Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

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

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: Розроблення гри "Динозавр" мовою C#

Виконав: студент 2 курсу групи ІСТС-21
спеціальності
126 "Інформаційні системи та технології"
(шифр і назва напряму підготовки, спеціальності)

Скульбеда О.С.
(прізвище та ініціали)

Керівник Головата С.Б., Борецька І.Б.
(прізвище та ініціали)

Рецензент Здобицький А.Я.
(прізвище та ініціали)

Львів – 2024

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра інформаційних систем та комп'ютерного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 "Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСКМ
_____ Сторожук О.Л.
"06" 02 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Скульбеда Олександр Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення гри "Динозавр" мовою C#
керівник роботи *Головата Софія Богданівна, Борецька Ірина Богданівна, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "06" лютого 2024 року №С-87
2. Термін подання студентом роботи 10 червня 2024 р.
3. Вихідні дані до роботи Формулювання задачі та її формалізація. Аналіз попередніх досліджень. Огляд програмних засобів для реалізації поставленого завдання.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз предметної області
2) Інформаційне забезпечення
3) Математичне забезпечення
4) Програмне забезпечення
5) Технічне забезпечення
6) Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайди доповіді, актуальність теми, постановка завдання, аналіз отриманих результатів, висновки
6. Дата видачі завдання 07 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примі
1	Огляд літературних та інших джерел згідно досліджуваної теми. Збір потрібних матеріалів.	07.02.2024– 13.02.2024 р.	Виконано
2	Постановка задачі та її формалізація.	13.02.2024- 18.02.2024 р.	Виконано
3	Вибір та обгрунтування методів і засобів розв'язання завдання.	18.02.2024- 01.03.2024 р.	Виконано
4	Програмна реалізація системи.	01.03.2024- 31.03.2024 р.	Виконано
5	Оформлення опису створеної програми.	31.03.2024- 15.04.2024 р.	Виконано
6	Аналіз отриманих результатів виконання програми.	16.04.2024 р.	Виконано
7	Здача пояснювальної записки на перевірку та виправлення виявлених помилок.	01.06.2024- 06.06.2024 р.	Виконано

Студент _____ Скульбеда О.С.
(підпис) (прізвище та ініціали)

Керівники роботи _____ Головата С. Б., Борецька І.Б
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 37 сторінки пояснювальної записки, 9 малюнків та 15 джерел. Розроблений програмний продукт є застосунком, який дозволяє користувачам грати у гру, додавати друзів та спілкуватися з ними.

Програмний продукт було реалізовано в середовищі програмування Visual Studio за допомогою мови програмування C#. База даних була створена у об’єктно-реляційній системі керування базами даних MySQL.

Гра ґрунтується на простому, але захоплюючому ігровому процесі, де головний герой, динозавр, повинен подолати різноманітні перешкоди, збирати бонуси та досягати нових рівнів. Така гра є прикладом аркадного жанру, який має широку аудиторію серед гравців різного віку.

Гра має інтуїтивне керування та привабливу візуальну графіку, що покращує загальний ігровий досвід. Користувачі можуть змагатися зі своїми друзями через таблиці лідерів, що сприяє відчуттю конкуренції та підтримує інтерес гравців. Загалом, мета гри полягає у наданні розваг та задоволення гравцям, сприяючи соціальній взаємодії та залученню до ігрової спільноти.

Ключові слова: аркадна гра, C#, MySQL, GUI, соціальна взаємодія, 2D-гра, система керування базами даних.

ABSTRACT

The diploma thesis comprises 37 pages of explanatory notes, 9 illustrations, and 15 sources. The developed software product is an application that allows users to play a game, add friends, and communicate with them.

The software product was implemented in the Visual Studio programming environment using the C# programming language. The database was created in the object-relational database management system MySQL.

The game is based on a simple yet captivating gameplay where the main character, a dinosaur, must overcome various obstacles, collect bonuses, and reach new levels. Such a game serves as an example of an arcade genre with a wide audience among players of different ages.

The game features intuitive controls and attractive visual graphics, enhancing the overall gaming experience. Users can compete with their friends through leaderboards, fostering a sense of competition and supporting player engagement. Overall, the goal of the game is to provide entertainment and satisfaction to players, promoting social interaction and engagement within the gaming community.

Keywords: arcade game, C#, MySQL, GUI, social interaction, 2D game, database management system.

ТЕХНІЧНЕ ЗАВДАННЯ

Загальні відомості

Назва проекту: Гра "Динозавр"

Мета проекту

Створення розважальної аркадної гри для ПК, яка буде цікавою та корисною для дітей і дорослих.

Вимоги до функціоналу

1. **Інтерфейс користувача**:
    - Інтуїтивно зрозумілий інтерфейс.
    - Головне меню з кнопками "Почати гру", "Рекорди", "Друзі", "Вихід".
2. **Ігровий процес**:
    - Керування персонажем (динозавром) з використанням клавіш.
    - Різні типи перешкод, які необхідно уникати.
    - Поступове збільшення складності гри.
3. **Графіка та звук**:
    - Якісна двовимірна графіка.
    - Звукові ефекти для дій персонажа та взаємодії з об'єктами.

Вимоги до продуктивності

- Швидке завантаження рівнів.
- Стабільна робота гри без зависань.

Етапи розробки

1. **Аналіз та проектування**:
    - Дослідження ринку та визначення основних вимог до гри.
    - Розробка концепції гри.
2. **Розробка**:
    - Створення прототипу гри.
    - Розробка ігрового коду та графіки.
3. **Тестування**:
    - Тестування гри на різних пристроях.
    - Виправлення помилок.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

- **MST** — мінімальне кістякове дерево (minimum spanning tree).
- **СУБД** — система управління базами даних.
- **ORM** — об'єктно-реляційне відображення (Object-Relational Mapping).
- **SQL** — мова структурованих запитів (Structured Query Language).
- **C#** — мова програмування C Sharp.
- **CPU** — центральний процесор (Central Processing Unit).
- **JIT** — Just-In-Time компіляція (компіляція під час виконання).
- **TPL** — бібліотека паралельного програмування (Task Parallel Library).
- **API** — інтерфейс програмування додатків (Application Programming Interface).
- **GUI** — графічний інтерфейс користувача (Graphical User Interface).
- **IDE** — інтегроване середовище розробки (Integrated Development Environment).

ВСТУП

Розвиток інформаційних технологій і зростання популярності комп'ютерних та мобільних ігор відкривають нові можливості для створення розважального та навчального контенту. Сучасні ігри не тільки забезпечують відпочинок, але й сприяють розвитку різних навичок: логічного мислення, швидкості реакції, координації та креативності. Вони стають важливою частиною культурного та освітнього простору.

Гра "Динозавр" має на меті створення продукту, який поєднує в собі розвагу та розвиток навичок користувача. Основна цільова аудиторія - це діти та підлітки, але гра буде цікавою і для дорослих, завдяки своїй простоті та динаміці.

Об’єкт дослідження – розробка та реалізація комп’ютерної гри «Динозавр».

Предмет дослідження - розробка та оптимізація програмного забезпечення для 2D гри на базі C# з використанням технології Windows Forms.

Мета роботи - розробка гри "Динозавр" із використанням сучасних технологій та інструментів програмування, таких як C#, Visual Studio, Windows Forms та MySQL, а також обґрунтування вибору цих технологій з точки зору їх продуктивності, зручності та надійності.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд існуючих аркадних ігор

Аркадні ігри користуються великою популярністю серед гравців різного віку завдяки своїй простоті, захоплюючому геймплею та можливості грати в короткі проміжки часу. Для аналізу були обрані такі популярні аркадні ігри: Chrome Dino (T-Rex Run), Flappy Bird та Temple Run.

Chrome Dino (T-Rex Run)

Історія створення та популярність серед користувачів

Chrome Dino, також відома як T-Rex Run, була створена розробниками Google як пасхальне яйце, яке активується при відсутності інтернет-з'єднання у браузері Google Chrome. З моменту свого запуску гра здобула величезну популярність завдяки своїй доступності та простоті[1].

Особливості геймплею

Гравець керує динозавром, який постійно біжить вперед по пустелі. Основне завдання – уникати перешкод у вигляді кактусів та птеродактилів, стрибаючи через них або нахиляючись. Гра триває доти, доки динозавр не зіткнеться з перешкодою.

Графіка та дизайн

Chrome Dino має мінімалістичний дизайн, виконаний у чорно-білій палітрі. Такий стиль забезпечує високу продуктивність навіть на слабких пристроях і робить гру легко впізнаваною.

Аналіз сильних сторін

- Простота управління: лише одна кнопка для стрибків.
- Доступність: гра інтегрована у браузер Google Chrome і доступна без додаткових завантажень.
- Висока реіграбельність: нескінченний ігровий процес, що спонукає гравців намагатися побити власні рекорди.

Аналіз слабких сторін

- Обмежений контент: відсутність різноманітних рівнів та сюжетних ліній.
- Монотонність: одноманітний геймплей може швидко набриднути деяким гравцям.

Flappy Bird

Історія створення та причини популярності

Flappy Bird була розроблена в'єтнамським програмістом Донг Нгуєн і випущена у 2013 році. Гра швидко здобула популярність завдяки своїй високій складності та простоті, але була видалена з магазинів додатків самим розробником через особисті причини[2].

Особливості геймплею

Гравець керує маленькою пташкою, яка повинна пролітати між рядами зелених труб. Кожен тап по екрану змушує пташку підстрибувати вгору. Завдання полягає в тому, щоб пролетіти якнайдалі, не врізавшись у труби.

Графіка та дизайн

Flappy Bird має ретро-стиль з піксельною графікою, яка нагадує класичні ігри 8-бітної ери. Дизайн простий, але дуже впізнаваний.

Аналіз сильних сторін

- Висока складність: гра є викликом для гравців, що стимулює повторні спроби.
- Проста механіка: лише одна дія (тап по екрану) для керування пташкою.
- Вірусний ефект: популярність гри швидко зростала завдяки соціальним мережам та обговоренням серед гравців.

Аналіз слабких сторін

- Фрустрація від високої складності: гра може бути занадто складною для деяких гравців, що викликає негативні емоції.
- Одноманітність: відсутність різноманітності в геймплеї та рівнях.

Temple Run

Історія створення та вплив на жанр нескінченних бігунів

Temple Run була розроблена студією Imangi Studios і випущена у 2011 році. Гра швидко стала популярною і задала тенденцію для жанру нескінченних бігунів, що надихнуло багатьох розробників на створення подібних ігор[3].

Особливості геймплею

Гравець керує персонажем, який тікає від злих мавп через серію лабіринтів. Управління включає стрибки, підкати, повороти та нахили, щоб уникати перешкод і збирати монети.

Графіка та дизайн

Temple Run має тривимірну графіку з деталізованими середовищами, включаючи храми, джунглі та обриви. Гра виглядає візуально привабливо і має багато різноманітних елементів дизайну.

Аналіз сильних сторін

- Різноманітність рівнів: кожна гра генерується випадковим чином, що робить кожне проходження унікальним.
- Тривимірна графіка: високоякісна графіка забезпечує візуальне задоволення.
- Прогресія та розблокування: наявність системи прогресу, розблокування нових персонажів та покращень.

Аналіз слабких сторін

- Високе навантаження на ресурси: гра може не працювати плавно на старіших або слабких пристроях.
- Складність управління: багатофункціональне управління може бути складним для нових гравців.

Висновки з аналізу аналогів

Аналіз існуючих аркадних ігор дозволив виділити кілька ключових факторів, які повинні бути враховані при розробці гри "Динозавр":

- Простота управління та доступність.
- Інтуїтивно зрозумілий геймплей, що не вимагає тривалого навчання.
- Висока реіграбельність за рахунок нескінченного геймплею або випадково генерованих рівнів.
- Візуально привабливий дизайн, що привертає увагу гравців.
- Оптимізація для плавної роботи на різних пристроях, включаючи старіші моделі.

Ці фактори стануть основою для розробки гри "Динозавр", що забезпечить її успіх та популярність серед користувачів.

1.2. Висновок з огляду

Аналіз аналогів і ринку ігор дозволив виявити кілька ключових факторів успіху, які повинні бути враховані при розробці гри "Динозавр". Ці фактори сприяють створенню гри, яка буде цікавою, доступною та привабливою для широкої аудиторії гравців.

