Міністерство освіти і науки України

Кам’янець-Подільський національний університет імені Івана ОгієнкаФізико-математичний факультет

Кафедра комп’ютерних наук

Дипломна робота бакалавра

з теми: «Розробка онлайн-платформи для гри в шахи»

Виконав: здобувач вищої освіти групи KN1-B20

спеціальності 122 Комп’ютерні науки

Колесник Денис Олександрович

(прізвище, ім’я та по батькові здобувача вищої освіти)

Керівник: Моцик Ростислав Васильович, кандидат педагогічних наук, доцент

(прізвище, ім’я та по батькові, науковий ступінь, вчене звання керівника)

Рецензент: Сморжевський Ю. Л, кандидат педагогічних наук, доцент кафедри математики

(прізвище, ім’я та по батькові, науковий ступінь, вчене звання рецензента)

м. Кам’янець-Подільський – 2024 р.

АНОТАЦІЯ

Дипломна робота “Розробка онлайн-платформи для гри в шахи”

Кам’янець-Подільський національний університет імені Івана Огієнка

Кафедра комп’ютерних наук

Студент — Колесник Денис Олександрович

Керівник — кандидат педагогічних наук, доцент, Моцик Ростислав Васильович

Дана робота присвячена дослідженню та розробці онлайн-платформи для гри в шахи. У вступі визначено актуальність теми та мету дослідження.

У першому розділі наведено історію розвитку комп'ютерних ігор з акцентом на шахи, проведено порівняльну характеристику онлайн-платформ для створення ігрових застосунків та обґрунтовано необхідність створення нових інструментів для навчання новачків.

Другий розділ присвячено безпосередньо розробці платформи. Описано процес створення бібліотеки з алгоритмами гри в шахи, розробку шахової дошки та фігур засобами Unity, створення бази даних за допомогою SQL Server Management Studio та створення клієнт-сервера. У висновках підсумовано основні результати дослідження.

Ключові слова: онлайн-платформа, шахи, комп'ютерні ігри, Unity, база даних, SQL Server Management Studio, клієнт-сервер, алгоритми гри, навчання новачків, ігрові застосунки, історія комп'ютерних ігор.

Зміст

ВСТУП

Актуальність. На сьогодні розробка онлайн-платформи для гри в шахи є надзвичайно важливим завданням, вона зумовлена завдяки зростаючій популярності гри шахи і необхідній допомозі новачкам, які хочуть розпочати своє навчання основам цієї гри, саме для цього необхідно створювати різні інструменти для навчання нових гравців. Також для ще більшої популяризації цієї гри потрібно додавати нові функції в додатки з шахами, щоб дедалі більше людей цікавилось, розвивалось завдяки шахам. Розроблена мною платформа надає новачкам нетиповий і водночас гнучкий формат для вивчення усіх правил і стратегій для гри в шахи.

Об'єктом дослідження є усі кроки у навчанні та грі у шахи для нових гравців а також інші онлайн-платформи, завдяки дослідженню яких я можу ввести нові функції для розширення аудиторії гравців у шахи і навчання новеньких, ще недосвідчених людей, які раніше не грали, або взагалі не знайомі з цією грою.

Предмет дослідження — розробка онлайн-платформи для гри в шахи з допоміжними функціями для нових гравців.

Метою дослідження є створення онлайн-платформи з функціоналом, який спрямовано на підтримку початківців у шахах. Завданням даного дослідження є реалізація помічника для нових гравців та інтегрувати можливість транслювати ігри на платформі для стримів "Twitch".

Завдання дослідження:

1. Аналіз сучасних онлайн-платформ гри в шахи
2. Аналіз сучасних технологій створення комп'ютерних ігор
3. Провести порівняльну характеристику платформ для створення ігор онлайн

4. Розробка онлайн-платформи для гри в шахи

Методи дослідження. Для втілення цієї мети використовувалися методи ООП, також двигун “Unity” для побудови графічної складової програми, а також “SQL Server Management Studio” для створення і коректної роботи бази даних яка буде забезпечувати безперебійну роботу створеного додатку.

Практичне значення отриманих результатів. Моя шахова онлайн-платформа з функцією допомоги новачкам може стати в нагоді новачкам які освоюють шахи, а також вона є засобом поширення цієї чудової гри на широкий загал за допомогою трансляцій, які можна запустити прямо з застосунку. Вона буде сприяти розвитку шахової культури і буде підвищувати особистий рівень гри як досвідчених, так і нових користувачів

Структура дипломної роботи. Робота складається з вступу, двох розділів, у яких висвітлюються хід, опис платформи і результати мого дослідження, також у роботі наявний список використаних ресурсів, необхідних для виконання дослідження а також висновок про значимість цього проекту.

Під час виконання роботи моя увага була зосереджена на розробку простої онлайн-шахової платформи, яка має особливість транслювати хід гри у прямому ефірі на онлайн сервісі для прямих трансляцій.

РОЗДІЛ 1. ОНЛАЙН-ПЛАТФОРМИ, СТВОРЕННЯ ІГРОВИХ ЗАСТОСУНКІВ

1.1 Історія розвитку комп’ютерних ігор, зокрема шахів

Сучасні технології створення комп'ютерних ігор значно еволюціонували, охоплюючи такі аспекти, як високоякісна графіка, реалістична фізика та штучний інтелект. Використання потужних ігрових рушіїв, таких як Unreal Engine та Unity, дозволяє розробникам створювати захоплюючі візуальні світи з детальною графікою та складними анімаціями. Інтеграція реалістичної фізики та динамічних освітлювальних ефектів сприяє створенню більш поглинаючого ігрового досвіду. Крім того, застосування алгоритмів штучного інтелекту дозволяє покращити поведінку NPC, роблячи їх більш реалістичними та адаптивними. Віртуальна реальність та доповнена реальність відкривають нові можливості для ігрового процесу, надаючи гравцям можливість повного занурення у віртуальні світи. Розвиток хмарних технологій та потокової передачі даних також змінює спосіб розповсюдження та грації у відеоігри, забезпечуючи доступ до високоякісних ігор на різних платформах.

Вибір платформи для створення онлайн гри є критично важливим, оскільки він впливає на продуктивність, сумісність та подальшу підтримку проекту. Кожна платформа має свої технічні можливості та обмеження, що можуть вплинути на якість гри. Важливо також враховувати, на яких пристроях гравці будуть грати, і які ринки та аудиторії платформа дозволить охопити. Додатково, платформа повинна підтримувати інтеграцію з мультиплеєрними сервісами, аналітикою, а також забезпечувати можливість майбутніх оновлень. Правильний вибір платформи може визначити комерційний успіх гри та задоволення користувачів, тому він вимагає ретельного аналізу та зважування всіх факторів.

Було проаналізовано 4 платформи для створення ігор онлайн, результати цього дослідження наведено у таблиці 1.1:

Платформа	Легкість використання	Функціональні можливості	Вартість	Спільнота
Unity	Підходить для новаків і професіоналів	Підтримка 2D і 3D, VR/AR,	Безкоштовна версія з обмеженнями, платні плагіни	Велика спільнота, активний форум
Unreal Engine	Високий поріг входження через складний інтерфейс	Гарна графіка, реалістична фізика, підтримка VR/AR	Безкоштовне використання з роялті після певного доходу	Активна спільнота, багато навчальних ресурсів
Godot	Інтуїтивний інтерфейс, що полегшує роботу новачкам	Підтримка 2 і 3 графіки, відкритий вихідний код	Повністю безкоштовний та відкритий	Зростаюча спільнота
Construct	Дуже дружній до новачків, візуальне програмування без коду	Підтримка 2 ігор, обмежена підтримка 3	Платні підписки з різними рівнями доступу до функцій	Активна спільнота, багато прикладів і шаблонів

Таблиця 1.1 Порівняння різних платформ для створення ігор

Початки історії розвитку комп’ютерних ігор лежать в університетах та лабораторіях, де ентузіасти створювали перші комп'ютерні ігри, як експерименти. Такі ігри, як "OXO" та "Spacewar!", демонстрували можливості ранніх комп'ютерів і закладали основи для майбутнього розвитку індустрії.

Згодом індустрія почала формуватися з появою аркадних автоматів, які зробили комп'ютерні ігри доступними широкій аудиторії. Ігри на зразок "Pong" швидко здобули популярність і започаткували нову еру розваг. Наступним кроком стало поширення домашніх ігрових консолей, таких як Atari 2600, які дозволили грати вдома. Це призвело до вибухового зростання індустрії.

Однак індустрія зіткнулася з кризою через перенасичення ринку низькоякісними іграми, що спричинило тимчасовий спад. Проте ринок відновився завдяки появі нових високоякісних ігор та консолей. З цього моменту почався так званий золотий вік відеоігор, коли з'явилися класичні ігри та консолі від компаній, таких як Nintendo та Sega, що встановили нові стандарти якості та ігрового процесу.

Перехід до тривимірної графіки кардинально змінив індустрію. Ігри стали більш реалістичними та складними, з'явилися нові жанри та можливості для гравців. Інтернет також революціонізував комп'ютерні ігри, дозволяючи грати з іншими гравцями по всьому світу. Масові багатокористувацькі онлайн-ігри стали новим феноменом.

Сьогодні комп'ютерні ігри охоплюють широкий спектр жанрів і платформ, включаючи мобільні пристрої, консолі та ПК. Віртуальна реальність та доповнена реальність відкривають нові горизонти для розвитку ігор, а кіберспорт став глобальним явищем, залучаючи мільйони глядачів і професійних гравців.

Ігри стали важливою частиною сучасної культури, впливаючи на кіно, літературу та мистецтво. Вони також використовуються в освіті, медицині та інших галузях, демонструючи свій потенціал як інструмент навчання та

терапії. Історія розвитку комп'ютерних ігор відображає їхній постійний розвиток і зростаюче значення у сучасному світі.

Комп'ютерні ігри стали важливою частиною сучасної культури і технічного прогресу. Особливе місце серед них займають шахи, якими інженери та програмісти цікавилися з самого початку розвитку комп'ютерних технологій. Завдяки своїй складності та глибокій стратегічній природі шахи стали чудовим полігоном для розробки алгоритмів штучного інтелекту та теорії ігор. У цьому підрозділі я розглянув історію комп'ютерних ігор, а особливу увагу приділив грі шахи, від їхнього зародження до наших днів.

