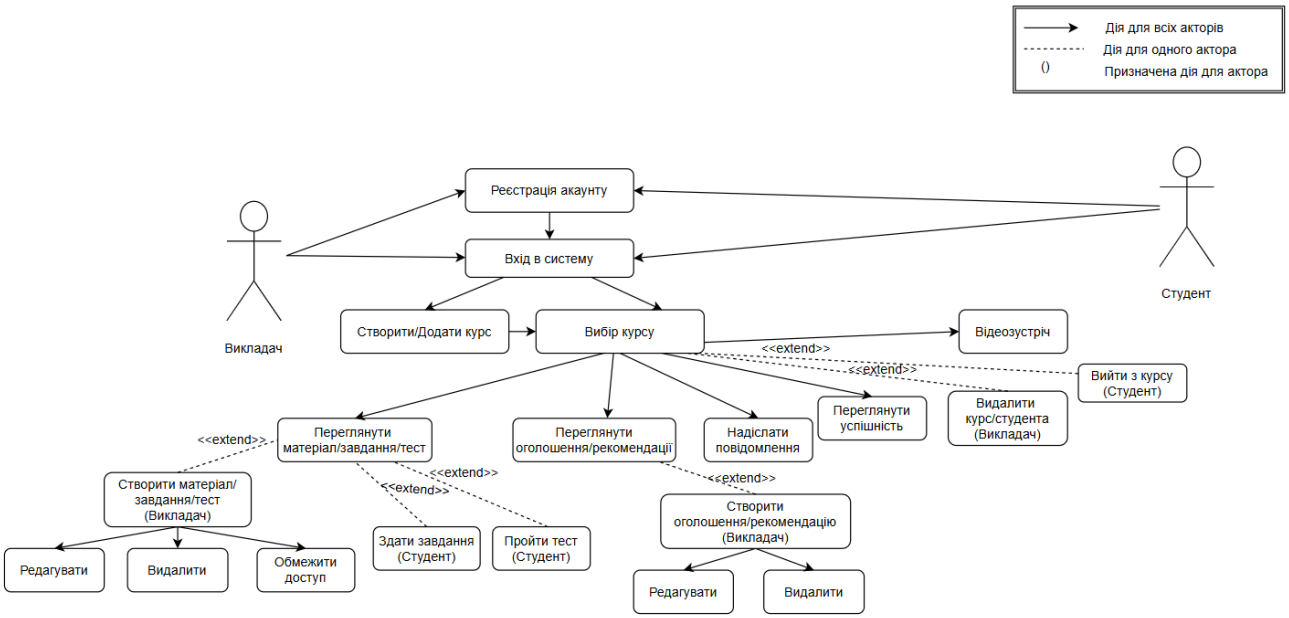

Рисунок В.1 – Use case діаграма

ДОДАТОК Г

Архітектура застосунку

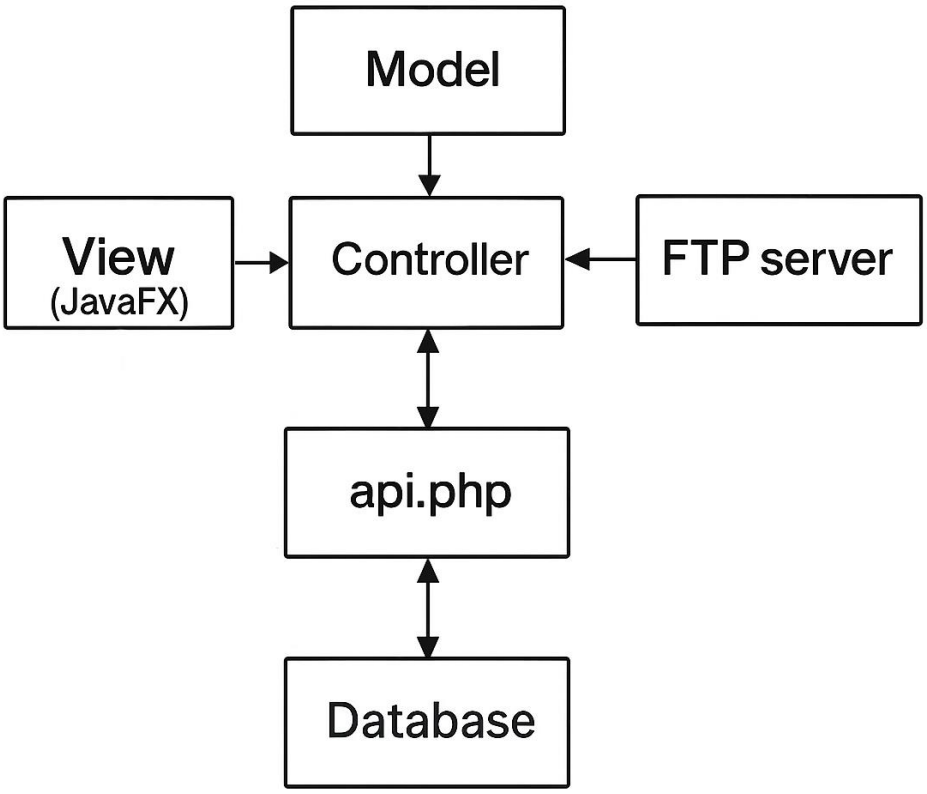

Рисунок Г.1 – Архітектура застосунку за патерном Model-View-Controller(MVC)

ДОДАТОК Д

Програмний код головних класів та процедур програми

```
package com.example.sparksnet;
import javafx.animation.*;
import javafx.application.Platform;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Rectangle2D;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.effect.ColorAdjust;
import javafx.scene.effect.Glow;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.StackPane;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.shape.Rectangle;
import javafx.stage.Screen;
import javafx.stage.Stage;
import javafx.util.Duration;
import java.io.BufferedReader;
import java.io.File;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.Map;

public class Tools {
    private static double widthScreen, heightScreen;
    private final BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
    private final String API_URL = "http://ravonite.zzz.com.ua/api.php";

    public void newWindow(String title, String fxml,Button btn){
        try {
            setScreenSize();
            FXMLLoader loader = new FXMLLoader(Main.class.getResource(fxml));
            Scene scene = new Scene(loader.load(), widthScreen, heightScreen);
            Stage stage = new Stage();
            stage.setTitle(title);
            stage.setScene(scene);
            stage.getIcons().add(new
Image(getClass().getResource("logo_icon.png").toExternalForm()));
            stage.show();
        }catch (Exception e){
```

```
            alertWindow("Error","ERROR!", Alert.AlertType.ERROR);
        };
    }

    private void setScreenSize(){
        if(widthScreen==0) {
            Rectangle2D screenSize = Screen.getPrimary().getBounds();
            widthScreen = screenSize.getWidth();
            heightScreen = screenSize.getHeight() * 0.93;
        }
    }

    public MediaPlayer soundAnimationButton(String path){
        String soundPath = Paths.get(path).toUri().toString();
        Media sound = new Media(soundPath);
        return new MediaPlayer(sound);
    }

    public void alertWindow(String title, String text, Alert.AlertType alertType){
        Alert alert= new Alert(alertType);
        alert.setHeaderText(null);
        alert.setContentText(text);
        alert.setTitle(title);
        alert.getDialogPane().setStyle(
                "-fx-font-family: 'Cambria';" +
                        "-fx-font-size: 16px;" +
                        "-fx-background-color: linear-gradient(to bottom right,
#2c142d, #231025);"
        );
        alert.getDialogPane().lookup(".content").setStyle("-fx-text-fill:
white;");
        alert.showAndWait();
    }

    public boolean isFieldEmpty(TextField... fields) {
        for (TextField field : fields) {
            if (field.getText().trim().isEmpty()) {
                return true;
            }
        }
        return false;
    }

    public String hashCodePassword(String password){
        return encoder.encode(password);
    }

    public boolean checkPassword(String password, String hashedPassword){
        return encoder.matches(password,hashedPassword);
    }

    public static void rememberLanguage(ComboBox<String> cmbLanguage){
        Language.setLanguage(cmbLanguage.getValue());

Language.writeChosenLanguage("components/main_language.txt",cmbLanguage.getValue()
);
```

```
    }

    public byte[] imageToByteArray(String imagePath) throws IOException {
        File file = new File(imagePath);
        return Files.readAllBytes(file.toPath());
    }

    public double getValidScore(String scoreText) {
        try {
            double score = Double.parseDouble(scoreText);
            if (score < 0) {
                Platform.runLater(()->alertWindow("Error", "Score cannot be
negative!", Alert.AlertType.ERROR));
                return -1;
            }
            return score;
        } catch (NumberFormatException e) {
            Platform.runLater(()->alertWindow("Error", "Score must be a valid or
integer number!", Alert.AlertType.ERROR));
            return -1;
        }
    }

    public String getFileType(String fileUrl) {
        try {
            Path path = Paths.get(new URL(fileUrl).getPath());
            String mimeType = Files.probeContentType(path);
            return (mimeType != null) ? mimeType : "Unknown";
        } catch (Exception e) {
            return "Unknown";
        }
    }

    public String getFileName(String url,String letter) {
        try {
            URI uri = new URI(url);
            String path = uri.getPath();
            String fileName = path.substring(path.lastIndexOf('/') + 1);
            return fileName.replaceFirst("_"+letter+"(\\d+)(?=\\.[^\\.]+$)", "");
        } catch (URISyntaxException e) {
            return "File";
        }
    }


    public void connectionGettingDataError(Timeline timeline,ProgressBar
prBar,StackPane blockPane){
        Platform.runLater(()->{
            if(timeline!=null) {
                stopBlocker(timeline, prBar, blockPane);
            }
            alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR);
        });
    }
```

```
    public String callApi(Map<String, String> params) throws IOException {
        StringBuilder postData = new StringBuilder();
        for (Map.Entry<String, String> param : params.entrySet()) {
            if (postData.length() != 0) postData.append('&');
            postData.append(URLEncoder.encode(param.getKey(), "UTF-8"));
            postData.append('=');
            postData.append(URLEncoder.encode(param.getValue(), "UTF-8"));
        }
        byte[] postDataBytes =
postData.toString().getBytes(StandardCharsets.UTF_8);
        URL url = new URL(API_URL);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
        conn.setDoOutput(true);
        conn.getOutputStream().write(postDataBytes);
        BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream(), "UTF-8"));
        StringBuilder response = new StringBuilder();
        String line;
        while ((line = in.readLine()) != null) {
            response.append(line);
        }
        in.close();
        return response.toString();
    }

    public String imageToBase64(byte[] imageBytes) {
        return Base64.getEncoder().encodeToString(imageBytes);
    }

}

package com.example.sparksnet;
import java.io.*;
import java.util.Locale;
import java.util.ResourceBundle;

public class Language {
    private static Locale current;
    private static ResourceBundle bundle;

    public static void setLanguage(String lang) {
        current = new Locale(lang);
        bundle = ResourceBundle.getBundle("language.interface_" + lang, current);
    }

    public static String get(String key) {
        return bundle.getString(key);
    }

    public static String readChosenLanguage(String filePath) {
        String line;
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath)))
{
```

```
            line = reader.readLine();
        } catch (IOException e) {
            line="English";
        }
        return line;
    }

    public static void writeChosenLanguage(String filePath, String line) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath)))
{
            writer.write(line);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

package com.example.sparksnet;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import javafx.animation.Timeline;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.StackPane;
import javafx.scene.media.MediaPlayer;
import javafx.scene.shape.Circle;
import javafx.stage.FileChooser;
import org.json.JSONObject;

public class SignUp extends Tools {
    @FXML
    private StackPane blockPane;
    @FXML
    private ProgressBar prBar;
    @FXML
    private Button btnBack;
    @FXML
    private Button btnSignUp;
    @FXML
    private CheckBox chkShowPassword;
    @FXML
    private ComboBox<String> cmbCategoria;
    @FXML
    private ComboBox<String> cmbLanguage;
    @FXML
    private Label lblComboBox;
    @FXML
    private Label lblCountry;
    @FXML
```

```
    private Label lblFisrtName;
    private MediaPlayer player;
    private byte[] imgForDatabase=null;
    private Timeline timeline;

    @FXML
    void onBackClick(ActionEvent event) {
        animationSubButton("SparksNet","chooseCategory.fxml",btnBack,player);
    }

    @FXML
    void onSignInClick(ActionEvent event) {
        if(isFieldEmpty(txtFirstName, txtLastName, txtCountry, txtLogin,
txtPasword) || cmbCategoria.getValue()==null){
            alertWindow("Error", Language.get("Message.emptyFields"),
Alert.AlertType.ERROR);
        }else {
            new Thread(()->{
                progressCreate();
            }).start();
        }
    }

    private void progressCreate(){
        startBlocker(timeline,prBar,blockPane);
        String categoria= cmbCategoria.getValue().equals("Teacher")
?"teachers":"student";
        Map<String, String> params = new HashMap<>();
        params.put("action", "check_user_exists");
        params.put("login", txtLogin.getText());
        params.put("category", categoria);
        params.put("token", "sparksnet253");
        try {
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            if (json.getBoolean("success") && json.getInt("rowCount") > 0) {
                Platform.runLater(() -> {
                    stopBlocker(timeline, prBar, blockPane);
                    alertWindow("Error", Language.get("Message.loginExists"),
Alert.AlertType.ERROR);
                });
            } else {
                insertUser(categoria);
            }
        } catch (Exception e) {
            e.printStackTrace();
            connectionGettingDataError(timeline, prBar, blockPane);
        }
    }

    private void insertUser(String categoria) throws IOException {
        String imageBase64 = (imgForDatabase != null) ?