Ключові фактори успіху аркадних ігор

1. **Простота управління та доступність**

Аркадні ігри, такі як Chrome Dino і Flappy Bird, демонструють, що простота управління є однією з головних причин їх популярності. Ігри з одним або двома видами дій, що легко освоюються, привертають більшу кількість гравців, незалежно від їхнього досвіду в іграх. Це дозволяє навіть новачкам швидко зануритися в геймплей і отримати задоволення від гри[4].

2. **Інтуїтивно зрозумілий геймплей**

Ігри повинні мати чіткі правила та інтуїтивно зрозумілий ігровий процес. У випадку з Chrome Dino і Flappy Bird, гравці відразу розуміють, що потрібно робити, що зменшує час на навчання і збільшує час, проведений у грі. Це підвищує задоволення від гри та стимулює гравців повертатися до неї знову і знову.

3. **Висока реіграбельність**

Ігри, що забезпечують високу реіграбельність, такі як Temple Run, утримують інтерес гравців протягом тривалого часу. Нескінченний геймплей, випадково генеровані рівні або можливість досягати нових рекордів є важливими елементами, які сприяють цьому. Гравці завжди мають стимул повернутися до гри, щоб поліпшити свій результат або відкрити нові можливості.

4. **Візуально привабливий дизайн**

Хоча графіка не є головним аспектом аркадних ігор, візуальна привабливість все ж має велике значення. Яскравий, чіткий та стильний дизайн, як у Temple Run,

привертає увагу гравців і створює позитивні враження від гри. Використання мінімалістичного підходу, як у Chrome Dino, або ретро-стилю, як у Flappy Bird, також може бути ефективним[5].

5. **Оптимізація**

Оптимізація дозволяє охопити більшу аудиторію і забезпечити комфортний геймплей для всіх користувачів. Легка вага гри, швидке завантаження та стабільна робота на різних платформах сприяють її популярності.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Опис концепції гри "Динозавр"

Загальна концепція

Гра "Динозавр" є нескінченним аркадним бігуном, де гравець керує маленьким динозавром, що біжить через пустелю, уникаючи різноманітних перешкод. Гра розроблена для ПК, що забезпечує її доступність та зручність для користувачів персональних комп'ютерів. Основна мета гравця – пробігти якомога далі, уникаючи зіткнень з перешкодами для збільшення свого рахунку.

Ключові особливості

1) **Простота управління**
   - Гравець керує динозавром за допомогою клавіатури. Натискання на стрілку вверх викликає стрибок динозавра, а натискання на стрілку вниз змушує його присідати.
2) **Інтуїтивно зрозумілий геймплей**
   - Гра має чіткі та прості правила, що дозволяє гравцям швидко зануритися в ігровий процес без довгого навчання.
3) **Мінімалістична графіка**
   - Графічний стиль гри є простим та привабливим, з використанням чорно-білої палітри, що забезпечує високу продуктивність і чистий візуальний стиль.
4) **Захоплююча атмосфера**
   - Гра забезпечує захоплюючий досвід завдяки поєднанню динамічного геймплею та плавної анімації.

Основна мета гри

Метою гри є досягнення максимального можливого рахунку шляхом уникання перешкод. Гравець повинен реагувати швидко та точно, щоб уникнути зіткнень і продовжити біг динозавра якнайдовше.

Цільова аудиторія

Гра "Динозавр" розрахована на широку аудиторію, включаючи дітей, підлітків та дорослих. Завдяки своїй простоті та доступності, вона підходить для гравців різного віку та рівня досвіду.

Платформа

Гра розроблена для ПК, що дозволяє використати переваги великого екрану та клавіатури для керування. Це робить гру зручною для користувачів персональних комп'ютерів і забезпечує оптимальний геймплей.

2.2. Розробка ігрового процесу

Основні механіки гри

Гра "Динозавр" спирається на нескладні, але ефективні ігрові механіки, що роблять її доступною та захоплюючою для гравців усіх вікових категорій. Основні механіки включають:

1) **Автоматичний біг**
    - Динозавр автоматично рухається вперед із постійною швидкістю. Гравець не має контролю над рухом вперед, а лише над діями динозавра (стрибок, присідання).
2) **Стрибок**
    - Основна дія, що дозволяє динозавру уникати перешкод. Гравець здійснює стрибок, натискаючи клавішу (наприклад, пробіл). Довжина і висота стрибка можуть бути змінними в залежності від часу утримання натискання.
3) **Присідання**
    - Додаткова дія, що дозволяє динозавру уникати низьких перешкод. Гравець може виконати цю дію, натискаючи певну клавішу (наприклад, стрілку вниз).

Система перешкод

У грі "Динозавр" передбачено різноманітні перешкоди, що роблять геймплей більш різноманітним і цікавим:

1) **Перешкоди**
    - **Кактуси**: Розташовані на різних відстанях один від одного, вони змушують гравця реагувати швидко, щоб уникнути зіткнення.
    - **Птеродактилі**: Літаючі вороги, що з'являються на різних висотах, що вимагає від гравця стрибати або присідати у потрібний момент.
    - **Камені**: Низькі перешкоди, які потребують стрибка або присідання для уникнення.

Балансування складності

Балансування складності є ключовим аспектом для забезпечення захоплюючого ігрового процесу:

1) **Поступове збільшення складності**
    - Швидкість бігу динозавра поступово збільшується з часом, ускладнюючи уникнення перешкод. Це дозволяє новим гравцям адаптуватися до управління, а досвідченим – продовжувати викликати себе.
2) **Рандомізація перешкод**
    - Перешкоди генеруються випадковим чином, що робить кожну сесію унікальною і зберігає інтерес гравця.

Інтерактивний користувацький інтерфейс

Інтерфейс користувача має бути максимально зрозумілим і інтуїтивним:

1) **Головне меню**
    - Містить кнопки для початку гри, перегляду рекордів та іншої інформації. Інтерфейс повинен бути простим і зрозумілим.
2) **Ігровий екран**
    - Відображає поточний рахунок, найвищий рекорд, рівень і поточну швидкість.
3) **Екран результатів**

- Відображає набрані очки, найвищий рекорд та статистику поточної сесії. Цей екран також може містити опції для повторного початку гри або повернення в головне меню.

2.3 Вибір архітектури системи

Розроблювана система повинна мати архітектуру Model View Controller(MVC). **Архітектура Model-View-Controller (MVC)[6]** - це шаблон проектування програмного забезпечення, який розділяє програму на три основні компоненти: Model (Модель), View (Вид) і Controller (Контролер). Кожен з цих компонентів виконує свої відповідні функції, і взаємодіє з іншими компонентами для забезпечення ефективної роботи програми. Розглянемо кожен з них:

Модель (Model):

- Вона представляє собою усі дані, що стосуються гри: наприклад, стан гравця, поточний рівень гри, рахунок, швидкість, висота перешкод і т.д.
- Модель відповідає за обробку усіх правил гри, перевірку зіткнень, оновлення стану гравця та інших об'єктів у грі.
- Вона також може відправляти події (наприклад, "гравець зіткнувся з перешкодою") до контролера для подальшої обробки.

Вид (View):

- У випадку Dino Game, Вид - це графічний інтерфейс гри, який відображається на екрані браузера.
- Він відповідає за відображення гравця, перешкод, фону і інших графічних елементів.
- View також може слідкувати за станом гри і оновлювати себе відповідно, коли відбуваються зміни в моделі.

Контролер (Controller):

- У грі, Контролер відповідає за обробку вхідних подій, таких як натискання клавіш або дотик до екрану (у випадку мобільних пристроїв).

- Він інтерпретує ці події і відправляє відповідні команди до моделі для оновлення стану гри або до відображення для зміни інтерфейсу користувача.
- Контролер може включати обробку різних сценаріїв, таких як початок гри, кінець гри, тощо.

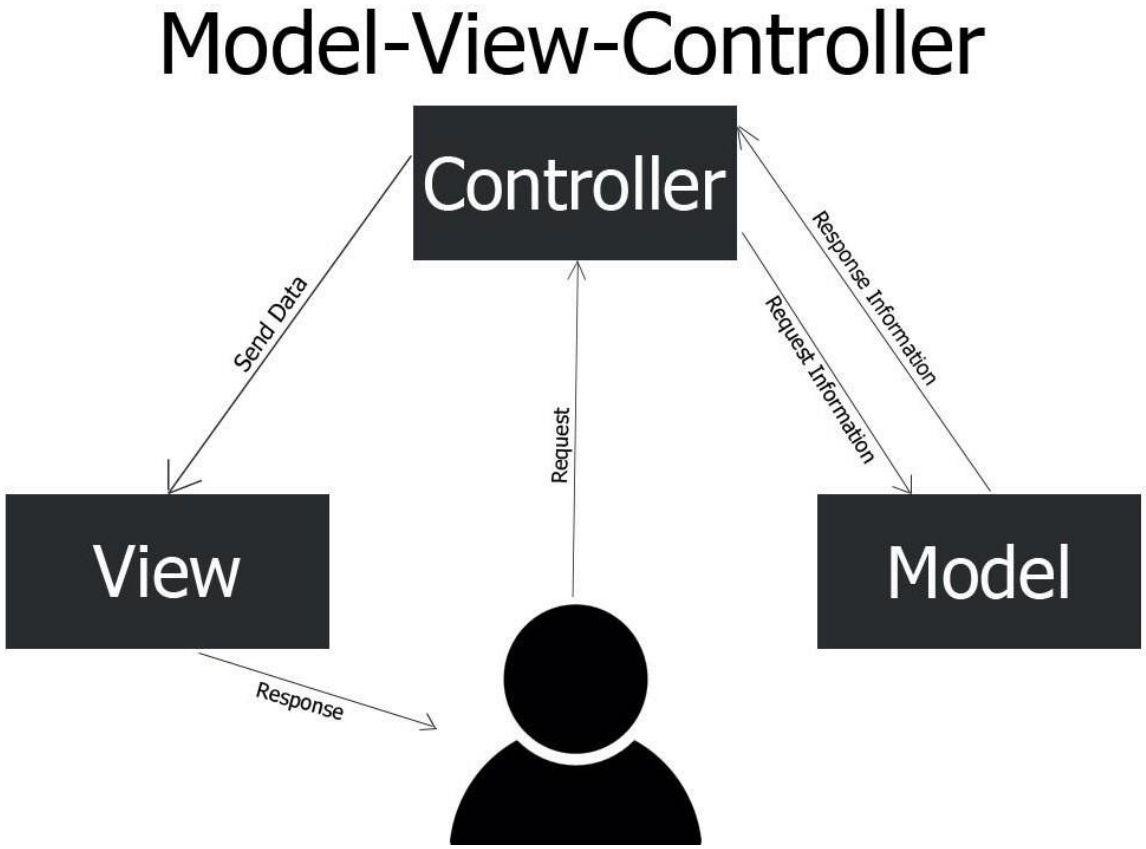

Рисунок 2.1 – схематичне зображення MVC архітектури.

Контролер відповідає за відстеження визначених подій, що виникають в результаті взаємодії користувача з програмою. Він допомагає структурувати код, групуючи пов'язані дії в окремий клас. Наприклад, в типовому MVC-проєкті може бути контролер, який містить методи, пов'язані з управлінням обліковим записом користувача, такі як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події перетворюються в різні запити, які направляються до компонентів моделі або об'єктів, відповідальних за відображення даних. Розділення моделі від вигляду даних дозволяє використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер змінює

дані моделі, то інформація, яку бачать візуальні компоненти, автоматично оновлюється відповідно до змін, що відбулися.

2.4. Вибір мови програмування та середовища розробки

Гра "Динозавр" розроблена у середовищі Visual Studio 2020 з використанням мови програмування C# та технології Windows Forms. Давайте розглянемо, чому було обрано саме ці технології.

2.4.1. Мова програмування C#

C# – це сучасна, об'єктно-орієнтована мова програмування, що була розроблена корпорацією Microsoft. Основні переваги C# включають[7]:

1) Синтаксис

Синтаксис C# є чистим і зрозумілим, що робить його легким для вивчення і використання. Це спрощує процес розробки і знижує кількість помилок.

- **Чіткість і простота**: Синтаксис C# запозичений з C-подібних мов, таких як C++ і Java, що робить його інтуїтивно зрозумілим для розробників з досвідом у цих мовах.
- **Типобезпечність**: C# є строго типізованою мовою, що допомагає виявляти помилки на етапі компіляції, зменшуючи ризик виникнення багів у рантаймі.
- **Інтеграція з Visual Studio**: Visual Studio забезпечує відмінну підтримку синтаксису C#, включаючи інтелектуальні підказки (IntelliSense), автоматичне форматування коду, дебагінг та рефакторинг.

