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: Ігровий додаток «Квест-гра для вивчення історії України учнями 5-го класу»

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

Виконавець роботи: студент групи ІТ-53-7 Хвайра Таєр Самер Тавфік

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «__» _____2019 р.

Науковий керівник _____ к.т.н., доц., Алексенко О.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-2019

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
«__» _____ 2019 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Хвайра Таєр Самер Тавфік

1 Тема роботи Ігровий додаток «Квест-гра для вивчення історії України учнями 5-го класу»

керівник роботи *Алексенко Ольга Василівна, к.т.н., доцент*,

затверджені наказом по університету від « 17 » травня 2019 р. № 084-III

2 Строк подання студентом роботи «3» червня 2019 р.

3 Вхідні дані до роботи_____
_____
_____
_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____
_____
_____
_____

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

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка

Студент _____ Хвайра Т.С.Т.

(підпис)

Керівник роботи _____ к.т.н., доц. Алексенко О.В.

(підпис)

РЕФЕРАТ

Тема бакалаврської роботи: Ігровий додаток «Квест гра для вивчення історії України учнями 5-го класу».

Пояснювальна записка містить вступ, 4 розділи, висновки, додатки та список використаних джерел, включає 129 сторінок, 8 таблиць, 38 рисунків, 26 джерел.

В першому розділі визначається актуальність дипломного проекту. Розглядаються сучасні методи інтерактивного навчання історії України, їх переваги та недоліки. Обґрунтовується актуальність роботи.

У другому розділі визначається мета роботи, встановлюються задачі роботи та виконується вибір методів дослідження. Проводиться планування робіт по створенню додатку. Розроблене технічне завдання, яке наведено у додатку А. Результати з планування робіт наведені у додатку Б.

У третьому розділі демонструється повне, поетапне проектування програмного додатку, створення моделей та діаграм, розробка архітектури додатку. Також наводяться функціональні та нефункціональні можливості додатку.

Четвертий розділ описує процес розробки програмного додатку. Показує практичне застосування створених моделей та діаграм. Детально наводяться етапи розробки та демонструються функціональні можливості додатку. Результати розробки у вигляді програмних кодів наведені у додатках.

Результатом дипломного проектування є програмний додаток «Квест-гра для вивчення історії України учнями 5-го класу».

Ключові слова: програмний додаток, розробка, методи інтерактивного навчання, паттерн MVVM.

ЗМІСТ

ВСТУП

Впровадженню мультимедійних систем у навчання учнів на даний момент приділяється велика увага. Основні теоретичні відомості про користь гейміфікованого навчання подані у офіційних документах МОН [1]. Саме через це, питання вдосконалення методів викладання стоїть на першому місці у Міністерстві науки та освіти України.

Актуальність дипломної роботи зумовлена, переходом світу на інтерактивне навчання учнів, де необхідні принципово нові форми навчання. Існує необхідність заміни застарілих та неефективних методів викладання, які значно затримують розвиток освіти.

Предметна область проекту Історія України, через те що саме історія є одною з головних наук сьогодення. На часі використання мультимедійних систем дуже лімітоване у процесі навчання. Саме тому інтерактивне вивчення Історії України учнями 5-го класу є найбільш привабливою, у порівнянні з застарілими формами, і забезпечує зацікавленість учнів у вивченні Історії України.

Предмет дослідження – інтерактивні методи вивчення Історії України учнями 5-го класу.

Об’єкт дослідження –методи навчання у школі.

Мета дипломної роботи: розробка гейміфікованого додатку «Квест-гра для вивчення Історії України учнями 5-го класу» для підвищення зацікавленості учнів та рівня засвоєння навчальних матеріалів.

Для досягнення мети дипломної роботи був визначений перелік завдань:

– розглянути основні методи інтерактивного навчання сьогодення, та проаналізувати їх особливості;

– проаналізувати методи інтерактивного навчання, які використовується у Сумській спеціалізованій школі №10;

– на основі даних аналізу спроектувати програмний додаток для вивчення Історії України учнями 5-го класу;

– на основі створеного проекту, розробити гейміфікований додаток «Квест-гра для вивчення Історії України учнями 5-го класу».

Методи дослідження:

– аналіз та узагальнення досвіду викладання Історії України у Сумській спеціалізованій школі №10;

– моделювання (створення структурно-функціональної моделі вивчення Історії України для розуміння процесу навчання; створення моделей програмного додатку як сукупності діаграми варіантів використання, діаграми класів програми; створення моделі «сутність – зв'язок» для розуміння відносин даних між собою).

Практичне значення полягає у розробці гейміфікованого додатку «Квест-гра для вивчення Історії України учнями 5-го класу» для підвищення зацікавленості учнів 5-го класу у вивченні Історії України**.**

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний аналіз стану викладання Історії України у Сумській спеціалізованій школі №10

У даний момент організація навчального процесу включає в себе частіше традиційні методи навчання, аніж інтерактивні або гейміфіковані. Але життя у 21 сторіччі диктує свої правила, через що традиційні методи втрачають свою ефективність.

Не інтерактивні методи навчання представлені уже готовими знаннями, а тому мають великий недолік: знання здобуті таким методом швидко забуваються і не пов’язуються із застосуванням їх у майбутньому.

До основних методів традиційного навчання відносяться:

– розповідь;

– бесіда;

– лекція;

– наочні методи навчання.

Всі ці методи, нажаль, не відповідають потребам та інтересам учнів сьогодення, тому необхідність збільшення відсотку використання інтерактивних методів неминуче.

Прикладною областю дослідження стала Сумська спеціалізована школа № 10, де, у результаті зменшення зацікавленості учнів і, як наслідок зменшення відсотку успішності, виникла необхідність створення програмного додатку для інтерактивного навчання учнів 5-го класу.

Зараз процент використання інтерактивних методів дуже малий. Основним методом у даний час є використання мультимедійних систем для наочного подання знань, що не є достатнім для підняття зацікавленості учнів.

Так склалося що життя сучасного учня тісно пов’язано з комп’ютерами та з іграми, саме тому актуальною є задача створення програмного продукту для інтерактивного гейміфікованого навчання.

1.2 Огляд існуючих інтерактивних методів викладання Історії України

Зараз існує велика кількість інтерактивних методів викладання, які використовуються у школах не тільки України а й Америки, Франції та інших. Ці методи були створені дуже давно, а тому вони стрімко втрачають свою ефективність.

Нижче будуть розглянуті основні та самі популярні методи інтерактивного навчання, які навіть зараз є ефективними та використовуються у багатьох навчальних закладах світу [2].

Мозковий штурм

Цей метод показує високий рівень ефективності, але може використовуватись лише у специфічних ситуаціях. Метод представляє собою великий діалог, коли кожен учасник розмови видає велику кількість ідей рішення поставленої задачі. Його використання значно підвищує активність учнів на урочі, та примушує їх включатися у роботу у класі. Учні отримують можливість продемонструвати свої знання, та показати свої думки. В той же час це допомагає учням коротко і зрозуміло пояснювати свої думки.

Використання методу мозкового штурму має велику кількість переваг, таких як:

– розвиток креативності та творчості учнів;

– розвиток комунікативних навиків;

– розвиток уявлення та фантазії;

– розвиток сприйняття конструктивної критики інших учнів.

Але метод має і низку недоліків, таких як:

– некерованість процесу;

– неможливість використання у комплексних, складних запитаннях.

Використання методу мозкового штурму дозволяє показати учням, що у кожної проблеми існує декілька варіантів вирішення. І все залежить лише від ситуації та поставлених умов. А вміння висказувати свої думки сприяє розвитку креативності та інтелектуальності дітей.

Метод кластерного вивчення

Один з інтерактивних методів вивчення, який представляє собою групування інформації у кластери (групи) за певними ознаками. Після формування кластерів вони складаються у схематичне дерево. Воно являє собою зображення, що сприяє систематизації та узагальненню вивченого матеріалу.

У такого методу є велика кількість переваг, наприклад:

– може вмістити в себе велику кількість інформації;

– розвиває вміння виділяти головне із великої кількості інформації;

– розвиває навички аналізу та порівняння;

– розвиває аналітичне мислення.

Одним з основних недоліків такого методу є неможливість використання у ситуаціях коли виділити головну ідею неможливо.

Використання кластерного методу доцільне у школярів старшої школи, це допоможе їм систематизувати отриманні знання. Навчить учнів виробляти та висловлювати своє власне судження по темі. Також розвине навики розглядання одразу декількох задач, та тем.

Метод круглого столу

Цей метод подібний до методу мозкового штурму і включає в себе усі представлені вище переваги, але виключає деякі недоліки. Метод круглого столу представляє собою обговорення однієї ідеї рішення поставленої задачі у групі.

Метод має такі переваги:

– розвиває навички роботи у команді;

– розвиває навички креативного та творчого мислення.

Та у порівнянні з методом мозкового штурму не має недоліків окрім неможливості використання у специфічних ситуаціях, таких як вирішення комплексних питань.

Використання методу круглого столу є доцільним при розгляді простих та зрозумілих питань, щоб кожен учень міг висловити свою думку щодо поставленої задачі. В інших випадках використання цього метода є недоцільним.

Інтерактивний урок з використанням інформаційно-комп'ютерних технологій

Один з найпоширеніших методів інтерактивного навчання. В основі цього методу лежить використання ІКТ, таких як проектор, комп'ютер. Метод представляє собою демонстрацію навчального матеріалу у графічній форму (презентація, фільм, тощо). Такий метод дуже гарно показує себе у реаліях сьогодення і дуже часто використовується у школах України. Ефективність методу зумовлена зацікавленістю учнів у перегляді ілюстрованого матеріалу.

На основі цього методу можна побудувати велику кількість видів діяльності, наприклад:

– перегляд рефератів;

– розробку та проводження тестових занять;

– підготовку до конкурсів, турнірів, олімпіад;

– роботу з електронними книгами;

– роботу з картами та атласами;

– перегляд історичних джерел;

– перегляд історичних фільмів;

– використання презентацій для проведення уроків.

Метод має низку переваг:

– учні сприймають інформацію не лише як сухе джерело, а підкріплюють її графічним відображенням;

– підвищення зацікавленості учнів;

– зміцнення знань вивчених на уроці;

– підвищення рівня успішності учнів.

Метод гейміфікації навчання

Один з нових, але самий перспективний метод навчання, який включає у себе усі методи названі раніше, але представляє собою зовсім інший вид навчання. Цей метод має велику підтримку Міністерства науки і освіти України на даний час, тому саме він був обраний для дослідження і розробки програмного продукту.

У основі методу лежить дуже просте поняття – учень повинен виконувати завдання для досягнення мети, і у результаті отримати інформацію потрібну для просування по уроку.

Метод включає усі переваги інших методів, але можна виділити основні саме для цього методу:

– великий відсоток зацікавленості учнів;

– великі показники успішності;

– великий показник вивченого матеріалу.

Переваги методу очевидні для кожного, саме тому цей метод користується такою популярністю.

Усі інтерактивні методи навчання створені для вирішення головного завдання – навчити дитину вчитися. Тобто знання не повинні подаватися у готову вигляді, а учень повинен прикласти деякі зусилля для здобуття інформації. Набагато важливіше розвивати критичне мислення, засноване на аналізі ситуації, самостійному пошуку інформації, побудови логічного ланцюжка і прийняття зваженого і аргументованого рішення. Саме тому гейміфікація інтерактивних методів вважається дуже перспективною у реаліях сьогодення.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі

За результатами аналізу предметної області була сформульована мета дипломної роботи: розробка гейміфікованого додатку «Квест-гра для вивчення Історії України учнями 5-го класу» для підвищення зацікавленості учнів та рівня засвоєння навчальних матеріалів.

Для досягнення мети дипломної роботи був визначений перелік завдань:

− дослідити процес вивчення історії в школі, в результаті чого сформулювати технічне завдання на розробку

− обрати методи та інструменти розробки

− спроектувати додаток, у т.ч. сформувати схему бази даних та розробити моделі програмного продукту

− реалізувати та протестувати додаток

Програмний додаток повинен забезпечувати автоматизацію роботи вчителів, а саме здійснювати перевірку контрольних та практичних роботі, додати інтерактивної взаємодії у процесі навчання учнів 5-го класу, та підвищити рівень засвоюваності навчального матеріалу.

Використання програмного продукту спрямоване на підвищення рівня зацікавленості учнів, підвищення якості та показників успішності учнів. Також одною із частин мети є автоматизація перевірки контрольних та практичних робіт, а це значить зниження кількості роботи яку виконують вчителі.

Впровадження додатку «Квест-гра для навчання» передбачає такі функції:

– вибір теми;

– проходження інтерактивних уроків;

– виконання практичних робіт по пройденим темам;

– виконання контрольних робіт по пройденим темам;

– перегляд оцінок;

– автоматичне оцінювання практичних і контрольних робіт.

Також було розроблене технічне завдання, представлене у додатку А.

2.2 Методи дослідження

Для досягнення поставлених задач було обрано два основні методи дослідження:

– аналізу – метод дослідження який вивчає предмет пізнання, а саме усі його властивості, ознаки та відношення окремо один від одного. Цей метод буде використаний для аналізу предметної області, для більш глибинного розуміння існуючих методів інтерактивного навчання, їх переваг та недоліків;

– моделювання – метод дослідження об’єкту, явищ або процесів, що ґрунтується на заміні оригіналу дослідження на подібну модель. За допомогою цього методу будуть створенні моделі: IDEF0 [3] – для розуміння предметної області, недоліків та переваг існуючого процесу вивчення Історії України учнями 5-го класу; UML діаграм [4], а саме діаграма варіантів використання (Use Case [5]) – для розуміння розмежування доступу користувачів до програмного додатку, та можливості використання певними групами користувачів обмеженою кількістю функцій, діаграма класів (Class diagram [6]) – для розуміння класів, типів даних та відношень між класами, ER-діаграму [7] – для розуміння структури бази даних та відношень між базами та таблицями.

Також було виконане планування робіт, створений календарний план та виконана оцінка ризиків які знаходяться у додатку Б.

2.3 Вибір засобів реалізації

Для реалізації програмного додатку можливе використання великої кількісті мов програмування, СУБД та фреймворків.

У якості мови програмування було обрано три основні:

– Java;

– C++;

– C#.

Java – об’єктно-орієнтована мова програмування, створена у 1995 році компанією Sun Microsystems. Синтаксис даної мови значно запозичений з C та C++.Є суворо типізованою мовою програмування[8]. Має підтримку таких парадигм:

– імперативна;

– структурна;

– об'єктно-орієнтована.

Основним недоліком є компілювання та запуск програми за допомогою віртуальної машини, що впливає на швидкість роботи.

C++ – мова програмування високого рівня розроблена у 1979 році та базується на мові програмування C. Синтаксис даної мови значно запозичений з C. Є суворо типізованою мовою програмування[9]. Має підтримку таких парадигм:

– процедурна;

– об'єктно-орієнтована;

– узагальнена.

C# – об’єктно-орієнтована мова програмування, яка відноситься до сі-подібних мов. Синтаксис цієї мови найбільше схожий на синтаксис мов C++ та Java, тому що вона була створена під впливом цих двох мов і ввібрала в себе все найкраще від них, та позбулася майже усіх недоліків своїх попередників. Має строгу статичну типізацію та була розроблена для програмування на платформі .Net [10].

Саме тому вона була обрана для створення програмного продукту.

Через те що у якості мови програмування було обрано C# можна виділити два основних можливих середовища розробки:

– JetBrains Rider

– Visual Studio

JetBrains Rider – це кросплатформене інтегроване середовище розробки для розробки на платформі .Net [11]. Має велику кількість переваг, таких як:

– велика кількість плагінів, таких як Git та Mercurial;

– автоматичне доповнення коду;

– кросплатформеність.

Головним недоліком даної середи розробки є те що вона поширюється на платній основі.

Як вказано у [12] Visual Studio – це кросплатформене інтегроване середовище для розробки на багатьох мовах програмування.

Має велику кількість переваг, таких як:

– велика кількість плагінів;

– вбудовану систему контролю версій;

– автоматичне доповнення коду;

– безкоштовність;

– офіційна платформа для розробки під .Net;

– велика спільнота розробників;

– доступність ресурсу документації та допомоги Microsoft Docs;

– кросплатформеність.

Найголовнішою перевагою цього середовища є вбудована можливість будувати UML діаграми.

Через це середою розробки була обрана Visual Studio.

Враховуючи поставлені вище задачі та попередній досвід оптимальним засобом для розробки даного продукту є застосування мови програмування C# у середовищі програмування Visual Studio 2013 з використанням фреймворку .Net [13,

інструментарію MVVM Light Toolkit [14] для паттерну MVVM, бібліотеки SQLite [15].

Visual Studio – це середа розробки програмних засобів, таких як програмних застосунків, веб додатків, API та мобільних додатків. Використовує платформи розробки компанії Microsoft, такі як Microsoft Silverlight, Windows Forms, WPF, Магазин Windows та Windows API. Включає в себе редактор коду, засоби тестування, засоби для створення схем та моделей. Підтримує 36 мов програмування в тому числі і C#.

.Net – платформа розробки створена компанією Microsoft. Використовується для розробки програмних додатків для Microsoft Windows. Платформа включає в себе бібліотеку класів іменовану як Framework Class Library (FCL) [16] і забезпечує взаємодію мов програмування (кожна мова може використовувати код, написаний на інших мовах). Програми, написані з використанням .NET Framework, виконуються в програмному середовищі Common Language Runtime (CLR) [17]. CLR – це віртуальна машина, яка виконує управління пам'яттю та обробкою виключень. Таким чином FCL і CLR разом утворюють .NET Framework.

MVVM Light Toolkit – це інструментарій для використання одного з самих відомих паттернів програмування MVVM. Основною метою інструментарію є прискорення створення та розробки додатків з використанням паттерну MVVM у Windows Universal, WPF [18], Silverlight, Xamarin.iOS, Xamarin.Android та Xamarin.Forms. MVVM Light Toolkit допомагає відокремити View (сукупність вікон та форм) від Model (функціональний код додатку), що допоможе створювати додатки, які є більш чистими та простішими в обслуговуванні. Вона також створює програми з можливістю тестування і дозволяє створювати набагато більш тонкий шар інтерфейсу користувача (який важче тестувати автоматично).

SQLite – це бібліотека створена на мові C, яка реалізує невеликий, швидкий, автономний, високонадійний, повнофункціональний засіб для використання баз даних SQL. SQLite є найбільш часто використовуваним механізмом для використання баз даних у світі. SQLite вбудований у всі мобільні телефони та

більшість комп'ютерів і поставляється в безлічі інших програм, які люди використовують щодня.

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Структурно-функціональне моделювання процесу вивчення історії України учнями 5-го класу

Функціональна модель IDEF0 існує для відображення функціювання системи, управління системою та функціонального потоку процесів життєвого циклу. Діаграма IDEF0 здатна графічно представити широкий спектр ділових, виробничих та інших видів діяльності підприємства до будь-якого рівня деталізації. Вона забезпечує строгий і точний опис процесу виробництва і сприяє послідовності використання і інтерпретації.

Для створення структурно-функціональної моделі був використаний програмний додаток Erwin Process Modeler.

Розглянемо процес вивчення учнями 5-го класу історії України у Сумській спеціалізованій школі №10. Для відображення, як бізнес-процес структурований в цей час, було використано модель робіт AS-IS (Як є). Дана модель використовується для встановлення «відправної точки», необхідної для розробки програми. А саме для розуміння процесів та методів вивчення історії України у даний момент часу, їх вдосконалення та спрощення.

Контекстна діаграма є вершиною всієї структури діаграм в ході структурно-функціонального моделювання і є найбільш загальним описом досліджуваних процесів та їх взаємодії з навколишнім середовищем.

Контекстну діаграму процесу вивчення історії учнями 5-го класу представлено на рис. 3.1.

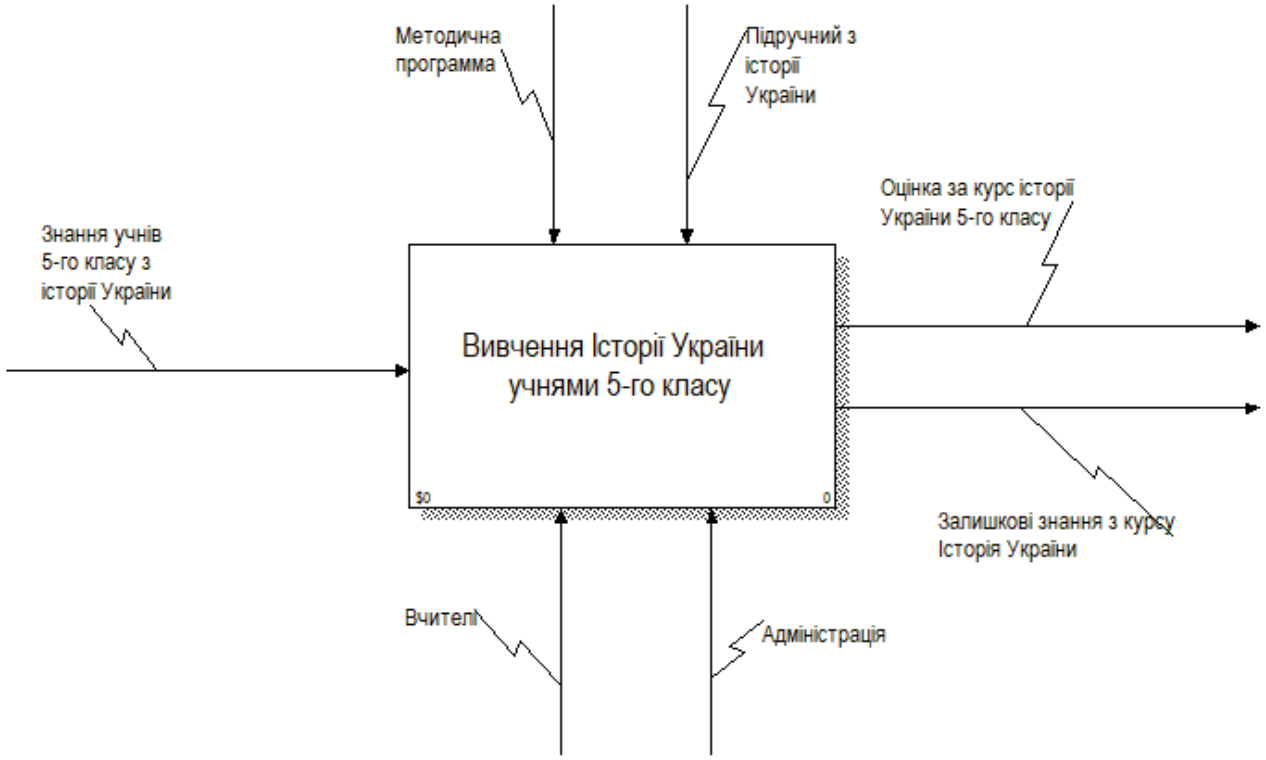


Рисунок 3.1 – Контекстна діаграма

Контекстна діаграма складається з одного блоку, який максимально чітко описує процес або систему що моделюється, та стрілок або дуг. Вхідна стрілка представляє собою вхідні дані для перетворення системою. Вихідна стрілка представляє собою фінальний результат після закінчення досліджуваного процесу або системи. Стрілки що входять у блок зверху відображають інструменти, які використовуються для перетворення вхідних даних. Стрілки що входять знизу відображають механізми, які сприяють перетворенню вхідних даних [19].

Вхідною стрілкою до функції «Вивчення Історії України учнями 5-го класу» є «Знання учнів 5-го класу з історії України; вихідними – «Оцінка за курс історії України 5-го класу» та «Залишкові знання учнів з курсу історії України»; стрілками контролю – «Методична програма» та «Підручник з історії України»; стрілками механізмів – «Вчителі» та «Адміністрація».

Після опису головної функції зазвичай виконується її розбиття на фрагменти. Процес розбивання контекстної діаграми на блоки називають декомпозицією

діаграми на функціональні блоки. Вони описують кожен фрагмент та взаємодію між фрагментами і називаються діаграмами декомпозиції.

Відповідно до [20] діаграма декомпозиції складається з блоків діяльності, що відображають функцію, та стрілок, що представляють собою ресурси. Діяльність – це процес, функція або задача результатом якої є досягнення мети проекту. Блок діяльності зображується прямокутником, у середині якого зазначається назва функції та його номер. Блоки зазвичай розташовуються у порядку зменшення важливості. У верхньому лівому куті розташовують найбільш важливий блок, а у нижньому правому – найменш. Таким чином, структура блоків показує, які функції впливають на інші. Блоки у діаграмі нумеруються у порядку впливу.

На першому рівні декомпозиції моделі головна функція деталізується на наступні блоки представлені на рис. 3.2.

- вивчення лекційного матеріалу;
- проходження практичного заняття;
- проведення контрольної роботи;
- проведення підсумкової роботи.

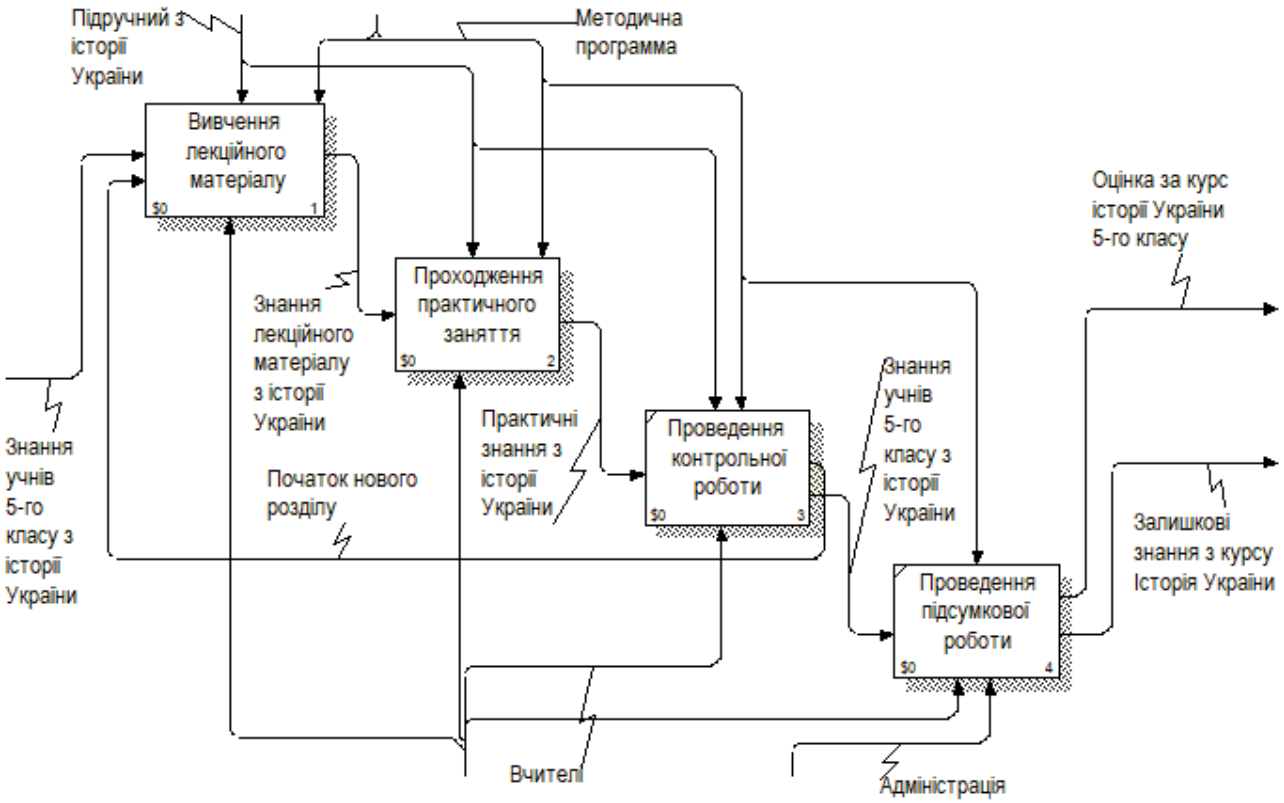


Рисунок 3.2 – Перший рівень декомпозиції IDEF0

Вхідними стрілками до діяльності «Вивчення лекційного матеріалу» є «Знання учнів 5-го класу з історії України» та стрілка зворотного зв’язку «Початок нового розділу»; вихідною – «Знання лекційного матеріалу з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Проходження практичного заняття» є «Знання лекційного матеріалу з історії України»; вихідною – «Практичні знання з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Проведення контрольної роботи» є «Практичні знання з історії України»; вихідною – «Знання учнів 5-го класу з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Проведення підсумкової роботи» є «Знання учнів 5-го класу з історії України»; вихідними – «Оцінка за курс історії України 5-го класу» та «Залишкові знання учнів з курсу історії України»; стрілкою контролю – «Методична програма»; стрілками механізмів – «Вчителі» та «Адміністрація».

Всього в розробленій структурно-функціональній моделі три рівні декомпозиції. Діаграму декомпозиції процесу «Вивчення лекційного матеріалу», що відноситься до другого рівня, зображено на рис. 3.3.

