

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ДЕСКТОПНОЇ

КЛІЄНТ-СЕРВЕРНОЇ ГРИ BLACKJACK

DEVELOPMENT AND SOFTWARE IMPLEMENTATION OF THE

BLACKJACK DESKTOP CLIENT-SERVER GAME

Виконав(ла): студент(ка) 4 курсу, групи КНТ-110

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАВЛЕНКО К.Є.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПАВЛЕНКО Кирило Євгенійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка та програмна реалізація десктопної клієнт-серверної гри Blackjack. Development and Software Implementation of the Blackjack Desktop Client-Server Game.

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “24” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат. Перелік скорочень та умовних познач. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Експерименти та результати. Висновки. Перелік джерел посилань. Додатки.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтролер	ДЕЙНЕГА Л.Ю., старший викладач		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4,5,6
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Кирило ПАВЛЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РФЕРАТ

Пояснювальна записка до дипломної роботи: 156 с., 28 табл., 48 рис., 4 дод., 32 джерел.

БЛЕКДЖЕК, КЛІЄНТ-СЕРВЕР, ДЕСКТОП, ГРА, КАРТИ, WINDOWS, C#, WPF, WCF.

Об'єкт проєктування – клієнт-серверний десктопний ігровий застосунок.

Предмет дослідження – методи та технології розробки клієнт-серверних застосунків для реалізації мультиплеєрних ігрових сесій за правилами карткової гри «Блекджек».

Мета роботи – розроблення програмного забезпечення, яке буде реалізувати мультиплеєрні ігрові сесії, за правилами карткової гри «Блекджек».

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Microsoft Visual Studio 2019, підсистеми WPF та WCF.

Результати. Створено програмне рішення, яке надає користувачу зручний інтерфейс та функціонал які реалізують ігровий процес за правилами карткової гри «Блекджек».

Висновки. Розроблене програмне забезпечення дозволяє грати в гру «Блекджек» з іншими гравцями по мережі використовуючи комп'ютер з ОС Windows.

Галузь використання – приватні особи.

ABSTRACT

Explanatory note to the diploma qualifying work of the paper: 156 pages, 48 figures, 28 tables, 4 appendixes, 32 sources.

BLACKJACK, CLIENT-SERVER, DESKTOP, GAME, CARDS, WINDOWS, C#, WPF, WCF.

Research Object – client-server desktop gaming application.

Subject of Research – methods and technologies for developing client-server applications to implement multiplayer gaming sessions according to the rules of the card game «Blackjack».

The aim of the work is development of software that will implement multiplayer gaming sessions according to the rules of the card game «Blackjack».

Materials, methods and technical means: C# programming language, Microsoft Visual Studio 2019 development environment, WPF and WCF subsystems.

The results. A software solution has been created that provides the user with a convenient interface and functionality that implements the game process according to the rules of the card game «Blackjack».

Conclusions. The developed software allows you to play the game «Blackjack» with other players on the network using a computer with Windows OS.

The field of use is private individuals.

ЗМІСТ

	С.
Вступ.....	10
1 Аналіз предметної області та огляд існуючих аналогів	11
1.1 Огляд існуючих методів вирішення завдання	13
1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання	15
1.2.1 Онлайн гра «Blackjackist»	15
1.2.2 Програма «Blackjack Gold»	16
1.2.3 Мобільна гра «Blackjack».....	18
1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів.....	20
1.3 Постановка завдання роботи.....	21
1.4 Висновки за розділом 1.....	21
2 Вибір та обґрунтування структури системи, яка проектується.....	23
2.1 Вибір інструментарію розробки програмного забезпечення	23
2.1.1 Огляд особливостей мови програмування	23
2.1.2 Огляд особливостей обраного середовища розробки	27
2.1.3 Огляд особливостей обраної системи управління базами даних	31
2.2 Структурна схема програми	32
2.3 Опис модулів розроблюваної системи.....	33
2.4 Висновки за розділом 2	34
3 Основні рішення щодо реалізації компонентів системи.....	36
3.1 Алгоритм функціонування розробленої програми.....	36
3.1.1 Алгоритм процесу авторизації	37
3.1.2 Алгоритм процесу реєстрації	38
3.1.3 Алгоритм налаштування лобі	38
3.1.4 Алгоритм створення лобі	39
3.1.5 Алгоритм під'єднання до лобі.....	40
3.1.6 Алгоритм зміна статусу готовності	40

3.1.7 Алгоритм процесу ходу.....	41
3.2 Вибір шаблону проектування	42
3.3 Опис модулів розроблюваної системи.....	44
3.3.1 Клас AboutTheProgramWindow	47
3.3.2 Клас AuthorizationWindow	47
3.3.3 Клас BlackjackClientException	48
3.3.4 Клас CreateLobbyWindow.....	49
3.3.5 Клас EndOfGameWindow	50
3.3.6 Клас GameProcess.....	50
3.3.7 Клас GameRulesWindow	51
3.3.8 Клас GameWindow	52
3.3.9 Клас LobbySearchWindow	53
3.3.10 Клас LobbyWindow	55
3.3.11 Клас MainWindow	56
3.3.12 Клас RegistrationWindow	58
3.3.13 Клас LobbyMember	59
3.3.14 Клас BlackjackException	59
3.3.15 Клас DataBaseAPI.....	60
3.3.16 Клас Deck.....	60
3.3.17 Клас Card.....	61
3.3.18 Клас Game.....	61
3.3.19 Клас Lobby	62
3.3.20 Клас ServerUser	62
3.3.21 Клас ServiceBlackjack	63
3.3.22 Інтерфейс IServiceBlackjack.....	64
3.3.23 Інтерфейс IServiceBlackjackCallback.....	65
3.4 Проектування бази даних	66
3.5 Основні рішення щодо розробки інтерфейсу.....	67
3.6 Висновки за розділом 3	83
4 Керівництво програміста.....	84

4.1 Призначення та умови застосування програми	84
4.1.1 Призначення програми.....	84
4.1.2 Функції програми.....	84
4.1.3 Умови застосування програми для клієнтської та серверної частини	85
4.2 Характеристика програми	85
4.2.1 Структура програми	85
4.3 Звертання до програми	86
4.4 Вхідні та вихідні дані.....	86
4.4.1 Вхідні дані	86
4.4.2 Вихідні дані	87
4.5 Повідомлення	87
5 Керівництво оператора	88
5.1 Призначення програми	88
5.2 Умови виконання програми	88
5.2.1 Апаратні вимоги програми	88
5.2.2 Вимоги до операторів.....	88
5.3 Виконання програми.....	89
5.3.1 Запуск програми.....	89
5.3.2 Виконання роботи з програмою	89
5.4 Повідомлення оператора	95
6 Тестування і результати.....	97
6.1 Опис методики проведення тестування.....	97
6.2 Результати використання розробленого програмного забезпечення	98
Висновки	101
Перелік джерел посилання	102
Додаток А Технічне завдання	105
Додаток Б Опис програми	112
Додаток В Текст програми	118
Додаток Г Слайди презентації	146

ПЕРЕЛІК УМОВНИХ ОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер;
ТЗ	– технічне завдання;
API	– Application Programming Interface.
DRY	– Don't Repeat Yourself
IT	– Information Technology
IDE	– Integrated Development Environment
WCF	– Windows Communication Foundation

ВСТУП

Сьогодні комп'ютер став невід'ємною частиною життя людини. Ми використовуємо його для навчання, роботи, розваг, наукових досліджень.

У сучасному цифровому світі ігрова індустрія набуває все більшої популярності, пропонуючи користувачам різноманітні розважальні варіанти. Одним із найулюбленіших ігрових жанрів є карточна гра «Blackjack».

Актуальність розробки клієнт-серверної десктопної ігрової програми «Blackjack» полягає в тому, що ця гра є однією з найпопулярніших карточних ігор у світі. Створення цієї гри «Blackjack» розширить можливості гравців, дозволяючи їм змагатись з іншими гравцями у режимі реального часу. Це надасть їм можливість відчувати справжню атмосферу гри та взаємодіяти з іншими гравцями, навіть якщо вони знаходяться на значній відстані один від одного.