2) Підтримка об'єктно-орієнтованого програмування

C# має потужні можливості для роботи з об'єктами, що дозволяє створювати модульні та легко підтримувані програми.

- **Інкапсуляція, наслідування та поліморфізм**: C# повністю підтримує основні принципи об'єктно-орієнтованого програмування (ООП), що дозволяє створювати складні та реюзабельні програмні компоненти.

- **Абстракція**: Завдяки інтерфейсам і абстрактним класам, C# забезпечує високий рівень абстракції, що спрощує розробку складних систем.
- **Управління пам'яттю**: C# використовує автоматичний збирач сміття (Garbage Collector), що значно полегшує управління пам'яттю та запобігає витокам пам'яті.

3) Висока продуктивність

C# компілюється в ефективний машинний код, що забезпечує високу швидкість виконання програм.

- **JIT-компіляція**: C# використовує Just-In-Time компіляцію, яка перетворює байт-код у машинний код безпосередньо перед виконанням, оптимізуючи продуктивність.
- **Агресивна оптимізація**: Компилятор C# виконує агресивні оптимізації коду, такі як інлайнінг методів і усунення мертвого коду.
- **Паралельне програмування**: Завдяки бібліотекам, таким як TPL (Task Parallel Library), C# дозволяє легко реалізовувати багатопотокові програми, що підвищує продуктивність при виконанні паралельних завдань.

4) Широкий спектр бібліотек і фреймворків

C# має багатий набір бібліотек, які спрощують розробку графічного інтерфейсу, обробку подій, роботу з базами даних та інші аспекти програмування.

- **.NET Framework**: Багатий набір класів і методів для роботи з мережею, файловою системою, шифруванням, введенням/виведенням даних та ін.
- **Windows Forms та WPF**: Потужні інструменти для створення графічних інтерфейсів користувача, які підтримують візуальне проектування і забезпечують інтерактивність додатків.
- **Entity Framework**: ORM (Object-Relational Mapping) фреймворк, який спрощує роботу з базами даних, дозволяючи працювати з даними на рівні об'єктів.

5) Кросплатформеність

Завдяки .NET Core, C# підтримує кросплатформену розробку, що дозволяє створювати додатки для різних операційних систем.

- **.NET Core**: Легка і модульна версія .NET, яка дозволяє створювати високопродуктивні додатки, що працюють на Windows, Linux і macOS.
- **Xamarin**: Платформа для розробки мобільних додатків на C#, яка дозволяє створювати нативні додатки для Android і iOS з використанням загального коду.
- **ASP.NET Core**: Потужний фреймворк для розробки веб-додатків і API, який працює на всіх основних платформах.

2.4.2. Середовище розробки Visual Studio

Visual Studio 2020 є одним з найпопулярніших середовищ розробки завдяки своїм багатим можливостям[8]:

1) Інтеграція з C#

Visual Studio надає повну підтримку для C#, включаючи потужні засоби для відладки, рефакторингу коду та управління проєктом.

- **Підтримка синтаксису та IntelliSense**: Завдяки IntelliSense, розробники можуть швидше писати код за допомогою автоматичних підказок та автозавершення. Це значно зменшує кількість помилок і прискорює процес розробки.
- **Інструменти відладки**: Visual Studio включає в себе потужні інструменти для відладки, які дозволяють відстежувати виконання коду в реальному часі, аналізувати стек викликів і контролювати значення змінних.
- **Рефакторинг**: Інструменти рефакторингу допомагають автоматизувати процеси зміни структури коду, зберігаючи його функціональність. Це включає перейменування змінних, екстракцію методів, перетворення циклів та багато іншого.

2) Зручний інтерфейс

Інтерфейс Visual Studio є інтуїтивно зрозумілим і зручним, що сприяє підвищенню продуктивності розробника.

- **Налаштовуваний інтерфейс**: Користувачі можуть налаштовувати інтерфейс під свої потреби, створюючи власні макети вікон, панелей інструментів і шорткатів.
- **Проектні шаблони**: Visual Studio надає широкий вибір шаблонів проектів, які дозволяють швидко розпочати розробку з попередньо налаштованими середовищами для різних типів додатків.

3) Інструменти для тестування

Вбудовані інструменти для тестування дозволяють легко писати і виконувати тести, що підвищує якість програмного забезпечення.

- **Юніт-тестування**: Visual Studio підтримує юніт-тестування, надаючи вбудовані засоби для написання та запуску тестів. Це допомагає розробникам переконатися, що кожен окремий модуль працює правильно.
- **Інтеграційне тестування**: Інструменти для інтеграційного тестування дозволяють перевіряти взаємодію між різними частинами програми.
- **Покриття тестами**: Visual Studio може аналізувати покриття тестами, вказуючи, які частини коду були протестовані, а які - ні.

4) Інтеграція з Git

Підтримка Git забезпечує зручне управління версіями коду та спільну роботу над проектом.

- **Інтеграція з репозиторіями**: Visual Studio дозволяє підключатися до локальних і віддалених репозиторіїв, здійснювати коміти, пуши, пулли та мерджі безпосередньо з середовища розробки.
- **Інструменти для вирішення конфліктів**: Вбудовані інструменти для вирішення конфліктів допомагають розробникам легко справлятися з конфліктами під час злиття гілок.

2.4.3. Використання Windows Forms

Windows Forms – це технологія для створення графічного інтерфейсу користувача (GUI) у додатках Windows. Основні переваги Windows Forms включають[9]:

1) Простота створення інтерфейсу

Windows Forms надає інтуїтивні засоби для створення та налаштування елементів інтерфейсу користувача.

- **Візуальний дизайнер**: Visual Studio включає візуальний дизайнер для Windows Forms, який дозволяє розробникам перетягувати елементи управління на форму і налаштовувати їхні властивості в інтерактивному режимі.
- **Кодова підтримка**: Під дизайнером генерується код, що робить процес створення інтерфейсу зрозумілим та прозорим.

2) Висока продуктивність

Додатки, створені за допомогою Windows Forms, працюють швидко і ефективно на всіх версіях Windows.

- **Низькі системні вимоги**: Windows Forms додатки, як правило, не вимагають значних ресурсів і можуть ефективно працювати навіть на менш потужних комп’ютерах.
- **Швидкий рендеринг**: Завдяки інтеграції з Windows API, Windows Forms додатки можуть швидко рендерити графічні елементи.

3) Можливості кастомізації

Windows Forms дозволяє легко налаштовувати вигляд і поведінку елементів управління відповідно до вимог конкретного додатка.

- **Події та обробники**: Розробники можуть легко додавати обробники подій для будь-яких елементів управління, що робить додатки інтерактивними та динамічними.
- **Темізація та стилізація**: Можливість налаштування стилів та тем для елементів управління дозволяє створювати унікальні та привабливі інтерфейси.

2.5. Порівняння з іншими технологіями

Для обґрунтування вибору C#, Visual Studio і Windows Forms, розглянемо альтернативи, такі як Python і JavaScript[10].

2.5.1. Python

Python – популярна мова програмування завдяки своїй простоті та широкому спектру бібліотек. Однак для розробки графічних додатків Python має деякі недоліки:

1) Швидкість виконання

Python є інтерпретованою мовою, що може призводити до меншої швидкості виконання програм у порівнянні з компільованими мовами, такими як C#.

- **Інтерпретованість**: Інтерпретовані мови, такі як Python, виконують код рядок за рядком, що може сповільнювати виконання програм, особливо ресурсомістких.
- **Оптимізація**: Хоча існують способи оптимізації Python коду, такі як використання Cython або компіляція в машинний код, вони додають складності в розробці.

2) Графічні бібліотеки

Хоча Python має бібліотеки для створення GUI (наприклад, Tkinter, PyQt), вони часто є менш потужними і зручними у використанні порівняно з Windows Forms.

- **Tkinter**: Хоча Tkinter є стандартною бібліотекою для створення GUI в Python, вона обмежена у своїх можливостях і дизайні.
- **PyQt**: PyQt є більш потужною, але вимагає додаткового навчання та може бути складнішою у використанні для новачків.

2.5.2. JavaScript

JavaScript широко використовується для веб-розробки, але для створення настільних додатків він має деякі обмеження:

1) Складність налагодження

Розробка настільних додатків на JavaScript часто потребує використання додаткових фреймворків (наприклад, Electron), що ускладнює процес налагодження та тестування.

- **Electron**: Хоча Electron дозволяє створювати кросплатформені настільні додатки з використанням веб-технологій, він додає складності в налаштуванні та налагодженні.
- **Інструменти налагодження**: Інструменти налагодження для JavaScript часто орієнтовані на веб-розробку і можуть бути менш ефективними для настільних додатків.

2) Продуктивність

JavaScript не завжди забезпечує таку ж високу продуктивність, як C#, особливо для ресурсомістких додатків.

- **Обмеження продуктивності**: JavaScript, як правило, працює повільніше в порівнянні з компільованими мовами, особливо в середовищах з високими вимогами до продуктивності.
- **Використання ресурсів**: Electron-додатки, наприклад, можуть споживати значну кількість пам'яті і процесорного часу, що робить їх менш ефективними для ресурсомістких задач.

2.6. Додаткові технічні аспекти

1) Безпека

Використання C# та .NET Framework забезпечує високий рівень безпеки завдяки вбудованим механізмам управління пам'яттю та обробки винятків.

- **Управління пам'яттю**: .NET Framework автоматично управляє пам'яттю за допомогою збору сміття, що знижує ризик витоків пам'яті і покращує стабільність додатків.
- **Обробка винятків**: C# має потужну систему обробки винятків, що дозволяє ефективно управляти помилками і забезпечувати надійну роботу додатків.

2) Масштабованість

C# дозволяє легко масштабувати додаток, додаючи нові функції та модулі без значних змін в існуючому коді.

- **Модульність**: Об'єктно-орієнтована природа C# сприяє створенню модульного коду, який легко розширювати і підтримувати.
- **Розширення функціональності**: Завдяки багатому набору бібліотек і фреймворків, розробники можуть легко додавати нові функції без необхідності переписування значної частини існуючого коду.

3) Підтримка спільноти

Велика і активна спільнота розробників C# та Visual Studio забезпечує широкий доступ до ресурсів, бібліотек та інструментів, що значно полегшує процес розробки.

- **Форуми та документація**: Існує безліч форумів, блогів, і веб-сайтів з документацією, де розробники можуть знайти відповіді на свої питання.
- **Відкритий код**: Багато бібліотек і фреймворків для C# доступні у відкритому коді, що дозволяє розробникам вносити свій вклад у їх розвиток і використовувати їх у своїх проектах.

2.7. Алгоритм Прима

Алгоритм Прима (також відомий як **алгоритм Пріма-Джарніка**) є жадібним методом для побудови мінімального кістякового дерева (MST) у зв'язаному неорієнтованому графі. MST — це підграф, який включає всі вершини оригінального графа та має мінімальну сумарну вагу ребер. Названий на честь Роберта Прима, алгоритм був уперше описаний Войтехом Йарніком у 1930 році.

2.7.1. Основні етапи алгоритму Прима

Алгоритм Прима(рис 2.2) базується на жадібному підході, виконуючи послідовне додавання вершин та ребер до MST на основі мінімальної ваги ребра. Його основні кроки такі:

1. **Ініціалізація**:
    - Вибирається довільна початкова вершина графа.
    - Створюється порожня множина для MST та додається обрана вершина.
2. **Розширення MST**:
    - З множини всіх ребер, які з'єднують вершини в MST з вершинами поза MST, вибирається ребро з найменшою вагою, і воно додається до MST разом із новою вершиною.
3. **Повторення**:
    - Процедура повторюється, доки MST не включатиме всі вершини графа.
4. **Завершення**:
    - Процес завершено, коли всі вершини додано до MST.