imageToBase64(imgForDatabase) : "";
        Map<String, String> params = new HashMap<>();
        params.put("action", "register_user");
        params.put("first_name", txtFirstName.getText());
```

```
        params.put("last_name", txtLastName.getText());
        params.put("country", txtCountry.getText());
        params.put("login", txtLogin.getText());
        params.put("password", hashCodePassword(txtPasword.getText()));
        params.put("image", imageBase64);
        params.put("category", categoria);
        params.put("token", "sparksnet253");
        callApi(params);
        Tools.rememberLanguage(cmbLanguage);
        Platform.runLater(() -> {
            stopBlocker(timeline, prBar, blockPane);
            animationSubButton("SparksNet", "signIn.fxml", btnSignUp, player);
        });
    }

    @FXML
    void showPassword(ActionEvent event) {
        if(chkShowPassword.isSelected()){
            txtShowPassword.setText(txtPasword.getText());
            txtShowPassword.setVisible(true);
            txtPasword.setVisible(false);
        } else {
            txtShowPassword.setVisible(false);
            txtPasword.setVisible(true);
        }
    }

    @FXML
    void onTyped(KeyEvent event) {
        String text = txtLogin.getText();
        txtLogin.setText(text.toLowerCase());
        txtLogin.positionCaret(text.length());
    }

    @FXML
    void onSelectClick(ActionEvent event) throws IOException {
        FileChooser fileChooser= new FileChooser();
        FileChooser.ExtensionFilter filter= new FileChooser.ExtensionFilter("Image
Files","*.jpg","*.png","*.jpeg","*.gif");
        fileChooser.getExtensionFilters().add(filter);
        File file=fileChooser.showOpenDialog(btnSelect.getScene().getWindow());
        if(file!=null){
            if(file.length()>52*1024){
                alertWindow("Error","The image cannot be larger than 52 KB!",
Alert.AlertType.ERROR);
            }
            else {
                imgForDatabase=imageToByteArray(file.getPath());
                img.setImage(new Image(file.toURI().toString()));
            }
        }
    }

    @FXML
    void initialize() {
        Platform.runLater(()->{
```

```
            componentNames();
            player=soundAnimationButton("components/SubButton_sound.mp3");
            cmbLanguage.setValue("English");
            cmbCategoria.getItems().addAll("Teacher", "Student");
            txtLogin.setTextFormatter(new TextFormatter<>(change ->
                    change.getControlNewText().length() > 40 ? null : change)
            );
            img.setClip(new Circle(36.5, 31.5, 31.5));
            timeline=setAnimationBar(prBar);
            effectToAllButtons(btnBack.getScene().getRoot());
        });
    }

    private void componentNames(){
        lblTitle.setText(Language.get("SignUp.title"));
        lblFisrtName.setText(Language.get("SignUp.lblFisrtName"));
        lblLastName.setText(Language.get("SignUp.lblLastName"));
        lblCountry.setText(Language.get("SignUp.lblCountry"));
        lblLanguage.setText(Language.get("SignUp.lblLanguage"));
        lblComboBox.setText(Language.get("SignUp.lblComboBox"));
        lblLogin.setText(Language.get("SignUp.lblLogin"));
        lblPassword.setText(Language.get("SignUp.lblPassword"));
        chkShowPassword.setText(Language.get("SignUp.chkShowPassword"));
        btnBack.setText(Language.get("SignUp.btnBack"));
        btnSignUp.setText(Language.get("SignUp.btnSignUp"));
        btnSelect.setText(Language.get("ChangeOwnDetails.btnSelect"));
    }

}


package com.example.sparksnet;
import javafx.animation.Timeline;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.media.MediaPlayer;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.stage.FileChooser;
import org.json.JSONObject;
import java.io.File;
import java.io.IOException;
import java.security.SecureRandom;
import java.sql.SQLException;
import java.util.*;

public class NewCourse extends Tools {
    @FXML
    private StackPane blockPane;
```

```
    @FXML
    private ProgressBar prBar;
    @FXML
    private Button btnBack;
    @FXML
    private Button btnCreate;
    @FXML
    private Button btnSelect;
    @FXML
    private Button btnChange;
    private MediaPlayer player;
    private int teacherNewIdCourse=0;
    private byte[] imgForDatabase;
    public static int act;
    private Timeline timeline;

    @FXML
    void onBackClick(ActionEvent event) {
        if(act==1){
            animationSubButton("SparksNet", "mainPanel.fxml", btnBack, player);
        }
        else {
            newWindow("SparksNet", "courseContent.fxml", null);
            btnBack.getScene().getWindow().hide();
        }
    }

    @FXML
    void onCreateClick(ActionEvent event) throws SQLException, IOException {
        if(!isFieldEmpty(txtName) && img.getImage()!=null) {
            addToDatabase();
        }
        else {
            alertWindow("Error", Language.get("Message.emptyFields"),
Alert.AlertType.ERROR);
        }
    }

    @FXML
    void onChangeClick(ActionEvent event) {
        updateQuery();
    }

    private void addToDatabase(){
        startBlocker(timeline,prBar,blockPane);
        new Thread(()->addingCourseData()).start();
    }

    private void addingCourseData(){
        setNewIdTeacherCourse();
        if (teacherNewIdCourse != 0) {
            String code = generateCourseCode() + MainPanel.id +
teacherNewIdCourse;
            String color = colorToHex(selectColor.getValue());
            String imageBase64 =
Base64.getEncoder().encodeToString(imgForDatabase);
```

```
            Map<String, String> params = new HashMap<>();
            params.put("action", "add_course");
            params.put("name", txtName.getText());
            params.put("image", imageBase64);
            params.put("code", code);
            params.put("color", color);
            params.put("teacher_id", String.valueOf(MainPanel.id));
            params.put("token", "sparksnet253");
            startExecuteInsert(params);
        }else {
            Platform.runLater(()->stopBlocker(timeline,prBar,blockPane));
        }
    }

    private void startExecuteInsert(Map<String, String> params){
        try {
            callApi(params);
            createFileCourseDirectory();
            Platform.runLater(() -> {
                stopBlocker(timeline, prBar, blockPane);
                animationSubButton("SparksNet", "mainPanel.fxml", btnCreate,
player);
            });
        } catch (Exception e) {
            connectionGettingDataError(timeline, prBar, blockPane);
        }
    }

    private void setNewIdTeacherCourse(){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_new_course_id");
            params.put("table_name", "courses");
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            if (json.getBoolean("success")) {
                teacherNewIdCourse =json.getInt("AUTO_INCREMENT");
            } else {
                teacherNewIdCourse = 1;
            }
        } catch (Exception e) {
            teacherNewIdCourse = 0;
            connectionGettingDataError(timeline, prBar, blockPane);
        }
    }

    public static String generateCourseCode() {
        String symbols = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
        SecureRandom random = new SecureRandom();
        StringBuilder code = new StringBuilder();
        for (int i = 0; i < 8; i++) {
            int index = random.nextInt(symbols.length());
            code.append(symbols.charAt(index));
        }
        return code.toString();
```

```
    }

    private String colorToHex(Color color) {
        int r = (int) (color.getRed() * 255);
        int g = (int) (color.getGreen() * 255);
        int b = (int) (color.getBlue() * 255);
        return String.format("#%02X%02X%02X", r, g, b);
    }


    @FXML
    void onSelectClick(ActionEvent event) throws IOException {
        FileChooser fileChooser= new FileChooser();
        FileChooser.ExtensionFilter filter= new FileChooser.ExtensionFilter("Image
Files","*.jpg","*.png","*.jpeg","*.gif");
        fileChooser.getExtensionFilters().add(filter);
        File file=fileChooser.showOpenDialog(btnSelect.getScene().getWindow());
        if(file!=null){
            if(file.length()>52*1024){
                alertWindow("Error",Language.get("Message.imageError"),
Alert.AlertType.ERROR);
            }
            else {
                imgForDatabase=imageToByteArray(file.getPath());
                img.setImage(new Image(file.toURI().toString()));
                pnlImg.setStyle("-fx-border-width: 0; -fx-border-color:
transparent;");
            }
        }
    }

    @FXML
    void onKeyTyped(KeyEvent event) {
        lblNameView.setText(txtName.getText());
    }

    @FXML
    void initialize() {
        timeline=setAnimationBar(prBar);
        player=soundAnimationButton("components/SubButton_sound.mp3");
        componentNames();
        txtName.setTextFormatter(new TextFormatter<>(change ->
                change.getControlNewText().length() > 35 ? null : change)
        );
        buttonVisibility();
        Platform.runLater(() -> {
            effectToAllButtons(btnBack.getScene().getRoot());
        });
    }

    private void updateQuery() {
        List<String> errors = validateFields();
        if (!errors.isEmpty()) {
            alertWindow("Error", String.join("\n", errors),
Alert.AlertType.ERROR);
            return;
```

```
        }
        Map<String, String> params = buildUpdateQuery();
        if (params==null) {
            alertWindow("Error", Language.get("Message.noSelectedField"),
Alert.AlertType.ERROR);
            return;
        }
        startBlocker(timeline,prBar,blockPane);
        executeUpdate(params);
    }

    private List<String> validateFields() {
        List<String> errors = new ArrayList<>();
        if (chImage.isSelected() && img.getImage() == null) {
            errors.add(Language.get("Message.noImage"));
        }
        if (chName.isSelected() && isFieldEmpty(txtName)) {
            errors.add(Language.get("Message.emptyName"));
        }
        return errors;
    }

    private Map<String, String> buildUpdateQuery() {
        Map<String, String> params = new HashMap<>();
        params.put("action", "update_course");
        if (chImage.isSelected()) {
            params.put("image",
Base64.getEncoder().encodeToString(imgForDatabase));
            CourseContent.imageCourse = img.getImage();
        }
        if (chName.isSelected()) {
            params.put("name", txtName.getText());
            CourseContent.nameCourse = txtName.getText();
        }
        if (chColor.isSelected()) {
            String getColor = colorToHex(selectColor.getValue());
            params.put("color", getColor);
            CourseContent.color=getColor;
        }
        if (params.size() == 1) return null;
        params.put("id", String.valueOf(CourseContent.idCourse));
        params.put("token", "sparksnet253");
        return params;
    }

    private void executeUpdate(Map<String, String> params) {
        new Thread(() -> {
            try {
                callApi(params);
                Platform.runLater(() -> {
                    stopBlocker(timeline, prBar, blockPane);
                    newWindow("SparksNet", "courseContent.fxml", null);
                    btnChange.getScene().getWindow().hide();
                });
            } catch (Exception e) {
                connectionGettingDataError(timeline, prBar, blockPane);
```

```
            }
        }).start();
    }
}


package com.example.sparksnet;
import javafx.scene.control.Alert;
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPFile;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;


public class FTPUpload extends Tools {
    private final String SERVER = "ravonite.zzz.com.ua";
    private final int PORT = 21;
    private final String USERNAME = "ravonite";
    private final String PASSWORD = "Ravonite.rd2004sn";
    private final String REMOTE_DIR = "/ravonite.zzz.com.ua";
    private final String BASE_URL = "http://ravonite.zzz.com.ua/";

    public void uploadFile(File localFile,int courseId,int idMaterial,String
letter) throws IOException {
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect(SERVER, PORT);
            ftpClient.login(USERNAME, PASSWORD);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
            uploading(localFile,ftpClient,courseId,idMaterial,letter);
        } catch (IOException ex) {
            alertWindow("Error", "Can't connect with server!",
Alert.AlertType.ERROR);
        } finally {
            ftpClient.logout();
            ftpClient.disconnect();
        }
    }

    public void uploading(File localFile,FTPClient ftpClient, int courseId, int
idMaterial,String letter) throws IOException {
        FileInputStream inputStream = new FileInputStream(localFile);
        String fileName = localFile.getName();
        int dotIndex = fileName.lastIndexOf('.');
        String name = fileName.substring(0, dotIndex);
        String ext = fileName.substring(dotIndex + 1);
        ftpClient.storeFile(REMOTE_DIR + "/" +courseId+ "/"+
name+"_"+letter+idMaterial+"."+ext, inputStream);
    }

    public boolean createDirectory(String dirName) {
```

```
        FTPClient ftpClient = new FTPClient();
        boolean success = false;
        try {
            ftpClient.connect(SERVER, PORT);
            ftpClient.login(USERNAME, PASSWORD);
            ftpClient.enterLocalPassiveMode();