Завданням дипломної роботи є розробка клієнт-серверної десктопної ігрової програми під назвою «Blackjack», яка має на меті надати користувачам можливість насолоджуватись цією популярною грою у віртуальному середовищі. Клієнт-серверна архітектура обрана для того, щоб забезпечити можливість багатокористувацьких ігор, де гравці з різних куточків світу можуть змагатись один з одним..

У процесі розробки програмного продукту будуть враховані вимоги до функціональності, надійності, ефективності та безпеки. Крім того, буде приділено увагу створенню зручного та інтуїтивно зрозумілого інтерфейсу, що дозволить користувачам легко орієнтуватись у грі та зробити необхідні рішення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

Спочатку був утворений опис предметної області вирішуваної проблеми.

Предметна область: Розробка клієнт-серверної десктопної ігрової програми «Blackjack».

Основна проблема: Гравці бажають мати можливість грати в популярну гру «Blackjack» у віртуальному середовищі, взаємодіючи з іншими гравцями та насолоджуючись реалістичним досвідом гри, незалежно від їх фізичного розташування.

Мета роботи: Розробити клієнт-серверну десктопну ігрову програму «Blackjack», яка надасть користувачам можливість грати в гру у реальному часі, взаємодіючи з іншими гравцями по мережі. Програма повинна мати зручний інтерфейс, надійну архітектуру та відповідати вимогам надійності та ефективності.

Тому було проведено пошук у інтернеті існуючих програмних забезпечень (ПЗ) для гри «Blackjack». В результаті були знайдені схожі за функціоналом та змістом програми: онлайн гра «Blackjackist», комп'ютерна «Blackjack Gold», мобільна гра «Blackjack». Основні поняття предметної області: гра «Blackjack» та клієнт-серверна архітектура.

У казино азартних ігор, блекджек вважається грою з найкращими шансами. Навіть стверджують, що існують стратегії, які дозволяють гравцю перемагати казино. Метою блекджека є витягування карт до досягнення якомога більшої суми очок, яка не перевищує 21. У казино блекджек грається проти банку. Блекджек практично симетричний для гравців і банку, тому це більш-менш справедлива гра [1].

У грі блекджек використовується звичайна колода з 52 картами. Карти від 2 до 10 набирають очки, рівні своєму номіналу, в той час як валет, король

і дама приносять гравцеві по 10 очок. Туз може бути варіантом 1 або 11 очок, залежно від ситуації [2].

Важливим аспектом блекджеку, подібно до покеру, є ставки гравців. У різних варіаціях гри ставки можуть починатися або після того, як усі учасники отримають по одній карті, або ж після роздачі двох карт кожному гравцеві [2].

Після отримання карт кожен гравець може вирішити, чи взяти додаткову карту, чи залишитися з поточними. При цьому одна з карт дилера залишається відкритою. Коли всі гравці зроблять свій вибір, дилер відкриває другу карту і визначає переможця – того, хто набрав 21 очко або найближчу до цього кількість [2].

Клієнт-серверна модель — це структура розподіленого програмного забезпечення, що розподіляє завдання або навантаження між серверами, які надають ресурси або послуги, і клієнтами, які запитують ці послуги. Сервери обробляють запити від численних клієнтів, забезпечуючи централізоване управління даними та процесами. Такий підхід дозволяє легко масштабувати систему, а також забезпечує підвищену безпеку та контроль доступу до ресурсів [3].

Клієнт - це комп'ютер або інший пристрій із встановленим програмним забезпеченням, що звертається за сервісами до сервера. Сервер - це комп'ютерна система, що працює під управлінням спеціального програмного забезпечення для надання послуг іншим пристроям. Залежно від наданих послуг, сервери можуть бути файловими, баз даних, медіа, друку, веб-серверами або хмарними серверами, що зберігають віртуальні машини. Модель клієнт-сервер описує, як сервер надає ресурси та послуги одному або кільком клієнтам, відповідаючи на їхні запити. Сервери обслуговують різні клієнтські пристрої, такі як ноутбуки, настільні комп'ютери, планшети або смартфони[4].

Однією з ключових переваг клієнт-серверної моделі є її централізована структура, яка спрощує захист даних через реалізацію політик безпеки для контролю доступу [5].

1.1 Огляд існуючих методів вирішення завдання

Огляд існуючих методів вирішення завдання розробки ігрової програми «Blackjack» дає змогу розглянути різні підходи та технології, що використовуються в цій області. Декілька можливих методів:

- нативний десктопний додаток. Десктопний додаток – це повнофункціональна програма, що розміщується на комп'ютері користувача. Десктопний додаток працює ізольовано від інших додатків і вимагає наявності оператора (людини, яка працює з програмою). Людина взаємодіє із десктопною програмою за допомогою стандартного інтерфейсу [6]. Цей метод включає в себе розробку програми, написану під конкретну операційну систему, таку як Windows, macOS або Linux;

- веб-додаток. Веб-додаток – це програмне забезпечення або програма, доступ до якої здійснюється через веб-браузер. Інтерфейс таких додатків розробляється за допомогою мов HTML, CSS, та JavaScript, які підтримуються всіма основними браузерами, такими як Opera, Chrome, Mozilla, та Yandex. Для створення серверної частини (Back-end) можуть використовуватися різні мови програмування або фреймворки, включаючи Python, PHP, Ruby, та Java [7];

- мобільний додаток. Мобільний додаток – це тип програмного забезпечення, створений для функціонування на портативних пристроях, таких як смартфон [8]. Розробка мобільного додатка для платформ, таких як iOS та Android, дає можливість гравцям грати у гру «Blackjack» на своїх смартфонах або планшетах;

- клієнт-серверна архітектура. Використання клієнт-серверної архітектури дозволяє розділити функціональність програми між клієнтською і серверною частинами. Клієнтська частина може бути реалізована як десктопний або мобільний додаток, а серверна частина - як веб-сервер або окремий додаток, що обробляє логіку гри та забезпечує комунікацію між гравцями;

- інтернет гри. Використання інтернет-технологій для створення мультиплеєрного середовища дозволяє гравцям грати у «Blackjack» з іншими гравцями по всьому світу. Це вимагає використання мережевих протоколів та інфраструктури серверів для забезпечення з'єднання та обміну даними між гравцями.

Кожен з цих методів має свої переваги та особливості. Одна для свого програмного продукту найкращим є клієнт-серверна архітектура десктопної програми. Створення клієнт-серверної десктопної ігрової програми «Blackjack» має кілька переваг, які роблять цей підхід привабливим:

- мультиплеєрний режим гри. Клієнт-серверна архітектура дозволяє створити мультиплеєрний режим гри, де гравці зможуть змагатися один з одним. Це робить гру більш захоплюючою та соціальною, сприяє взаємодії гравців і може збільшити їх задоволення від гри;

- керування грою на сервері. Завдяки клієнт-серверній архітектурі, логіка гри, перевірка правил і управління ігровим процесом можуть бути розміщені на серверній частині. Це забезпечує більшу безпеку та контроль над грою, оскільки сервер може виконувати перевірки, запобігаючи можливим шахрайствам або неправильному виконанню правил гри;

- оновлення та розширення. Завдяки централізованому серверу, можливо легко вносити оновлення та вдосконалення до ігрової програми «Blackjack». Нові правила гри, функції чи інші зміни можуть бути впроваджені на сервері і автоматично оновлювати клієнтські додатки гравців без необхідності завантаження нової версії програми;

- зменшення навантаження на клієнтські пристрої. Використання клієнт-серверної архітектури дозволяє зменшити навантаження на клієнтські пристрої, оскільки сервер відповідає за обробку складних алгоритмів та операцій, тоді як клієнтські додатки працюють з локальними даними та забезпечують інтерфейс користувача.

1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

1.2.1 Онлайн гра «Blackjackist»

«Blackjackist» – це онлайн гра, яка дозволяє гравцям підключатися до Інтернету та грати в блекджек з іншими гравцями з усього світу. Гравці можуть змагатися в режимі мультиплеєра, спілкуватися між собою через чат, приймати участь в турнірах та випробовувати свої навички гри проти реальних противників. [9].