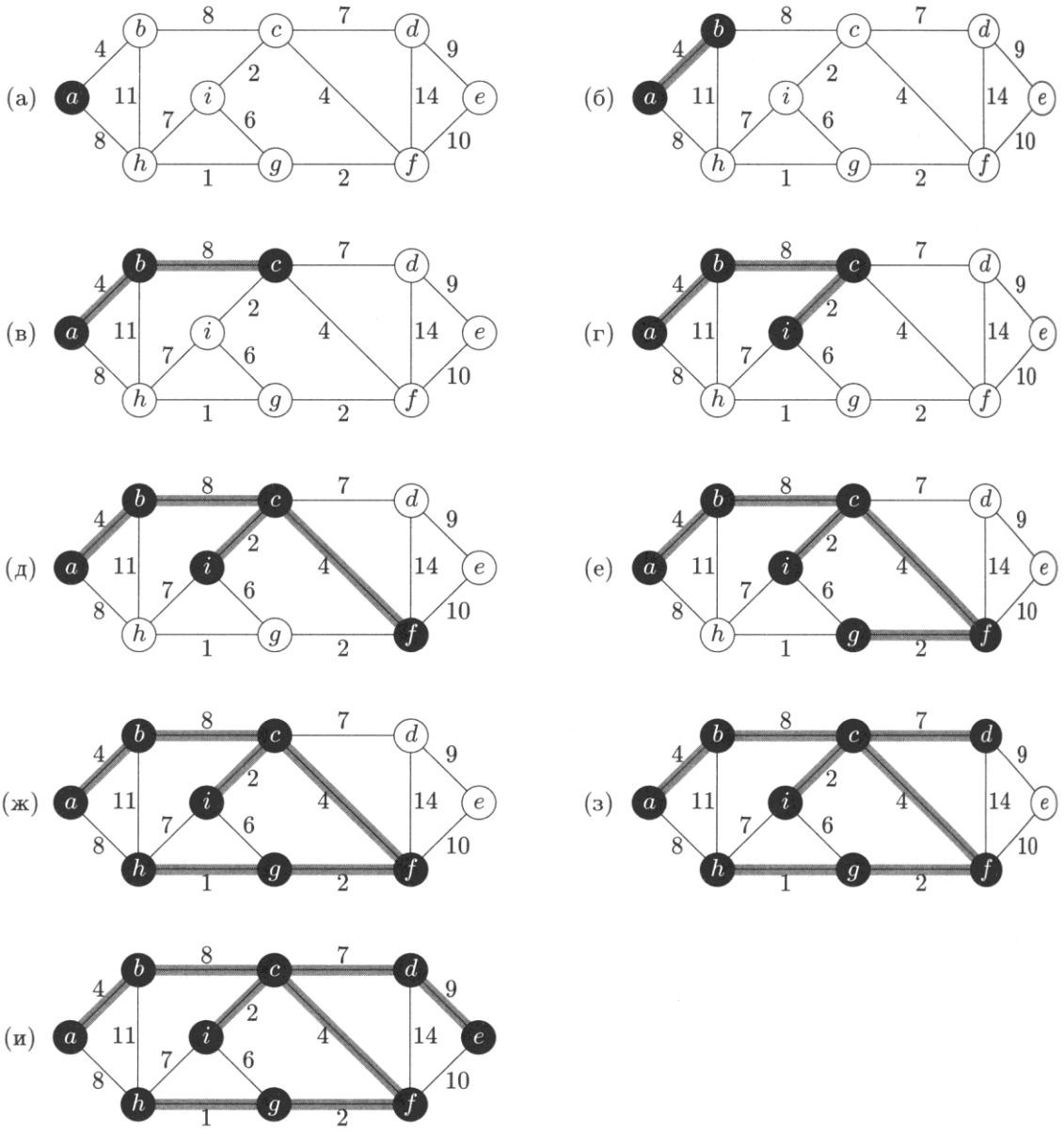

Рисунок 2.2 – Приклад алгоритму Прима

2.7.2. Використання в ігровій розробці

У розробці гри "Динозавр", алгоритм Прима може бути застосований для різних задач, наприклад:

- **Генерація ігрових карт**: Використання MST для створення карт з мінімальною кількістю шляхів або оптимізація існуючих шляхів між об'єктами на карті.
- **Оптимізація мереж**: Включення MST для оптимального розміщення об'єктів або визначення найкоротших шляхів між точками у грі.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 База даних

MySQL є популярною системою управління базами даних (СУБД), яка використовується для зберігання і управління даними в багатьох додатках[11]. Вибір цієї СУБД для нашої гри "Динозавр" обумовлений кількома ключовими факторами:

1) Висока продуктивність

Ця система забезпечує високу швидкість обробки запитів і ефективність роботи з великими обсягами даних.

- **Швидкість запитів**: СУБД оптимізована для швидкої обробки SQL-запитів, що дозволяє швидко отримувати необхідну інформацію з бази даних.
- **Індексація**: Підтримка індексації дозволяє значно прискорити доступ до даних, особливо у великих таблицях.

2) Надійність і стабільність

Ця СУБД відома своєю надійністю та стабільністю, що є критичними аспектами для будь-якої бази даних[12].

- **Механізми збереження**: Використовуються журнали транзакцій і механізми відновлення після збоїв, що забезпечує збереження і цілісність даних.
- **Підтримка транзакцій**: Завдяки підтримці транзакцій, система гарантує, що всі операції з базою даних виконуються коректно і цілісно.

3) Масштабованість

Система легко масштабується для роботи з великими обсягами даних і великою кількістю користувачів.

- **Горизонтальне масштабування**: Підтримка реплікації і шардінгу дозволяє розподіляти навантаження між кількома серверами.
- **Вертикальне масштабування**: Підтримка розширення апаратних ресурсів (процесора, пам'яті) дозволяє покращувати продуктивність бази даних без значних змін у конфігурації.

4) Гнучкість і простота використання

Ця СУБД пропонує простий і інтуїтивно зрозумілий інтерфейс, що полегшує роботу розробників[13].

- **SQL-сумісність**: Підтримка стандартного SQL робить систему зручною для розробників, які вже знайомі з цією мовою.
- **Інструменти адміністрування**: Існує багато інструментів для адміністрування бази даних, таких як phpMyAdmin, MySQL Workbench, які спрощують управління базами даних.

5) Підтримка спільноти та документація

Система має велику спільноту користувачів і розробників, а також велику кількість документації та ресурсів.

- **Форуми та обговорення**: Існує багато форумів і груп обговорення, де можна знайти рішення для будь-яких проблем, що виникають при роботі з цією СУБД.
- **Документація**: Офіційна документація є вичерпною і доступною, що дозволяє швидко знайти необхідну інформацію.

6) Взаємодія з C# та Visual Studio

Система легко інтегрується з мовою програмування C# і середовищем розробки Visual Studio[14].

- **MySQL Connector/NET**: Це офіційний драйвер для платформи .NET, який дозволяє додаткам на C# взаємодіяти з базою даних[15].
- **ORM (Object-Relational Mapping)**: Інтеграція з ORM-фреймворками, такими як Entity Framework, спрощує роботу з базою даних, дозволяючи використовувати об’єктно-орієнтований підхід для доступу до даних.

Архітектура бази повністю відповідає тематиці роботи. На рисунку 3.2 представлена схема бази даних:

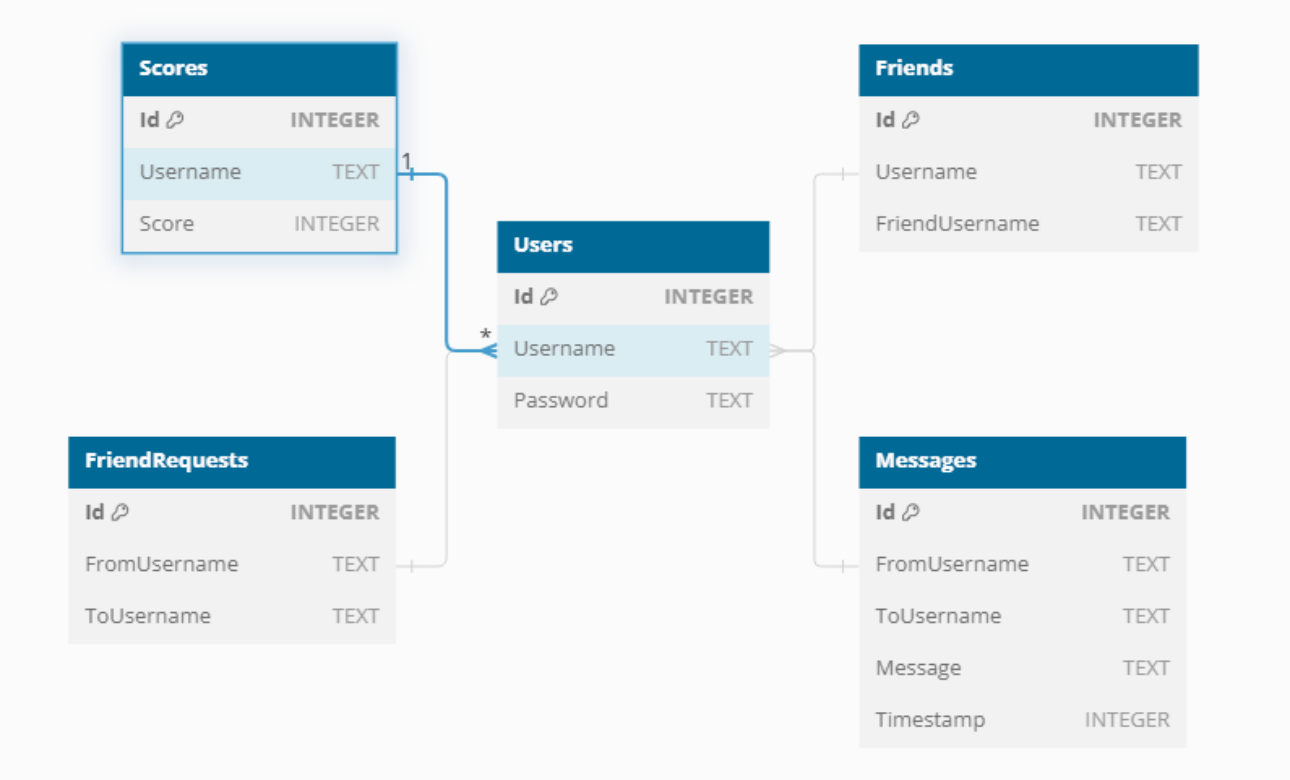

Рисунок 3.1 - Схема бази даних

3.2 Опис програмної реалізації

Щоб реалізувати доступ до системи лише авторизованим корисувачам, було розробленно авторизацію (рис. 3.2).

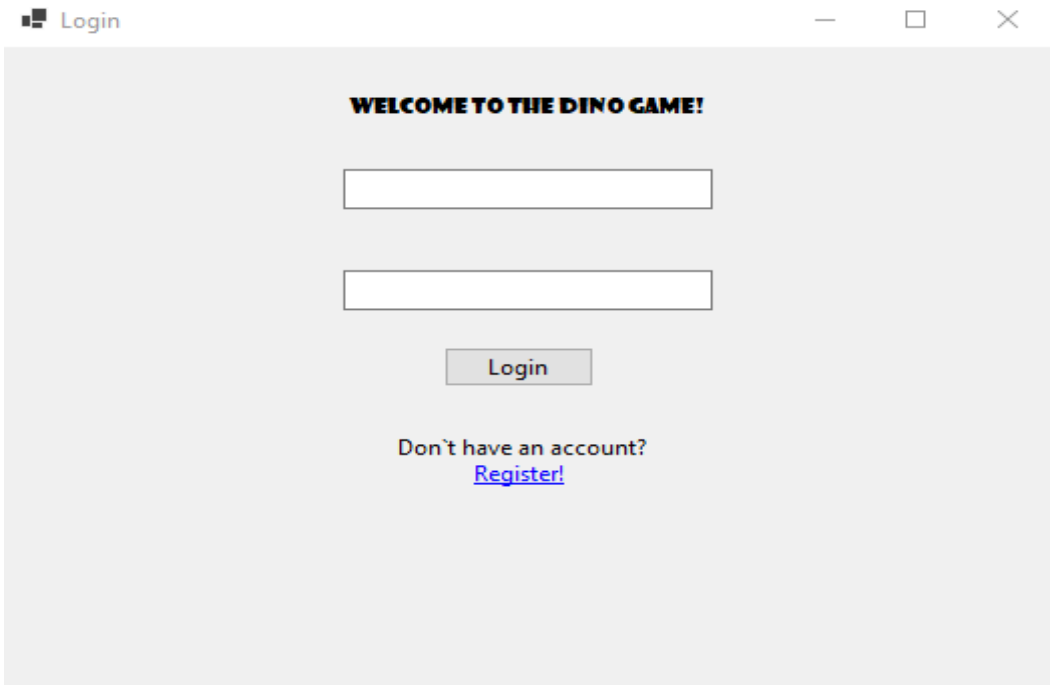

Рисунок 3.2 – Форма авторизації

Якщо користувач ще не створив акаунт, то в нього є можливість зареєструватися(рис. 3.3).