Історія автоматизації шахів бере свій початок ще до появи електронних комп'ютерів. У 1769 році угорським інженером було побудовано механічний пристрій під назвою "Турок", який імітував гру в шахи. Насправді це була шахрайська машина, якою керувала людина, схована всередині, але ідея автоматизації шахів викликала великий інтерес.

З появою електронного комп'ютера в середині 20-го століття шахи стали першою грою, яку намагались програмувати для машинного виконання. У 1950-их роках було опубліковано статтю, в якій пояснювались основні принципи розробки шахових програм. Це був перший реальний крок до створення комп'ютерних шахів.

У двадцятому столітті був розроблений алгоритм гри в шахи, який не був реалізований на комп'ютері через обмеження тогочасної техніки. Однак, ця робота заклала фундамент для майбутніх досліджень, що у майбутньому стало у нагоді при створенні першої шахової програми. Вона була створена у 1956 році і повністю працювала на комп'ютері. Ця програма могла розігрувати повноцінні шахові партії, хоча і на дуже обмеженому рівні. Наступні роки принесли багато нових програм, одна з яких стала першою шаховою програмою, що здобула перемогу над людиною в турнірі.

Починаючи з 1970-х по 1980-ті роки розвиток шахових програм отримав значний поштовх завдяки появі потужніших комп'ютерів і нових алгоритмів.

У цей період було створено багато відомих шахових програм, одна з яких у 1977 році перемогла чемпіона США з шахів на бліц-турнірі.

У 1980-х роках з'явилися спеціалізовані шахові комп'ютери, які використовували спеціалізоване апаратне забезпечення для обробки шахових ходів. Ці машини могли розраховувати мільйони ходів за секунду, що значно підвищило рівень гри комп'ютерів.

Кульмінацією розвитку шахових комп'ютерів 20-го століття стала програма Deep Blue, яка у 1997 році вперше в історії перемогла чинного чемпіона світу в матчі з шести партій. Це досягнення стало знаковим моментом в історії штучного інтелекту і викликало широкий резонанс у суспільстві.

Перемога цієї програми мала глибокий вплив на подальший розвиток шахових програм. Після цього шахові програми стали доступними для широкого загалу, і їхній рівень гри продовжував зростати. З'явилися нові потужні програми, які стали незамінними інструментами для аналізу та тренування шахістів по усьому світу.

У 21-му столітті завдяки використанню нейронних мереж та машинного навчання було розвинуто програми та алгоритми шахів. Одним з найбільш відомих проектів став той, який у 2017 році за лічені години навчання здобув перемогу над найкращими шаховими програмами, використовуючи новаторський підхід до самонавчання.

Ця програма не покладалася на традиційні алгоритми перебору варіантів, а використовувала підхід підкріплювального навчання, що дозволило їй навчитися грати на рівні, недосяжному для попередніх програм. Це відкриття стало новою віхою в історії шахів і показало потенціал нейронних мереж у вирішенні складних інтелектуальних задач.

Розвиток комп'ютерних шахів пройшов довгий шлях від механічних автоматів до сучасних нейронних мереж. Цей процес супроводжувався значними досягненнями в галузі штучного інтелекту, які мали широкий вплив

не тільки на шахи, але й на інші сфери життя. Шахові програми стали незамінним інструментом для шахістів і науковців, сприяючи подальшому розвитку теорії ігор і обчислювальних технологій.

1.2 Аналіз інструментів для освоєння шахістів

Новачкам може бути важко розібратись у правилах шахів і усіх особливостях цієї гри, тому для того, щоб полегшити адаптацію новачків потрібно вигадувати, реалізовувати ефективні інструменти для навчання початківців. Таким способом можна переконати невпевнених у собі людей, та допомогти їм зробити перші кроки у дослідженні гри в шахи. Це позитивно вплине на збільшення числа гравців, які під час гри будуть розвивати свої розумові здібності.

Одним з дієвих способів навчання новачків є відеоуроки та посібники присвячена основам гри, але такий спосіб залучити людину до навчання не дуже цікавий та ефективний, скоріш за все через недостатню мотивацію та нудний процес адаптації новачок кине спроби навчитись грати шахи. Також на певних платформах є можливість виконувати завдання та вправи заготовлені заздалегідь, такий спосіб зберігає інтерес, але позбавляє можливості новачка практикуватись у реальних партіях, в мене на меті створити допоміжний інструмент, щоб люди одночасно навчались, відточували свої практичні навички і розвивались.

Саме тому у цій роботі я реалізував власну функцію, яка допоможе новоспеченим гравцям в шахи вивчити необхідну базу для гри в шахи. Створення простої функції для підсвічування доступних фігур які можуть здійснити хід повинно полегшити освоєння правил гри, тобто гравцеві все одно потрібно продумувати кожен крок заздалегідь, водночас він буде бачити усі доступні опції для розвитку своїх фігур і самостійно буде приймати рішення щодо здійснення ходу. У разі, якщо гравці з різним рівнем підготовки будуть грати один проти одного ця функція повинна дещо нівелювати перевагу досвідченого гравця.

Також ця функція допоможе новачку підвищити бачення грального дошки, він сам не помітить як наскільки швидко нагальність у цій функції відпаде і він не буде потребувати жодних підказок ані зі сторони додатку чи платформи, ані зів сторони інших гравців.

Також завдяки цьому середній рівень гравців у онлайн-платформі буде підвищуватись з кожним днем. Це стане непоганим інструментом для популяризації додатка в майбутньому. Одним з недоліків такої функції можна вважати неможливість пояснити важливість усіх шахових термінів, наприклад, дебют, середина гри, ендшпіль, також відсутність пояснення таких правил шахів як взяття на проході чи рокірування. Тому, підсумовуючи — не існує ідеального інструменту для навчання новеньких гравців. Існує безліч різних підходів для тренування та відточення своїх навичок, але створити ідеальний допоміжний інструмент неможливо через занадто велику кількість особливостей кожної людини, комусь цікаво читати довідники з шах, а хтось бажає вдосконалювати навички шляхом практикування. Єдиний спосіб створити багато різних підходів для підготовки нових гравців.

1.3 Порівняльна характеристика онлайн-платформ для гри в шахи

Порівняння різних онлайн-платформ для гри в шахи є важливим з кількох причин. Кожна платформа пропонує унікальний набір функцій, які можуть значно впливати на досвід гри. Наприклад, деякі платформи можуть мати інтуїтивно зрозумілий інтерфейс, тоді як інші можуть пропонувати більш складні налаштування для досвідчених гравців. Також важливим аспектом є якість супровідних матеріалів, таких як навчальні відео, аналітика партій та доступ до тренувальних програм.

Я вирішив проаналізувати три платформи для гри в шахи: Lichess, Chess.com, Playchess. Мій вибір був зумовлений їхньою популярністю серед шахістів усього світу. На цих платформах багато активних користувачів, що свідчить про високу якість і надійність. Крім того, вони широко згадуються в шаховому співтоваристві і в професійних оглядах, підтверджуючи їх

авторитет і важливість в шаховому світі. Завдяки своїй популярності ці платформи пропонують широкий спектр функцій і можливостей для гравців різного рівня, що робить їх ідеальними для поглибленого аналізу.

Також, різні платформи мають різні рівні активності користувачів і спільноти. Для гравців, які бажають постійно знаходити партнерів для гри або брати участь у турнірах, важливо обрати платформу з великою та активною спільнотою. Крім того, деякі платформи можуть мати кращу систему рейтингів та зручніший підбір суперників за рівнем майстерності.

Третім аспектом є технічні характеристики платформи, такі як швидкість завантаження, стабільність роботи, наявність мобільних додатків та підтримка різних операційних систем. Гравці, які цінують зручність і доступність, будуть схильні обирати платформи, які дозволяють грати на різних пристроях без втрати якості.

Не менш важливим фактором є рівень захисту від шахрайства та наявність модерації. Деякі платформи мають потужні алгоритми для виявлення нечесної гри, що забезпечує чесність змагань і зберігає інтерес до гри.

У період війни саме завдяки онлайн-платформам користувачі мають змогу грати шахи з друзями, які знаходяться в інших містах або за кордоном. В мене на меті є дослідити популярні онлайн-платформи і скласти порівняльну табличку, завдяки якій визначити найбільш зручну платформу для навчання гри в шахи.

Платформа “Lichess”[11] — є безкоштовною платформою для гри в шахи, у якій реалізовано різні режими гри, до прикладу класичні шахи або ж шахи, наявна функція аналізу зіграних партій також на сайті присутні інтерактивні уроки. Є можливість грати шахи проти бота.

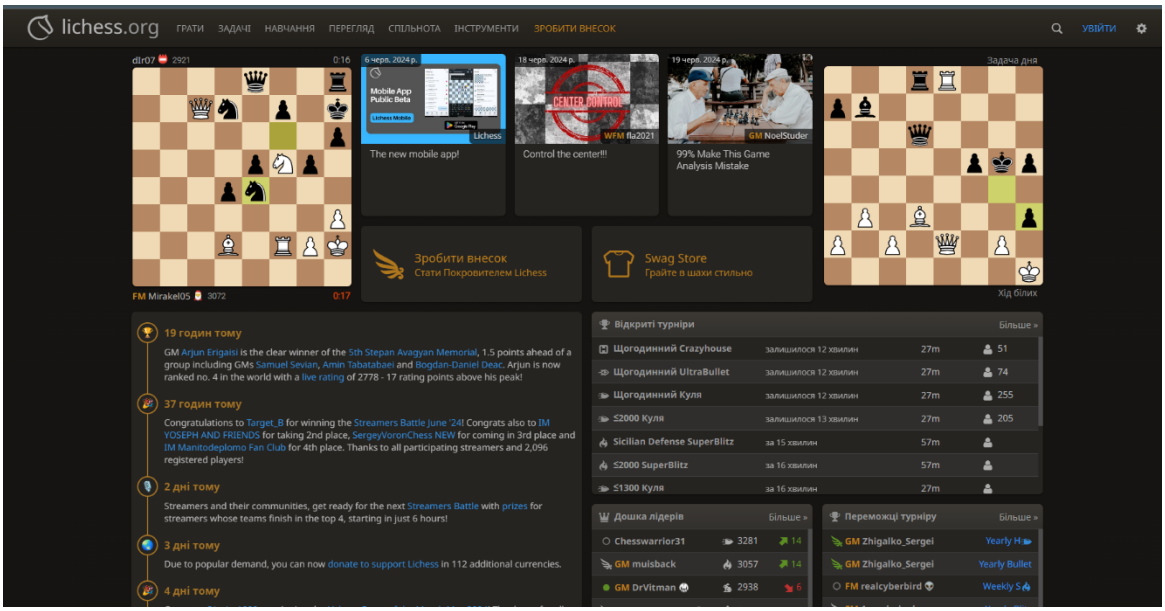

Рис 1.1 платформа “Lichess”

Також однією з найбільш популярних платформ для гри в шахи є “Chess.com”[10] на цій платформі також присутня функція аналізу партій, на сайті містяться навчальні матеріали від посилань на навчальні статті до цілих відеоуроків. Найбільш привабливим у цьому сайті є те, що він організовує онлайн-турніри різних ігрових форматів, кожен гравець може брати участь у змаганнях з призами, особливістю є групи та клуби де користувачі можуть ділитись враженнями щодо партій, обговорювати їх, а також давати поради один одному.