Переваги:

- можливість грати з гравцями з усього світу. «Blackjackist» дозволяє вам з'єднатися з гравцями з різних країн і культур, що розширює ваші можливості спілкування та взаємодії;
- режим мультиплеєра. «Blackjackist» пропонує режим мультиплеєра, де ви можете змагатися з іншими гравцями у реальному часі. Це створює більш соціальне та захоплююче ігрове досвід, де ви можете випробувати свою тактику та навички проти справжніх суперників;
- варіанти гри та налаштування. Різні варіанти блекджеку та можливості налаштування правил гри. Ви можете вибрати свої улюблені правила гри або спробувати щось нове, що додає різноманіття і цікавість до геймплею.

Недоліки:

- немає українського інтерфейсу;
- залежність від Інтернет-з'єднання. Для гри в необхідне постійне та стабільне Інтернет-з'єднання. Відсутність з'єднання або погана якість мережі може призвести до втрати зв'язку під час гри або спричинити затримки;
- можлива наявність шахрайства. Як у будь-якій онлайн-грі, існує ризик стикнутися з шахраями або недобросовісними гравцями, які можуть намагатися зловживати правилами або використовувати недозволені методи. Варто бути обережним і грати лише на надійних платформах;

– витрати на внутрішні покупки. Може містити внутрішні покупки, такі як валюта гри, бонуси або предмети, які можна придбати за реальні гроші. Це може вплинути на вартість гри або створити ситуацію, де деякі гравці мають перевагу за рахунок своїх фінансових можливостей.

На рисунку 1.1 зображено вигляд онлайн гри «Blackjackist» [9].

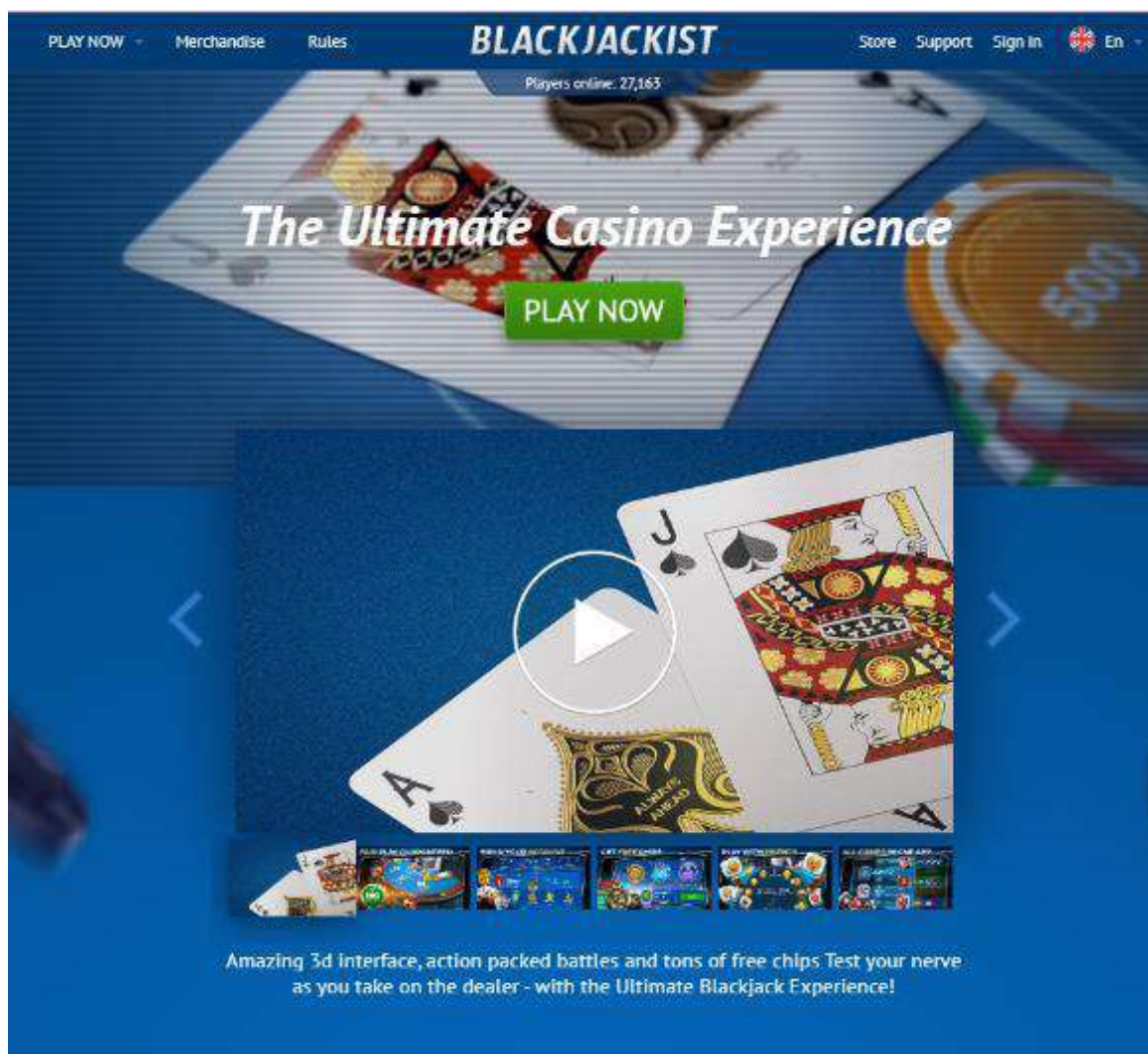


Рисунок 1.1 – Онлайн гра «Blackjackist» [9]

1.2.2 Програма «Blackjack Gold»

«Blackjack Gold» – це комп'ютерна гра в блекджек, яка пропонує реалістичний ігровий досвід та різноманітність режимів гри. Гравці можуть насолоджуватися класичним блекджеком, брати участь в турнірах, грати

проти комп'ютерних супротивників або інших гравців в режимі мультиплеєра. [10].

Переваги:

- реалістичний ігровий досвід. Програма має деталізовану графіку, звукові ефекти та анімацію, що допомагають зануритися в атмосферу казино;
- різноманітність режимів гри. Гра пропонує різні режими гри, такі як класичний блекджек, турніри, мультиплеєр та багато інших. Це дозволяє гравцям вибирати той режим, який найбільше їм до вподоби;
- налаштування правил гри. Гравці можуть налаштувати правила гри відповідно до своїх вимог. Вони можуть встановити ставки, обмеження, кількість колод та інші параметри, що дозволяють грати згідно зі своїми вподобаннями.

Недоліки:

- немає українського інтерфейсу;
- відсутність реальних гравців. У деяких випадках гра може використовувати комп'ютерних супротивників або дилера, а не реальних гравців. Це може зменшити соціальний аспект гри і позбавити можливості змагатися з реальними суперниками.
- обмежений мультиплеєр. Мультиплеєрні можливості гри «Blackjack Gold» можуть бути обмеженими порівняно з онлайн-іграми, де гравці з усього світу можуть з'єднуватися та грати разом. Це може зменшити рівень взаємодії та розмаїття ігрового досвіду;
- вартість та доступність. Деякі версії гри можуть бути платними.

В цілому, «Blackjack Gold» є ігровим програмним засобом, який надає можливість насолоджуватися грою в блекджек на комп'ютері з реалістичним ігровим середовищем та варіативністю режимів гри.

На рисунку 1.2 зображено вигляд програми «Blackjack Gold» [10].



Рисунок 1.2 – Програма «Blackjack Gold» [10]

1.2.3 Мобільна гра «Blackjack»

«Blackjack» є популярною мобільною грою, доступною на різних платформах, таких як iOS та Android. Ця гра дає можливість користувачам грати в класичний блекджек безпосередньо на своїх мобільних пристроях [11].

Переваги:

- мобільна доступність. Можливість грати в улюблену гру в будь-який час і в будь-якому місці, оскільки вони доступні на мобільних пристроях, таких як смартфони і планшети;
- зручність і простота. Інтуїтивний і зручний інтерфейс, що спрощує гру із мобільного пристрою. Ви можете легко керувати грою, виконувати рухи карт і приймати рішення всього лише кількома дотиками екрана;
- різноманітність варіантів: Є різні варіації гри, які дозволяють вам вибрати певний стиль гри або правила, які вам до вподоби.

Недоліки:

- обмежений екран. Мобільні пристрої мають обмежений екран, що може призводити до менш комфортного геймплею порівняно зі ширшим екраном комп'ютера або телевізора. Це може вплинути на відображення графіки та зручність керування грою;
- відсутність реальних гравців. У деяких випадках гра може використовувати комп'ютерних супротивників або дилера, а не реальних гравців;
- немає українського інтерфейсу.