            String fullPath = REMOTE_DIR + "/" + dirName;
            success = ftpClient.makeDirectory(fullPath);
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException ex) {}
        return success;
    }

    public List<String> getFileNamesByMaterialId(int courseId, int materialId,
String letter) {
        List<String> matchedFileNames = new ArrayList<>();
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect(SERVER, PORT);
            ftpClient.login(USERNAME, PASSWORD);
            ftpClient.enterLocalPassiveMode();
            ftpClient.changeWorkingDirectory(REMOTE_DIR + "/" + courseId);
            FTPFile[] files = ftpClient.listFiles();
            for (FTPFile file : files) {
                if (file.isFile()) {
                    String name = file.getName();
                    int dotIndex = name.lastIndexOf(".");
                    if (dotIndex > 0) {
                        String nameWithoutExtension = name.substring(0, dotIndex);
                        if (nameWithoutExtension.endsWith("_"+letter+materialId))
{
                            matchedFileNames.add(name);
                        }
                    }
                }
            }
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException ex) {}
        return matchedFileNames;
    }

    public String getBASE_URL(){
        return BASE_URL;
    }

    public void deleteSingleFile(String fileName, int courseId) {
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect(SERVER, PORT);
            ftpClient.login(USERNAME, PASSWORD);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
            String targetDir = REMOTE_DIR + "/" + courseId;
            ftpClient.changeWorkingDirectory(targetDir);
```

```
            ftpClient.deleteFile(targetDir + "/" + fileName);
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException ex) {}
    }
}

package com.example.sparksnet;
import java.awt.*;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.*;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.*;
import java.util.List;
import javafx.animation.*;
import javafx.application.Platform;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.MenuItem;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.effect.ColorAdjust;
import javafx.scene.effect.Glow;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.*;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;
import javafx.stage.Window;
import javafx.util.Duration;
import org.json.JSONArray;
import org.json.JSONObject;

public class CourseContent extends Tools {
    private TranslateTransition hideMenu, showMenu;
```

```
    private boolean menuVisible=true;
    private final ColorAdjust darken = new ColorAdjust();
    public static byte[] imgIcon;
    public static int idCourse;
    public static Image imageCourse;
    public static String nameCourse;
    public static String color;
    private String code;
    private int menuWidth;
    private List<Button> allButtons = new ArrayList<>();
    private Timeline chatUpdater;
    private Timestamp timeMessage=Timestamp.valueOf("2024-01-01 00:00:00");
    private boolean isChangeMessage=false;
    private boolean isChangeAnnouncement=false;
    private int idMessageForChange=0;
    private int idAnnouncementForChange=0;
    private int studentConversation=0, lastId=0;
    private int currentMaterial=0;
    private double maxScore=0;
    private int currentRecommendationCourse=0;
    private FTPUpload ftp = new FTPUpload();
    private Timeline timelinePrBar;
    public static int showPanelIndex=0;
    private Map<String, String> assignmentMap =  new HashMap<>();
    private Map<String, Double> assignmentMaxScores = new HashMap<>();
    private Map<String, Double> testmaxScoresTable = new HashMap<>();
    private List<HBox> allRecommendations=new ArrayList<>();

    @FXML
    void OnClickAnnouncement(ActionEvent event){
        clickedButton(btnAnnouncement);
        turnOffAllPanels();
        vBoxAnnouncements.getChildren().clear();
        pnlAnnouncementsBtn.setVisible(true);
vBoxAnnouncements.getScene().getStylesheets().add(getClass().getResource("textAreaA
nnouncement.css").toExternalForm());
        if (MainPanel.category.equals("teachers")) {
            setupAnnouncementArea();
        }
        createAnnouncementsFields();
    }

    @FXML
    void onClickAssignments(ActionEvent event) {
        clickedButton(btnAssignments);
        showAssignments();
    }

    private void showAssignments(){
        showPanelIndex=2;
        turnOffAllPanels();
        pnlAllAssignments.getChildren().clear();
        pnlAssignmentsBtn.setVisible(true);
        getMaterials("assignments");
    }
```

```
@FXML
void onClickTests(ActionEvent event) {
    clickedButton(btnTests);
    showTests();
}

private void showTests(){
    showPanelIndex=3;
    turnOffAllPanels();
    pnlAllTests.getChildren().clear();
    pnlTestsBtn.setVisible(true);
    getMaterials("tests");
}

@FXML
void onClickGradebook(ActionEvent event){
    turnOffAllPanels();
    tableViewResults.getItems().clear();
    tableViewResults.getColumns().clear();
    assignmentMap.clear();
    assignmentMaxScores.clear();
    testmaxScoresTable.clear();
    vBoxAllScores.setVisible(true);
    startBlocker(timelinePrBar,prBar,blockPane);
    creatingTable();
}

private void creatingTable(){
    startBlocker(timelinePrBar,prBar,blockPane);
    new Thread(()->{
        try {
            if(MainPanel.category.equals("teachers")){
                setTableProperties();
            }else{
                studentResults();
            }
            Platform.runLater(()->stopBlocker(timelinePrBar,prBar,blockPane));
        }catch (Exception e){}
    }).start();
}

@FXML
void onClickChats(ActionEvent event) {
    clickedButton(btnChats);
    turnOffAllPanels();
    pnlAllChats.getChildren().clear();
    pnlChatsBtn.setVisible(true);
    addAllChats();
}

@FXML
void onClickCoursePanel(ActionEvent event){
    clickedButton(btnCoursePanel);
    showCoursePanel();
}
```

```
    private void showCoursePanel(){
        showPanelIndex=0;
        turnOffAllPanels();
        pnlCoursePanelBtn.setVisible(true);
        startBlocker(timelinePrBar,prBar,blockPane);
        conditionShowFirstPanel();
    }

    @FXML
    void onClickMaterials(ActionEvent event) {
        clickedButton(btnMaterials);
        showMaterials();
    }

    private void showMaterials(){
        showPanelIndex=1;
        turnOffAllPanels();
        pnlAllMaterials.getChildren().clear();
        pnlMaterialsBtn.setVisible(true);
        getMaterials("materials");
    }

    @FXML
    void onClickMeeting(ActionEvent event) throws SQLException, IOException,
URISyntaxException {
        clickedButton(btnMeeting);
        if (MainPanel.category.equals("teachers")) {
            handleTeacherMeeting();
        } else {
            handleStudentMeeting();
        }
    }

    @FXML
    void onClickStudents(ActionEvent event) {
        clickedButton(btnStudents);
        turnOffAllPanels();
        pnlStudentsBtn.setVisible(true);
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            buildingStudentsPanel();
            Platform.runLater(()->stopBlocker(timelinePrBar,prBar,blockPane));
        }).start();

    }

    @FXML
    void OnClickRecommendation(ActionEvent event) {
        clickedButton(btnRecommendation);
        showRecommendations();
    }

    private void showRecommendations(){
        showPanelIndex=4;
        turnOffAllPanels();
        pnlRecommendationBtn.setVisible(true);
```

```
        pnlAllRecommendation.getChildren().clear();
        showAllRecommendation();
    }

    @FXML
    void onClickExit(ActionEvent event) {
        showPanelIndex=0;
        clickedButton(btnCoursePanel);
        newWindow("SparksNet","mainPanel.fxml",null);
        btnMenu.getScene().getWindow().hide();
    }

    @FXML
    void onMouseEntered(MouseEvent event) {
        Button button=(Button) event.getTarget();
        button.setEffect(new Glow(0.25));
    }

    @FXML
    void onMouseExited(MouseEvent event) {
        Button button=(Button) event.getTarget();
        button.setEffect(null);
    }

    @FXML
    void onClickMenu(ActionEvent event) {
        double startWidth = menuVisible ? menuWidth : 0;
        double endWidth = menuVisible ? 0 : menuWidth;
        Timeline timeline = new Timeline(
                new KeyFrame(Duration.ZERO, new KeyValue(menu.prefWidthProperty(),
startWidth)),
                new KeyFrame(Duration.seconds(0.5), new
KeyValue(menu.prefWidthProperty(), endWidth))
        );
        timeline.play();
        TranslateTransition transition = new
TranslateTransition(Duration.seconds(0.5), menu);
        transition.setToX(menuVisible ? -menuWidth : 0);
        transition.play();
        menuVisible = !menuVisible;
    }

    @FXML
    void onClearChatClick(ActionEvent event) {
        ButtonType confirmation = confirmationDialog("Confirmation", "Do you
really want to clear this chat?");
        if (confirmation == ButtonType.YES) {
            new Thread(()->clearChatProcess()).start();
        }
    }

    private void clearChatProcess(){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "clear_chat");
            params.put("teacher_id", String.valueOf(MainPanel.id));
```

```
            params.put("student_id", String.valueOf(studentConversation));
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            callApi(params);
            timeMessage = Timestamp.valueOf("2024-01-01 00:00:00");
        } catch (Exception e) {
            Platform.runLater(()-
>alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR));
        }
    }

    private void deleteRecord() {
        clickedButton(btnDeleteStudentBtn);
        ButtonType confirmation = confirmationDialog("Confirmation",
Language.get("Message.deleteCourse"));
        if (confirmation == ButtonType.YES) {
            startBlocker(timelinePrBar,prBar,blockPane);
            new Thread(()->deleteStudentRecordProcess(MainPanel.id,true)).start();
        }
        btnDeleteStudentBtn.setEffect(null);
    }

    private void deleteStudentRecordProcess(int idStudent, boolean isCloseWindow){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "delete_student_course_data");
            params.put("student_id", String.valueOf(idStudent));
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            callApi(params);
            if(isCloseWindow) {
                previousWindow();
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void previousWindow(){
        Platform.runLater(()->{
            stopBlocker(timelinePrBar,prBar,blockPane);
            newWindow("SparksNet", "mainPanel.fxml", null);
            btnExit.getScene().getWindow().hide();
        });
    }

    @FXML
    void onDeleteClick(ActionEvent event) {
        ButtonType confirmation = confirmationDialog("Confirmation",
Language.get("Message.deleteCourse"));
        if (confirmation == ButtonType.YES) {
            startBlocker(timelinePrBar,prBar,blockPane);
            new Thread(()->deleteCourseProcess()).start();
        }
    }
```

```
    private void deleteCourseProcess(){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "delete_course");
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            callApi(params);
            ftp.deleteCourseFolder(idCourse);
            previousWindow();
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    @FXML
    void onDeleteStudentBtn(ActionEvent event) {
       deleteRecord();
    }

    @FXML
    void onEditClick(ActionEvent event) {
        clickedButton(btnEdit);
        NewCourse.act=2;
        newWindow("New Course","newCourse.fxml",null);
        btnEdit.getScene().getWindow().hide();
    }

    private void clickedButton(Button button){
        button.setEffect(darken);
    }

    private void setIcon(){
        if(imgIcon!=null){
            icon.setImage(new Image(new ByteArrayInputStream(imgIcon)));
        }
        else {
            icon.setImage(new
Image(getClass().getResource("user.png").toExternalForm()));
        }
    }
    private void getCodeCourse(){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_course_code");
            params.put("id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            if (json.getBoolean("success")) {
                code = json.getString("code");
            }
        } catch (Exception e) {}
    }

    private void getNumberOfStudents() throws IOException {
```

```
        Map<String, String> params = new HashMap<>();
        params.put("action", "get_number_of_students");
        params.put("course_id", String.valueOf(idCourse));
        params.put("token", "sparksnet253");
        String response = callApi(params);
        JSONObject json = new JSONObject(response);
        if (json.getBoolean("success")) {
            int totalStudents = json.getInt("total_students");
            Platform.runLater(() ->
txtNumberOfStudents.setText(String.valueOf(totalStudents)));
        }
    }

    private void getCourseRating(TextField txtRating) throws IOException {
        Map<String, String> params = new HashMap<>();
        params.put("action", "get_course_rating");
        params.put("course_id", String.valueOf(idCourse));
        params.put("token", "sparksnet253");
        String response = callApi(params);
        JSONObject json = new JSONObject(response);
        if (json.getBoolean("success")) {
            int rank = json.getInt("course_rank");
            int total = json.getInt("total_courses");