Рис 1.2 Платформа “Chess.com”

Ще одним об’єктом мого аналізу стала платформа “Playchess”[12] цей сервіс існує ще з 1999 року, незважаючи на досить поважний вік сайт залишається досить актуальною платформою, яка забезпечує турнірні змагання, можливість живого спілкування з іншими шахістами на форумах і також приємно дивує кількістю доступних форматів для гри в шахи

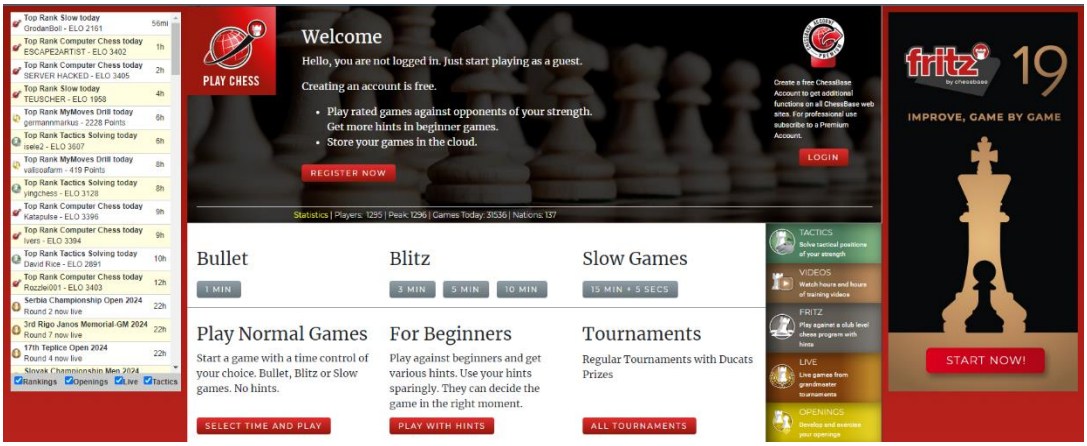

Рис 1.3 Платформа “Playchess”

Усі ці сервіси мають спільні риси і відмінності тому для зручності аналізу усієї інформації я створив порівняльну таблицю де зіставляються усі функції платформ, їх наявність чи відсутність та найголовніше доступність.

Функція / Платформа	Chess.com[10]	Lichess[11]	Playchess[12]
Доступність	Вебсайт, мобільні додатки на IOS, Android	Вебсайт, мобільні додатки на IOS, Android	Вебсайт, мобільний додаток наAndroid
Цінова політика плафторм	Безкоштовно, але присутня реклама, наявна преміум-підписка	Повністю безкоштовно, реклама відсутня	Безкоштовна, преміум підписка дає доступ до додаткових можливостей
Рангові системи	Elo-рейтинг	Elo-рейтинг	Elo-рейтинг
Турніри	Турніри для усіх охочих	Турніри для усіх охочих	Професійні турніри

Навчальні матеріали	Відеоуроки, інтерактивні уроки	Відеоуроки, інтерактивні уроки	Відеоуроки, статті
Соціальні функції	Чати, форуми, блоги	Чати, форуми	Чати, форуми, блоги
Особливості	Головоломки	Відкритий код і повна безкоштовність	Інтеграція з ChessBase

Таблиця 1.2 Порівняльна таблиця сервісів для гри в шахи

Отже, порівняння різних онлайн-платформ для гри в шахи дозволяє знайти оптимальний варіант, який найкраще відповідає потребам та очікуванням гравців. Це сприяє покращенню загального досвіду гри та допомагає уникнути розчарувань, пов'язаних з технічними проблемами або невідповідністю очікувань.

На мою думку найкраща на сьогодні онлайн-платформа для гри в шахи це “Lichess”[11], вона має надзвичайно великий функціонал в тому числі і для новачків, приємний інтерфейс і найбільше приваблює його повна безкоштовність, тобто доступність для усіх бажаючих.

Висновок до розділу 1

Цей розділі присвячений дослідженню історії комп'ютерних ігор, зокрема шахів, тут було розглянуто еволюцію шахових програм від їхніх перших зразків до сучасних високотехнологічних платформ. Детальний аналіз інструментів для освоєння шахів показав, як сучасні технології сприяють підвищенню рівня гри користувачів. Порівняльна характеристика онлайн-платформ для гри в шахи, дозволила виявити їхні сильні та слабкі сторони, а також визначити найбільш підходящі варіанти для різних категорій гравців. Розвиток комп'ютерних технологій значно вплинув на популяризацію шахів і надав гравцям безпрецедентні можливості для навчання, гри та змагань на найвищому рівні.

РОЗДІЛ 2. РОЗРОБКА ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ГРИ В ШАХИ

2.1 Створення бібліотеки з алгоритмами гри в шахи

Під час вибору мови програмування для створення проекту я обрав C#, зокрема цей вибір був спричинений тим, що ця мова інтегрується з .NET, що надає доступ до великої кількості бібліотек, зокрема до ASP.NET — це знадобиться для розробки мого проекту. Крім того, для налаштування візуалу моїх шахів я буду використовувати “Unity”[6], а C# це одна з основних мов програмування для цього рушія. А об’єктно-орієнтований підхід надасть мені можливість створювати класи, необхідні мені для написання логіки гри, методи дозволять створити рух фігур і багато іншого. Підсумовуючи вибір C# та об’єктно-орієнтованого підходу дасть платформі зрозумілу, чітку структуру і містку архітектуру, яка буде доступна для розширення у майбутньому.

Спершу було створено незмінний клас Chess для спрощення операцій у подальшій роботі з створення онлайн шах, тому коли користувачі будуть здійснювати хід клас Chess не буде трансформуватися. Для реалізації ходу я написав окремий клас Move у якому за допомогою різних методів буде здійснювати хід фігур.

Після цього написано функцію GetFigureAt, у якому створюється об’єкт Square і створено отримання фігури на вказаному квадраті і врешті-решт повертає результат, тобто є перевірка щодо відстутності фігури у вказаному квадраті, якщо фігура присутня на квадраті, то змінна приводиться у тип char і повертає потрібний символ. Повна реалізація виглядає ось так:

```
        public char GetFigureAt (string xy)
{
    Square square = new Square(xy);
    Figure f = board.GetFigureAt(square);
    return f == Figure.none ? '.' : (char)f;
}
```

Для того щоб було можливо мвідслідковувати коректність написання бібліотеки створено консольний проект, у який додав посилання на бібліотеку.

Далі створено набір іменованих констант, у якому представлені різні типи шахових фігур: відсутність фігури, король, ферзь, тура, слон, кінь та пішак. Для того, щоб система розрізняла фігури я розробив ось такий список:

```
enum Figure
{
    none,

    whiteKing = 'K',
    whiteQueen = 'Q',
    whiteRook = 'R',
    whiteBishop = 'B',
    whiteKnight = 'N',
    whitePawn = 'P',

    blackKing = 'k',
    blackQueen = 'q',
    blackRook = 'r',
    blackBishop = 'b',
    blackKnight = 'n',
    blackPawn = 'p'
}
```

Він дозволяє розрізняти шахові фігури за допомогою їхніх символів, фігури білого кольору були представленні мною великими літерами, а чорного — маленькими.

Для втілення мого задуму потрібен ще один список, у ньому містяться кольори.

```
enum Color
{
    none,
    white,
    black
```

```
}
```

Також тут зроблено метод FlipColor, у якому перевіряється значення кольору якщо воно дорівнює чорному кольору то повертає білий і навпаки, у випадку, коли колір не є чорним і білим повертається none. Такий спосіб написання дає можливість додавати нові методи і розширення для типу колір. Повний код:

```
static class ColorMethods
 {
     public static Color FlipColor (this Color color)
     {
         if (color == Color.black) return Color.white;
         if (color == Color.white) return Color.black;
         return Color.none;
     }
 }
```

Наступним етапом розробки є структура Square — це комірка шахової дошки, яка використовує координати x та y, статичне поле none дасть мені можливість представити порожню комірку, це стане мені у нагоді для перевірок, а властивості присвоєні координатам забезпечуватимуть контроль доступу до координат, дозволяючи зміну через конструктор і методи всередині класу:

```
public static Square none = new Square(-1, -1);

 public int x { get; private set; }
 public int y { get; private set; }

 public Square (int x, int y)
 {
     this.x = x;
     this.y = y;
 }
```

Після виконання цих дій розпочато створення методу, який перевіряє чи знаходиться ця клітинка всередині дошки цей метод буде використовуватись досить часто під час створення мого проекту.

Реалізація:

```
public bool OnBoard ()
{
    return x >= 0 && x < 8 &&
           y >= 0 && y < 8;
}
```

Ще одним класом, який буде необхідний для створення є FigureOnSquare. Конструктор класу приймає фігуру та квадрат на якому знаходиться фігура, далі значення параметра figure прирівнюється властивості figure і це дозволяє зберегти цю фігуру, що знаходиться на квадраті, такий самий принцип застосовується до параметра square.