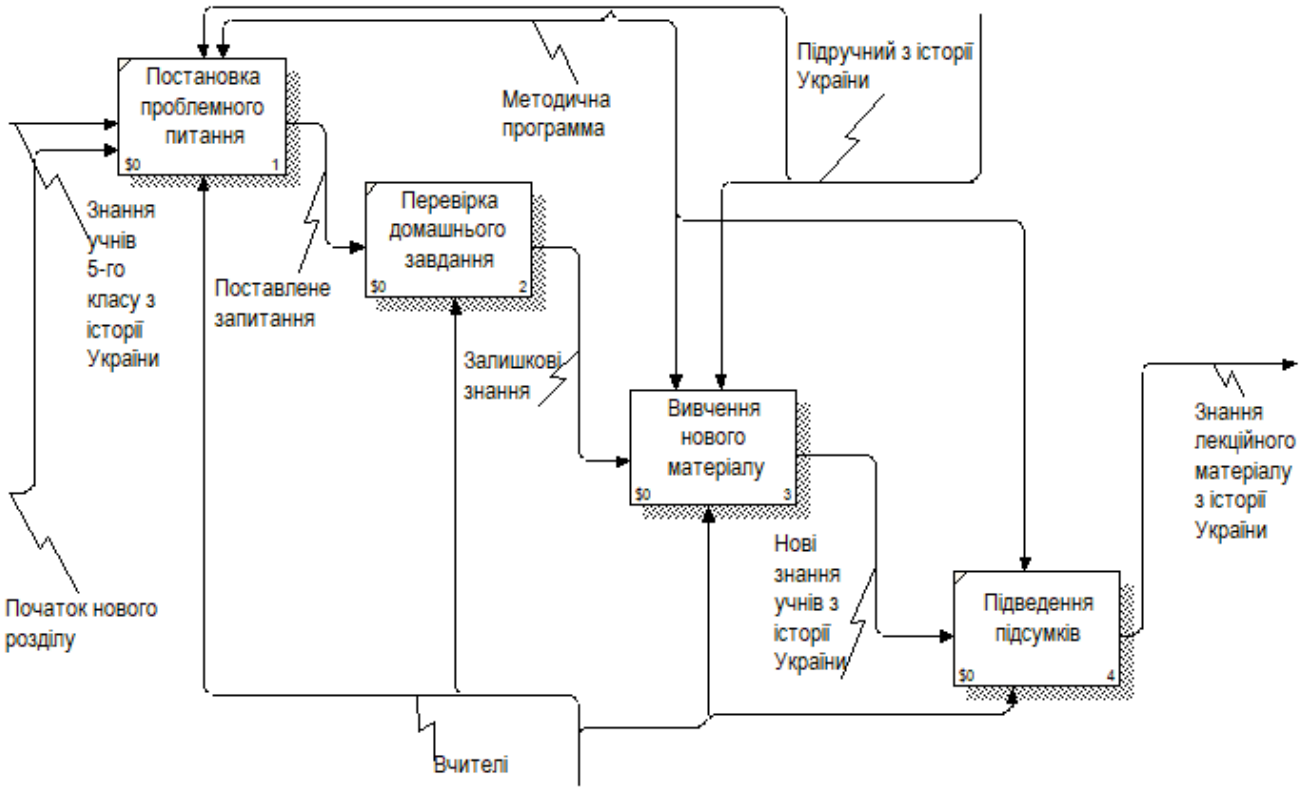


Рисунок 3.3 – Діаграма декомпозиції IDEF0 «Вивчення лекційного матеріалу»

Вхідною стрілкою до діяльності «Постановка проблемного питання» є «Знання учнів 5-го класу з історії України» » та стрілка зворотного зв’язку «Початок нового розділу»; вихідною – «Поставлене запитання»; стрілкою контролю – «Методична програма» та «Підручник з історії України»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Перевірка домашнього завдання» є «Поставлене запитання»; вихідною – «Залишкові знання»; стрілкою контролю – «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Вивчення нового матеріалу» є «Залишкові знання»; вихідною – «Нові знання з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілками механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Підсумки лекції» є «Нові знання з історії України»; вихідною – «Знання лекційного матеріалу з історії України»; стрілкою контролю – «Методична програма»; стрілкою механізмів – «Вчителі».

Декомпозицію блоку третього рівня «Вивчення нового матеріалу» зображено на рис. 3.4.

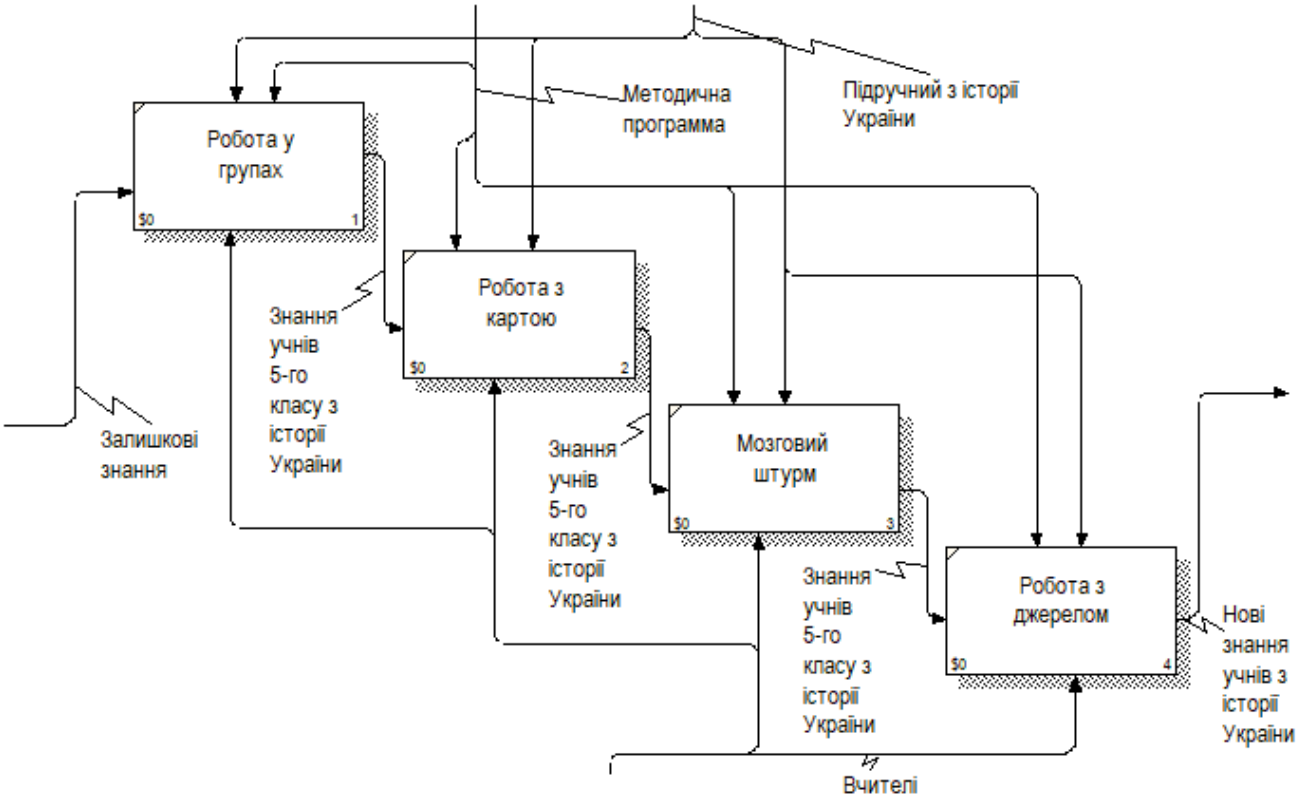


Рисунок 3.4 – Діаграма декомпозиції IDEF0 «Вивчення нового матеріалу»

Вхідною стрілкою до діяльності «Робота у групах» є «Залишкові знання»; вихідною – «Знання учнів 5-го класу з історії України»; стрілками контролю –

«Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Робота з картою» є «Знання учнів 5-го класу з історії України»; вихідною – «Знання учнів 5-го класу з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Мозковий штурм» є «Знання учнів 5-го класу з історії України»; вихідною – «Знання учнів 5-го класу з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Робота з джерелом» є «Знання учнів 5-го класу з історії України»; вихідною – «Знання учнів 5-го класу з історії України»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Декомпозицію блоку «Проходження практичного заняття» зображено на рис. 3.5.

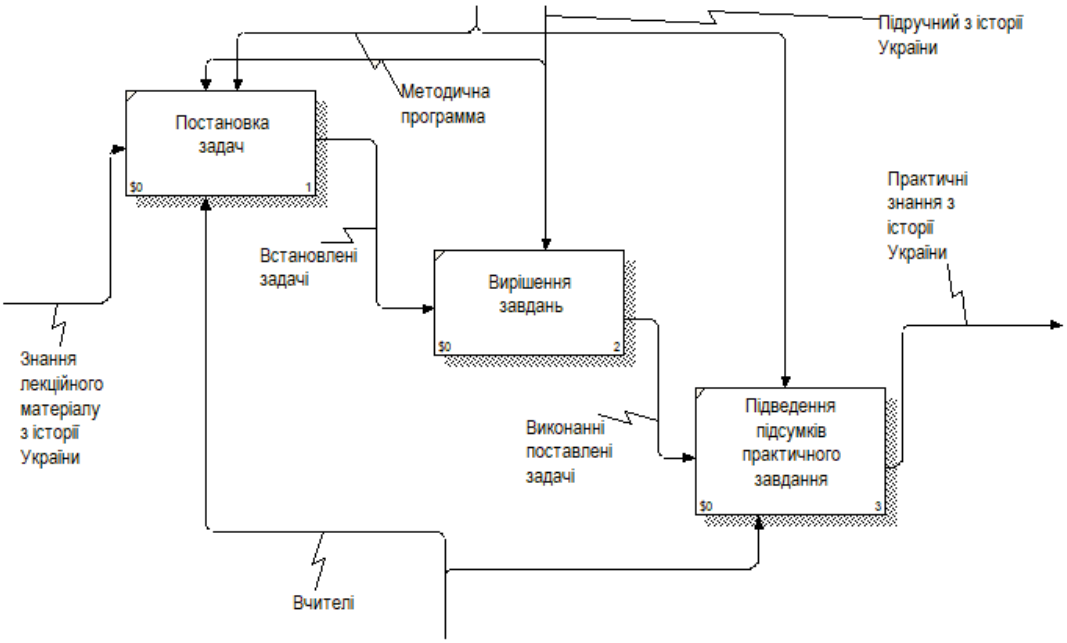


Рисунок 3.5 – Діаграма декомпозиції IDEF0 «Проходження практичного заняття»

Вхідною стрілкою до діяльності «Постановка задач» є «Знання лекційного матеріалу з історії України»; вихідною – «Встановлені задачі»; стрілками контролю – «Методична програма» та «Підручник з історії України»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Вирішення завдань» є «Встановлені задачі»; вихідною – «Виконані поставлені задачі»; стрілками контролю – «Підручник з історії України» та «Методична програма»; стрілкою механізмів – «Вчителі».

Вхідною стрілкою до діяльності «Підсумки практичної роботи» є «Виконані поставлені задачі»; вихідною – «Практичні знання з історії України»; стрілкою контролю – «Методична програма»; стрілкою механізмів – «Вчителі».

3.2 Моделювання даних програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Для моделювання даних, з якими має працювати додаток, потрібно виділити інформаційні сутності, визначити їх атрибути та показати відношення між сутностями. Для показу структури даних та зв'язків між елементами інформації буде застосована ER-діаграма. Даний підхід зручний тим, що процес моделювання даних є ітераційним.

Інформаційна модель типу ER-діаграми включає в себе такі основні елементи:

− сутностності бази даних що створюється;

− атрибути сутностей;

− опис взаємозв'язків між сутностями.

Коли ми говоримо про сутність, ми зазвичай говоримо про деякий аспект реального світу, який можна виділити поміж інших аспектів. Як зазначено в [21] сутність – це збірне поняття, деяка абстракція реального об'єкта, процесу, явища чи деякого уявлення про об'єкт. Хоча термін сутність більш важливий, потрібно розрізняти поняття типу сутності та екземпляру сутності. Поняття тип сутності

відноситься до набору однорідних особистостей, предметів, подій або ідей, що виступають як ціле. Екземпляр сутності відноситься до конкретної речі в наборі.

Основними сутностями бази даних можна виділити:

– користувачів;

– інформацію про запитання;

– інформацію про відповіді;

– інформацію про фрази;

– інформацію про теми;

– інформацію про параграфи;

– інформацію про успішність;

– інформацію про карту.

Кожна із зазначених сутностей має свій набір атрибутів, який наведено у табл. 3.1.

Таблиця 3.1 – Таблиця опису сутностей та їх атрибутів

Назва таблиці	Назва сутності	Назва поля	Опис атрибуту
loginInfo	Інформація про користувача	id	id користувача
		group	інформація про групу користувача
		login	логін користувача
		pass	пароль користувача
marksInfo	Інформація про успішність	class	клас користувача
		subject	назва предмету
		first_name	ім'я користувача
		second_name	прізвище користувача
		patronymic	по батькові користувача
		marks	інформація про оцінки користувача
themeInfo	Інформація про тему	id	id теми
		idt	id для зв'язку між таблицями
		topic	назва теми

Продовження таблиці 3.1 – Таблиця опису сутностей та їх атрибутів

paragraphInfo	Інформація про параграф	id	id параграфу
		idt	id для зв'язку між таблицями
		game_id	інформація про набір даних для гри
		type	інформація про режим роботи
		paragraph	назва параграфа
answerData	Інформація про відповіді	game_id	інформація про обраний набір даних
		id	id набору відповідей
		fr_answer	текст першої відповіді
		sc_answer	текст другої відповіді
		th_answer	текст третьої відповіді
		fo_answer	текст четвертої відповіді
phraseData	Інформація про фрази	game_id	інформація про обраний набір даних
		id	id фрази
		phrase	текст фрази
questionData	Інформація про запитання	game_id	інформація про обраний набір даних
		id	id запитання
		question	текст запитання
mapData	Інформація про завдання з картою	game_id	інформація про обраний набір даних
		id	id інформації про карту
		mrkOneLoc	набір даних з положенням першого маркеру
		mrkTwoLoc	набір даних з положенням другого маркеру
		mrkThreeLoc	набір даних з положенням третього маркеру

На основі виділених сутностей та їх атрибутів можна створити діаграму типу «сутність-зв'язок».

ER-діаграму додатку можна побачити на рис. 3.6.

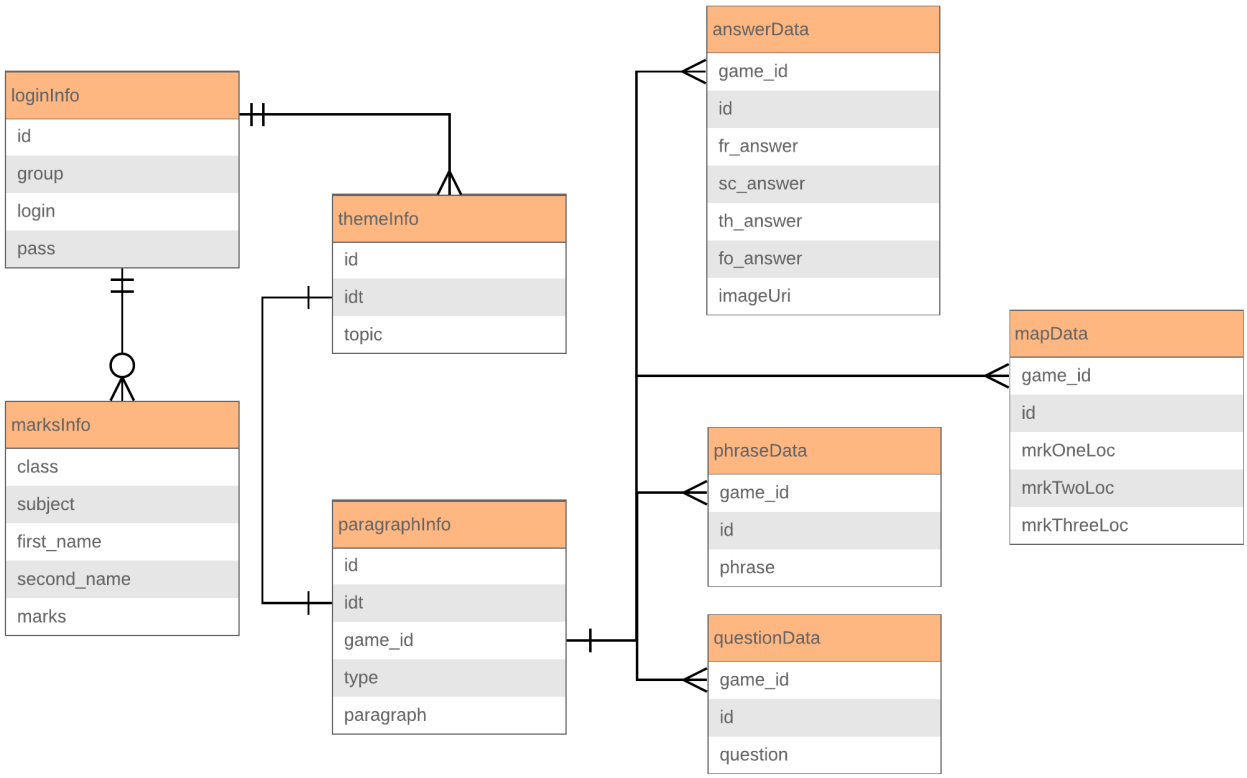


Рисунок 3.6 – Діаграма бази даних типу «сутність-зв’язок»

На зображеній діаграмі можна побачити зв’язки між сутностями. Так наприклад між таблицею з інформацією про користувача (loginInfo) встановлений зв’язок з таблицею що містить інформацію про успішність (marksInfo). Між ними встановлено зв’язок один і тільки один – нуль або багато. Такі зв’язки були використані через те що лише один користувач може зайти у додаток, але він може звернутися до талиці з оцінками багато разів, а може не звернутися взагалі.

Між таблицями з інформацією про параграфи (paragraphInfo) та теми(themeInfo) встановлено зв’язок один – один через те що лише один параграф може бути обраний у обраній темі.

Між таблицею з параграфами та таблицями що містять інформацію необхідну для гри встановлено зв’язки один – багато, тому що обравши один параграф додаток буде багато разів звертатися до таблиць з інформацію про запитання (questionData), фрази (phrasesData), відповіді (answerData) та карти (mapData).

3.3 Побудова моделі варіантів використання програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Для повного розуміння функціювання програмного додатку використовується модель варіантів використання, яка складається із опису акторів, варіантів використання системи, взаємодії між ними у формі Use Case-діаграми (діаграми прецедентів) та прототипу інтерфейсу.

Як вказано у [22] діаграма варіантів використання являє собою список дій, які зазвичай визначають взаємодію між роллю (відомої в уніфікованій мові моделювання як актор), і системи для досягнення мети. Актор може бути людиною або іншою зовнішньою системою.

Відповідно до даних структурно-функціональної діаграми процесу вивчення історії та за вимогами технічного завдання були виділені три актори, які взаємодіють із розроблюваною системою:

– Учні – користувачі системі, які використовують систему і мають частковий доступ до розділу успішності учнів (можливість переглядати лише оцінки поточного користувача);

– Вчителі – користувачі системі, які використовують систему і мають повний доступ до розділу успішності учнів (можливість переглядати оцінки всіх користувачів системи);

– СКБД – система керування базою даних, що надає функціонал обробки сукупності даних, які використовуються системою.

Також були виділені такі варіанти використання:

– ВВ 1 Авторизація – ВВ надає доступ до програмного додатку з певним рівнем доступу; та

– ВВ 2 Перегляд оцінок – ВВ дозволяє користувачам переглядати поточний стан успішності;

– ВВ 3 Вибір теми – ВВ дозволяє обрати потрібний розділ та тему для

вивчення та обрати один з трьох варіантів роботи додатку: режим вивчення, режим практичної роботи або режим контрольної роботи;

– ВВ 4 Проходження гри – ВВ дозволяє користувачам використовувати додаток у обраному режимі роботи.

Програмний додаток включатиме в себе 3 основні режими роботи:

− режим вивчення – режим у якому учні, в інтерактивній формі, спілкуючись з професором (уявним персонажем гри) будуть вивчати предмет, а саме Історію України. Цей режим представлятиме собою діалог між учнем та грою, для більшого заохочення до навчання. Оцінка за проходження теми у цьому режимі роботи не виставляється;

− режим практичної роботи – режим у якому учням будуть представлені практичні завдання (робота з картою, робота з джерелом, тестові завдання). Даний режим також буде реалізовано у вигляді діалогу, але на відміну від першого, він буде спрямований на практичне засвоєння матеріалу. Також цим режимом передбачене виставлення оцінок після завершення роботи;

− режим контрольної роботи – режим, що складається лише с запитань та відповідей і не містить у собі загальних фраз професора. Режим також передбачає виставлення оцінок після завершення роботи.

Також розробка моделі варіантів використання передбачає створення прототипу інтерфейсу програмного додатку. Прототип інтерфейсу можна знайти у технічному завданні, що знаходиться у додатку А.

З сценаріями варіантів використання програмного додатку, для більш детального їх розуміння можна ознайомитися у додатку В.

На основі виділених вище варіантів використання та акторів була створена діаграма варіантів використання яка зображена на рис. 3.7.

Рисунок 3.7 – Діаграма варіантів використання

3.4 Архітектура програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Для повного розуміння архітектури програмного додатку розроблена діаграма компонентів за допомогою нотації мови UML.

У якості компонентів у даній діаграмі виступають складові кожного із програмних модулів, а саме:

− компонент бізнес-логіки – компонент, який містить в собі програмну логіку додатку. Відповідає за зв’язок з базою даних, обробку та завантаження інформації. Зазвичай містить слово Model у назві класу;

− компонент моделі відображення – компонент, який відповідає за зв’язок між логікою додатку та його інтерфейсом. Містить у собі змінні для прив’язки даних з параметром INotifiPropertyChanged. Зазвичай містить слово ViewModel у назві класу;

− компонент інтерфейсу – компонент, який представляє собою інтерфейс додатку. Може бути представлений у вигляді вікна або користувацького компоненту. Зазвичай містить слово View у назві класу.

Через обраний паттерн розробки зв'язок між інтерфейсом та моделлю відображення відбувається за допомогою прив’язок, а тому буде відображатися пунктирною лінією. Для кожного із програмних модулів схема виглядає однаково.

У табл. 3.2 наведені опис та назва кожного компоненту.

Таблиця 3.2 – компоненти програмного додатку та їх опис

SQLite	компонент, що відповідає за роботу з базами даних
LoginModel	компонент з бізнес-логікою модуля авторизації
MarksModel	компонент з бізнес-логікою модуля успішності
GameModel	компонент з бізнес-логікою модуля вивчення
ThemeModel	компонент з бізнес-логікою модуля вибора теми
LoginViewModel	компонент зі змінними для модуля авторизації, і призначений для зв'язку бізнес-логіки з інтерфейсом
MarksViewModel	компонент зі змінними для модуля успішності, і призначений для зв'язку бізнес-логіки з інтерфейсом
MainViewModel	компонент зі змінними для контейнера меню, і призначений для зв'язку бізнес-логіки з інтерфейсом
GameViewModel	компонент зі змінними для модуля вивчення, і призначений для зв'язку бізнес-логіки з інтерфейсом
ThemeViewModel	компонент зі змінними для модуля вибору теми, і призначений для зв'язку бізнес-логіки з інтерфейсом
LoginView	компонент інтерфейсу модуля авторизації

Продовження таблиці 3.2 – компоненти програмного додатку та їх опис

MarksView	компонент інтерфейсу модуля успішності
MainWindow	компонент інтерфейсу контейнера меню
GameView	компонент інтерфейсу модуля вивчення
ThemeView	компонент інтерфейсу модуля вибору теми

На основі виділених компонентів була створена діаграма компонентів розроблюваного додатку, яка показана на рис. 3.8.

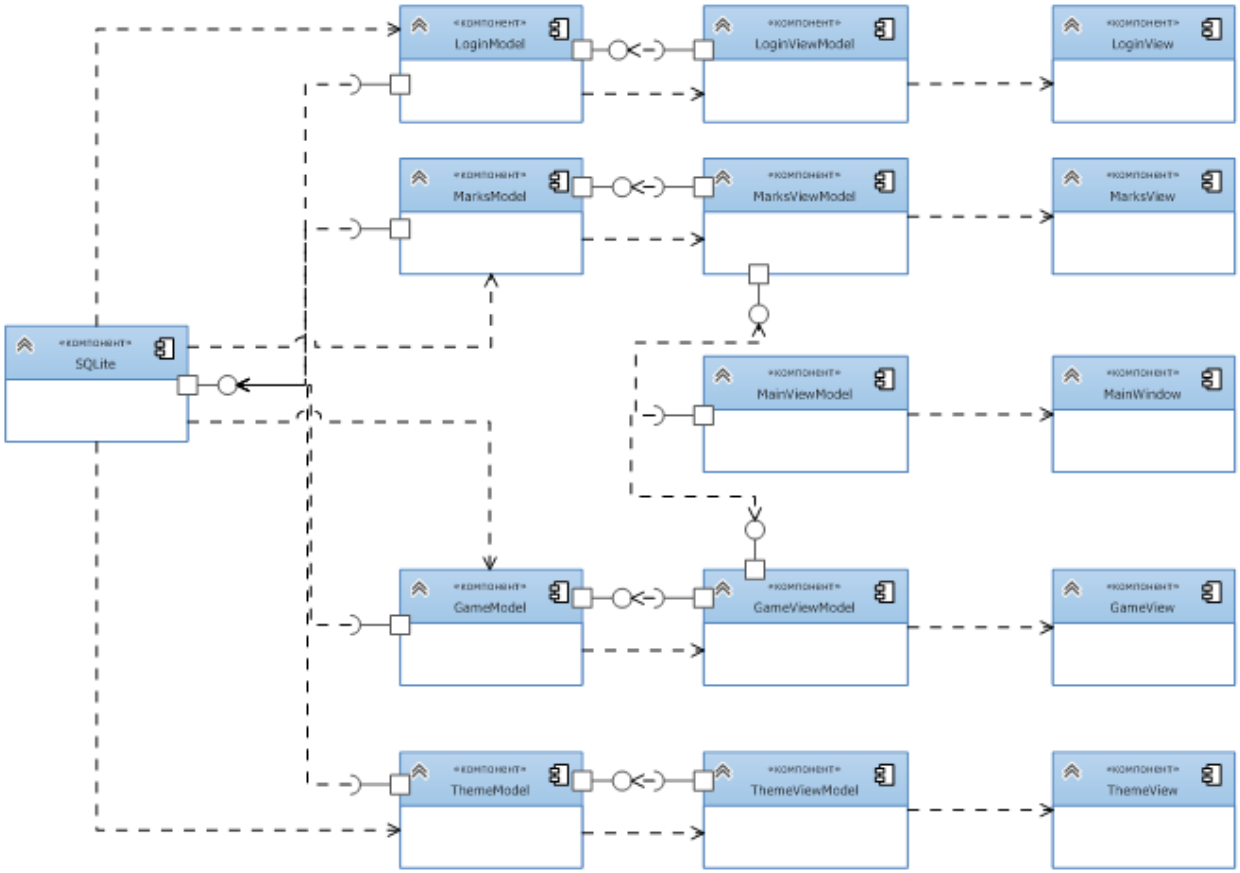


Рисунок 3.8 – Діаграма компонентів

3.5 Побудова діаграми класів програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Для опису структури розроблюваного додатку буде застосована діаграма класів, яка відповідно до [23] визначень уніфікованої мови моделювання (UML) є статичною діаграмою, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) і відносини між об'єктами.

Діаграма класів є основою об'єктно-орієнтованого моделювання. Вона використовується не тільки для загального концептуального моделювання структури програми, а також для детального моделювання, максимально наближеного до програмного коду. Класи в діаграмі класів представляють як основні елементи взаємодії в додатку, так і класи, що програмуються.

На діаграмі класи представлені блоками, які містять три відділення:

– верхній відсік містить назву класу. Він друкується жирним шрифтом і вирівнюється по центру, а перша буква використовується великими літерами;

– середній відсік містить атрибути класу. Вони вирівнюються ліворуч;

– нижній відсік містить операції, які клас може виконувати. Вони також вирівнюються по лівому краю.

Використання паттерну MVVM полегшує створення діаграми класів додатку, через те що кожний компонент відповідає за свої функції. Для реалізації, зазначеного в технічному завданні (Додаток А) функціоналу, потрібно розробити 3 програмні модулі, які формують собою меню користувача, контейнер для їх відображення та модуль гри. Для кожного із модулів буде створено по три класи. У загальному вигляді перший клас відповідає за логіку додатку та називається Model. Другий клас створюється автоматично разом із створенням нового вікна (Window) або користувацького компоненту (User Control) і називається View. Він відповідає за графічне відображення. Третій клас створюється для реалізації компоненту

ViewModel, який в свою чергу відповідає за зв'язок між першим класом та інтерфейсом.

Використання набору інструментів MVVM Light передбачає створення ще одного класу, а саме класу ViewModelLocator, який створює новий екземпляр моделі-відображення для кожного екземпляру інтерфейсу при запуску додатку.

Як було сказано раніше у додатку планується розробка 4 основних модулів, а саме:

– модуль авторизації;

– модуль успішності;

– модуль вибору теми;

– модуль гри.

Кожен із модулів має свій набір атрибутів і операцій, які планується реалізувати у додатку. Також на діаграмі слід відобразити усі допоміжні класи, які також є невід’ємною частиною розробки, та включають в себе атрибути та операції.

Перелік усіх класів, що планується створити, їх опис, атрибути та операції наведені у табл. 3.3 та у табл. 3.4.

