МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Інтерактивний додаток для навчання студентів діагностиці дитячих інфекційних хвороб»

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

Виконавець роботи: студент групи ІТ-62 Большунов Денис Михайлович

Кваліфікаційна робота бакалавра захищена на засіданні ЕК з оцінкою _____ «__» _____ 2020 р.

Науковий керівник _____ к.т.н., доц., Баранова І.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

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

Студент _____
(підпис)

Суми-2020

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

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
« » 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Большунов Денис Михайлович

1 Тема роботи *Інтерактивний додаток для навчання студентів діагностиці дитячих інфекційних хвороб*

керівник роботи ***Баранова Ірина Володимирівна, к.т.н., доцент*** **,** затверджені наказом по університету від « 14 » травня 2020 р. № 0576-III

2 Строк подання студентом роботи «1» червня 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку інтерактивного додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування інформаційної системи, практична реалізація додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, мета та задачі, аналіз аналогів, функціональні вимоги, засоби реалізації, структурно-функціональне моделювання роботи додатку, проектування бази даних, схема додатку, практична реалізація, висновки (всього 22 слайди презентації)

6 Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____01.10.2019_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	15.10.19 – 31.10.19	
2	Огляд аналогів та засобів реалізації	31.10.19 – 14.11.19	
3	Проектування додатку	14.11.19 – 30.11.19	
4	Створення бази даних	30.11.19 – 15.01.20	
5	Розробка дизайну інтерфейсу	15.01.20 – 08.02.20	
6	Написання логіки додатку (back end)	08.02.20 – 25.05.20	
7	Тестування	25.05.20 – 27.05.20	
8	Виправлення помилок та формлення документації	27.05.20 – 01.06.20	

Студент _____ Большунов Д.М.

(підпис)

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

(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Інтерактивний додаток для навчання студентів діагностиці дитячих інфекційних хвороб».

Кваліфікаційну роботу бакалавра присвячено розробці тренажера, що допоможе студентам-медикам закріпити знання та навички в області педіатрії.

В першому розділі проведено аналіз предметної області та програм-аналогів, сформульовано постановку задачі, а також вибір засобів реалізації.

В другому розділі описано структурно-функціональне моделювання роботи додатку, виконано моделювання варіантів використання, діаграми діяльності, проектування бази даних та діаграми класів.

В третьому розділі описано практичну реалізацію додатку. Наведено опис прототипу додатку, переходи між вікнами, програмна реалізація та опис використання програмного додатку.

Результатом проведеної роботи є розроблений інтерактивний додаток під управлінням операційної системи Windows для студентів-медиків.

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 23 найменувань та 3 додатків. Загальний обсяг роботи – 95 сторінок, у тому числі 56 сторінок основного тексту, 3 сторінки списку використаних джерел, 29 рисунків та 13 таблиць.

Ключові слова: ІНТЕРАКТИВНИЙ ДОДАТОК, ТРЕНАЖЕР, НАВЧАННЯ, ПЕДІАТРІЯ, ДІАГНОСТИКА, JAVA, ВІДЕО-ПИТАННЯ, БАЗА ДАНИХ, РЕЗУЛЬТАТ.

ЗМІСТ

ВСТУП

Ні для кого не секрет, що в медицині вже давно користуються цифровими технологіями. Сучасний світ неможливо уявити без інформаційних технологій, й медицина в цьому напрямку не є виключенням. Вона недарма є одним із лідерів по впровадженню комп'ютерних технологій. Проведення складних операцій, запис до лікаря на прийом, різні електронні сервіси, проведення онлайн конференцій, онлайн-джерела інформації, інтерактивні додатки й т.д., – це звичайно маленька частина того, на що здатні інформаційні технології в медицині.

На сьогодні існує велика кількість інтерактивних тест-додатків. Основне їх призначення – перевірка знань. В масі своїй додатки такого типу підтримуються на різних операційних системах, що дозволяє проходити тести в різних місцях. Але, як показує практика, в основному такі додатки вимагають підключення до інтернету, що звичайно інколи не зручно.

Тому актуальною задачею є розробка інтерактивного додатку, котрий спрямований на студентів медичних закладів, але обмежень на використання звичайним користувачам немає. Всі бажаючі можуть користуватися додатком, – виключення є те, що не кожен зможе зрозуміти інформацію, яка міститься в ньому.

Не менш важливим є середовище запуску, тобто під якою операційною системою буде працювати додаток. Незважаючи на те, що розробка додатку велась на кросплатформній мові програмування, все ж таки для роботи визначено систему Windows, як найбільш поширену ОС.

Отже, метою роботи є розробка додатку для навчання діагностиці дитячих інфекційних хвороб й перевірки знань студентів в цій області.

Для досягнення мети був визначений перелік завдань:

- Проаналізувати подібні рішення та методи інтерактивного навчання;
- Визначити програмні засоби для розробки додатку;

- Спроєктувати макет додатку на основі отриманих даних та вказівок від замовника;
- Створити базу даних для збереження відео матеріалу та запитань до відео;
- Розробити тестовий додаток, створити всі необхідні методи (функції), класи, інтерфейси, для його повноцінної роботи;
- Протестувати додаток.

При виникненні запитань з приводу використання додатком, користувачам буде надано інструкцію, в якій буде все детально описано.

Практичне значення проекту полягає в підвищенні якості навчання та перевірки знань у студентів-медиків.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд останніх досліджень і публікацій

У медичній практиці існує ланцюг захворювань, які лікарі називають, дитячими. Дитячі інфекційні захворювання виділені в окрему групу, тому що даними захворюваннями хворіють лише діти в ранньому та дошкільному віці. Всі дитячі хвороби є заразними, внаслідок чого всі діти, які спілкувалися або контактували із хворою дитиною, так само захворіють. Єдиним плюсом є, то, що після перенесених інфекційних захворювань, у дитини виробляється міцний імунітет, який зміцнюється та допомагає протягом усього життя. Існує велика кількість дитячих інфекційних хвороб, до них можна віднести такі захворювання: кір, краснуха, скарлатина, коклюш і так далі [1].

Через це, відповідно до даних Всесвітньої організації охорони здоров’я, інфекційні хвороби продовжують займати друге місце в світі, як провідна причина смерті хворих. Крім цього, близько мільйону смертних випадків в світі визначено вже перенесеними інфекційними захворюваннями [2].

Організм дитини, на відміну від дорослого чоловіка або жінки, не має достатньо сильної імунної системи. Діти найчастіше заражаються різними інфекціями, у них частіше з’являється захворювання, яке може піти з ускладненнями. Спробувати швидко вилікуватися та відвести розвитку інфекційних ускладнень у дитини – в першу чергу призначення лікаря [3].

Питання покарaщення діагностики, лікування та профілактики інфекційних захворювань у дітей постійно залишаються у центрі уваги науковців та практиків. Даному питанню присвячують числені науково-практичні конференції, зокрема такі, як “Інфекційні хвороби у дітей. Нинішній погляд на діагностику, лікування та профілактику” [4].

Використання сьогоднішніх методів контролю пояснюється необхідністю та потрібністю діючої структури освіти, оскільки класичний усний чи рукописний контроль знань учнів, який використовується більшістю закладів, не відповідає сучасним умовам навчального процесу. Труднощі технічного характеру, які виникають при впровадженні електронних тестів у навчальний процес, поєднуються з процесом формування тестових матеріалів в електронній формі та їх застосування [5].

У статті [6] висвітлено суть та основну властивість тестового контролю знань учнів заради навчального процесу, оцінено можливості використання тестових технологій у новій навчальній моделі майбутнього лікаря. Метою даної роботи є оцінка ступеня організації тестового контролю студентів з клінічної імунології та алергології на кафедрі внутрішньої медицини, клінічної імунології та алергології. Визначені ключові типи тестів зіставної стадії навчального процесу, на якій формулюються реалізація та ключові умови створення тестових завдань.

Сучасний медичний фахівець змушений володіти не тільки теоретичними знаннями, а й точними перевіреними навичками, щоб швидко робити правильні висновки в умовах стресу та відстуності достаньої кількості часу [7].

Одним із варіантів одержання управлінської педагогічної інформації є результати тестів. У порівнянні з класичними методами контролю (наприклад, письмовими іспитами), тести нерідко виявляються здебільше дійсним та корисним способом контролю.

Нині створено велику кількість програмних продуктів для підготовки та проведення тестування. Втім, всупереч зручності та простоті комп’ютерного тестування, даний метод через фізіологічні випадки не підходить для велико-габаритних тестів. Зокрема, медичний ліцензійний іспит Крок працює у формі порожнього тесту з прийдешнім комп'ютерним опрацюванням результатів [8].

Інформативна децентралізація освіти, що виконується в рамках свіжого проекту “суспільства знань”, певною мірою слугує укладанню чинних проблем по забезпеченню соціальної рівності, а саме створення однакових можливостей задля

отримання лікарської підмоги та медичної освіти незалежно від розташування, а також стану здоров'я і соціального рівня. Таким чином, є реальний шанс отримати освіту або телемедичну послугу в медичних центрах, навчальних закладах, дослідницьких центрах всякого міста України або іншої держави, не виїжджаючи за місцем проживання [9].

З огляду зазначених та інших досліджень можна зробити висновок, що питання проведення тестувань в навчальному процесі медиків є актуальною та затребуваною задачею.

1.2. Аналіз програмних продуктів – аналогів

На сьогодні існують десятки додатків для медичних наук у формі тесту. Для порівняння з іншими проектами, які мають зв’язок із тест-додатками, візьмемо три із них.

Серед більшості можна винести наступні:

1. Медичний тест– iPhone додаток, призначений для студентів медичних закладів, який допомагає перевіряти та закріпляти знання з медичних дисциплін. Мова – російська, завантаження безкоштовне [10].

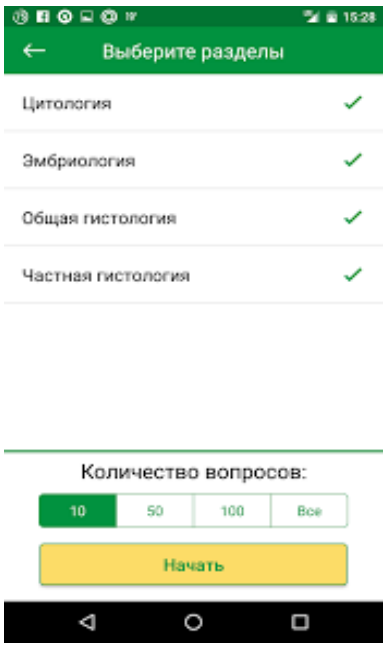

Рисунок 1.1 – Медичний тест

2. Тест для фармакології – Android додаток. Використання даного додатку буде корисним для студентів, які захотіли перевірити свої знання по дозуванню, групам, формам, призначенням і діючих речовин ліків. Мова – російська, завантаження безкоштовне [11].

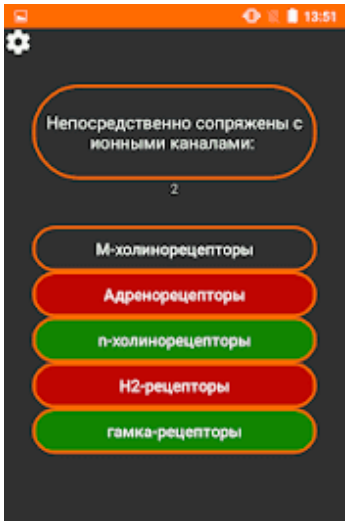

Рисунок 1.2 – Тест для фармакології

3. «Загальна патологія інфекційних хвороб» – онлайн тест охоплює вихідні лекції щодо заразних хвороб до особистої інфектології. Мова – російська, проходження тесту – безкоштовно [12].

Рисунок 1.3 – Онлайн тест «Загальна патологія інфекційних хвороб»

Також було розглянуто тренажер [13], призначений для діагностики дитячої віспи. Даний тренажер розроблений за допомогою програмного продукту AutoPlay Media Studio. До недоліків цього тренажеру слід віднести те, що використана програма розробки не є безкоштовною, має обмежений набір інструментів редагування та модифікації, тому зміна питань тесту чи інтерфейсу додатку є проблемною.

В результаті аналізу розглянутих додатків до їх переваг можна віднести наступні: всі з перерахованих додатків є безкоштовними, більшість мають інтуїтивно зрозумілий інтерфейс, підтримку різних ОС, специфічну тематику. Основна мета їх – перевірка знань у студентів медичних дисциплін. Все зроблено для того, щоб студенти були зосереджені на проходженні тесту. При аналізі було виявлено, що більшість проєктів створенні для мобільних пристроїв та браузерів.

Тим не менш, слід зазначити їх недоліки: обидва додатки та онлайн тест мають підтримку виключно російської мови, недостатньо регулярні оновлення, не відповідають темі роботи, тому що не містять даних про дитячі інфекційні хвороби.

Інтерфейс вказаних додатків також не має підтримки додаткових мов, (крім онлайн тесту, який за необхідності можна перекласти за допомогою Google Translate), але звичайно це є недоліком, тому що додаткової мови не передбачено.

З аналізу наведених прикладів можна зробити висновок про необхідність розробки власного додатку, який не матиме зазначених недоліків.

1.3 Постановка задачі

1.3.1 Мета та задачі дослідження

Метою дипломної роботи є розробка інтерактивного додатка для навчання студентів діагностиці дитячих інфекційних хвороб. Даний проєкт допоможе студентам медичних закладів оновити та перевірити свої знання в області дитячої педіатрії.

Для досягнення мети було визначено наступні задачі:

- Провести аналіз існуючих додатків зі схожою тематикою;
- Визначити послідовність дій створення тест додатку, для якісної та надійної роботи без перебоїв та зависань, враховуючи оптимальні технології задля досягнення поставленої мети;
- Провести моделювання роботи додатку засобами UML;
- Розробити структуру додатку, визначити необхідні модулі;
- Розробити базу даних для зберігання візуальної інформації;
- Виконати практичну реалізацію додатку;
- Провести тестування роботи додатку.

Функціональні вимоги до інтерактивного додатку:

- *Зміна мови*. Українську мову буде виставлено за замовчуванням, проте іноземні студенти, які не розуміють української мови, матимуть змогу переключитися на англійську;
- *Проходження тесту.* Користувачі матимуть змогу пройти тестування для перевірки знань з даної теми;
- *Збереження результату в .pdf.* Щоб виключити можливість редагування

результату студентом після проходження тесту, користувачу буде надано повідомлення про збереження результату в .pdf файлі;

- *Збереження результату на пошту.* Після проходження тесту, користувачу буде надано повідомлення про збереження результату на пошту.

Також в додатку необхідно передбачити збереження (виведення) даних відео матеріалів та запитань із бази даних. Інтерфейс додатку повинен бути інтуїтивно зрозумілим навіть тим користувачам, які не мають ніякого відношення до медичних наук.

Забезпечити легке встановлення додатку на настільний комп'ютер з операційною системою Windows 8 і вище, для чого передбачити створення майстра.

Результатом проходження тесту є виведення та збереження результату тестування в .pdf файл без подальшого редагування, відправлення результату на пошту викладача.

Даний проєкт буде слугувати для можливого подальшого створення мобільного додатку та реалізації кросплатформенності.

Повністю вимоги до розробки інтерактивного додатка для навчання студентів діагностиці дитячих інфекційних хвороб наведені у технічному завданні (Додатку А).

Під час планування реалізації проекту були чітко поставлені цілі роботи, мета (у тому числі за SMART-методом). Для наочного відображення ієрархії виконавців та виконуваних робіт у часі виконані структурні діаграми WBS та OBS, діаграми Ганта. Були ідентифіковані та проаналізовані ризики для запобігання непередбачуваних обставин, що можуть відобразитися на якості виконання роботи. Повністю етап планування робіт наведено у додатку Б.

1.3.2 Вибір засобів реалізації

Для того, щоб розробити додаток, було використано декілька програмних продуктів, а саме JetBrains Intellij IDEA, JavaFX Scene Builder та MySql Workbench.

JetBrains Intellij IDEA – це одне із передових середовищ розробки для

швидкого програмування на мові Java. IntelliJ IDEA – це високотехнологічна сукупність компактно інтегрованих інструментів програмування, яка включає в себе інтелектуальний редактор джерел із покращенними засобами автоматизації, потужний інструментарій для редагування коду, вбудовану підтримку таких технологій як J2EE, підтримка механізмів інтеграції із середовищем тестування Ant / JUnit та системами управління версіями Github, унікальний інструмент для покращення та перевірки на правильність написання коду за допомогою Code Inspection, а також новітній візуальний дизайнер графічних інтерфейсів.

Дані переваги JetBrains IntelliJ IDEA, які було перечислено вище, надають розробнику гнучкості, для вчасного скорочення похибки та підвищення якості програмного коду [14].

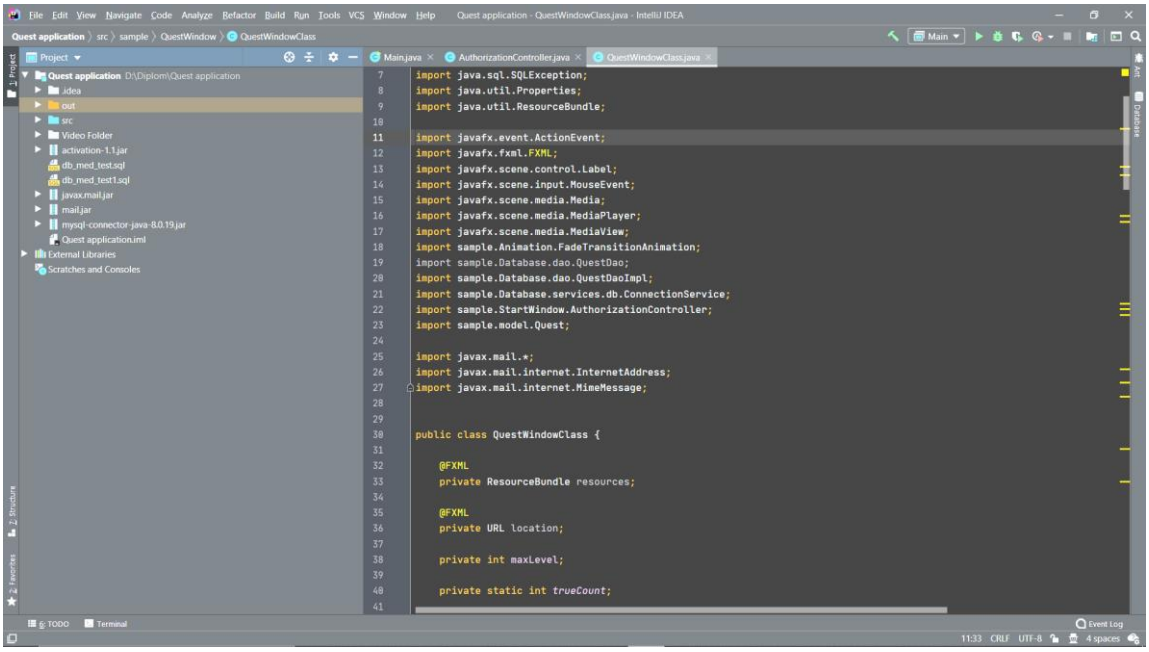

Рисунок 1.4 - Інтерфейс Intellij IDEA

Деякі можливості Intellij IDEA:

- Інструменти для якісного аналізування програмного коду, зручна та зрозуміла навігація, розумне автозаповнення, має широкий інструментарій переробки коду та форматування для Java, Groovy, Scala, HTML, CSS, JavaScript;
- Підтримка відомих платформ та фреймворків;
- Інтеграція з серверами прикладного програмного забезпечення;

- Можливість роботи із різними базами даних та файлами SQL;
- Підтримка інтеграції із системою (и) управління версіями (GitHub);

За допомогою Intellij IDEA, був створений код додатку, тобто вся функціональність, – це і спрацювання подій, наприклад, коли користувач натискає на якусь із кнопок, спрацьовує певна подія, ця подія може перенести користувача на наступне вікно додатку, ідентифікації, завершення тестування тощо. Також Intellij IDEA використовувався для написання запитів до бази даних.

JavaFX Scene Builder – це візуальний інструмент планування (компонування), який дозволяє розробникам програмного забезпечення швидко створювати користувальницькі інтерфейси додатків JavaFX без написання програмного коду [15-16].

Розробники можуть перетягувати компоненти робочого інтерфейсу на робочу область в додатку, змінювати їх властивості, застосовувати таблиці стилів (CSS), та FXML-код, який автоматично створюється у фоновому режимі, після чого його можна з легкістю скопіювати в основний проект, а також редагувати безпосередньо із додатку. Результатом є – файл FXML, який в подальшому можна комбінувати з проектом Java, пов'язуючи графічний інтерфейс з логікою програми [17].

Додатково для будь-якого компонента можна задати графічні ефекти типу відображення, тіні, розмиття та ін.

Також можна визначити CSS-стилі як для всіх компонентів так і для окремих [18]. Майже всі дії з ефектами, можна виконати за допомогою каскадних таблиць стилів, це набагато зручніше ніж користуватися вбудованим функціями в Scene Builder. Все, що потребується від розробника, – тільки підключити необхідний CSS-файл.

Набір компонентів включає як стандартні кнопки, гріди, комбобокси тощо, так і графічні примітиви для малювання, а також графіки декількох типів.

Відредагований макет зберігається в файл FXML і може бути використаний в програмах на JavaFX.

Вся графічна частина, яка присутня в додатку, була створена за допомогою

програми Scene Builder (рис. 1.5), але FXML код зберігається в Intellij IDEA.