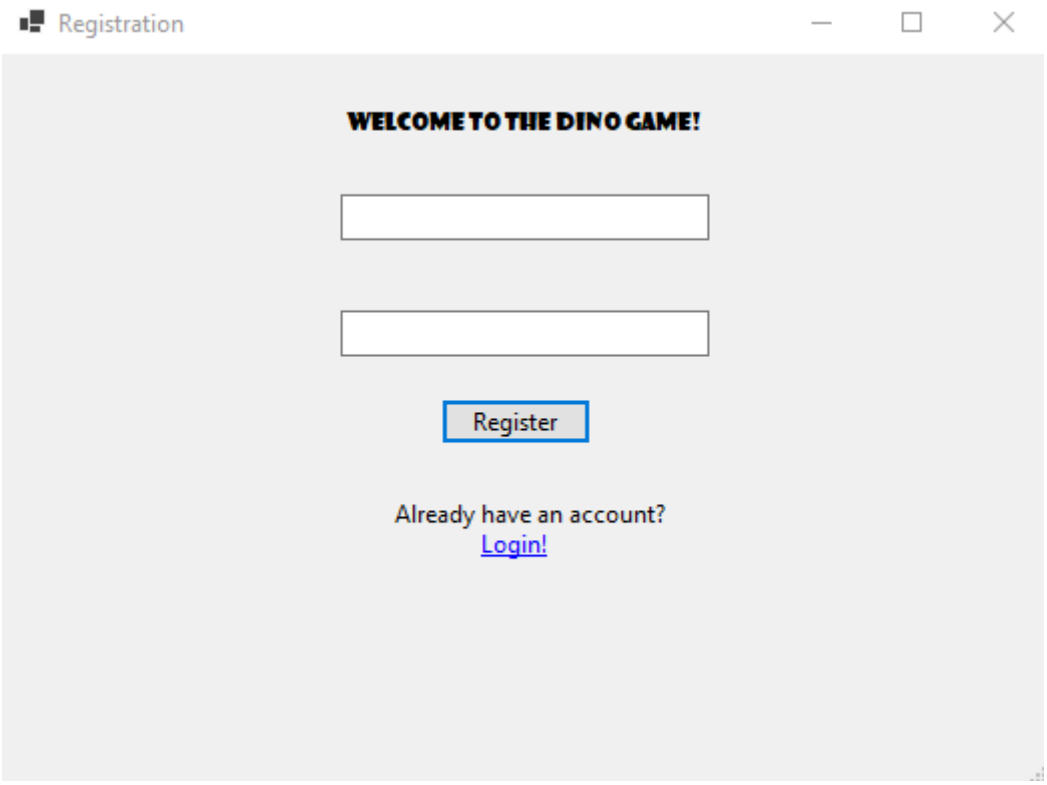

Рисунок 3.3 – Форма реєстрації

Після успішного входу в систему користувач попадає до головного меню, де має змогу почати саму гру, або ж переглянути таблицю рекордів чи додати друзів(рис. 3.4).

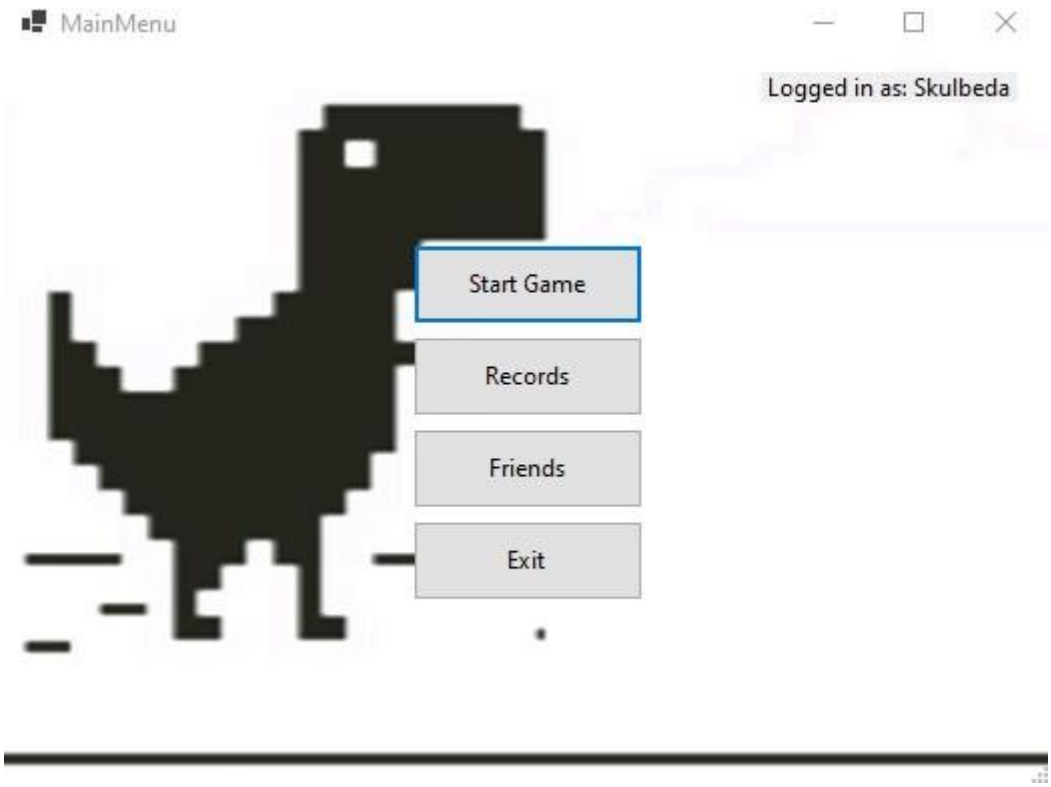

Рисунок 3.4 – Головне меню

При початку гри користувача перекидує на меню гри, де він керує динозавриком(рис. 3.5).

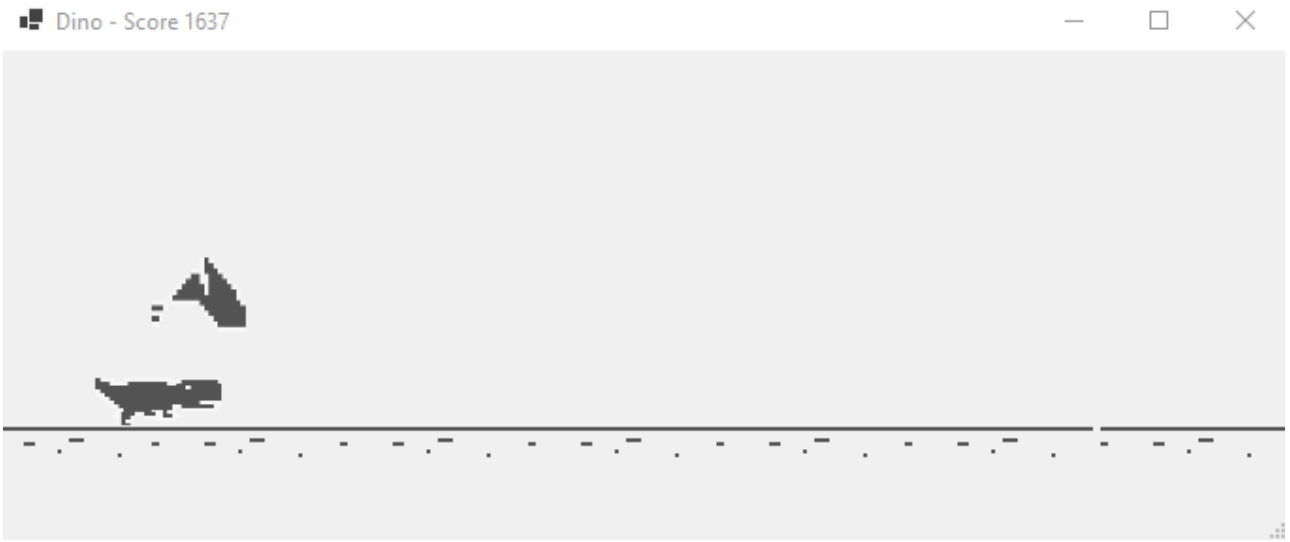

Рисунок 3.5 – Меню гри

Після програшу гравцю дається вибір між виходом на головне меню, переглядом таблиці рекордів, та продовженням гри. У таблиці рекордів ім’я поточного користувача виділяється(рис. 3.6).

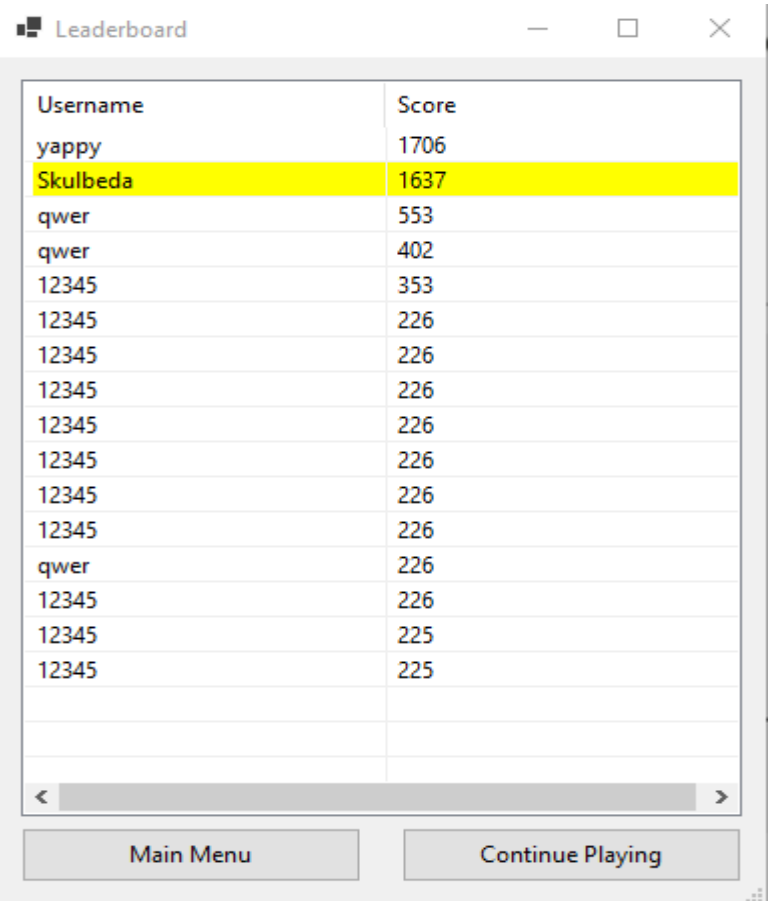

Рисунок 3.6 – Таблиця рекордів

В меню ‘Друзі’ гравець може відшукати якогось гравці, відправляти чи приймати запити на дружбу(рис. 3.7).

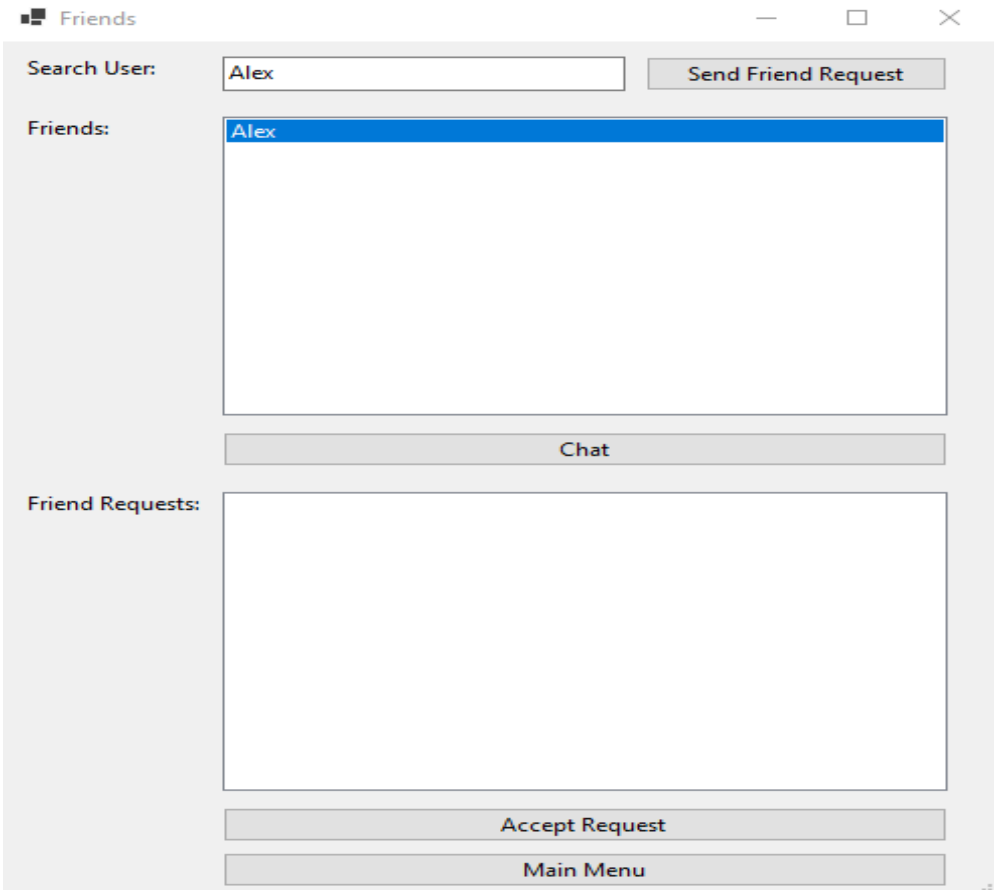

Рисунок 3.7 – Сторінка друзів

Також гравці можуть спілкуватись між собою за допомогою функції чату(рис. 4.8).