Таблиця 3.3 – Опис класів та атрибутів

Назва класу	Опис класу	Назва атрибуту	Опис атрибуту
LoginView-Model	Клас відображення модуля авторизації	_check	змінна з результатом авторизації
		_group	група користувача
		_login	логін користувача
		_pass	пароль користувача
MainGameView-Model	Клас відображення модуля вивчення	_btnOneText	текст першої кнопки
		_btnTwoText	текст другої кнопки
		_btnThreeText	текст третьої кнопки
		_btnFourText	текст четвертої кнопки
		_game_id	id для ініціалізації гри
		_text	текст для запитань та фраз
MainView-Model	Клас відображення контейнеру меню	game_id	id для ініціалізації гри
		group	група користувача
		login	логін користувача
		type	режим роботи додатку

Продовження таблиці 3.3 – Опис класів та атрибутів

MarksView-Model	Клас відображення модуля успішності	_marks	інформація про успішність користувача
ViewModel-Locator	Клас локатору моделей відобржаення для створення інстанцій цих моделей	Game	змінна, що містить інстанцію моделі відображення для модуля гри
		Login	змінна, що містить інстанцію моделі відображення для модуля авторизації
		Main	змінна, що містить інстанцію моделі відображення для контейнеру меню
		Marks	змінна, що містить інстанцію моделі відображення для модуля успішності
		Theme	змінна, що містить інстанцію моделі відображення для модуля вибору тем

Таблиця 3.4 – Опис класів та операцій

Назва класу	Опис класу	Назва операції	Опис операції
LoginModel	Бізнес-логіка модуля авторизації	getGroup()	отримання групи користувача
		loginCheck()	валідації авторизації
MainGame-Model	Бізнес-логіка модуля вивчення	checkAnswer()	перевірка правильності відповіді
		getAnswerOne()	отримання тексту першої відповіді
		getAnswerTwo()	отримання тексту другої відповіді
		getAnswerThree()	отримання тексту третьої відповіді
		getAnswerFour()	отримання тексту четвертої відповіді
		getPhrase()	отримання тексту фрази
		gerQuestion()	отримання тексту запитання
		writeMarks()	запис оцінки отриманої користувачем
MarksModel	Бізнес-логіка модуля успішності	getMarks()	отримання інформації про успішність
ThemeModel	Бізнес-логіка модуля вибору теми	checkContent()	отримання id для ініціалізації гри
		getType()	вибір режиму гри
		loadParagraphs()	завантаження списку параграфів с бази

		loadTheme()	завантаження списку тем с бази

Продовження таблиці 3.4 – Опис класів та операцій

LoginView-Model	Клас відображення модуля авторизації	initiateLogin()	ініціалізація процесу авторизації
MainGameView-Model	Клас відображення модуля вивчення	BtnOneOnClick()	команда, що виконується при натисканні на кнопку один
		BtnTwoOnClick()	команда, що виконується при натисканні на кнопку один
		BtnThreeOnClick()	команда, що виконується при натисканні на кнопку один
		BtnFourOnClick()	команда, що виконується при натисканні на кнопку один
		initiateGame()	ініціалізація процесу гри
MainView-Model	Клас відображення контейнеру меню	CloseWindow()	закриття вікна
		RestartWindow()	операція виходу з аккаунту
MarksView-Model	Клас відображення модуля успішності	loadMarks()	завантаження оцінок користувача
ThemeView-Model	Клас відображення модуля вибору тем	TreeViewItem_Selected()	команда, що виконується при натисканні на обрану тему
ViewModel-Locator	Клас локатору моделей відображення для створення інстанцій цих моделей	Cleanup()	операція очищення ресурсів моделей відображення

Діаграму класів програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу» можна побачити на рис. 3.9.

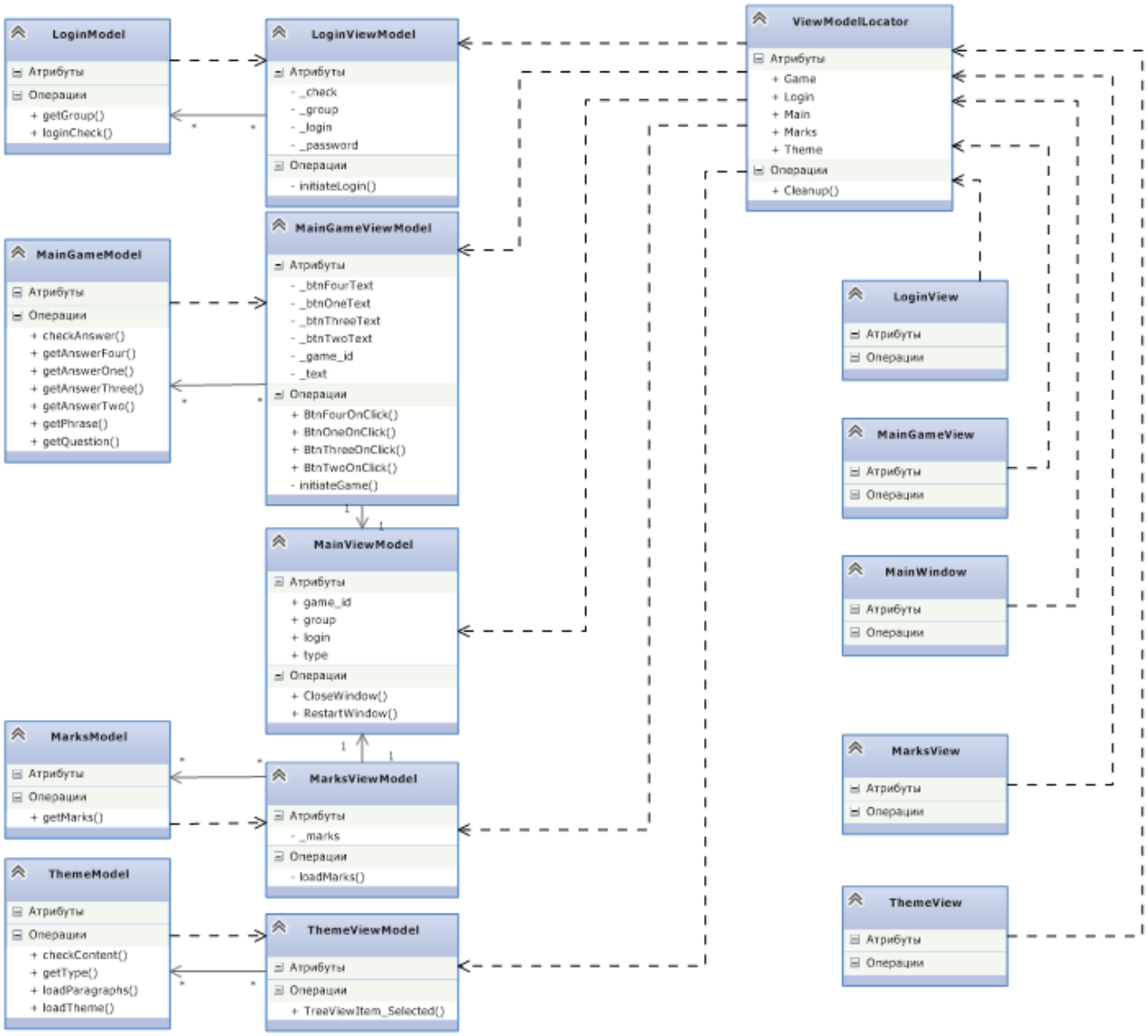


Рисунок 3.9 – Діаграма класів

4 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

4.1 Розгляд основного паттерну розробки програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Для реалізації програмного додатку було обрано паттерн MVVM (Model-View-ViewModel), який реалізується за допомогою набору інструментів MVVM Light. В основі даного паттерну лежить принцип розмежування інтерфейсу від бізнес-логіки. Згідно з [24] паттерн MVVM має три основних компоненти: модель (Model), яка представляє бізнес-логіку додатка, уявлення (View) призначеного для користувача інтерфейсу XAML, і модель-представлення (View Model), в якій міститься вся логіка побудови графічного інтерфейсу і посилання на модель .

На рис. 4.1 представлена діаграма, яка показує відношення між елементами програмного додатку необхідні для реалізації паттерну MVVM.

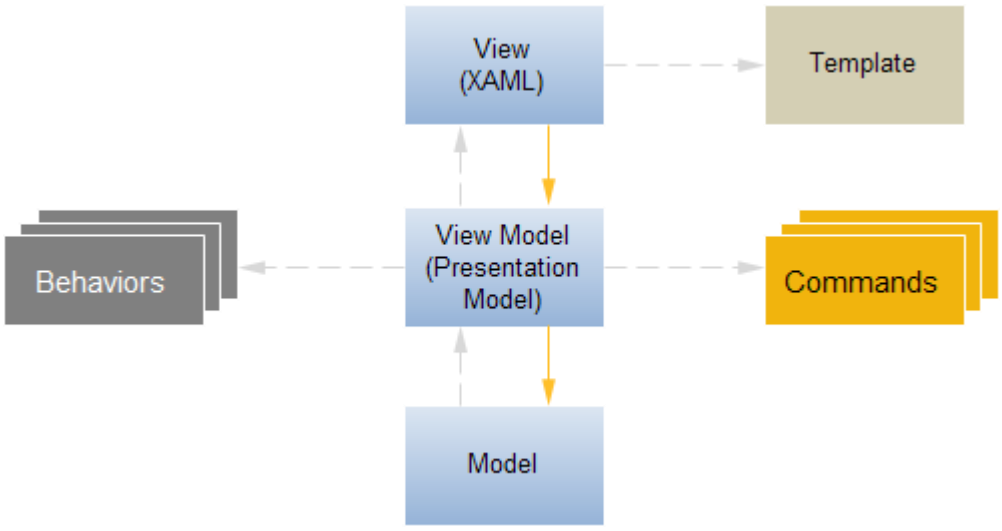


Рисунок 4.1 – Діаграма відношень між складовими паттерну MVVM

Для використання паттерну MVVM у підсистемі WPF компанією Microsoft був реалізований механізм прив'язок (Binding).

Для динамічної прив'язки до властивості XAML-елементу у наборі інструментів MVVM Light існує інтерфейс INotifyPropertyChanged, який реалізовано через метод Set(ref _var, value). Цей інтерфейс реалізує систему повідомлень яка активується, коли значення властивості змінюється. Це потрібно у моделі-представлення, щоб зробити механізм прив'язки, призначеного для користувача інтерфейсу XAML, динамічним.

Наступний інтерфейс реалізує можливості команд. Такі можливості надаються інтерфейсом ICommand, який доступний для WPF і Silverlight. Команди можуть прив'язуватися до певного XAML-елементу і визначати поведінку даного елемента при певних діях. У наборі інструментів MVVM Light даний інтерфейс реалізовано за допомогою класу Relay Command.

Далі для прикладу використання паттерну MVVM показано повний процес розробки модуль авторизації користувача.

Першим кроком розробки з використанням паттерну MVVM є створення графічного представлення програмного додатку, а саме створення інтерфейсу. Інтерфейс програмного додатку реалізується за допомогою мови розмітки для додатків XAML.

XAML лістинг інтерфейсу можна знайти у додатку Г, а саме у підпункті «Лістинг LoginView».

На рис. 4.2 зображений інтерфейс програмного додатку у конструкторі.

А на рис. 4.3 зображений інтерфейс програмного додатку після запуску додатку.

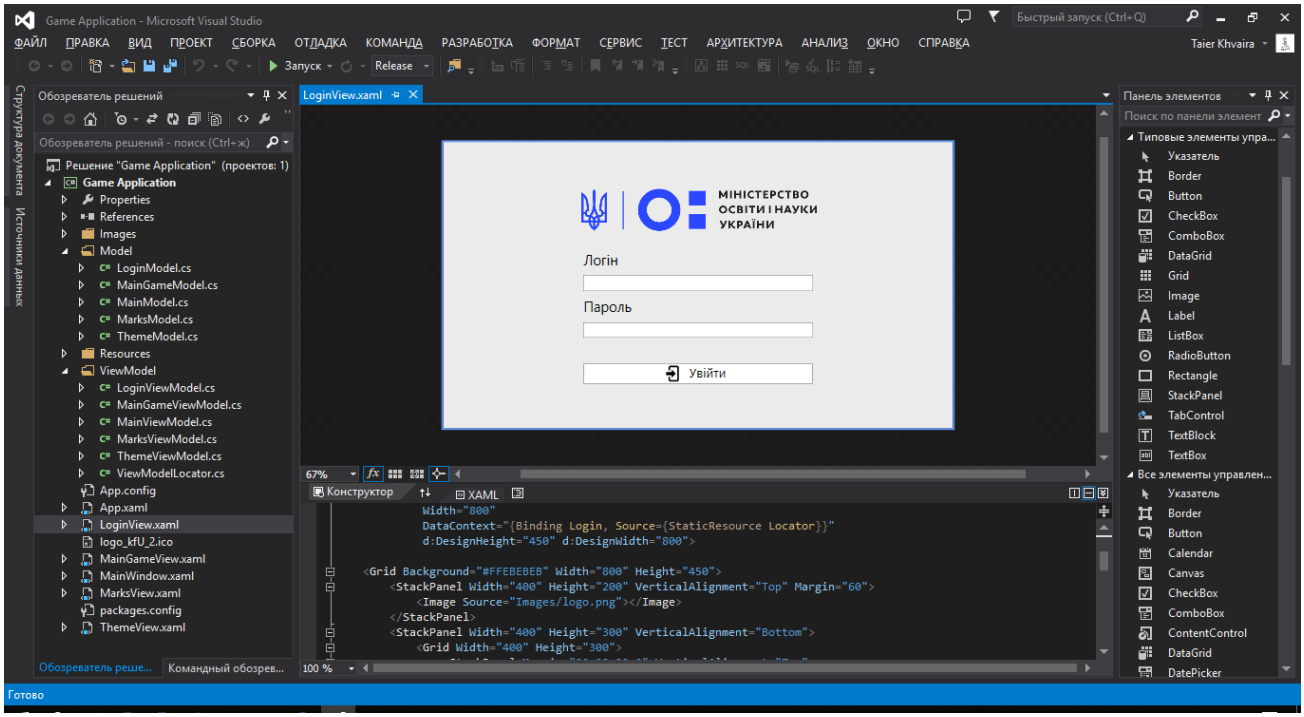


Рисунок 4.2 – Інтерфейс вікна авторизації у конструкторі

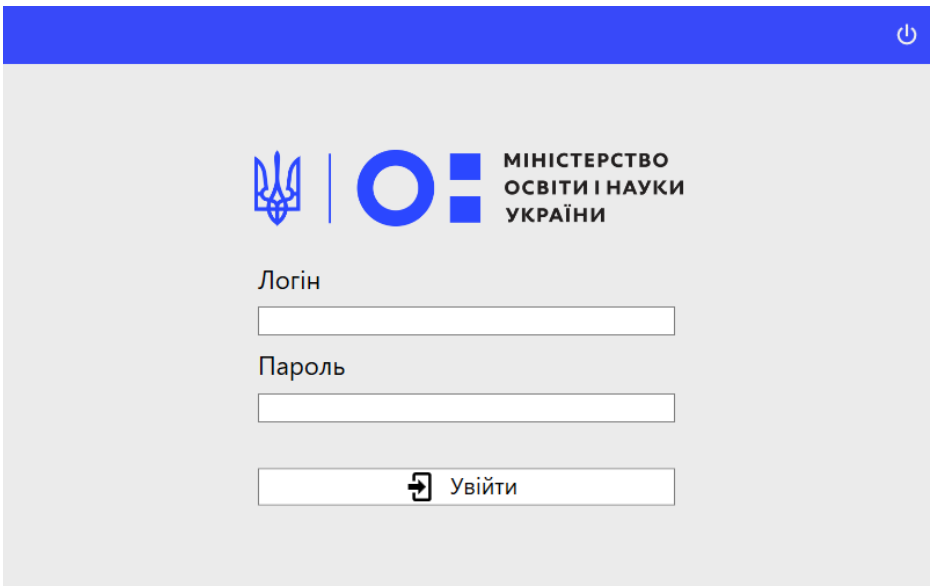


Рисунок 4.3 – Інтерфейс вікна авторизації

Наступним етапом розробки програмного додатку з використанням паттерну MVVM є створення моделі відображення. Зв'язування інтерфейсу та моделі представлення виконується за допомогою властивості Data Context, але у випадку

використання набору інструментів за це відповідає ViewModelLocator. Зв'язування між моделлю представлення та елементами інтерфейсу відбувається за допомогою властивості Binding. Подія яка відбувається при натисканні на кнопку реалізована у вигляді команди. Код команд та повний лістинг моделі відображення знаходиться у додатку Г, а саме у підпункті «Лістинг LoginViewModel.

Останнім етапом є створення моделі програмного продукту, а саме опис бізнес-логіки. У моделі модуля авторизації реалізовано 2 методи:

– метод валідації даних для авторизації (loginCheck());

– метод отримання групи користувача для розподілення прав доступу (get-Group()).

Така послідовність розробки необов’язкова, розробку за допомогою паттерну можна починати з будь-якої частини (з моделі або інтерфейсу).

Програмний код моделі модуля авторизації можна побачити у додатку Г, а саме у підпункті «Лістинг LoginModel».

На рис. 4.4 можна побачити вигляд програмного коду у редакторі IDE Visual Studio.

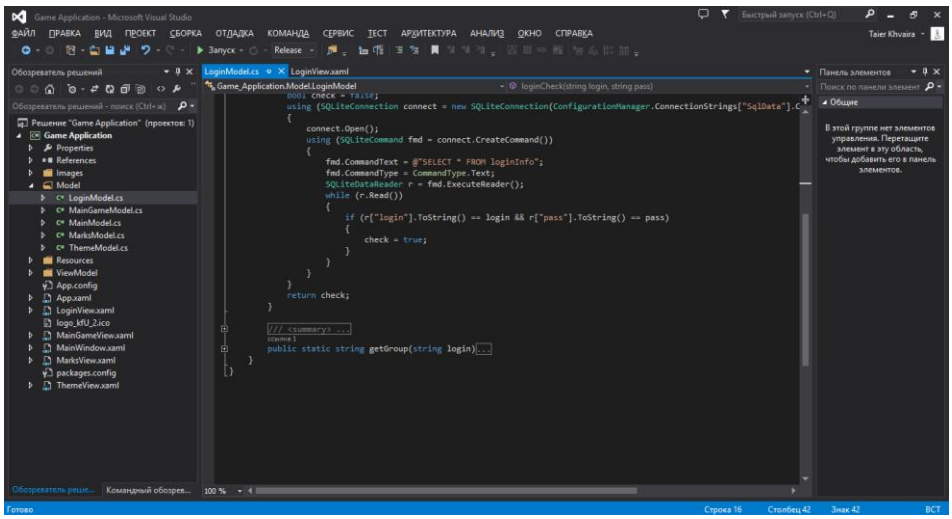


Рисунок 4.4 – Процес розробки бізнес-логіки модуля авторизації

4.2 Розробка модуля успішності програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Розробку модуля успішності розпочнемо зі створення інтерфейсу. Інтерфейс даного модуля буде складатися лише із компонента таблиці (DataGrid), для відображення завантажених із бази оцінок.

XAML лістинг інтерфейсу можна знайти у додатку Ґ, а саме у підпункті «Лістинг MarksView».

На рис. 4.5 зображений інтерфейс модуля успішності у конструкторі.

Та на рис. 4.6 зображений інтерфейс модуля після запуску програмного додатку.

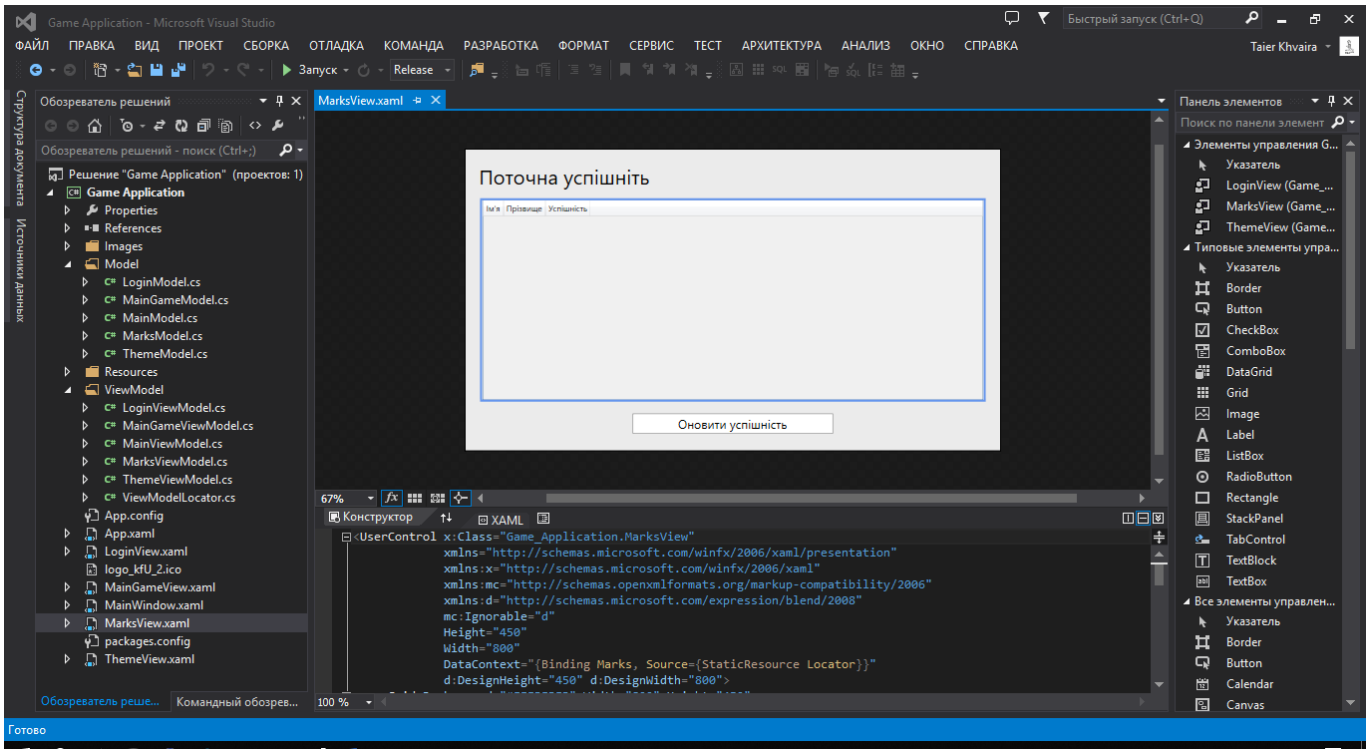


Рисунок 4.5 – Інтерфейс модуля успішності у режимі конструктора

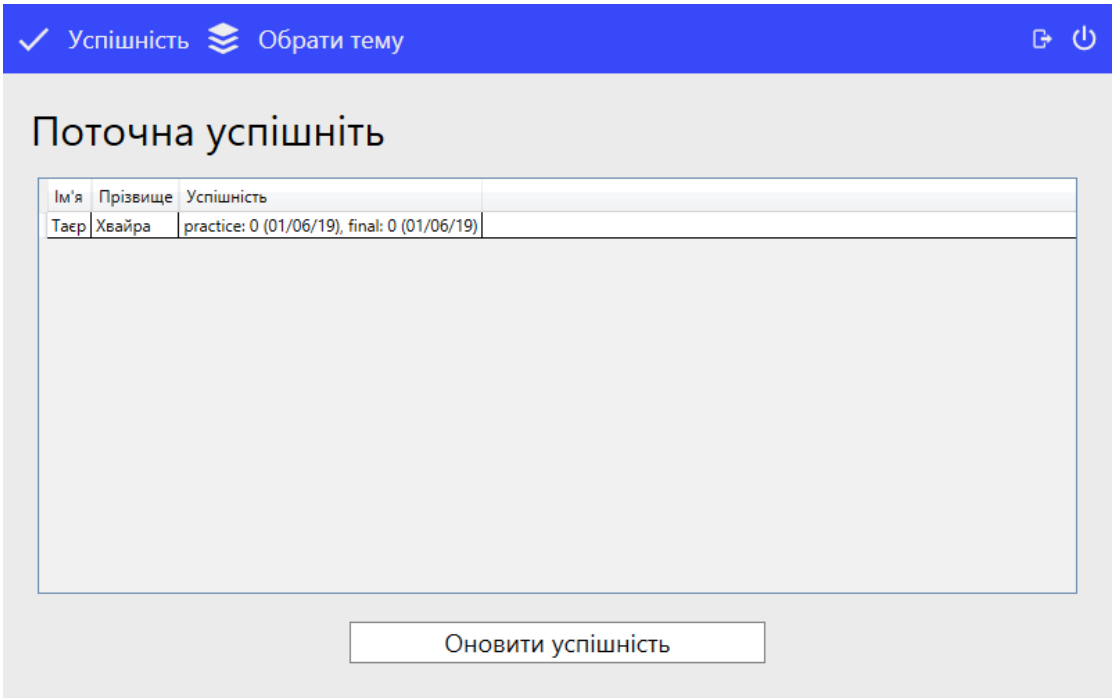


Рисунок 4.6 – Інтерфейс модуля успішності

Наступним кроком, аналогічно до створення модуля авторизації, є створення моделі відображення. Дана модель вміщує у собі змінні, які зберігають отримані від моделі дані, та передають їх до інтерфейсу за допомогою прив’язок. Також дана модель реалізує команду оновлення даних у компоненті таблиці на інтерфейсі.

Код команди оновлення та повний лістинг моделі відображення знаходиться у додатку Ґ, а саме у підпункті «Лістинг MarksViewModel.

Останнім етапом є створення моделі програмного продукту, а саме опис бізнес-логіки. У моделі модуля успішності реалізовано всього 1 метод:

– метод завантаження оцінок з бази даних (loadMarks());

Програмний код моделі модуля успішності можна побачити у додатку Ґ, а саме у підпункті «Лістинг MarksModel».

На рис. 4.7 можна побачити вигляд програмного коду у редакторі IDE Visual Studio.

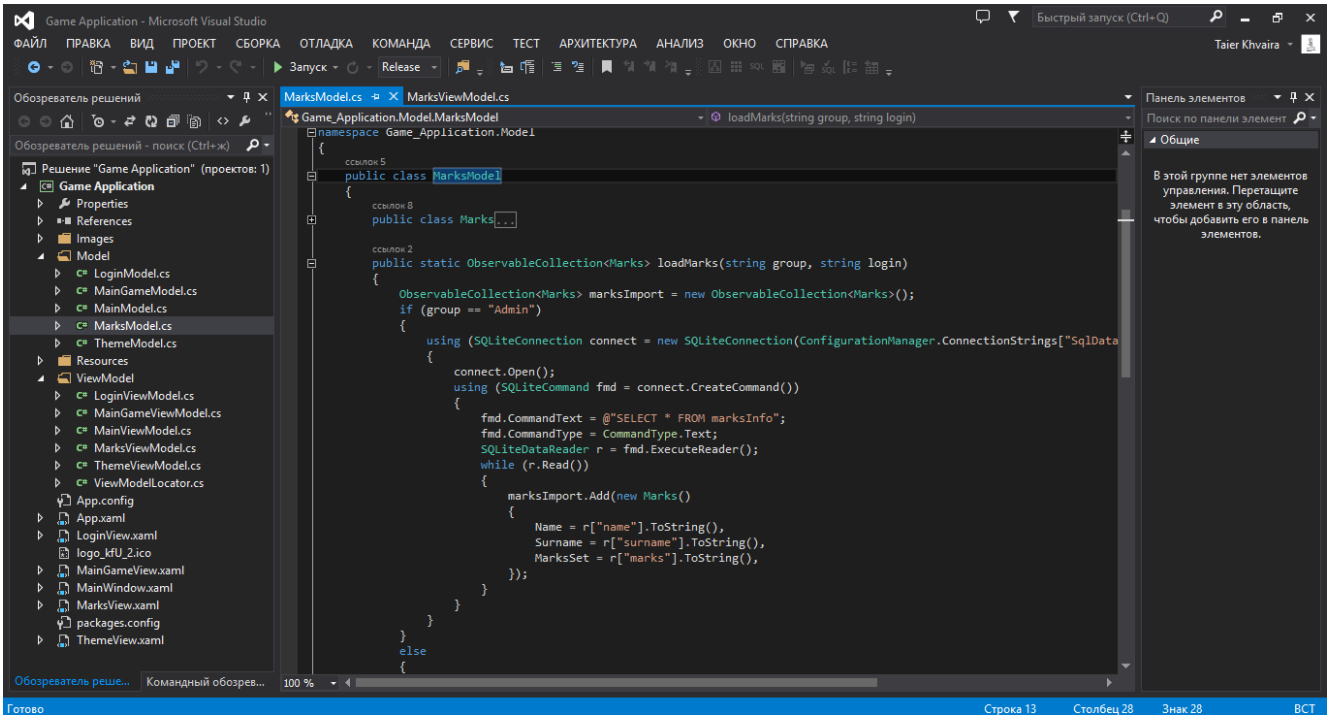


Рисунок 4.7 – Процес розробки бізнес-логіки модуля успішності

4.3 Розробка модуля вибору теми програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

По аналогії з попередніми модулями даний модуль починаємо розробляти з інтерфейсу. Головний компонент модуля – компонент перегляду інформації у вигляді дерева (TreeView).

XAML лістинг інтерфейсу можна знайти у додатку Д, а саме у підпункті «Лістинг ThemeView».

На рис. 4.8 зображений інтерфейс модуля вибору теми у конструкторі.

А на рис. 4.9 зображений інтерфейс модуля після запуску програмного додатку.