На рисунку 1.3 зображено вигляд мобільної гри «Blackjack» [11].



Рисунок 1.3 – Мобільна гра «Blackjack» [11]

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

В таблиці 1.1 представлено порівняння існуючих аналогів та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.1 – Порівняння існуючих аналогів

Можливості	Десктопна клієнт-серверна гра Blackjack	Онлайн гра Blackjackist	Програма Blackjack Gold	Мобільна гра Blackjack
Український інтерфейс	+	-	-	-
Реєстрація та авторизація користувачів	+	+	+	+
Режим гри проти комп'ютерного дилера	+	+	+	+
Мультиплеєрний режим	+	+	+	Частково
Налаштування гри	-	+	-	+
Система ставок та виграшів	+	+	+	+
Перегляд існуючих лобі	+	Частково	Частково	Частково
Перегляд правил гри	+	+	+	+
Можливість шахрайства	-	+	-	-

В таблиці 1.1 можна побачити, що онлайн гра «Blackjackist» має великий ризик шахрайства, на відміну від інших. Мобільна гра «Blackjack» частково має мультиплеєрний режим, на відміну від інших. Також жоден з аналогів немає повної можливості перегляду існуючих лобі.

Виходячи з отриманих даних з таблиці 1.1, як прототип було обрано десктопну програму «Blackjack Gold», з наступних причин:

- функціональність;
- привабливий інтерфейс користувача;
- доступність джерел коду;
- має хороші відгуки від користувачів і високу репутацію серед гравців.

1.3 Постановка завдання роботи

В технічному завданні (ТЗ) було описано основні функціональні характеристики клієнт-серверної десктопної ігрової програми «Blackjack», а саме:

- реєстрація та авторизація користувачів. можливість створення облікового запису користувача з унікальними обліковими даними (логін, пароль) та можливість авторизуватися в програмі для доступу до гри;
- режим гри проти комп'ютерного дилера можливість грати в «Блекджек» проти комп'ютерного дилера згідно з правилами гри;
- мультиплеєрність, можливість грати в «Блекджек» з іншими гравцями в режимі реального часу;
- налаштування гри: можливість створювати та налаштовувати ігрове лобі;
- привабливий інтерфейс користувача з графічними елементами, які відображають картки, столи і інші елементи гри;
- можливість переглядати існуючі лобі, та підключатися до них;
- забезпечення взаємодії між клієнтом та сервером;
- можливість переглядати правила гри.

1.4 Висновки за розділом 1

Серед відомих методів розроблення гри «Blackjack» було вирішено створювати десктопну програму з клієнт-серверною архітектурою. Одним з найефективніших методів є використання мережевої архітектури клієнт-сервер, де клієнтська програма взаємодіє з сервером для отримання інформації та обміну даними.

Під час дослідження аналогів було виявлено кілька існуючих програмних засобів, таких як «Blackjackist», «Blackjack Gold» і мобільна гра «Blackjack». За прототип обрану програму «Blackjack Gold».

В результаті аналізу та порівняння аналогів було вирішено розробити власний привабливий та цікавий користувачам інтерфейс, який буде мати доповнений функціонал з виявлених аналогів та змінену архітектуру:

- можливість створювати та налаштовувати ігрове лобі;
- можливість переглядати існуючі лобі, та підключатися до них;
- клієнт-серверна архітектура.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Вибір інструментарію розробки програмного забезпечення є важливим етапом у розробці клієнт-серверної десктопної ігрової програми «Blackjack». Цей процес включає аналіз різних інструментів і технологій, які можуть бути використані для реалізації програмного продукту. Основні аспекти, які було розглянуто:

- мову програмування;
- інтегроване середовища розробки (IDE);
- системи керування базами даних (СКБД) для зберігання даних програми;
- структурну схему ПЗ;
- основні модулі програми.

2.1.1 Огляд особливостей мови програмування

Мова програмування C# є однією з популярних мов для розробки десктопних, веб- та серверних додатків в середовищі .NET [12].

Згідно щорічного дослідження, проведеного компанією JetBrains, яке надає глибокий аналіз популярності мов програмування, інструментів та технологій серед розробників, мова C# залишається популярною разом з іншими мовами. І з таблиці на рисунку 2.1 видно, що у 2024 році C# має приріст 4% у використанні [13].

2017	2018	2019	2020	2021	2022	2023	Likely to adopt	
65%	64%	69%	70%	69%	65%	61%	3%	JavaScript
32%	41%	49%	55%	52%	53%	54%	5%	Python
60%	55%	61%	61%	60%	54%	52%	1%	HTML / CSS
42%	47%	56%	56%	54%	49%	52%	2%	SQL
47%	51%	50%	54%	49%	48%	49%	3%	Java
–	29%	40%	39%	37%	34%	34%	1%	Shell
12%	17%	25%	28%	29%	34%	34%	6%	TypeScript
17%	18%	20%	27%	23%	25%	25%	4%	C++
20%	22%	24%	22%	21%	23%	21%	4%	C#
15%	16%	17%	23%	19%	20%	19%	2%	C
30%	26%	29%	27%	32%	20%	18%	1%	PHP
8%	12%	18%	19%	17%	19%	17%	9%	Go
2%	9%	16%	17%	14%	16%	15%	6%	Kotlin
–	2%	5%	7%	6%	9%	10%	10%	Rust
9%	8%	11%	9%	7%	7%	6%	4%	Swift
10%	8%	11%	8%	6%	5%	4%	2%	Ruby
7%	5%	6%	5%	3%	3%	3%	1%	Scala
7%	5%	6%	4%	3%	3%	2%	0%	Objective-C

Рисунок 2.1 – Дослідження компанією JetBrains [13]

За індексом ТІОВЕ, С# стала найкращою мовою програмування 2023 року. Вона поступово зайняла частину ринку Java і стає все більш популярним у таких сферах, як веб-додатки та ігри [14].

Наразі С# стала однією з найуспішніших і широко використовуваних мов Microsoft із широким застосуванням серед професійних розробників [15].

С# використовується для наступних цілей [16]:

- розробки веб-сайтів. Ця мова дозволяє створювати динамічні веб-сайти на платформі .NET або з використанням відкритого програмного забезпечення;
- розробки програм для Windows. Так як С# розроблена у Microsoft, вона частіше використовується для створення програм і додатків, специфічних для архітектури платформи Microsoft;
- розробки ігор. С# легко інтегрується з двигуном Unity. Завдяки мультиплатформеності, цю мову можна використовувати на будь-якому сучасному мобільному пристрої або ігровій консолі.

Переваги мови програмування C# включають [17]:

- бути головною мовою розроблення Microsoft і основною мовою для платформи Windows;
- використання платформи .NET, яка постійно розвивається, а також революційного .NET Core;
- високорівневий синтаксис, який спрощує розуміння для людей;
- строга типізація, що робить код безпечнішим і зменшує ризик помилок;
- спрямована на реалізацію принципів об'єктно-орієнтованого програмування, що полегшує написання коду;
- універсальність застосування;
- зручність у розробленні для 2D і 3D ігор;
- ефективність у розробленні корпоративних та серверних веб-додатків;
- використання Visual Studio, яке є одним з найзручніших середовищ розроблення ПЗ з багатим функціоналом;
- наявність докладної документації, численного ком'юніті та підтримки від Microsoft;
- оптимальна крива навчання, яка забезпечує плавний перехід від простого до складного.

Серед недоліків є [17]:

- простота синтаксису C# є відносною порівняно з іншими мовами, такими як JavaScript, Python, Ruby. Однак C# сприймається легше за більш складні мови, такі як C/C++, а також за низькорівневі мови типу Assembler тощо;
- продуктивність C# нижча порівняно з складнішими мовами програмування, наприклад, C/C++. Це пов'язано з компіляцією та абстракцією за допомогою віртуальної машини, а також іншими особливостями мови C# та платформи .NET. Ці механізми можуть вимагати додаткових системних

ресурсів, але вони забезпечують C# ту ж лаконічність і інтуїтивність синтаксису, які цінуються користувачами.