            String data = rank + "/" + total;
            Platform.runLater(() -> {
                txtRating.setText(data);
                stopBlocker(timelinePrBar, prBar, blockPane);
            });
        }
    }

    private void contextMenuToAvatar(){
        ContextMenu contextMenu = new ContextMenu();
        contextMenu.setStyle("-fx-background-color: linear-gradient(to bottom
right, #231025, #be823a, #231025);" +
                "-fx-font-family: Cambria; -fx-font-weight: bold; -fx-font-size:
14px;");
        MenuItem changeImageItem = new
MenuItem(Language.get("CourseContent.changeOwnDetails"));
        changeImageItem.setStyle("-fx-text-fill: white;");
        changeImageItem.setOnAction(event -> {
            newWindow("SparksNet","changeOwnDetails.fxml",null);
            menu.getScene().getWindow().hide();
        });
        contextMenu.getItems().add(changeImageItem);
        icon.setOnContextMenuRequested(e->{
            if (!contextMenu.isShowing()){
                contextMenu.show(icon,e.getSceneX(),e.getSceneY());
            }
        });
    }

    private void buildingStudentsPanel(){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_all_students");
```

```
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            JSONArray rows = json.getJSONArray("rows");
            for (int i = 0; i < rows.length(); i++) {
                JSONObject row = rows.getJSONObject(i);
                byte[] image = Base64.getDecoder().decode(row.getString("image"));
                String firstName = row.getString("first_name");
                String lastName = row.getString("last_name");
                int id = row.getInt("id");
                Platform.runLater(() -
>pnlStudentsBtn.getChildren().add(createStudentHBox(image, firstName, lastName,
id)));
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private HBox createStudentHBox(byte [] image,String first, String last, int
id){
        ImageView imageView = createImage(image);
        TextField txt = createTextField(first, last);
        HBox hBox = new HBox(10, imageView, txt, createDeleteButton(id));
        hBox.setAlignment(Pos.CENTER_LEFT);
        return hBox;
    }

    private ImageView createImage(byte[] result){
        ImageView imageView= new ImageView();
        imageView.setFitWidth(73);
        imageView.setFitHeight(63);
        imageView.setClip(new Circle(36.5, 31.5, 31.5));
        if(result.length > 0){
            imageView.setImage(new Image(new ByteArrayInputStream(result)));
        }
        else {
            imageView.setImage(new
Image(getClass().getResource("user.png").toExternalForm()));
        }
        return imageView;
    }

    private TextField createTextField(String first, String second){
        TextField txt= new TextField();
        txt.setFont(Font.font("Cambria", FontWeight.BOLD,18));
        txt.setStyle("-fx-text-fill: white; " +
                "-fx-background-color: linear-gradient(to bottom right, #231025,
#be823a, #231025);");
        txt.setAlignment(Pos.CENTER_LEFT);
        txt.setPrefSize(300,38);
        txt.setText(first+" "+second);
        txt.setEditable(false);
        return txt;
    }
```

```
    private Button createButton(String name,int idStudent){
        Button btn=new Button(name);
        btn.setPrefSize(150,38);
        btn.setFont(Font.font("Cambria", FontWeight.BOLD,18));
        btn.setStyle("-fx-text-fill: white; " +
                "-fx-background-color:  linear-gradient(to bottom right, #be823a,
#231025);");
        btn.setUserData(idStudent);
        return btn;
    }

    private Button createDeleteButton(int studentId) {
        Button btnDelete = createButton(Language.get("NewTest.delete"),
studentId);
        btnDelete.setOnAction(e -> {
            deleteStudent(btnDelete);
            btnDelete.setEffect(null);
        });
        return btnDelete;
    }

    private void deleteStudent(Button button) {
        clickedButton(button);
        ButtonType confirmation = confirmationDialog("Confirmation",
Language.get("Message.deleteStudent"));
        if (confirmation == ButtonType.YES) {
            pnlStudentsBtn.getChildren().clear();
            startBlocker(timelinePrBar,prBar,blockPane);
            new Thread(()->{
                deleteStudentRecordProcess((Integer) button.getUserData(),false);
                buildingStudentsPanel();
                Platform.runLater(()->stopBlocker(timelinePrBar,prBar,blockPane));
            }).start();
        }
    }

    private void showTeacherEditPanel(){
        panelEditTeacher.setVisible(true);
        panelStatistics.setVisible(true);
        txtCourseCode.setText("");
        txtCourseCode.setText(code);
        new Thread(()->{
            try {
                getNumberOfStudents();
                getCourseRating(txtRatingStudents);
            }catch (Exception e){
                connectionGettingDataError(timelinePrBar, prBar, blockPane);
            }
        }).start();
    }

    private void showStudentPanelStatistic(){
        panelStudentStatistics.setVisible(true);
        panleStudentGPA.setVisible(true);
        new Thread(()->{
```

```
        try {
            getCourseRating(txtStudentRating);
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar, prBar, blockPane);
        }
    }).start();
}

private void conditionShowFirstPanel(){
    if(MainPanel.category.equals("teachers")){
        showTeacherEditPanel();
    }
    else {
        showStudentPanelStatistic();
    }
    stopBlocker(timelinePrBar,prBar,blockPane);
    pnlCoursePanelBtn.setVisible(true);
    pnlCoursePanel.setVisible(true);
    pnlSecondHalf1.setVisible(true);
}

private void turnOffAllPanels(){
    for (Node node : pnlMain.getChildren()) {
        if (node instanceof Pane pane) {
            pane.setVisible(false);
        }
        if (node == pnlStudentsBtn) {
            pnlStudentsBtn.getChildren().clear();
        }
    }
}

private void addAllChats() {
    allButtons.clear();
    startBlocker(timelinePrBar,prBar,blockPane);
    new Thread(()->gettingChatsInfo()).start();
}

private void gettingChatsInfo(){
    try {
        Map<String, String> params = new HashMap<>();
        params.put("action", "get_chats");
        params.put("user_type",MainPanel.category);
        params.put("user_id", String.valueOf(MainPanel.id));
        params.put("course_id", String.valueOf(idCourse));
        params.put("token", "sparksnet253");
        String response = callApi(params);
        JSONObject json = new JSONObject(response);
        createChat(json.getJSONArray("rows"));
        Platform.runLater(() ->stopBlocker(timelinePrBar,prBar,blockPane));
    }catch (Exception e){
        connectionGettingDataError(timelinePrBar,prBar,blockPane);
    }
}

private void createChat(JSONArray rows){
```

```
        for (int i = 0; i < rows.length(); i++) {
            JSONObject row = rows.getJSONObject(i);
            byte[] image = Base64.getDecoder().decode(row.getString("image"));
            String name=row.getString("first_name")+"
"+row.getString("last_name");
            int userId = row.getInt("user_id");
            int status = row.getInt("status");
            Platform.runLater(() -> buildChats(image, name, userId, status));
        }
    }

    private void buildChats(byte [] image,String name,int id,int status){
        Button button= new Button(name);
        ImageView imageView=createImage(image);
        button.setGraphic(imageView);
        button.setPrefSize(700,30);
        button.setFont(Font.font("Cambria", FontWeight.BOLD,18));
        button.setStyle("-fx-text-fill: white; -fx-background-color: linear-
gradient(to bottom right, #231025, #be823a, #231025);" +
                " -fx-background-radius:10; -fx-border-radius:10;");
        button.setUserData(id);
        button.setAlignment(Pos.CENTER_LEFT);
        button.setOnAction(e->{
            txtName.setText("");
            ImageView view= (ImageView) button.getGraphic();
            showChosenConversation(view.getImage(),button.getText(), (int)
button.getUserData());
        });
        if(status==1){
            button.setEffect(new Glow(0.5));
        }
        pnlAllChats.getChildren().add(button);
        allButtons.add(button);
    }

    private void filterSearchChat(String text){
        pnlAllChats.getChildren().clear();
        for (Button button : allButtons) {
            if (button.getText().toLowerCase().contains(text.toLowerCase()) ||
text.isEmpty()) {
                pnlAllChats.getChildren().add(button);
            }
        }
    }

    private void showChosenConversation(Image imageConversation,String
nameConversation, int id){
        turnOffAllPanels();
        pnlMessages.getChildren().clear();
        pnlConversation.setVisible(true);
        imgConversation.setClip(new Circle(36.5, 31.5, 31.5));
        imgConversation.setImage(imageConversation);
        nameRecipient.setText(nameConversation);
        timeMessage=Timestamp.valueOf("2024-01-01 00:00:00");
        txtMessageText.setText("");
        Object[] info = setInfoConversation(id);
```

```
        String type_sender = (String) info[0];
        String otherPerson = (String) info[1];
        int teacherId = (int) info[2];
        int studentId = (int) info[3];
        setupTextFieldHandler(id,teacherId,studentId,type_sender);
        new Thread(()->{
            messagesFromDatabase(MainPanel.category,otherPerson,id);
            updateMessageReadStatus(teacherId,studentId);

updateConversationIfNeed(teacherId,studentId,MainPanel.category,otherPerson,id);
        }).start();
    }

    private Object[] setInfoConversation(int id){
        String type_sender,otherPerson;
        int teacherId,studentId;
        if(MainPanel.category.equals("teachers")){
            type_sender="teachers";
            teacherId=MainPanel.id;
            studentId=id;
            studentConversation=id;
            otherPerson="student";
        }
        else {
            type_sender="student";
            teacherId=id;
            studentId=MainPanel.id;
            otherPerson="teachers";
        }
        return new Object[]{type_sender, otherPerson, teacherId, studentId};
    }

    private void messagesFromDatabase(String currentPerson,String otherPerson,int
idRecipient) {
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_messages");
            params.put("course_id", String.valueOf(CourseContent.idCourse));
            params.put("current_id", String.valueOf(MainPanel.id));
            params.put("other_id", String.valueOf(idRecipient));
            params.put("current_key", currentPerson);
            params.put("other_key", otherPerson);
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            addingAllMessages(json.getJSONArray("rows"),currentPerson);
            Platform.runLater(() -> scrollWholeConversation.setVvalue(1.0));
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void addingAllMessages(JSONArray rows,String currentPerson){
        for (int i = 0; i < rows.length(); i++) {
            JSONObject row = rows.getJSONObject(i);
            String message = row.getString("message");
```

```
            if (!message.trim().isEmpty()) {
                String typeSender = row.getString("type_sender");
                timeMessage = Timestamp.valueOf(row.getString("sent_at"));
                int id = row.getInt("id");
                Label lbl = createMessageLabel(message);
                Label time = createTimeLabel(timeMessage);
                VBox messageContainer = new VBox(2, lbl, time);
                senderMessageDefinition(currentPerson, typeSender, lbl, id,
messageContainer);
            }
        }
    }

    private void senderMessageDefinition(String currentPerson,String
type_sender,Label lbl, int id,VBox messageContainer){
        Platform.runLater(()->{
            if (currentPerson.equals(type_sender)) {
                styleSentMessage(lbl);
                lbl.setUserData(id);
                messageContainer.setAlignment(Pos.CENTER_RIGHT);
                addContextMenu(lbl);
            } else {
                messageContainer.setAlignment(Pos.CENTER_LEFT);
            }
            pnlMessages.getChildren().add(messageContainer);
        });
    }

    private void addContextMenu(Label lbl){
        ContextMenu contextMenu = contextMenuForMessage(lbl);
        lbl.setOnContextMenuRequested(e -> {
            if (!contextMenu.isShowing()) {
                contextMenu.show(lbl, e.getSceneX(), e.getSceneY());
            }
        });
    }

    private Label createMessageLabel(String message) {
        Label lbl = new Label(message);
        lbl.setPrefWidth(200);
        lbl.setWrapText(true);
        lbl.setFont(new Font("Cambria", 17));
        lbl.setStyle("-fx-text-fill: white; " +
                "-fx-background-color: linear-gradient(to bottom right, #231025,
#be823a, #231025);" +
                "-fx-border-radius: 20; -fx-background-radius: 20");
        lbl.setPadding(new Insets(5));
        return lbl;
    }

    private Label createTimeLabel(Timestamp timestamp) {
        Label time = new Label(transformDataToLocal(timestamp));
        time.setPrefWidth(108);
        time.setFont(new Font("Cambria", 12));
        time.setStyle("-fx-text-fill: white;");
        return time;
```

```
    }

    private void styleSentMessage(Label lbl) {
        ColorAdjust darkenEffect = new ColorAdjust();
        darkenEffect.setBrightness(-0.3);
        lbl.setEffect(darkenEffect);