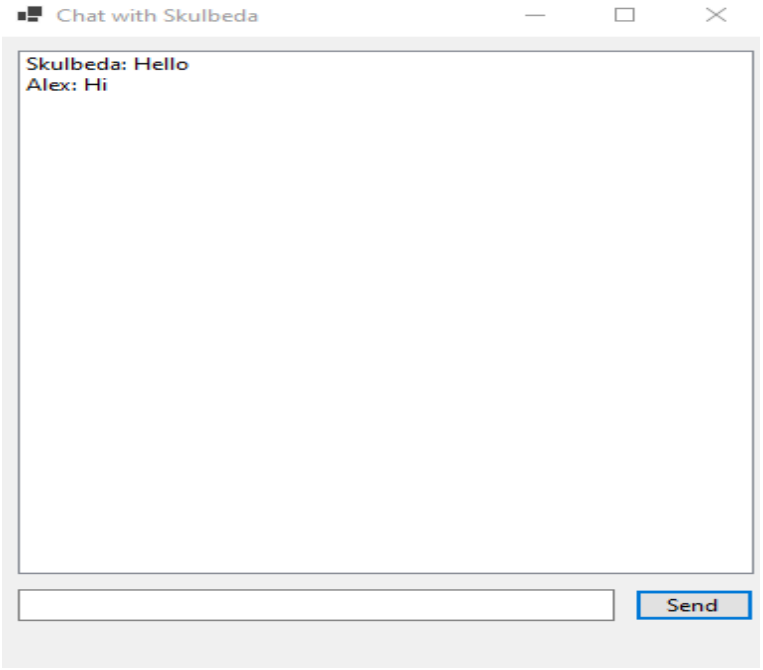

Рисунок 3.8 – Чат

ВИСНОВКИ

У даній дипломній роботі було розроблено та реалізовано гру "Динозавр" із використанням мови програмування C#, середовища розробки Visual Studio 2020, технології Windows Forms та системи управління базами даних MySQL. У процесі роботи над проектом було досягнуто кілька важливих результатів та зроблено низку висновків.

Середовище розробки Visual Studio 2020 забезпечило всі необхідні інструменти для успішного виконання проекту. Інтеграція з мовою програмування C# дозволила використовувати потужні засоби для відладки, рефакторингу коду та управління проектом. Зручний інтерфейс та вбудовані інструменти для тестування підвищили продуктивність розробки та забезпечили високу якість програмного забезпечення.

Використання технології Windows Forms для створення графічного інтерфейсу виявилося ефективним рішенням. Простота створення інтерфейсу, висока продуктивність та можливості кастомізації дозволили розробити зручний та естетично привабливий інтерфейс користувача. Windows Forms також забезпечує стабільну роботу додатка на всіх версіях операційної системи Windows, що є важливою перевагою.

Система управління базами даних MySQL була обрана завдяки своїй високій продуктивності, надійності та масштабованості. СУБД забезпечує швидку обробку запитів, ефективне управління великими обсягами даних та підтримку транзакцій, що гарантує цілісність та збереження даних. Крім того, Таким чином, результати даної дипломної роботи свідчать про успішність вибору технологій та підходів для розробки гри "Динозавр". Використання C#, Visual Studio, Windows Forms та MySQL дозволило створити високопродуктивний, надійний та зручний додаток, що відповідає сучасним вимогам до програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **WikiWand.** Dinoaur Game [Електронний ресурс]: https://www.wikiwand.com /uk/Dinosaur_Game. Доступний стан на 09.02.2024.
2. **Flappy Bird.** [Електронний ресурс]: Вікіпедія. https://uk.wikipedia.org/wiki/Flappy_Bird – Доступний стан на 09.02.2024.
3. **Jerry Momoda**, Endless Runner Games: Evolution and Future. [Електронний ресурс]: Game Analysys. http://jerrymomoda.com/analysis-endless-runners/ Доступний стан на 10.02.2024.
4. **MONSTERAPLAY.** Why Simple Games Like the Chrome Dino Are So Addictive, [Електронний ресурс]: Режим доступу: https://www.monsteraplay.com/articles/how_can_a_casual_game_like_the_dinosaur_game_be_so_addictive – Доступний стан на 12.02.2024.
5. **Jesse Schell.** The Art of Game Design: A Book of Lenses, Third Edition [Книга]. 652 p.
6. **"Модель-Вид-контроллер,"** Вікіпедія. [Електронний ресурс]: https://uk.wikipedia.org/wiki/Модель-Вид-контролер. – Доступний стан на 18.02.2024.
7. **Retro Style Games.** Why C# is the Best Programming Language in the World for Building Games, [Електронний ресурс]. – https://retrostylegames.com /blog/programming-language-for-games/ – Доступний стан на 18.02.2024.
8. **Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio**. [Електронний ресурс]: https://visualstudio.microsoft.com/. Доступний стан на 18.02.2024.
9. **"Introduction to Windows Forms in C#,"** Microsoft Docs, accessed on June 4, 2024. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0. – Доступний стан на 18.02.2024.
10. **MEDIUM.** Decoding the Secrets of Top Programming Languages: JavaScript, Python, Java, C#, and C++**,** [Електронний ресурс]: https://medium.com/@keyboardwordsmith/decoding-the-secrets-of-top-

programming-languages-javascript-python-java-c-and-c-8b184f966bb7. Доступний стан на 18.02.2024.

11. **Документація MySQL,** MySQL. [Електронний ресурс]: https://dev.mysql.com/doc/. Доступний стан на 20.02.2024.
12. **Schwartz, B., Zaitsev, P., & Tkachenko, V.** High Performance MySQL, O'Reilly, – 816 с. [Електронний ресурс]: https://www.oreilly.com/library/view/high-performance-mysql/9781449332471/. – Доступний стан на 20.02.2024.
13. **MySQL Tutorial,** MySQL. [Електронний ресурс]: https://www.mysqltutorial.org/. Доступний стан на 21.02.2024.
14. **Офіційна документація MySQL Connector/NET,** MySQL. [Електронний ресурс]: https://dev.mysql.com/doc/connector-net/en/. – Доступний стан на 21.02.2024.
15. **Entity Framework Documentation,** Microsoft Docs. [Електронний ресурс]: https://docs.microsoft.com/en-us/ef/. – Доступний стан на 22.02.2024.

ДОДАТКИ

Додаток А – Код реалізації форм

LoginForm.cs

```
using DINO.Classes;
using System;
using System.Windows.Forms;

namespace DINO
{
    public partial class LoginForm : Form
    {
        private DatabaseManager dbManager;
        private string currentUsername;

        public LoginForm()
        {
            InitializeComponent();
            dbManager = new DatabaseManager();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            string username = txtLoginUsername.Text;
            string password = txtLoginPassword.Text;

            if (dbManager.Login(username, password))
            {
                MessageBox.Show("Login successful!");
                currentUsername = username;

                MainMenuForm mainMenuForm = new MainMenuForm(currentUsername)
                {
                    StartPosition = FormStartPosition.Manual,
                    Location = this.Location
                };
                this.Hide();
                mainMenuForm.ShowDialog();
                this.Close();
            }
            else
            {
                MessageBox.Show("Invalid username or password.");
            }
        }

        private RegisterForm registerForm;

        private void linkRegister_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            this.Visible = false;

            if (registerForm == null || registerForm.IsDisposed)
            {
```

```
            registerForm = new RegisterForm
            {
                StartPosition = FormStartPosition.Manual,
                Location = this.Location
            };
            registerForm.Closed += (s, args) => this.Visible = true;
        }

        this.Hide();
        registerForm.ShowDialog();
        this.Close();
    }
  }
}
```

RegisterForm.cs

```
using System;
using System.Windows.Forms;
using DINO.Classes;

namespace DINO
{
  public partial class RegisterForm : Form
  {
    private DatabaseManager dbManager;

    public RegisterForm()
    {
      InitializeComponent();
      dbManager = new DatabaseManager();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      string username = txtRegisterUsername.Text;
      string password = txtRegisterPassword.Text;

      if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
      {
        MessageBox.Show("Username and password cannot be empty.");
        return;
      }

      if (dbManager.UserExists(username))
      {
        MessageBox.Show("Username already exists. Please choose a different one.");
        return;
      }

      if (dbManager.Register(username, password))
      {
        MessageBox.Show("Registration successful!");
        LoginForm loginForm = new LoginForm
        {
```

```
                StartPosition = FormStartPosition.Manual,
                Location = this.Location
            };
            this.Hide();
            loginForm.ShowDialog();
            this.Close();
        }
        else
        {
            MessageBox.Show("Registration failed. Please try again.");
        }
    }

    private void linkLogin_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        LoginForm loginForm = new LoginForm
        {
            StartPosition = FormStartPosition.Manual,
            Location = this.Location
        };
        this.Hide();
        loginForm.ShowDialog();
        this.Close();
    }
  }
}
```

MainMenuForm.cs

```
using System;
using System.Windows.Forms;

namespace DINO
{
  public partial class MainMenuForm : Form
  {
    private string currentUsername;

    public MainMenuForm(string username)
    {
        InitializeComponent();
        currentUsername = username;
        lblCurrentUser.Text = $"Logged in as: {currentUsername}";
    }

    private void btnStartGame_Click(object sender, EventArgs e)
    {
        Form1 gameForm = new Form1(currentUsername)
        {
            StartPosition = FormStartPosition.Manual,
            Location = this.Location
        };
        this.Hide();
        gameForm.ShowDialog();
        this.Close();
    }

    private void btnViewRecords_Click(object sender, EventArgs e)
    {
```

```
            RecordForm recordForm = new RecordForm(currentUsername)
            {
                StartPosition = FormStartPosition.Manual,
                Location = this.Location
            };
            this.Hide();
            recordForm.ShowDialog();
            this.Close();
        }

        private void btnFriends_Click(object sender, EventArgs e)
        {
            FriendsForm friendsForm = new FriendsForm(currentUsername)
            {
                StartPosition = FormStartPosition.Manual,
                Location = this.Location
            };
            this.Hide();
            friendsForm.ShowDialog();
            this.Close();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            LoginForm loginForm = new LoginForm()
            {
                StartPosition = FormStartPosition.Manual,
                Location = this.Location
            };
            this.Hide();
            loginForm.ShowDialog();
            this.Close();
        }
    }
}
```