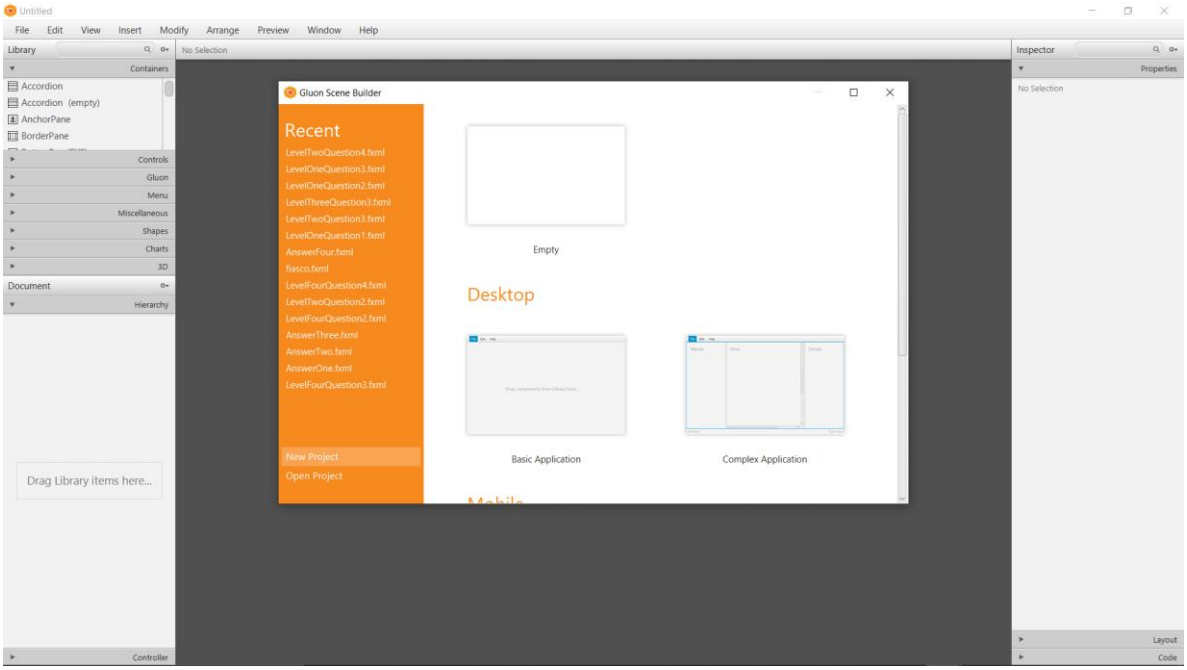

Рисунок 1.5 - Інтерфейс Scene Builder

MySQL Workbench – додаток для візуального створення баз даних, інтегрує проектування, моделювання, створення та використання баз даних у цілому вільному середовищі для бази даних MySQL [19].

Три основні можливості MySQL Workbench:

- Розробка SQL: Дозволяє створювати та регулювати з'єднаннями з серверами баз даних. Окрім того, що Workbench дозволяє налаштовувати параметри з'єднання, дане ПЗ надає можливість виконувати запити SQL на підключеннях до бази даних за допомогою інтегрованого редактора SQL;
- Моделювання даних: Дозволяє створювати графічні моделі (схеми) бази даних, та редагувати всі аспекти бази даних за допомогою вбудованого в додаток Редактора таблиць;
- Адміністрування сервера: Дозволяє створювати та керувати екземплярами сервера.

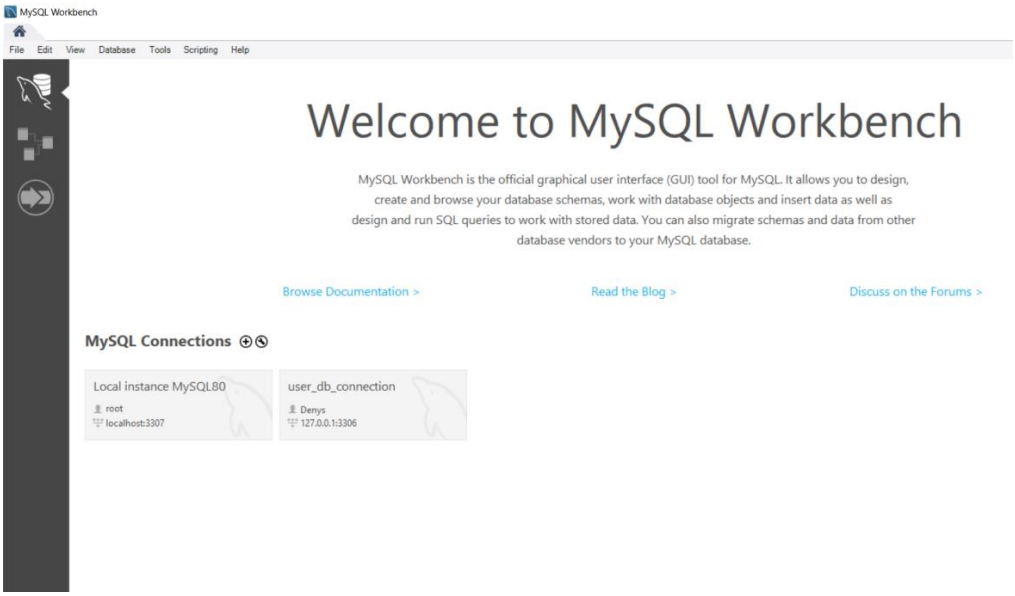

Рисунок 1.6 – Вікно MySQL Workbench

Зазначених програмних продуктів буде достатньо для практичної реалізації створюваного додатку.

2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Структурно-функціональне моделювання роботи додатку

Для побудови відображення функціонування додатку, було обрано методологію IDEF0. IDEF0 – це багатофункціональна методика прогнозування (функціональне моделювання) та графічне позначення, призначене для нормалізації та зображення бізнес-процесів. Характерним неординарним виключенням IDEF0 є його наголос на підпорядкованості об'єктів. IDEF0 вивчає регулярні зв’язки між робочими місцями, але не їх тимчасову чергу (робочий процес) [20].

Багатофункціональна модель IDEF0 являє собою послідовність блоків, будь-який з яких – це «чорна скринька» з входами та виходами, елементами управління та механізмами, які детально розкладаються (декомпозуються) до необхідно рівня декомпозиції. Найбільш важлива функція розташована у верхньому лівому куті. Функції пов’язані між собою за допомогою стрілок та описів багатофункціональних блоків. У цьому випадку будь-яка версія стрілки чи дії має своє значення. Наведена модель дає можливість зобразити всі основні варіанти процесу, як адміністративні, так і організаційні.

Основним компонентом моделі IDEF0 є функція або процес, яка показана на діаграмі як багатофункціональний блок – прямокутник, всередині якого позначена дія (процес). Дія може бути різною за масштабом – від діяльності компанії загалом і до конкретної маніпуляції зокрема.

Незалежно від масштабу операцій, всі функції відображаються однорідно і обов'язково містять 4 основні потоки, які твердо закріплені з боків багатофункціонального блоку:

- зліва – входи або засоби, що використовуються для виконання операцій;
- праворуч – виходи або очікуваний результат після виконання всіх

операй;

- зверху – керуючі стрілки несуть інформацію, які вказують, що повинен виконувати процес;
- знизу – механізми, які відображають, хто і за допомогою чого змушений здійснити дану роботу.

Контекстна діаграма (діаграма верхнього рівня), будучи вершиною структури дерева діаграм, демонструє призначення системи (основна функція) та її взаємодію із зовнішнім середовищем. У кожній модифікації може бути лише одна контекстна діаграма. Після охарактеризування фундаментальної функції робиться багатофункціональне розкладання, тобто визначаються функції, що утворюють основну діаграму.

Контекстну діаграму навчання студентів діагностиці дитячих інфекційних хвороб показано на рис. 2.1.

Рисунок 2.1 - Контекстна діаграма верхнього рівня – IDEF0

На вхід надходять матеріали (input data) – інформація, яка буде слугувати даними для отримання результату. Інформацією, яка поступає в процес, є Відео матеріал, Запитання до відео та Дані користувача.

В якості ресурсу для керування механізмом (mechanism), або механізми, які маніпулюють вхідними даними, виступають Розробник, Користувач, Програмне забезпечення та Апаратне забезпечення.

Дані управління (controll) необхідні для того, щоб процес розумів, як правильно керувати вхідними даними, тобто управління виступає в якості обмеження, в якості управління виступають наступні дані: Технічне завдання, Обмеження поштового сервісу й Методика діагностики хвороб.

Після виконання всіх описаних маніпуляцій, на виході (output data) представлено очікуваний результат, а саме Відправлення результату на пошту та збереження результату в .pdf файл.

Після того, як закінчили розробку контекстної діаграми, проведемо декомпозицію, описавши послідовність навчання студентів діагностиці дитячих інфекційних хвороб:

- Ідентифікація;
- Зміна мови;
- Проходження тесту;
- Підключення до бази даних;
- Результат.

Декомпозицію діаграми рівня IDEF0 наведено на рис. 2.2.

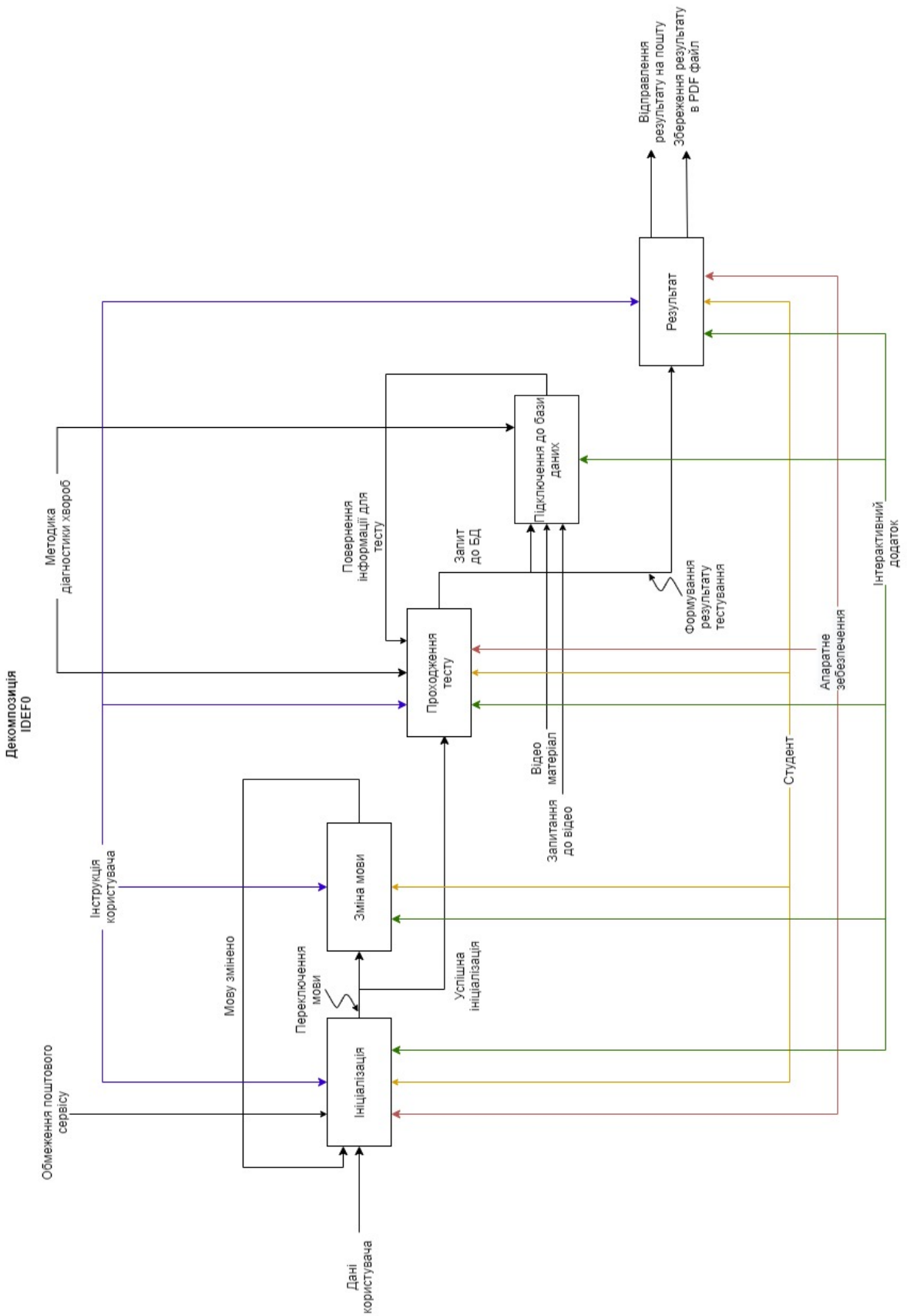

Рисунок 2.2 - Декомпозиція IDEF діаграми

2.2.Моделювання варіантів використання

По завершенню моделювання IDEF діаграми, тобто діаграми функціональних процесів, переходимо до розробки діаграми Варіантів використання. Діаграма ВВ входить в один із етапів моделювання системи засобами UML [21].

Під UML (Unified Modeling Language) розуміють уніфіковану мову об’єктно-орієнтованого моделювання. Основне призначення UML є підтримка життєвого циклу продукту та активності розробників на різних етапах проекту.

Щодо діаграми ВВ, основна її роль полягає в відображенні сценаріїв використання (Use Cases), на діаграмі вони позначаються овалом, та користувачів (Actors), на діаграмі позначаються в формі чоловічка, які користуються її функціями [22].

На (рис 2.3) зображено систему в цілому та зовнішні елементи. До зовнішніх елементів відносяться: Користувач – актор, який взаємодіє в певною сутністю в системі, База даних – елемент, призначенням якого є збереження та вивантаження даних, Прецедент – елемент, котрий показує дії, що виконуються із даною системою, Електронна пошта та Файл виконують збереження та відправлення результату відповідно.

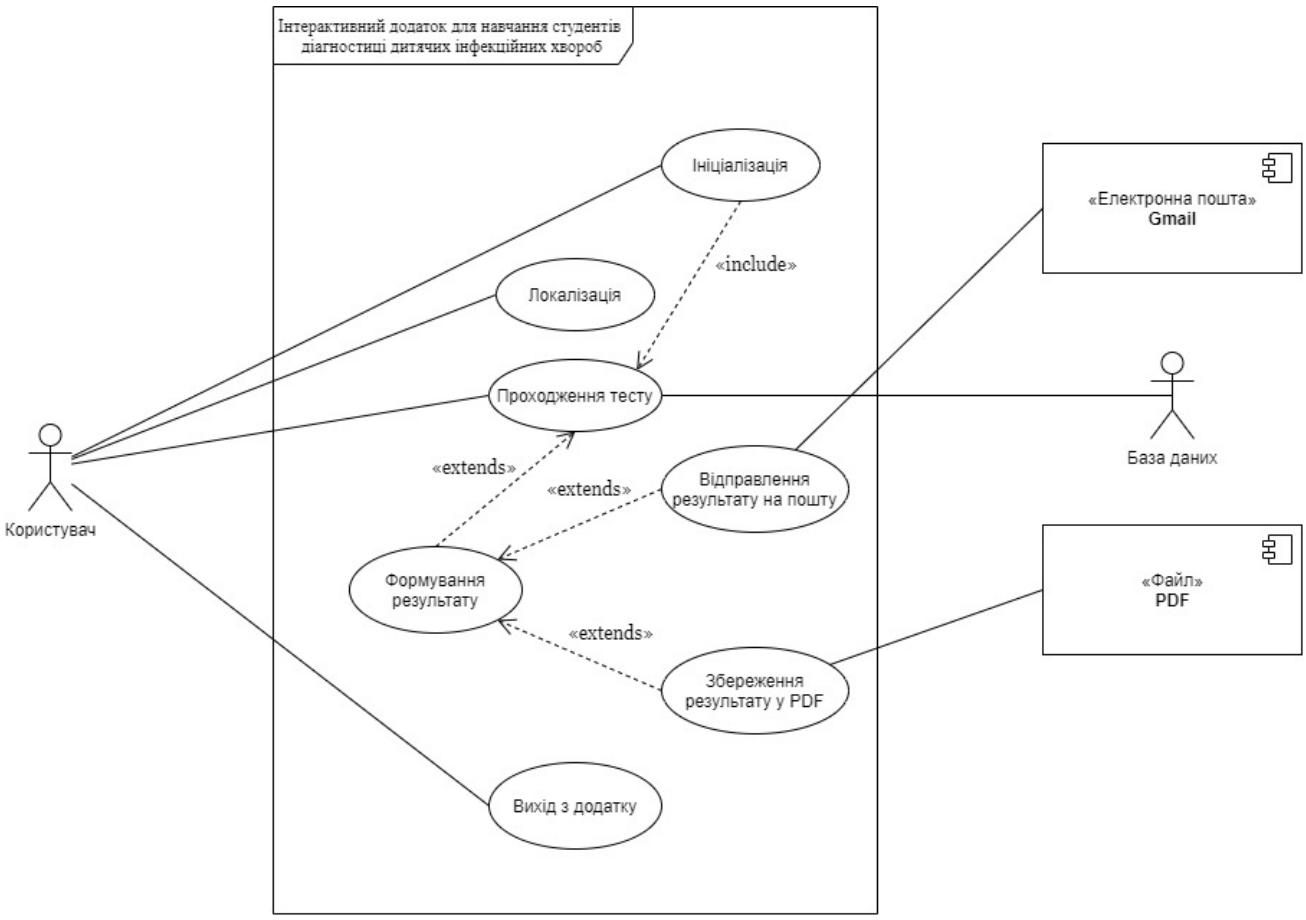

Рисунок 2.3 - Діаграма варіантів використання

Користувачу надаються наступні функції, а саме Ідентифікація, Локалізація, Проходження тесту, перегляд результату в файлі з розширенням .pdf та на пошті.

Далі детальніше описано кожен із Варіантів використання:

— Ідентифікація – кожному користувачу необхідно пройти Ідентифікацію для можливості подальшого проходження тесту;

— Локалізація – зміна мови всього додатку;

— Проходження тесту – розпочати тестування;

— Формування результату – виведення повідомлення про те, що користувач закінчив проходження тесту;

— Відправлення результату на пошту – збереження кінцевого результату, та відправлення за вказаним е-мейлом;

— Збереження результату в PDF – формування підсумку проходження тесту та збереження в файл .pdf без подальшого редагування;

— Вихід із додатку – завершення проходження тесту.

2.3.Моделювання діаграм діяльності

Діаграма діяльності є необхідною поведінковою діаграмою в діаграмі UML, для відображення динамічних етапів системи. Діаграма дій – це, насправді, більш розширений варіант блок-схеми, яка імітує перехід від одної дії до іншої.

Ми застосовуємо діаграми діяльності, щоб проілюструвати потік управління в системі та спиратися на кроки, з'єднані з виконанням варіантів використання. Показуємо моделювання почергових й паралельних дій, використовуючи діаграми діяльності. Діаграма діяльності концентрується на стані потоку і послідовності, в якій це відбувається.

На рисунку 2.4 зображено діаграму діяльності вікна ініціалізації, а на рисунку 2.5 зображено діаграму діяльності вікна привітання.

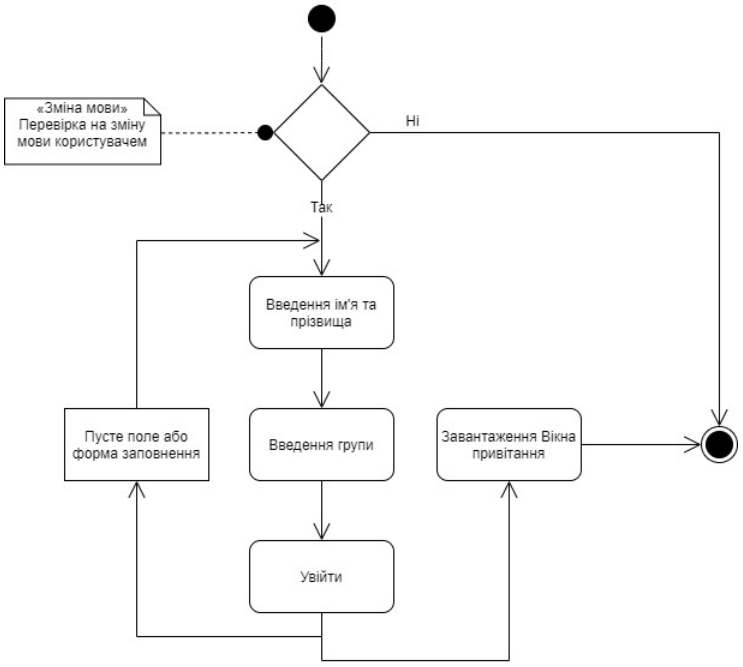

Рисунок 2.4 – Діаграма діяльності вікна Ініціалізації

Рисунок 2.5 – Діаграма діяльності вікна привітання

На рисунку 2.6 зображено діаграму діяльності вікна проходження тесту та вікна результату.