Мова програмування C# – це потужний та сучасний інструмент, для створення мультиплатформенного ПЗ, котрий постійно розвивається. Вона має високу затребуваність на сучасному ринку ІТ-розроблення. Саме тому для свого ПЗ я обрав мову C#.

Ця мова має декілька особливостей, які можуть бути корисними для розробки клієнт-серверної десктопної ігрової програми «Blackjack». Ось деякі з них:

- об'єктно-орієнтована мова. C# підтримує об'єктно-орієнтовану парадигму програмування, що дозволяє легко організувати код у вигляді класів і об'єктів. Це сприяє створенню модульного та розширюваного програмного коду;

- можливості платформи .NET. C# є однією з основних мов програмування платформи .NET, що надає доступ до широкого спектру функціональності та бібліотек, таких як робота з мережею, базами даних, графікою та багато іншого. Це дозволяє ефективно реалізувати різноманітні аспекти клієнт-серверної архітектури;

- мінімальні вимоги до розробки. Для розробки програм на C# використовується інтегроване середовище розробки (IDE) Visual Studio, яке надає потужні інструменти для створення, налагодження та тестування додатків. Це спрощує процес розробки та забезпечує зручну роботу з кодом;

- безпека та надійність. Мова C# надає вбудовану підтримку для використання механізмів безпеки та перевірки типів, що допомагає уникати багів та забезпечує надійність програмного забезпечення. Також можливе застосування політик безпеки для обмеження доступу до ресурсів.

2.1.2 Огляд особливостей обраного середовища розробки

Обране середовище розробки грає важливу роль у процесі створення клієнт-серверної десктопної ігрової програми «Blackjack». В таблиці 2.1 було порівняно основні IDE для розробки мовою C#.

Таблиця 2.1 – Порівняння середовищ розробки

Характеристика	Microsoft Visual Studio	Project Rider	Visual Studio Code
Підтримка мови C#	+	+	+
Підтримка WCF	+	+	-
Підтримка WCF	+	+	-
Інтегрована розробка	Широкий набір	Обмежений набір	Обмежений набір
Налагоджувальні засоби	Повністю функціональні	Повністю функціональні	Обмежені засоби
Розширення та плагіни	+	+	+
Операційні системи	Windows	Кросплатформенний	Кросплатформенний
Вартість	Частково платно	Платно	Безкоштовно

Таблиця 2.1 надає порівняльний огляд трьох IDE: Microsoft Visual Studio, Project Rider та Visual Studio Code. Microsoft Visual Studio та Project Rider є повноцінними IDE з підтримкою мови C#, WPF та WCF. Вони мають розширений набір інструментів та налагоджувальних засобів.

Visual Studio Code є легким та кросплатформеним редактором коду, який також підтримує мову C#. Він має обмежений набір інструментів та налагоджувальних засобів, але забезпечує широкий вибір розширень та плагінів.

Microsoft Visual Studio є потужним та повнофункціональним інструментом розробки для мови C# та платформи Windows. Воно має широкий функціонал, інтегрованість з екосистемою Microsoft, підтримку різних мов програмування та велике співтовариство розробників. Це робить його відмінним вибором для розробки клієнт-серверної десктопної ігрової програми «Blackjack».

Опитування від Stack Overflow показало, що Microsoft Visual Studio Code і повноцінна Visual Studio IDE, встановили домінування, посідаючи номери 1 і 2 відповідно в рейтингу IDE за останні п'ять років [18]. Це можна побачити на рисунку 2.2 [19].



Рисунок 2.2 – Опитування від Stack Overflow

Visual Studio – це потужний інструмент розробника, який можна використовувати для завершення всього циклу розробки в одному місці. Це комплексне інтегроване середовище розробки (IDE). Воно дозволяє писати, редагувати, налагоджувати та створювати код, а також розгортати програми. Крім цього, Visual Studio включає в себе компілятори, інструменти автодоповнення коду, системи керування версіями, розширення та інші корисні функції для полегшення кожного етапу розробки програмного забезпечення [20].

Різноманітні версії цієї IDE здатні створювати різні типи програм: веб-додатки, мобільні застосунки, десктопні програми, відеоігри тощо. Цей рід програмного забезпечення містить широкий набір інструментів для тестування сумісності програм. Завдяки своїм можливостям Visual Studio стає найкращим та найлегшим інструментом для створення різних програм, як і для звичайних користувачів, так і для спеціалізованих фірм. Visual Studio є майже найстарішим IDE для створення звичаних консольних програм, так і навіть для програмних продуктів, що мають не звичайний інтерфейс, а графічний. Можливість додавати різні плагіни надає розширення функціональності середовища, включаючи підтримку кроссплатформенності [21].

Основні особливості [21]:

- розширена бібліотека додатків, що постійно поповнюється;
- можливість налаштовувати панель та закріплювати вікна;
- простий робочий процес і структурована файлова ієрархія;
- моніторинг продуктивності в реальному часі;
- інструменти автоматизації;
- легкий рефакторинг коду та вставка фрагментів;
- підтримка розділеного екрану;
- зручний список помилок для швидкої налагодження.

Компанія Microsoft розробляє інтегроване середовище розробки, спрямоване переважно на створення програмного забезпечення для платформи .NET, відоме під назвою Visual Studio [22].

.NET – це платформа від Майкрософт, призначена для розробки та виконання програмного забезпечення. Основними мовами програмування для цієї платформи є C#, F тощо [17].

.NET уможлиблює збереження всіх необхідних компонентів та модулів у компактному форматі. Це сприяє втіленню принципу «не повторюй себе», економить простір на комп'ютері та дозволяє розробникам .NET зосередитися на розвитку одного спеціалізованого середовища, яке підтримує декілька мов програмування одночасно, а також надає інші зручності [17].

WPF (Windows Presentation Foundation) - це остання технологія від Microsoft для розробки інтерфейсу користувача (GUI) на платформі .NET [23].

GUI (Graphical User Interface) - це візуальний інтерфейс користувача, цей інфейс є майже в кожній програмі. Windows має свій власний інтерфейс для своїх власних програмних продуктів. Є безліч різних бібліотек GUI, які використовують розробники для своїх програм, але для розробки на .NET найбільш цікавими є WinForms і WPF [23].

Переваги використання WPF наступні[24]:

- WPF - це новіша технологія, яка краще відповідає сучасним стандартам;
- компанія Microsoft використовує WPF у багатьох своїх нових програмах, таких як Visual Studio;
- WPF є більш гнучкою, і ви можете досягти більшого функціоналу без необхідності писати власні або придбувати сторонні елементи керування;
- оскільки WPF є новішою за WinForms, то, коли вам все ж потрібні якісь специфічні елементи керування від інших розробників, ймовірніше, що ці розробники будуть більше зосереджені саме на WPF, ніж на WinForms;
- використання XAML дозволяє розділити роботу між дизайнером і програмістом;
- за допомогою зв'язування даних можна досягти більшої чіткості у відносинах між даними і графічним інтерфейсом користувача;

– WPF використовує графічний процесор для малювання GUI, що сприяє прискоренню роботи програм.

2.1.3 Огляд особливостей обраної системи управління базами даних

Система керування базами даних (СКБД) — системне програмне забезпечення для створення та керування базами даних. СКБД дозволяє кінцевим користувачам створювати, захищати, читати, оновлювати та видаляти дані в базі даних [25].

Основні характеристики [25]:

- контроль за надмірністю даних для уникнення зайвої інформації;
- забезпечення несуперечливості даних для уникнення суперечок в інформації;
- підтримка цілісності бази даних, що включає коректність та відсутність суперечностей;
- цілісність описується за допомогою обмежень, що допомагає забезпечити правильність даних;
- забезпечення незалежності прикладних програм від даних, щоб зміни в даних не впливали на функціонування програм;
- можливість спільного використання даних між різними програмами для оптимізації ресурсів та підвищення ефективності;
- підвищений рівень безпеки для захисту конфіденційності та цілісності даних.

Microsoft SQL Server є реляційною системою управління базами даних, створеною компанією Microsoft [26].