Сутність цього класу це структура даних, яка використовується для збереження інформації про фігуру та її місцезнаходження на гральній дошці:

```
internal class FigureOnSquare
{
    public Figure figure { get; private set; }
    public Square square { get; private set; }

    public FigureOnSquare (Figure figure, Square square)
    {
        this.figure = figure;
        this.square = square;
    }
```

Ще одним елементом моєї програми є FigureMoving, він відображає рух фігури на шаховій дошці у ньому написані два конструктори, один з яких приймає:

1. Фігуру на початковому квадраті (fs)
2. Квадрат, на який потрібно перемістити фігуру (to)

3. Параметр, який показує на яку фігуру змінюється пішак, який дістався до кінця дошки

Інший конструктор представляє хід у форматі гри в шахи, а також встановлює відповідні властивості об’єктів

Демонстрація цього елементу:

```
public Figure figure { get; private set; }
public Square from { get; private set; }
public Square to { get; private set; }
public Figure promotion { get; private set; }

public FigureMoving (FigureOnSquare fs, Square to, Figure promotion = Figure.none)
{
    this.figure = fs.figure;
    this.from = fs.square;
    this.to = to;
    this.promotion = promotion;
}

public FigureMoving (string move)
{
    this.figure = (Figure)move[0];
    this.from = new Square(move.Substring(1, 2));
    this.to = new Square(move.Substring(3, 2));
    this.promotion = (move.Length == 6) ? (Figure)move[5] : Figure.none;
}
```

Клас Board є відповідальним за представлення та обробку сиуації шахової поверхні у грі, він має поля: рядок, який зберігає розташування фігур на дошці, масив фігур, що зберігає фігури на кожній комірці дошки, колір фігур, яким зараз належить право здійснити хід та номер ходу у триваючій грі. Інціалізацію об’єкту дошки виконує рядок зі розташуванням фігур на ігровій поверхні.

У цьому класі закладені усі базові функції для шахової логіки, сюди входить: визначення ходів, які доступні для здійснення, перевірка, яка визначає чи здійснено шах та інші. Цей клас надзвичайно масивний, тому був поміщений у додаток (див. Додаток А)

І останньою частинкою моєї бібліотеки став клас Moves, його обов'язок ревізувати можливість руху фігур на ігровій дошці згідно з дотриманням правил гри в шахи.

Цей клас визначає чи можливий хід, який намагається здійснити гравець відповідно до поточного стану дошки. Кожен метод у класі здійснює перевірку для певного типу фігур, це забезпечує коректність руху кожної фігури на шахові дошці. Завдяки такій реалізації в подальшому можна вдосконалювати код і розширювати загальний функціонал для усіх фігур.

Підсумовуючи хочу представити схему класів для реалізації функціоналу онлайн-платформи для гри в шахи.

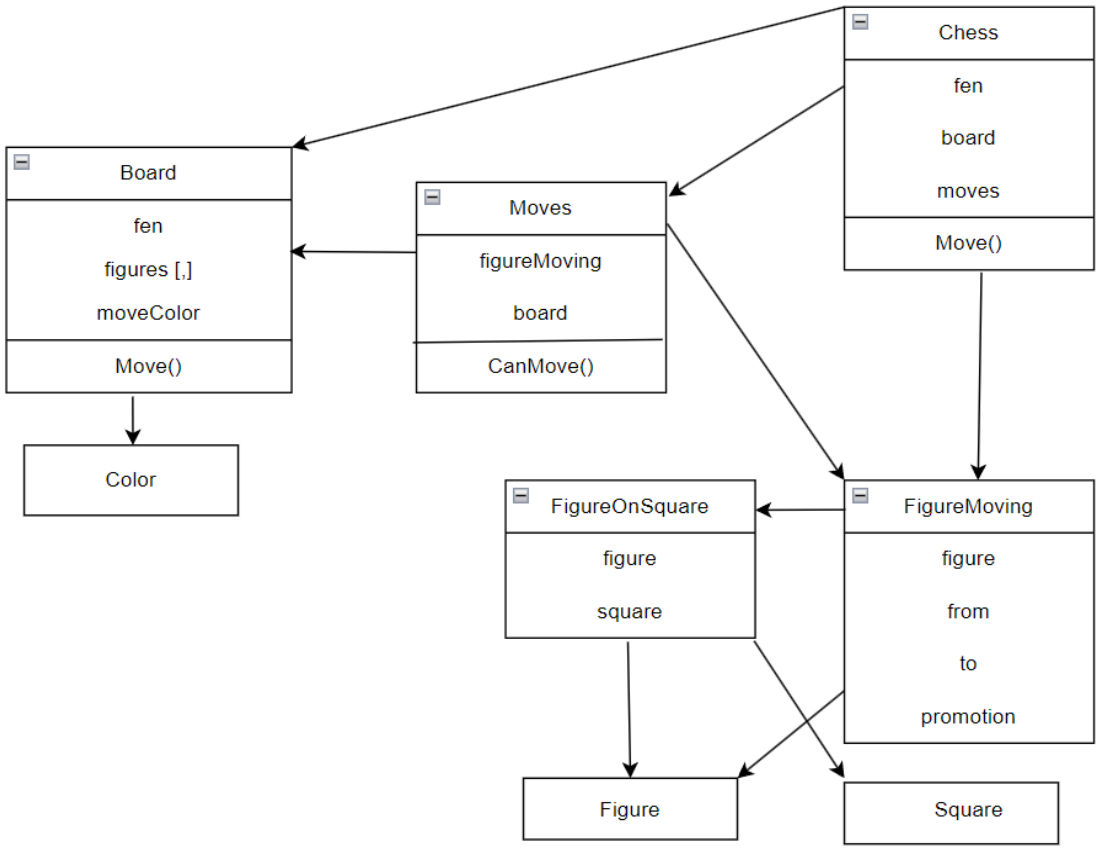

Рис 2.1 Схема класів для реалізації функціоналу

2.2 Створення шахової дошки та фігур засобами “Unity”[5]

Початком моєї роботи у середовищі став етап планування алгоритму для реалізації програми. Після роздумів було розроблено такий план:

1. Створення функції для промальовування шахів
2. Створення методу для розміщення фігур на гральній дошці
3. Створення грального дошки інструментами Unity
4. Кодова реалізація розміщення фігур на дошці
5. Реалізація перетягування фігур
6. Створення функції для підсвічування фігур під час ходу

Отже шлях розпочинався з створення коду для функції завдяки якій користувач зможе перетягувати шахові фігури на дошці.

Цей клас матиме назву DragandDrop саме він визначатиме механізм перетягування та “кидання” фігур в графічному інтерфейсі. Отже цей клас складається з полів класу, які відповідають за поточний стан претягування, інформацію про ігрову фігуру, яка на даний момент перетягується і зміщення між положенням маніпуляратора миші та центром об’єкту, тобто ігрової фігури.

Метод Action виконує алгоритм перетягування і кидання залежно від стану фігури. Інформація щодо приватних методів і їх функцій:

1. PickUp — захоплює об’єкт, після чого встановлює стан перетягування і зберігає зміщення миші та об’єкта.
2. Drag — відповідає за переміщення фігури маніпулятором миша
3. Drop — завершує перетягування і встановлює стан none на позиції звідки розпочиналось перетягування і зберігає позицію на дошці куди був скинутий об’єкт.

Тут можна побачити код завдяки виконанню якого можливі функції описані раніше:

```
class DragandDrop
{
```

```
enum State
{
    none,
    drag
}

public Vector2 pickPosition { get; private set; }
public Vector2 dropPosition { get; private set; }

State state;
GameObject item;
Vector2 offset;

public DragandDrop()
{
    state = State.none;
    item = null;
}

public bool Action()
{
    switch (state)
    {
        case State.none:
            if (IsMouseButtonPressed())
                PickUp();
            break;
        case State.drag:
            if (IsMouseButtonPressed())
                Drag();
            else
            {
                Drop();
                return true;
            }
```

```
                break;
            }
            return false;
        }

}
```

Саме завдяки цьому на ранньому етапі розробки вже є можливість зіграти шахи:

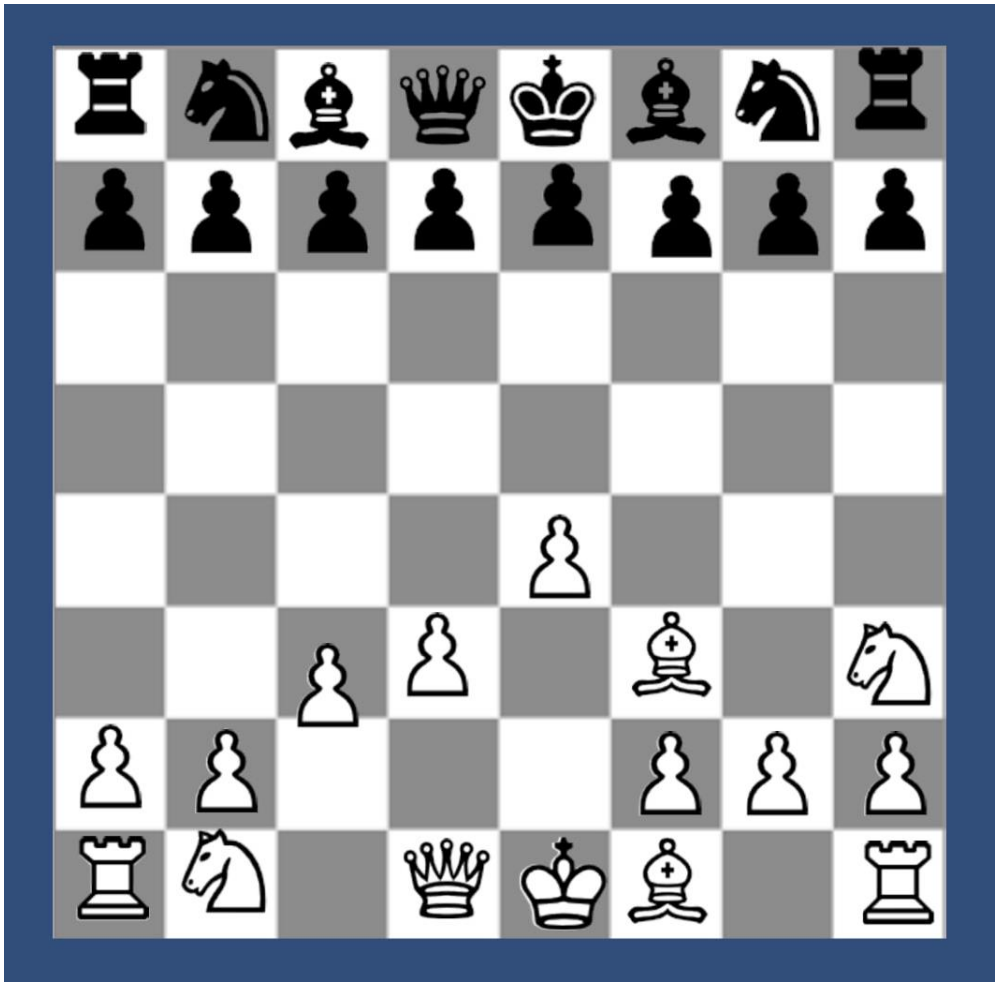

Рис 2.2 Тестування функції DragandDrop

Далі додано у цей проект dll бібліотеку з написаними алгоритмами і правилами гри в шахи та приступив до написання функції, що буде забезпечувати достовірне розміщення фігури на своїй позиції. Принцип роботи цієї функції полягає у переборі усіх клітинок дошки для цього виконується цикл, він перебирає рядки та стовпці дошки. Якщо на клітинці є об'єкт, то викликається метод PlaceFigure, саме він розміщує фігуру на намальованій дошці згідно з позицією. Щоб отримати фігуру яка існує на поточній клітинці викликається метод GetFigureAt, він отримує фігуру яка стоїть на позиції з координатами x, y.