    }

    private void setupTextFieldHandler(int idReceiver,int teacherId,int
studentId,String type_sender) {
        txtMessageText.setOnKeyPressed(event -> {
            if (event.isControlDown() && event.isShiftDown()) {
                txtMessageText.appendText("\n");
                event.consume();
            }
            if (event.getCode() == KeyCode.ENTER && !isChangeMessage) {
                addMessage(teacherId,studentId,type_sender);
                event.consume();
            }
            else if (event.getCode() == KeyCode.ENTER && isChangeMessage){
                editMessage();
            }
            if(event.getCode()==KeyCode.ESCAPE && isChangeMessage){
                txtMessageText.clear();
                isChangeMessage = false;
                idMessageForChange = 0;
            }
        });
    }

    private void addMessage(int teacherId,int studentId,String type_sender){
        try {
            if (!txtMessageText.getText().trim().isEmpty()) {
                Map<String, String> params = new HashMap<>();
                params.put("action", "insert_message");
                params.put("teacher_id", String.valueOf(teacherId));
                params.put("student_id", String.valueOf(studentId));
                params.put("course_id", String.valueOf(CourseContent.idCourse));
                params.put("type_sender", type_sender);
                params.put("message", txtMessageText.getText());
                params.put("token", "sparksnet253");
                callApi(params);
            }
            txtMessageText.clear();
        }catch (Exception e){
            Platform.runLater(()-
>alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR));
        }
    }

    private void editMessage(){
        try {
            if (!txtMessageText.getText().trim().isEmpty()) {
                Map<String, String> params = new HashMap<>();
                params.put("action", "edit_message");
```

```
                params.put("message", txtMessageText.getText());
                params.put("message_id", String.valueOf(idMessageForChange));
                params.put("token", "sparksnet253");
                callApi(params);
                timeMessage = Timestamp.valueOf("2024-01-01 00:00:00");
                isChangeMessage = false;
                idMessageForChange = 0;
            }
            txtMessageText.clear();
        }catch (Exception e){
            Platform.runLater(()->alertWindow("Error",Language.get("DatabaseMessage.connect"),Alert.AlertType.ERROR));
        }
    }

    private boolean isChanges(int teacherId, int studentId){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "check_changes");
            params.put("teacher_id", String.valueOf(teacherId));
            params.put("student_id", String.valueOf(studentId));
            params.put("course_id", String.valueOf(CourseContent.idCourse));
            params.put("last_time", timeMessage.toString());
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            return json.getBoolean("success");
        } catch (Exception e){}
        return false;
    }

    private void updateConversationIfNeed(int teacherId, int studentId,String currentPerson,String otherPerson,int idRecipient){
        if (chatUpdater != null) {
            chatUpdater.stop();
        }
        chatUpdater = new Timeline(new KeyFrame(Duration.seconds(2), event -> chatUpdaterAction(teacherId,studentId,currentPerson,otherPerson,idRecipient)));
        chatUpdater.setCycleCount(Animation.INDEFINITE);
        chatUpdater.play();
    }

    private void chatUpdaterAction(int teacherId, int studentId,String currentPerson,String otherPerson,int idRecipient){
        if(!pnlConversation.isVisible()){
            chatUpdater.stop();
            return;
        }
        if (isChanges(teacherId,studentId)) {
            pnlMessages.getChildren().clear();
            new Thread(()->{
                updateMessageReadStatus(teacherId,studentId);
                messagesFromDatabase(currentPerson,otherPerson,idRecipient);
            }).start();
        }
```

```
    }


    private void deleteMessageContextMenu(String messageId){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "delete_message");
            params.put("message_id", messageId);
            params.put("token", "sparksnet253");
            callApi(params);
            timeMessage = Timestamp.valueOf("2024-01-01 00:00:00");
        }catch (Exception ex){
            Platform.runLater(()-
>alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR));
        }
    }

    private void editMessageSelected(Label lbl){
        txtMessageText.setText(lbl.getText());
        idMessageForChange= (int) lbl.getUserData();
        isChangeMessage=true;
    }

    private ContextMenu contextMenu(){
        ContextMenu contextMenu = new ContextMenu();
        contextMenu.setStyle("-fx-background-color: linear-gradient(to bottom
right, #2c142d, #231025);" +
                "-fx-font-family: Cambria; -fx-font-weight: bold; -fx-font-size:
14px;");
        return contextMenu;
    }

    private MenuItem createMenuItem(String text, EventHandler<ActionEvent> action)
{
        MenuItem item = new MenuItem(text);
        item.setStyle("-fx-text-fill: white;");
        item.setOnAction(action);
        return item;
    }

    private void createAnnouncementsFields(){
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            try{
                Map<String, String> params = new HashMap<>();
                params.put("action", "get_announcements");
                params.put("course_id", String.valueOf(idCourse));
                params.put("token", "sparksnet253");
                String response = callApi(params);
                JSONObject json = new JSONObject(response);

setAllAnnouncements(setAllAnnouncements(json.getJSONArray("rows")));
            }catch (Exception e){
                connectionGettingDataError(timelinePrBar,prBar,blockPane);
            }
```

```
        }).start();
    }

    private List<Node> setAllAnnouncements(JSONArray rows){
        List<Node> announcementNodes = new ArrayList<>();
        for (int i = 0; i < rows.length(); i++) {
            JSONObject obj = rows.getJSONObject(i);
            String text = obj.getString("text");
            int id = obj.getInt("id");
            Timestamp time = Timestamp.valueOf(obj.getString("time"));
            Node node = createAnnouncementLabel(text, id, time);
            announcementNodes.add(node);
        }
        return announcementNodes;
    }

    private void setAllAnnouncements(List<Node> announcementNodes){
        Platform.runLater(()->{
            vBoxAnnouncements.getChildren().clear();
            vBoxAnnouncements.getChildren().addAll(announcementNodes);
            stopBlocker(timelinePrBar,prBar,blockPane);
        });
    }

    private void deleteAnnouncementContextMenu(String announcementId){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "delete_announcements");
            params.put("announcement_id", announcementId);
            params.put("token", "sparksnet253");
            callApi(params);
        }catch (Exception ex){
            Platform.runLater(()-
>alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR));
        }
    }

    private void editAnnouncementSelected(VBox vBox){
        TextArea label= (TextArea) vBox.getChildren().get(1);
        txtAnnouncement.setText(label.getText());
        idAnnouncementForChange= (int) vBox.getUserData();
        isChangeAnnouncement=true;
    }

    private VBox createAnnouncementLabel(String message, int idAnnouncement,
Timestamp time) {
        Label timeformat=new Label(transformDataToLocal(time));
        timeformat.setFont(Font.font("Cambria", 14));
        timeformat.setStyle("-fx-text-fill: white;");
        TextArea lbl = new TextArea(message);
        lbl.setFont(Font.font("Cambria", FontWeight.BOLD, 18));
        lbl.setPrefHeight(100);
        lbl.getStyleClass().add("text-area-announcement");
        lbl.setEditable(false);
        VBox vBox= new VBox(12,timeformat,lbl);
```

```
        vBox.setPadding((new Insets(10)));
        vBox.setUserData(idAnnouncement);
        lastId=idAnnouncement;
        vBox.setStyle("-fx-background-color:  linear-gradient(to bottom right,
#be823a, #231025); -fx-background-radius:20; -fx-border-radius:20;");
        addMenuForAnnouncement(vBox);
        return vBox;
    }

    private void addAnnouncement(){
        if(!txtAnnouncement.getText().trim().isEmpty()){
            new Thread(() -
>addingEditingAnnouncement("add_announcement",idCourse)).start();
        }
    }

    private void addingEditingAnnouncement(String name,int id){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", name);
            params.put("course_id", String.valueOf(id));
            params.put("text", txtAnnouncement.getText());
            params.put("token", "sparksnet253");
            callApi(params);
            createAnnouncementsFields();
            txtAnnouncement.clear();
        } catch (Exception e) {
            connectionGettingDataError(timelinePrBar, prBar, blockPane);
        }
    }

    private void editAnnouncement(){
        if(!txtAnnouncement.getText().trim().isEmpty()) {
            new Thread(() ->{

addingEditingAnnouncement("edit_announcement",idAnnouncementForChange);
                isChangeAnnouncement = false;
                idAnnouncementForChange = 0;
                lastId=0;
            }).start();
        }
    }

    private void handleTeacherMeeting() throws IOException, URISyntaxException {
        if (!isWindowOpen("Meeting")) {
            ButtonType confirmation = confirmationDialog("Confirmation",
Language.get("Message.startMeeting"));
            if (confirmation == ButtonType.YES) {
                startMeeting();
            }
        } else {
            showMeetingError(Language.get("Message.alreadyOnTheMeeting"));
        }
    }
```

```
    private void handleStudentMeeting() throws SQLException, IOException,
URISyntaxException {
        String meetLink = isMeetingStart();
        if (!meetLink.equals("")) {
            ButtonType confirmation = confirmationDialog("Confirmation",
Language.get("Message.joinMeeting"));
            if (confirmation == ButtonType.YES) {
                openMeet(meetLink);
            }
        } else {
            showMeetingError(Language.get("Message.errorMeeting"));
        }
    }

    private void startMeeting() throws IOException, URISyntaxException {
        openMeet("https://meet.google.com/new");
        newWindow("Meeting", "meeting.fxml", null);
    }

    private void showMeetingError(String message) {
        alertWindow("Error", message, Alert.AlertType.ERROR);
    }

    public void openMeet(String meetUrl) throws IOException, URISyntaxException {
        if (Desktop.isDesktopSupported()) {
            Desktop.getDesktop().browse(new URI(meetUrl));
        }
    }

    private String isMeetingStart(){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_meeting_link");
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            if (json.getBoolean("success")) {
                return json.getString("link");
            }
        } catch (Exception e) {
            alertWindow("Error", Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR);
        }
        return "";
    }

    public boolean isWindowOpen(String title) {
        for (Window window : Stage.getWindows()) {
            if (window instanceof Stage && ((Stage)
window).getTitle().equals(title)) {
                return true;
            }
        }
        return false;
    }
```

```
    @FXML
    void onNewMaterials(ActionEvent event) {
        clickedButton(btnNewMaterial);
        NewMaterial.indexIdent=0;
        newWindow("New Material","newMaterial.fxml",null);
        btnNewMaterial.getScene().getWindow().hide();
    }


    private void getMaterials(String tableName){
        Object[] config = getTableConfig(tableName);
        String table = (String) config[0];
        VBox vBox = (VBox) config[1];
        String nameColumn = (String) config[2];
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            allDetailsElement(tableName,table,vBox,nameColumn);
        }).start();
    }

    private Object[] getTableConfig(String tableName) {
        String table;
        VBox vBox;
        String nameColumn;
        if(tableName.equals("materials")){
            table = "accessmaterial";
            vBox = pnlAllMaterials;
            nameColumn = "material_id";
        } else if(tableName.equals("assignments")){
            table = "accessassignments";
            vBox = pnlAllAssignments;
            nameColumn = "assignment_id";
        } else {
            table = "accesstest";
            vBox = pnlAllTests;
            nameColumn = "test_id";
        }
        return new Object[]{table, vBox, nameColumn};
    }

    private void allDetailsElement(String tableName,String table, VBox vBox,
String nameColumn){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_all_details");
            params.put("course_id", String.valueOf(idCourse));
            params.put("table_name", tableName);
            params.put("table", table);
            params.put("name_column", nameColumn);
            params.put("user_id", String.valueOf(MainPanel.id));
            params.put("category", MainPanel.category);
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONArray array = new JSONArray(response);