Form1.cs

```
using DINO.Classes;
using System;
using System.Drawing;
using System.Windows.Forms;

namespace DINO
{
    public partial class Form1 : Form
    {
        Player player;
        Timer mainTimer;
        private string currentUsername;
        private DatabaseManager dbManager;

        public Form1(string username)
        {
            InitializeComponent();
            currentUsername = username;
            dbManager = new DatabaseManager();

            this.Width = 700;
```

```
    this.Height = 300;
    this.DoubleBuffered = true;
    this.Paint += new PaintEventHandler(DrawGame);
    this.KeyUp += new KeyEventHandler(OnKeyboardUp);
    this.KeyDown += new KeyEventHandler(OnKeyboardDown);
    mainTimer = new Timer();
    mainTimer.Interval = 10;
    mainTimer.Tick += new EventHandler(Update);

    Init();
}

private void OnKeyboardDown(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.Down:
            if (!player.physics.isJumping)
            {
                player.physics.isCrouching = true;
                player.physics.isJumping = false;
                player.physics.transform.size.Height = 25;
                player.physics.transform.position.Y = 174;
            }
            break;
    }
}

private void OnKeyboardUp(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.Up:
            if (!player.physics.isCrouching)
            {
                player.physics.isCrouching = false;
                player.physics.AddForce();
            }
            break;
        case Keys.Down:
            player.physics.isCrouching = false;
            player.physics.transform.size.Height = 50;
            player.physics.transform.position.Y = 150.2f;
            break;
    }
}

public void Init()
{
    GameController.Init();
    player = new Player(new PointF(50, 149), new Size(50, 50));
    mainTimer.Start();
    Invalidate();
}

public void Update(object sender, EventArgs e)
{
```

```
            player.score++;
            this.Text = "Dino - Score " + player.score;
            if (player.physics.Collide())
            {
                mainTimer.Stop();
                HandleGameOver();
            }

            player.physics.ApplyPhysics();
            GameController.MoveMap();
            Invalidate();
        }

        public void DrawGame(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            player.DrawSprite(g);
            GameController.DrawObjets(g);
        }

        private void HandleGameOver()
        {
            dbManager.SaveScore(currentUsername,  player.score);

            var result = MessageBox.Show("Game Over! Your score is " + player.score + ".\nWould you like to restart the game, view the leaderboard, or return to the main menu?", "Game Over", MessageBoxButtons.YesNoCancel);

            if (result == DialogResult.Yes)
            {
                Init();
                mainTimer.Start();
            }
            else if (result == DialogResult.No)
            {
                RecordForm leaderboardForm = new RecordForm(currentUsername);
                this.Hide();
                leaderboardForm.ShowDialog();
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                MainMenuForm mainMenuForm = new MainMenuForm(currentUsername);
                this.Hide();
                mainMenuForm.ShowDialog();
                this.Close();
            }
        }
    }
}
```

RecordForm.cs

```
using DINO.Classes;
using System;
using System.Collections.Generic;
using System.Drawing;
```

```
using System.Windows.Forms;

namespace DINO
{
    public partial class RecordForm : Form
    {
        private DatabaseManager dbManager;
        private string currentUsername;

        public RecordForm(string username)
        {
            InitializeComponent();
            dbManager = new DatabaseManager();
            currentUsername = username;
            LoadLeaderboard();
        }

        private void LoadLeaderboard()
        {
            List<(string username, int score)> leaderboard = dbManager.GetTopScores();
            listViewLeaderboard.Items.Clear();

            foreach (var entry in leaderboard)
            {
                var item = new ListViewItem(new[] { entry.username, entry.score.ToString() });
                if (entry.username == currentUsername)
                {
                    item.BackColor = Color.Yellow;
                }
                listViewLeaderboard.Items.Add(item);
            }
        }

        private void btnMainMenu_Click(object sender, EventArgs e)
        {
            MainMenuForm mainMenu = new MainMenuForm(currentUsername);
            this.Hide();
            mainMenu.ShowDialog();
            this.Close();
        }

        private void btnContinuePlaying_Click(object sender, EventArgs e)
        {
            Form1 gameForm = new Form1(currentUsername);
            this.Hide();
            gameForm.ShowDialog();
            this.Close();
        }
    }
}
```

FriendsForm.cs

```
using DINO.Classes;
using System;
using System.Collections.Generic;
```

```
using System.Drawing;
using System.Windows.Forms;

namespace DINO
{
    public partial class FriendsForm : Form
    {
        private DatabaseManager dbManager;
        private string currentUsername;
        private TextBox txtSearchUser;
        private ListBox lstFriends;
        private ListBox lstFriendRequests;

        public FriendsForm(string username)
        {
            InitializeComponent();
            dbManager = new DatabaseManager();
            currentUsername = username;
            InitializeFriendFormComponents();
            LoadFriends();
            LoadFriendRequests();
        }

        private void InitializeFriendFormComponents()
        {
            this.Text = "Friends";
            this.Width = 510;
            this.Height = 610;

            Label lblSearch = new Label() { Text = "Search User:", Left = 10, Top = 10 };
            txtSearchUser = new TextBox() { Left = 110, Top = 10, Width = 200 };
            Button btnSendRequest = new Button() { Text = "Send Friend Request", Left = 320, Top = 10, Width = 150
};

            btnSendRequest.Click += BtnSendRequest_Click;

            Label lblFriends = new Label() { Text = "Friends:", Left = 10, Top = 50 };
            lstFriends = new ListBox() { Left = 110, Top = 50, Width = 360, Height = 200 };

            Button btnChat = new Button() { Text = "Chat", Left = 110, Top = 260, Width = 360 };
            btnChat.Click += BtnChat_Click;

            Label lblFriendRequests = new Label() { Text = "Friend Requests:", Left = 10, Top = 300 };
            lstFriendRequests = new ListBox() { Left = 110, Top = 300, Width = 360, Height = 200 };

            Button btnAcceptRequest = new Button() { Text = "Accept Request", Left = 110, Top = 510, Width = 360
};
            btnAcceptRequest.Click += BtnAcceptRequest_Click;

            Button btnMainMenu = new Button() { Text = "Main Menu", Left = 110, Top = 540, Width = 360 };
            btnMainMenu.Click += BtnMainMenu_Click;

            this.Controls.Add(lblSearch);
            this.Controls.Add(txtSearchUser);
            this.Controls.Add(btnSendRequest);
            this.Controls.Add(lblFriends);
            this.Controls.Add(lstFriends);
```

```
    this.Controls.Add(btnChat);
    this.Controls.Add(lblFriendRequests);
    this.Controls.Add(lstFriendRequests);
    this.Controls.Add(btnAcceptRequest);
    this.Controls.Add(btnMainMenu);
}

private void BtnSendRequest_Click(object sender, EventArgs e)
{
    string searchUsername = txtSearchUser.Text;
    if (!string.IsNullOrEmpty(searchUsername))
    {
        bool requestSent = dbManager.SendFriendRequest(currentUsername, searchUsername);
        if (requestSent)
        {
            MessageBox.Show("Friend request sent!");
        }
        else
        {
            MessageBox.Show("Unable to send friend request. User may not exist or request already sent.");
        }
    }
}

private void BtnChat_Click(object sender, EventArgs e)
{
    if (lstFriends.SelectedItem != null)
    {
        string friendUsername = lstFriends.SelectedItem.ToString();
        ChatForm chatForm = new ChatForm(currentUsername, friendUsername);
        chatForm.Show();
    }
}

private void BtnAcceptRequest_Click(object sender, EventArgs e)
{
    if (lstFriendRequests.SelectedItem != null)
    {
        string requestUsername = lstFriendRequests.SelectedItem.ToString();
        dbManager.AcceptFriendRequest(requestUsername, currentUsername);
        LoadFriends();
        LoadFriendRequests();
    }
}

private void BtnMainMenu_Click(object sender, EventArgs e)
{
    MainMenuForm mainMenu = new MainMenuForm(currentUsername)
    {
        StartPosition = FormStartPosition.Manual,
        Location = this.Location
    };
    this.Hide();
    mainMenu.ShowDialog();
    this.Close();
}
```

```
        private void LoadFriends()
        {
            lstFriends.Items.Clear();
            List<string> friends = dbManager.GetFriends(currentUsername);
            foreach (var friend in friends)
            {
                lstFriends.Items.Add(friend);
            }
        }

        private void LoadFriendRequests()
        {
            lstFriendRequests.Items.Clear();
            List<string> friendRequests = dbManager.GetFriendRequests(currentUsername);
            foreach (var request in friendRequests)
            {
                lstFriendRequests.Items.Add(request);
            }
        }
    }
}
```

ChatForm.cs

```
using DINO.Classes;
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace DINO
{
    public partial class ChatForm : Form
    {
        private string currentUsername;
        private string friendUsername;
        private DatabaseManager dbManager;

        public ChatForm(string currentUsername, string friendUsername)
        {
            InitializeComponent();
            this.currentUsername = currentUsername;
            this.friendUsername = friendUsername;
            this.dbManager = new DatabaseManager();
            InitializeChatComponents();
            this.Text = $"Chat with {friendUsername}";
            LoadMessages();
        }

        private void InitializeChatComponents()
        {
            this.Width = 400;
            this.Height = 500;

            TextBox txtMessage = new TextBox() { Left = 10, Top = 400, Width = 300 };
            Button btnSend = new Button() { Text = "Send", Left = 320, Top = 400, Width = 60 };
            ListBox lstMessages = new ListBox() { Left = 10, Top = 10, Width = 370, Height = 380 };
            lstMessages.TabIndex = 2;
            btnSend.Click += (sender, e) =>
            {
```

```
            string message = txtMessage.Text;
            if (!string.IsNullOrEmpty(message))
            {
                dbManager.SendMessage(currentUsername, friendUsername, message);
                lstMessages.Items.Add($"{currentUsername}: {message}");
                txtMessage.Clear();
            }
        };

        this.Controls.Add(txtMessage);
        this.Controls.Add(btnSend);
        this.Controls.Add(lstMessages);
    }

    private void LoadMessages()
    {
        ListBox lstMessages = (ListBox)this.Controls[2];
        var messages = dbManager.GetMessages(currentUsername, friendUsername);
        lstMessages.Items.Clear();
        foreach (var msg in messages)
        {
            lstMessages.Items.Add($"{msg.fromUser}:  {msg.message}");
        }
    }
  }
}
```

Додаток Б – Код реалізації класів

Player.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
{
   public class Player
   {
      public Physics physics;
      public int frameCount = 0;
      public int animationCount = 0;
      public int score = 0;

      public Player(PointF position, Size size)
      {
         physics = new Physics(position, size);
         frameCount = 0;
         score = 0;
      }

      public void DrawSprite(Graphics g)
      {
         if (physics.isCrouching)
         {
            DrawNeedSprite(g, 1870, 40, 109, 51, 118, 1.35f);
         }
         else
         {
            DrawNeedSprite(g, 1518, 0, 79, 91, 88, 1);
         }
      }

      public void DrawNeedSprite(Graphics g, int srcX, int srcY, int width, int height, int delta, float multiplier)
      {
         frameCount++;
         if (frameCount <= 10)
            animationCount = 0;
         else if (frameCount > 10 && frameCount <= 20)
            animationCount = 1;
         else if (frameCount > 20)
            frameCount = 0;

         g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)physics.transform.position.X,
(int)physics.transform.position.Y), new Size((int)(physics.transform.size.Width * multiplier),
physics.transform.size.Height)), srcX + delta * animationCount, srcY, width, height, GraphicsUnit.Pixel);

      }
   }
}
```

GameController.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
{
   public class GameController
   {
      public static Image spritesheet;
      public static List<Road> roads;
      public static List<Cactus> cactuses;
      public static List<Bird> birds;

      public static int dangerSpawn=10;
      public static int countDangerSpawn = 0;


      public static void Init()
      {
         roads = new List<Road>();
         cactuses = new List<Cactus>();
         birds = new List<Bird>();
         spritesheet = Properties.Resources.sprite;
         GenerateRoad();
      }

      public static void MoveMap()
      {
         for (int i = 0; i < roads.Count; i++)
         {
            roads[i].transform.position.X -= 4;
            if (roads[i].transform.position.X + roads[i].transform.size.Width < 0)
            {
               roads.RemoveAt(i);
               GetNewRoad();
            }
         }

         for (int i = 0; i < cactuses.Count; i++)
         {
            cactuses[i].transform.position.X -= 4;
            if (cactuses[i].transform.position.X + cactuses[i].transform.size.Width < 0)
            {
               cactuses.RemoveAt(i);
            }
```

```
    }

    for (int i = 0; i < birds.Count; i++)
    {
        birds[i].transform.position.X -= 4;
        if (birds[i].transform.position.X + birds[i].transform.size.Width < 0)
        {
            birds.RemoveAt(i);
        }
    }
}

public static void GetNewRoad()
{
    Road road = new Road(new PointF(0 + 100 * 9, 200), new Size(100, 17));
    roads.Add(road);
    countDangerSpawn++;

    if (countDangerSpawn >= dangerSpawn)
    {
        Random r = new Random();
        dangerSpawn = r.Next(5, 9);
        countDangerSpawn = 0;
        int obj = r.Next(0, 2);
        switch (obj)
        {
            case 0:
                Cactus cactus = new Cactus(new PointF(0 + 100 * 9, 150), new Size(50, 50));
                cactuses.Add(cactus);
                break;
            case 1:
                Bird bird = new Bird(new PointF(0 + 100 * 9, 110), new Size(50, 50));
                birds.Add(bird);
                break;
        }
    }
}
public static void GenerateRoad()
{
    for (int i = 0; i < 10; i++)
    {
        Road road = new Road(new PointF(0 + 100 * i, 200), new Size(100, 17));
        roads.Add(road);
        countDangerSpawn++;
    }
}
public static void DrawObjets(Graphics g)
{
    for (int i = 0; i < roads.Count; i++)
```

```
            {
                roads[i].DrawSprite(g);
            }
            for (int i = 0; i < cactuses.Count; i++)
            {
                cactuses[i].DrawSprite(g);
            }
            for (int i = 0; i < birds.Count; i++)
            {
                birds[i].DrawSprite(g);
            }
        }
    }
}
```