Кодова реалізація методу:

```
void ShowFigures()
 {
     int nr = 0;
     for (int y = 0; y < 8; y++)
         for (int x = 0; x < 8; x++)
         {
             string figure = chess.GetFigureAt(x, y).ToString();
             if (figure == ".") continue;
             PlaceFigure("box" + nr, figure, x, y);
             nr++;
         }
     for (; nr < 32; nr++)
         PlaceFigure("box" + nr, "q", 9, 9);
       }
```

Метод PlaceFigure. Суть метода полягає у динамічному розміщенні шахових фігур на дошці, змінюючи їхні спрайти відповідно до параметрів. Він знаходить об'єкти, які розміщені на сцені і переміщує їх на задану позицію завдяки спрайтам це і дає можливість правильно відобразити фігури у інтерфейсі. Повний код цього методу:

```
void PlaceFigure(string box, string figure, int x, int y)
{
    GameObject goBox = GameObject.Find(box);
```

```
    GameObject goFigure = GameObject.Find(figure);
    GameObject goSquare = GameObject.Find("" + y + x);

    var spriteFigure = goFigure.GetComponent<SpriteRenderer>();
    var spriteBox = goBox.GetComponent<SpriteRenderer>();
    spriteBox.sprite = spriteFigure.sprite;

    goBox.transform.position = goSquare.transform.position;
}
```

Нарешті прийшов час для створення підсвітки під час ходу гравця у цій частині спочатку знаходиться клітинка на дошці і визначається її колір після цього обирається відповідний спрайт і власне піля отримання компонентів спрайту реалізовується заміна спрайту. Ця функція повинна допомогти новачкам під час їхнього навчання гри в шахи.

```
void MarkSquare(int x, int y, bool isMarked)
 {
     GameObject goSquare = GameObject.Find("" + y + x);
     GameObject goCell;
     string color = (x + y) % 2 == 0 ? "Black" : "White";
     if (isMarked)
         goCell = GameObject.Find(color + "SquareMarked");
     else
         goCell = GameObject.Find(color + "Square");
     var spriteSquare = goSquare.GetComponent<SpriteRenderer>();
     var spriteCell = goCell.GetComponent<SpriteRenderer>();
     spriteSquare.sprite = spriteCell.sprite;
 }
```

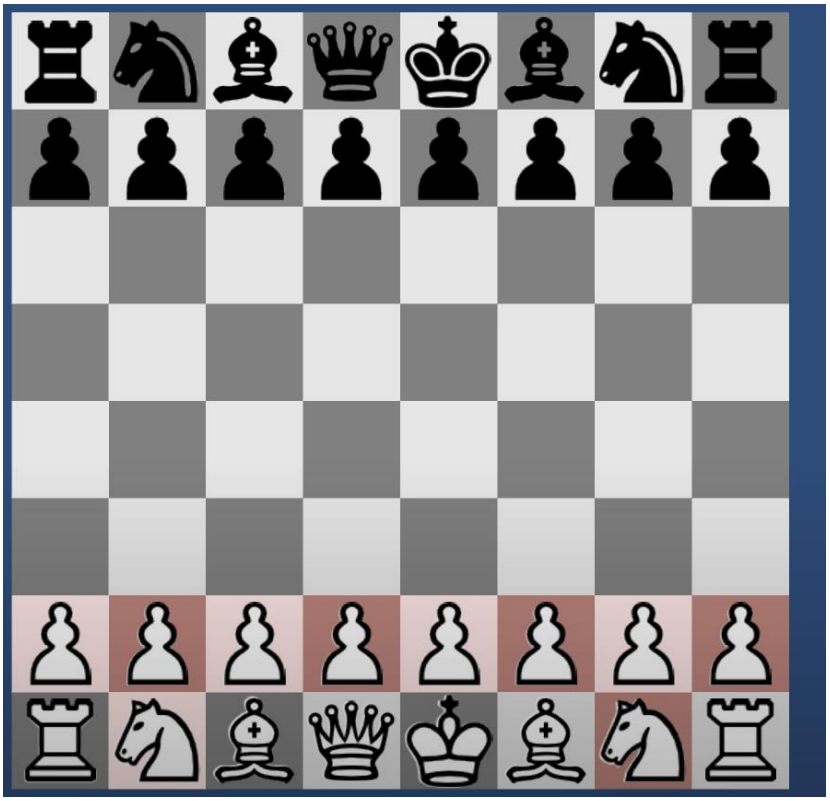

Рис 2.3 Робота функції “Допомога новачку”

Також було прийняте рішення додати кнопку, яка надасть користувачам можливість запускати прямі трансляції своїх партій на стримінговій платформі “Twitch” і власне написав скрипт завдяки якому ця кнопка працює. Код реалізації поміщено у додатках (див. Додаток Б).

2.3 Створення бази даних засобами “SQL Server Management Studio”[15] та створення клієнт-сервера

Створюючи базу даних для моєї онлайн-платформи ухвалено рішення використовувати “SQL Server Management Studio”[15], саме тут задовольняються усі мої потреби як розробника, завдяки своєму зручному і водночас простому інтерфейсу, у ньому міститься увесь необхідний функціонал для створення мого проекту. Ця програма створює усі умови, задля створення надійної і продуктивної бази даних.

Першим кроком стало проектування бази даних і тут прийнято рішення спершу спроектувати на аркуші паперу базу даних після чого намалювати її на ПК

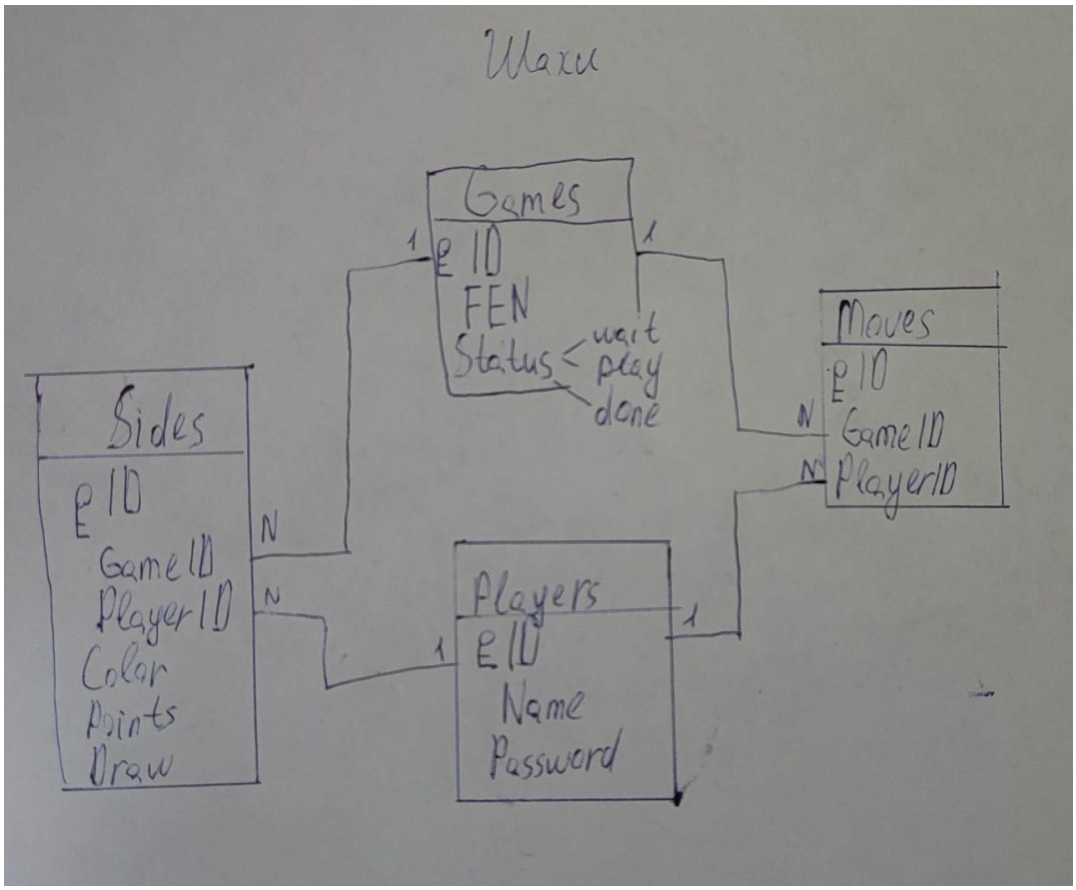

Рис 2.4 Планування бази даних на папері

Після отриманого результату перенесено ці таблиці у цифровий формат для зручності і покращення наглядності.