Основні переваги MS SQL Server включають наступне:

- надійність і стабільність. MS SQL Server відомий своєю надійністю та стабільністю. Він має потужні механізми для забезпечення цілісності даних, управління транзакціями і відновлення даних після збоїв;

- висока продуктивність. MS SQL Server має оптимізовані механізми обробки запитів і роботи з даними, що дозволяє досягати високої продуктивності при великому обсязі роботи з базою даних. Він також підтримує паралельну обробку запитів і може працювати з багатопотоковими застосунками;

- розширені можливості. MS SQL Server надає багато розширених можливостей, таких як реплікація даних, кластеризація, вищий рівень безпеки, аналітичні засоби та інструменти бізнес-інтелекту. Він також підтримує високу масштабованість, що дозволяє розширювати базу даних зростанням обсягу даних та навантаження;

- інструменти розробки і адміністрування. MS SQL Server постачається з потужними інструментами розробки, такими як Microsoft SQL Server Management Studio (SSMS), які допомагають в розробці, налагодженні та адмініструванні баз даних. Ці інструменти мають дружній інтерфейс та великий набір функцій;

- широкий спектр підтримки. MS SQL Server підтримує різноманітні мови програмування, такі як C#, Java, Python, PHP і багато інших. Він також підтримує стандарти SQL і забезпечує можливість інтеграції з іншими інструментами та платформами.

Саме через ці переваги я обрав для свого застосунку MS SQL Server.

2.2 Структурна схема програми

Для реалізації моєї програми я використав клієнт-серверну архітектуру, яка дозволяє розділити функціональність між клієнтом і сервером. Це має кілька переваг, включаючи ефективне керування та обробку даних, можливість масштабування, централізоване керування та забезпечення безпеки даних.

Для реалізації цієї архітектури використовувався WCF (Windows Communication Foundation) в середовищі Visual Studio.

Для забезпечення виконання функціоналу, зазначеного у ТЗ, була створена структурна схема, яка ілюструє взаємодію між клієнтом і сервером. Деталі цієї схеми можна знайти у рисунку 2.3.

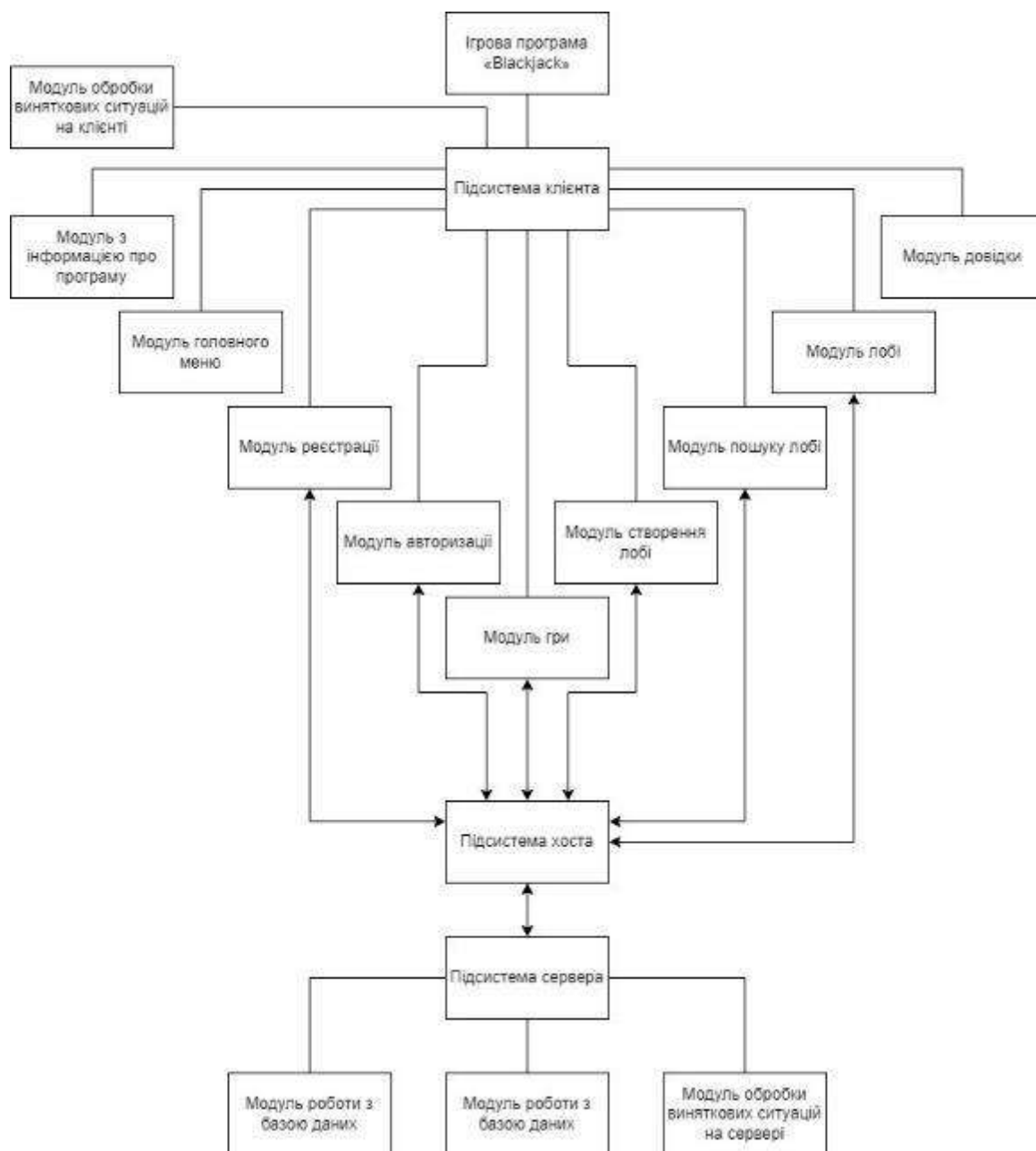


Рисунок 2.3 – Структурна схема

2.3 Опис модулів розроблюваної системи

У таблиці 2.2 наведено опис модулів реалізованого ПЗ з визначенням їх основних функцій.

Таблиця 2.2 – Опис модулів реалізованого ПЗ з визначенням основних функцій

Модуль	Опис
Авторизації	Відповідає за авторизацію користувачів у застосунок
Реєстрації	Відповідає за реєстрацію нових акаунтів в застосунок
Головного меню	Відповідає за навігацію в застосунку за допомоги реалізації головного меню
Створення лобі	Відповідає за створення і налаштування лобі
Пошуку лобі	Відповідає за пошук і підключення до лобі
Лобі	Відповідає за реалізацію ігрового лобі та відслідковування статусу готовності гравців
Гри	Відповідає за ігровий процес, за правилами гри «Blackjack»
Довідки	Відповідає за надання інформації про те як користуватися застосунком
Інформації про програму	Відповідає за надання інформації про програму і розробника
Роботи з БД	Відповідає за взаємодію з базою даних (додавання, зміна і отримання інформації)
Обробки виняткових ситуацій на клієнті	Відповідає обробку виняткових ситуацій на клієнті та повідомлення про них
Обробки виняткових ситуацій на сервері	Відповідає обробку виняткових ситуацій на сервері та робить записи про них

2.4 Висновки за розділом 2

Під час проектування системи для клієнт-серверної десктопної ігрової програми «Blackjack» було зроблено вибір та обґрунтовано структуру

системи, яка максимально задовольняє вимоги та потреби проекту. Основні висновки щодо вибору та обґрунтування структури системи такі:

- клієнт-серверна архітектура була обрана з метою розділення функціональності між клієнтом і сервером. Це дозволяє ефективно керувати та обробляти дані, забезпечує масштабованість, централізоване керування та забезпечує безпеку даних;

- використання мови програмування C# для реалізації клієнт-серверної десктопної ігрової програми має свої переваги. C# є мовою з високим рівнем абстракції, має широкі можливості для розробки графічного інтерфейсу за допомогою WPF, а також має підтримку для реалізації клієнт-серверної архітектури за допомогою WCF;

- використання Microsoft Visual Studio як інтегрованого середовища розробки є обґрунтованим вибором. Visual Studio надає потужні інструменти для розробки програмного забезпечення на мові C#, має велику спільноту розробників, підтримує роботу з WPF і WCF, а також забезпечує зручний інтерфейс для управління проектом;

- використання MS SQL Server для бази даних є обґрунтованим вибором з урахуванням його основних переваг, таких як надійність, масштабованість, висока продуктивність та підтримка різноманітних функціональних можливостей.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування розробленої програми

Основні алгоритми, які можуть виконуватися при роботі із розробленою програмою зображені на рисунку 3.1.