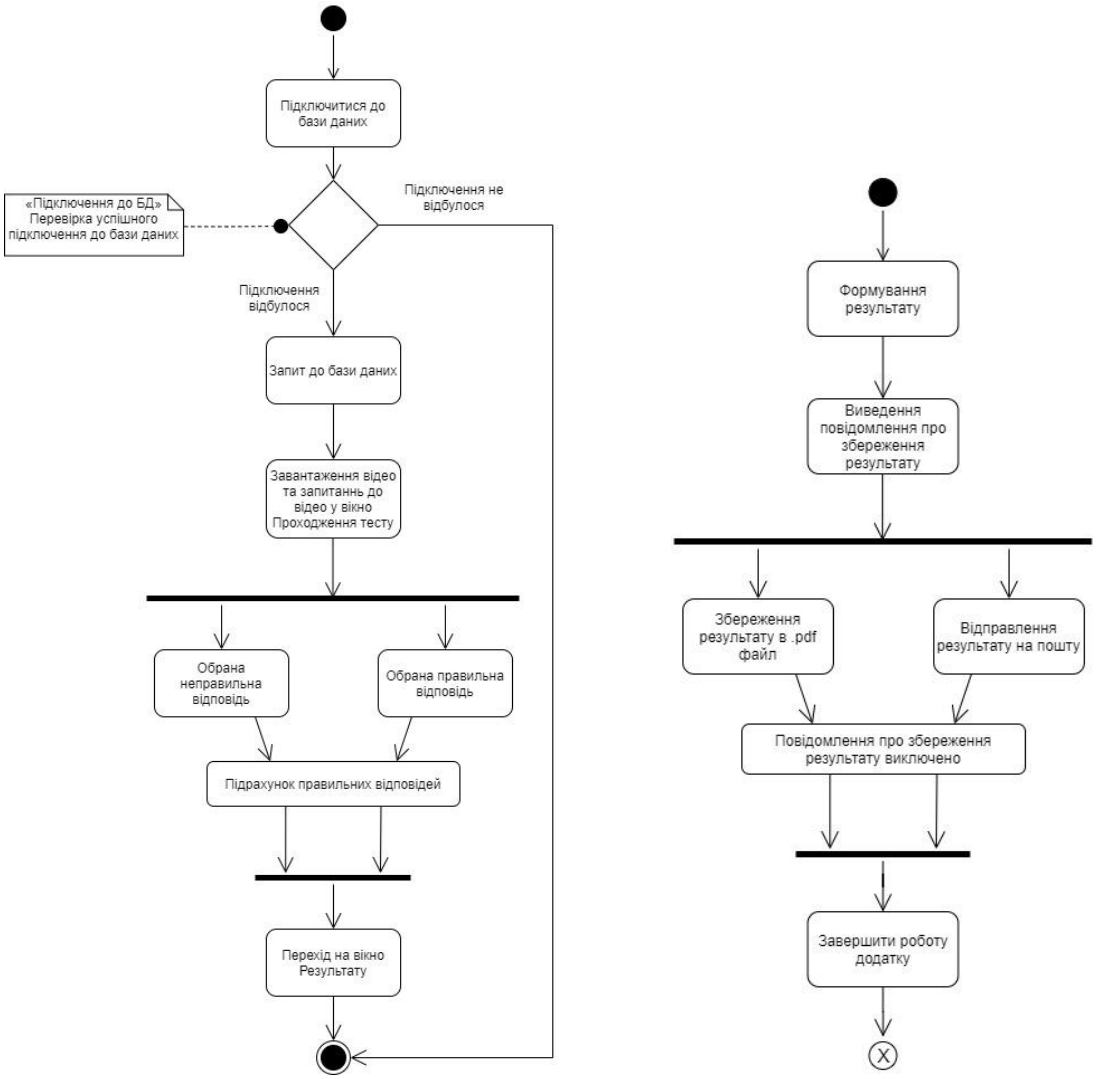

Рисунок 2.6 – Діаграма діяльності вікна проходження та результату тесту

2.4.Моделювання діаграми класів

Діаграма класів UML - це графічний запис, що застосовується для побудови й зображення об'єктноорієнтованих систем. Діаграма класів в UML представляє собою різновидність статичної структурної діаграми, яка відображає структуру системи, демонструючи її:

- класи;
- атрибути класів;
- методи класів;

- та відносини між об’єктами.

UML був створений як уніфікована модель для представлення підходу об'єктноорієнтованого програмування. Через те, що класи представляють собою будівельний блок для об'єктів, діаграми класів є будівельними блоками UML. Різноманітні компоненти в діаграмі класів можуть показати класи, які будуть практично запрограмовані, а також ключові об'єкти або взаємозв'язок між класами та об’єктами.

На рисунку 2.7 зображено діаграми класів інтерактивного додатку.

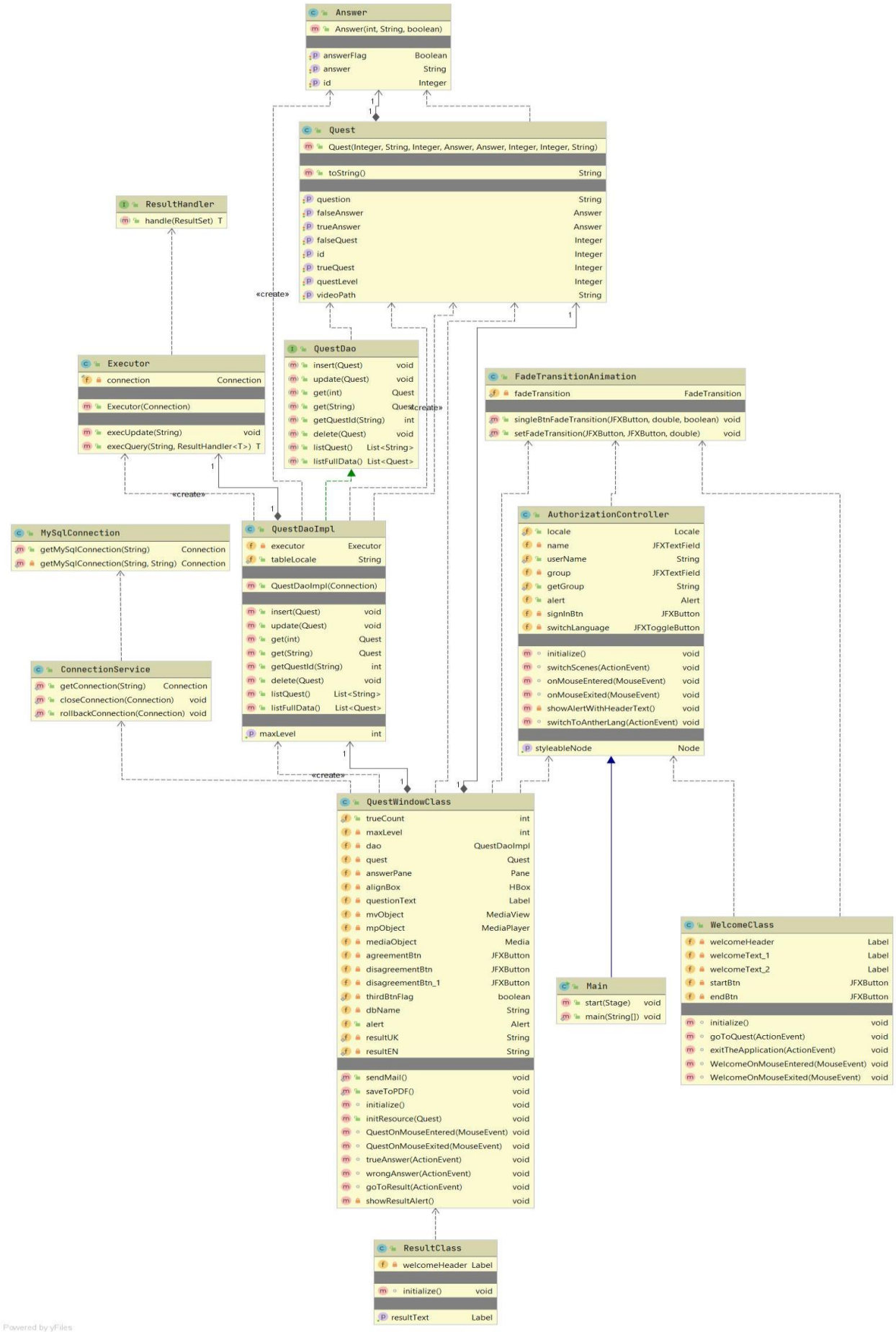

Рисунок 2.7 – Діаграма класів

2.5.Проектування бази даних

Перед початком програмування бази даних гарною практикою є створення графічного представлення бази даних, тобто таке представлення бази даних, яке зручно читати та вносити зміни без написання громіздких запитів. Для подібних цілей існує так звана ER діаграма.

На рисунку 2.8 приведено розроблену для інтерактивного додатку модель бази даних.

ER діаграма (модель) – це діаграма, призначена для проектування Бази даних. За допомогою ER діаграми можна побудувати будь-яку базу даних будь то реляційна, мережева, ієрархічна, об’єктна та багато інших. Фундаментом в ER-моделі є сутність, зв’язок та атрибут [23].

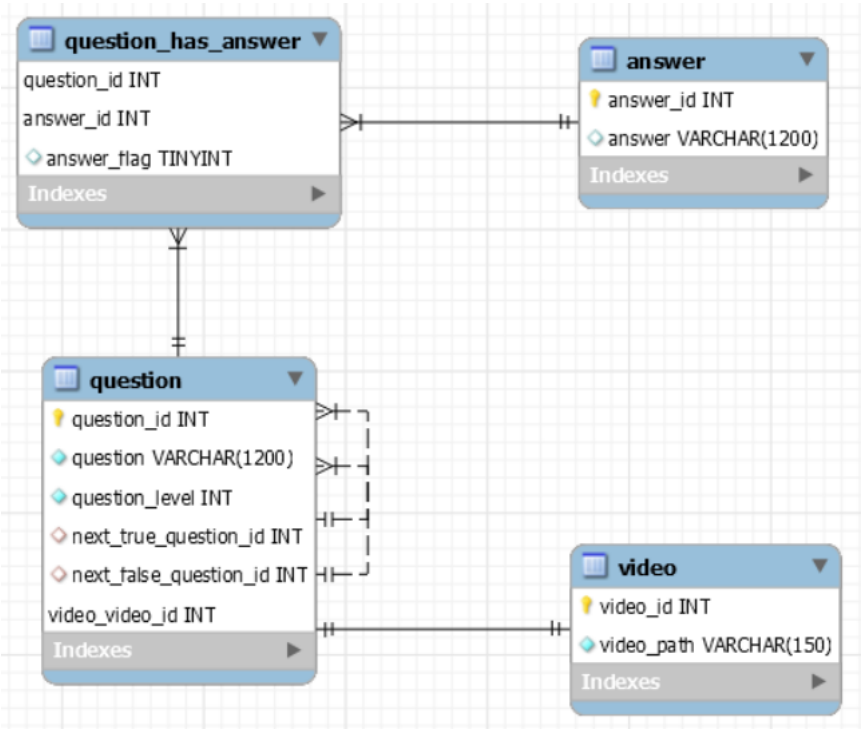

Рисунок 2.8 – ER діаграма

До сутностей відносяться наступні елементи:

— question – сутність для зберігання запитань до відео;

— video – сутність, в якій зберігається відео матеріал;

— answer – сутність для збереження ідентифікатора на наступну кнопку;

question_has_answer – сутність, яка зберігає поточний рівень ієрархічної таблиці із запитаннями та відео.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1. Складові етапи розробки програмного додатку

Для опису програмного додатку виділимо три основні складові етапи розробки. Перше – необхідно підібрати оптимальну мову програмування. Друге – скласти схему варіативності проходження тесту. Трете – визначити, або створити прототип перехід між вікнами додатку. Опишемо всі ці етапи детальніше.

Перед початком розробки додатку необхідно було підібрати відповідну мову програмування. Для аналізу було обрано декілька мов програмування, серед них: C++, C# та Java. Після ретельного аналізу, до аналізу входили наступні вимоги: парадигма ООП, підтримка роботи з графікою, підтримка роботи з мультимедіа даними, можливість роботи із Базою даних, підтримка відправлення даних на пошту, можливість збереження даних в .pdf файл. Тому було обрано мову програмування Java, а саме одну із бібліотек мови Java, яка має назву JavaFX. JavaFX – це платформа з відкритим кодом, для побудови кросплатформних графічних додатків.

Наступним кроком реалізовано схему варіативності проходження тесту, наведену на рисунку 3.1. На даній схемі зображено як працює додаток, та в якому напрямку йдуть переходи між варіантами відповідей.

Користувачу пропонується питання з відео-демонстрацією. Запитання на схемі позначено у вигляді чотирикутника. Коли користувач обирає один із варіантів відповідей (правильну позначено зеленою стрілкою, а неправильну – червоною), додаток відповідно схемі робить запит до бази даних до конкретного питання нижчого рівня. Після цього у вікні додатку відтворюється відповідні відеофрагмент та питання даного рівня. Аналогічно далі при наступній відповіді перехід іде до наступного відповідного питання нижчого рівня.

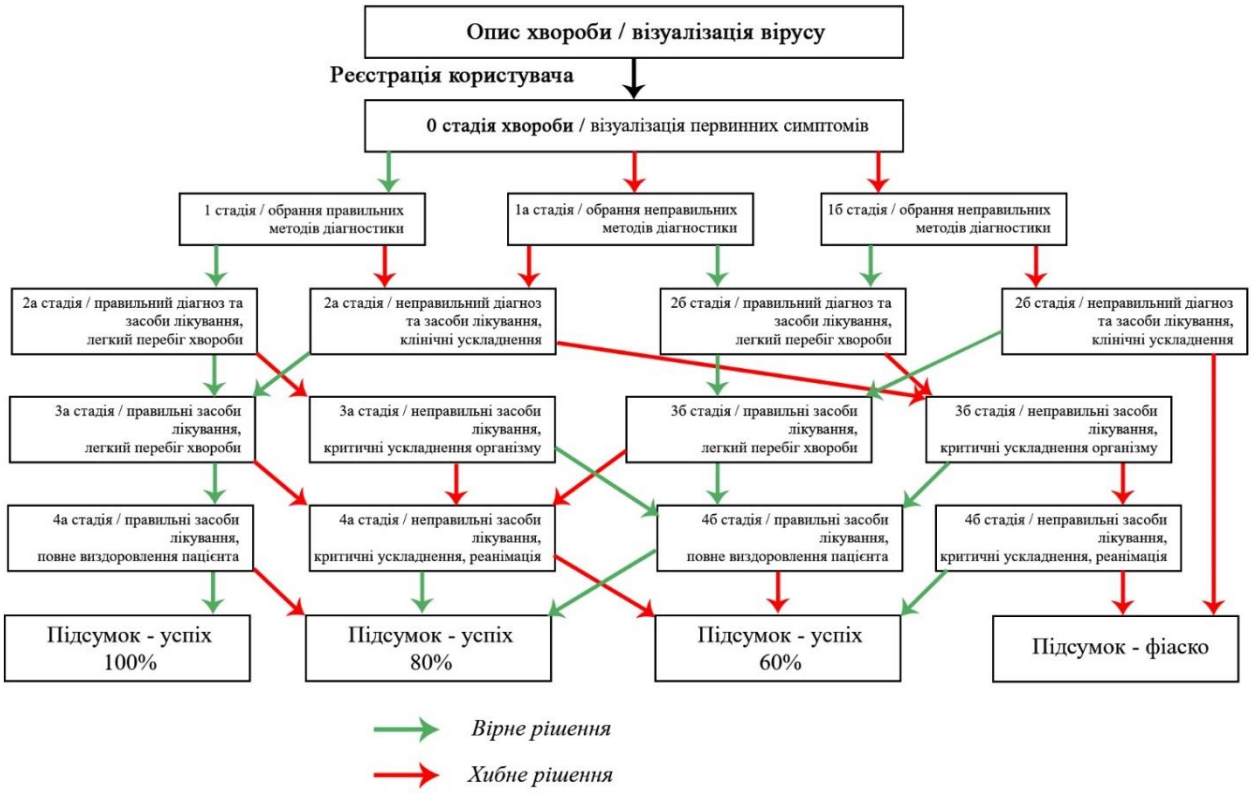

Рисунок 3.1 – Схема варіативності проходження тесту

На останньому етапі створення структури додатку було розроблено прототип майбутнього додатку, та реалізовано перехід між вікнами (рис. 3.2).

При запуску додатку, користувач потрапляє у вікно ініціалізації. У цьому вікні є два поля обов’язкові для заповнення. Якщо при натисненні на кнопку «Увійти» хоча б одне поле буде пусте, спрацює обробка помилки введення. Також у вікні ініціалізації знаходиться перемикач на англійський інтерфейс.

Після успішної ініціалізації користувача буде переадресовано на вікно привітання. В цьому вікні міститься інформація про додаток та мету роботи додатку. Якщо з якихось причин користувач не захоче проходити тест, він може вийти, для цього слугує однойменна кнопка.

Після натискання на кнопку «Почати» користувач потрапить у вікно проходження тесту. У цьому вікні буде відображено відео фрагмент, запитання до відео та варіанти відповідей, як описано вище.

По закінченню проходження тесту користувача буде переадресовано на останнє вікно додатку, а саме на вікно результату. В цьому вікні зазначено

кількість правильних відповідей та показано повідомлення про відправлення результату на пошту й збереження в .pdf файл.

Якщо користувач вибере іншу мову, відбудеться все теж саме, але англійською мовою.

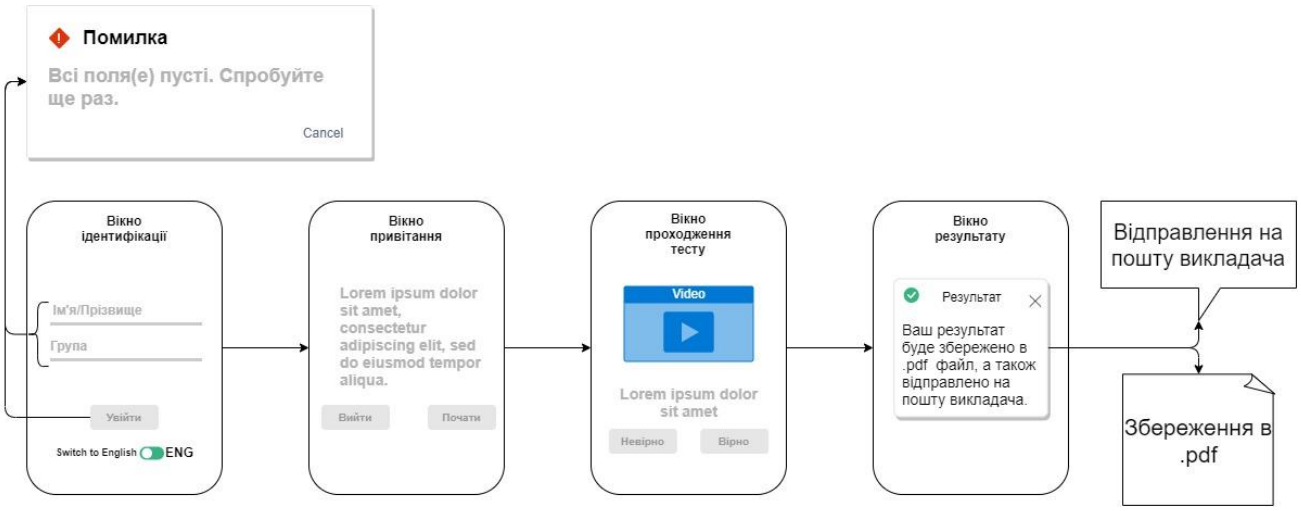

Рисунок 3.2 – Прототип майбутнього додатку

3.2.Програмна реалізація

На етапі програмної реалізації було написано всю логіку додатку. Тобто було розроблено повне функціонування додатку від вікна ідентифікації до формування .pdf файлу з результатом та відправленням на пошту.

В цьому підрозділі описано про найважливіші складові додатку а саме: пакети, класи, методи та сторонні файли, які використовувались при створенні додатку. Також описано функціонування бази даних з додатком.

Код програмногго продукту наведено в додатку В.

Конкретизуємо складові додатку, які було реалізовано за допомогою:

- Intellij IDEA – використовувався для написання коду, підключення сторонніх бібліотек та файлів, установлення зв’язку між іншим програмним забезпеченням, запуску додатку на перевірку працездатності.

- Scene Builder – використовувався для створення графічної складової додатку, тобто: створення графічного інтерфейсу, підключення всіх необхідних бібліотек, розміщення графічних компонентів для зручного використання, формування FXML коду з підтримкою подальшого редагування в середовищі розробки та відтворення виконаних змін в Scene Builder.
- MySql Workbench слугував для створення та редагування бази даних, проектування ER діаграми.

Перейдемо до описання розробки основних складових додатку.

Для організації файлів класів та інших даних, необхідних для роботи додатку, було створено пакети, зображені на рисунку 3.3. Пакет – це так званий організаційний каталог, в якому знаходяться необхідні файли.

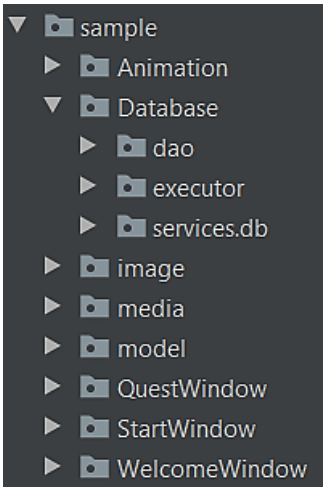

Рисунок 3.3 – Пакети із файлами даних

Опишемо більш детально файли, котрі знаходяться в цих пакетах. На рисунку 3.4 зображено приклад файлів, які знаходяться в пакетах ***sample*** та ***media***.

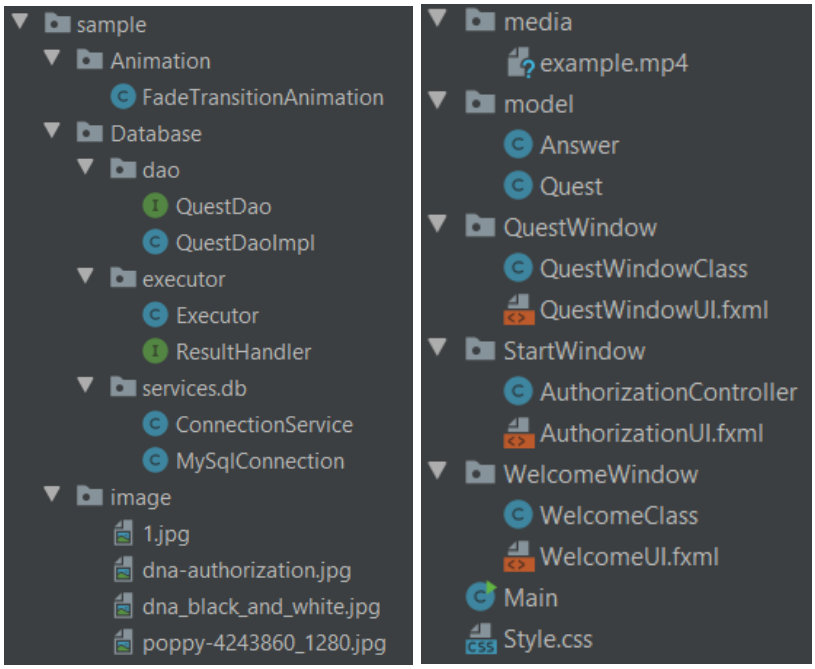

Рисунок 3.4 – Приклад розміщених файлів в пакетах

Пакет ***sample*** – це пакет в якому розташовані всі інші пакети. Так званий пакет верхнього рівня.