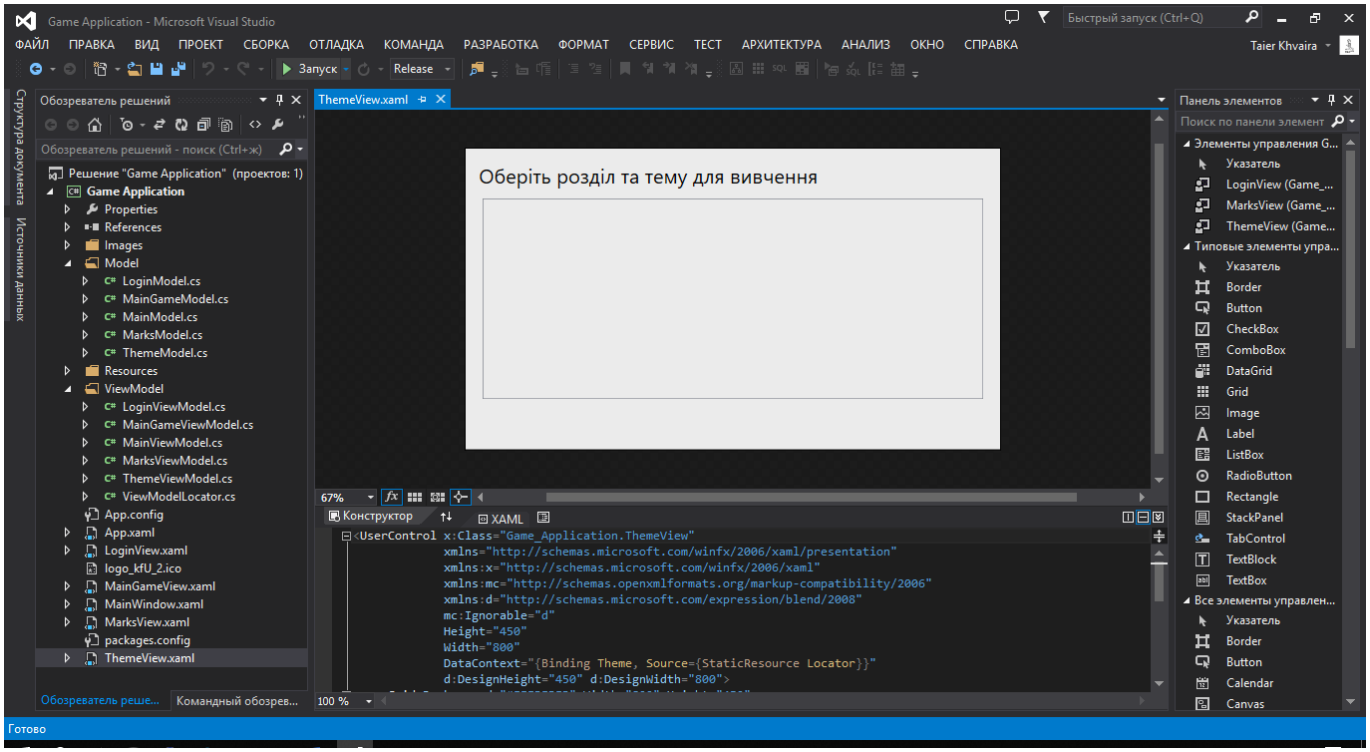


Рисунок 4.8 – Інтерфейс модуля вибору теми у режимі конструктора

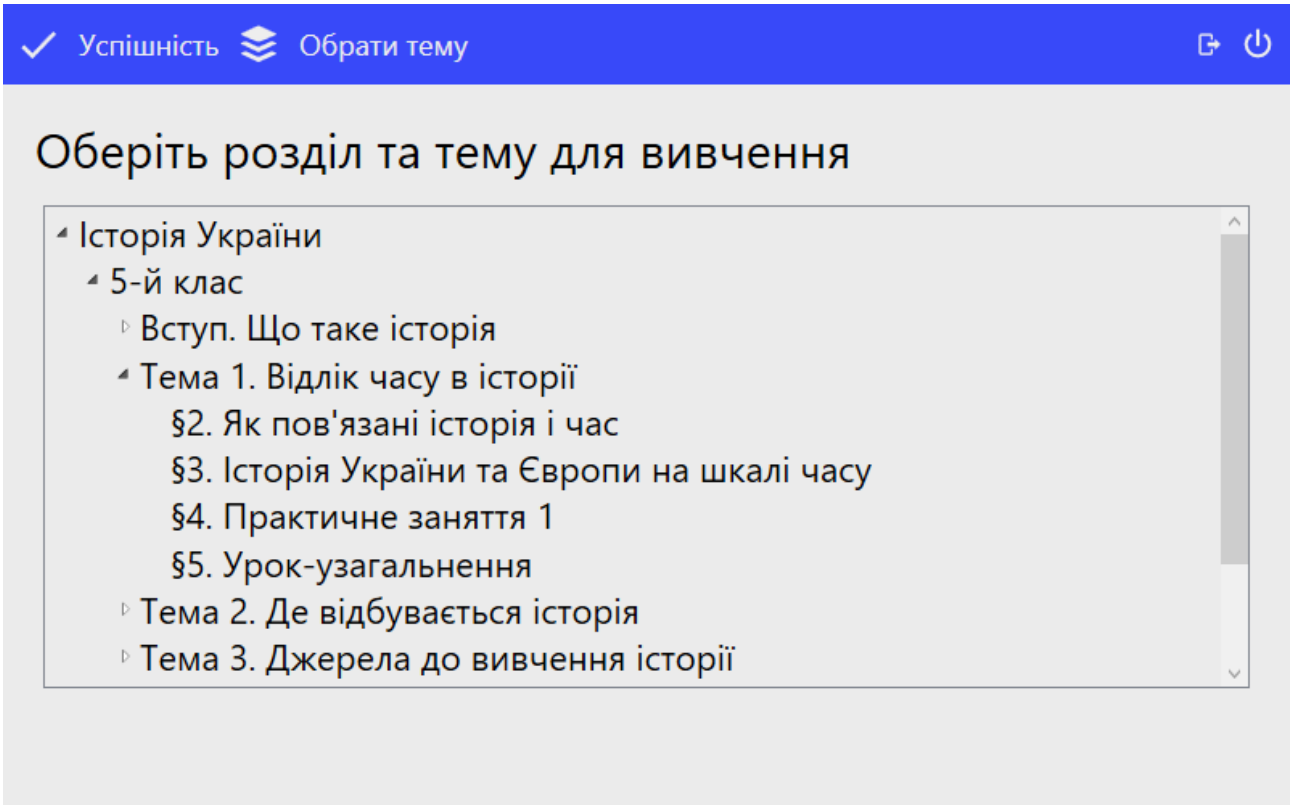


Рисунок 4.9 – Інтерфейс модуля вибору теми вивчення

Далі створимо модель відображення для модуля вибору теми. Дана модель вміщує у собі змінні, які зберігають отримані від моделі дані про параграфи та теми, та передають їх до інтерфейсу за допомогою прив’язок. Також дана модель реалізує команду запуску гри у обраному режимі та з обраним параграфом.

Повний лістинг моделі відображення знаходиться у додатку Ґ, а саме у підпункті «Лістинг ThemeViewModel».

Завершальним етапом є створення моделі програмного продукту, а саме опис бізнес-логіки модуля. У моделі модуля вибору теми реалізовано 4 методи:

– метод завантаження переліку параграфів з бази даних (loadParagraphs());

– метод завантаження переліку тем з бази даних (loadTopics());

– метод отримання обраного режиму роботи (getType());

– метод перевірки доступності теми у базі та отримання номеру гри(checkContent()).

Програмний код моделі модуля вибору теми можна побачити у додатку Ґ, а саме у підпункті «Лістинг ThemeModel».

На рис. 4.10 можна побачити вигляд програмного коду у редакторі IDE Visual Studio.

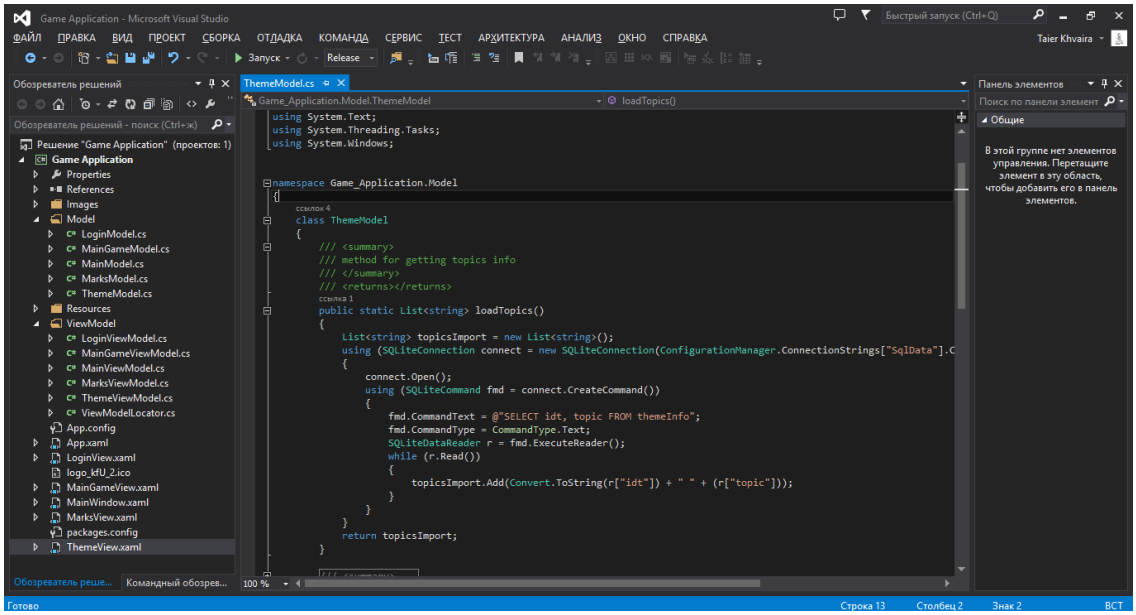


Рисунок 4.10 – Процес розробки бізнес-логіки модуля вибору теми

4.4 Розробка модуля вивчення програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Початок створення основного модуля, а саме модуля вивчення, розпочнемо зі створення інтерфейсу. Інтерфейс даного модуля включає в себе компонент зображення для відображення, компонент текстового поля для відображення фраз та запитань. Також включає 4 кнопки, до яких будуть прив'язані команди. У верхній частині розташовані кнопки: виходу с додатку, виходу з системи та повернення у головне меню.

XAML лістинг інтерфейсу можна знайти у додатку Е, а саме у підпункті «Лістинг MainGameView».

На рис. 4.11 зображений інтерфейс модуля вибору теми у конструкторі.

А на рис. 4.12 зображений інтерфейс модуля після запуску програмного додатку.

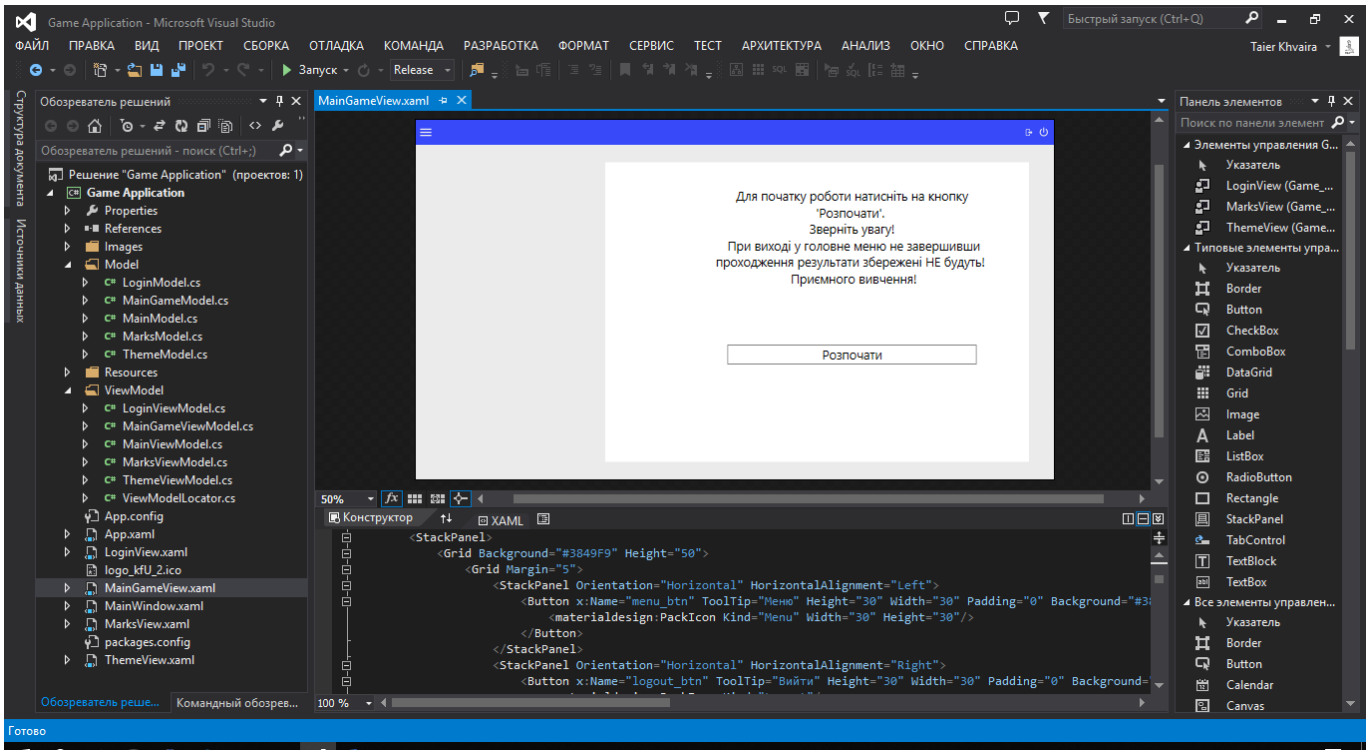


Рисунок 4.11 – Інтерфейс модуля вивчення у режимі конструктора

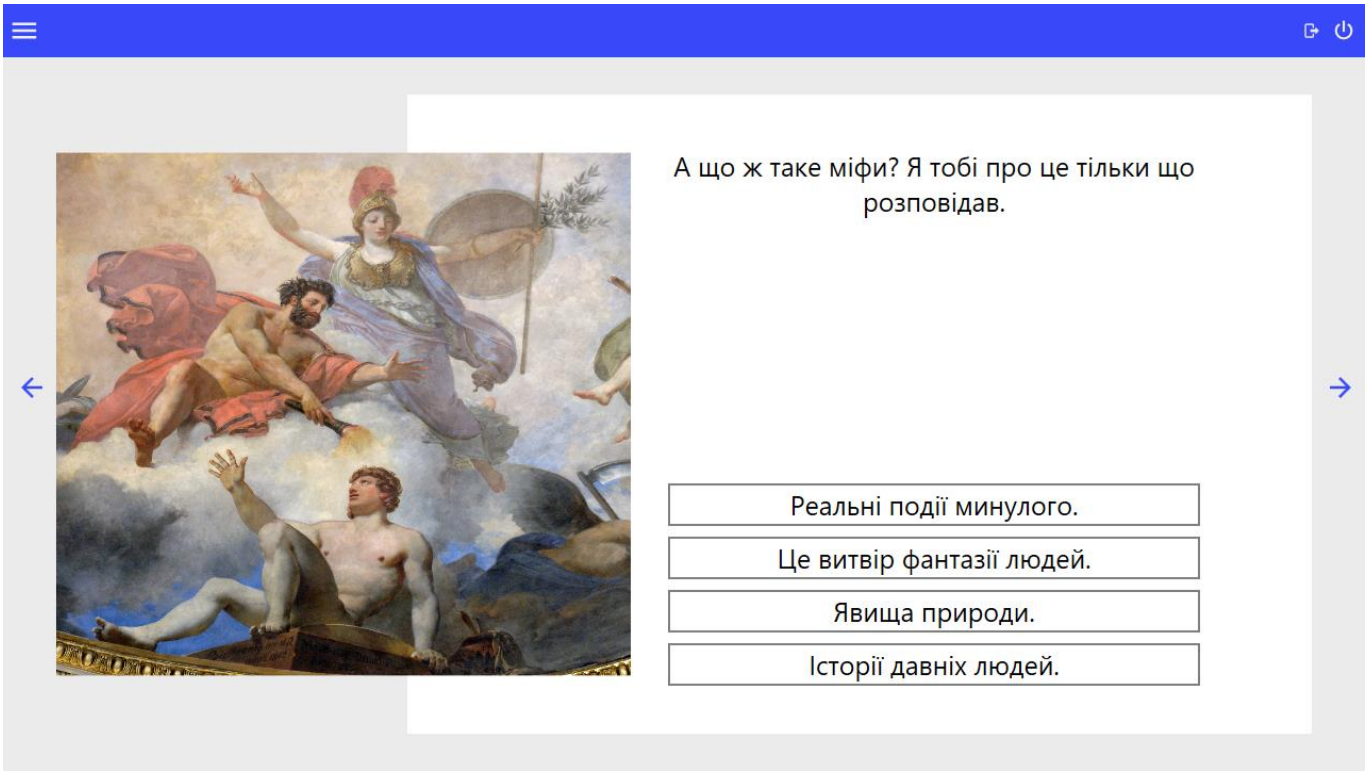


Рисунок 4.12 – Інтерфейс модуля вивчення

Далі створимо модель відображення для модуля вибору теми.

Модель відображення даного модуля включає в себе змінні для тексту конопок відповідей, змінну для тексту запитань та фраз, змінну для визначення номеру гру, змінну для рахунку гри, змінну для посилання на зображення. Також у моделі відображення реалізовано 12 команд:

– команда виходу з системи (RestartWindowCommand);

– команда виходу з додатку (CloseWindowCommand);

– команда для виходу у головне меню (MainWindowCommand);

– команда, що виконується при натисканні на першу кнопку;

– команда, що виконується при натисканні на другу кнопку;

– команда, що виконується при натисканні на третью кнопку;

– команда, що виконується при натисканні на четверту кнопку;

– команда, що виконується при натисканні на ліву стрілку;

– команда, що виконується при натисканні на праву стрілку;

– команда, що виконується при натисканні на перший маркер;

– команда, що виконується при натисканні на другий маркер;

– команда, що виконується при натисканні на третій маркер.

Повний лістинг моделі відображення знаходиться у додатку Е, а саме у підпункті «Лістинг MainGameViewModel».

Завершальним етапом розробки модуля є створення моделі програмного продукту, а саме опис бізнес-логіки. У моделі модуля вивчення реалізовано 9 методів:

– метод завантаження фрази з бази даних (loadPhrase());

– метод завантаження запитання з бази даних (loadQuestion());

– метод завантаження першої відповіді з бази даних (getAnswerOne());

– метод завантаження другої відповіді з бази даних (getAnswerTwo());

– метод завантаження третьої відповіді з бази даних (getAnswerThree());

– метод завантаження четвертої відповіді з бази даних (getAnswerFour());

– метод перевірки правильності відповіді (checkAnswer());

– метод запису оцінок після проходження гри (writeMarks());

– метод завантаження зображення з бази даних (loadImage());

– метод завантаження положення першого маркера (getMrkOneLocation());

– метод завантаження положення другого маркера (getMrkTwoLocation());

– метод завантаження положення третього маркера (getMrkThreeLocation()).

Програмний код моделі модуля вивчення можна побачити у додатку Е, а саме у підпункті «Лістинг MainGameModel».

На рис. 4.13 можна побачити вигляд програмного коду у редакторі IDE Visual Studio.

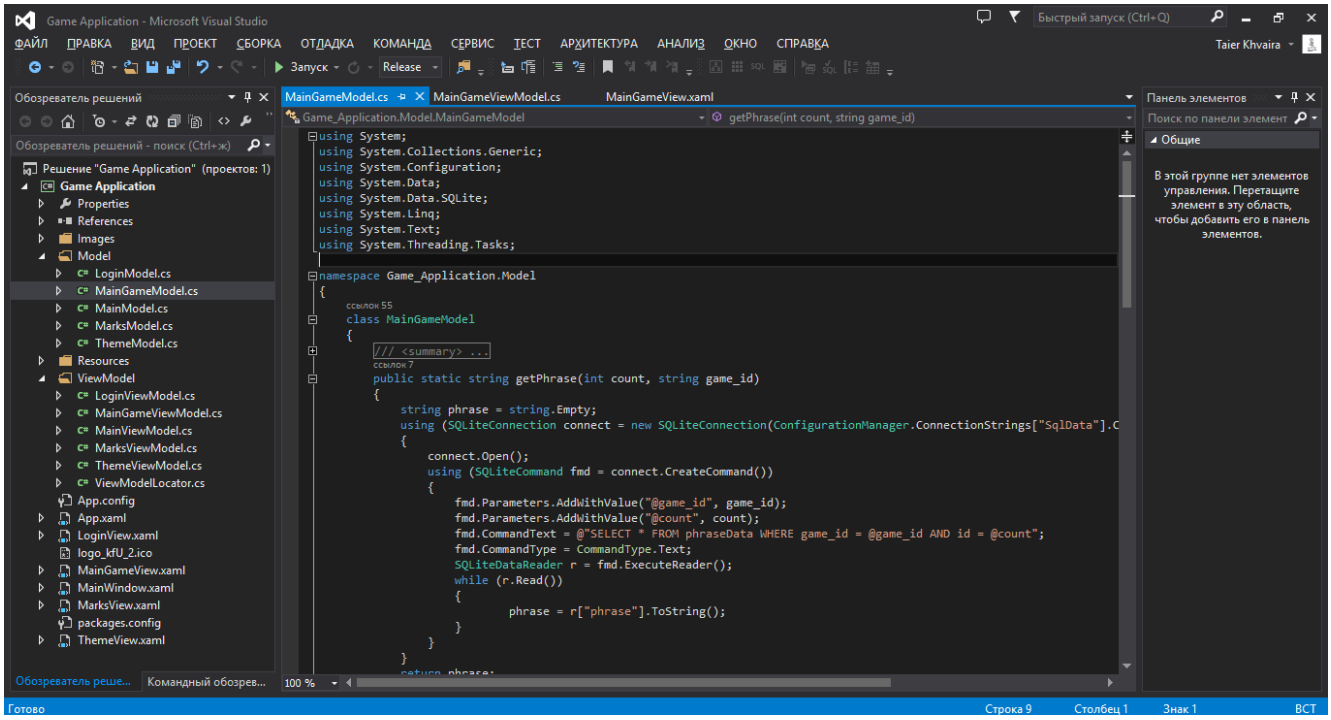


Рисунок 4.13 – Процес розробки бізнес-логіки модуля вивчення

4.5 Створення інсталятора для програмного додатку «Квест-гра для вивчення історії України учнями 5-го класу»

Для створення інсталятора було обрано NSIS. NSIS (Nullsoft Scriptable Install System)[25] – це система з відкритими кодом, що призначена для створення інсталяторів для операційної системи Windows. Вона створена, для зменшення розмірів та збільшення гнучкості і тому найкраще підходить для поширення додатку у мережі Інтернет.

Для створення скрипту використаємо програму HM NIS Edit, та за її допомоги створемо скрипт.

На рис. 4.14 можна побачити початок процесу створення скрипту. Встановлення назви, версії та розробника додатку.

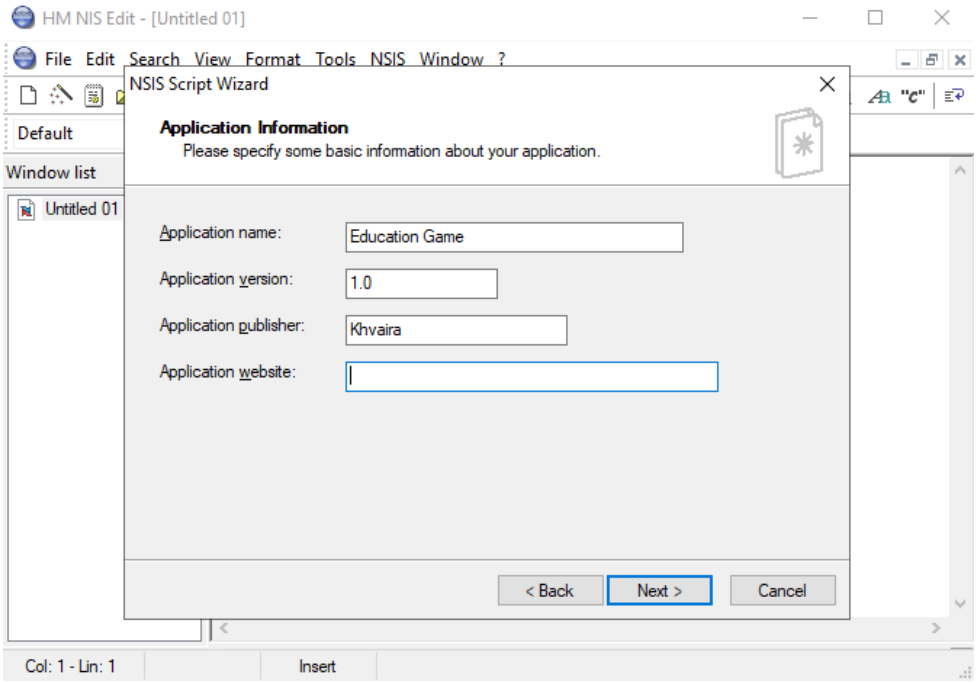


Рисунок 4.14 – Процес створення скрипту для компіляції інсталятора

Продовжуючи автоматичне створення треба вказати файли додатку що був нами створений. Цей процес можна побачити на рис. 4.15.

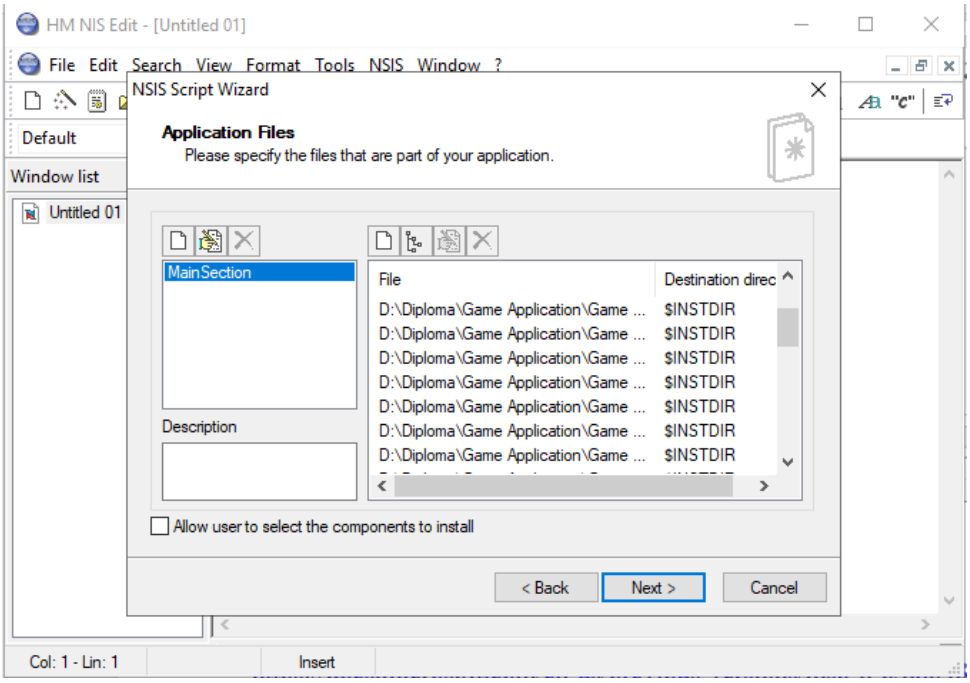


Рисунок 4.15 – Завантаження файлів створеного додатку у скрипт інсталятора

Після завершення процесу створення скрипту буде створено файл який треба скомпілювати у програмі NSIS. Це можна зробити навіть у середині програми де ми створювали скрипт, як це показано на рис. 4.16.

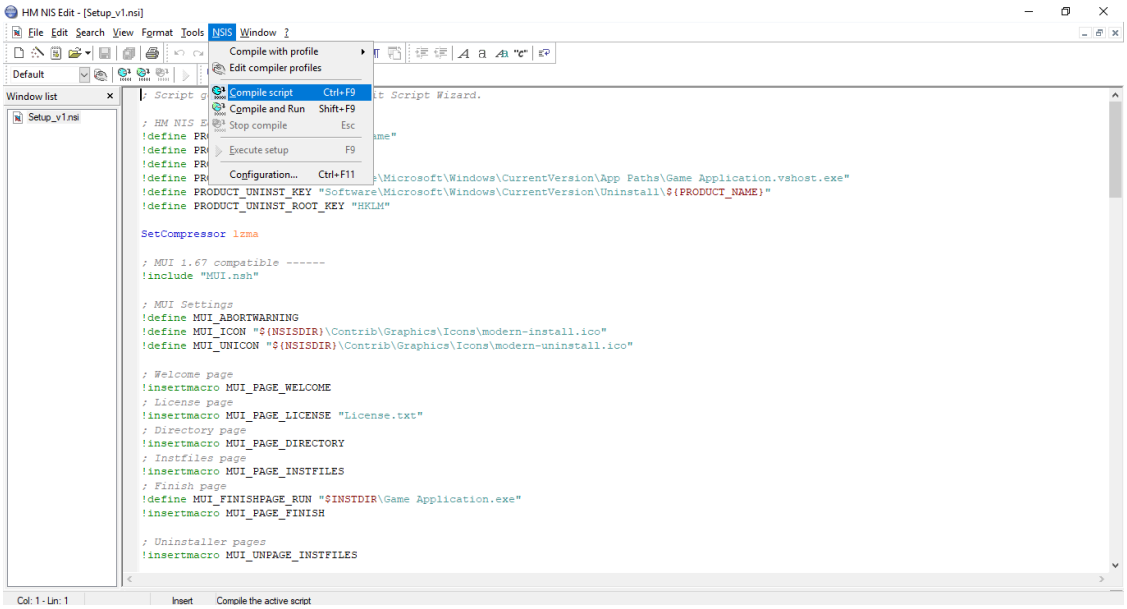


Рисунок 4.16 – Компіляція отриманого скрипта

У результаті компіляції буде створено інсталятор додатку.