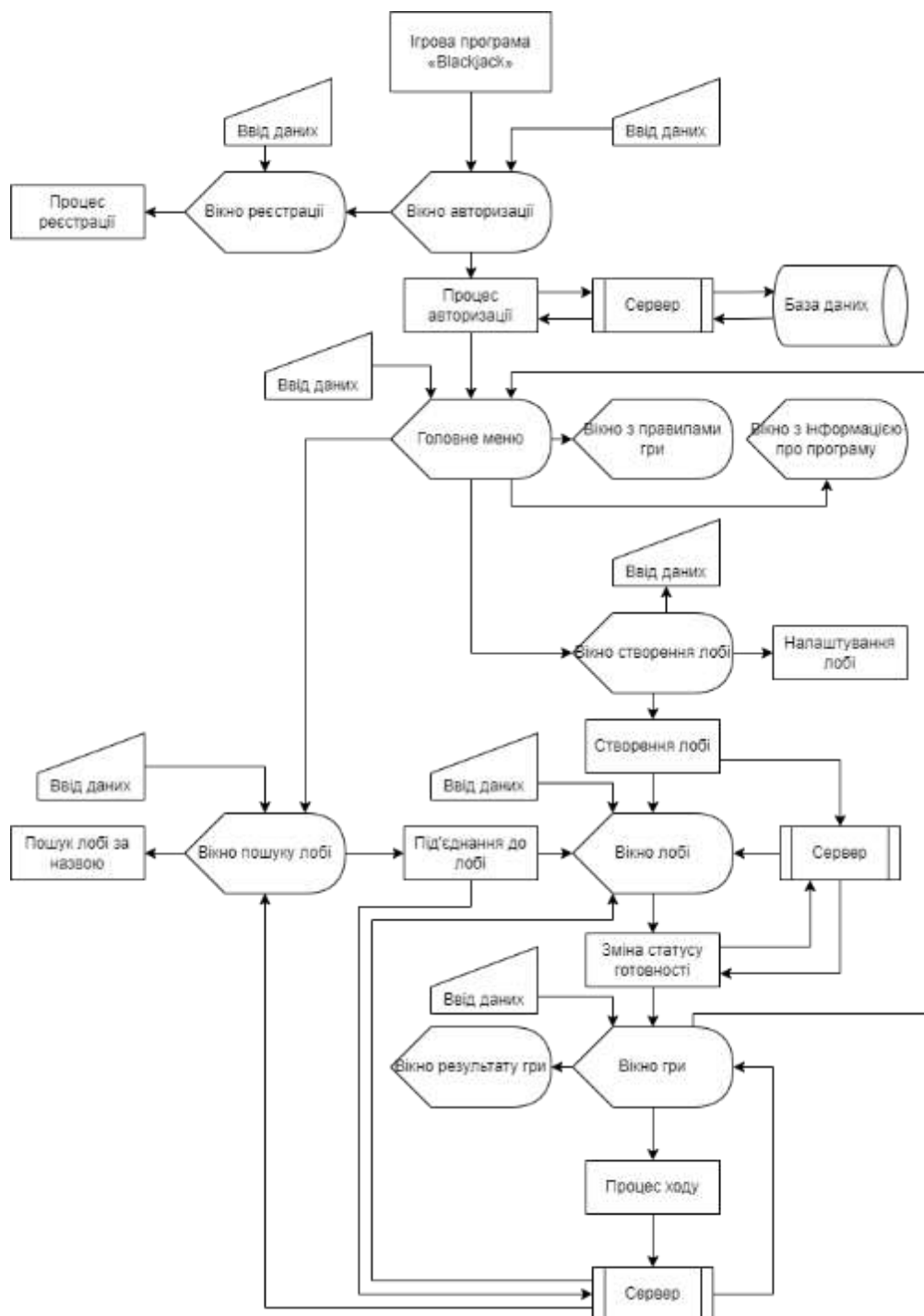


Рисунок 3.1 – Функціональна схема

Під час розробки ПЗ «Blackjack», були розроблені наступні алгоритми:

- процес авторизації;
- процес реєстрації;
- налаштування лобі;
- створення лобі;
- під'єднання до лобі;
- зміна статусу готовності;
- процес ходу.

3.1.1 Алгоритм процесу авторизації

Цей алгоритм створено для реалізації авторизації у застосунок, під обраним логіном. На рисунку 3.2 зображено блок-схема алгоритму.

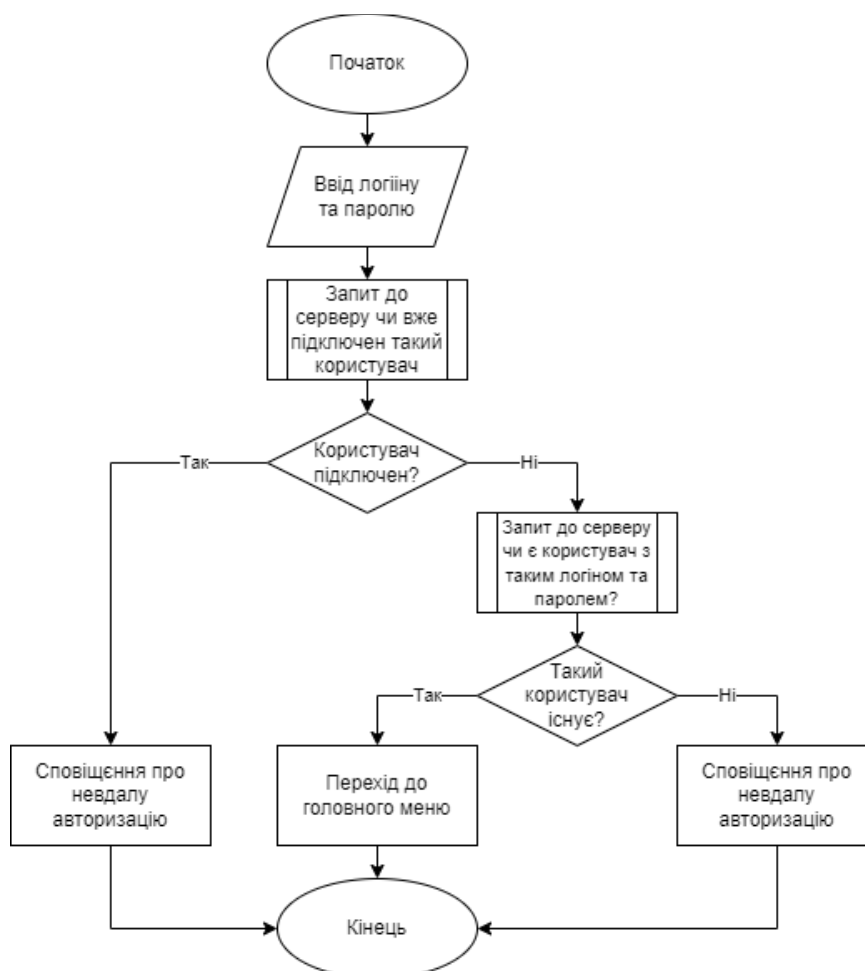


Рисунок 3.2 – Блок-схема для алгоритму процесу авторизації

3.1.2 Алгоритм процесу реєстрації

Цей алгоритм створено для реалізації створення акаунту з обраним логіном та паролем, для подальшої авторизації. На рисунку 3.3 зображено блок-схема алгоритму.