Пакет ***Animation*** – цей пакет містить класи, що використовуються для відтворення анімації.

Пакет ***Database*** – в цьому пакеті знаходяться класи та інтерфейси, які служать для правильної роботи додатку.

Пакети ***dao, executor, services.db*** – в цих пакетах знаходяться файли, які необхідні для чіткої роботи бази даних, а саме з’єднання бази даних з додатком та виконання запитів.

Пакет ***image*** – використовується для збереження та вивантаження картинок при зверненні до цього пакету.

Пакет ***media*** – в цьому пакеті знаходяться медіа файли, які використовуються для перевірки працездатності додатку, тобто в якості прикладу.

Пакет ***model*** – в даному пакеті знаходяться файли класів для роботи із базою даних.

Пакет ***QuestWindow*** – в цьому пакеті розташовані файли, які слугують для

чіткої роботи вікна Проходження тесту.

Пакет ***StartWindow*** - в цьому пакеті розташовані файли, які слугують для чіткої роботи вікна Ініціалізації.

Пакет ***WelcomeWindow*** - в цьому пакеті розташовані файли, які слугують для чіткої роботи вікна привітання.

У таблиці 3.1 наведено детальний опис файлів, що містяться у зазначених нижче пакетах.

Таблиця 3.1 – Опис файлів, які знаходяться в пакетах ***sample*** та ***Animation***

Клас(и), файл(и)	Метод(и)
Пакет ***sample***	
Main – головний клас додатку, з якого починається робота додатку.	***main()*** – саме до цього методу звертається додаток, коли необхідно розпочати роботу додатку. ***start()*** – метод main() викликає даний метод, щоб завантажити вікно Ініціалізації.
Style.css – файл із CSS стилями.	В цьому файлі містяться стилі для окремих класів чи графічних компонентів додатку, таких як кнопки, перемикачі, блоки і так далі.
Пакет ***Animation***	
FadeTransitionAnimation – клас для анімації, написані методи, які спрацьовують при наведенні курсора миші на кнопку в додатку.	***singleBtnFadeTransition()*** – метод, розрахований для однієї кнопки, тобто анімація буде працювати лише для певної кнопки.
	setFadeTransition() – метод, розрахований для декількох кнопок, тобто анімація буде працювати для декількох кнопок.

Класи, файли та методи пакету Database наведені у таблиці 3.2.

Таблиця 3.2 – Опис файлів, які знаходяться в пакеті ***Database***

Клас(и), файл(и)	Метод(и)
Пакет ***Database*** { ***dao*** ***executor*** ***services.db***	
QuestDao – в даному інтерфейсі, шаблоні написані методи, які використовуються для взаємодії із базою даних.	***get()*** – метод-шаблон.
QuestDaoImpl – клас, який містить методи для запиту до БД.	***get()*** – повертає об'єкт, який містить інформацію про: рівень проходження тесту, запитання до відео, відео фрагмент, правильну та неправильну відповідь, шлях до папки із відео фрагментами.
Executor – клас, який виконує запит до бази даних	***execQuery()*** – метод, котрий обробляє запит до бази даних та повертає результат.
ResultHandler – шаблон, який обробляє отриману відповідь від бази даних та переводить його в об’єкт.	Не містить методів.
ConnectionService – в цьому класі відбувається налаштування з’єднання з базою даних.	***getConnection()*** – метод, який повертає результат, після успішного налаштування.
MysqlConnection – даний клас необхідний для безпосереднього з’єднання з базою даних.	***getMySqlConnection()*** – в цьому методі написані команди, які слугують для підключення до БД. Перелік деяких команд: хост, порт, ім’я бази даних, пароль і т.д.

Класи, файли та методи пакету ***model*** наведені у таблиці 3.3.

Таблиця 3.3 – Опис файлів, які знаходяться в пакеті model

Клас(и), файл(и)	Метод(и)
Пакет ***model***	
Answer – клас описує параметри та властивості об'єкта відповіді, до яких відносяться: ідентифікатор відповіді, безпосереднього сама відповідь та флаг відповіді (категорія відповіді), правильна або неправильна.	***getAnswer()*** – метод, який повертає поточне запитання.
	getAnswer() – метод, який повертає поточне запитання.
Quest – клас відповідає за присвоєння та обробку даних, котрі надходять із бази даних.	***getQuestLevel*** – повертає рівень запитання та відео фрагмента. ***getQuestion*** – повертає запитання да відео. ***getVideoPath*** – повертає шлях до відео фрагменту, котрий розташований в папці на комп'ютері. ***getTrueAnswer()*** – повертає правильну відповідь. ***getFalseAnswer()*** – повертає неправильну відповідь. ***toString()*** – повертає присвоєні дані після натиснення кнопки у вікні Проходження тесту до бази даних, для порівняння із таблицею даних.

Таблиця 3.4 - Опис файлів, які знаходяться в пакеті QuestWindow

Клас(и), файл(и)	Метод(и)
Пакет ***QuestWindow***	
QuestWindowClass – клас відповідає за логіку вікна Проходження тесту.	***sendMail()*** – метод використовується для відправлення результату на пошту. ***saveToPDF()*** – метод використовується для збереження результату в .pdf файл. ***initialize()*** – метод необхідний для ініціалізації початкових значень об'єктів або змінних. ***initResource()*** – метод використовується для ініціалізації об'єкта відео, а також присвоєння кнопкам необхідних назв. ***QuestOnMouseEntered(), QuestOnMouseExited()*** – ці методи слугують для спрацювання анімації при наведенні на кнопки у вікні Проходження тесту.
	trueAnswer() – метод рахує кількість правильних відповідей та зберігає результат на пошту та в файл. Також необхідний, щоб повідомити базі даних, яку із кнопок було натиснуто. ***wrongAnswer()*** – даний метод виконує аналогічні операції, що і попередній метод, крім того, що не підраховує кількість правильних відповідей та дає зрозуміти базі даних, якщо користувач натиснув неправильну відповідь
QuestWindowUI – fxml файл відповідає за графічну частину вікна Проходження тесту.	Не містить методів.

У таблицях 3.5 і 3.6 наведені класи, файли та методи пакетів ***StartWindow*** і ***WelcomeWindow***.

Таблиця 3.5 - Опис файлів, які знаходяться в пакеті StartWindow

Клас(и), файл(и)	Метод(и)
AuthorizationController – клас, який відповідає за роботу вікна Ініціалізації.	***initialize()*** – ініціалізація початкових значень об'єктів або змінних, якщо такі знадобляться. ***switchScene()*** – метод виконує перехід на вікно Привітання. ***OnMouseEntered(), OnMouseExited()*** – ці два методи слугують для спрацювання анімації при наведенні на кнопки у вікні Ідентифікації. ***showAlertWithHeaderText()*** – метод для спрацювання помилки, якщо користувач залишить поля вводу пустими. ***switchToAnotherLang()*** – метод, котрий відповідає за перемикання мови.
AuthorizationUI – fxml файл, котрий відповідає за графічну частину вікна Ідентифікації.	Не містить методів.

Таблиця 3.6 - Опис файлів, які знаходяться в пакеті WelcomeWindow

Клас(и), файл(и)	Метод(и)
WelcomeClass - клас, який відповідає за роботу вікна Привітання.	initialize() – ініціалізація початкових значень об'єктів або змінних, якщо такі знадобляться. goToQuest() - метод для переходу на вікно Проходження тесту. exitTheApplication() – метод для завершення роботи додатку. WelcomeOnMouseEntered(), WelcomeOnMouseExited() – ці два методи слугують для спрацювання анімації при наведенні на кнопки у вікні Привітання.
WelcomeUI - fxml файл, котрий відповідає за графічну частину вікна Привітання.	Не містить методів.

Перейдемо до опису бази даних. Опишемо всі поля (стовпці) та їх призначення. На етапі створення бази даних було вирішено та розроблено дві бази даних, показаних на рисунку 3.5. Одна із них слугує для україномовного інтерфейсу, друга – для англомовного. Поля (стовпці) змінам не підлягали. Лише дані змінювались на ту мову, котра потрібна для даної бази даних, котра завантажується після перемикання мови на вікні Ідентифікації.

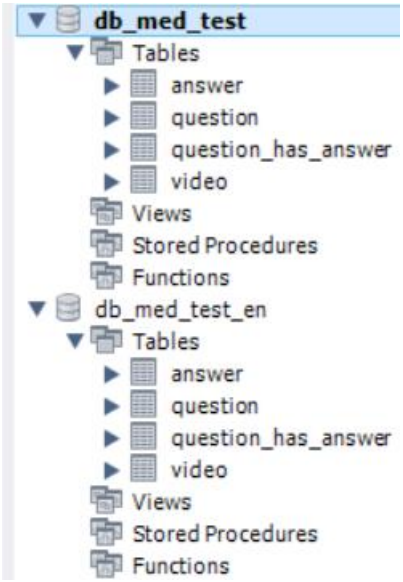

Рисунок 3.5 – Бази даних

Як можна спостерігати на малюнку, бази даних мають назви ***db_med_test*** та ***db_med_test_en*** відповідно. Таблиці мають назви: ***answer, question, question_has_answer*** та ***video***. Опишемо більш детально кожну із таблиць.

Таблиця 3.7 – Опис таблиці answer

Назва таблиці	Опис даних таблиці
answer – збереження даних кнопки про вибір правильної або неправильної відповіді.	***answer_id*** – ідентифікатор відповіді.
	answer – в цьому полі зберігається назва кнопки, котра показується після чергової відповіді користувачем в вікні Проходження тесту.

Таблиця 3.8 – Опис таблиці question

Назва таблиці	Опис даних таблиці
***question* –** збереження даних про запитання.	***question_id*** – ідентифікатор запитання.
	question – в даному полі зберігаються запитання до відео.
	video_id – ідентифікатор відео.
	question_level – це поле показую, на якому із рівнів знаходиться дане запитання.
	next_true_question_id – ідентифікатор на наступне правильне запитання.
	next_true_false_id – ідентифікатор на наступне неправильне запитання.

Таблиця 3.9 - Опис таблиці question_has_answer

Назва таблиці	Опис даних таблиці
question_has_answer – перевіряє, чи є у запитання продовження.	***question_id*** – ідентифікатор запитання.
	answer_id – ідентифікатор відповіді.
	answer_flag – логічна (0, 1) перевірка на наявність продовження запитання.

Таблиця 3.10 – Опис таблиці video

Назва таблиці	Опис даних таблиці
video – таблиця, в якій зберігаються шляхи до відео фрагментів.	***video_id*** – ідентифікатор відео.
	video_path – поле, яке містить в собі шляхи до відео на комп’ютері.

Останнім описом програмної реалізації будь файли, котрі були застосовані для розробки програмного додатку, на різних етапах програмування. В цілому це .jar файли, які представляють собою zip-файл. Основним його призначенням є збереження даних з класами. На рисунку 3.6 наведено саме .jar файли та папку із відео фрагментами.

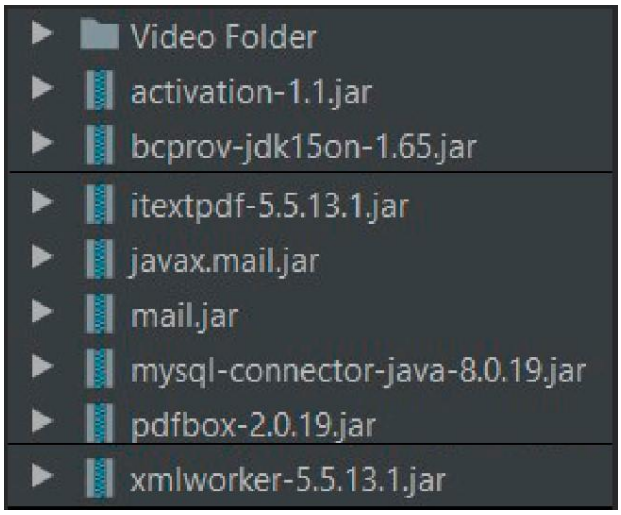

Рисунок 3.6 - Файли класів .jar

В папці ***Video Folder*** знаходяться відео фрагменти.

itextpdf-5.5.13.1.jar – в цьому файлі знаходяться класи, котрі були використані для написання коду на збереження результату в .pdf файл.

mail.jar, javax.mail.jar – в цих файлах знаходяться класи, котрі були використані для написання коду, щоб відправити результат на пошту.

mysql-connector-java-8.0.19.jar – файл, котрий використовується, для з'єднання MySql бази даних та додатку.

pdfbox-2.0.19.jar, bcprov-jdk15on-1.65.jar, xmlworker-5.5.13.1.jar, activation-1.1.jar – файли, котрі використовувались в якості тестування.

Далі у таблицях 3.11-3.13 навдений опис класів, що були використані із перелічених вище файлів.

Таблиця 3.11 – Опис класів, які були взяті з файлу itextpdf-5.5.13.1.jar

Клас	Опис класів (у)
itextpdf-5.5.13.1.jar	***Document*** – створення пустого екземпляра pdf файла. ***DocumentException*** – сповіщає, що в документі відбудась помилка. ***Font*** – охоплює всі властивості шрифту: сімейство шрифту, розмір, стиль і колір. ***Paragraph*** – створює екземпляр так званого абзаца або розділу текста в pdf файлі. ***BaseFont*** – клас, для підтримки декількох шрифтів в файлі. ***PdfWriter*** – даний клас служить, для можливості запису даних в pdf файл.

Таблиця 3.12 – Опис класів, які були взяті з файлів mail.jar, javax.mail.jar

Клас	Опис класів (у)
javax.mail.jar	***InternetAddress*** – даний клас відображує адресу електронної пошти в Інтернеті, застосовуючи синтаксис RFC822.
mail.jar	***MimeMessage*** – в цьому класі відбувається формування електронного листа стилю MIME (багатоцільові розширення інтернет-пошти).

Таблиця 3.13 - Опис класів, які були взяті з файлів mysql-connector-java-8.0.19.jar

Клас	Опис класів (у)
mysql-connector-java-8.0.19.jar	***Connection*** – цей клас створює об’єкт, для з’єднання з певною базою даних. Виконуються оператори Sql, а відповіді повертаються в контексті з'єднання. ***SQLException*** – даний клас надає інформацію якщо трапилась помилка при доступі до бази даних. ***ResultSet*** - клас ResultSet відтворює результуючий набір даних, тобто даних, котрі були отримані при запиті до бази даних. ***Statement*** – створює об’єкт, для виконання запиту до бази даних. ***Driver*** – реєструє jdbc драйвер в пам’яті комп’ютера для подальшого з’єднання з базою даних. ***DriverManager*** – обирає необхідний драйвер для певної бази даних, в даному випадку для MySql.

3.3.Використання програмного додатку

В цьому підрозділі описано працездатність додатку – показано крок за кроком, як працює додаток та зберігається результат.

Після запуску тренажеру користувач порапляє у вікно ініціалізації, показане на рисунку 3.7. В цьому вікні користувач вводить свої ім’я, прізвище та групу. При необхідності, користувач може змінити мову. Кнопка «Увійти» слугує для переходу на вікно привітання при умові, якщо всі поля заповненні. В іншому випадку користувач отримає повідомлення про помилку.

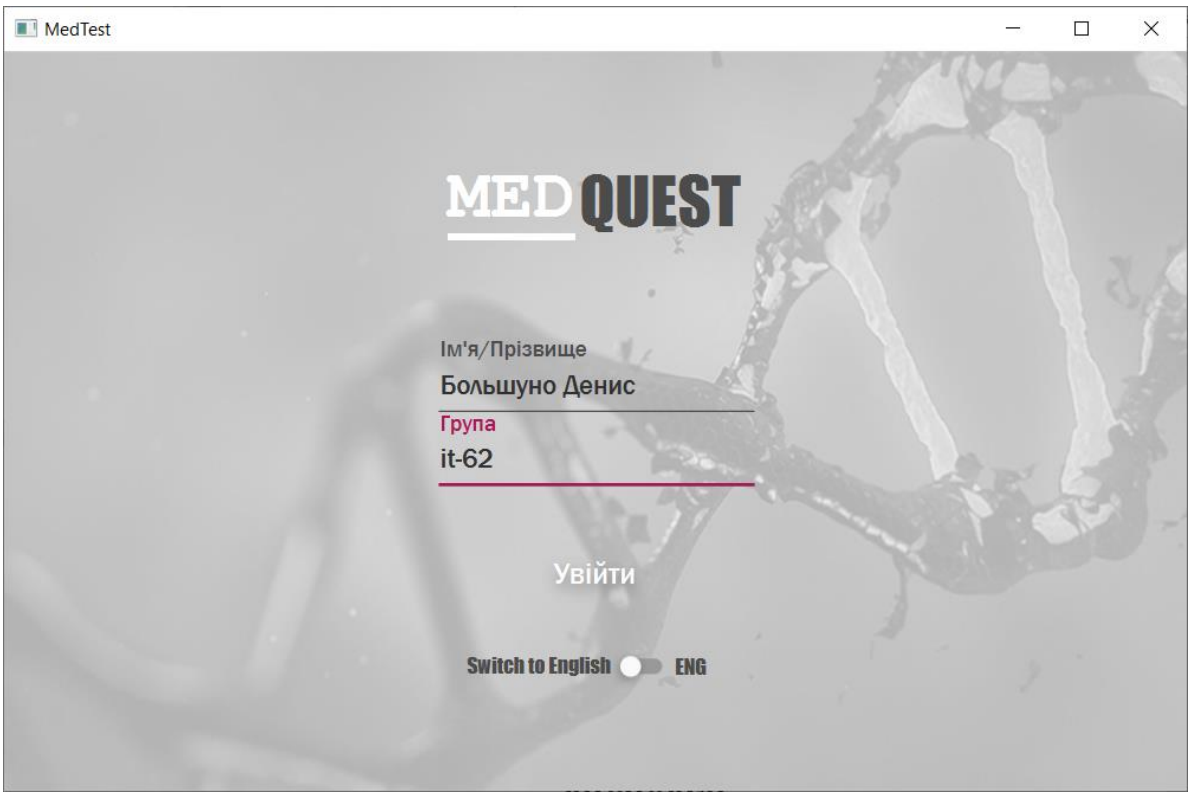

Рисунок 3.7 – Вікно ідентифікації користувача

Повідомлення про помилку наведено на рисунку 3.8.

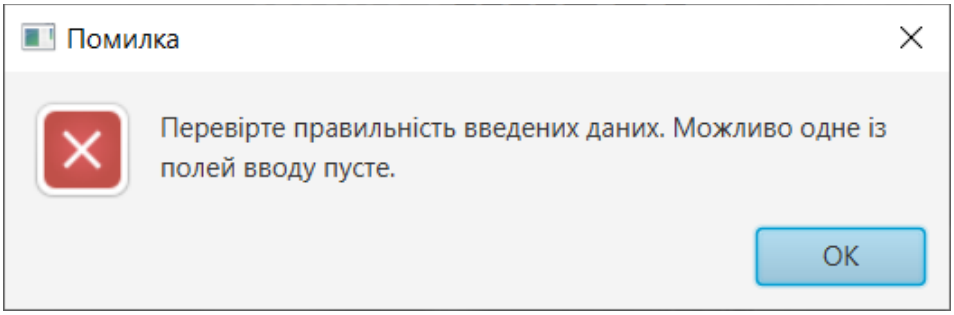

Рисунок 3.8 – Спрацювання помилки

При успішній Ідентифікації, користувач потрапляє на Вікно привітання (рис.3.9). В цьому вікні наведена інформація про додаток та мету тесту. Також в даному вікні є дві кнопки. Кнопка «Вийти» слугує для закінчення роботи додатку та повного виходу з нього. Кнопка «Почати» перенаправить користувача у Вікно проходження тесту.

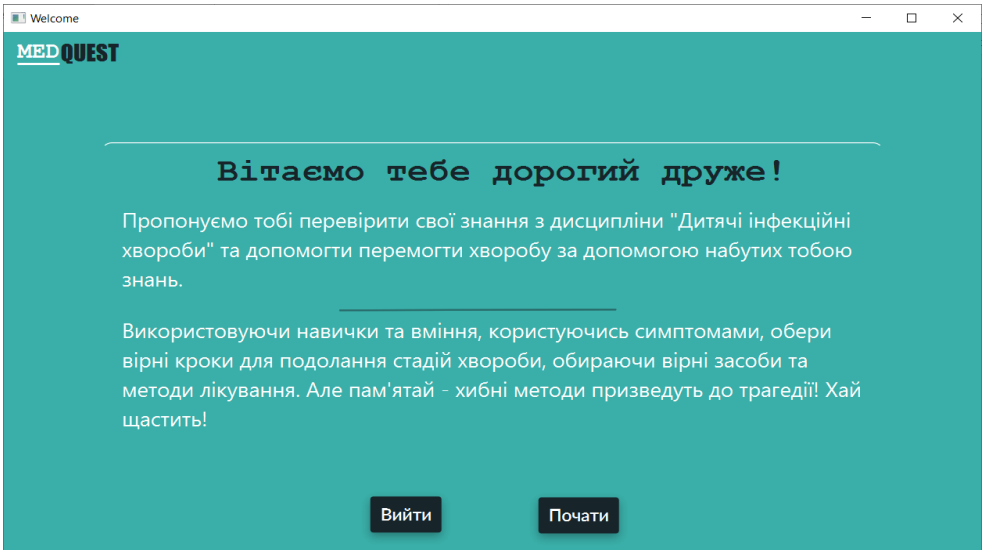

Рисунок 3.9 – Вікно привітання

У вікні проходження тесту (рис.3.10) відображається відео фрагмент, запитання до відео та варіанти відповідей на запитання. Користувач після перегляду відео читає запитання та обирає одну із відповідей. Кожен із варіантів відповідей має своє значення – правильне чи неправильне.