ВИСНОВКИ

У ході виконання дипломного проекту була вирішена актуальна проблема – розроблення гейміфікованих засобів навчання для півищення рівня зацікавленості учнів. А саме розроблено гейміфікований додаток «Квест-гра для вивчення історії України учнями 5-го класу», який планується впровадити у процес навчання учнів у школі № 10 м. Суми.

В ході роботи був досліджений процес вивчення історії України в школі та розглянуті сучасні методи інтерактивного навчання. У результаті ретельного розгляду предметної області було розроблене технічне завдання на створення додатку. Були обрані методи та інструменти реалізації програмного додатку. Додаток був спроектований як сукупність моделей, а саме структурно-функціональної моделі процесу навчання, моделі типу «сутність-зв’язок» розроблюваної бази даних та моделей UML (моделі варіантів використання, моделі проектування та діаграми компонентів). Реалізований додаток у середовищі Visual Studio мовою програмування C# з використанням архітектурного патерну MVVM. Додаток було протестовано, в ході чого було підтверджено його працездатність.

Результати роботи були апробовані на науковій конференції IMA 2019. Детально з тезисами роботи можна ознайомитися у додатку Є. З публікацією можна ознайомитися за посиланням [26].

Також створений програмний продук буде впроваджено у роботу Сумської спеціалізованої школи № 10. Акт впровадження знаходиться у додатку Ж.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 О. Рома. Гра по-новому, навчання по-іншому: методичний підручник. Київ, 2018. 44 с. URL: https://mon.gov.ua/storage/app/media/nova-ukrainska-shkola/LEGO/po-novomu-navchannya-po-inshomu.pdf (дата звернення 04.04.19).

2 N. Yakovleva, E. Yakovlev. Interactive teaching methods in contemporary higher education: підручник. Челябінськ, 2011. 80 с.

3 Defense Acquisition University Press. Systems Engineering Fundamentals: підручник. Вірджинія, 2001. 222 с. URL: https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-885j-aircraft-systems-engineering-fall-2005/readings/sefguide_01_01.pdf (дата звернення 11.04.19).

4 G. Booch, J. Rumbaugh, I. Jacobson. Unified Modeling Language User Guide, The 2nd Edition: підручник. Addison-Wesley, 2005. 496 с.

5 Ian Sommerville. Software Engineering, 8th edition: підручник. Pearson Education, 2007. 843 с.

6 D. Goodwin. "Modelling and Simulation, 26 с." (PDF): лекція. The University of Warwick. 2015. 44 с. URL: https://warwick.ac.uk/fac/sci/physics/research/condensedmatt/imr_cdt/students/david_goodwin/teaching/modelling/l3_objectorientation.pdf (дата звернення 28.11.15)

7 S. Bagui, R. Earp. Database design using entity-relationship diagrams: підручник. Hoboken, NJ, 2011. 362 с.

8 J. Gosling, H. McGilton. The Java Language Environment: документація. Sun Microsystems, Inc., 1996. URL: https://www.oracle.com/technetwork/java/index-136113.html (дата звернення 06.04.19).

9 Timothy A. Budd. C++ for Java Programmers: підручник. Oregon State University, Corvallis, Oregon, 1998. 29 с. URL: http://web.engr.oregonstate.edu/~budd/Books/cforj/info/preface.pdf (дата звернення 23.12.18).

10 C # /. NET History Lesson. James Kovacs 'Weblog: веб-сайт. URL: http://jameskovacs.com/2007/09/07/cnet-history-lesson/ (дата звернення 04.02.19).

11 What is JetBrains Rider. .NET TOOLS BLOG: веб-сайт. URL: https://blog.jetbrains.com/dotnet/2017/08/03/rider-2017-1-jetbrains-net-ide-hits-rtm/ (дата звернення 11.04.19).

12 What are the best C# IDEs? Slant: веб-сайт. URL: https://www.slant.co/topics/4118/viewpoints/2/~c-ides~visual-studio (дата звернення 11.04.19).

13 М. Прайс. C # 7 і .NET Core. Кросплатформена розробка для фахівців: підручник. Пітер, 2016. 640 с.

14 MVVM Light Toolkit. MVVM Light: веб-сайт. URL: http://www.mvvmlight.net/

15 What Is SQLite? Sqlite: веб-сайт. URL: https://www.sqlite.org/index.html (дата звернення 23.03.19).

16 .NET Framework Class Library. Javatpoint: веб-сайт. URL: https://www.javatpoint.com/net-framework-class-library (дата звернення 11.02.19).

17 Д. Ріхтер. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C#: підручник. Пітер, 2018. 896 с.

18 М. Мак-Дональд. WPF: Windows Presentation Foundation у .NET 3.5: підручник. Вильямс, 2008. 928 с.

19 IDEF0 – Part 1 (understanding it). Syque: веб-сайт. URL: http://www.syque.com/quality_tools/tools/Tools19.htm (дата звернення 19.04.19).

20 Основи роботи з пакетом BPWin. Студопедія: веб-сайт. URL: https://studopedia.org/10-17823.html (дата звернення 14.05.19).

21 Короткий опис ER-методу проектування реляційних баз даних. Кафедра алгоритмічних мов ВМК МГУ: веб-сайт. URL: http://al.cs.msu.su/system/files/ER_method.pdf (дата звернення 04.05.19).

22 F. Armour, G. Miller. Advanced Use Case Modeling: підручник. Software Systems. Addison-Wesley, 2001. 425 с.

23 М. Фаулер. UML. Основи: інструкція по стандартному мови об'єктного моделювання: підручник. Символ-Плюс, 2013. 192 с.

24 The MVVM Pattern. Microsoft: веб-сайт. URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10) (дата звернення 30.04.19).

25 Nullsoft Scriptable Install System. NSIS Wiki: веб-сайт. URL: https://nsis.sourceforge.io/Main_Page (дата звернення 24.05.19).

26 eSSUIR - Electronic Sumy State University Institutional Repository: веб-сайт. URL: http://er.chdtu.edu.ua/bitstream/ChSTU/304/1/IMA-2019.pdf (дата звернення 01.06.19).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку програмного продукту «Квест-гра для вивчення історії України учнями 5-го класу»

Суми 2019

1 Призначення й мета створення програмного продукту

1.1 Призначення програмного продукту

Програмний додаток повинен представляти квест-гру для вивчення історії України учнями 5-го класу.

1.2 Мета створення програмного продукту

Інтерактивне вивчення історії України учнями 5-го класу загальноосвітніх шкіл та автоматична перевірка якості засвоєння знань у вигляді практичних та контрольних робіт.

1.3 Зацікавлені особи проекту (стейкхолдери проекту)

Замовники:

– Сумська спеціалізована школа I-III ступенів №10 імені Героя Радянського Союзу О. Бутка в особі, директора школи Купреєвої Н.М.

Адреса фактична: вул. Новомістенська , 30, м. Суми, 40000

Телефон / Факс: +3 8(0542) 22-12-62

– Сумський державний університет в особі кандидата технічних наук, доцента Алексенко О.В.

Адреса фактична: вул. РИМСЬКОГО-КОРСАКОВА, 2, м. Суми, 40007

Телефон / Факс: +3 8(0542) 68-78-54

Розробник / Тестувальник:

– Студент Сумського державного університету, Хвайра Т.С.Т.

Адреса фактична: м. Суми

Телефон / Факс: +3 8(050) 877-61-09

– Студент Сумського державного університету, Пархоменко С.В.

Адреса фактична: м. Суми

Телефон / Факс: +3 8(050) 524-52-52

Користувачі:

– учні 5-го класу сумської спеціалізованої школи I-III ступенів №10 імені Героя Радянського Союзу О. Бутка;

– вчителі сумської спеціалізованої школи І-ІІІ ступенів №10 імені Героя Радянського Союзу О. Бутка;

– адміністрація сумської спеціалізованої школи І-ІІІ ступенів №10 імені Героя Радянського Союзу О. Бутка.

1.4 Цільова аудиторія

До цільової аудиторії програмного продукту можна віднести наступні групи:

– вчителі – автоматизація навчання Історії України учнів 5-го класу;

– учні – покращення успішності; ефективніше засвоєння знань;

– адміністрація – контроль рівня засвоєння отриманої на уроці учнями 5-го класу інформації.

2 Вимоги до програмного продукту

2.1 Вимоги до програмного продукту в цілому

2.1.1 Вимоги до структури й функціонування програмного продукту

Програмний продукт повинен бути реалізований у вигляді десктоп-додатку. Додаток повинен складатися із взаємозалежних модулів із чітко розділеними функціями. Основний функціонал програмного додатку полягає у інтерактивному проходженні уроків історії України, виконання контрольних та практичних робіт. Інтерактивне проходження уроків буде реалізоване методом, який полягає у проходженні завдань для подальшого просування по уроку. Наприклад учень обрав тему уроку та розпочав проходження, після отримання певної інформації перед учнем з’явиться завдання яке він повинен виконати для продовження гри. Більш детально про структуру та функції модулів у пункті 2.2.2.1 Структура додатку.

2.1.2 Вимоги до персоналу

Для підтримки додатку від персоналу не повинно вимагатися спеціальних технічних навичок, знання технологій або програмних продуктів, за винятком загальних навичок роботи з персональним комп'ютером. Оновлення додатку відбуватиметься через мережу Інтернет автоматично.

2.1.3 Вимоги до розмежування доступу

Інформація, розташовувана у додатку, є загальнодоступною.

Користувачів сайту можна розділити на 2 групи відповідно до прав доступу:

– учень;

– вчитель.

Вчитель має доступ до всього функціоналу додатку який описаний у пункті 2.2.1 Вимоги до структури й функціонування програмного продукту, у тому числі має доступ до усіх оцінок.

Учень має доступ до всього функціоналу додатку який описаний у пункті 2.2.1 Вимоги до структури й функціонування програмного продукту, та доступ до своїх оцінок.

Доступ до додатку повинен здійснюватися з використанням унікального логіна й пароля.

2.2 Вимоги до функцій, виконуваних додатком

2.2.1 Основні вимоги

2.2.1.1 Структура додатку

Додаток повинен складатися з наступних модулів

– Вікно авторизації – початкова сторінка з полями авторизації;

– Меню – сторінка у вигляді пунктів меню яка містить інформацію про вибір тем, перегляд оцінок, інформацію про користувача;

– Модуль гри – основна частина контенту, представлена у вигляді інтерактивної гри с декількома режимами які обираються у меню вибору теми: режим гри – режим інтерактивного вивчення Історії України, режим практичної роботи – режим перевірки засвоєних знань за попередній урок, режим контрольної роботи – режим перевірки знань за певний період навчання.

Прототип інтерфейсу вікон авторизації, меню, гри можна побачити у пункті 2.2.1.6 Загальні вимоги.

2.2.1.2 Навігація

Користувацький інтерфейс додатку повинен забезпечувати наочне, інтуїтивно зрозуміле представлення структури розміщеної на ньому інформації, швидкий і логічний перехід до розділів і сторінок. Навігаційні елементи повинні забезпечувати однозначне розуміння користувачем їх змісту: посилання на сторінки повинні мати заголовок, умовні позначки відповідати загальноприйнятим. Графічні елементи навігації повинні бути мати альтернативний підпис.

Додаток повинен забезпечувати навігацію по всіх доступних користувачеві ресурсах і відображати відповідну інформацію.

2.2.1.3 Наповнення додатку

Модулі додатку повинні формуватися програмним шляхом на підставі інформації з бази даних.

Модифікація вмісту баз даних повинна здійснюватися за допомогою системи керування базами даних (DM Browser for SQLite), яка без застосування спеціальних навичок програмування (без використання програмування й спеціального кодування

або форматування) повинна передбачати можливість редагування інформаційного вмісту додатку.

2.2.1.4 Взаємозв’язок між модулями

Взаємозв'язок між модулями додатку представлено на рис. А.2.1.

Рисунок А.2.1 – Взаємозв'язок між модулями додатку

2.2.1.5 Функціональні можливості додатку

Додаток «Квест-гра для навчання» має містити у собі такі можливості:

– вибір теми;

– проходження інтерактивних уроків;

– виконання практичних робіт по пройденій темі;

– виконання контрольних робіт по пройденим темам;

– перегляд оцінок;

– автоматичне оцінювання практичних і контрольних робіт.

Стиль додатку можна описати як сучасний, діловий. У якості основних кольорів додатку рекомендується використовувати чорно-білі.

Додаток повинен бути зручний користувачам у плані використання й цікавий для багаторазового відвідування.

Розташування елементів вікна авторизації схематично показано на рис. А.2.2.

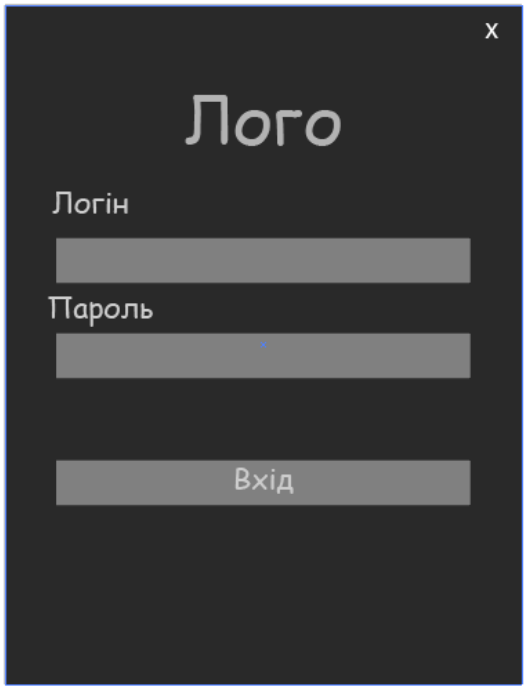


Рисунок А.2.2 – Макет вікна авторизації

Розташування елементів меню схематично показано на рис. А.2.3.

Рисунок А.2.3 – Макет вікна меню

Розташування елементів модуля гри схематично показано на рис. А.2.4.

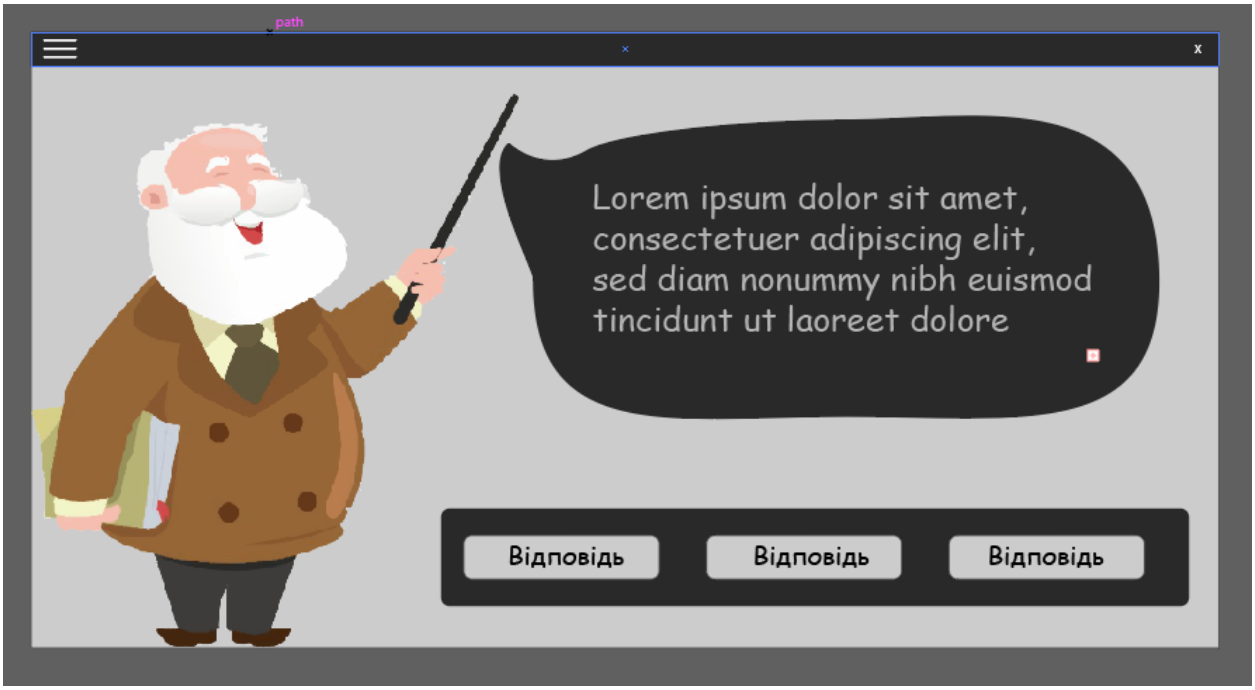


Рисунок А.2.4 – Макет модуля гри

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до супровідної документації

У вигляді супровідної документації проекту виступає інструкція користувача, та пояснювальна записка до створюваного програмного продукту.

2.3.2 Вимоги до структури даних

Логін та пароль користувача повинен зберігатися у вигляді тексту та типу даних text, який представляє собою текст довжиною до 65 кб.

Id користувача повинно зберігатися у вигляді числа і типу даних int, який представляє собою цілі числа від -2147483648 до 2147483647, і займає 4 байта.

Перелік тем повинен зберігатися у вигляді тексту і типу даних text, який представляє собою текст довжина до 65 кб.

Id тими, розділу і пункту повінню зберігатися у вигляді цифр і типу даних int, який представляє собою цілі числа від -2147483648 до 2147483647, і займає 4 байта.

Перелік питань і Відповідей модулю гри повінню зберігатися у вигляді тексту і типу даних text, який представляє собою текст довжина до 65 кб з розділовим знаком enter між ними.

2.3.3 Вимоги до інформаційного забезпечення

У вигляді інформаційного забезпечення виступають UML діаграми, діаграми класів, діаграми варіантів використання які будуть створенні під час планування проекту.

Реалізація додатку відбувається з використанням:

– SQLite

– Visual Studio 2013

2.3.4 Вимоги до лінгвістичного забезпечення

Додаток повинен бути виконаний українською мовою.

2.3.5 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– .Net Framework 4.6

2.3.6 Вимоги до апаратного забезпечення

Апаратне забезпечення серверної частини повинне задовольняти наступним вимогам:

– Windows 10, 64-bit/ Windows 8.0, 64-bit/ Windows 7 SP1, 64-bit.

Монітор з роздільною здатністю 1366 x 768 або вище.

– Клавіатура та миша.

– Доступ до мережі Інтернет.

Апаратне забезпечення клієнтської частини повинне забезпечувати підтримку програмного забезпечення клієнтської частини, зазначеного в п. 2.2.3.

3 Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення додатку наведено в табл. А.3.1.

Таблиця А.3.1 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Проектування: Створення юзкейс-діаграм, структури БД, проектування класів	5 днів
2	Розробка інтерфейсу: Створення WPF вікон та попереднє розташування елементів	3 дні
3	Розробка модуля авторизації: Розроблення та реалізація вікна авторизації користувача	2 дні
4	Розробка модуля меню: Розроблення та реалізація вікна меню додатку	2 дні
5	Розробка модуля вибору тем: Розроблення та реалізація модуля вибору тем	2 дні
6	Розробка модуля оцінок: Розроблення та реалізація модуля з оцінками учнів	2 дні
7	Розробка головного модуля: Розроблення та реалізація вікна гри. Реалізація запланованих режимів	2 дні

Продовження таблиці А.3.1 – Етапи створення додатку

8	Тестування: Тестування розроблених модулів. Проведення модульного та інтеграційного тестування.	4 дні
	Загальна тривалість робіт (з урахуванням резервного строку на налагодження й виправлення помилок) і строк закінчення проекту	32 дні

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

Для створення умов функціонування, при яких гарантується відповідність створюваного додатку вимогам ТЗ і можливість його ефективної роботи, в організації Замовника повинен бути проведений комплекс заходів на комп’ютерах, де має працювати програма:

– Забезпечити виконання вимог до програмного ти апаратного забезпечення.

– Встановити отриманий від Розробника інсталятор та виконати автоматичне встановлення програми.

– Запустити додаток за допомогою ярлика с робочого столу.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

1. Опис IT-проекту на фазі ініціалізації

1.1 Розробка концепції проекту

1.1.1 Ідентифікація ідеї проекту

У даний момент часу освіта та інформаційні технології тісно пов’язані. Міністерство освіти все більше і більше спонукає викладачів користуватися мультимедійними засобами на заняттях. Але у сьогоднішніх реаліях технічні засоби не використовують на сто відсотків.

Викладачі звикли до традиційних методів навчання таких як лекції, практичні заняття із використанням паперових навчально-методичних матеріалів. Але у час новітніх технологій, можливості яких зараз дозволяють додати інтерактиву у вивчення предмету, традиційні методи не ефективні. Саме тому створення програмного додатку для вивчення історії у ігровому форматі є актуальною задачею для старту використання IT-технологій в освітніх процесах у повному обсязі.

Усе вище зазначене дозволяє сформулювати мету проекту – це створення квест-гри для вивчення історії України учнями 5-го класу.

1.1.2 Деталізація мети проекту методом SMART

Мета проекту: створити програмний продукт, що дозволить вивчати історію України учнями 5-го класу у загальноосвітніх школах, виконувати перевірку знань та рівня засвоєння інформації на уроку, використовуючи мультимедійне забезпечення. Проект буде виконано вчасно, відповідно до календарного плану проекту.

1.1.3 Опис функціонування продукту IT-проекту

Для початку роботи користувач мусить обрати розділ та тему заняття. Після чого відкриється головне вікно гри с завданнями обраного розділу та теми. В

залежності від обраного типу буде розпочато виконання програми. Повний перелік вимог до програми наведений у технічному завданні (додаток А).

Для коректної роботи програмного продукту рекомендовані наступні мінімальні системні вимоги:

– операційна система Windows 7/8/10;

– процесор з частотою 1ГГц або кращий;

– оперативна пам'ять розміром від 2 ГБ;

– .Net Framework версії 4 або вище.

1.2 Техніко-економічне дослідження

1.2.1 Дослідження продукту ІТ-проекту, організації, ринку, регіону

Розроблений протягом виконання проекту програмний продукт необхідний для впровадження у діяльність загальноосвітньої школи №10.

Результати проекту можуть бути використані у діяльності інших освітянських закладах регіону, наприклад у коледжах, школах та навіть у вищих навчальних закладах.

Розвиток використання мультимедійних засобів у навчанні сприяє активізації попиту на подібні програмні продукти, тому що саме через такий спосіб пізнання інформація засвоюється найкраще.

1.2.2 Дослідження проекту в соціально-економічному, технічному, комерційному, економічному, фінансовому, соціально-інституційному аспектах

У соціально-економічному аспекті програмний продукт підвищить якість викладання у освітніх закладах тому що використання мультимедійних засобів підвищує інтерес учнів до представленого матеріалу.

У технічному аспекті створення програмного продукту дозволить більш ефективно використовувати наявні технічні засоби у навчальних закладах, тобто на сто відсотків використовувати їх для навчання школярів.

У комерційному, економічному та фінансовому аспектах програмний продукт значно знизить витрати на друкування книг для вивчення, контрольних робіт для

перевірки знань, все це доступно для використання у електронному вигляді, тому потреби друкувати щось немає.

У соціально-інституційному аспекті програмний продукт не потребує найму більш кваліфікованих робітників тому що використання продукту відбувається на інтуїтивному рівні. Потреби у додатковому навчанні користуванню програмою немає.

1.2.3 Оцінка цінності, життєздатності, економічної ефективності та життєсталості ІТ-проекту

Під цінностями прийнято розуміти все те, що людям особливо дорого, корисно, необхідно для життя, до чого ставляться з хвилюванням, повагою, визнанням, пошаною. Цінностей безліч. Це можуть бути думки, почуття, погляди, інтереси, принципи поведінки або будь-які предмети програмні продукти, явища та їх властивості.

Життєздатність проекту зумовлена швидким ростом популярності використання мультимедійних засобів у навчанні. Усі новітні програми Міністерства освіти і науки України спонукають вчителів використовувати комп’ютер.

1.3 Підготовка оціночного висновку

1.3.1 Опис причини ініціалізації відібраної після експертизи альтернатив ІТ-проекту

На даний момент у освіті вже використовують технічне забезпечення, але його можливості не розкрито на сто відсотків. Тому впровадження створюваного програмного продукту дуже доцільно, так як він допоможе у повному обсязі розкрити потенціал ІТ технологій у навчанні. Це в свою чергу також допоможе:

– підвищити зацікавленість учнів у навчанні;

– підвищити відсоток засвоєного матеріалу протягом уроку;

– покращити показники засвоєння навчального матеріалу з Історії України учнями 5-го класу;

– автоматизувати перевірку контрольних та тестових робіт.

1.3.2 Попередній опис змісту проекту

– аналіз предметної області;

– постановка задачі;

– проектування десктопного додатку;

– програмна реалізація;

– тестування;

– запровадження у дію.

1.3.3 Опис обмежень та допущень проекту

Створюваний продукт на даному етапі можна буде використовувати лише для навчання учнів 5-го класу та лише для вивчення історії України.

Також програмний продукт має лише українську мову, людина яка не володіє українською не зможе використовувати продукт.

Для даного проекту є також обмеження у часі. Дата початку проекту 15.04.19. Проект повинен бути виконаний до 25.05.19.

2 Опис фази розроблення IT-проекту

2.1 Планування змісту структури робіт IT-проекту

Планування змісту структури робіт включає в себе розбиття дій або заходів, які ведуть до досягнення мети, на елементарні роботи. Таким чином розбивка цих дій триває доти, поки не відбувається виконання дій елементарних робіт.

Елементарні роботи – це роботи, які мають один чіткий результат, який використовується при прийнятті цієї роботи.

Зазвичай декомпозиція завершується тоді, коли для розкриття змісту потрібні вузькі фахівці, що знають технологічні особливості їх виконання.

Таким чином після розбиття дій на елементарні роботи отримаємо WBS діаграму даного проекту, зображену на рис. Б.2.1.

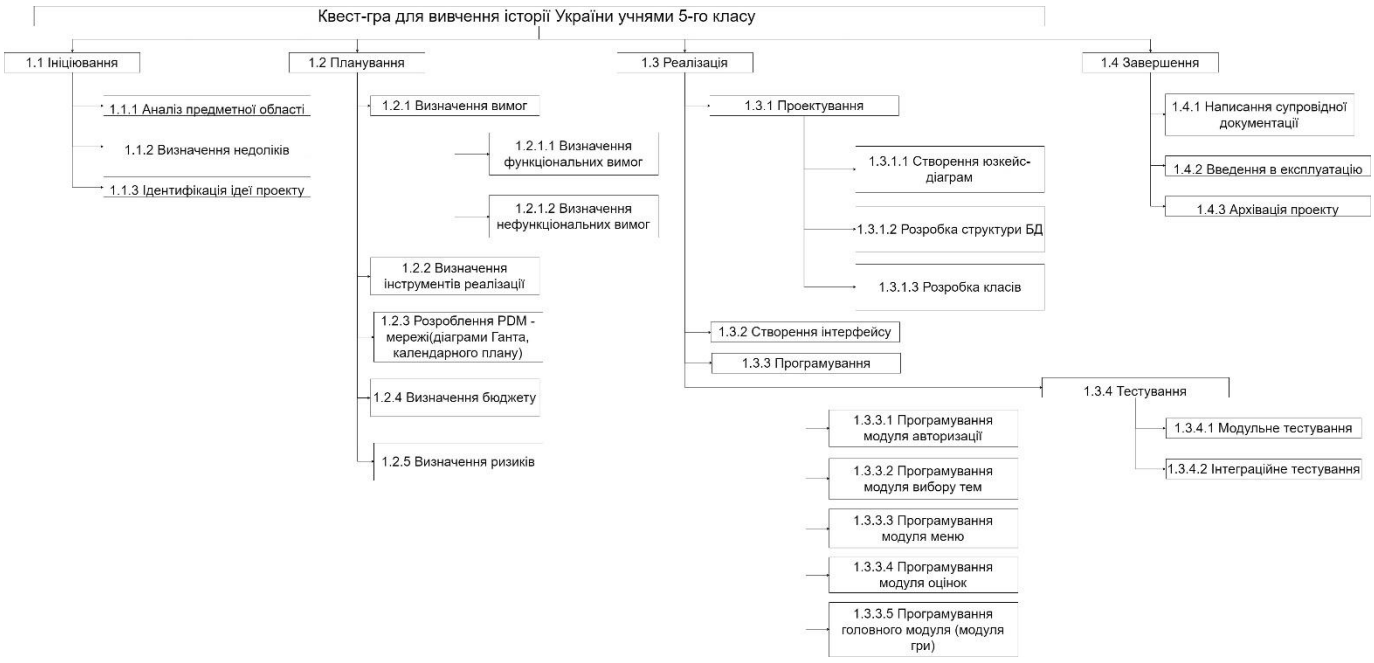


Рисунок Б.2.1 – WBS діаграма проекту

2.2 Планування структури організації, для впровадження готового проекту

Після побудови WBS розробляють організаційну структуру виконавців.