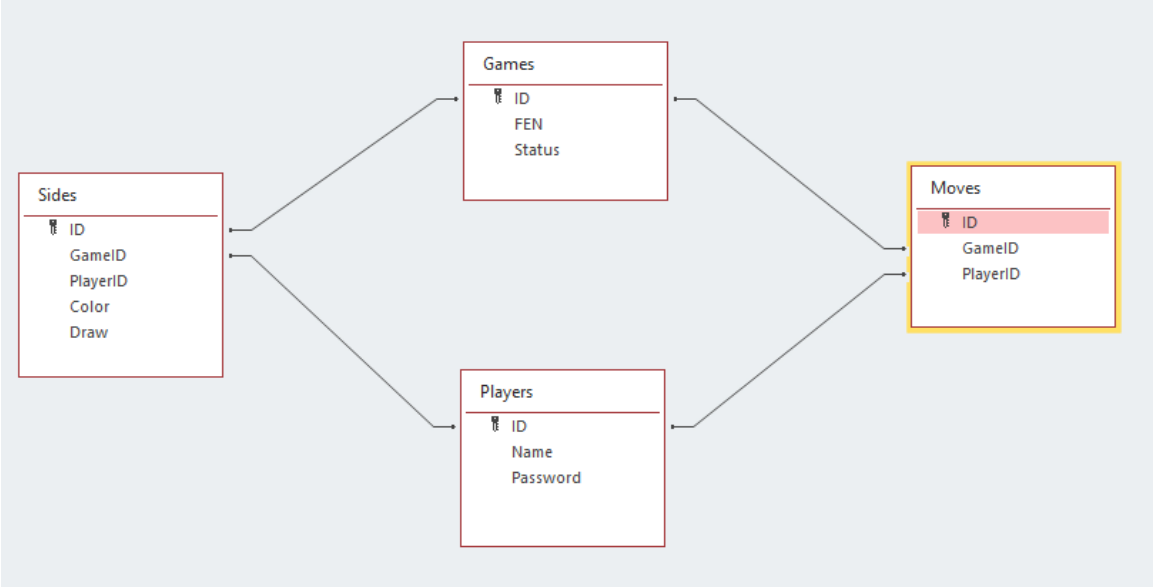

Рис 2.5 Готова схема бази даних

Наступний етап робота з “SQL Server Management Studio”[13].Моїм завданням було створення ось цих чотирьох таблиць засобами цієї програми:

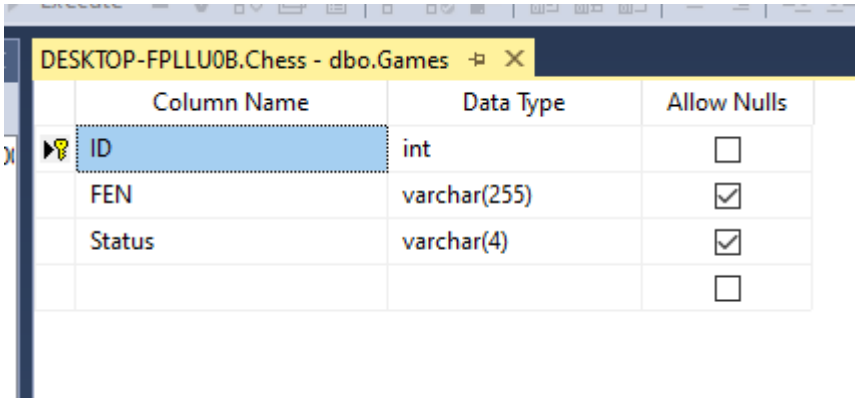

Рис 2.6 Створення таблиць бази даних

Після створення таблиць розпочато створення ASP.NET Web API застосунку. Першим кроком стало створення і налаштування контролера, який буде повідомлювати користувачів, яка версія програми зараз доступна. У контролері наявний внутрішній клас APIVersion, основним завданням якого і є представити модель даних для версії. Тут міститься лише два поля ім’я та версія. Метод GetVersion створює екземпляр класу і повертає його як відповідь на HTTP-запит. Request.CreateResponse створює відповідь з кодом стану і вмістом, який відповідає екземплярам класу.

Кодова реалізація:

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Web.Http;

namespace Chess2API.Controllers
{
    public class VersionController : ApiController
    {
        public class APIVersion
        {
            public string name = "ChessAPI";
            public string version = "0.2";
```

```
        }

        public HttpResponseMessage GetVersion()
        {
            APIVersion version = new APIVersion();
            return Request.CreateResponse(System.Net.HttpStatusCode.OK, version);
        }
    }
}
```

Далі реалізовано підключення проекту до бази даних шляхом додавання ADO.NET Entity Data Model.

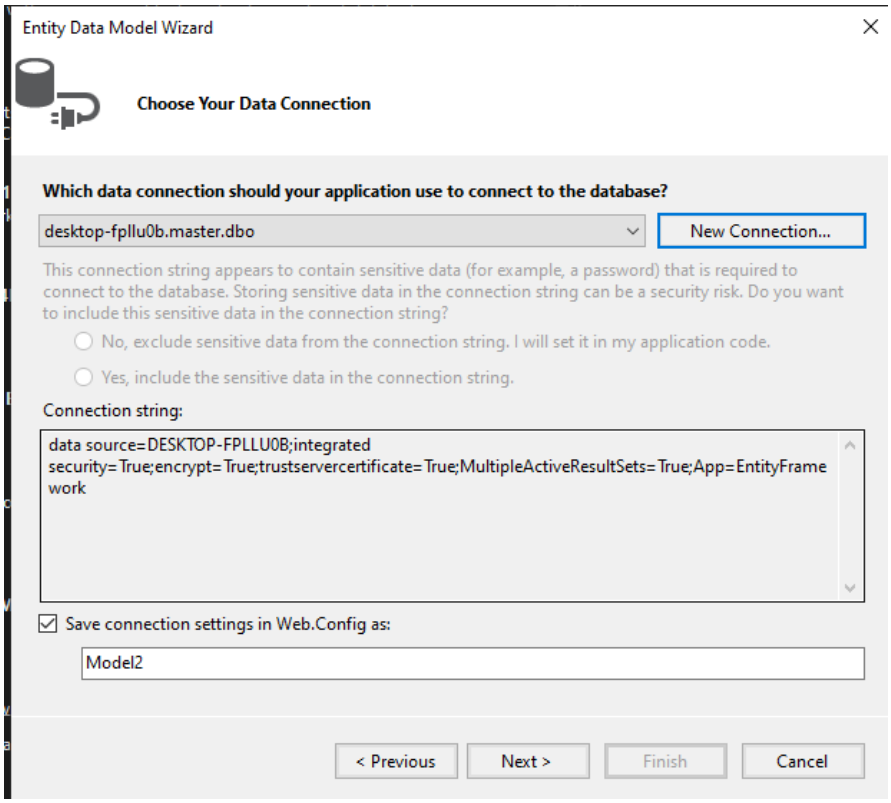

Рис 2.7 Підключення бази даних

На цьому етапі створено модель Games, яка представляє дані про шахову гру в базі даних, ця модель використовує Entity Framework, що дає змогу взаємодіяти з базою даних у вигляді об’єктів C#

Наступний елемент розробки це контролер для роботи з шаховими партіями він надає можливість отримувати інформацію про гру, яка зараз розігрується, для цієї мети метод GetGames обробляє запит і повертає поточну

гру завдяки об’єкту Logic. Ще одним завданням цього контролера є отримання партії за індентифікатором, тобто id для цього реалізовано метод подібний до попереднього тільки навідміну від примітивного методу він отримує інформацію про конкретну гру за ідентифікатором. І нарешті здійснення ходу у партії метод приймає два параметри id та рядок який описує хід.

Детальний код:

```
public class GamesController : ApiController
  {
      private Model1 db = new Model1();

      public Games GetGames()
      {
          Logic logic = new Logic();
          Games game = logic.GetCurrentGame();
          return game;
      }

      public Games GetGames(int id)
      {
          Logic logic = new Logic();
          Games game = logic.GetGame(id);
          return game;
      }

      public Games GetMove(int id, string move)
      {
          Logic logic = new Logic();
          Games game = logic.MakeMove(id, move);
          return game;
      }



      protected override void Dispose(bool disposing)
```

```
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }

        private bool GamesExists(int id)
        {
            return db.Games.Count(e => e.ID == id) > 0;
        }
    }
}
```

Створено клас Logic, який включає в себе декілька дуже важливих методів. Отож основним завданням методу з назвою CreateNewGame створення нової гри. Спершу він ініціалізує новий об’єкт ігри та встановлює початкову позицію гри у форматі фен та додає гру до бази даних. По завершенню кроків зміни зберігаються у базі даних і повертається нова гра.

Ще одним методом написаним тут є MakeMove він приймає номер гри та здійснений хід у формі рядка і виконує такі кроки:

1. Отримання гри за вказаним номером.
2. Якщо такої гри не було знайдено або її статус не є “play”, то метод повертає цю саму гру без змін.
3. Відбувається ініціалізація об’єкта шахи з поточним станом гри (FEN)
4. Здійснюється хід у грі
5. Оновлюється стан гри значенням FEN
6. Якщо після здійснення ходу король суперника під шахом або є патова ситуація статус гри змінюється на завершено
7. Збереження змін у базі даних.

Повний код наведено у додатку В (див. додаток В)

Спочатку було ухвалено рішення про створення шахового клієнта. З цією ціллю я створив бібліотеку яка і стане у нагоді для реалізації мого задуму.

Перед описом реалізації потрібно описати явище парсингу. Парсинг це процес, під час якого аналізуються вхідні рядки заради розбору структури відповідно до певної граматики. У класі ChessClient я створив метод GetCurrentGame він викликає сервер для отримання поточної гри, потім парсить отриманий json-об’єкт та реалізовує створення GameInfo з отриманими даними і зберігає номер поточної гри у спеціальній властивості.

```
public GameInfo GetCurrentGame()
{
    try
    {
        string json = CallServer();
        NameValueCollection gameData = ParseJson(json);
        GameInfo game = new GameInfo(gameData);
        CurrentGameID = int.Parse(game.GameId);
        return game;
    }
    catch (Exception ex)
    {
        throw new ApplicationException("Не вдалося отримати поточну гру", ex);
    }
}
```

Далі тут наявний метод SendMove, основним завданням якого є відправка ходу на сервер для обробки і повертання об’єкту у якому містяться дані про гру.