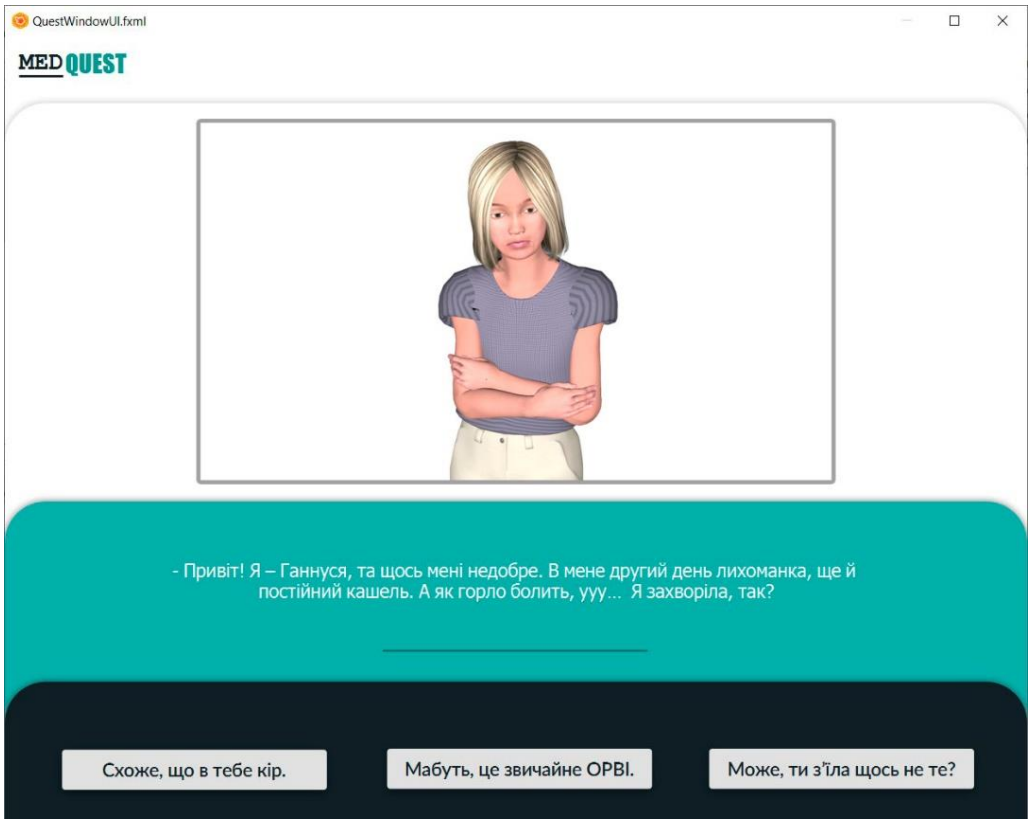

Рисунок 3.10 – Вікно Проходження тесту

Залежно від відповіді додаток переміщує користувача згідно варіативної схеми проходження тесту на питання наступного рівня. При цьому відбувається підключення до бази даних, із неї обирається відповідний відео-фрагмент та текст питання, які відтворюються у вікні додатку.

Приклад питання проміжного рівня наведено на рисунку 3.11.

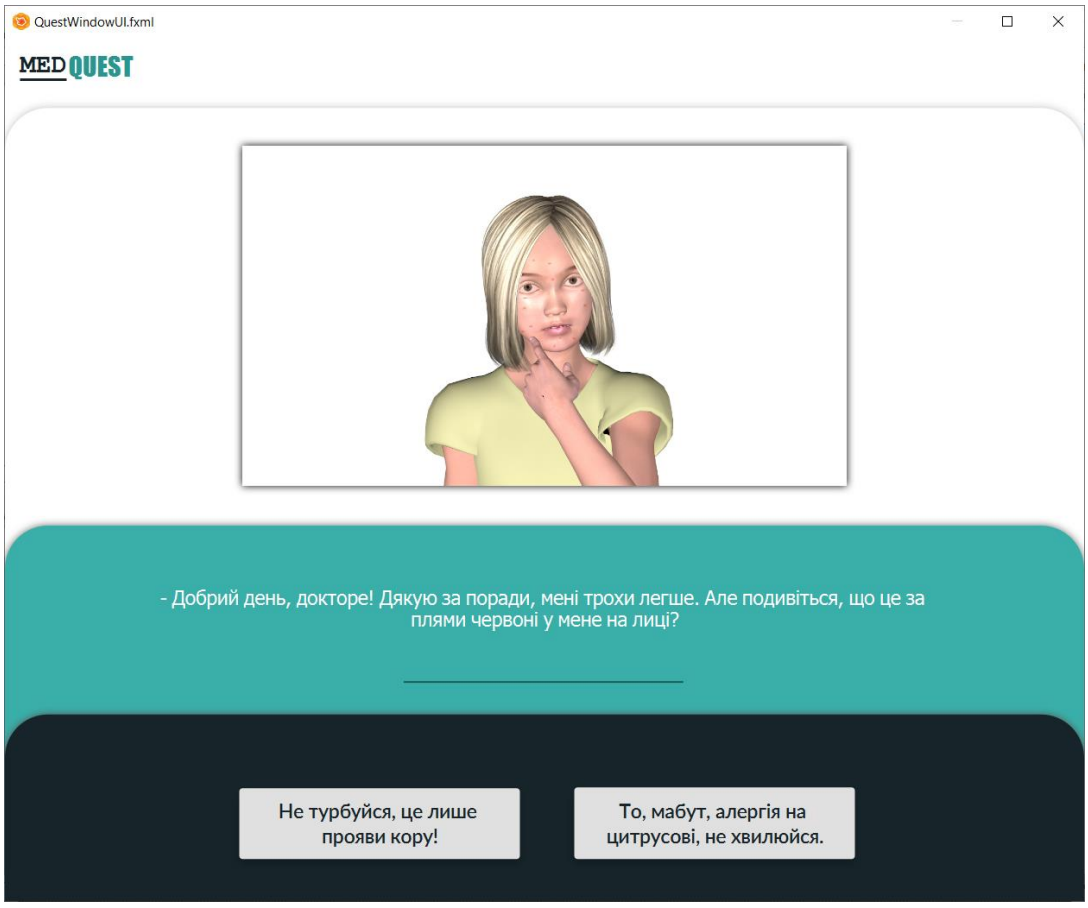

Рисунок 3.11 – Проміжний рівень Вікна проходження тесту

Після закінчення проходження тесту, додаток сформує повідомлення про закінчення тестування та про те, що результат буде надіслано на пошту й збережено в .pdf файл із підрахованою кількістю правильних відповідей.

Пошта, на котру прийде результат, буде у викладача. Акаунт спеціально створений для додатку, доступ наданий тільки викладачу – це обумовлено вимогами замовника. Зручність такого підходу полягає в тому, що пошта від студентів-користувачів додатку буде виокремлена від іншої пошти викладача, і

буде зручніше переглядати результати виконання тренажера. Результат збережений у .pdf файл не підлягає редагуванню, тому користувачі не зможуть змінити дані в файлі із результатом. Приклад повідомлення про закінчення проходження тесту наведено на рисунку 3.12.

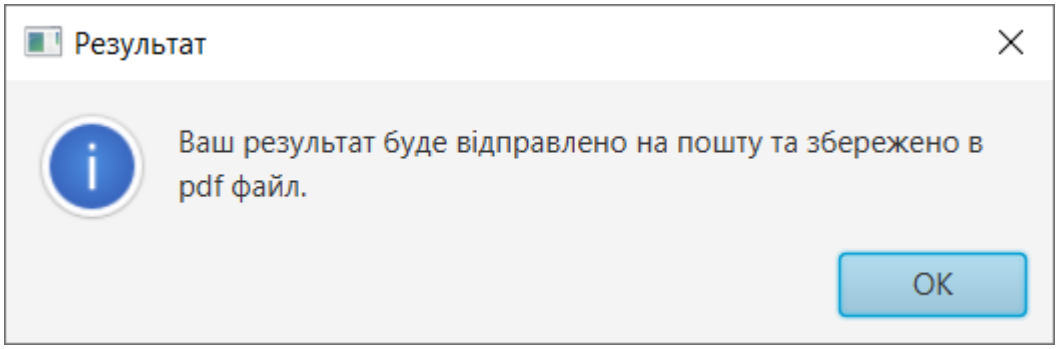

Рисунок 3.12 – Повідомлення про закінчення проходження тесту

Також на рисунку 3.13 безпосередньо зображено вікно результату.

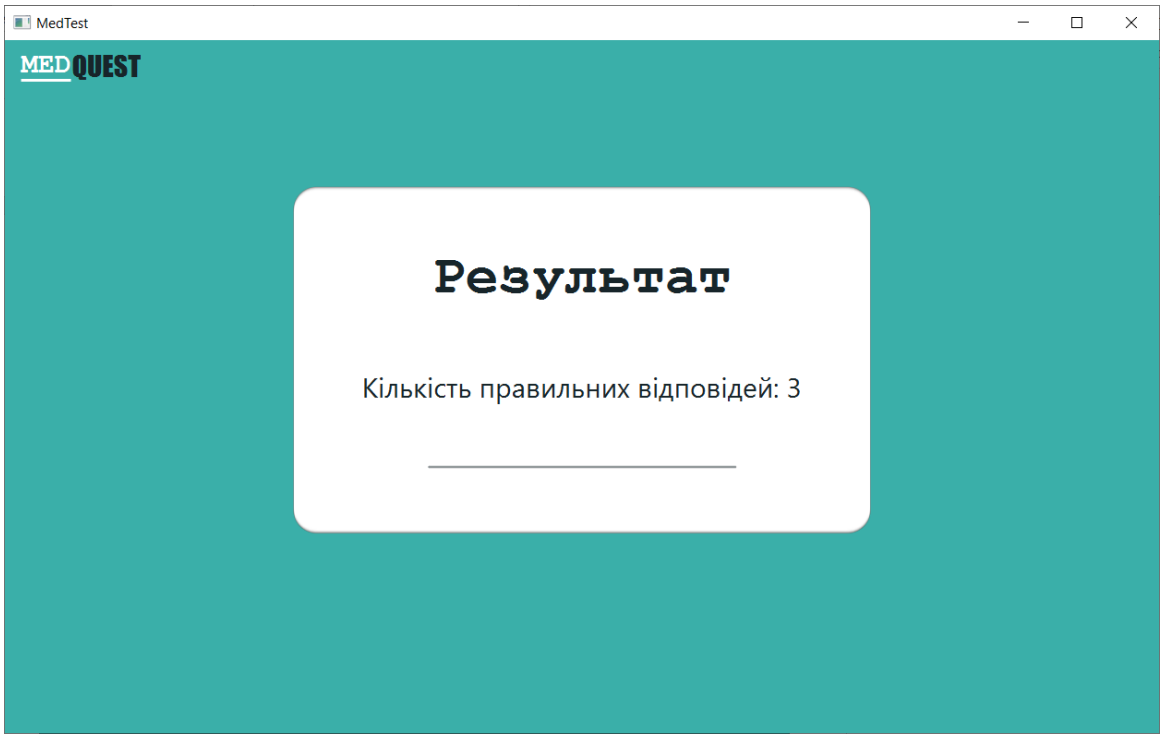

Рисунок 3.13 – Вікно результату

Результат відправлення на пошту зображено на рисунку 3.14.

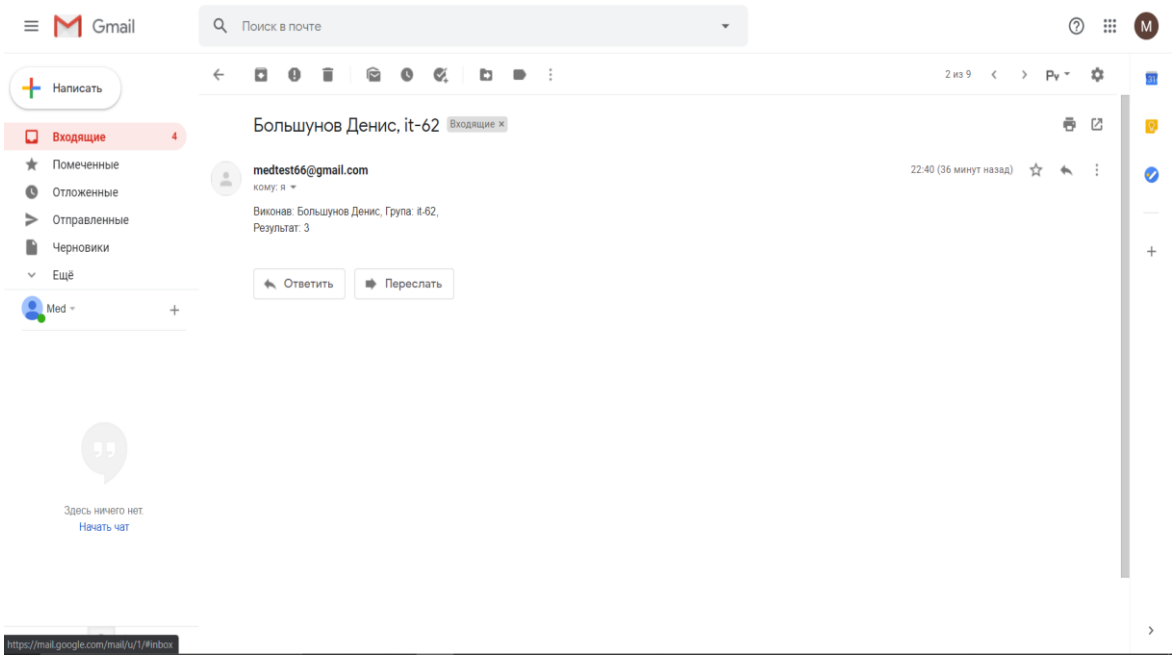

Рисунок 3.14 – Результат збереження на пошту

Результат збереження в .pdf файл зображено на рисунку 3.15.

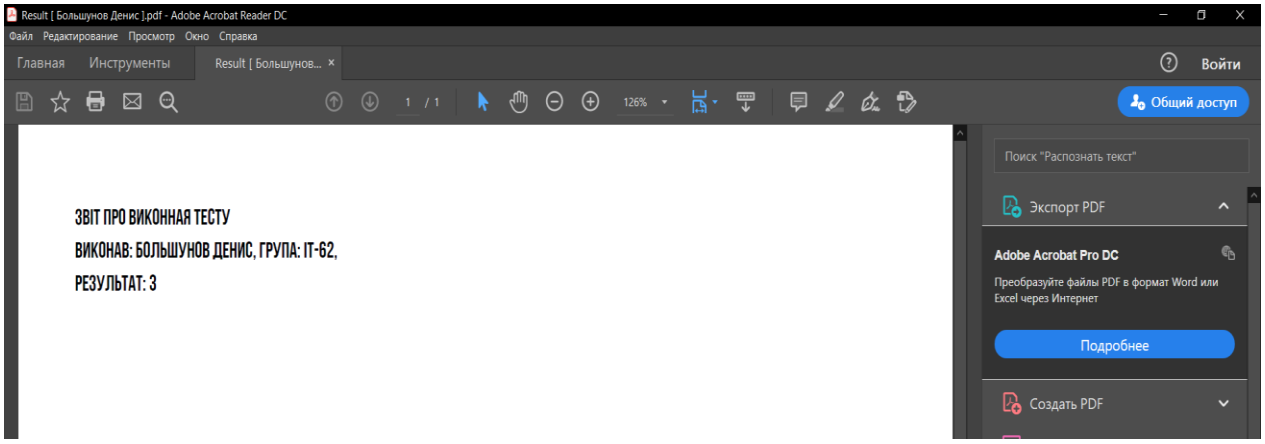

Рисунок 3.15 – Результат збереження в .pdf файл

Таким чином можна зробити висновок, що поставлені задачі виконано – розроблений інтерактивний тренажер, який дозволить провести тестування студентів медичного профілю при вивченні тем дитячих інфекційних хвороб.

ВИСНОВКИ

В результаті виконання дипломної роботи було проведено огляд останніх досліджень та публікацій, аналіз існуючих додатків зі схожою тематикою.

Було сформульовано мету та задачі дослідження, сформульовані вимоги до створеного додатку. В якості засобів реалізації додатку обрані мова Java та середовище для швидкого програмування JetBrains Intellij IDEA, для розробки інтерфейсу додатку обрано JavaFX Scene Builder та для бази даних обрано MySQL Workbench.

Було проведено структурно-функціональне моделювання роботи додатку, планування робіт з реалізації IT-проекту, визначено календарний план робіт, організаційну структуру та можливі ризики під час виконання проєкту.

Визначено послідовність дій створення додатку для якісної та надійної роботи.

Детально описано функціонування додатку зі всіма його пакетами, класами методами та додатковими файлами, а також з’єднання бази даних з додатком. Розроблено прототип майбутнього додатку та схему варіативності проходження тесту. Реалізовано базу даних для зберігання відео-фрагментів та контенту додатку, логіку проходження тестових питань.

Також було реалізовано збереження результату проходження тесту в файлі .pdf та на пошту викладача.

Розроблений інтерактивний додаток для навчання студентів діагностиці дитячих інфекційних хвороб дозволить оновити та закріпити знання і навички при вивченні спеціальних дисциплін в області дитячої педіатрії.

Проект дозволить підвищити якість навчання студентів-медиків. В подальшому можливе створення мобільного додатку та реалізації кросплатформенності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дитячі інфекційні хвороби: стаття. URL: https://spravochnick.ru/ medicina/ detskie_infekcionnye_bolezni/ (дата звернення: 20.03.2020).
2. Інфекційні захворювання у дітей: діагностика, лікування та профілактика: стаття. URL: http://uldc.com.ua/ru/all-articles/189-infektsionnye-zabolevaniya-u-detej-diagnostika-lechenie-i-profilaktika.html (дата звернення: 17.03.2020).
3. Дитячі інфекції: профілактика, симптоми і лікування: стаття. URL: https://vchaspik.ua/otdohni/krasota-i-zdorove/15/460180-detskie-infekcii-profilaktika-simptomy-i-lechenie (дата звернення: 18.03.2020).
4. Інфекційні захворювання у дітей. Сучасний погляд на діагностику, лікування та профілактику: публікація. URL: https://www.umj.com.ua/article/139394/infektsijni-zahvoryuvannya-u-ditej-suchasnij-poglyad-na-diagnostiku-likuvannya-ta-profilaktiku-3 (дата звернення: 18.03.2020).
5. Гуманітарні проблеми медицини та питання викладання у вищій медичній школі: стаття. URL: https://cyberleninka.ru/article/n/rozrobka-ta-vprovadzhennya-sistemi-elektronnogo-testuvannya-pri-vikladanni-medichnoyi-ta-biologichnoyi-fiziki/viewer (дата звернення: 19.03.2020).
6. Особливості застосування тестового контролю знань студентів при вивченні внутрішньої медицини, клінічної імунології та алергології у вищих медичних навчальних закладах: стаття. URL: https://cyberleninka.ru/article/n/osoblivosti-zastosuvannya-testovogo-kontrolyu-znan-studentiv-pri-vivchenni-vnutrishnoyi-meditsini-klinichnoyi-imunologiyi-ta/viewer (дата звернення: 20.03.2020).
7. Удосконалення підготовки лікарів-інтернів із фаху «Урологія»: URL: http://elib.umsa.edu.ua/jspui/bitstream/umsa/9819/1/Udoskonalennya_pidgotovki_likariv.pdf (дата звернення: 21.03.2020).
8. Програмний комплекс "Інформаційна система перевірки знань в медичній

освіті": наукова робота. URL: https://medinf.tdmu.edu.ua/sceince/software-developement/test/programnijkompleksinformacijnasistemaperevirkiznanvmedicnijosviti (дата звернення: 25.03.2020).

9. Реформування системи медичної освіти в світлі концепції «суспільство знань»: наукова дискусія. URL: https://www.umj.com.ua/article/541/reformuvannya-sistemi-medichnoi-osviti-v-svitli-koncepcii-suspilstvo-znan (дата звернення: 23.03.2020).
10. Медичні тести – основні дисципліни: програмний додаток. URL: https://apps.apple.com/ru/app/%D0%BC%D0%B5%D0%B4%D0%B8%D1%86%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B5-%D1%82%D0%B5%D1%81%D1%82%D1%8B/id1059865223 (дата звернення: 15.03.2020).
11. Тести з фармакології: програмний додаток. URL: https://play.google.com/store/apps/details?id=com.antomy.stomatologytest (дата звернення: 16.03.2020).
12. Загальна патологія інфекційних хвороб. Особливості інфекційного процесу: онлайн тест. URL: https://onlinetestpad.com/ru/testview/265270-obshhaya-patologiya-infekcionnykh-boleznej-osobennosti-infekcionnogo-proces (дата звернення: 18.03.2020).
13. Коваль, М.В. Інформаційна система підтримки процесу навчання медичних спеціалістів [Текст]: робота на здобуття кваліфікаційного ступеня магістра; спец.: 122 – Комп'ютерні науки / М.В. Коваль; наук. керівник І.В. Баранова. - Суми: СумДУ, 2018. - 74 с.
14. Середовище розробки JetBrains Intellij IDEA: програмне забезпечення. URL: https://intellij-support.jetbrains.com/hc/en-us/?intellij-idea&_ga=2.171930400.1038308580.1587906108-712935554.1586009080# (дата звернення: 01.04.2020).
15. Документація JavaFX Scene Builder: програмне забезпечення. URL: https://gluonhq.com/products/scene-builder/ (дата звернення: 01.04.2020).
16. Документація JavaFX: програмне забезпечення. URL:

https://openjfx.io/javadoc/14/ (дата звернення: 01.04.2020).

17. Визначення інтерфейсу в FXML: документація. URL: https://metanit.com/java/javafx/2.1.php (дата звернення: 10.04.2020).
18. Java FX 2.0 з використанням CSS: стаття. URL: https://devcolibri.com/javafx-2-0-c-%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC-css/ (дата звернення: 20.04.2020).
19. Основи роботи з MySQL Workbench: швидкий старт, управління схемою даних: стаття. URL: http://mithrandir.ru/professional/soft-and-hardware/mysql-workbench-basics.html (дата звернення: 18.04.2020).
20. Методологія IDEF0. URL: https://itteach.ru/bpwin/metodologiya-idef0 (дата звернення: 20.04.2020).
21. Уніфікована мова моделювання UML: електронні навчальні курси. URL: http://www.znannya.org/?view=uml (дата звернення: 25.04.2020).
22. Побудова діаграм варіантів використання (UseCase Diagrams): реферат. URL: http://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/laboratorna-robota-5-diahramy-variantiv-vykorystannja.pdf (дата звернення: 25.04.2020).
23. Поняття ER-моделі. Поняття сутності (entity). Атрибути. види атрибутів: стаття. URL: https://www.bestprog.net/ru/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ru/ (дата звернення: 05.04.2020).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку інтерактивного додатку для навчання студентів діагностиці дитячих інфекційних хвороб

Суми 2020

1. Призначення й мета створення додатку

1.1.Призначення інтерактивного додатку

Інтерактивний додаток призначений для навчання студентів методам діагностики інфекційних хвороб у педіатрії.