Організаційна структура представляє собою графічне відображення учасників проекту та їх відповідальних осіб, які задіяні в реалізації проекту. Основними учасниками розробки проекту виступають Алексенко О.В. у ролі куратору проекту, Хвайра Т.С.Т у ролі програміста, та Пархоменко С.В. у ролі тестувальника.

Таким чином на основі побудованої раніше WBS діаграми(рис. Б.2.1) будуємо OBS- структуру проекту, яка зображена на рис. Б.2.2.

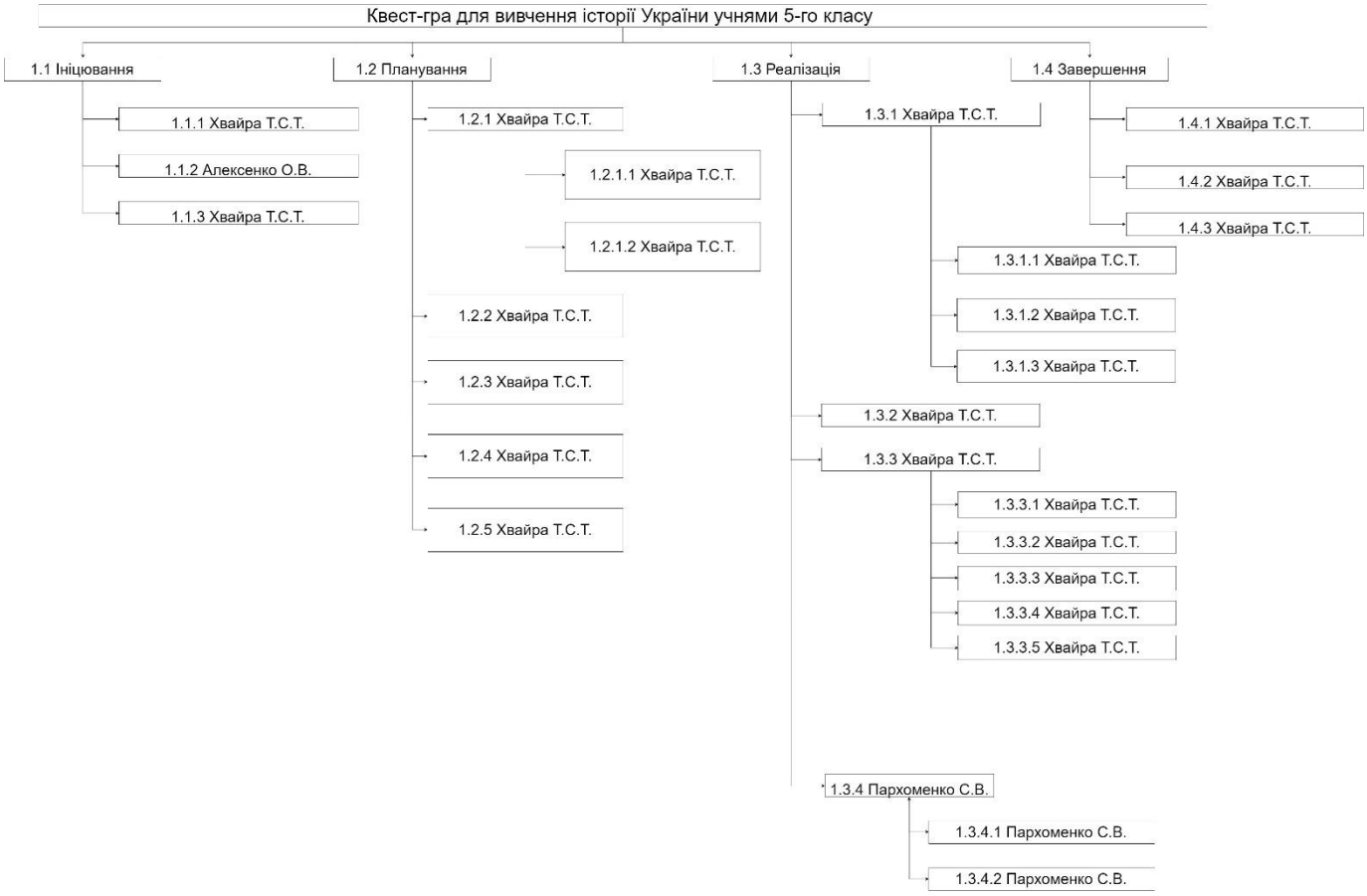


Рисунок Б.2.2 – OBS-структура проекту

2.3 Побудова матриці відповідальності

На підставі OBS та WBS структур будують матрицю відповідальності проекту. Матриця відповідальності закріплює за кожною елементарною роботою виконавця.

Таким чином на основі побудованої раніше WBS діаграми (рис. Б.2.1) та OBS-структури проекту (рис. Б.2.2) будуємо матрицю відповідності, яка зображена у табл. Б.2.1.

Таблиця Б.2.1 – Матриця відповідальності

WBS \ OBS			Алексенко О.В.	Хвайра Т.С.Т.	Пархоменко С.В.
1.1	1.1.1			X	
	1.1.2		X		
	1.1.3			X	
1.2	1.2.1	1.2.1.1		X	
		1.2.1.2		X	
	1.2.2			X	
	1.2.3			X	
	1.2.4			X	
	1.2.5			X	
1.3	1.3.1	1.3.1.1		X	
		1.3.1.2		X	
		1.3.1.3		X	
	1.3.2			X	
	1.3.3	1.3.3.1		X	
		1.3.3.2		X	
		1.3.3.3		X	
		1.3.3.4		X	
		1.3.3.5		X	
	1.3.4	1.3.4.1			X
		1.3.4.2			X
1.4	1.4.1			X	
	1.4.2			X	
	1.4.3			X	

2.4 Розробка календарного плану, діаграми Ганта, та PDM-мережі

Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткової мережевої моделі, а також, проекту в цілому з урахуванням вихідних та святкових днів, будують календарний графік робіт.

Він є реальним розподілом робіт з пакету за календарними датами, тобто своєрідним розкладом виконання робіт.

За допомогою програмного додатку автоматично буде побудована діаграма Ганта, яка представляє собою прямокутники робіт розташованих на шкалі часу, та мережевий графік, який показує відношення робіт та залежності їх друг від друга.

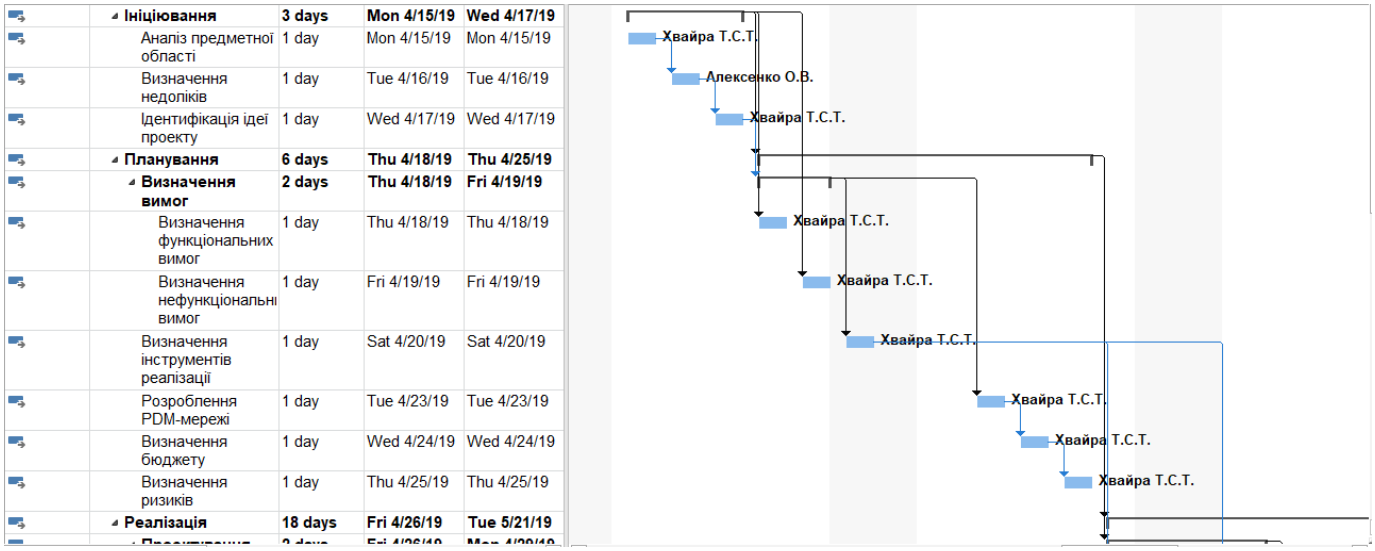
На основі створеної раніше WBS діаграми був створений календарний план зображений на рисунках Б.2.3 та Б.2.4, графік Ганта частина якого зображена на рис. Б.2.5, та мережевий графік частина якого зображена на рис. Б.2.6.

	Квест-гра для вивчення історії України учнями 5-го класу	**32 days**	**Mon 4/15/19**	**Sat 5/25/19**			**5,632.00 ₴**	**0%**
	Ініціювання	**3 days**	**Mon 4/15/19**	**Wed 4/17/19**			**1,024.00 ₴**	**0%**
	Аналіз предметної області	1 day	Mon 4/15/19	Mon 4/15/19		Хвайра Т.С.Т.	72.00 ₴	0%
	Визначення недоліків	1 day	Tue 4/16/19	Tue 4/16/19	3	Алексенко О.В	880.00 ₴	0%
	Ідентифікація ідеї проекту	1 day	Wed 4/17/19	Wed 4/17/19	4	Хвайра Т.С.Т.	72.00 ₴	0%
	Планування	**6 days**	**Thu 4/18/19**	**Thu 4/25/19**	**5**		**432.00 ₴**	**0%**
	Визначення вимог	**2 days**	**Thu 4/18/19**	**Fri 4/19/19**	**5**		**144.00 ₴**	**0%**
	Визначення функціональних вимог	1 day	Thu 4/18/19	Thu 4/18/19	2	Хвайра Т.С.Т.	72.00 ₴	0%
	Визначення нефункціональних вимог	1 day	Fri 4/19/19	Fri 4/19/19	2	Хвайра Т.С.Т.	72.00 ₴	0%
	Визначення інструментів реалізації	1 day	Sat 4/20/19	Sat 4/20/19	7	Хвайра Т.С.Т.	72.00 ₴	0%
	Розроблення PDM-мережі	1 day	Tue 4/23/19	Tue 4/23/19	7	Хвайра Т.С.Т.	72.00 ₴	0%
	Визначення бюджету	1 day	Wed 4/24/19	Wed 4/24/19	11	Хвайра Т.С.Т.	72.00 ₴	0%
	Визначення ризиків	1 day	Thu 4/25/19	Thu 4/25/19	12	Хвайра Т.С.Т.	72.00 ₴	0%
	Реалізація	**20 days**	**Fri 4/26/19**	**Tue 5/21/19**	**13**		**3,888.00 ₴**	**0%**
	Проектування	**2 days**	**Fri 4/26/19**	**Mon 4/29/19**	**6**		**216.00 ₴**	**0%**
	Створення юзкейс-діаграм	1 day	Sun 4/28/19	Mon 4/29/19	10	Хвайра Т.С.Т.	72.00 ₴	0%
	Розробка структури БД	1 day	Fri 4/26/19	Sat 4/27/19	10	Хвайра Т.С.Т.	72.00 ₴	0%
	Розробка класів	1 day	Fri 4/26/19	Fri 4/26/19	10	Хвайра Т.С.Т.	72.00 ₴	0%
	Створення інтерфейсу	3 days	Tue 4/30/19	Thu 5/2/19	15	Хвайра Т.С.Т.	216.00 ₴	0%
	Програмування	**11 days**	**Fri 5/3/19**	**Wed 5/15/19**	**19**		**3,168.00 ₴**	**0%**
	Програмування модуля меню	9 days	Fri 5/3/19	Sun 5/12/19	19	Хвайра Т.С.Т.	648.00 ₴	0%
	Програмування модуля вибору тем	9 days	Fri 5/3/19	Sat 5/11/19	19	Хвайра Т.С.Т.	648.00 ₴	0%
	Програмування модуля оцінок	9 days	Fri 5/3/19	Sat 5/11/19	19	Хвайра Т.С.Т.	648.00 ₴	0%
	Програмування модуля авторизації	9 days	Fri 5/3/19	Sat 5/11/19	19	Хвайра Т.С.Т.	576.00 ₴	0%
	Програмування головного модуля(модуля гри)	9 days	Fri 5/3/19	Sat 5/11/19	19	Хвайра Т.С.Т.	648.00 ₴	0%

Рисунок Б.2.3 – Календарний графік проекту

	Інтеграційне тестування	2 days	Mon 5/20/19	Tue 5/21/19	20	Пархоменко С.В	144.00 ₴	0%
	Завершення	**4 days**	**Wed 5/22/19**	**Sat 5/25/19**	**14**		**288.00 ₴**	**0%**
	Написання супровідної документації	2 days	Wed 5/22/19	Thu 5/23/19	14	Хвайра Т.С.Т.	144.00 ₴	0%
	Введення в експлуатацію	1 day	Fri 5/24/19	Fri 5/24/19	29	Хвайра Т.С.Т.	72.00 ₴	0%
	Архівація проекту ПЗ	1 day	Sat 5/25/19	Sat 5/25/19	30	Хвайра Т.С.Т.	72.00 ₴	0%

Рисунок Б.2.4 – Календарний графік проекту



Ініціювання	**3 days**	**Mon 4/15/19**	**Wed 4/17/19**
Аналіз предметної області	1 day	Mon 4/15/19	Mon 4/15/19
Визначення недоліків	1 day	Tue 4/16/19	Tue 4/16/19
Ідентифікація ідеї проекту	1 day	Wed 4/17/19	Wed 4/17/19
Планування	**6 days**	**Thu 4/18/19**	**Thu 4/25/19**
Визначення вимог	**2 days**	**Thu 4/18/19**	**Fri 4/19/19**
Визначення функціональних вимог	1 day	Thu 4/18/19	Thu 4/18/19
Визначення нефункціональні вимог	1 day	Fri 4/19/19	Fri 4/19/19
Визначення інструментів реалізації	1 day	Sat 4/20/19	Sat 4/20/19
Розроблення PDM-мережі	1 day	Tue 4/23/19	Tue 4/23/19
Визначення бюджету	1 day	Wed 4/24/19	Wed 4/24/19
Визначення ризиків	1 day	Thu 4/25/19	Thu 4/25/19
Реалізація	**18 days**	**Fri 4/26/19**	**Tue 5/21/19**

Рисунок Б.2.5 – Діаграма Ганта

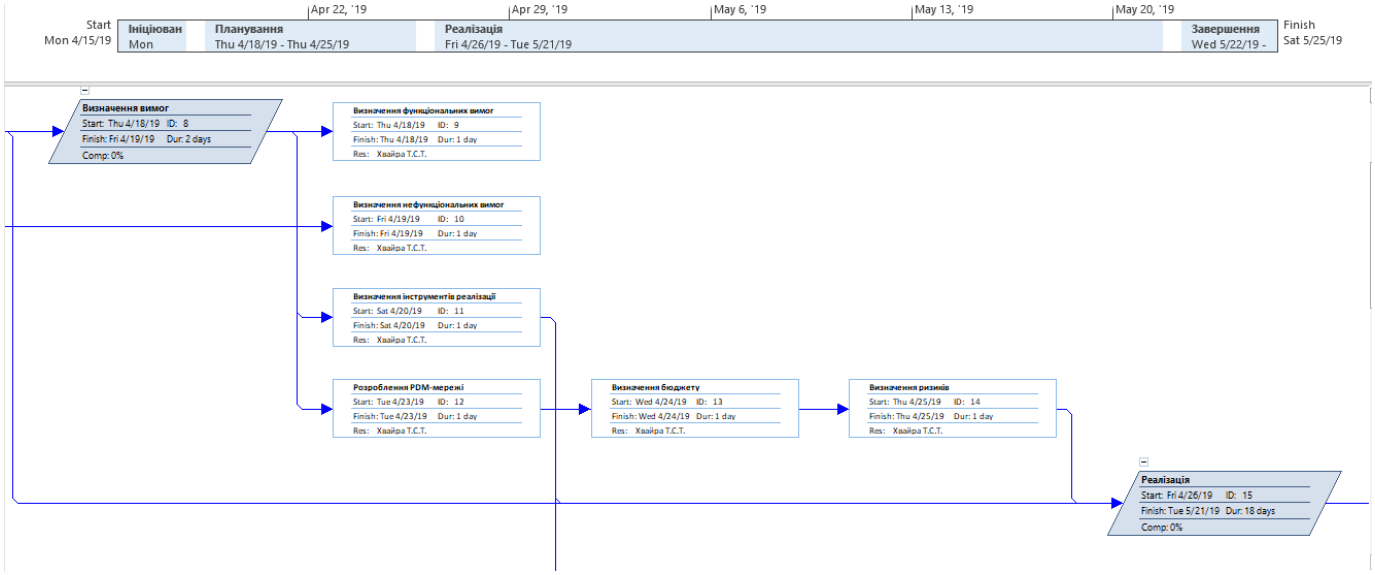


Рисунок Б.2.6 – Мережевий графік проекту

2.5 Визначення ресурсів та бюджету проекту

У ролі ресурсів виступають усі учасники проекту: Алексенко О.В., Хвайра Т.С.Т., Пархоменко С.В.. Кожен ресурс виступає у ролі виконавця заданих простих робіт у проекті. Кожен ресурс може виконувати лише одну роботу одночасно.

Для розрахунку бюджету проекту були використані стандартні ставки часу роботи кожного виконавця. Ціна години роботи студента коштує 9 гривень. Ціна години роботи куратору проекту коштує 109 гривень. Бюджетну вартість кожної з

робіт розраховують на підставі кількості використаного ресурсу, його вартості та тривалості використання.

За допомогою програмного додатку MS Project була встановлена ціна ресурсів за годину праці та була розрахована ціна кожної роботи що в сумі дало ціну проекту що складає 5056 гривень. Перелік ресурсів з цінами на годину праці зображено на рис. Б.2.7, ціна кожної роботи та сумарна ціна проекту зображено на рис. Б.2.8 та на рис. Б.2.9.

	Хвайра Т.С.Т.	Work		X	9.00 ₴/hr	9.00 ₴/hr	0.00 ₴	Prorated	Стандартный
	Алексенко О.В.	Work		A	110.00 ₴/hr	110.00 ₴/hr	0.00 ₴	Prorated	Стандартный
	Пархоменко С.В.	Work		П	9.00 ₴/hr	9.00 ₴/hr	0.00 ₴	Prorated	Стандартный

Рисунок Б.2.7 – Ресурси проекту

Название задачи	Cost
Квест-гра для вивчення історії України учнями 5-го класу	**5,056.00 ₴**
Ініціювання	**1,024.00 ₴**
Аналіз предметної області	72.00 ₴
Визначення недоліків	880.00 ₴
Ідентифікація ідеї проекту	72.00 ₴
Планування	**432.00 ₴**
Визначення вимог	**144.00 ₴**
Визначення функціональних вимог	72.00 ₴
Визначення нефункціональних вимог	72.00 ₴
Визначення інструментів реалізації	72.00 ₴
Розроблення PDM-мережі	72.00 ₴
Визначення бюджету	72.00 ₴
Визначення ризиків	72.00 ₴
Реалізація	**3,312.00 ₴**
Проектування	**216.00 ₴**
Створення юзкейс-діаграм	72.00 ₴
Розробка структури БД	72.00 ₴
Розробка класів	72.00 ₴
Створення інтерфейсу	216.00 ₴
Програмування	**2,592.00 ₴**
Програмування модуля меню	648.00 ₴
Програмування модуля вибору тем	648.00 ₴

Рисунок Б.2.8 – Ціна проекту

Програмування модуля оцінок	648.00 ₴
Програмування модуля авторизації	576.00 ₴
Програмування головного модуля(модуля гри)	648.00 ₴
Тестування	**288.00 ₴**
Модульне тестування	144.00 ₴
Інтеграційне тестування	144.00 ₴
Завершення	**288.00 ₴**
Написання супровідної документації	144.00 ₴
Введення в експлуатацію	72.00 ₴
Архівація проекту ПЗ	72.00 ₴

Рисунок Б.2.9 – Ціна проекту

2.6 Управління ризиками проекту

Ретельне і детальне планування управління ризиками спрямовано на підвищення вірогідності досягнення цілей проекту. Даний процес повинен бути завершений на ранній стадії планування.

Якісний аналіз ризиків включає розстановку пріоритетів для ідентифікованих ризиків, результати якої використовуються потім у ході кількісного аналізу ризиків і планування реагування на ризики.

У ході якісного оцінювання використовується нечислова шкала вірогідності, наприклад:

– дуже слабкий вплив;

– слабкий вплив;

– середній вплив;

– сильний вплив;

– дуже сильний вплив.

Оцінку ризиків проводять за допомогою оцінки вірогідності і наслідків. Один з можливих прикладів подібної матриці, в якій показані тільки оцінки наслідків, представлений у табл. Б.2.2.

Таблиця Б.2.2 – Матриця оцінки наслідків

Ціль проекту	Дуже слабкий вплив – 0,05	Слабкий вплив - 0,01	Середній вплив - 0,2	Сильний вплив - 0,4	Дуже сильний вплив - 0,8
Вартість	Несуттєве збільшення бюджету	Збільшення бюджету до 10%	Збільшення бюджету на 10-20%	Збільшення бюджету на 20-30%	Збільшення бюджету більше, ніж на 40%
Терміни	Несуттєве збільшення календарного плану	Порушення календарного плану не більше ніж на 5%	Порушення календарного плану на 5-10%	Порушення календарного плану на 10-20%	Порушення календарного плану більш ніж на 20%

Продовження таблиці Б.2.2 – Матриця оцінки наслідків

Якість	Несуттєве зниження якості	Суттєве зниження якості	Зниження якості потребує узгодження з замовником	Зниження якості неприйнятне для замовника	Результат проекту повністю даремний

Існують три стратегії реагування на появу негативних ризиків, до яких відносяться:

– ухилення від ризику передбачає змінення плану управління проектом таким чином, щоб виключити ризик, убезпечити цілі проекту від наслідків ризику;

– передача ризику має на увазі перекладення негативних наслідків на третю сторону. Передача ризику просто переносити відповідальність за його управління іншій стороні, але ризик при цьому не зникає. Передача ризику є найбільш ефективною у відношенні фінансових ризиків. Передача ризику практично завжди передбачає виплату премії за ризик стороні, яка прийняла ризик на себе. В якості інструментів передачі ризиків використовуються страховки, гарантійні зобов’язання. У ряді випадків витрати на ризики можуть перекладатися на покупця або продавця, що оговорюється у контракті;

– зниження ризику передбачає зниження вірогідності наслідків негативної ризикованої події до прийнятних границь. Прийняття попереджувальних заходів зі зниження вірогідності настання ризику або його наслідків часто буває більш ефективним, ніж зусилля з усунення негативних наслідків, що здійснюються після настання події ризику. В якості прикладів заходів зі зниження ризиків можна привести: упровадження менш складних процесів, проведення великої кількості випробувань або вибір постачальника, поставки якого мають більш стабільний характер.

До стратегій реагування на позитивні ризики (сприятливі можливості) відносяться:

– використання. Дана стратегія призначена для усунення всіх невизначеностей, пов'язаних з ризиком верхнього рівня. До числа заходів прямого реагування відноситься, наприклад, залучення до участі у проекті більш талановитих спеціалістів для того, щоб скоротити строки виконання проекту або домогтися більш високої якості;

– спільне використання. Спільне використання позитивних ризиків передбачає передачу відповідальності третій стороні, яка здатна найкращим чином скористатися сприятливою можливістю в інтересах проекту. До числа таких заходів відносяться: утворення партнерства зі спільною відповідальністю за ризики команд, спеціалізованих компаній або спільних підприємств;

– посилення. При застосуванні цієї стратегії змінюється «розмір» сприятливої можливості шляхом підвищення вірогідності виникнення позитивного впливу.

ДОДАТОК В

Сценарії варіантів використання

1 Потік подій для ВВ Авторизація (1 Авторизація)

1.1 Передумови

Програма запущена, вікно «Авторизація» відкрите

1.2 Процес

Користувач (Editor або Customer) вводить логін у поле та пароль у поле «Пароль» вікна «Авторизація».

Коли поля заповнені користувач натискає кнопку «Ввійти», програма зчитує дані із полів (якщо поле «Логін» не заповнене виконується Е-1, якщо поле «Пароль» не заповнене виконується Е-2), порівнює значення полів із даними про авторизованих користувачів (у програмі передбачено 2 типи користувачів, дані авторизації яких наведені у табл. В.1.1).

Таблиця В.1.1 - Авторизовані користувачі програми

ID користувача	**Користувач**	**Поле «Логін»**	**Поле «Пароль»**
1	User	Stud	111
2	Admin	Teach	222

Програма отримує інформацію про тип доступу для користувача і закриває форму «Авторизація». Якщо дані збігаються із даними користувача «Учень», виконується S-1; якщо дані збігаються із даними користувача «Вчитель» виконується S-2; якщо дані не збігаються, виконується E-3.

1.3 Субпотоки

S-1: дані авторизації відповідають даним користувача «Учень». Програма відкриває вікно «Меню» із дозволеними для даного користувача функціями (перегляд оцінок лише для поточного користувача)

S-2: дані авторизації відповідають даним користувача «Вчитель». Програма відкриває вікно «Меню» із дозволеними для даного користувача функціями (перегляд повного списку оцінок учнів)

1.4 Альтернативні потоки

E-1: якщо поле «Логін» не заповнене, користувач інформується про необхідність введення даних у поле «Логін»;

E-2: якщо поле «Пароль» не заповнене, користувач інформується про необхідність введення даних у поле «Пароль»;

E-3: якщо дані авторизації не збігаються із заданими у програмі, користувач інформується про невірні дані авторизації**.**

2 Потік подій ВВ Перегляд оцінок (2 Перегляд оцінок)

2.1 Передумови

ВВ Авторизація(1 Авторизація) виконаний;

2.2 Процес

Користувач обирає в меню елемент «Перегляд оцінок» та натискає на нього. Якщо авторизація виконана із даними користувача «Учень», виконується S-1; якщо авторизація виконана із даними користувача «Вчитель» виконується S-2.

2.3 Субпотоки

S-1: програма відображує компонент «Перегляд оцінок» та формує список оцінок поточного користувача (якщо дані відсутні виконується E-1);

S-2: програма відображує компонент «Перегляд оцінок» та формує список оцінок усіх користувачів системи, що відносяться до типу «Учень» (якщо дані відсутні виконується E-1);

2.4 Альтернативні потоки

E-1: якщо програмі не вдалося завантажити дані з бази даних, користувач інформується про відсутність даних у базі**.**

3 Потік подій ВВ Вибір теми (3 Вибір теми)

3.1 Передумови

ВВ Авторизація(1 Авторизація) виконаний;

3.2 Процес

Користувач обирає в меню елемент «Вибір теми» та натискає на нього. Якщо авторизація виконана та дані доступні у базі виконується S-1.

3.3 Субпотоки

S-1: програма відображує компонент «Вибір теми» та формує список доступних тем та розділів (якщо дані відсутні виконується E-1);

3.4 Альтернативні потоки

E-1: якщо програмі не вдалося завантажити дані з бази даних, користувач інформується про відсутність даних у базі**.**

4 Потік подій ВВ Проходження гри (4 Проходження гри)

4.1 Передумови

ВВ Вибір теми (3 Вибір теми) виконаний;

4.2 Процес

Після вибору потрібного розділу користувач може вибрати один з трьох варіантів роботи: режим вивчення, режим практичної роботи, режим контрольної роботи.

− режим вивчення – режим інтерактивного діалогу між учнем та професором (уявним персонажем гри) для вивчення теоретичних відомостей з Історії України. Оцінка за проходження теми у цьому режимі роботи не виставляється;

− режим практичної роботи – режим інтерактивного діалогу учень-процесор, у якому учням будуть даватись практичні завдання (робота з картою, робота з джерелом, тестові завдання), спрямований на практичне засвоєння матеріалу. Передбачене оцінювання роботи;

− режим контрольної роботи – режим тестування учня (запитання-відповідь), передбачає оцінювання.

Якщо було обрано режим вивчення програма відкриває вікно «Гра» та ініціалізує початок вивчення. Після завершення роботи у режимі вивчення виконується S-1; Якщо було обрано режим тестування (контрольна або практична роботи) програма відкриває вікно «Гра» та ініціалізує початок тестування. Після завершення роботи у режимі контрольної або практичної роботи виконується S-2.