            addAllElement(array,tableName,vBox);
```

```
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void addAllElement(JSONArray array,String tableName, VBox vBox){
        List<Node>nodes=new ArrayList<>();
        for (int i = 0; i < array.length(); i++) {
            JSONObject obj = array.getJSONObject(i);
            int materialId = obj.getInt("id");
            String name = obj.getString("name");
            nodes.add(createMaterialPanel(name, materialId, tableName));
        }
        Platform.runLater(()->{
            vBox.getChildren().addAll(nodes);
            stopBlocker(timelinePrBar, prBar, blockPane);
        });
    }

    private HBox createMaterialPanel(String name, int id, String tableName){
        ImageView imageView= new ImageView(new
Image(getClass().getResource(getPhotoToPanel(tableName)).toExternalForm()));
        imageView.setFitWidth(100);
        imageView.setFitHeight(75);
        Label label= new Label(name);
        label.setWrapText(true);
        label.setFont(Font.font("Cambria", FontWeight.BOLD, 20));
        label.setStyle("-fx-text-fill: white; ");
        HBox hBox=new HBox(2,imageView,label);
        hBox.setStyle("-fx-background-color: linear-gradient(to bottom right,
#be823a, #231025);" +
                "-fx-background-radius:20; -fx-border-radius:20");
        hBox.setAlignment(Pos.CENTER_LEFT);
        hBox.setPrefHeight(80);
        hBox.setUserData(id);
        hBox.setOnMouseClicked(e->{
            if(tableName.equals("materials")){
                showContentMaterial(name,id);
            }
            else if(tableName.equals("assignments")){
                showContentAssignment(name,id);
            }
            else {
                showContentTest(name,id);
            }
        });
        return hBox;
    }

    private String getPhotoToPanel(String tableName){
        String photo="";
        if(tableName.equals("materials")){
            photo="materials_icon1.png";
        }else if(tableName.equals("assignments")){
            photo="assingments_icon.png";
        }
```

```
    else {
        photo="tests_icon.png";
    }
    return photo;
}

private void showContentMaterial(String name,int id){
    turnOffAllPanels();
    vBoxMaterialContent.setVisible(true);
    lblMaterialTitle.setText(name);
    flowPaneMaterial.getChildren().clear();
    currentMaterial=id;
    setTextMaterial(id);
    allMaterials(id,flowPaneMaterial,"");
}

private void showContentAssignment(String name,int id){
    turnOffAllPanels();
    panelDoublePanelAssignment.setVisible(true);
    lblAssignmentTitle.setText(name);
    flowPaneAssignment.getChildren().clear();
    currentMaterial=id;
    setTextAssignment(id);
    setPropertiesToContentAssignment();
    allMaterials(id,flowPaneAssignment,"a");
}

private void showContentTest(String name,int id){
    turnOffAllPanels();
    lblStartDate.setText(Language.get("CourseContent.lblStartDate"));
    lblEndDate.setText(Language.get("CourseContent.lblEndDate"));
    lblMaxAttempts.setText(Language.get("CourseContent.lblMaxAttempts"));
    lblTestLimit.setText(Language.get("CourseContent.lblTestLimit"));
    vBoxTestContent.setVisible(true);
    lblTestTitle.setText(name);
    pnlAllTestAttempts.getChildren().clear();
    currentMaterial=id;
    setTestProperties();
    if(MainPanel.category.equals("student")){
        btnStartTest.setDisable(true);
        new Thread(()->{
            isShowStartButton();
            setTextButtonViewing(false);
        }).start();
    }else {
        new Thread(()->setTextButtonViewing(true)).start();
    }
}

private void setTestProperties() {
    startBlocker(timelinePrBar,prBar,blockPane);
    new Thread(()->testProperties()).start();
}

private void testProperties(){
    try{
```

```
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_test_properties");
            params.put("test_id", String.valueOf(currentMaterial));
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            setTestDetails(json);
        }catch (Exception e){
            e.printStackTrace();
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void setTestDetails(JSONObject json){
        if (json.getBoolean("success")) {
            String start = json.getString("start");
            String end = json.isNull("end") ? "" : json.getString("end");
            int attempts = json.getInt("attempts");
            String timeLimit = json.optString("time", " -");
            Platform.runLater(() -> {
                getLocalDate(Timestamp.valueOf(start), lblStartDate);
                getLocalDate(end.isEmpty()?null:Timestamp.valueOf(end),
lblEndDate);

lblMaxAttempts.setText(Language.get("CourseContent.lblMaxAttempts") + attempts);
                lblTestLimit.setText(Language.get("CourseContent.lblTestLimit") +
timeLimit);
            });
        }
    }

    private void getLocalDate(Timestamp timestamp, Label lbl){
        if (timestamp != null) {
            lbl.setText(lbl.getText() + " " + transformDataToLocal(timestamp));
        }
        else {
            lbl.setText(lbl.getText() + " -");
        }
    }

    private void isShowStartButton() {
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "can_start_test");
            params.put("test_id", String.valueOf(currentMaterial));
            params.put("student_id", String.valueOf(MainPanel.id));
            params.put("token", "sparksnet253");
            String response = callApi(params);
            JSONObject json = new JSONObject(response);
            if (json.getBoolean("can_start")) {
                Platform.runLater(() -> btnStartTest.setDisable(false));
            }
        } catch (Exception e) {
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }
```

```
    private void setTextButtonViewing(boolean isTextSetButton){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_test_viewing");
            params.put("test_id", String.valueOf(currentMaterial));
            params.put("token", "sparksnet253");
            if(isTextSetButton){
                setTextViewing(callApi(params));
            }else {
                showReviewTestResults(callApi(params));
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void setTextViewing(String response){
        JSONObject json = new JSONObject(response);
        if (json.getBoolean("success")) {
            int viewing = json.getInt("viewing");
            Platform.runLater(() -> {
                if (viewing == 0) {

btnTestAllowViewing.setText(Language.get("CourseContent.btnTestAllowViewing"));
                } else {

btnTestAllowViewing.setText(Language.get("CourseContent.btnTestAllowViewingOther")
);
                }
                stopBlocker(timelinePrBar, prBar, blockPane);
            });
        }
    }

    private void showReviewTestResults(String response) throws SQLException {
        JSONObject json = new JSONObject(response);
        if (json.getBoolean("success")) {
            createAttemptsResultPanel(json.getInt("viewing") == 0);
        }
    }

    private void createAttemptsResultPanel(boolean disableBtn){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_test_attempts");
            params.put("test_id", String.valueOf(currentMaterial));
            params.put("student_id", String.valueOf(MainPanel.id));
            params.put("token", "sparksnet253");
            JSONObject response=new JSONObject(callApi(params));
            getAttemptsResults(response,disableBtn);
            Platform.runLater(() -> stopBlocker(timelinePrBar, prBar, blockPane));
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }
```

```
    private void getAttemptsResults(JSONObject response,boolean disableBtn){
        if (response.getBoolean("success")) {
            JSONArray attempts = response.getJSONArray("rows");
            for (int i = 0; i < attempts.length(); i++) {
                JSONObject attempt = attempts.getJSONObject(i);
                int id = attempt.getInt("id");
                int score = attempt.getInt("score");
                int maxScore = attempt.getInt("max_score");
                int index = i + 1;
                Platform.runLater(() ->
pnlAllTestAttempts.getChildren().add(creatingResultAttempts(index, disableBtn,
score + "/" + maxScore, id)));
            }
        }
    }

    private Label getScoreTestReview(boolean disableBtn,String points){
        Label result=new Label(disableBtn?"-":points);
        result.setStyle("-fx-text-fill:white;");
        result.setFont(Font.font("Cambria",FontWeight.BOLD,30));
        result.setPrefWidth(150);
        result.setAlignment(Pos.CENTER);
        return  result;
    }

    private Button getButtonReview(boolean disableBtn, int id, int studentId) {
        Button button=new Button(Language.get("CourseContent.buttonReview"));
        button.setFont(Font.font("Cambria",FontWeight.BOLD,22));
        button.setStyle("-fx-text-fill:white; -fx-background-color:  linear-
gradient(to bottom right, #231025, #be823a, #231025);");
        button.setOnAction(e->buttonReviewAction(id,studentId));
        button.setPrefSize(150,26);
        button.setDisable(disableBtn);
        return button;
    }

    private void buttonReviewAction(int id, int studentId){
        if (!isWindowOpen("Results")) {
            TestResults.studentId = studentId;
            TestResults.currentTestResultId = id;
            TestResults.idTest = currentMaterial;
            newWindow("Results", "testResults.fxml", null);
        }else {
            alertWindow("Error",Language.get("Message.resultsOpen"),
Alert.AlertType.ERROR);
        }
    }

    private void setPropertiesToContentAssignment(){
        if(MainPanel.category.equals("student")){
            vBoxAssignmentContent.getChildren().remove(hBoxButtonsAssignment);
            new Thread(()->linkScoreStudent()).start();
        }else{

panelDoublePanelAssignment.getChildren().remove(vBoxAssignmentStudent);
```

```
        }
    }

    private void linkScoreStudent(){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_student_work");
            params.put("assignment_id", String.valueOf(currentMaterial));
            params.put("student_id", String.valueOf(MainPanel.id));
            params.put("token", "sparksnet253");
            JSONObject response =new JSONObject(callApi(params));
            setlinkScore(response);
        } catch (Exception e) {
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void setlinkScore(JSONObject response){
        if (response.getBoolean("success")) {
            String link = response.getString("link");
            double studentScore = response.getDouble("score");
            Platform.runLater(() -> {
                txtLink.setText(link);
                lblPoints.setText(studentScore != -1 && maxScore != 0.0 ?
studentScore + "/" + maxScore : "");
                makeUrlWork();
            });
        } else {
            Platform.runLater(() -> {
                btnDeleteLink.setManaged(false);
                btnDeleteLink.setVisible(false);
            });
        }
    }

    private void makeUrlWork(){
        btnSendWork.setDisable(true);
        txtLink.setEditable(false);
        txtLink.setStyle("-fx-background-color:  linear-gradient(to bottom right,
#231025, #be823a); -fx-text-fill: blue; -fx-underline: true;");
        txtLink.setOnMouseClicked(e->{
            try {
                Desktop.getDesktop().browse(new URI(txtLink.getText()));
            } catch (Exception ex) {}
        });
    }

    public void setTextMaterial(int id) {
        imageElement.setImage(new
Image(getClass().getResource("materials_icon1.png").toExternalForm()));
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            try {
                getTextMaterial(id);
            }catch (SQLException e){}
        }).start();
```

```
    }

    private void getTextMaterial(int id) throws SQLException {
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_material_text");
            params.put("material_id", String.valueOf(id));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            if (response.getBoolean("success")) {
                String text = response.getString("text");
                Platform.runLater(() -> {
                    txtTextMaterial.setText(text);
                    txtTextMaterial.setPrefHeight(text.equals("") ? 0 : 90);
                });
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    public void setTextAssignment(int id) {
        imageElement.setImage(new
Image(getClass().getResource("assingments_icon.png").toExternalForm()));
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            try{
                Map<String, String> params = new HashMap<>();
                params.put("action", "get_assignment_text");
                params.put("assignment_id", String.valueOf(id));
                params.put("token", "sparksnet253");
                JSONObject response = new JSONObject(callApi(params));
                setTextInfoAssignment(response);
            }catch (Exception e){
                connectionGettingDataError(timelinePrBar,prBar,blockPane);
            }
        }).start();
    }

    private void setTextInfoAssignment(JSONObject response){
        if (response.getBoolean("success")) {
            String text = response.getString("text");
            double score = response.getDouble("score");
            String deadlineStr = response.optString("deadline", null);
            Timestamp deadline = deadlineStr != null && !deadlineStr.isEmpty() ?