1.2.Мета створення інтерактивного додатку

Закріпити отримання теоретичних знань та практичних навичок студентами медичних спеціальностей в галузі педіатрії.

1.3.Цільова аудиторія

До цільової аудиторії інтерактивного додатку можна віднести наступні групи:

1. Студенти.
2. Викладачі.
3. Інші зацікавлені відвідувачі.

2. Вимоги до інтерактивного додатку

2.1.Вимоги до інтерактивного додатку в цілому

2.1.1. Вимоги до структури й функціонування інтерактивного додатку

Додаток повинен бути реалізований у вигляді готового програмного продукту, доступ до якого можна отримати після його завантаження та інсталяції на комп’ютер. Додаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.1.2. Вимоги до персоналу

Від персоналу не повинно вимагатися спеціальних технічних навичок, знання технологій або програмних продуктів, за винятком загальних навичок роботи з персональним комп'ютером та настільними додатками.

2.1.3. Вимоги до збереженні інформації

Збереження необхідних даних в додатку забезпечується за допомогою створеної бази даних.

2.1.4. Вимоги до розмежування доступу

Розмежування прав доступу в додатку можна розділити на такі групи:

1. Відвідувачі
2. Розробник

Відвідувачі мають доступ до загальнодоступної частини додатку.

Розробник має такі права:

- редагувати базу даних;
- додавати нові розділи та матеріали в додаток;

редагувати програмний код.

2.2.Вимоги до функцій, виконуваних сайтом

2.2.1. Основні вимоги

Структура додатку містить в собі наступні розділи:

- Вікно ідентифікації – початкове вікно, в котрому користувач вводить свої дані для фіксації результату проходження тесту, та (варіативно) пошту викладача для відправлення результату, обирання мови для зручного використання додатку;
- Головне вікно – вікно вітання та опис задачі, яку виконує додаток, перехід на вікно тесту або вихід з додатку;
- Вікно тесту – основна частина додатку;
- Вікно виведення результату, відправлення на пошту та збереження в .pdf файл.

Всі елементи меню та призначеного для користувача інтерфейсу повинні бути добре видимими та інтуїтивно зрозумілими для будь-якого користувача.

Графічне оформлення — елементи навігації повинні бути контрастними

фону і відрізнятися від основного тексту, але при цьому гармоніювати із загальним дизайном ресурсу.

Система повинна забезпечувати навігацію по всіх доступних користувачеві ресурсам і відображати відповідну інформацію. Зручна навігація по додатку є частиною роботи над юзабіліті - зручності для користувачів сайту за низкою основних ознак.

Взаємозв'язок між розділами додатку представлено на Рисунку 1.

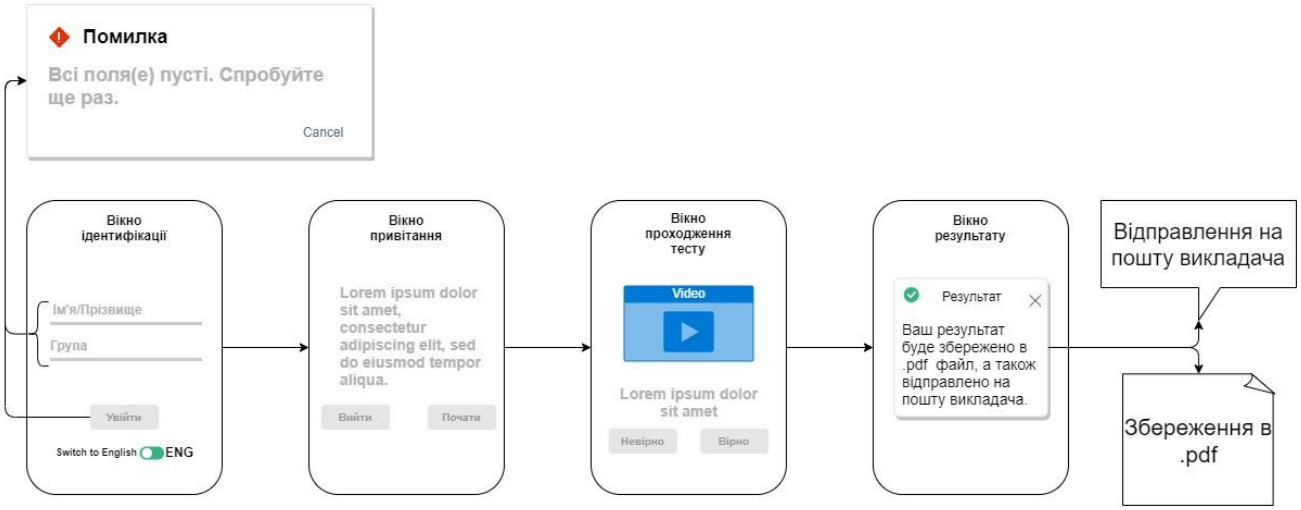

Рисунок 1 – Карта додатку

Наповнення додатку формується за допомогою інформації, яка зберігається в базі даних, а саме запитань та відео роликів. У програмному коді реалізовані команди та функції, які дозволяють при роботі додатку завантажувати відповідну інформацію з бази даних, таким чином наповнюючи додаток необхідним контентом.

2.2.2. Вимоги до функціональних можливостей

Інтерактивний додаток повинен забезпечити такі функціональні можливості:

- Ідентифікація студентів, для подальшої роботи в додатку;
- Відтворення питань з відеороликами;
- Можливість перемикання мов в додатку;

- Оцінювання правильності відповідей;
- Відображення результату у вікні додатку;
- Збереження даних результату на робочий стіл в форматі PDF, без подальшого редагування;
- Відправлення результату на e-mail викладача.

Виконавець надає замовнику додаток в повністю готовому до використання вигляді і працездатністю всіх модулів. Мова додатку – українська та англійська. Усі елементи та системні повідомлення додатку повинні бути відповідною мовою. Платформа повинна мати зручний і інтуїтивно зрозумілий інтерфейс.

Розташування елементів у Вікні тренажеру показано на Рисунку 2.

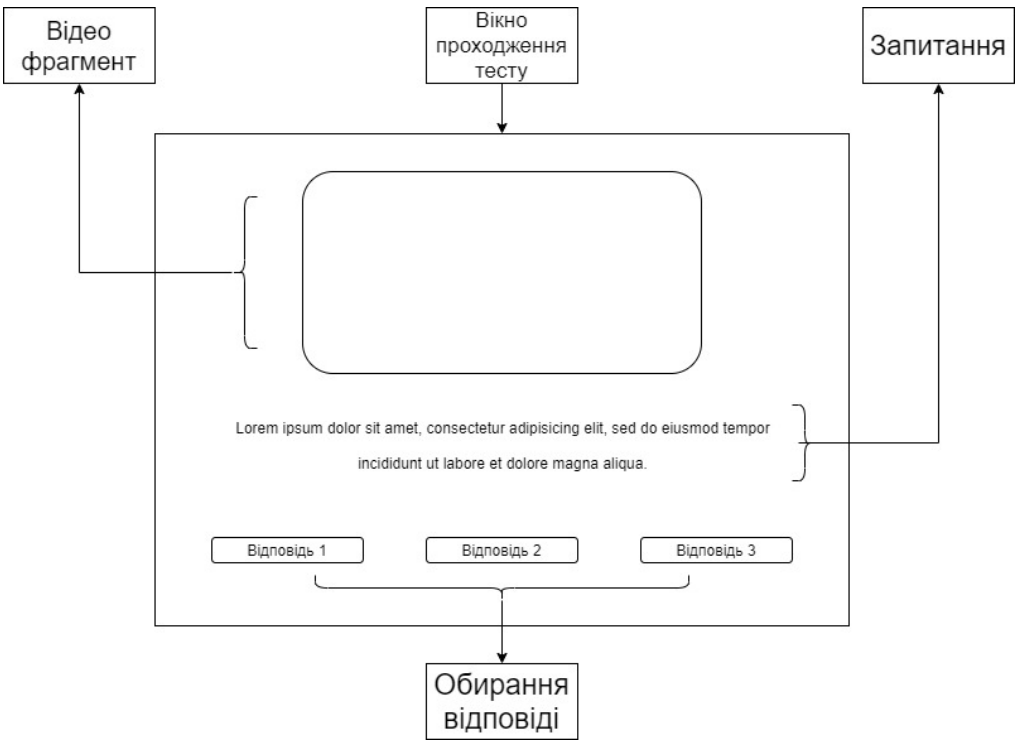

Рисунок 2 – Вікно тренажеру

Типові навігаційні й інформаційні елементи

– Відео ролик
– Розділ із запитанням
– Кнопка підтвердження

– Кнопка скасування

Вікно тренажеру містить вікно відеоролику, в якому вбудована функція setAutoPlay(). Вона забезпечує повторне відтворення ролику до тих пір, поки користувач не підтвердить свій вибір.

Розділ із запитанням не несе в собі ніяких функціональних можливостей, він лише відтворює запитання для користувача.

При натисненні на кнопку Я спробую, користувач матиме змогу розпочати проходження тесту, кожна кнопка буде мати свою оцінку (ця оцінка зберігається в базі даних), яка дозволяє підрахувати кінцевий результат проходження тесту.

При натисненні на кнопку Мені ніколи, користувача буде переправлено на головну сторінку додатку.

3.4.Вимоги до видів забезпечення

Реалізація додатку відбувається з використанням:

- JetBrains Intellij IDEA
- JavaFX Scene Builder
- MySql Workbench

Додаток має бути розроблений на мові Java, а саме з використання фреймворку JavaFX.

Програмне забезпечення користувача повинне задовольняти наступним вимогам:

– Операційна система Windows 10, 8, 7;

Апаратне забезпечення повинне задовольняти наступним вимогам:

– Процесор з тактовою частотою 1.6 Ггц і вище;

– Оперативна пам’ять 2 ГБ і більше;

– Не менш 500 МБ вільного місця на диску.

Апаратне забезпечення користувача повинно задовольняти програмне забезпечення користувача.

4. Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення тренажер-додатку наведено в табл. 1.

Таблиця 1 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Проектування: Розташування медіа, графічних та текстових компонентів в додатку	2 дні
2	Авторизація: Розроблення та реалізація блоку Авторизації в додатку	8 днів
3	Вікно вітання: Розроблення та реалізація Вікна вітання	5 днів
4	Вікно тесту: Розроблення та реалізація Вікна тесту	15 днів
5	База даних: Проектування, та реалізація бази даних, а саме таблиць, для збереження інформації	30 днів
8	Завершення роботи: Проведення стилістичних виправлень додатку, перевірка (тестування) реалізованого функціоналу	2 дні
	Загальна тривалість робіт	62 дні

5. Вимоги до складу й змісту робіт із введення сайту в експлуатацію

Перехід від тестового режиму до робочого проводиться після перевірки всіх розділів та функцій, які виконує додаток, при узгоджені з замовником. Для Замовника повинно бути проведено навчання користувачів по роботі з додатком. Розробник не повинен вносити поправки в дані, котрі отримав від замовника, попередньо не проконсультувавшись з ним.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Метою переддипломної роботи є розробка інтерактивного додатка для навчання студентів діагностиці дитячих інфекційних хвороб. Даний проєкт допоможе студентам медичних закладів оновити та перевірити свої знання в області дитячої педіатрії.

S. Розробка структури інтерактивного тест додатку для вивчення медичних дисциплін з педіатрії.

M. Кінцевим результатом даного проекту повинен бути інтерактивний тест додаток, відправлення результату на пошту та збереження в pdf файл. Користувачу необхідно Ідентифікуватися, для подальшого проходження тесту. Додаток повинен містити тільки ту інформацію, яка відносяться безпосередньо до заданого завдання.

A. Так як проект буде мати мультимедійний інтерфейс, необхідно підібрати певну мову програмування, розробник повинен володіти знаннями з ООП, тому що мову програмування було обрано Java, використання мови програмування JavaFX або Swing, також необхідно володіти мовою MySQL, для проектування та написання запитів до БД, і Scene Builder (XML), для побудови графічної частини додатку.

R. Поставлену мету звичайно можна реалізувати тому, що на даний момент існує велика кількість ПЗ та ресурсів, які за допомогою своїх функцій, скорочують час побудови, проектування, розробки проекту.

T. Обмеженість в часі зумовлена рішенням замовника, щоб якомога швидше отримати програмний продукт (додаток).

Структурна декомпозиція роботи **(структура розбиття роботи, WBS)** – це ієрархічна побудова роботи, побудована з метою логічного розподілу всіх служб

для виконання проекту та подана у графічній формі. Це поєднання декількох рівнів, кожен з яких в кінцевому рахунку розвивається між розподілом роботи попереднього рівня на його компоненти. Компонент найнижчого рівня - це склад роботи, або так званий робочий пакет.

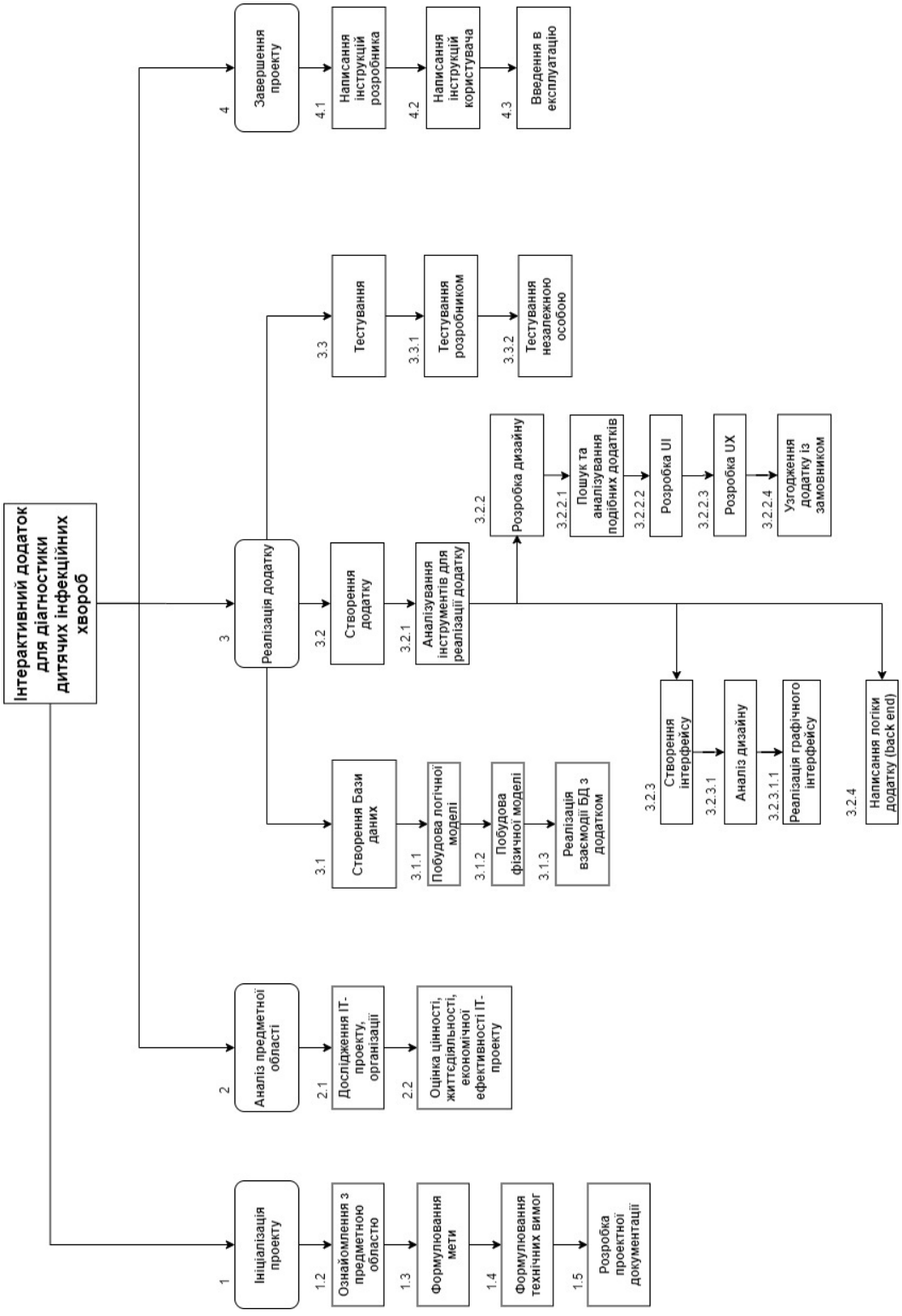

Рисунок Б.1 - Структура проекту

Наступним кроком буде створення OBS структура. **Організаційна структура проекту (OBS)** – представляється графічним відображенням учасників проекту (фізичних та юридичних осіб) та їх відповідальних осіб, які беруть участь у здійсненні проекту. На найвищому рівні проекту OBS розташовані менеджер та команда з управління проектами; на наступному рівні виступають виконавці. Кінцевий рівень структури OBS представлений виконавцями. Не обов'язково бути менеджерами, а тими працівниками, яким безпосередньо доручено створити та відповідати підряднику для впровадження конкретного компонента структури WBS.

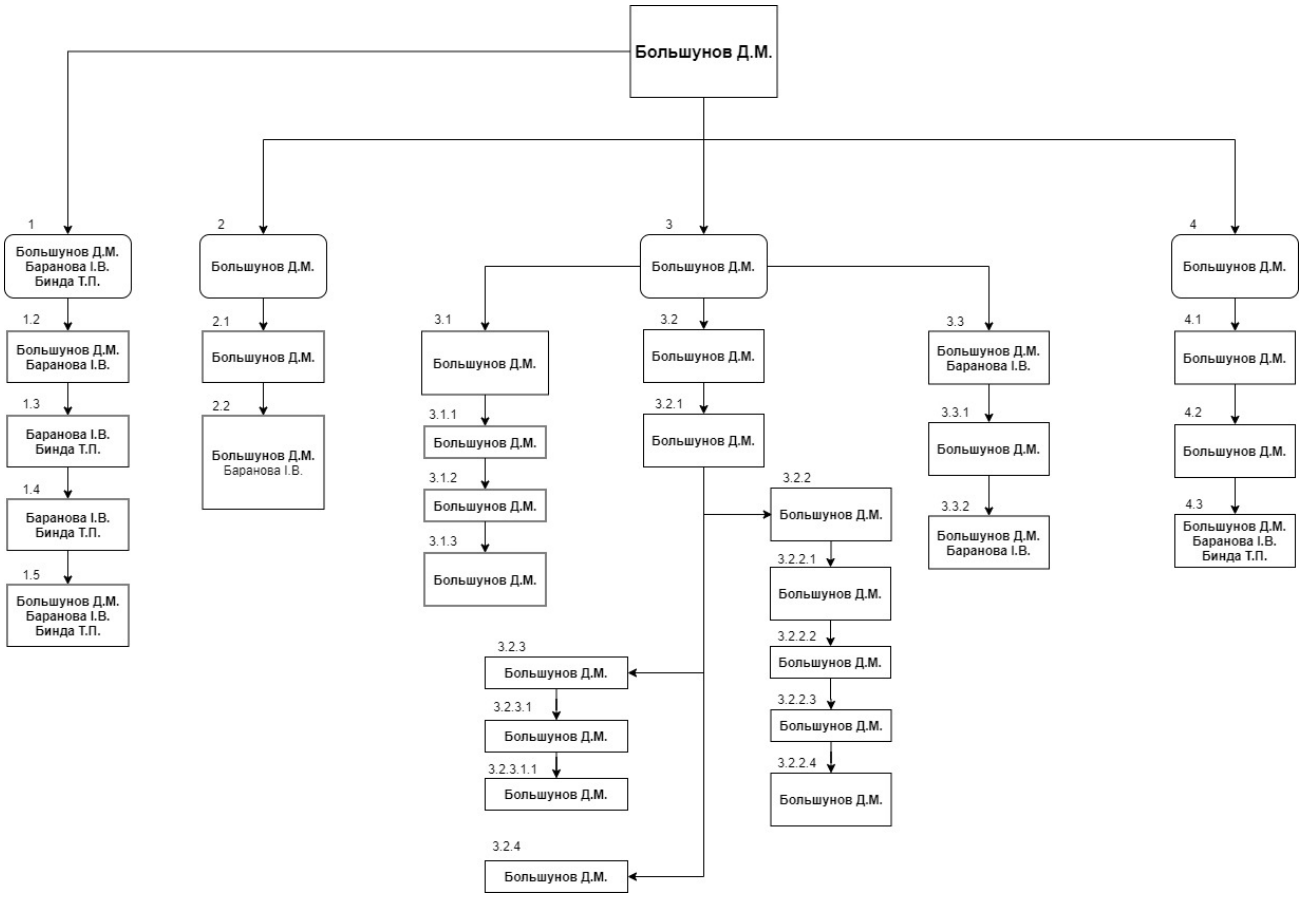

Рисунок Б.2 - Організаційна структура проекту

Діаграма Ганта – це відома версія діаграми (придумана Генрі Ганттом), яка застосовується для планування та контролю за виконанням проекту. Така інтерактивна мережева діаграма існує майже у всіх системах управління проектами. На схемі (структурі) відзначено завдання та стадії плану з

урахуванням часу їх виконання. Завдання на діаграмі можуть існувати залежно одне від одного (наприклад, одне завдання може з’являтися лише в кінці другого). Крім того, може відображатися відсоток виконання будь-якого завдання та відповідальність за його виконання.