```
public GameInfo SendMove(string move)
 {
     try
     {
         string json = CallServer($"{CurrentGameID}/{move}");
         NameValueCollection gameData = ParseJson(json);
         return new GameInfo(gameData);
```

```
        }
        catch (Exception ex)
        {
            throw new ApplicationException("Не вдалося відправити хід", ex);
        }
    }
```

Метод CallServer викликає сервер зі вказаним параметром і повертає відповідь сервера у вигляді рядка json. У свою чергу метод ParseJson розбирає відповідний рядок і створює колекцію пар ключ-значення і використовує регулярні вирази для вилучення даних з json.

```
public string CallServer(string parameter = "")
{
    try
    {
        WebRequest request = WebRequest.Create($"{Host}/{User}/{parameter}");
        using (WebResponse response = request.GetResponse())
        using (Stream stream = response.GetResponseStream())
        using (StreamReader reader = new StreamReader(stream))
        {
            return reader.ReadToEnd();
        }
    }
    catch (Exception ex)
    {
        throw new ApplicationException("Помилка при зверненні до сервера", ex);
    }
}
```

У класі ж GameInfo представляється інформація про гру отриману від сервера. Конструктор приймає колекцію пар ключ-значень з даними про партію та ініціалізує властивості об’єкта на основі отриманих даних.

```
{
    public string GameId { get; private set; }
    public string Status { get; private set; }
```

```
    public GameInfo(NameValueCollection gameData)
    {
        GameId = gameData["gameId"];
        Status = gameData["status"];
    }
}
```

І врешті-решт розпочато написання коду, який буде взаємодіяти з сервером шахів через мережу у клієнтській додатку. Він включає в себе логіку ініціалізації та взаємодії з графічним інтерфейсом, і отримання та відправлення даних на сервер. Константа HOST визначає адресу сервера з яким клієнт буде взаємодіяти, а константа USER несе у собі ім’я користувача, яке буде задіяне у взаємодії з сервером. І власне конструктор який:

1. Ініціалізує компонент форми, який можна використовувати для віждображення інтерфейсу
2. Створює новий об’єкт класу Шахи, та проходить ініціалізацію константами HOST та USER
3. Викликає метод для запуску панелі
4. Встановлює для змінної wait значення true
5. Створює новий об’єкт шахового класу та надсилає актуальний стан партії, отриманої з сервера
6. Викликає метод Показати позицію, з метою відображення поточного стану партії
7. Отримує новий стан партії від сервера, шляхом відправлення ходу, який передається як аргумент методу SendMove

Підбиваючи підсумки система являє собою серверну частину, що надає можливість для керування шаховими партіями та клієнтської частини, яка взаємодіє з API, вона використовує Entity Framework для забезпечення коректної роботи з створеною мною базою даних. Сервер надає можливість створення нових ігор, отримання поточного стану про гру та оновлення стану після кожного ходу здійсненого користувачем протягом партії. Клієнт

співпрацює з сервером для отримання та відправки даних щодо стану гри. Архітектура забезпечує чітке розділення обов’язків та полегшує підтримку системи.

2.4 Тестування мобільного застосунку гри в шахи

У цьому підрозділі будуть проведені тести, розроблені для перевірки основної функціональної механіки перетягування у моїй онлайн-платформі для гри в шахи на платформі Android. Далі наведено пояснення і реалізація цих тестів, та що вони означають.

1)Підняття, встановлення та розташування елемента: Цей тест має на меті перевірити процес початкового підняття елемента. Він перевіряє, чи скрипт DragAndDrop правильно ідентифікує об'єкт, на який клацнули, і точно позиціонує його на основі позиції дотику:

```
using UnityEngine;
using NUnit.Framework;

public class DragAndDropTests
{
    public void PickUp_SetItemAndPosition_Correctly()
    {
        GameObject itemObject = new GameObject();
        GameObject squareObject = new GameObject();
        itemObject.transform.position = new Vector3(0, 0, 0);
        squareObject.transform.position = new Vector3(1, 1, 0);

        DragAndDrop dragAndDrop = new DragAndDrop();


        Vector2 touchPosition = new Vector2(1.1f, 1.1f
        Camera.main = new Camera();
        RaycastHit2D hit = Physics2D.Raycast(touchPosition, Vector2.zero);
              itemObject.GetComponent<Collider2D>().enabled = true;
```

У цьому тесті перевіряється правильність роботи скрипту DragAndDrop у встановленні поточного піднятого елемента та коректному встановленні позиції підняття. Підготовка полягає в створенні двох GameObject: itemObject, який знаходиться у початковій позиції (0, 0, 0), і squareObject, розміщеному в (1, 1, 0). Далі ініціюється новий екземпляр скрипту DragAndDrop. У дії тесту імітується дотик у позиції (1.1f, 1.1f). За допомогою RaycastHit2D перевіряється, чи перетинає дотик будь-який колайдер. Хоча компонент колайдера для itemObject не активований, скрипт DragAndDrop ідентифікує цей об'єкт як itemObject.

В результаті скрипт DragAndDrop правильно зареєстрував itemObject, як піднятий елемент і встановив внутрішню змінну PickPosition на позицію дотику (1.1f, 1.1f).

2) Перетягування елемента за допомогою дотику.

Цей тест імітує перетягування піднятого елемента. Він перевіряє, чи скрипт оновлює позицію елемента на основі руху дотику користувача:

```
using UnityEngine;
using NUnit.Framework;

public class DragAndDropTests
{

    public void Drag_MoveItemWithTouch_Correctly()
    {

        GameObject itemObject = new GameObject();
        itemObject.transform.position = new Vector3(0, 0, 0);

        DragAndDrop dragAndDrop = new DragAndDrop();
        dragAndDrop.Action();


        Vector2 touchDelta = new Vector2(1.0f, 1.0f);
```

```
        Camera.main = new Camera();
        itemObject.GetComponent<Collider2D>().enabled = true;
```

Даний тест перевіряє правильність переміщення об'єкту під час "перетягування" у скрипті DragAndDrop. Підготовка включає створення itemObject і його розміщення у початковій позиції координат. Також створюється екземпляр скрипту DragAndDrop. Скрипт вже знаходиться у стані "перетягування" з піднятим itemObject,. У дії тесту імітується delta дотику, що відображає рух пальця користувача (1.0f, 1.0f).

Результат: скрипт DragAndDrop коректно оновлює позицію itemObject відповідно до delta дотику. Таким чином, об'єкт переміщається на (1.0f, 1.0f) по осях X і Y.

3) Скидання та випуск елемента. Цей тест фокусується на скиданні перетягнутого елемента. Він гарантує, що скрипт скидає свій внутрішній стан (PickPosition, DropPosition) і встановлює загальний стан перетягування назад у "None" після скидання елемента:

```
using UnityEngine;
using NUnit.Framework;

public class DragAndDropTests
{

    public void Drop_ReleaseItem_Correctly()
    {

        GameObject itemObject = new GameObject();
        itemObject.transform.position = new Vector3(0, 0, 0);

        DragAndDrop dragAndDrop = new DragAndDrop();
        dragAndDrop.Action();
        dragAndDrop.Action();
```