4.3 Субпотоки

S-1: програма закриває вікно «Гра», та повертається до вікна «Вибір теми»;

S-2: програма відображує результат проходження практичної або контрольної роботи та заносить їх у базу даних (якщо доступ до бази даних відсутній виконується E-1);

4.4 Альтернативні потоки

E-1: якщо програмі не вдалося отримати доступ до бази даних, користувач інформується про відсутність доступу до бази даних**.**

ДОДАТОК Г

Лістинг модуля авторизації

Лістинг LoginView

```
<UserControl x:Class="Game_Application.LoginView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:materialdesign="http://materialdesigninxaml.net/winfx/xaml/themes"
             xmlns:ap="clr-namespace:Game_Application.ViewModel"
             mc:Ignorable="d"
             Height="450"
             Width="800"
             DataContext="{Binding Login, Source={StaticResource Locator}}"
             d:DesignHeight="450" d:DesignWidth="800">

    <Grid Background="#FFEBEBEB" Width="800" Height="450">
        <StackPanel Width="400" Height="200" VerticalAlignment="Top" Margin="60">
            <Image Source="Images/logo.png"></Image>
        </StackPanel>
        <StackPanel Width="400" Height="300" VerticalAlignment="Bottom">
            <Grid Width="400" Height="300">
                <StackPanel Margin="20 20 20 0" VerticalAlignment="Top">
```

```
                    <TextBlock Text="Логін" FontSize="22"/>
                    <TextBox Height="25"  Margin="0 10 0 0" BorderBrush="Gray" Font-
Size="18"     Text="{Binding     Login,     Mode=OneWayToSource,     UpdateSourceTrig-
ger=PropertyChanged}"/>
                    <TextBlock Text="Пароль" FontSize="22" Margin="0 10 0 0"/>
                    <PasswordBox Height="25"   Margin="0 10 0 0" BorderBrush="Gray"
FontSize="18"                                ap:PasswordBoxAssistant.BindPassword="true"
ap:PasswordBoxAssistant.BoundPassword="{Binding   Path=Password,   Mode=TwoWay,   Up-
dateSourceTrigger=PropertyChanged}"/>
                    <Button  Command="{Binding  initLoginCommand}"  Padding="0"  Back-
ground="White" BorderBrush="Gray" Foreground="#000000" Margin="0 40 0 0">
                        <Button.Style>
                            <Style TargetType="{x:Type Button}">
                                <Setter Property="Template">
                                    <Setter.Value>
                                        <ControlTemplate   TargetType="{x:Type   But-
ton}">
                                            <Border      Background="{TemplateBinding
Background}" BorderBrush="Gray" BorderThickness="1">
                                                <ContentPresenter    HorizontalAlign-
ment="Center" VerticalAlignment="Center"/>
                                            </Border>
                                        </ControlTemplate>
                                    </Setter.Value>
                                </Setter>
                            </Style>
                        </Button.Style>
                        <StackPanel Orientation="Horizontal">
                            <materialdesign:PackIcon     Kind="Login"     Width="30"
Height="30" Padding="0" VerticalAlignment="Center"/>
                            <TextBlock Text="Увійти" FontSize="20" Padding="10 0"/>
                        </StackPanel>
                    </Button>
                </StackPanel>

            </Grid>
        </StackPanel>
    </Grid>
</UserControl>
```

Лістинг LoginViewModel

```
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Command;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Game_Application.Model;
using Game_Application;
using System.Windows.Input;
using System.Windows;
using System.Windows.Controls;
using GalaSoft.MvvmLight.Messaging;

namespace Game_Application.ViewModel
{

    /// <summary>
    /// class for binding passwordbox to viwemodel
    /// </summary>
    public static class PasswordBoxAssistant
```

```
    {
        public static readonly DependencyProperty BoundPassword =
            DependencyProperty.RegisterAttached("BoundPassword",      typeof(string),
typeof(PasswordBoxAssistant),   new   PropertyMetadata(string.Empty,   OnBoundPassword-
Changed));

        public  static  readonly  DependencyProperty  BindPassword  =  DependencyProper-
ty.RegisterAttached(
            "BindPassword", typeof(bool), typeof(PasswordBoxAssistant), new Proper-
tyMetadata(false, OnBindPasswordChanged));

        private static readonly DependencyProperty UpdatingPassword =
            DependencyProperty.RegisterAttached("UpdatingPassword",     typeof(bool),
typeof(PasswordBoxAssistant), new PropertyMetadata(false));

        private  static  void  OnBoundPasswordChanged(DependencyObject  d,  Dependency-
PropertyChangedEventArgs e)
        {
            PasswordBox box = d as PasswordBox;

            // only handle this event when the property is attached to a PasswordBox
            // and when the BindPassword attached property has been set to true
            if (d == null || !GetBindPassword(d))
            {
                return;
            }

            // avoid recursive updating by ignoring the box's changed event
            box.PasswordChanged -= HandlePasswordChanged;

            string newPassword = (string)e.NewValue;

            if (!GetUpdatingPassword(box))
            {
                box.Password = newPassword;
            }

            box.PasswordChanged += HandlePasswordChanged;
        }

        private  static  void  OnBindPasswordChanged(DependencyObject  dp,  Dependency-
PropertyChangedEventArgs e)
        {
            // when the BindPassword attached property is set on a PasswordBox,
            // start listening to its PasswordChanged event

            PasswordBox box = dp as PasswordBox;

            if (box == null)
            {
                return;
            }

            bool wasBound = (bool)(e.OldValue);
            bool needToBind = (bool)(e.NewValue);

            if (wasBound)
            {
                box.PasswordChanged -= HandlePasswordChanged;
            }

            if (needToBind)
            {
```

```
            box.PasswordChanged += HandlePasswordChanged;
        }
    }

    private static void HandlePasswordChanged(object sender, RoutedEventArgs e)
    {
        PasswordBox box = sender as PasswordBox;

        // set a flag to indicate that we're updating the password
        SetUpdatingPassword(box, true);
        // push the new password into the BoundPassword property
        SetBoundPassword(box, box.Password);
        SetUpdatingPassword(box, false);
    }

    public static void SetBindPassword(DependencyObject dp, bool value)
    {
        dp.SetValue(BindPassword, value);
    }

    public static bool GetBindPassword(DependencyObject dp)
    {
        return (bool)dp.GetValue(BindPassword);
    }

    public static string GetBoundPassword(DependencyObject dp)
    {
        return (string)dp.GetValue(BoundPassword);
    }

    public static void SetBoundPassword(DependencyObject dp, string value)
    {
        dp.SetValue(BoundPassword, value);
    }

    private static bool GetUpdatingPassword(DependencyObject dp)
    {
        return (bool)dp.GetValue(UpdatingPassword);
    }

    private static void SetUpdatingPassword(DependencyObject dp, bool value)
    {
        dp.SetValue(UpdatingPassword, value);
    }
}

public class LoginViewModel : ViewModelBase
{

    /// <summary>
    /// The <see cref="Login" /> property's name.
    /// </summary>
    public const string LoginPropertyName = "Login";

    private string _login = string.Empty;

    /// <summary>
    /// Sets and gets the Login property.
    /// Changes to that property's value raise the PropertyChanged event.
    /// </summary>
    public string Login
    {
        get
```

```
    {
        return _login;
    }

    set
    {
        Set(ref _login, value);
    }
}


/// <summary>
/// The <see cref="Password" /> property's name.
/// </summary>
public const string PasswordPropertyName = "Password";

private string _password = string.Empty;

/// <summary>
/// Sets and gets the Password property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Password
{
    get
    {
        return _password;
    }

    set
    {
        Set(ref _password, value);
    }
}

/// <summary>
/// The <see cref="Check" /> property's name.
/// </summary>
public const string CheckPropertyName = "Check";

private bool _check = false;

/// <summary>
/// Sets and gets the Check property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public bool Check
{
    get
    {
        return _check;
    }

    set
    {
        Set(ref _check, value);
    }
}

/// <summary>
/// The <see cref="Group" /> property's name.
/// </summary>
public const string GroupPropertyName = "Group";
```

```
        private string _group = string.Empty;

        /// <summary>
        /// Sets and gets the Group property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Group
        {
            get
            {
                return _group;
            }

            set
            {
                Set(ref _group, value);
            }
        }

        public RelayCommand initLoginCommand { get; private set; }

        private object login_txt = "Login";
        private object group_txt = "Group";

        public LoginViewModel()
        {
            initLoginCommand = new RelayCommand(initiateLogin);
        }

        private void initiateLogin()
        {
            Check = LoginModel.loginCheck(Login, Password);
            Messenger.Default.Send<NotificationMessage>(new NotificationMessage(this,
login_txt, Login));
            Group = LoginModel.getGroup(Login);
            if (Check == true)
            {
                MainWindow main = new MainWindow();
                Application.Current.MainWindow.Close();
                main.Show();
                main.Container.Content = new ThemeView();
                main.menu_list.Visibility = Visibility.Visible;
                main.logout_btn.Visibility = Visibility.Visible;
                Messenger.Default.Send<NotificationMessage>(new         Notification-
Message(this, group_txt, Group));
            }
            else
            {
                MessageBox.Show("Невірний логін або пароль!", "Помилка!");
            }
        }
    }
}
```

Лістинг LoginModel

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SQLite;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
namespace Game_Application.Model
{
    class LoginModel
    {
        /// <summary>
        /// function for check login info
        /// </summary>
        /// <param name="login"></param>
        /// <param name="pass"></param>
        /// <returns></returns>
        public static bool loginCheck(string login, string pass)
        {
            bool check = false;
            using (SQLiteConnection connect = new SQLiteConnection(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.CommandText = @"SELECT * FROM loginInfo";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (r["login"].ToString() == login && r["pass"].ToString() == pass)
                        {
                            check = true;
                        }
                    }
                }
            }
            return check;
        }

        /// <summary>
        /// function for getting user group
        /// </summary>
        /// <param name="login"></param>
        /// <returns></returns>
        public static string getGroup(string login)
        {
            string group = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnection(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.CommandText = @"SELECT * FROM loginInfo";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (r["login"].ToString() == login)
                        {
                            group = r["group"].ToString();
                        }
                    }
```

```
                }
            }
            return group;
        }
    }
}
```

ДОДАТОК Ґ

Лістинг модуля успішності

Лістинг MarksView

```
<UserControl x:Class="Game_Application.MarksView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             Height="450"
             Width="800"
             DataContext="{Binding Marks, Source={StaticResource Locator}}"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid Background="#FFEBEBEB" Width="800" Height="450">
        <StackPanel>
            <TextBlock Text="Табель успішності" FontSize="30" Margin="20"/>
        </StackPanel>
        <DataGrid Height="300" Width="750" ItemsSource="{Binding Marks}" AutoGener-
ateColumns="False" CanUserSortColumns="False" IsReadOnly="True">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Ім'я" Binding="{Binding Name}"/>
                <DataGridTextColumn Header="Прізвище" Binding="{Binding Surname}"/>
                <DataGridTextColumn Header="Успішність" Binding="{Binding MarksSet}"
/>
            </DataGrid.Columns>
        </DataGrid>
```

```
        <Button Command="{Binding updateMarksCommand}" Padding="0" Background="White"
BorderBrush="Gray" Foreground="#000000" Margin="0 0 0 25" Width="300" Height="30"
VerticalAlignment="Bottom" Content="Оновити успішність" FontSize="18">
            <Button.Style>
                <Style TargetType="{x:Type Button}">
                    <Setter Property="Template">
                        <Setter.Value>
                            <ControlTemplate TargetType="{x:Type Button}">
                                <Border Background="{TemplateBinding Background}"
BorderBrush="Gray" BorderThickness="1">
                                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                                </Border>
                            </ControlTemplate>
                        </Setter.Value>
                    </Setter>
                </Style>
            </Button.Style>
        </Button>

    </Grid>
</UserControl>
```

Лістинг MarksViewModel

```
using GalaSoft.MvvmLight;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Game_Application.Model;
using GalaSoft.MvvmLight.Messaging;
using System.Windows;
using System.Collections.ObjectModel;
using GalaSoft.MvvmLight.Command;

namespace Game_Application.ViewModel
{
    public class MarksViewModel : ViewModelBase
    {

        private ObservableCollection<MarksModel.Marks> _marks = new ObservableCollec-
tion<MarksModel.Marks>();
        public ObservableCollection<MarksModel.Marks> Marks
        {
            get
            {
                return _marks;
            }
            set
            {
                Set(ref _marks, value);
            }
        }

        public RelayCommand updateMarksCommand { get; private set; }
        public MarksViewModel()
        {
            updateMarksCommand = new RelayCommand(updateMarks);
            var marks = MarksModel.loadMarks(MainViewModel.Group, MainViewMod-
el.Login);
```

```
            foreach (var item in marks)
            {
                Marks.Add(item);
            }
        }

        private void updateMarks()
        {
            Marks.Clear();
            var marks = MarksModel.loadMarks(MainViewModel.Group, MainViewMod-
el.Login);
            foreach (var item in marks)
            {
                Marks.Add(item);
            }
        }
    }
}
```

Лістинг MarksModel

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Configuration;
using System.Data;
using System.Data.SQLite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Game_Application.Model
{
    public class MarksModel
    {
        public class Marks
        {
            public string Name { get; set; }
            public string Surname { get; set; }
            public string MarksSet { get; set; }

        }

        public static ObservableCollection<Marks> loadMarks(string group, string log-
in)
        {
            ObservableCollection<Marks> marksImport = new ObservableCollec-
tion<Marks>();
            if (group == "Admin")
            {
                using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
                {
                    connect.Open();
                    using (SQLiteCommand fmd = connect.CreateCommand())
                    {
                        fmd.CommandText = @"SELECT * FROM marksInfo";
                        fmd.CommandType = CommandType.Text;
                        SQLiteDataReader r = fmd.ExecuteReader();
                        while (r.Read())
                        {
                            marksImport.Add(new Marks()
```

```
                            {
                                Name = r["name"].ToString(),
                                Surname = r["surname"].ToString(),
                                MarksSet = r["marks"].ToString(),
                            });
                        }
                    }
                }
            }
            else
            {
                using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
                {
                    connect.Open();
                    using (SQLiteCommand fmd = connect.CreateCommand())
                    {
                        fmd.Parameters.AddWithValue("@login", login);
                        fmd.CommandText = @"SELECT * FROM marksInfo WHERE login =
@login";
                        fmd.CommandType = CommandType.Text;
                        SQLiteDataReader r = fmd.ExecuteReader();
                        while (r.Read())
                        {
                            marksImport.Add(new Marks()
                            {
                                Name = r["name"].ToString(),
                                Surname = r["surname"].ToString(),
                                MarksSet = r["marks"].ToString(),
                            });
                        }

                    }
                }
            }
            return marksImport;
        }
    }
}
```

ДОДАТОК Д

Лістинг модуля вибору теми

Лістинг ThemeView

```
<UserControl x:Class="Game_Application.ThemeView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             Height="450"
             Width="800"
             DataContext="{Binding Theme, Source={StaticResource Locator}}"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid Background="#FFEBEBEB" Width="800" Height="450">
        <StackPanel>
            <TextBlock Text="Оберіть розділ та тему" FontSize="30" Margin="20"/>
        </StackPanel>
        <TreeView x:Name="topicTree" Width="750" Height="300" Back-
ground="Transparent" FontSize="22">

        </TreeView>
    </Grid>
</UserControl>
```

Лістинг ThemeViewModel

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

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Game_Application.Model;
using GalaSoft.MvvmLight.Command;
using GalaSoft.MvvmLight.Messaging;

namespace Game_Application.ViewModel
{
    public class ThemeViewModel : ViewModelBase
    {
            TreeViewItem history = new TreeViewItem() { Header = "Історія України" };
            TreeViewItem fifthGrade = new TreeViewItem() { Header = "5-й клас" };
            history.Items.Add(fifthGrade);
            topicTree.Items.Add(history);

            TreeViewItem topic = new TreeViewItem();
            TreeViewItem paragraph = new TreeViewItem();

            List<string> topics = new List<string>();
            topics = ThemeModel.loadTopics();
            List<string> paragraphs = new List<string>();
            paragraphs = ThemeModel.loadParagraphs();

            foreach (string itemT in topics)
            {
                string[] idT = itemT.Split(' ');
                string itemTopic = itemT.Remove(0, 4);
                topic = new TreeViewItem() { Header = "" + itemTopic + "", };
                fifthGrade.Items.Add(topic);
                foreach (string itemP in paragraphs)
                {
                    string[] idP = itemP.Split(' ');
                    if (idT[0] == idP[0])
                    {
                        string itemParagraph = itemP.Remove(0, 4);
                        paragraph = new TreeViewItem() { Header = "" + itemParagraph
+ "" };
                        paragraph.Selected += TreeViewItem_Selected;
                        topic.Items.Add(paragraph);
                    }
                }
            }
        }

        private object game_txt = "Game_Id";
        private object type_txt = "Type";

        private void TreeViewItem_Selected(object sender, RoutedEventArgs e)
        {
            TreeViewItem itemP = (TreeViewItem)sender;

            string game_id = ThemeModel.checkContent(itemP.Header.ToString());
            string type = ThemeModel.getType(itemP.Header.ToString());
```

```
            Messenger.Default.Send<NotificationMessage>(new NotificationMessage(this,
game_txt, game_id));
            Messenger.Default.Send<NotificationMessage>(new NotificationMessage(this,
type_txt, type));

            if (game_id == null)
            {
                return;
            }
            else
            {
                MainGameView game = new MainGameView(game_id);
                if (game != null)
                {
                    game.Show();
                    var main = Window.GetWindow(this);
                    main.Close();
                }

            }
        }

    }
}
```

Лістинг ThemeModel

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SQLite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;


namespace Game_Application.Model
{
    class ThemeModel
    {
        /// <summary>
        /// method for getting topics info
        /// </summary>
        /// <returns></returns>
        public static List<string> loadTopics()
        {
            List<string> topicsImport = new List<string>();
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.CommandText = @"SELECT idt, topic FROM themeInfo";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
```

```
                        topicsImport.Add(Convert.ToString(r["idt"]) + " " +
(r["topic"]));
                    }
                }
            }
            return topicsImport;
        }

        /// <summary>
        /// method for getting paragraphs info
        /// </summary>
        /// <returns></returns>
        public static List<string> loadParagraphs()
        {
            List<string> paragraphsImport = new List<string>();
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.CommandText = @"SELECT idt, paragraph FROM paragraphInfo";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        paragraphsImport.Add(Convert.ToString(r["idt"] + " " +
r["paragraph"]));
                    }
                }
            }
            return paragraphsImport;
        }

        /// <summary>
        /// method for getting info about current theme
        /// </summary>
        /// <param name="paragraph"></param>
        /// <returns></returns>
        public static string checkContent(string paragraph)
        {
            string game_id = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.CommandText = @"SELECT game_id, paragraph FROM paragraphIn-
fo";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if ((string)r["paragraph"] == paragraph)
                        {
                            game_id = (string)r["game_id"];
                            break;
                        }
                    }
                    if (game_id == string.Empty)
                    {
```

```
                        MessageBox.Show("Такий параграф відсутній, оберіть інший
параграф.", "Помилка");

                    }
                }
            }
            return game_id;
        }

        /// <summary>
        /// method for getting type of game
        /// </summary>
        /// <param name="paragraph"></param>
        /// <returns></returns>
        public static string getType(string paragraph)
        {
            string type = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.CommandText = @"SELECT type, paragraph FROM paragraphInfo";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if ((string)r["paragraph"] == paragraph)
                        {
                            type = (string)r["type"];
                            break;
                        }
                    }
                }
            }
            return type;
        }
    }
}
```

ДОДАТОК Е

Лістинг модуля вивчення

Лістинг MainGameView

```
<Window x:Class="Game_Application.MainGameView"
             x:Name="GameWindow"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:materialdesign="http://materialdesigninxaml.net/winfx/xaml/themes"
             mc:Ignorable="d"
             Height="720"
             Width="1280"
             DataContext="{Binding Game, Source={StaticResource Locator}}"
             d:DesignHeight="720" d:DesignWidth="1280" WindowStyle="None" Resize-
Mode="NoResize" SizeToContent="Height" WindowStartupLocation="CenterScreen">

    <Window.Resources>
        <BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter" />
    </Window.Resources>

    <Grid Background="#FFEBEBEB" MouseDown="Grid_MouseDown">
        <StackPanel>
            <Grid Background="#3849F9" Height="50">
                <Grid Margin="5">
                    <StackPanel Orientation="Horizontal" HorizontalAlignment="Left">
```

```
                        <Button x:Name="menu_btn" ToolTip="Меню" Height="30"
Width="30" Padding="0" Background="#3849F9" BorderBrush="{x:Null}" Fore-
ground="#FFEEEEEE" Click="menu_btn_Click">
                            <materialdesign:PackIcon Kind="Menu" Width="30"
Height="30"/>
                        </Button>
                    </StackPanel>
                    <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
                        <Button x:Name="logout_btn" ToolTip="Вийти" Height="30"
Width="30" Padding="0" Background="#3849F9" BorderBrush="{x:Null}" Fore-
ground="#FFEEEEEE" Command="{Binding RestartWindowCommand, Mode=OneWay}" CommandPa-
rameter="{Binding ElementName=GameWindow}">
                            <materialdesign:PackIcon Kind="Logout"/>
                        </Button>
                        <Button x:Name="exit_btn_m" ToolTip="Вихід" Height="30"
Width="30" Padding="0" Background="#3849F9" BorderBrush="{x:Null}" Fore-
ground="#FFEEEEEE" Command="{Binding CloseWindowCommand, Mode=OneWay}" CommandParame-
ter="{Binding ElementName=GameWindow}">
                            <materialdesign:PackIcon Kind="Power" Width="25"
Height="25"/>
                        </Button>
                    </StackPanel>
                </Grid>
            </Grid>
        </StackPanel>
        <Grid Width="1280" Height="670" HorizontalAlignment="Right" Margin="0 50 0
0">
            <StackPanel Background="White" Width="850" Height="600" VerticalAlign-
ment="Center" HorizontalAlignment="Right" Margin="0 0 50 0">
                <Grid Width="550" Height="300" Margin="0 50 80 0" HorizontalAlign-
ment="Right">
                    <TextBlock Text="{Binding Text}"  FontSize="25" TextAlign-
ment="Center" TextWrapping="WrapWithOverflow"/>
                </Grid>
                <StackPanel Width="550" Height="220" VerticalAlignment="Bottom" Hori-
zontalAlignment="Right" Margin="0 0 80 0">
                    <Button x:Name="BtnOne" Width="500" Height="40" Margin="0 15 0
10" Content="{Binding BtnOneText}" Visibility="{Binding Path=IsEnabledBtn, Convert-
er={StaticResource BooleanToVisibilityConverter}}" FontSize="25" Command="{Binding
BtnOneOnClickCommand}" Padding="0" Background="White" BorderBrush="Gray">
                        <Button.Style>
                            <Style TargetType="{x:Type Button}">
                                <Setter Property="Template">
                                    <Setter.Value>
                                        <ControlTemplate TargetType="{x:Type But-
ton}">
                                            <Border Background="{TemplateBinding
Background}" BorderBrush="Gray" BorderThickness="2">
                                                <ContentPresenter HorizontalAlign-
ment="Center" VerticalAlignment="Center"/>
                                            </Border>
                                        </ControlTemplate>
                                    </Setter.Value>
                                </Setter>
                            </Style>
                        </Button.Style>
                    </Button>
                    <Button x:Name="BtnTwo" Width="500" Height="40" Margin="0 0 0 10"
Content="{Binding BtnTwoText}" Visibility="{Binding Path=ButtonVisibility, Convert-
er={StaticResource BooleanToVisibilityConverter}}" FontSize="25" Command="{Binding
BtnTwoOnClickCommand}" Padding="0" Background="White" BorderBrush="Gray">
                        <Button.Style>
                            <Style TargetType="{x:Type Button}">
```

```
                                <Setter Property="Template">
                                    <Setter.Value>
                                        <ControlTemplate TargetType="{x:Type But-
ton}">
                                            <Border Background="{TemplateBinding
Background}" BorderBrush="Gray" BorderThickness="2">
                                                <ContentPresenter HorizontalAlign-
ment="Center" VerticalAlignment="Center"/>
                                            </Border>
                                        </ControlTemplate>
                                    </Setter.Value>
                                </Setter>
                            </Style>
                        </Button.Style>
                    </Button>
                    <Button x:Name="BtnThree" Width="500" Height="40" Margin="0 0 0
10" Content="{Binding BtnThreeText}" Visibility="{Binding Path=ButtonVisibility, Con-
verter={StaticResource BooleanToVisibilityConverter}}" FontSize="25" Com-
mand="{Binding BtnThreeOnClickCommand}" Padding="0" Background="White" Border-
Brush="Gray">
                        <Button.Style>
                            <Style TargetType="{x:Type Button}">
                                <Setter Property="Template">
                                    <Setter.Value>
                                        <ControlTemplate TargetType="{x:Type But-
ton}">
                                            <Border Background="{TemplateBinding
Background}" BorderBrush="Gray" BorderThickness="2">
                                                <ContentPresenter HorizontalAlign-
ment="Center" VerticalAlignment="Center"/>
                                            </Border>
                                        </ControlTemplate>
                                    </Setter.Value>
                                </Setter>
                            </Style>
                        </Button.Style>
                    </Button>
                    <Button x:Name="BtnFour" Width="500" Height="40" Margin="0 0 0
10" Content="{Binding BtnFourText}" Visibility="{Binding Path=ButtonVisibility, Con-
verter={StaticResource BooleanToVisibilityConverter}}" FontSize="25" Com-
mand="{Binding BtnFourOnClickCommand}" Padding="0" Background="White" Border-
Brush="Gray">
                        <Button.Style>
                            <Style TargetType="{x:Type Button}">
                                <Setter Property="Template">
                                    <Setter.Value>
                                        <ControlTemplate TargetType="{x:Type But-
ton}">
                                            <Border Background="{TemplateBinding
Background}" BorderBrush="Gray" BorderThickness="2">
                                                <ContentPresenter HorizontalAlign-
ment="Center" VerticalAlignment="Center"/>
                                            </Border>
                                        </ControlTemplate>
                                    </Setter.Value>
                                </Setter>
                            </Style>
                        </Button.Style>
                    </Button>
                </StackPanel>
            </StackPanel>
        </Grid>
```

```
        <Grid Width="540" Height="600" HorizontalAlignment="Left" VerticalAlign-
ment="Bottom" Margin="50 0 10 35">
            <Image Source="{Binding Image}"/>
        </Grid>
        <Grid Width="540" Height="600" HorizontalAlignment="Left" VerticalAlign-
ment="Bottom" Margin="50 0 10 35">
            <Button Width="20" Height="20" Background="#3849F9" Command="{Binding
MrkOneOnClickCommand}" Margin="{Binding MrkOneLoc}" Visibility="{Binding
Path=MarkerVisibility, Converter={StaticResource BooleanToVisibilityConverter}}">

            </Button>
            <Button Width="20" Height="20" Background="#3849F9" Command="{Binding
MrkTwoOnClickCommand}" Margin="{Binding MrkTwoLoc}" Visibility="{Binding
Path=MarkerVisibility, Converter={StaticResource BooleanToVisibilityConverter}}">

            </Button>
            <Button Width="20" Height="20" Background="#3849F9" Command="{Binding
MrkThreeOnClickCommand}" Margin="{Binding MrkThreeLoc}" Visibility="{Binding
Path=MarkerVisibility, Converter={StaticResource BooleanToVisibilityConverter}}">

            </Button>
        </Grid>
        <Button Width="30" Height="100" HorizontalAlignment="Left" Margin="10 0 0 0"
BorderBrush="Transparent" Background="Transparent" Foreground="#3849F9" Com-
mand="{Binding LeftArrOnClickCommand}" Visibility="{Binding Path=LeftArrowVisibility,
Converter={StaticResource BooleanToVisibilityConverter}}">
            <Button.Style>
                <Style TargetType="{x:Type Button}">
                    <Setter Property="Template">
                        <Setter.Value>
                            <ControlTemplate TargetType="{x:Type Button}">
                                <Border Background="{TemplateBinding Background}"
BorderBrush="Transparent" BorderThickness="2">
                                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                                </Border>
                            </ControlTemplate>
                        </Setter.Value>
                    </Setter>
                </Style>
            </Button.Style>
            <materialdesign:PackIcon Kind="ArrowLeft" Width="30" Height="30" Pad-
ding="0" VerticalAlignment="Center"/>
        </Button>
        <Button Width="30" Height="100" HorizontalAlignment="Right" Margin="0 0 10 0"
BorderBrush="Transparent" Background="Transparent" Foreground="#3849F9" Com-
mand="{Binding RightArrOnClickCommand}" Visibility="{Binding
Path=RightArrowVisibility, Converter={StaticResource BooleanToVisibilityConverter}}">
            <Button.Style>
                <Style TargetType="{x:Type Button}">
                    <Setter Property="Template">
                        <Setter.Value>
                            <ControlTemplate TargetType="{x:Type Button}">
                                <Border Background="{TemplateBinding Background}"
BorderBrush="Transparent" BorderThickness="2">
                                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                                </Border>
                            </ControlTemplate>
                        </Setter.Value>
                    </Setter>
                </Style>
            </Button.Style>
```

```
            <materialdesign:PackIcon Kind="ArrowRight" Width="30" Height="30" Pad-
ding="0" VerticalAlignment="Center"/>
        </Button>
    </Grid>
</Window>
```

Лістинг MainGameViewModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using GalaSoft.MvvmLight;
using Game_Application.Model;
using GalaSoft.MvvmLight.Command;
using Game_Application;
using System.Windows;
using GalaSoft.MvvmLight.Messaging;

namespace Game_Application.ViewModel
{
    public class MainGameViewModel : ViewModelBase
    {
        /// <summary>
        /// The <see cref="Text" /> property's name.
        /// </summary>
        public const string TextPropertyName = "Text";

        private string _text = string.Empty;

        /// <summary>
        /// Sets and gets the Text property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Text
        {
            get
            {
                return _text;
            }

            set
            {
                Set(ref _text, value);
            }
        }

        /// <summary>
        /// The <see cref="BtnOneText" /> property's name.
        /// </summary>
        public const string BtnOneTextPropertyName = "BtnOneText";

        private string _btnonetext = string.Empty;

        /// <summary>
        /// Sets and gets the BtnOneText property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string BtnOneText
        {
            get
```