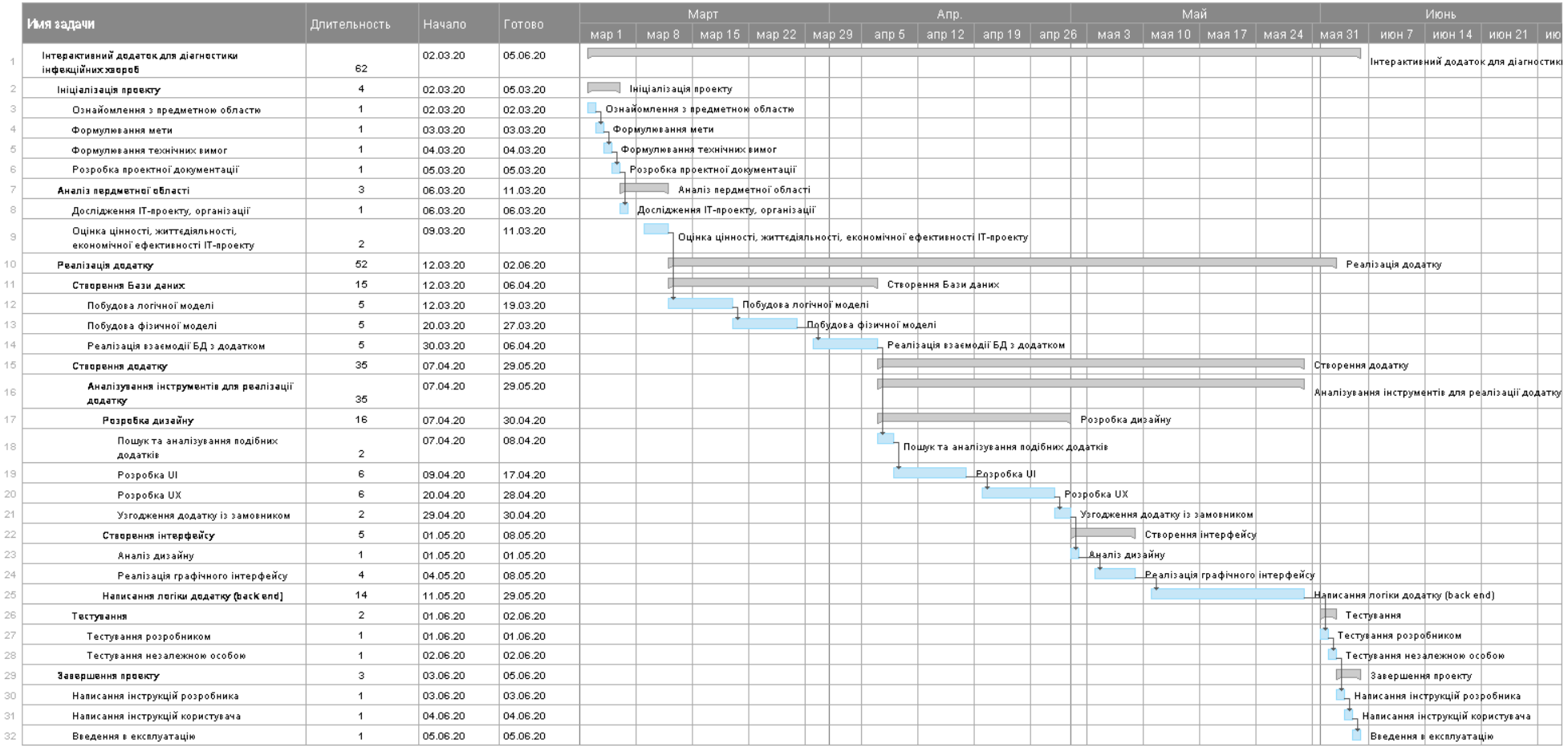

Рисунок Б.2 - Діаграма Ганта

Побудова матриці відповідності. Матриця відповідальності – це спеціальна методика пошуку функціональних зон, ключових сфер діяльності, критеріїв прийняття адміністративних рішень, де є неоднозначності. Усі розбіжності, що виникають у процесі передбаченого процесу, можуть бути винесені на загальний розгляд та згодом вирішені шляхом прийняття спільного рішення. У таблиці Б.1 показано матрицю відповідальності проекту.

Таблиця Б.1 – RAM

№	**Фази**	Большунов Д.М.	Баранова І.В.	Бинда Т.П.
1)	Ідентифікація проекту	A	A	R
2)	Ознайомлення з предметною областю	R	C	
3)	Формулювання мети	R	A	
4)	Формулювання технічних вимог	R	C	C
5)	Розробка проектної документації	R	C	
6)	Аналіз предметної області	R	C	
7)	Дослідження ІТ-проекту, організації	R		
8)	Оцінка цінності, життєздатності, економічної ефективності ІТ-проекту		R	
9)	Реалізація додатку	R	C	
10)	Створення Бази Даних	R	C	
11)	Побудова логічно моделі	R	C	
12)	Побудова фізичної моделі	R	C	
13)	Реалізація взаємодії БД із додатком	R	C	
14)	Створення додатку	R	A	C
15)	Обирання мови програмування	R		
16)	Створення інтерфейсу	R	C	C
17)	Написання логіки додатку	R		
18)	Розробка дизайну	R	C	C
19)	Пошук та аналізування подібних додатків	R		
20)	Розробка UI	R	A	
21)	Розробка UX	R	A	
22)	Узгодження дизайну із замовником	R	C	C
23)	Тестування розробником	R	I	
24)	Тестування незалежною особою	I	R	
25)	Завершення проекту	R	C	I
26)	Написання інструкцій адміністратора	R	C	
27)	Написання інструкцій для користувача	R	C	I
28)	Введення в експлуатацію	R	A	I

Ризик – це можлива подія, яка у разі настання негативно або позитивно впливатиме на проект.

Процес управління ризиками включає наступні етапи:

1. ідентифікація
2. процес оцінювання ризиків, який включає в себе якісний, кількісний аналіз.
3. заходи реагування на ризики
4. моніторинг заходів і ризиків

Ризики проекту представлені у таблиці Б.2.

Таблиця Б.2 – Risk Register

№	Назва ризику	Ймовірність	Величина втрат
1	Замовник довго не відповідає	1	4
2	Неоптимальний розподіл часу	4	4
3	Неполадки з Інтернет зв’язком	4	5
4	Недостатній рівень знань розробника	1	3
5	Замовника не влаштовує кінцевий результат	2	2
6	Захворювання розробника	2	3
7	Проблеми із Базою Даних	2	3
8	Платне програмне забезпечення	4	3
9	Вихід з ладу апаратного забезпечення	5	2
10	Затримка надання матеріалу	2	4

1. За імовірністю виникнення:
   - слабо ймовірнісні - 1;
   - мало ймовірнісні - 2;
   - імовірні - 3;

- досить імовірні - 4;
- майже імовірні - 5.

2. За величиною втрат:
   - Мінімальна - 1;
   - Низька - 2;
   - Середня - 3;
   - Висока - 4;
   - Максимальна - 5.

Таблиця Б.3 - Матриця ризиків

5		9			
4			8	2	3
3					
2		**5**	**6, 7**	**10**	
1			**4**	**1**	
Величина втрат / Ймовірність	1	2	3	4	5

Формування бюджету. Останнім етапом планування проекту – є етап розподіл бюджету даного проекту. Були визначені особи, які брали участь в проекті та між якими необхідно розподілити бюджет.

У таблиці Б.4 приведений бюджет на витрачення заробітної плати.

Таблиця Б.4 – Розподілення бюджету

Людина	За 1 год/грн	Робочих годин	Сума
Большунов Д.М.	35	730	25550
Баранова І.В.	28	400	11200
Бинда Т.П.	23	13	174

Бюджет заробітної плати складає **36924 грн.**

ДОДАТОК В

В.1 Головний клас

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("StartWindow/AuthorizationUI.fxml"));
        primaryStage.setTitle("MedTest");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }


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

В.2 Ідентифікація користувача

```
package sample.StartWindow;

import com.jfoenix.controls.JFXButton;

import java.io.IOException;
import java.util.Locale;
import java.util.ResourceBundle;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import com.jfoenix.controls.JFXTextField;
import com.jfoenix.controls.JFXToggleButton;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;
import sample.Animation.FadeTransitionAnimation;

public class AuthorizationController extends Node {

    public static Locale locale = new Locale("uk", "UA");

    @FXML
```

```
    private JFXTextField name;

    public static String userName;

    @FXML
    private JFXTextField group;

    public static String getGroup;

    public Alert alert;

    @FXML
    private JFXButton signInBtn;

    @FXML
    private JFXToggleButton switchLanguage;

    @FXML
    void initialize() {

        ResourceBundle rb = ResourceBundle.getBundle("data", locale);
        name.setPromptText(rb.getString("userName"));
        group.setPromptText(rb.getString("userGroup"));
        signInBtn.setText(rb.getString("startButton"));

        alert = new Alert(Alert.AlertType.ERROR);
    }

    @Override
    public Node getStyleableNode() {
        return null;
    }

    @FXML
    void switchScenes(ActionEvent event) throws IOException {

        getGroup = group.getText();
        userName = name.getText();

        if(name.getText().isEmpty() || group.getText().isEmpty()){
            showAlertWithHeaderText();
        }
        else {
            Parent loadWelcomeWindow =
FXMLLoader.load(getClass().getResource("../WelcomeWindow/WelcomeUI.fxml"));
            Scene loadWelcomeScene = new Scene(loadWelcomeWindow);
            Stage welcomeWindowStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            welcomeWindowStage.setScene(loadWelcomeScene);
            welcomeWindowStage.hide();
            welcomeWindowStage.setTitle("Welcome");
            welcomeWindowStage.show();
        }
    }

    @FXML
    void onMouseEntered(MouseEvent event) {
        FadeTransitionAnimation.singleBtnFadeTransition(signInBtn, 0.1, true);
    }

    @FXML
    void onMouseExited(MouseEvent event) {
        FadeTransitionAnimation.singleBtnFadeTransition(signInBtn, 0.0, false);
    }
```

```
    private void showAlertWithHeaderText() {
        alert.setTitle("Помилка");
        alert.setHeaderText(null);
        alert.setContentText("Перевірте правильність введених даних. Можливо одне із полей вводу пусте.");
        alert.showAndWait();
    }

    @FXML
    void switchToAntherLang(ActionEvent event) {

        if (switchLanguage.isSelected()){
            locale = new Locale("en", "US");

        }
        else{
            locale = new Locale("uk", "UA");
        }

        ResourceBundle rb = ResourceBundle.getBundle("data", locale);
        name.setPromptText(rb.getString("userName"));
        group.setPromptText(rb.getString("userGroup"));
        signInBtn.setText(rb.getString("startButton"));

    }
}
```

В.3 Вікно Привітання

```
package sample.WelcomeWindow;

import com.jfoenix.controls.JFXButton;

import java.io.IOException;
import java.util.ResourceBundle;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;
import sample.Animation.FadeTransitionAnimation;
import sample.StartWindow.AuthorizationController;

public class WelcomeClass {

    @FXML
    private Label welcomeHeader;

    @FXML
    private Label welcomeText_1;

    @FXML
    private Label welcomeText_2;

    @FXML
    private JFXButton startBtn;
```

```
    @FXML
    private JFXButton endBtn;

    @FXML
    void initialize() {

        ResourceBundle rb = ResourceBundle.getBundle("data", AuthorizationController.locale);
        welcomeHeader.setText(rb.getString("welcomeHeader"));
        welcomeText_1.setText(rb.getString("welcomeText_1"));
        welcomeText_2.setText(rb.getString("welcomeText_2"));
        startBtn.setText(rb.getString("startBtn"));
        endBtn.setText(rb.getString("exitBtn"));

    }

    @FXML
    void goToQuest(ActionEvent event) throws IOException {
        Parent loadQuestWindow = FXMLLoader.load(getClass().getResource("../QuestWindow/QuestWindowUI.fxml"));
        Scene loadQuestScene = new Scene(loadQuestWindow);
        Stage questWindowStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        questWindowStage.setScene(loadQuestScene);
        questWindowStage.hide();
        questWindowStage.setTitle("MedTest");
        questWindowStage.show();
    }

    @FXML
    void exitTheApplication(ActionEvent event) {
        System.exit(0);
    }

    @FXML
    void WelcomeOnMouseEntered(MouseEvent event) {
        FadeTransitionAnimation.setFadeTransition(startBtn, endBtn, 0.1);
    }

    @FXML
    void WelcomeOnMouseExited(MouseEvent event) {
        FadeTransitionAnimation.setFadeTransition(startBtn, endBtn, 0.0);
    }

}
```

В.4 Вікно Проходження тесту

```
package sample.QuestWindow;

import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Font;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.BaseFont;
import com.itextpdf.text.pdf.PdfWriter;
import com.jfoenix.controls.JFXButton;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
```

```
import javafx.scene.media.MediaView;
import sample.Animation.FadeTransitionAnimation;
import sample.Database.dao.QuestDaoImpl;
import sample.Database.services.db.ConnectionService;
import sample.StartWindow.AuthorizationController;
import sample.model.Quest;

import javax.mail.*;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Properties;


public class QuestWindowClass {

    private static int trueCount;

    private int maxLevel;
    private QuestDaoImpl dao;

    private Quest quest;

    @FXML
    private Label questionText;
    @FXML
    private MediaView mvObject;

    private MediaPlayer mpObject;

    private Media mediaObject;

    @FXML
    private JFXButton agreementBtn;

    @FXML
    private JFXButton disagreementBtn;

    private String dbName;

    public Alert alert;

    private static String resultUK = "Виконав: %s, Група: %s, \nРезультат: %s";
    private static String resultEN = "Completed: %s, Group: %s, \nResult: %s";

    //Отправка на почту преподавателя
    public static void sendMail() throws MessagingException {

        final String username = "medtest66@gmail.com";
        final String password = "21medtest";

        Properties prop = new Properties();
        prop.put("mail.smtp.host", "smtp.gmail.com");
        prop.put("mail.smtp.port", "465");
        prop.put("mail.smtp.auth", "true");
        prop.put("mail.smtp.socketFactory.port", "465");
        prop.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");

        Session session = Session.getInstance(prop,
                new javax.mail.Authenticator() {
```

```
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(username, password);
                }
            });

        try {

            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(username));
            message.setRecipients(
                    Message.RecipientType.TO,

                    //Введите почту преподавателя(получателя)
                    InternetAddress.parse(username)
            );

            message.setSubject("" + AuthorizationController.userName + ", " + AuthorizationController.getGroup);
            if (AuthorizationController.locale.getLanguage().equals("uk")) {
                message.setText(String.format(resultUK, AuthorizationController.userName,
AuthorizationController.getGroup, trueCount));
            } else {
                message.setText(String.format(resultEN, AuthorizationController.userName,
AuthorizationController.getGroup, trueCount));
            }

            Transport.send(message);

        } catch (MessagingException e) {
            e.printStackTrace();
        }

    }

    //Сохранение результата в PDF
    public static void saveToPDF() throws IOException, DocumentException {

        Document document = new Document();
        PdfWriter.getInstance(document, new FileOutputStream("Result [ " + AuthorizationController.userName + " ].pdf"));
        document.open();
        BaseFont bf = BaseFont.createFont("Bebas_Neue_Cyrillic.ttf", BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
//подключаем файл шрифта, который поддерживает кириллицу
        Font font = new Font(bf);


        if (AuthorizationController.locale.getLanguage().equals("uk")) {
            Paragraph chunk = new Paragraph("Звіт про виконная тесту", font);
            Paragraph chunkUser = new Paragraph(String.format(resultUK, AuthorizationController.userName,
AuthorizationController.getGroup, trueCount), font);
            document.add(chunk);
            document.add(chunkUser);
        } else {
            Paragraph chunk = new Paragraph("Test report", font);
            Paragraph chunkUser = new Paragraph(String.format(resultEN, AuthorizationController.userName,
AuthorizationController.getGroup, trueCount), font);
            document.add(chunk);
            document.add(chunkUser);
        }
        document.close();
    }

    @FXML
    void initialize() {
```

```
        alert = new Alert(Alert.AlertType.INFORMATION);

        try {

            if (AuthorizationController.locale.getLanguage().equals("uk")) {
                dbName = "db_med_test?";
            } else {
                dbName = "db_med_test_en?";
            }

            dao = new QuestDaoImpl(ConnectionService.getConnection(dbName));

            maxLevel = dao.getMaxLevel();
            dao = new QuestDaoImpl(ConnectionService.getConnection(dbName));

            quest = dao.get(1);

            trueCount = 0;

            if (quest == null) {
                System.exit(0);
            }

            initResource(quest);

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }

    }

    public void initResource(Quest quest) {
        String videoPath = new File(quest.getVideoPath()).getAbsolutePath();
        mediaObject = new Media(new File(videoPath).toURI().toString());
        mpObject = new MediaPlayer(mediaObject);
        mvObject.setMediaPlayer(mpObject);
        mpObject.setAutoPlay(true);

        questionText.setText(quest.getQuestion());

        agreementBtn.setText(quest.getTrueAnswer().getAnswer());
        disagreementBtn.setText(quest.getFalseAnswer().getAnswer());
    }

    @FXML
    void QuestOnMouseEntered(MouseEvent event) {
        FadeTransitionAnimation.setFadeTransition(agreementBtn, disagreementBtn, 0.1);
    }

    @FXML
    void QuestOnMouseExited(MouseEvent event) {
        FadeTransitionAnimation.setFadeTransition(agreementBtn, disagreementBtn, 0.0);
    }

    @FXML
    void trueAnswer(ActionEvent event) throws SQLException, InstantiationException, IllegalAccessException,
ClassNotFoundException, MessagingException, IOException, DocumentException {
```

```
        trueCount++;

        dao = new QuestDaoImpl(ConnectionService.getConnection(dbName));
        if (quest.getQuestLevel() == maxLevel) {
            showResultAlert();
            sendMail();
            saveToPDF();
            System.exit(0);
        }
        int trueId = quest.getTrueQuest();
        quest = dao.get(trueId);
        if (quest == null) {
            System.exit(0);
        }
        initResource(quest);
    }

    @FXML
    void wrongAnswer(ActionEvent event) throws SQLException, InstantiationException, IllegalAccessException,
ClassNotFoundException, MessagingException, IOException, DocumentException {

        dao = new QuestDaoImpl(ConnectionService.getConnection(dbName));
        if (quest.getQuestLevel() == maxLevel) {
            showResultAlert();
            sendMail();
            saveToPDF();
            System.exit(0);
        }
        int falseId = quest.getFalseQuest();
        quest = dao.get(falseId);
        initResource(quest);
    }

    private void showResultAlert() {
        alert.setTitle("Результат");
        alert.setHeaderText(null);
        alert.setContentText("Ваш результат буде відправлено на пошту за зебережено в pdf файл.");
        alert.showAndWait();
    }

}
```