```
        Assert.IsNull(dragAndDrop.PickPosition);
        Assert.IsNull(dragAndDrop.DropPosition);
        Assert.AreEqual(DragAndDrop.State.None, dragAndDrop.GetState());
    }
}
```

У цьому коді спочатку створюється тестовий об'єкт itemObject, який представляє перетягуваний елемент і розміщується у точці (0, 0, 0). Далі ініціалізується екземпляр класу DragAndDrop і викликається метод Action() двічі. Перше викликання Action() симулює початок перетягування елемента, а друге — скидання елемента. Відповідно до очікувань тесту, після скидання перетягуваного елемента всі внутрішні стани класу DragAndDrop були скинуті.

У тесті провіряється, що після скидання:

- PickPosition, тобто позиція, з якої було піднято елемент має бути null.
- DropPosition — позиція, на яку було відпущено елемент, також має бути null.
- Стан перетягування має бути встановлений у None, це означає, що жоден елемент в даний момент не перетягується.

Таким чином, цей тест перевіряє правильність роботи механізму скидання внутрішніх станів після завершення операції перетягування, забезпечуючи коректність функціонування системи у відповідних умовах використання.

Висновок до розділу 2

У розділі, присвяченому розробці мобільного застосунку для гри в шахи, було виконано кілька ключових етапів. Спочатку було створено бібліотеку з алгоритмами гри в шахи, що забезпечує логіку та функціональність гри. Далі, за допомогою інструментів "Unity" було розроблено візуальне представлення шахової дошки та фігур, що забезпечує інтерактивність та зручність використання. Після цього створено базу даних засобами "SQL Server

Management Studio", яка дозволяє зберігати інформацію про ігри, гравців та їхні результати. Наступним кроком було налаштування клієнт-серверної архітектури, що забезпечує стабільну роботу застосунку та можливість гри в реальному часі з іншими користувачами. Завершальним етапом стало тестування мобільного застосунку, яке підтвердило його функціональність та готовність до використання. Таким чином, усі етапи розробки були успішно виконані, що дозволило створити повноцінний ігровий продукт, готовий до впровадження та використання.

ВИСНОВКИ

У процесі дослідження було детально проаналізовано еволюцію комп'ютерних шахів, розроблено алгоритми гри, створено візуалізацію шахової дошки та фігур за допомогою "Unity", а також базу даних засобами "SQL Server Management Studio". Впровадження клієнт-серверної архітектури та тестування мобільного застосунку підтвердили його функціональність та готовність до використання. Ці етапи дозволили створити сучасний ігровий продукт, який поєднує в собі багатий історичний контекст і нові технологічні рішення, забезпечуючи ефективне навчання.

Під час виконання дипломної роботи було:

- проведено аналіз сучасних онлайн-платформ,
- проаналізовано сучасні технології створення комп’ютерних ігор,
- проведено порівняльну характеристику платформ для створення ігор онлайн та
- розроблено шахову онлайн-платформу, яка повинна задовільнити вимоги і потреби новачків у цій захопливій грі.

Основні зусилля були спрямовані на створення інструментів для навчання нових людей і освоєння ними правил гри.

Платформа сприяє поширенню культури гри в шахи завдяки можливості гравців транслювати свої партії н а стримінговій платформі.

Під час створення і написання проекту використано графічний двигун, методи об’єктно-орієнтованого програмування та програми для забезпечення коректної взаємодії програм з базою даних. Це дозволило створити додаток таким, яким він і планувався.

Отримані результати дозволять мені відродити інтерес користувачів до гри в шахи, а також привернути увагу нових людей за допомогою популярної стримінгової платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Database Systems: The Complete Book”. Хектор Гарсія-Моліна, Джеффрі Д. Уллман, Дженніфер Відом. 2008 рік, с. 1248
2. “Designing Data-Intensive Applications”. Мартін Клеппманн. 2017 рік, с. 616
3. “Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition”. Душан Петкович. 2019 рік, с. 848
4. “SQL Server Query Tuning and Optimization”. Бенджамін Неварез. 2022 рік, с. 424
5. “Unity Game Development Cookbook”. Паріс Батфілд-Едісон, Джон Меннінг, Тім Наджент. 2019 рік, с. 408
6. “Unity in Action”. Ніл Акерман, Майкл Андерсон. 2017 рік, 576 с. URL: https://www.manning.com/books/unity-in-action-second-edition
7. “Вивчення C# через розробку ігор на Unity”. Гаррісон Ферроне. 2022 рік, с. 458
8. “Вступ до дизайну ігрових прототипів та розробки”. Джеремі Гібсон Бонд. 2022 рік, с. 1296
9. Онлайн посібник по C# від Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/csharp/
10. Платформа Chess.com. URL: https://www.chess.com/
11. Платформа Lichess. URL: https://lichess.org/
12. Платформа Playchess. URL: https://play.chessbase.com/en
13. Посібник користувача SQL Server Management Studio від Microsoft. URL: https://learn.microsoft.com/uk-ua/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16
14. Посібник користувача Unity. URL: https://docs.unity3d.com/Manual/index.html
15. Стаття про Studio SQL Server Management Studio від Microsoft: https://techcommunity.microsoft.com/t5/sql-server-blog/bg-p/SQLServer

ДОДАТОК А.

Реалізація класу Board

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

namespace Дипломна_шахи_онлайн
{
    internal class Board
    {
        public string fen { get; private set; }
        Figure[,] figures;
        public Color moveColor { get; private set; }
        public int moveNumber { get; private set; }

        public Board (string fen)
        {
            this.fen = fen;
            figures = new Figure[8, 8];
            Init();
        }

        void Init()
        {
            // "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"
            string[] parts = fen.Split();
            if (parts.Length != 6) return;
            InitFigures(parts[0]);
            moveColor = (parts[1] == "b") ? Color.black : Color.white;
            moveNumber = int.Parse(parts[5]);
        }
```

```
        void InitFigures(string data)
        {
            for (int j = 8; j >= 2; j--)
                data = data.Replace(j.ToString(), (j - 1).ToString() + "1");
            data = data.Replace("1", ".");
            string[] lines = data.Split('/');
            for (int y = 7; y >= 0; y--)
                for (int x = 0; x < 8; x++)
                    figures[x, y] = lines[7 - y][x] == '.' ? Figure.none :
                        (Figure)lines[7 - y][x];
        }

        public IEnumerable<FigureOnSquare> YieldFigures()
        {
            foreach (Square square in Square.YieldSquares())
                if (GetFigureAt(square).GetColor() == moveColor)
                    yield return new FigureOnSquare(GetFigureAt(square), square);
        }

        void GenerateFEN ()
        {
            fen = FenFigures() + " " +
                   (moveColor == Color.white ? "w" : "b") +
                " - - 0 " + moveNumber.ToString();
        }

        string FenFigures ()
        {
            StringBuilder sb = new StringBuilder();
            for (int y = 7; y >= 0; y--)
            {
                for (int x = 0; x < 8; x++)
                    sb.Append(figures[x, y] == Figure.none ? '1' : (char)figures[x,
y]);
                if (y > 0)
```

```
                sb.Append('/');
            }
            string eight = "11111111";
            for (int j = 8; j >= 2; j--)
                sb.Replace(eight.Substring(0, j), j.ToString());
            return sb.ToString();
        }


        public Figure GetFigureAt (Square square)
        {
            if (square.OnBoard())
                return figures[square.x, square.y];
            return Figure.none;
        }

        void SetFigureAt (Square square, Figure figure)
        {
            if (square.OnBoard())
                figures[square.x, square.y] = figure;
        }

        public Board Move (FigureMoving fm)
        {

            Board next = new Board(fen);
            next.SetFigureAt(fm.from, Figure.none);
            next.SetFigureAt(fm.to, fm.promotion == Figure.none ? fm.figure :
fm.promotion);
            if (moveColor == Color.black)
                next.moveNumber++;
            next.moveColor = moveColor.FlipColor();
            next.GenerateFEN();
            return next;
        }
```

```
        bool CanEatKing ()
        {
            Square badKing = FindBadKing();
            Moves moves = new Moves(this);
            foreach (FigureOnSquare fs in YieldFigures())
            {
                FigureMoving fm = new FigureMoving(fs, badKing);
                if (moves.CanMove(fm))
                    return true;
            }
            return false;
        }
        private Square FindBadKing()
        {
            Figure badKing = moveColor == Color.black ? Figure.whiteKing :
Figure.blackKing;
            foreach (Square square in Square.YieldSquares())
                if (GetFigureAt(square) == badKing)
                    return square;
            return Square.none;
        }

        public bool IsCheck ()
        {
            Board after = new Board(fen);
            after.moveColor = moveColor.FlipColor();
            return after.CanEatKing();
        }

        public bool IsCheckAfterMove (FigureMoving fm)
        {
            Board after = Move(fm);
            return after.CanEatKing();
```

ДОДАТОК Б.

Реалізація кнопки трансляції гри

```
public class TwitchStream : MonoBehaviour
{
    public Button startStreamButton;
    private static readonly string clientId = "YOUR_CLIENT_ID";
    private static readonly string clientSecret = "YOUR_CLIENT_SECRET";
    private static readonly string redirectUri = "http://localhost:8080";
    private string accessToken;

    private HttpListener httpListener;

    void Start()
    {
        startStreamButton.onClick.AddListener(StartStream);



        httpListener = new HttpListener();
        httpListener.Prefixes.Add(redirectUri + "/");
        httpListener.Start();
        httpListener.BeginGetContext(new AsyncCallback(OnAuthResponse),
httpListener);
    }

    async void StartStream()
    {
        if (string.IsNullOrEmpty(accessToken))
        {
            Application.OpenURL(GetAuthUrl());
        }
        else
        {
            await StartTwitchStream();
        }
    }
```

```
    string GetAuthUrl()
    {
        return
$"https://id.twitch.tv/oauth2/authorize?client_id={clientId}&redirect_uri={redirectU
ri}&response_type=token&scope=channel:manage:broadcast";
    }

    async void OnAuthResponse(IAsyncResult result)
    {
        var context = httpListener.EndGetContext(result);
        var response = context.Response;

        string responseString = "<html><body>Authentication successful. You can
close this tab.</body></html>";
        byte[] buffer = System.Text.Encoding.UTF8.GetBytes(responseString);
        response.ContentLength64 = buffer.Length;
        var output = response.OutputStream;
        output.Write(buffer, 0, buffer.Length);
        output.Close();

        string rawUrl = context.Request.RawUrl;
        accessToken = ExtractAccessToken(rawUrl);

        if (!string.IsNullOrEmpty(accessToken))
        {
            await StartTwitchStream();
        }
    }

    string ExtractAccessToken(string rawUrl)
    {
        var fragments = rawUrl.Split('#');
        if (fragments.Length > 1)
        {
            var parameters = fragments[1].Split('&');
            foreach (var param in parameters)
```

```
            {
                var keyValue = param.Split('=');
                if (keyValue.Length == 2 && keyValue[0] == "access_token")
                {
                    return keyValue[1];
                }
            }
        }
        return null;
    }

    async Task StartTwitchStream()
    {
        using (HttpClient client = new HttpClient())
        {
            client.DefaultRequestHeaders.Add("Client-ID", clientId);
            client.DefaultRequestHeaders.Add("Authorization", "Bearer " +
accessToken);

            var data = new
            {
                title = "Test Stream",
                game_id = "509658",
                broadcaster_language = "en"
            };

            StringContent content = new
StringContent(Newtonsoft.Json.JsonConvert.SerializeObject(data),
System.Text.Encoding.UTF8, "application/json");
            HttpResponseMessage response = await
client.PatchAsync("https://api.twitch.tv/helix/channels", content);

            if (response.IsSuccessStatusCode)
            {
                Debug.Log("Stream started on Twitch!");
            }
            else
```

```
        {
            Debug.LogError("Failed to start stream: " + response.ReasonPhrase);
        }
    }
}

void OnDestroy()
{
    if (httpListener != null && httpListener.IsListening)
    {
        httpListener.Stop();
        httpListener.Close();
    }
}
      }
```

ДОДАТОК В.

Реалізація класу Logic

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Дипломна_шахи_онлайн;
using static Дипломна_шахи_онлайн.Chess;


namespace Chess2API.Models
{
    public class Logic
    {
        private Model1 db;

        public Logic ()
        {
            db = new Model1();
        }
        public Games GetCurrentGame()
        {
            Games game = db
                .Games.Where(g => g.Status == "play")
                .OrderBy(g => g.ID)
                .FirstOrDefault();
            if (game == null)
                game = CreateNewGame();
            return game;
        }

        public Games GetGame(int id)
        {
            return db.Games.Find(id);
```

```
}
private Games CreateNewGame()
{
    Games game = new Games();

    Дипломна_шахи_онлайн.Chess chess = new Chess();
    game.FEN = chess.fen;
    game.Status = "play";

    db.Games.Add(game);
    db.SaveChanges();

    return game;
}

public Games MakeMove(int id, string move)
{
    Games game = GetGame(id);
    if (game == null) return game;

    if (game.Status != "play")
        return game;

    Chess chess = new Chess(game.FEN);
    Chess chessNext = chess.Move(move);

    if (chessNext.fen == game.FEN)
        return game;

    game.FEN = chessNext.fen;
    if (chessNext.IsCheck || chess.IsStalemate)
        game.Status = "done";
    db.Entry(game).State = System.Data.Entity.EntityState.Modified;
    db.SaveChanges();
```