```
    {
        return _btnonetext;
    }

    set
    {
        Set(ref _btnonetext, value);
    }
}

/// <summary>
/// The <see cref="BtnTwoText" /> property's name.
/// </summary>
public const string BtnTwoTextPropertyName = "BtnTwoText";

private string _btntwotext = string.Empty;

/// <summary>
/// Sets and gets the BtnTwoText property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string BtnTwoText
{
    get
    {
        return _btntwotext;
    }

    set
    {
        Set(ref _btntwotext, value);
    }
}

/// <summary>
/// The <see cref="BtnThreeText" /> property's name.
/// </summary>
public const string BtnThreeTextPropertyName = "BtnThreeText";

private string _btnthreetext = string.Empty;

/// <summary>
/// Sets and gets the BtnThreeText property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string BtnThreeText
{
    get
    {
        return _btnthreetext;
    }

    set
    {
        Set(ref _btnthreetext, value);
    }
}

/// <summary>
/// The <see cref="BtnFourText" /> property's name.
/// </summary>
public const string BtnFourTextPropertyName = "BtnFourText";
```

```
private string _btnfourtext = string.Empty;

/// <summary>
/// Sets and gets the BtnFourText property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string BtnFourText
{
    get
    {
        return _btnfourtext;
    }

    set
    {
        Set(ref _btnfourtext, value);
    }
}


/// <summary>
/// The <see cref="Count" /> property's name.
/// </summary>
public const string CountPropertyName = "Count";

private int _count = -1;

/// <summary>
/// Sets and gets the Count property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int Count
{
    get
    {
        return _count;
    }

    set
    {
        Set(ref _count, value);
    }
}

/// <summary>
/// The <see cref="ButtonVisibility" /> property's name.
/// </summary>
public const string ButtonVisibilityPropertyName = "ButtonVisibility";

private bool _buttonvisibility = false;

/// <summary>
/// Sets and gets the ButtonVisibility property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public bool ButtonVisibility
{
    get
    {
        return _buttonvisibility;
    }

    set
```

```
            {
                Set(ref _buttonvisibility, value);
            }
        }


        /// <summary>
        /// The <see cref="Image" /> property's name.
        /// </summary>
        public const string ImagePropertyName = "Image";

        private Uri _image;

        /// <summary>
        /// Sets and gets the Image property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public Uri Image
        {
            get
            {
                return _image;
            }

            set
            {
                Set(ref _image, value);
            }
        }


        /// <summary>
        /// The <see cref="LeftArrowVisibility" /> property's name.
        /// </summary>
        public const string LeftArrowVisibilityPropertyName = "LeftArrowVisibility";

        private bool _leftarrrowvisibility = false;

        /// <summary>
        /// Sets and gets the LeftArrowVisibility property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public bool LeftArrowVisibility
        {
            get
            {
                return _leftarrrowvisibility;
            }

            set
            {
                Set(ref _leftarrrowvisibility, value);
            }
        }


        /// <summary>
        /// The <see cref="RightArrowVisibility" /> property's name.
        /// </summary>
        public const string RightArrowVisibilityPropertyName = "RightArrowVisibil-
ity";

        private bool _rightarrrowvisibility = false;
```

```
/// <summary>
/// Sets and gets the RightArrowVisibility property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public bool RightArrowVisibility
{
    get
    {
        return _rightarrrowvisibility;
    }

    set
    {
        Set(ref _rightarrrowvisibility, value);
    }
}

/// <summary>
/// The <see cref="MarkerVisibility" /> property's name.
/// </summary>
public const string MarkerVisibilityPropertyName = "MarkerVisibility";

private bool _markervisibility = false;

/// <summary>
/// Sets and gets the MarkerVisibility property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public bool MarkerVisibility
{
    get
    {
        return _markervisibility;
    }

    set
    {
        Set(ref _markervisibility, value);
    }
}

/// <summary>
/// The <see cref="IsEnabledBtn" /> property's name.
/// </summary>
public const string IsEnabledBtnPropertyName = "IsEnabledBtn";

private bool _isenabledbtn = true;

/// <summary>
/// Sets and gets the IsEnabledBtn property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public bool IsEnabledBtn
{
    get
    {
        return _isenabledbtn;
    }

    set
    {
        Set(ref _isenabledbtn, value);
```

```
    }
}


/// <summary>
/// The <see cref="MrkOneLoc" /> property's name.
/// </summary>
public const string MrkOneLocPropertyName = "MrkOneLoc";

private Thickness _mrkoneloc;

/// <summary>
/// Sets and gets the MrkOneLoc property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public Thickness MrkOneLoc
{
    get
    {
        return _mrkoneloc;
    }

    set
    {
        Set(ref _mrkoneloc, value);
    }
}

/// <summary>
/// The <see cref="MrkTwoLoc" /> property's name.
/// </summary>
public const string MrkTwoLocPropertyName = "MrkTwoLoc";

private Thickness _mrktwoloc;

/// <summary>
/// Sets and gets the MrkTwoLoc property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public Thickness MrkTwoLoc
{
    get
    {
        return _mrktwoloc;
    }

    set
    {
        Set(ref _mrktwoloc, value);
    }
}

/// <summary>
/// The <see cref="MrkThreeLoc" /> property's name.
/// </summary>
public const string MrkThreeLocPropertyName = "MrkThreeLoc";

private Thickness _mrkthreeloc;

/// <summary>
/// Sets and gets the MrkThreeLoc property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
```

```
        public Thickness MrkThreeLoc
        {
            get
            {
                return _mrkthreeloc;
            }

            set
            {
                Set(ref _mrkthreeloc, value);
            }
        }

        private string game_id = string.Empty;
        private string type = string.Empty;
        private int btn_clicked = 0;
        private bool answerCheck = false;
        private int game_count = 0;
        private string btntext_check = string.Empty;
        private Uri ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
        private int mrk_clicked = 0;

        public RelayCommand<MainGameView> CloseWindowCommand { get; private set; }
        public RelayCommand<MainGameView> RestartWindowCommand { get; private set; }
        public RelayCommand<MainGameView> MainWindowCommand { get; private set; }
        public RelayCommand<MainGameView> BtnOneOnClickCommand { get; private set; }
        public RelayCommand<MainGameView> BtnTwoOnClickCommand { get; private set; }
        public RelayCommand<MainGameView> BtnThreeOnClickCommand { get; private set;
}
        public RelayCommand<MainGameView> BtnFourOnClickCommand { get; private set; }
        public RelayCommand<MainGameView> LeftArrOnClickCommand { get; private set; }
        public RelayCommand<MainGameView> RightArrOnClickCommand { get; private set;
}
        public RelayCommand<MainGameView> MrkOneOnClickCommand { get; private set; }
        public RelayCommand<MainGameView> MrkTwoOnClickCommand { get; private set; }
        public RelayCommand<MainGameView> MrkThreeOnClickCommand { get; private set;
}

        public MainGameViewModel()
        {
            this.CloseWindowCommand             =             new             Relay-
Command<MainGameView>(this.CloseWindow);
            this.RestartWindowCommand             =             new             Relay-
Command<MainGameView>(this.RestartWindow);
            this.MainWindowCommand = new RelayCommand<MainGameView>(this.NewWindow);
            this.BtnOneOnClickCommand             =             new             Relay-
Command<MainGameView>(this.BtnOneOnClick);
            this.BtnTwoOnClickCommand             =             new             Relay-
Command<MainGameView>(this.BtnTwoOnClick);
            this.BtnThreeOnClickCommand             =             new             Relay-
Command<MainGameView>(this.BtnThreeOnClick);
            this.BtnFourOnClickCommand             =             new             Relay-
Command<MainGameView>(this.BtnFourOnClick);
            this.LeftArrOnClickCommand             =             new             Relay-
Command<MainGameView>(this.LeftArrOnClick);
            this.RightArrOnClickCommand             =             new             Relay-
Command<MainGameView>(this.RightArrOnClick);
            this.MrkOneOnClickCommand             =             new             Relay-
Command<MainGameView>(this.MrkOneOnClick);
            this.MrkTwoOnClickCommand             =             new             Relay-
Command<MainGameView>(this.MrkTwoOnClick);
            this.MrkThreeOnClickCommand             =             new             Relay-
Command<MainGameView>(this.MrkThreeOnClick);
```

```
            if (Count == -1)
            {
                ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
                Image = ImageUri;
                ButtonVisibility = false;
                MarkerVisibility = false;
                IsEnabledBtn = true;
                BtnOneText = "Розпочати";
                Text = "Для початку роботи натисніть на кнопку 'Розпочати'. \r\n
Зверніть увагу! \r\n При виході у головне меню не завершивши проходження результати
збережені НЕ будуть! \r\n Приємного вивчення!";
                Count++;
            }
        }

        private void CloseWindow(MainGameView main)
        {
            if (main != null)
            {
                main.Close();
            }
        }

        private void RestartWindow(MainGameView main)
        {
            if (main != null)
            {
                main.Close();
                Sys-
tem.Diagnostics.Process.Start(Environment.GetCommandLineArgs()[0]);
            }
        }

        private void NewWindow(MainGameView main)
        {
            if (main != null)
            {
                MainWindow window = new MainWindow();
                window.Show();
                window.Container.Content = new ThemeView();
                window.menu_list.Visibility = Visibility.Visible;
                window.logout_btn.Visibility = Visibility.Visible;
                main.Close();
            }
        }

        private void BtnClicked()
        {
            MarkerVisibility = false;
            ButtonVisibility = true;
            IsEnabledBtn = true;
            Text = string.Empty;
            BtnOneText = string.Empty;
            BtnTwoText = string.Empty;
            BtnThreeText = string.Empty;
            BtnFourText = string.Empty;
            if (type.Equals("lesson"))
            {
                RightArrowVisibility = true;
                LeftArrowVisibility = true;
            }
            string imgName = MainGameModel.loadImage(Count, game_id);
```

```
            ImageUri = new Uri("pack://application:,,,/Images/" + game_id + "/" +
imgName + "");
            Image = ImageUri;
            Text = MainGameModel.getPhrase(Count, game_id);

            if (Text == string.Empty)
            {
                Text = MainGameModel.getQuestion(Count, game_id);
            }

            BtnOneText = MainGameModel.getAnswerOne(Count, game_id);
            BtnTwoText = MainGameModel.getAnswerTwo(Count, game_id);
            BtnThreeText = MainGameModel.getAnswerThree(Count, game_id);
            BtnFourText = MainGameModel.getAnswerFour(Count, game_id);

            if (BtnOneText.Equals("Продовжити."))
            {
                MarkerVisibility = true;
                IsEnabledBtn = false;
                MrkOneLoc = new Thickness(30, 20, 50, 40);
                MrkTwoLoc = new Thickness(0, -10, 40, -40);
                MrkThreeLoc = new Thickness(-20.0, 50.0, -40.0, 60.0);
            }
        }

        private void BtnOneOnClick(MainGameView main)
        {
            if (Count == 0)
            {
                game_id = MainViewModel.Game_Id;
                type = MainViewModel.Type;
                initiateGame();
            }
            else if (BtnOneText == "Закінчити проходження!")
            {
                if (type != "lesson")
                {
                    MainGameModel.writeMarks(game_count, MainViewModel.Login, type);
                }
                Count = -1;
                if (Count == -1)
                {
                    Text = string.Empty;
                    BtnOneText = string.Empty;
                    BtnTwoText = string.Empty;
                    BtnThreeText = string.Empty;
                    BtnFourText = string.Empty;
                    ButtonVisibility = false;
                    LeftArrowVisibility = false;
                    RightArrowVisibility = false;
                    MarkerVisibility = false;
                    Uri                 ImageUri                =                 new
Uri("pack://application:,,,/Images/Prof_lf.png");
                    Image = ImageUri;
                    BtnOneText = "Розпочати";
                    Text = "Для початку роботи натисніть на кнопку 'Розпочати'. \r\n
Зверніть увагу! \r\n При виході у головне меню не завершивши проходження результати
збережені НЕ будуть! \r\n Приємного вивчення!";
                    Count++;
                }
                game_count = 0;
            }
            else
```

```
            {
                BtnClicked();
                if (answerCheck.Equals(false))
                {
                    btn_clicked = 1;
                    Count--;
                    answerCheck   =   MainGameModel.checkAnswer(Count,   btn_clicked,
game_id);
                    Count++;
                    if (answerCheck.Equals(true))
                    {
                        game_count++;
                        answerCheck = false;
                    }
                }
                btn_clicked = 0;

                Count++;

                if (Text.Equals(string.Empty) && BtnOneText.Equals(string.Empty) &&
BtnTwoText.Equals(string.Empty)  &&  BtnThreeText.Equals(string.Empty)  &&  BtnFour-
Text.Equals(string.Empty))
                {
                    ButtonVisibility = false;
                    LeftArrowVisibility = false;
                    RightArrowVisibility = false;
                    ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
                    Image = ImageUri;
                    Text = "Це кінець! \r\n Для запису резульатів натисніть на кнопку
'Закінчити проходження!'. (У режимі гри оцінки не виставляються) \r\n І запамятай:
Наука це шлях до світлого майбутього!";
                    BtnOneText = "Закінчити проходження!";
                    Count = -1;
                }
                else          if          (BtnTwoText.Equals(string.Empty)          &&
BtnThreeText.Equals(string.Empty) && BtnFourText.Equals(string.Empty))
                {
                    ButtonVisibility = false;
                }
            }
        }

        private void BtnTwoOnClick(MainGameView main)
        {
            BtnClicked();
            if (answerCheck.Equals(false))
            {
                btn_clicked = 2;
                Count--;
                answerCheck = MainGameModel.checkAnswer(Count, btn_clicked, game_id);
                Count++;
                if (answerCheck.Equals(true))
                {
                    game_count++;
                    answerCheck = false;
                }
            }
            btn_clicked = 0;

            Count++;

            if (BtnOneText == string.Empty && BtnTwoText == string.Empty && Btn-
ThreeText == string.Empty && BtnFourText == string.Empty)
```

```
            {
                ButtonVisibility = false;
                RightArrowVisibility = false;
                LeftArrowVisibility = false;
                ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
                Image = ImageUri;
                Text = "Це кінець! \r\n Для запису результатів натисніть на кнопку 'Закінчити проходження!'. (У режимі гри оцінки не виставляються) \r\n І запамятай: Наука це шлях до світлого майбутьного!";
                BtnOneText = "Закінчити проходження!";
                Count = -1;
            }
            else if (BtnTwoText == string.Empty && BtnThreeText == string.Empty && BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
            }
        }

        private void BtnThreeOnClick(MainGameView main)
        {
            BtnClicked();

            if (answerCheck.Equals(false))
            {
                btn_clicked = 3;
                Count--;
                answerCheck = MainGameModel.checkAnswer(Count, btn_clicked, game_id);
                Count++;
                if (answerCheck.Equals(true))
                {
                    game_count++;
                    answerCheck = false;
                }
            }
            btn_clicked = 0;

            Count++;

            if (BtnOneText == string.Empty && BtnTwoText == string.Empty && Btn-ThreeText == string.Empty && BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
                RightArrowVisibility = false;
                LeftArrowVisibility = false;
                ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
                Image = ImageUri;
                Text = "Це кінець! \r\n Для запису результатів натисніть на кнопку 'Закінчити проходження!'. (У режимі гри оцінки не виставляються) \r\n І запамятай: Наука це шлях до світлого майбутьного!";
                BtnOneText = "Закінчити проходження!";
                Count = -1;
            }
            else if (BtnTwoText == string.Empty && BtnThreeText == string.Empty && BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
            }
        }

        private void BtnFourOnClick(MainGameView main)
        {
            BtnClicked();
```

```
            if (answerCheck.Equals(false))
            {
                btn_clicked = 4;
                Count--;
                answerCheck = MainGameModel.checkAnswer(Count, btn_clicked, game_id);
                Count++;
                if (answerCheck.Equals(true))
                {
                    game_count++;
                    answerCheck = false;
                }
            }
            btn_clicked = 0;

            Count++;

            if (BtnOneText == string.Empty && BtnTwoText == string.Empty && Btn-
ThreeText == string.Empty && BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
                RightArrowVisibility = false;
                LeftArrowVisibility = false;
                ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
                Image = ImageUri;
                Text = "Це кінець! \r\n Для запису резульатів натисніть на кнопку
'Закінчити проходження!'. (У режимі гри оцінки не виставляються) \r\n І запамятай:
Наука це шлях до світлого майбутьного!";
                BtnOneText = "Закінчити проходження!";
                Count = -1;
            }
            else if (BtnTwoText == string.Empty && BtnThreeText == string.Empty &&
BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
            }
        }

        private void LeftArrOnClick(MainGameView main)
        {
            Count--;
            btntext_check = MainGameModel.getAnswerOne(Count, game_id);
            if (btntext_check.Equals(string.Empty))
            {
                Count++;
                LeftArrowVisibility = false;
            }
            else
            {
                BtnClicked();

                if        (BtnTwoText.Equals(string.Empty)        &&        BtnThreeT-
ext.Equals(string.Empty) && BtnFourText.Equals(string.Empty))
                {
                    ButtonVisibility = false;
                }
            }
        }

        private void RightArrOnClick(MainGameView main)
        {
            btntext_check = MainGameModel.getAnswerOne(Count, game_id);
            if (btntext_check.Equals(string.Empty))
```

```
            {
                RightArrowVisibility = false;
            }
            else
            {
                BtnClicked();

                Count++;

                if        (BtnTwoText.Equals(string.Empty)        &&        BtnThreeT-
ext.Equals(string.Empty) && BtnFourText.Equals(string.Empty))
                {
                    ButtonVisibility = false;
                }
            }
        }

        private void MrkOneOnClick(MainGameView main)
        {
            mrk_clicked = 1;
            answerCheck = MainGameModel.checkAnswer(Count - 1, mrk_clicked, game_id);
            if (answerCheck == true)
            {
                Text = "Правильний вибір, можемо рухатись далі.";
                MarkerVisibility = false;
                IsEnabledBtn = true;
            }
            else
            {
                Text = "Спробуй ще раз, у тебе все вийде.";
            }
        }
        private void MrkTwoOnClick(MainGameView main)
        {
            mrk_clicked = 2;
            answerCheck = MainGameModel.checkAnswer(Count - 1, mrk_clicked, game_id);
            if (answerCheck == true)
            {
                Text = "Правильний вибір, можемо рухатись далі.";
                MarkerVisibility = false;
                IsEnabledBtn = true;
            }
            else
            {
                Text = "Спробуй ще раз, у тебе все вийде.";
            }
        }
        private void MrkThreeOnClick(MainGameView main)
        {
            mrk_clicked = 3;
            answerCheck = MainGameModel.checkAnswer(Count - 1, mrk_clicked, game_id);
            if (answerCheck == true)
            {
                Text = "Правильний вибір, можемо рухатись далі.";
                MarkerVisibility = false;
                IsEnabledBtn = true;
            }
            else
            {
                Text = "Спробуй ще раз, у тебе все вийде.";
            }
        }
```

```
        private void initiateGame()
        {
            ButtonVisibility = true;
            IsEnabledBtn = true;
            if (type.Equals("lesson"))
            {
                RightArrowVisibility = true;
            }
            Text = string.Empty;
            BtnOneText = string.Empty;
            BtnTwoText = string.Empty;
            BtnThreeText = string.Empty;
            BtnFourText = string.Empty;
            Uri ImageUri = new Uri("pack://application:,,,/Images/Prof_lf.png");
            Image = ImageUri;
            Text = MainGameModel.getPhrase(Count, game_id);

            if (Text == string.Empty)
            {
                Text = MainGameModel.getQuestion(Count, game_id);
            }

            BtnOneText = MainGameModel.getAnswerOne(Count, game_id);
            BtnTwoText = MainGameModel.getAnswerTwo(Count, game_id);
            BtnThreeText = MainGameModel.getAnswerThree(Count, game_id);
            BtnFourText = MainGameModel.getAnswerFour(Count, game_id);

            if (answerCheck.Equals(false))
            {
                btn_clicked = 1;
                Count--;
                answerCheck = MainGameModel.checkAnswer(Count, btn_clicked, game_id);
                Count++;
                if (answerCheck.Equals(true))
                {
                    game_count++;
                    answerCheck = false;
                }
            }
            btn_clicked = 0;

            Count++;

            if (BtnOneText == string.Empty && BtnTwoText == string.Empty && Btn-
ThreeText == string.Empty && BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
                RightArrowVisibility = false;
                LeftArrowVisibility = false;
                Text = "Це кінець! \r\n Для запису результатів натисніть на кнопку
'Закінчити проходження!'. (У режимі гри оцінки не виставляються) \r\n І запамятай:
Наука це шлях до світлого майбутьного!";
                BtnOneText = "Закінчити проходження!";
                Count = -1;
            }
            else if (BtnTwoText == string.Empty && BtnThreeText == string.Empty &&
BtnFourText == string.Empty)
            {
                ButtonVisibility = false;
            }
        }

    }
```

```
}
```

Лістинг MainGameModel

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SQLite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Game_Application.Model
{
    class MainGameModel
    {
        /// <summary>
        /// get a phrase for game
        /// </summary>
        /// <param name="count"></param>
        /// <param name="game_id"></param>
        /// <returns></returns>
        public static string getPhrase(int count, string game_id)
        {
            string phrase = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.Parameters.AddWithValue("@count", count);
                    fmd.CommandText = @"SELECT * FROM phraseData WHERE game_id =
@game_id AND id = @count";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                            phrase = r["phrase"].ToString();
                    }
                }
            }
            return phrase;
        }

        /// <summary>
        /// get question for game
        /// </summary>
        /// <param name="count"></param>
        /// <param name="game_id"></param>
        /// <returns></returns>
        public static string getQuestion(int count, string game_id)
        {
            string question = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
```

```
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.CommandText = @"SELECT * FROM questionData WHERE game_id =
@game_id";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (Convert.ToInt32(r["id"]) == count)
                        {
                            question = r["question"].ToString();
                        }
                    }
                }
            }
            return question;
        }

        /// <summary>
        /// get a first answers for game
        /// </summary>
        /// <param name="count"></param>
        /// <param name="game_id"></param>
        /// <returns></returns>
        public static string getAnswerOne(int count, string game_id)
        {
            string answerOne = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.CommandText = @"SELECT * FROM answerData WHERE game_id =
@game_id";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (Convert.ToInt32(r["id"]) == count)
                        {
                            answerOne = r["fr_answer"].ToString();
                        }
                    }
                }
            }
            return answerOne;
        }

        public static string getAnswerTwo(int count, string game_id)
        {
            string answerTwo = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.CommandText = @"SELECT * FROM answerData WHERE game_id =
@game_id";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
```

```
                    while (r.Read())
                    {
                        if (Convert.ToInt32(r["id"]) == count)
                        {
                            answerTwo = r["sc_answer"].ToString();
                        }
                    }
                }
            }
            return answerTwo;
        }

        public static string getAnswerThree(int count, string game_id)
        {
            string answerThree = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.CommandText = @"SELECT * FROM answerData WHERE game_id =
@game_id";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (Convert.ToInt32(r["id"]) == count)
                        {
                            answerThree = r["th_answer"].ToString();
                        }
                    }
                }
            }
            return answerThree;
        }

        public static string getAnswerFour(int count, string game_id)
        {
            string answerFour = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.CommandText = @"SELECT * FROM answerData WHERE game_id =
@game_id";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (Convert.ToInt32(r["id"]) == count)
                        {
                            answerFour = r["fo_answer"].ToString();
                        }
                    }
                }
            }
            return answerFour;
        }
```

```
        /// <summary>
        /// check answers
        /// </summary>
        /// <param name="count"></param>
        /// <param name="btn_clicked"></param>
        /// <param name="game_id"></param>
        /// <returns></returns>
        public static bool checkAnswer(int count, int btn_clicked, string game_id)
        {
            bool answerCheck = false;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.Parameters.AddWithValue("@count", count);
                    fmd.CommandText = @"SELECT * FROM answerData WHERE game_id =
@game_id AND id = @count";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        if (Convert.ToInt32(r["correct"]) == btn_clicked)
                        {
                            answerCheck = true;
                        }
                    }
                }
            }
            return answerCheck;
        }

        /// <summary>
        /// write mark in data base
        /// </summary>
        /// <param name="game_count"></param>
        /// <param name="login"></param>
        public static void writeMarks(int game_count, string login)
        {
            string marksSet = string.Empty;
            string newMarks = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand get = connect.CreateCommand())
                {
                    get.Parameters.AddWithValue("@login", login);
                    get.CommandText = @"SELECT * FROM marksInfo WHERE login =
@login";
                    get.CommandType = CommandType.Text;
                    SQLiteDataReader r = get.ExecuteReader();
                    while (r.Read())
                    {
                        marksSet = r["marks"].ToString();
                    }
                }
                newMarks = marksSet + ", " + game_count.ToString();

                using (SQLiteCommand set = connect.CreateCommand())
```

```
                {
                    set.Parameters.AddWithValue("@login", login);
                    set.Parameters.AddWithValue("@marks", newMarks);
                    set.CommandText = @"UPDATE marksInfo SET marks = @marks WHERE
login = @login";
                    set.CommandType = CommandType.Text;
                    set.ExecuteNonQuery();
                }
            }
        }

        /// <summary>
        /// gets image
        /// </summary>
        /// <param name="count"></param>
        /// <param name="game_id"></param>
        /// <returns></returns>
        public static string loadImage(int count, string game_id)
        {
            string imageUri = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.Parameters.AddWithValue("@count", count);
                    fmd.CommandText = @"SELECT * FROM answerData WHERE game_id =
@game_id AND id = @count";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        imageUri = r["imageUri"].ToString();
                    }
                }
            }
            return imageUri;
        }
        public static string getMrkOneLocation(int count, string game_id)
        {
            string MrkOneLoc = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.Parameters.AddWithValue("@count", count);
                    fmd.CommandText = @"SELECT * FROM mapData WHERE game_id =
@game_id AND id = @count";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        MrkOneLoc = r["mrkOneLocation"].ToString();
                    }
                }
            }
            return MrkOneLoc;
        }
```

```
        public static string getMrkTwoLocation(int count, string game_id)
        {
            string MrkTwoLoc = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.Parameters.AddWithValue("@count", count);
                    fmd.CommandText = @"SELECT * FROM mapData WHERE game_id =
@game_id AND id = @count";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        MrkTwoLoc = r["mrkTwoLocation"].ToString();
                    }
                }
            }
            return MrkTwoLoc;
        }

        public static string getMrkThreeLocation(int count, string game_id)
        {
            string MrkThreeLoc = string.Empty;
            using (SQLiteConnection connect = new SQLiteConnec-
tion(ConfigurationManager.ConnectionStrings["SqlData"].ConnectionString))
            {
                connect.Open();
                using (SQLiteCommand fmd = connect.CreateCommand())
                {
                    fmd.Parameters.AddWithValue("@game_id", game_id);
                    fmd.Parameters.AddWithValue("@count", count);
                    fmd.CommandText = @"SELECT * FROM mapData WHERE game_id =
@game_id AND id = @count";
                    fmd.CommandType = CommandType.Text;
                    SQLiteDataReader r = fmd.ExecuteReader();
                    while (r.Read())
                    {
                        MrkThreeLoc = r["mrkThreeLocation"].ToString();
                    }
                }
            }
            return MrkThreeLoc;
        }
    }
}
```

ДОДАТОК Є

Розробка квест-гри для вивчення історії України учнями 5-го класу

Автор: Хвайра Т.С.Т., *студент*
Сумський державний університет, Суми, України

На даний момент інформаційні технології відіграють визначну роль у навчанні молоді. Використання комп'ютеру та мультимедійних засобів покращує засвоєння навчального матеріалу та підвищує зацікавленість учнів у пізнанні нового матеріалу. Квест-гра для вивчення історії України учнями 5-го класу допоможе вчителям автоматизувати перевірку та проведення практичних та контрольних робіт, а для учнів стане надійним та цікавим провідником у світ науки.

Основний функціонал програмного додатку полягає у інтерактивному проходженні уроків історії України, виконання контрольних та практичних робіт. Також додатковою функцією буде можливість перегляду поточних оцінок. Буде реалізовано два рівні доступу до оцінок: «учень» - може переглядати лише свої оцінки, та «вчитель» - може переглядати усі доступні у додатку оцінки. Інтерактивне проходження уроків буде реалізоване методом, який полягає у проходженні завдань для подальшого просування по уроку. Наприклад учень обрав тему уроку та розпочав проходження, після отримання певної інформації перед учнем з'явиться завдання яке він повинен виконати для продовження гри. Такий метод підвищить кількість інформації, яку запам'ятає учень в результаті виконання завдань уроку.

Для реалізації даного проекту буде використано мову програмування C# на платформі .Net з використанням системи побудови інтерфейсу WPF(Windows Presentation Foundation). У якості баз даних буде використана SQLite. У якості середи розробки буде обрано Visual Studio, вона легка у використанні та найкраще підходить для програмування на мові C#.

Основою цільовою аудиторією програмного продукту будуть вчителі та учні 5-х класів загальноосвітніх закладів, тому що продукт на даний момент націлений на вивчення історії України у 5-му класі.

Керівник: Алексенко О.В., *доцент*

ДОДАТОК Ж

АКТ ВПРОВАДЖЕННЯ

Виконана студентом групи ІТ-53-7 факультету ЕлІТ, Сумського державного університету Хвайра Т.С.Т. дипломна робота на тему «Ігровий додаток «Квест-гра для вивчення історії України учнями 5-го класу»» відповідає замовленню, та має певну практичну значимість і планується до впровадження.

Директор КУ ССШ І-ІІІ № 10 _____ Купреєва Н.М.