В.5 База даних

```
package sample.Database.dao;

import sample.Database.executor.Executor;
import sample.model.Answer;
import sample.model.Quest;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

public class QuestDaoImpl implements QuestDao {

    private Executor executor;

    public static String tableLocale;

    public QuestDaoImpl(Connection connection) {
```

```
    this.executor = new Executor(connection);
}

@Override
public void insert(Quest quest) throws SQLException {

}

@Override
public void update(Quest quest) throws SQLException {

}

@Override
public Quest get(int id) throws SQLException {

    return executor.execQuery("select \n" +
                "  q.question_id, \n" +
                "  q.question, \n" +
                "  q.question_level, \n" +
                "  q.next_true_question_id, \n" +
                "  q.next_false_question_id, \n" +
                "  v.video_path, \n" +
                "  a.answer_id, \n" +
                "  a.answer, \n" +
                "  qha.answer_flag, \n" +
                "  a1.answer_id, \n" +
                "  a1.answer, \n" +
                "  qha1.answer_flag \n" +
                "from \n" +
                "  question q \n" +
                "  join question_has_answer qha on qha.question_id = q.question_id \n" +
                "  join question_has_answer qha1 on qha1.question_id = q.question_id \n" +
                "  join video v on v.video_id = q.video_id \n" +
                "  join answer a on a.answer_id = qha.answer_id \n" +
                "  join answer a1 on a1.answer_id = qha1.answer_id \n" +
                "where \n" +
                "  q.question_id = \n" + id +
                "  and qha.answer_flag = 1 \n" +
                "  and qha1.answer_flag = 0 \n",
            (result) -> {
                if (result.next()) {
                    return new Quest(result.getInt(1), result.getString(2), result.getInt(3),
                            new Answer(result.getInt(7), result.getString(8), result.getBoolean(9)),
                            new Answer(result.getInt(10), result.getString(11), result.getBoolean(12)),
                            result.getInt(4), result.getInt(5), result.getString(6));
                } else {
                    return null;
                }
            });

}

@Override
public Quest get(String questionName) throws SQLException {

    return executor.execQuery("select \n" +
                "  q.question_id, \n" +
                "  q.question, \n" +
                "  q.question_level, \n" +
                "  q.next_true_question_id, \n" +
                "  q.next_false_question_id, \n" +
                "  v.video_path, \n" +
```

```
                "  a.answer_id, \n" +
                "  a.answer, \n" +
                "  qha.answer_flag, \n" +
                "  a1.answer_id, \n" +
                "  a1.answer, \n" +
                "  qha1.answer_flag \n" +
                "from \n" +
                "  question q \n" +
                "  join question_has_answer qha on qha.question_id = q.question_id \n" +
                "  join question_has_answer qha1 on qha1.question_id = q.question_id \n" +
                "  join video v on v.video_id = q.video_id \n" +
                "  join answer a on a.answer_id = qha.answer_id \n" +
                "  join answer a1 on a1.answer_id = qha1.answer_id \n" +
                "where \n" +
                "  q.question = \n" + questionName +
                "  and qha.answer_flag = 1 \n" +
                "  and qha1.answer_flag = 0 \n",
            (result) -> {
               if (result.next()) {
                  return new Quest(result.getInt(1), result.getString(2), result.getInt(3),
                        new Answer(result.getInt(7), result.getString(8), result.getBoolean(9)),
                        new Answer(result.getInt(10), result.getString(11), result.getBoolean(12)),
                        result.getInt(4), result.getInt(5), result.getString(6));
               } else {
                  return null;
               }
            });

}

@Override
public int getQuestId(String questionName) throws SQLException {

   return executor.execQuery("SELECT", (resultSet) -> {
      resultSet.next();
      return resultSet.getInt(1);
   });

}

public int getMaxLevel() throws SQLException {
   return executor.execQuery("select MAX(question.question_level) from question", (resultSet) -> {
      if (resultSet.next()) {
         return resultSet.getInt(1);
      }
      else {
         return 0;
      }
   });
}

@Override
public void delete(Quest quest) throws SQLException {

}

@Override
public List<String> listQuest() throws SQLException {
   return null;
}

@Override
public List<Quest> listFullData() throws SQLException {
```

```
        return null;
    }
}
package sample.Database.dao;

import sample.model.Quest;

import java.sql.SQLException;
import java.util.List;

public interface QuestDao {

    public void insert(Quest quest) throws SQLException;
    public void update(Quest quest) throws SQLException;
    public Quest get(int id) throws SQLException;
    public Quest get(String questionName) throws SQLException;
    public int getQuestId(String questionName) throws SQLException;
    public void delete(Quest quest) throws SQLException;
    List<String> listQuest() throws SQLException;
    List<Quest> listFullData() throws SQLException;
}
package sample.Database.executor;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Executor {

    private final Connection connection;

    public Executor(Connection connection) {
        this.connection = connection;
    }

    public void execUpdate(String value) throws SQLException {
        Statement statement = connection.createStatement();
        statement.execute(value);
        statement.close();
    }

    public <T> T execQuery(String query, ResultHandler<T> handler) throws SQLException {
        if (connection != null) {
            Statement statement = connection.createStatement();
            statement.execute(query);
            ResultSet result = statement.getResultSet();
            T value = handler.handle(result);
            result.close();
            statement.close();
            connection.close();
            return value;
        } else {
            return null;
        }
    }

}
package sample.Database.executor;

import java.sql.ResultSet;
import java.sql.SQLException;
```

```
public interface ResultHandler<T> {

   T handle(ResultSet resultSet) throws SQLException;

}
package sample.Database.services.db;

import java.sql.Connection;
import java.sql.SQLException;

public class ConnectionService {

   public static Connection getConnection(String dbName) throws ClassNotFoundException, SQLException,
InstantiationException, IllegalAccessException {
      return MySqlConnection.getMySqlConnection(dbName);
   }

   public static void closeConnection(Connection connection) {
      try {
         if (connection != null) {
            connection.close();
         }
      } catch (SQLException e) {

      }
   }

   public static void rollbackConnection(Connection connection) {
      try {
         if (connection != null) {
            connection.rollback();
         }
      } catch (SQLException e) {

      }
   }

}
package sample.Database.services.db;

import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MySqlConnection {

   public static Connection getMySqlConnection(String dbName) throws ClassNotFoundException, SQLException,
InstantiationException, IllegalAccessException {
      String driver = "com.mysql.cj.jdbc.Driver";
      String host = "localhost:";
      String port = "3306/";
      String username = "user=root&";
      String password = "password=root&";
      String url = "jdbc:mysql://" +
            host +
            port +
            dbName +
            username +
            password +
            "&serverTimezone=UTC";
      return getMySqlConnection(driver, url);
   }
```

```
    private static Connection getMySqlConnection(String driver, String url) throws SQLException, ClassNotFoundException, IllegalAccessException, InstantiationException {
        DriverManager.registerDriver((Driver) Class.forName(driver).newInstance());
        return DriverManager.getConnection(url);
    }

}
package sample.model;

public class Answer {

    private Integer id;
    private String answer;
    private Boolean answerFlag;

    public Answer(int anInt, String string, boolean aBoolean) {
        this.id = anInt;
        this.answer = string;
        this.answerFlag = aBoolean;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }

    public Boolean getAnswerFlag() {
        return answerFlag;
    }

    public void setAnswerFlag(Boolean answerFlag) {
        this.answerFlag = answerFlag;
    }
}
package sample.model;

public class Quest {

    private Integer id;
    private Integer questLevel;
    private String question;
    private String videoPath;
    private Answer trueAnswer, falseAnswer;
    private Integer trueQuest, falseQuest;

    public Quest(Integer id, String question, Integer questLevel, Answer trueAnswer, Answer falseAnswer, Integer trueQuest, Integer falseQuest, String videoPath) {
        this.id = id;
        this.questLevel = questLevel;
        this.question = question;
        this.videoPath = videoPath;
```

```
    this.trueAnswer = trueAnswer;
    this.falseAnswer = falseAnswer;
    this.trueQuest = trueQuest;
    this.falseQuest = falseQuest;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public Integer getQuestLevel() {
    return questLevel;
}

public void setQuestLevel(Integer questLevel) {
    this.questLevel = questLevel;
}

public String getQuestion() {
    return question;
}

public void setQuestion(String question) {
    this.question = question;
}

public String getVideoPath() {
    return videoPath;
}

public void setVideoPath(String videoPath) {
    this.videoPath = videoPath;
}

public Answer getTrueAnswer() {
    return trueAnswer;
}

public void setTrueAnswer(Answer trueAnswer) {
    this.trueAnswer = trueAnswer;
}

public Answer getFalseAnswer() {
    return falseAnswer;
}

public void setFalseAnswer(Answer falseAnswer) {
    this.falseAnswer = falseAnswer;
}

public Integer getTrueQuest() {
    return trueQuest;
}

public void setTrueQuest(Integer trueQuest) {
    this.trueQuest = trueQuest;
}

public Integer getFalseQuest() {
```

```
        return falseQuest;
    }

    public void setFalseQuest(Integer falseQuest) {
        this.falseQuest = falseQuest;
    }

    @Override
    public String toString() {
        return "Quest{" +
                "id=" + id +
                ", questLevel=" + questLevel +
                ", question='" + question + '\'' +
                ", videoPath='" + videoPath + '\'' +
                ", trueAnswer=" + trueAnswer +
                ", falseAnswer=" + falseAnswer +
                ", trueQuest=" + trueQuest +
                ", falseQuest=" + falseQuest +
                '}';
    }
}
```

В.6 Анімація

```
package sample.Animation;

import com.jfoenix.controls.JFXButton;
import javafx.animation.FadeTransition;
import javafx.util.Duration;

public class FadeTransitionAnimation {

    private static FadeTransition fadeTransition;

    public static void singleBtnFadeTransition(JFXButton chooseBtn, double value, boolean setBoolean) {
        fadeTransition = new FadeTransition(Duration.millis(500), chooseBtn);
        fadeTransition.setFromValue(value);
        fadeTransition.setToValue(1.0);
        fadeTransition.setCycleCount(1);
        fadeTransition.setAutoReverse(setBoolean);
    }

    public static void setFadeTransition(JFXButton chooseFirstBtn, JFXButton chooseSecondBtn, double value) {
        fadeTransition = new FadeTransition();
        fadeTransition.setDuration(Duration.millis(500));
        fadeTransition.setFromValue(value);
        fadeTransition.setToValue(1.0);
        fadeTransition.setCycleCount(1);
        fadeTransition.setAutoReverse(true);
        if (chooseFirstBtn.isHover()) {
            fadeTransition.setNode(chooseFirstBtn);
        } else if (chooseSecondBtn.isHover()) {
            fadeTransition.setNode(chooseSecondBtn);
        }
        fadeTransition.play();
    }

}
```

В.7 Лістинг коду FXML файлів

В.7.1 Вікно Ініціалізації

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXTextField?>
<?import com.jfoenix.controls.JFXToggleButton?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.paint.LinearGradient?>
<?import javafx.scene.paint.Stop?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="500.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.StartWindow.AuthorizationController">
  <children>
    <Pane prefHeight="500.0" prefWidth="800.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
      <children>
        <ImageView fitHeight="507.0" fitWidth="800.0" pickOnBounds="true" preserveRatio="true">
          <image>
            <Image url="@../image/dna_black_and_white.jpg" />
          </image>
        </ImageView>
        <Pane layoutX="237.0" prefHeight="500.0" prefWidth="326.0">
          <children>
            <Label alignment="CENTER" contentDisplay="TOP" layoutX="63.0" layoutY="68.0" text="MED" textFill="WHITE" underline="true">
              <font>
                <Font name="Monospaced Bold" size="48.0" />
              </font>
            </Label>
            <Label layoutX="152.0" layoutY="73.0" text="QUEST" textFill="#4d4d4d">
              <font>
                <Font name="Impact" size="44.0" />
              </font>
            </Label>
            <JFXTextField fx:id="name" focusColor="#ad1457" labelFloat="true" layoutX="57.0" layoutY="208.0" prefHeight="34.0" prefWidth="213.0" promptText="Ім'я/Прізвище">
              <font>
                <Font name="Franklin Gothic Medium" size="18.0" />
              </font>
            </JFXTextField>
            <JFXTextField fx:id="group" focusColor="#ad1457" labelFloat="true" layoutX="57.0" layoutY="258.0" prefHeight="34.0" prefWidth="213.0" promptText="Група">
              <font>
                <Font name="Franklin Gothic Medium" size="18.0" />
              </font>
            </JFXTextField>
            <JFXButton fx:id="signInBtn" buttonType="RAISED" layoutX="115.0" layoutY="334.0" onAction="#switchScenes" onMouseEntered="#onMouseEntered" onMouseExited="#onMouseExited" prefHeight="35.0" prefWidth="96.0" styleClass="signInBtn" stylesheets="@../Style.css" text="Увійти" textFill="WHITE">
              <ripplerFill>
                <LinearGradient endX="1.0" endY="1.0">
                  <stops>
                    <Stop color="#878787" />
                    <Stop color="#acacac" offset="1.0" />
                  </stops>
                </LinearGradient>
              </ripplerFill>
              <font>
                <Font name="Franklin Gothic Medium" size="20.0" />
```

```
            </font>
          </JFXButton>
          <Label layoutX="76.0" layoutY="407.0" text="Switch to English" textFill="#4d4d4d">
            <font>
              <Font name="Impact" size="14.0" />
            </font>
          </Label>
          <JFXToggleButton fx:id="switchLanguage" layoutX="164.0" layoutY="393.0"
onAction="#switchToAntherLang" prefHeight="44.0" prefWidth="86.0" size="7.0" text="ENG" textFill="#4d4d4d"
toggleColor="#a4004a" toggleLineColor="#d56f9f">
            <font>
              <Font name="Impact" size="14.0" />
            </font>
          </JFXToggleButton>
        </children>
      </Pane>
    </children>
  </Pane>
 </children>
</AnchorPane>
```

В.7.2 Вікно Привітання

```
<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.shape.Line?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="600.0" prefWidth="1000.0" styleClass="root" stylesheets="@../Style.css"
xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.WelcomeWindow.WelcomeClass">
  <children>
    <Label alignment="CENTER" layoutX="58.0" layoutY="8.0" stylesheets="@../Style.css" text="QUEST"
textFill="#17252a">
      <font>
        <Font name="Impact" size="24.0" />
      </font>
    </Label>
    <Label alignment="CENTER" layoutX="14.0" layoutY="7.0" stylesheets="@../Style.css" text="MED"
textFill="WHITE" underline="true">
      <font>
        <Font name="Monospaced Bold" size="24.0" />
      </font>
    </Label>
    <VBox alignment="CENTER" layoutX="100.0" layoutY="120.0" prefHeight="332.0" prefWidth="783.0"
styleClass="welcomeVBox" stylesheets="@../Style.css" AnchorPane.leftAnchor="100.0"
AnchorPane.rightAnchor="100.0">
      <children>
        <Label fx:id="welcomeHeader" alignment="TOP_CENTER" stylesheets="@../Style.css" text="Вітаємо тебе
дорогий друже!" textFill="#17252a">
          <font>
            <Font name="Monospaced Bold" size="36.0" />
          </font>
        </Label>
        <Label fx:id="welcomeText_1" alignment="CENTER" prefHeight="106.0" prefWidth="783.0"
stylesheets="@../Style.css" text="Пропонуємо тобі перевірити свої знання з дисципліни &amp;quot;Дитячі інфекційні
хвороби&amp;quot; та допомогти перемогти хворобу за допомогою набутих тобою знань." textFill="WHITE"
wrapText="true">
```

```
            <font>
              <Font name="FontAwesome Regular" size="22.0" />
            </font>
            <VBox.margin>
              <Insets bottom="10.0" top="10.0" />
            </VBox.margin>
          </Label>
          <Line endX="-150.96205139160156" endY="-0.20619644224643707" fill="BLACK" opacity="0.5" startX="-
432.4787292480469" startY="0.5928627252578735" stroke="#17252a" strokeLineCap="ROUND" strokeWidth="2.0">
            <VBox.margin>
              <Insets left="-50.0" />
            </VBox.margin>
          </Line>
          <Label fx:id="welcomeText_2" alignment="CENTER" prefHeight="139.0" prefWidth="787.0"
stylesheets="@../Style.css" text="Використовуючи навички та вміння, користуючись симптомами, обери вірні кроки
для подолання стадій хвороби, обираючи вірні засоби та методи лікування. Але пам'ятай - хибні методи призведуть
до трагедії! Хай щастить!" textFill="WHITE" wrapText="true">
            <font>
              <Font name="FontAwesome Regular" size="22.0" />
            </font>
          </Label>
        </children>
        <padding>
          <Insets left="20.0" />
        </padding>
      </VBox>
      <AnchorPane layoutX="361.0" layoutY="476.0" prefHeight="90.0" prefWidth="278.0">
        <children>
          <JFXButton fx:id="endBtn" buttonType="RAISED" layoutX="14.0" layoutY="29.0"
onAction="#exitTheApplication" onMouseEntered="#WelcomeOnMouseEntered"
onMouseExited="#WelcomeOnMouseExited" styleClass="welcome-jfx-button" text="Выйти" />
          <JFXButton fx:id="startBtn" alignment="TOP_LEFT" buttonType="RAISED" layoutX="186.0" layoutY="30.0"
onAction="#goToQuest" onMouseEntered="#WelcomeOnMouseEntered" onMouseExited="#WelcomeOnMouseExited"
styleClass="welcome-jfx-button" stylesheets="@../Style.css" text="Почати" />
        </children>
      </AnchorPane>
    </children>
</AnchorPane>
```

В.7.3 Вікно Проходження тесту

```
<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>
<?import java.lang.String?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.effect.DropShadow?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.media.MediaView?>
<?import javafx.scene.shape.Line?>
<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="800.0" prefWidth="1000.0" styleClass="questRoot" stylesheets="@../Style.css"
xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.QuestWindow.QuestWindowClass">
  <children>
    <Label layoutX="14.0" layoutY="11.0" text="MED" textFill="#17252a" underline="true">
      <font>
```

```
        <Font name="Monospaced Bold" size="24.0" />
      </font>
    </Label>
    <Label layoutX="59.0" layoutY="12.0" text="QUEST" textFill="#2c9790">
      <font>
        <Font name="Impact" size="24.0" />
      </font>
    </Label>
    <AnchorPane layoutX="-9.0" layoutY="54.0" stylesheets="@../Style.css" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="65.0">
      <children>
        <VBox fx:id="videoResizePane" alignment="TOP_CENTER" layoutX="-3.0" layoutY="-1.0" prefHeight="386.0" prefWidth="1000.0" AnchorPane.bottomAnchor="349.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
          <children>
            <MediaView fx:id="mvObject" fitHeight="350.0" fitWidth="700.0">
              <VBox.margin>
                <Insets top="18.0" />
              </VBox.margin>
            </MediaView>
          </children>
        </VBox>
      </children>
      <styleClass>
        <String fx:value="roundCornerPane" />
        <String fx:value="roundCornerPane_1" />
      </styleClass>
      <effect>
        <DropShadow blurType="GAUSSIAN" color="#c2c2c2" height="25.0" radius="11.0" />
      </effect>
    </AnchorPane>
    <Pane fx:id="questionPane" layoutY="452.0" prefHeight="348.0" prefWidth="1000.0" stylesheets="@../Style.css" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="452.0">
      <children>
        <VBox alignment="CENTER" layoutX="135.0" layoutY="21.0" prefHeight="109.0" prefWidth="730.0">
          <children>
            <Label fx:id="questionText" alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="100.0" prefWidth="730.0" styleClass="questText" stylesheets="@../Style.css" text="- Привіт! Я – Ганнуся, та щось мені недобре. В мене другий день лихоманка, ще й постійний кашель. А як горло болить, ууу… Я захворіла, так?" textAlignment="CENTER" textFill="WHITE" wrapText="true">
              <font>
                <Font name="FontAwesome Regular" size="18.0" />
              </font>
              <VBox.margin>
                <Insets />
              </VBox.margin>
            </Label>
          </children>
        </VBox>
        <Line endX="100.0" layoutX="528.0" layoutY="145.0" opacity="0.5" startX="-157.80001831054688" startY="-6.103515625E-5" strokeLineCap="ROUND" strokeLineJoin="ROUND" strokeWidth="1.5" />
      </children>
      <styleClass>
        <String fx:value="roundCornerPane_2" />
        <String fx:value="roundCornerPane" />
      </styleClass>
      <effect>
        <DropShadow color="#717171" />
      </effect>
    </Pane>
    <Pane fx:id="answerPane" layoutY="627.0" prefHeight="173.0" prefWidth="1000.0" stylesheets="@../Style.css"
```

```
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="627.0">
      <children>
        <JFXButton fx:id="agreementBtn" alignment="CENTER" buttonType="RAISED" layoutX="56.0"
layoutY="66.0" onAction="#trueAnswer" onMouseEntered="#QuestOnMouseEntered"
onMouseExited="#QuestOnMouseExited" prefHeight="39.0" prefWidth="259.0" styleClass="quest-jfx-button"
stylesheets="@../Style.css" text="Схоже, що в тебе кір." wrapText="true" />
        <JFXButton fx:id="disagreementBtn_1" alignment="CENTER" buttonType="RAISED" layoutX="689.0"
layoutY="65.0" onAction="#wrongAnswer" onMouseEntered="#QuestOnMouseEntered"
onMouseExited="#QuestOnMouseExited" prefHeight="13.0" prefWidth="259.0" styleClass="quest-jfx-button"
stylesheets="@../Style.css" text="Може, ти з'їла щось не те?" textAlignment="CENTER" wrapText="true" />
        <JFXButton fx:id="disagreementBtn" alignment="CENTER" buttonType="RAISED" layoutX="374.0"
layoutY="65.0" onAction="#wrongAnswer" onMouseEntered="#QuestOnMouseEntered"
onMouseExited="#QuestOnMouseExited" prefHeight="36.0" prefWidth="259.0" styleClass="quest-jfx-button"
stylesheets="@../Style.css" text="Мабуть, це звичайне ОРВІ." wrapText="true" />
      </children>
      <styleClass>
        <String fx:value="roundCornerPane" />
        <String fx:value="roundCornerPane_3" />
      </styleClass>
      <effect>
        <DropShadow color="#575757" />
      </effect>
    </Pane>
  </children>
</AnchorPane>
```

В.8 Каскадна таблиця стилів

```
.root{
   -fx-background-color: #3aafa9;
}

.signInBtn{
   -fx-background-radius: 100px;
   -fx-border-radius: 100px;
   -fx-border-width: 3px;
}

.signInBtn:hover{
   -fx-border-color: white;
   -fx-font-family: "ADAM.CG PRO";
   -fx-text-fill: #424242;
   -fx-font-size: 14px;
   -fx-font-weight: bold;
}

.signInBtn:pressed{
   -fx-font-family: "Arial Rounded MT Bold";
   -fx-text-fill: white;
   -fx-font-size: 14px;
}

.logoWelcomeHB{
   -fx-font-weight: bold;
}

.welcomeVBox{
   -fx-border-color: white;
   -fx-border-width: 1px;
   -fx-border-radius: 10px;
   -fx-border-style: solid none none none;
```

```
}

.welcomeTextVBox{
    -fx-text-fill: #17252a;
    -fx-font-weight: bold;
}

.welcomeTextDescription{
    -fx-font-weight: bold;
}

.welcome-jfx-button {
    -jfx-button-type: RAISED;
    -fx-text-fill: white;
    -fx-font-size: 18px;
    -fx-font-family: "Lato Semibold";
    -fx-font-weight: bold;
    -fx-background-color: #17252a;
    -fx-padding: 8px 10px 8px 10px;
    -jfx-disable-visual-focus: true;
}

.welcome-jfx-button:hover{
    -fx-background-color: transparent;
    -fx-background-radius: 100px;
    -fx-border-radius: 100px;
    -fx-border-color: #17252a;
    -fx-border-width: 3px;
}

.welcome-jfx-button:pressed{
    -fx-background-color: transparent;
    -fx-background-radius: 100px;
    -fx-border-radius: 100px;
    -fx-border-color: transparent;
    -fx-border-width: 3px;
}

.questRoot {
    -fx-background-color: white;
}

.roundCornerPane {
    -fx-background-radius: 40px 40px 0 0;
    -fx-border-radius: 40px 40px 0 0;
}

.roundCornerPane_1 {
    -fx-background-color: white;
}

.roundCornerPane_2 {
    -fx-background-color: #3AAFA9;
    -fx-text-fill: #FFFFFF;
}

.roundCornerPane_3 {
    -fx-background-color: #17252a;
}

.questText {
    -fx-font-weight: bold;
}
```

```
.quest-jfx-button {
    -jfx-button-type: RAISED;
    -fx-text-fill: #17252a;
    -fx-font-size: 18px;
    -fx-font-family: "Lato Semibold";
    -fx-font-weight: bold;
    -fx-background-color: #E0E0E0;
    -fx-padding: 8px 10px 8px 10px;
    -jfx-disable-visual-focus: true;
}

.quest-jfx-button:hover {
    -fx-background-radius: 100px;
    -fx-border-radius: 100px;
}

.commonMediaClass {
    -fx-background-color: #BDBDBD;
    -fx-border-style: none;
    -fx-pref-width: 35px;
    -fx-pref-height: 35px;
}

.commonMediaClass:hover {
    -fx-background-radius: 100px;
    -fx-border-radius: 100px;
}
```