Timestamp.valueOf(deadlineStr.replace("T", " ")) : null;
            String forLabel = setLabelInfoAssignment(deadline, score);
            boolean isShow = !forLabel.isEmpty();
            updateAssignmentPaneUI(text, score, forLabel, isShow);
        }
    }

    private String setLabelInfoAssignment(Timestamp timestamp, double score){
        String forLabel="";
        if(timestamp!=null){
            String formatted = transformDataToLocal(timestamp);
```

```
            forLabel+=Language.get("CourseContent.deadline")+" "+formatted;
        }
        forLabel+=!forLabel.equals("") && score>0.0?"  |  ":"";
        forLabel+=score>0.0?Language.get("CourseContent.maxScore")+" "+score:"";
        return  forLabel;
    }

    private void updateAssignmentPaneUI(String text,double score,String forLabel,
boolean isShow){
        Platform.runLater(()->{
            txtTextAssignment.setText(text);
            txtTextAssignment.setPrefHeight(text.equals("")?0:90);
            maxScore=score;
            lblScoreAndDeadline.setText(forLabel);
            lblScoreAndDeadline.setVisible(isShow);
            lblScoreAndDeadline.setManaged(isShow);
        });
    }

    private String transformDataToLocal(Timestamp timestamp){
        Locale userLocale = Locale.getDefault();
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT).withLocale(userLocale);
        return timestamp.toLocalDateTime().format(formatter);
    }

    public void allMaterials(int id, FlowPane pane, String letter){
        new Thread(()->{
            List<String>
filesList=ftp.getFileNamesByMaterialId(CourseContent.idCourse,id,letter);
            Platform.runLater(()->{
                String baseUrl=ftp.getBASE_URL()+CourseContent.idCourse+"/";
                for (String name:filesList){
                    String url=baseUrl+name;
                    pane.getChildren().add(files(url,letter));
                }
                stopBlocker(timelinePrBar,prBar,blockPane);
            });
        }).start();
    }

    @FXML
    void onAccessClick(ActionEvent event) {
        modifyAccessToFile("accessmaterial","material_id",btnAccess);
    }

    private void modifyAccessToFile(String tableAccess,String name,Button btn){
        AccessToFile.materialId=currentMaterial;
        AccessToFile.accessTable=tableAccess;
        AccessToFile.nameColumnTableId=name;
        newWindow("SparksNet","accessToFile.fxml",null);
        btn.getScene().getWindow().hide();
    }

    @FXML
    void onMaterialDeleteClick(ActionEvent event) throws SQLException {
```

```
stepsToDeleteElement("materials","",pnlAllMaterials,pnlMaterialsBtn,vBoxMaterialCo
ntent);
    }

    private void stepsToDeleteElement(String tableName, String letter, VBox
pnlAllElements, VBox pnlElementBtn,Node currentPane) throws SQLException {
        ButtonType confirmation = confirmationDialog("Confirmation",
Language.get("Message.deleteElement"));
        if (confirmation == ButtonType.YES) {
            startBlocker(timelinePrBar,prBar,blockPane);
            new Thread(()-
>deletingElementInfo(tableName,letter,pnlAllElements,pnlElementBtn,currentPane)).s
tart();
        }
    }

    private void deletingElementInfo(String tableName, String letter, VBox
pnlAllElements, VBox pnlElementBtn,Node currentPane){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "delete_element");
            params.put("table", tableName);
            params.put("id", String.valueOf(currentMaterial));
            params.put("token", "sparksnet253");
            callApi(params);
            ftp.deleteFiles(idCourse,currentMaterial,letter);

actionAfterDeleteElement(tableName,pnlAllElements,pnlElementBtn,currentPane);
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void actionAfterDeleteElement(String tableName,VBox pnlAllElements,
VBox pnlElementBtn,Node currentPane){
        Platform.runLater(()->{
            stopBlocker(timelinePrBar,prBar,blockPane);
            pnlAllElements.getChildren().clear();
            currentPane.setVisible(false);
            pnlElementBtn.setVisible(true);
            getMaterials(tableName);
        });
    }

    @FXML
    void onMaterialEditClick(ActionEvent event) {
        openEditResource(1,btnMaterialEdit);
    }

    private void openEditResource(int editId,Button btn){
        NewMaterial.indexIdent=editId;
        NewMaterial.idMaterial=currentMaterial;
        newWindow("SparksNet","newMaterial.fxml",null);
        btn.getScene().getWindow().hide();
    }
```

```
    @FXML
    void onNewAssignment(ActionEvent event) {
        clickedButton(btnNewAssignment);
        NewMaterial.indexIdent=2;
        newWindow("New Assignment","newMaterial.fxml",null);
        btnNewAssignment.getScene().getWindow().hide();
    }

    @FXML
    void onNewTest(ActionEvent event) {
        NewTest.indexAct=0;
        NewTest.idTestEdit=0;
        newWindow("New Test","newTest.fxml",null);
        btnExit.getScene().getWindow().hide();
    }

    @FXML
    void onAccessAssignmentClick(ActionEvent event) {

modifyAccessToFile("accessassignments","assignment_id",btnAccessAssignment);
    }

    @FXML
    void onAssignmentDeleteClick(ActionEvent event) throws SQLException {

stepsToDeleteElement("assignments","a",pnlAllAssignments,pnlAssignmentsBtn,panelDo
ublePanelAssignment);
    }

    @FXML
    void onAssignmentEditClick(ActionEvent event) {
        openEditResource(3,btnAssignmentlEdit);
    }

    @FXML
    void onDeleteLinkClick(ActionEvent event) {
        ButtonType confirmation = confirmationDialog("Confirmation", "Do you want
to delete your work?");
        if (confirmation == ButtonType.YES) {
            try {
                Map<String, String> params = new HashMap<>();
                params.put("action", "delete_work_link");
                params.put("assignment_id", String.valueOf(currentMaterial));
                params.put("student_id", String.valueOf(MainPanel.id));
                params.put("token", "sparksnet253");
                callApi(params);
                clearFieldsWork();
            }catch (Exception e){
                alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR);
            }
        }
    }

    private void clearFieldsWork(){
```

```
        txtLink.clear();
        txtLink.setOnMouseClicked(null);
        txtLink.setStyle("-fx-background-color:  linear-gradient(to bottom right,
#231025, #be823a); -fx-text-fill: white; -fx-underline: false;");
        txtLink.setEditable(true);
        btnDeleteLink.setManaged(false);
        btnDeleteLink.setVisible(false);
        btnSendWork.setDisable(false);
    }


    @FXML
    void onWorksClick(ActionEvent event){
        turnOffAllPanels();
        pnlForAllWorks.setVisible(true);
        flowPaneAllWorks.getChildren().clear();
        buildPaneWorkStudent();
    }

    private void buildPaneWorkStudent(){
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            studentWorksInfoAssignment();
            Platform.runLater(()->stopBlocker(timelinePrBar,prBar,blockPane));
        }).start();
    }

    private void studentWorksInfoAssignment() {
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_student_works_info");
            params.put("assignment_id", String.valueOf(currentMaterial));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            if (response.getBoolean("success")) {
                addAllWorksOnPanel(response.getJSONArray("rows"));
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void addAllWorksOnPanel(JSONArray students){
        for (int i = 0; i < students.length(); i++) {
            JSONObject student = students.getJSONObject(i);
            byte[] image = Base64.getDecoder().decode(student.getString("image"));
            String first = student.getString("first_name");
            String last = student.getString("last_name");
            String link = student.getString("link");
            Timestamp timestamp = Timestamp.valueOf(student.getString("date"));
            double resScore = student.getDouble("score");
            int studentIdentify = student.getInt("id");
            Platform.runLater(() ->
flowPaneAllWorks.getChildren().add(createPaneWorkStudent(image, first, last, link,
timestamp, resScore, studentIdentify)));
        }
```

```
    }

    private VBox createPaneWorkStudent(byte[] image,String first, String
last,String link,Timestamp timestamp,double resScore,int studentIdentify){
        VBox vBox = createStyledVBox();

vBox.getChildren().addAll(createTopBlock(image,first,last),createHyperlinkBlock(li
nk),createDateBlock(timestamp));
        if (maxScore > 0.0) {
            vBox.getChildren().add(createScoreBlock(resScore,studentIdentify));
        }
        return vBox;
    }

    private VBox createStyledVBox() {
        VBox vBox = new VBox(5);
        vBox.setStyle("-fx-background-color: linear-gradient(to bottom right,
#be823a, #231025); -fx-background-radius: 10; -fx-border-radius: 10;");
        vBox.setAlignment(Pos.TOP_CENTER);
        vBox.setPadding(new Insets(5));
        vBox.setPrefSize(300, 150);
        return vBox;
    }

    private HBox createTopBlock(byte[] image, String first, String last){
        ImageView imageView = createImage(image);
        TextField txt = createTextField(first, last);
        txt.setStyle("-fx-background-color: transparent; -fx-text-fill: white;");
        HBox hBox = new HBox(2, imageView, txt);
        hBox.setPadding(new Insets(5));
        hBox.setAlignment(Pos.CENTER_LEFT);
        hBox.setStyle("-fx-background-color: linear-gradient(to bottom right,
#be823a, #231025);");
        return hBox;
    }

    private Hyperlink createHyperlinkBlock(String link){
        Hyperlink hyperlink = new Hyperlink(link);
        hyperlink.setFont(Font.font("Cambria", FontWeight.NORMAL, 16));
        hyperlink.setStyle("-fx-text-fill: white; -fx-background-color: linear-
gradient(to bottom right, #be823a, #231025);");
        hyperlink.setOnAction(e -> {
            try {
                Desktop.getDesktop().browse(new URI(hyperlink.getText()));
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });
        return hyperlink;
    }

    private HBox createDateBlock(Timestamp timestamp){
        Label label = new Label(transformDataToLocal(timestamp));
        label.setFont(Font.font("Cambria", FontWeight.BOLD, 16));
        label.setStyle("-fx-text-fill: white;");
        label.setPrefWidth(300);
```

```
        ImageView dateImg = new ImageView(new
Image(getClass().getResource("date.png").toExternalForm()));
        dateImg.setFitHeight(40);
        dateImg.setFitWidth(40);
        HBox hBox = new HBox(4, dateImg, label);
        hBox.setAlignment(Pos.CENTER_LEFT);
        hBox.setStyle("-fx-background-color: linear-gradient(to bottom right,
#be823a, #231025);");
        return hBox;
    }

    private HBox createScoreBlock(double resScore,int studentIdentify){
        TextField points = new TextField(resScore > -1 ? String.valueOf(resScore)
: "");
        points.setPrefSize(60, 15);
        points.setFont(Font.font("Cambria", FontWeight.BOLD, 18));
        points.setStyle("-fx-text-fill: black;");
        points.setOnKeyPressed(e -> handlePointsEnterKey(e, points,
studentIdentify));
        Label labelMaxScore = new Label("/" + maxScore);
        labelMaxScore.setFont(Font.font("Cambria", FontWeight.BOLD, 18));
        labelMaxScore.setStyle("-fx-text-fill: white;");
        ImageView scoreImg = new ImageView(new
Image(getClass().getResource("score.png").toExternalForm()));
        scoreImg.setFitHeight(45);
        scoreImg.setFitWidth(40);
        HBox hBox = new HBox(2, scoreImg, points, labelMaxScore);
        hBox.setAlignment(Pos.CENTER_LEFT);
        hBox.setStyle("-fx-background-color: linear-gradient(to bottom right,
#be823a, #231025);");
        return hBox;
    }

    private void handlePointsEnterKey(KeyEvent e, TextField points, int studentId)
{
        if (e.getCode() == KeyCode.ENTER) {
            if (!isFieldEmpty(points)) {
                double validScore = getValidScore(points.getText());
                if (validScore != -1) {
                    if (validScore <= maxScore) {
                        setPoints(points,studentId,validScore);
                    } else {
                        alertWindow("Error", "The score is higher than the
maximum!", Alert.AlertType.ERROR);
                    }
                }
            }
        }
    }

    private void setPoints(TextField points, int studentId,double validScore){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "update_student_score");
            params.put("assignment_id", String.valueOf(currentMaterial));
            params.put("student_id", String.valueOf(studentId));
```

```
            params.put("score", String.valueOf(validScore));
            params.put("token", "sparksnet253");
            callApi(params);
            points.getParent().requestFocus();
        }catch (Exception e){
            alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR);
        }
    }

    @FXML
    void onSendWorkClick(ActionEvent event) {
        ButtonType confirmation = confirmationDialog("Confirmation", "Do you want
to send your work?");
        if (confirmation == ButtonType.YES) {
            if (!isFieldEmpty(txtLink)) {
                workSendingCheck();
            } else {
                alertWindow("Error", Language.get("Message.emptyFields"),
Alert.AlertType.ERROR);
            }
        }
    }

    private void workSendingCheck(){
        try {
            new URL(txtLink.getText()).toURI();
            Timestamp nowDate = new Timestamp(System.currentTimeMillis());
            insertWork(nowDate);
            btnDeleteLink.setManaged(true);
            btnDeleteLink.setVisible(true);
            makeUrlWork();
        } catch (Exception e) {
            alertWindow("Error", "Incorrect link format!", Alert.AlertType.ERROR);
        }
    }

    private void insertWork(Timestamp nowDate){
        try {
            Map<String, String> params = new HashMap<>();
            params.put("action", "send_student_work");
            params.put("assignment_id", String.valueOf(currentMaterial));
            params.put("student_id", String.valueOf(MainPanel.id));
            params.put("link", txtLink.getText());
            params.put("date", nowDate.toString());
            params.put("token", "sparksnet253");
            callApi(params);
        }catch (Exception e){
            alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR);
        }
    }

    private void setTableProperties(){
        ObservableList<StudentRow> data = FXCollections.observableArrayList();
        getAssignments();
```

```
        Map<String, String> testMap = getTests();
        Map<String, String> allColumns = new LinkedHashMap<>(assignmentMap);
        allColumns.putAll(testMap);
        addStudentColumns(allColumns);
        Map<String, Double> allMaxScores = new HashMap<>(assignmentMaxScores);
        allMaxScores.putAll(testmaxScoresTable);
        setTableColumn(data,allMaxScores);
    }

    private void setTableColumn(ObservableList<StudentRow> data,Map<String,
Double> allMaxScores){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_students_with_scores");
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            if (response.getBoolean("success")) {
                setColumn(response.getJSONArray("rows"),data,allMaxScores);
            }
            Platform.runLater(()->tableViewResults.setItems(data));
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void setColumn(JSONArray studentsArray,ObservableList<StudentRow>
data,Map<String, Double> allMaxScores) throws SQLException {
        for (int i = 0; i < studentsArray.length(); i++) {