DatabaseManager.cs

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

namespace DINO.Classes
{
    public class DatabaseManager
    {
        public string connectionString = "Data
Source=C:/Users/HellPlay/source/repos/DINO/DINO/sql/DINOUserData.db";

        public bool Login(string username, string password)
        {
            using (SQLiteConnection connection = new SQLiteConnection(connectionString))
            {
                connection.Open();
                string query = "SELECT COUNT(*) FROM Users WHERE Username = @Username AND Password =
@Password";
                using (SQLiteCommand command = new SQLiteCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@Username", username);
                    command.Parameters.AddWithValue("@Password",  password);
                    long count = (long)command.ExecuteScalar();
                    return count > 0;
                }
            }
        }

        public bool Register(string username, string password)
        {
            using (SQLiteConnection connection = new SQLiteConnection(connectionString))
            {
                connection.Open();
                string query = "INSERT INTO Users (Username, Password) VALUES (@Username, @Password)";
                using (SQLiteCommand command = new SQLiteCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@Username",  username);
```

```
            command.Parameters.AddWithValue("@Password", password);
            int rowsAffected = command.ExecuteNonQuery();
            return rowsAffected > 0;
        }
    }
}

public bool UserExists(string username)
{
    using (SQLiteConnection connection = new SQLiteConnection(connectionString))
    {
        connection.Open();
        string query = "SELECT COUNT(*) FROM Users WHERE Username = @Username";
        using (SQLiteCommand command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Username", username);
            long count = (long)command.ExecuteScalar();
            return count > 0;
        }
    }
}

public List<(string username, int score)> GetTopScores()
{
    List<(string username, int score)> scores = new List<(string username, int score)>();

    using (SQLiteConnection connection = new SQLiteConnection(connectionString))
    {
        connection.Open();
        string query = "SELECT Username, Score FROM Scores ORDER BY Score DESC LIMIT 100";
        using (SQLiteCommand command = new SQLiteCommand(query, connection))
        {
            using (SQLiteDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string username = reader.GetString(0);
                    int score = reader.GetInt32(1);
                    scores.Add((username, score));
                }
            }
        }
    }
    return scores;
}

public int GetCurrentPlayerBestScore(string username)
{
    using (SQLiteConnection connection = new SQLiteConnection(connectionString))
    {
        connection.Open();
        string query = "SELECT MAX(Score) FROM Scores WHERE Username = @Username";
        using (SQLiteCommand command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Username", username);
            object result = command.ExecuteScalar();
            return result != DBNull.Value ? Convert.ToInt32(result) : 0;
```

```
            }
        }
    }

    public void SaveScore(string username, int score)
    {
        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();
            string query = "INSERT INTO Scores (Username, Score) VALUES (@Username, @Score)";
            using (SQLiteCommand command = new SQLiteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Username", username);
                command.Parameters.AddWithValue("@Score", score);
                command.ExecuteNonQuery();
            }
        }
    }

    public bool SendFriendRequest(string fromUsername, string toUsername)
    {
        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();

            string userCheckQuery = "SELECT COUNT(*) FROM Users WHERE Username = @ToUsername";
            using (SQLiteCommand userCheckCommand = new SQLiteCommand(userCheckQuery, connection))
            {
                userCheckCommand.Parameters.AddWithValue("@ToUsername", toUsername);
                long userCount = (long)userCheckCommand.ExecuteScalar();
                if (userCount == 0) return false;
            }

            string requestCheckQuery = "SELECT COUNT(*) FROM FriendRequests WHERE FromUsername =
@FromUsername AND ToUsername = @ToUsername";
            using (SQLiteCommand requestCheckCommand = new SQLiteCommand(requestCheckQuery,
connection))
            {
                requestCheckCommand.Parameters.AddWithValue("@FromUsername", fromUsername);
                requestCheckCommand.Parameters.AddWithValue("@ToUsername", toUsername);
                long requestCount = (long)requestCheckCommand.ExecuteScalar();
                if (requestCount > 0) return false;
            }

            string query = "INSERT INTO FriendRequests (FromUsername, ToUsername) VALUES
(@FromUsername, @ToUsername)";
            using (SQLiteCommand command = new SQLiteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@FromUsername", fromUsername);
                command.Parameters.AddWithValue("@ToUsername", toUsername);
                command.ExecuteNonQuery();
                return true;
            }
        }
    }

    public List<string> GetFriendRequests(string username)
```

```
    {
        List<string> friendRequests = new List<string>();

        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();
            string query = "SELECT FromUsername FROM FriendRequests WHERE ToUsername = @Username";
            using (SQLiteCommand command = new SQLiteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Username", username);
                using (SQLiteDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        friendRequests.Add(reader.GetString(0));
                    }
                }
            }
        }

        return friendRequests;
    }

    public void AcceptFriendRequest(string fromUsername, string toUsername)
    {
        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();

            string addFriendQuery = "INSERT INTO Friends (Username, FriendUsername) VALUES
(@Username1, @Username2), (@Username2, @Username1)";
            using (SQLiteCommand command = new SQLiteCommand(addFriendQuery, connection))
            {
                command.Parameters.AddWithValue("@Username1", fromUsername);
                command.Parameters.AddWithValue("@Username2", toUsername);
                command.ExecuteNonQuery();
            }

            string removeRequestQuery = "DELETE FROM FriendRequests WHERE FromUsername =
@FromUsername AND ToUsername = @ToUsername";
            using (SQLiteCommand command = new SQLiteCommand(removeRequestQuery, connection))
            {
                command.Parameters.AddWithValue("@FromUsername", fromUsername);
                command.Parameters.AddWithValue("@ToUsername", toUsername);
                command.ExecuteNonQuery();
            }
        }
    }

    public List<string> GetFriends(string username)
    {
        List<string> friends = new List<string>();

        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();
            string query = "SELECT FriendUsername FROM Friends WHERE Username = @Username";
```

```
            using (SQLiteCommand command = new SQLiteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Username", username);
                using (SQLiteDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        friends.Add(reader.GetString(0));
                    }
                }
            }
        }

        return friends;
    }

    public List<(string fromUser, string message, DateTime timestamp)> GetMessages(string username, string friendUsername)
    {
        List<(string fromUser, string message, DateTime timestamp)> messages = new List<(string fromUser, string message, DateTime timestamp)>();

        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();
            string query = "SELECT FromUsername, Message, Timestamp FROM Messages WHERE (FromUsername = @Username AND ToUsername = @FriendUsername) OR (FromUsername = @FriendUsername AND ToUsername = @Username) ORDER BY Timestamp";
            using (SQLiteCommand command = new SQLiteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Username", username);
                command.Parameters.AddWithValue("@FriendUsername", friendUsername);
                using (SQLiteDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        string fromUser = reader.GetString(0);
                        string message = reader.GetString(1);
                        DateTime timestamp = reader.GetDateTime(2);
                        messages.Add((fromUser, message, timestamp));
                    }
                }
            }
        }

        return messages;
    }

    public void SendMessage(string fromUsername, string toUsername, string message)
    {
        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();
            string query = "INSERT INTO Messages (FromUsername, ToUsername, Message, Timestamp) VALUES (@FromUsername, @ToUsername, @Message, @Timestamp)";
            using (SQLiteCommand command = new SQLiteCommand(query, connection))
            {
```

```
                command.Parameters.AddWithValue("@FromUsername", fromUsername);
                command.Parameters.AddWithValue("@ToUsername", toUsername);
                command.Parameters.AddWithValue("@Message", message);
                command.Parameters.AddWithValue("@Timestamp", DateTime.Now);
                command.ExecuteNonQuery();
            }
        }
    }
}
```

Transform.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
{
    public class Transform
    {
        public PointF position;
        public Size size;

        public Transform(PointF pos, Size size) {
            this.position = pos;
            this.size = size;
        }
    }
}
```

Physics.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
```

```
{
    public class Physics
    {
        public Transform transform;
        float gravity;
        float a;

        public bool isJumping;
        public bool isCrouching;


        public Physics(PointF position, Size size)
        {
            transform = new Transform(position, size);
            gravity = 0;
            a = 0.4f;
            isJumping = false;
            isCrouching = false;
        }

        public void ApplyPhysics()
        {
            CalculatePhysics();
        }

        public void CalculatePhysics()
        {
            if (transform.position.Y<150 || isJumping)
            {
                transform.position.Y += gravity;
                gravity += a;
            }
            if (transform.position.Y > 150)
                isJumping = false;
        }

        public bool Collide()
        {
            for (int i = 0; i < GameController.cactuses.Count; i++)
            {
                var cactus = GameController.cactuses[i];
                PointF delta = new PointF();
                delta.X = (transform.position.X + transform.size.Width / 2) - (cactus.transform.position.X +
cactus.transform.size.Width / 2);
                delta.Y = (transform.position.Y + transform.size.Height / 2) - (cactus.transform.position.Y +
cactus.transform.size.Height / 2);
                if (Math.Abs(delta.X) <= transform.size.Width / 2 + cactus.transform.size.Width / 2)
                {
                    if (Math.Abs(delta.Y) <= transform.size.Height / 2 + cactus.transform.size.Height / 2)
```

```
                {
                    return true;
                }
            }
        }

        for (int i = 0; i < GameController.birds.Count; i++)
        {
            var bird = GameController.birds[i];
            PointF delta = new PointF();
            delta.X = (transform.position.X + transform.size.Width / 2) - (bird.transform.position.X +
bird.transform.size.Width / 2);
            delta.Y = (transform.position.Y + transform.size.Height / 2) - (bird.transform.position.Y +
bird.transform.size.Height / 2);
            if (Math.Abs(delta.X) <= transform.size.Width / 2 + bird.transform.size.Width / 2)
            {
                if (Math.Abs(delta.Y) <= transform.size.Height / 2 + bird.transform.size.Height / 2)
                {
                    return true;
                }
            }
        }

        return false;
    }

    public void AddForce()
    {
        if (!isJumping)
        {
            isJumping = true;
            gravity = -10;
        }
    }
  }
}
```

Road.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
{
  public class Road
```

```
    {
        public Transform transform;

        public Road(PointF pos, Size size)
        {
            transform = new Transform(pos, size);
        }

        public void DrawSprite(Graphics g)
        {
            g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)transform.position.X,
(int)transform.position.Y), new Size(transform.size.Width, transform.size.Height)), 2300, 112, 100, 17,
GraphicsUnit.Pixel);
        }
    }
}
```

Cactus.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
{
    public class Cactus
    {
        public Transform transform;
        int srcX = 0;

        public Cactus(PointF pos, Size size)
        {
            transform = new Transform(pos, size);
            ChooseRandomPic();
        }

        public void ChooseRandomPic()
        {
            Random r = new Random();
            int rnd = r.Next(0, 4);
            switch (rnd)
            {
                case 0:
                    srcX = 754;
                    break;
                case 1:
```

```
                srcX = 804;
                break;
            case 2:
                srcX = 704;
                break;
            case 3:
                srcX = 654;
                break;
        }
    }

    public void DrawSprite(Graphics g)
    {
        g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)transform.position.X,
(int)transform.position.Y), new Size(transform.size.Width, transform.size.Height)), srcX, 0, 48, 100,
GraphicsUnit.Pixel);
    }
  }
}
```

Bird.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DINO.Classes
{
  public class Bird
  {
    public Transform transform;
    int frameCount = 0;
    int animationCount = 0;

    public Bird(PointF pos, Size size)
    {
      transform = new Transform(pos, size);
    }

    public void DrawSprite(Graphics g)
    {
      frameCount++;
      if (frameCount <= 10)
        animationCount = 0;
      else if (frameCount > 10 && frameCount <= 20)
        animationCount = 1;
```

```
            else if (frameCount > 20)
                frameCount = 0;

            g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)transform.position.X,
(int)transform.position.Y), new Size(transform.size.Width, transform.size.Height)), 246+92*animationCount, 6,
83, 71, GraphicsUnit.Pixel);
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DINO
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LoginForm());
        }
    }
}
```