            JSONObject student = studentsArray.getJSONObject(i);
            int studentId = student.getInt("id");
            String name = student.getString("first_name") + " " +
student.getString("last_name");
            Map<String, Double> assignmentScores = getStudentScores(studentId);
            Map<String, Double> testScores = getBestTestScores(studentId);
            Map<String, Double> allScores = new HashMap<>(assignmentScores);
            allScores.putAll(testScores);
            data.add(new StudentRow(name, allScores, allMaxScores));
        }
    }

    private void getAssignments(){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_assignments_scores");
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            setAssignmentScores(response);
        }catch (Exception e){
            e.printStackTrace();
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void setAssignmentScores(JSONObject response){
```

```
        if (response.getBoolean("success")) {
            JSONArray array = response.getJSONArray("rows");
            for (int i = 0; i < array.length(); i++) {
                JSONObject item = array.getJSONObject(i);
                assignmentMap.put("A_"+item.getInt("id"), item.getString("name"));
                assignmentMaxScores.put("A_"+item.getInt("id"),
item.getDouble("score"));
            }
        }
    }

    private void addStudentColumns(Map<String, String> assignmentMap) {
        TableColumn<StudentRow, String> nameCol = new TableColumn<>("Student");
        nameCol.setCellValueFactory(cell -> new
SimpleStringProperty(cell.getValue().getStudentName()));
        Platform.runLater(()->tableViewResults.getColumns().add(nameCol));
        for (Map.Entry<String, String> entry : assignmentMap.entrySet()) {
            String assignmentId = entry.getKey();
            String title = entry.getValue();
            TableColumn<StudentRow, String> col = createScoreColumn(assignmentId,
title);
            Platform.runLater(()->tableViewResults.getColumns().add(col));
        }
        TableColumn<StudentRow, String> avgCol = new TableColumn<>("Average");
        avgCol.setCellValueFactory(cell -> new
SimpleStringProperty(cell.getValue().getAverageScoreString()));
        Platform.runLater(()->tableViewResults.getColumns().add(avgCol));
    }

    private TableColumn<StudentRow, String> createScoreColumn(String assignmentId,
String title) {
        TableColumn<StudentRow, String> col = new TableColumn<>(title);
        col.setCellValueFactory(cell -> new
SimpleStringProperty(cell.getValue().getScoreForAssignment(assignmentId)));
        return col;
    }

    private Map<String, Double> getStudentScores(int studentId){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_all_student_scores");
            params.put("student_id", String.valueOf(studentId));
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            Map<String, Double> scores = setAllStudentScores(response);
            return scores;
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
        return new HashMap<>();
    }

    private Map<String, Double> setAllStudentScores(JSONObject response){
        Map<String, Double> scores = new HashMap<>();
        if (response.getBoolean("success")) {
```

```
            JSONArray rows = response.getJSONArray("rows");
            for (int i = 0; i < rows.length(); i++) {
                JSONObject item = rows.getJSONObject(i);
                int assignmentId = item.getInt("assignment_id");
                double score = item.getDouble("score");
                scores.put("A_"+assignmentId, score);
            }
        }
        return scores;
    }

    private void studentResults(){
        getAssignments();
        Map<String, String> testMap = getTests();
        Map<String, String> allColumns = new LinkedHashMap<>(assignmentMap);
        allColumns.putAll(testMap);
        addStudentColumns(allColumns);
        int studentId = MainPanel.id;
        try{
            ObservableList<StudentRow> data =setStudentResultsInTable(studentId);
            Platform.runLater(() -> tableViewResults.setItems(data));
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private ObservableList<StudentRow> setStudentResultsInTable(int studentId)
throws IOException, SQLException {
        ObservableList<StudentRow> data = FXCollections.observableArrayList();
        Map<String, String> params = new HashMap<>();
        params.put("action", "get_single_student");
        params.put("student_id", String.valueOf(studentId));
        params.put("token", "sparksnet253");
        JSONObject response = new JSONObject(callApi(params));
        if (response.getBoolean("success")) {
            String name = response.getString("first_name") + " " +
response.getString("last_name");
            Map<String, Double> assignmentScores = getStudentScores(studentId);
            Map<String, Double> testScores = getBestTestScores(studentId);
            Map<String, Double> allScores = new HashMap<>(assignmentScores);
            allScores.putAll(testScores);
            Map<String, Double> allMaxScores = new HashMap<>(assignmentMaxScores);
            allMaxScores.putAll(testmaxScoresTable);
            data.add(new StudentRow(name, allScores, allMaxScores));
        }
        return data;
    }


    private Map<String, String> getTests(){
        Map<String, String> testMap = new LinkedHashMap<>();
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_all_tests");
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
```

```
            JSONObject response = new JSONObject(callApi(params));
            if (response.getBoolean("success")) {
                JSONArray array = response.getJSONArray("rows");
                for (int i = 0; i < array.length(); i++) {
                    JSONObject item = array.getJSONObject(i);
                    testMap.put("T_"+item.getInt("id"), item.getString("name"));

testmaxScoresTable.put("T_"+item.getInt("id"),item.getDouble("max_score"));
                }
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
        return testMap;
    }

    private Map<String, Double> getBestTestScores(int studentId){
        Map<String, Double> bestScores = new HashMap<>();
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_best_test_scores");
            params.put("student_id", String.valueOf(studentId));
            params.put("course_id", String.valueOf(idCourse));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            if (response.getBoolean("success")) {
                JSONArray array = response.getJSONArray("rows");
                for (int i = 0; i < array.length(); i++) {
                    JSONObject item = array.getJSONObject(i);
                    bestScores.put("T_"+item.getInt("test_id"),
item.getDouble("best_score"));
                }
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
        return bestScores;
    }

    @FXML
    void onEditAnnouncementClick(ActionEvent event) {
        showWindowRecommend(true);
    }


    @FXML
    void onCreateAnnouncement(ActionEvent event) {
        showWindowRecommend(false);
    }

    private void showWindowRecommend(boolean isChange){
        newWindow("Recommend","createRecommendation.fxml",null);
        CreateRecommendation.isChange=isChange;
        btnExit.getScene().getWindow().hide();
    }
```

```
    @FXML
    void onAnswersTestClick(ActionEvent event){
        turnOffAllPanels();
        pnlForAllWorks.setVisible(true);
        flowPaneAllWorks.getChildren().clear();
        buildPaneTestResultsStudent();
    }

    private void buildPaneTestResultsStudent(){
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            try {
                studentWorkInfo();
                Platform.runLater(()->stopBlocker(timelinePrBar,prBar,blockPane));
            }catch (Exception e){}
        }).start();
    }

    private void studentWorkInfo(){
        try{
            Map<String, String> params = new HashMap<>();
            params.put("action", "get_test_results");
            params.put("test_id", String.valueOf(currentMaterial));
            params.put("token", "sparksnet253");
            JSONObject response = new JSONObject(callApi(params));
            if (response.getBoolean("success")) {
                addingAllTestWorks(response.getJSONArray("rows"));
            }
        }catch (Exception e){
            connectionGettingDataError(timelinePrBar,prBar,blockPane);
        }
    }

    private void addingAllTestWorks(JSONArray array){
        for (int i = 0; i < array.length(); i++) {
            JSONObject item = array.getJSONObject(i);
            int id = item.getInt("id");
            int resId = item.getInt("resId");
            int score = item.getInt("score");
            int maxScore = item.getInt("max_score");
            byte[] image = Base64.getDecoder().decode(item.getString("image"));
            String first = item.getString("first_name");
            String last = item.getString("last_name");
            Platform.runLater(() -
>flowPaneAllWorks.getChildren().add(createPaneTestResultsStudent(id, resId, score,
maxScore, image, first, last)));
        }
    }

    private VBox createPaneTestResultsStudent(int id,int resId, int score, int
max_score, byte [] image, String first, String last){
        VBox vBox = createStyledVBox();
        Button button=getButtonReview(false,resId,id);
        button.setMaxHeight(20);
        button.setFont(Font.font("Cambria",FontWeight.BOLD,18));
```

```
        vBox.getChildren().addAll(createTopBlock(image,first,last),
getScoreTestReview(false,(score+"/"+max_score)), button);
        return vBox;
    }

    @FXML
    void onAccessTestClick(ActionEvent event) {
        modifyAccessToFile("accesstest","test_id",btnAccessTest);
    }


    @FXML
    void onTestAllowViewingClick(ActionEvent event) {
        new Thread(()->{
            try {
                Map<String, String> params = new HashMap<>();
                params.put("action", "toggle_test_viewing");
                params.put("test_id", String.valueOf(currentMaterial));
                params.put("token", "sparksnet253");
                callApi(params);
                setTextButtonViewing(true);
            } catch (Exception e) {
                Platform.runLater(()-
>alertWindow("Error",Language.get("DatabaseMessage.connect"),
Alert.AlertType.ERROR));
            }
        }).start();
    }

    @FXML
    void onTestDeleteClick(ActionEvent event) throws SQLException {
        stepsToDeleteElement("tests","t",pnlAllTests,pnlTestsBtn,vBoxTestContent);
    }

    @FXML
    void onTestEditClick(ActionEvent event) {
        NewTest.indexAct=1;
        NewTest.idTestEdit=currentMaterial;
        newWindow("Edit Test","newTest.fxml",null);
        btnExit.getScene().getWindow().hide();
    }

    @FXML
    void onStartTestClick(ActionEvent event) {
        ButtonType confirmation = confirmationDialog("Confirmation", "Do you want
to start this test?");
        if (confirmation == ButtonType.YES) {
            Testing.idTest=currentMaterial;
            newWindow("Test","testing.fxml",null);
            btnExit.getScene().getWindow().hide();
        }
    }

    @FXML
    void initialize(){
        initializeUserView();
```

```
        initializeUIComponents();
        initializeSearchListener();
        startBlocker(timelinePrBar,prBar,blockPane);
        new Thread(()->{
            getCodeCourse();
            Platform.runLater(()->showSelectedPanel());
        }).start();
    }

    private void initializeUserView() {
        if (MainPanel.category.equals("student")) {
            setStudentPanelView();
        }
    }

}


package com.example.sparksnet;
import javafx.application.Application;
import javafx.geometry.Rectangle2D;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.StackPane;
import javafx.stage.Screen;
import javafx.stage.Stage;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.stage.StageStyle;
import java.nio.file.Paths;

public class Main extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        String videoPath = Paths.get("components/Logo.mp4").toUri().toString();
        MediaView mediaView = getMediaView(stage, videoPath);
        StackPane panel=new StackPane(mediaView);
        Scene scene= new
Scene(panel,mediaView.getFitWidth(),mediaView.getFitHeight());
        stage.setScene(scene);
        stage.initStyle(StageStyle.UNDECORATED);
        stage.getIcons().add(new
Image(getClass().getResource("logo_icon.png").toExternalForm()));
        stage.show();
    }

    private MediaView getMediaView(Stage stage, String videoPath) {
        Media media = new Media(videoPath);
        MediaPlayer player= new MediaPlayer(media);
        player.setAutoPlay(true);
        player.setOnEndOfMedia(()->{
            stage.hide();
Language.setLanguage(Language.readChosenLanguage("components/main_language.txt"));
            Tools tools= new Tools();
            tools.newWindow("SparksNet","chooseCategory.fxml",null);
```

```
        });
        Rectangle2D screenSize= Screen.getPrimary().getBounds();
        MediaView mediaView= new MediaView(player);
        mediaView.setFitWidth((screenSize.getWidth()*0.5));
        mediaView.setFitHeight((screenSize.getHeight()*0.5));
        return mediaView;
    }

    public static void main(String[] args) {
        launch();
    }
}
```

Серверна частина (PHP):

```
<?php
header('Content-Type: application/json');
$host = "ravonite.mysql.tools";
$user = "ravonite_sn";
$password = "3kTUg75&c#";
$dbname = "ravonite_sn";

$conn = new mysqli($host, $user, $password, $dbname);
if ($conn->connect_error) {
    echo json_encode(['success' => false, 'message' => 'Database connection
failed']);
    exit;
}

function sendQueryResponse(mysqli_stmt $stmt) {
    $stmt->execute();
    $result = $stmt->get_result();
    $rows = [];
    while ($row = $result->fetch_assoc()) {
        if (isset($row['image'])) {
            $row['image'] = base64_encode($row['image']);
        }
        $rows[] = $row;
    }
    echo json_encode([
        'success' => true,
        'rowCount' => count($rows),
        'rows' => $rows
    ]);
}

$token = $_POST['token'] ?? '';
if ($token !== 'sparksnet253') {
    exit;
}

$action = $_POST['action'] ?? '';
switch ($action) {
    case 'login_user':
        $login = $_POST['login'] ?? '';
        $category = $_POST['category'] ?? '';
```

```
        $query = "SELECT password, id, image FROM `$category` WHERE login = ?";
        $stmt = $conn->prepare($query);
        $stmt->bind_param("s", $login);
        sendQueryResponse($stmt);
        break;
    case 'check_login':
        $category = $_POST['category'] ?? '';
        $login = $_POST['login'] ?? '';
        $query = "SELECT login FROM `$category` WHERE login = ?";
        $stmt = $conn->prepare($query);
        $stmt->bind_param("s", $login);
        sendQueryResponse($stmt);
        break;
    case 'update_password':
        $category = $_POST['category'] ?? '';
        $login = $_POST['login'] ?? '';
        $password = $_POST['password'] ?? '';
        $query = "UPDATE `$category` SET password = ? WHERE login = ?";
        $stmt = $conn->prepare($query);
        $stmt->bind_param("ss", $password, $login);
        $success = $stmt->execute();
        echo json_encode(['success' => $success]);
        break;
    case 'check_user_exists':
        $login = $_POST['login'] ?? '';
        $category = $_POST['category'] ?? '';
        $query = "SELECT id FROM `$category` WHERE login = ?";
        $stmt = $conn->prepare($query);
        $stmt->bind_param("s", $login);
        sendQueryResponse($stmt);
        break;
```