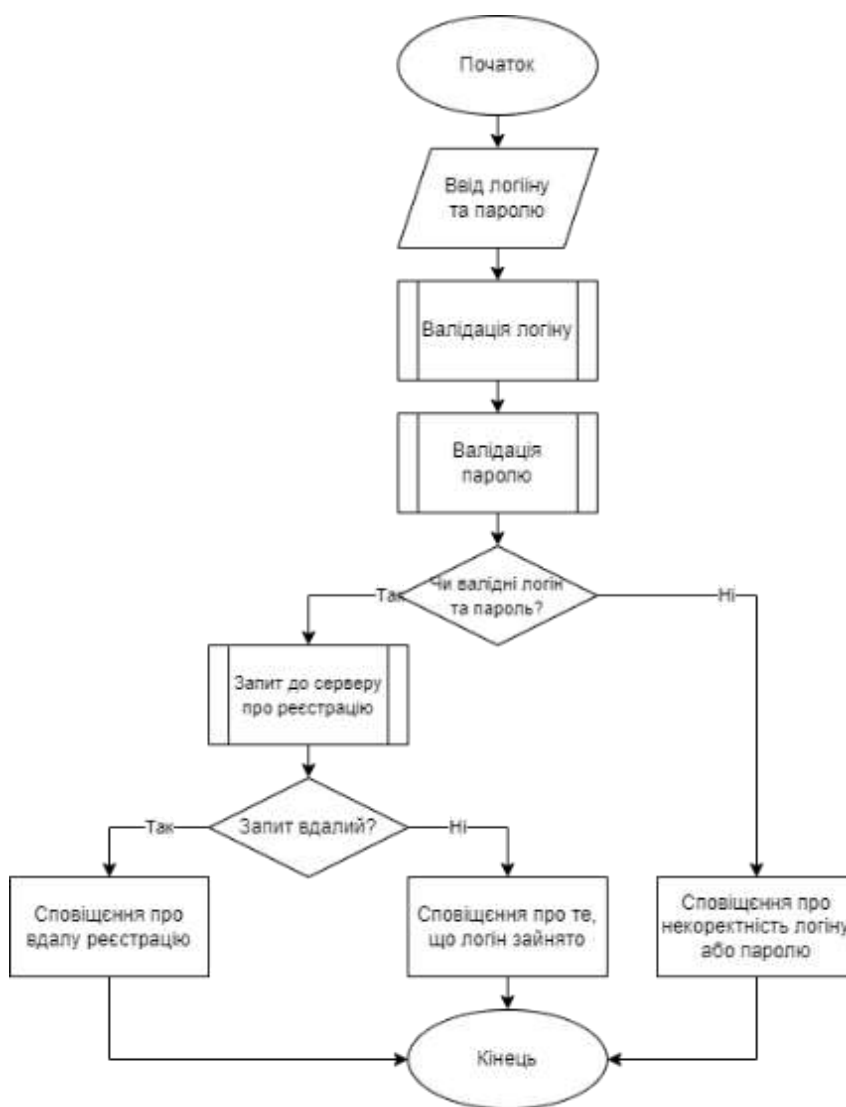


Рисунок 3.3 – Блок-схема для алгоритму реєстрації процесу авторизації

3.1.3 Алгоритм налаштування лобі

Цей алгоритм створено для реалізації внесення змін в налаштування лобі. На рисунку 3.4 зображено блок-схема алгоритму.



Рисунок 3.4 – Блок-схема для алгоритму налаштування лобі

3.1.4 Алгоритм створення лобі

Цей алгоритм створено для реалізації створення лобі, для подальшого пошуку гравців, для ігрової сесії. На рисунку 3.5 зображено блок-схема алгоритму.

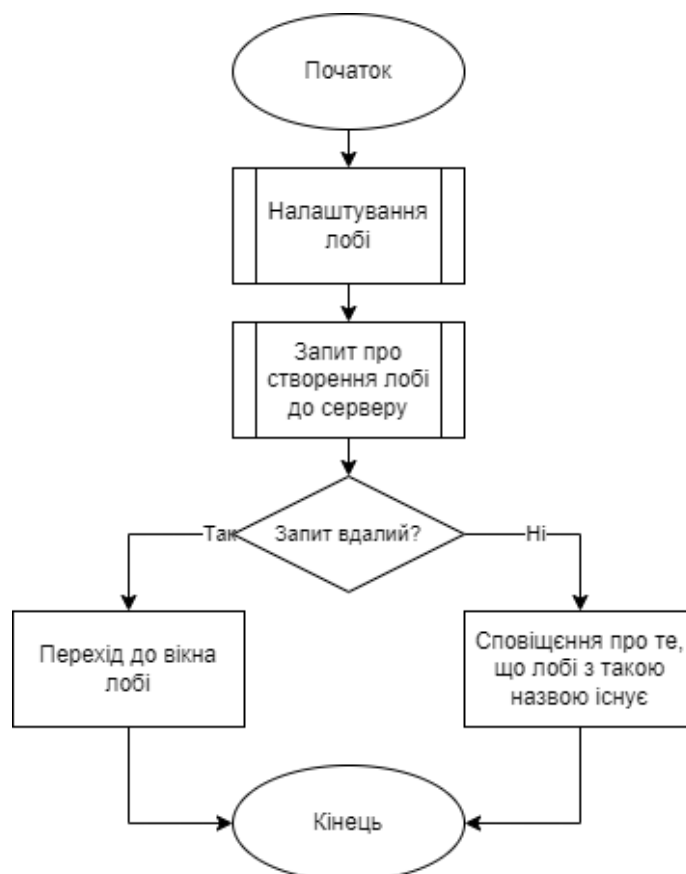


Рисунок 3.5 – Блок-схема для алгоритму створення лобі

3.1.5 Алгоритм під'єднання до лобі

Цей алгоритм створено для реалізації під'єднання користувача до обраного лобі. На рисунку 3.6 зображено блок-схема алгоритму.

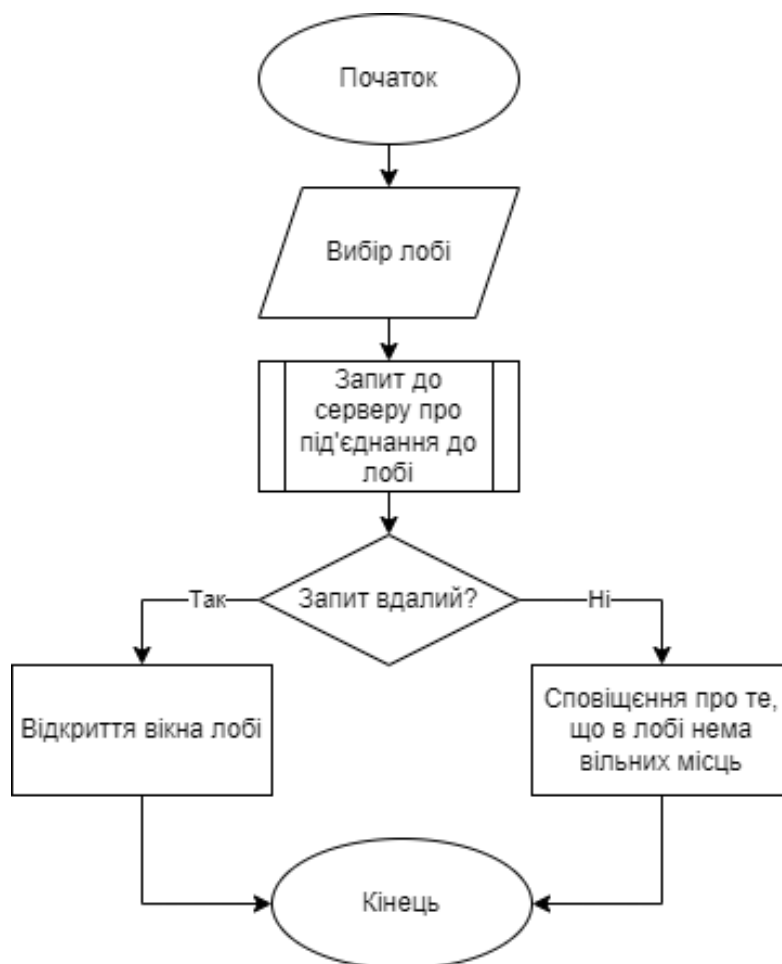


Рисунок 3.6 – Блок-схема для алгоритму під'єднання до лобі

3.1.6 Алгоритм зміна статусу готовності

Цей алгоритм створено для реалізації зміни статусу готовності користувачів, для відстеження моменту, коли треба розпочати ігрову сесію. На рисунку 3.7 зображено блок-схема алгоритму.



Рисунок 3.7 – Блок-схема для алгоритму зміна статусу готовності

3.1.7 Алгоритм процесу ходу

Цей алгоритм створено для реалізації процесу ходу гравців та дилера, що необхідно для створення ігрового процесу. На рисунку 3.8 зображено блок-схема алгоритму.

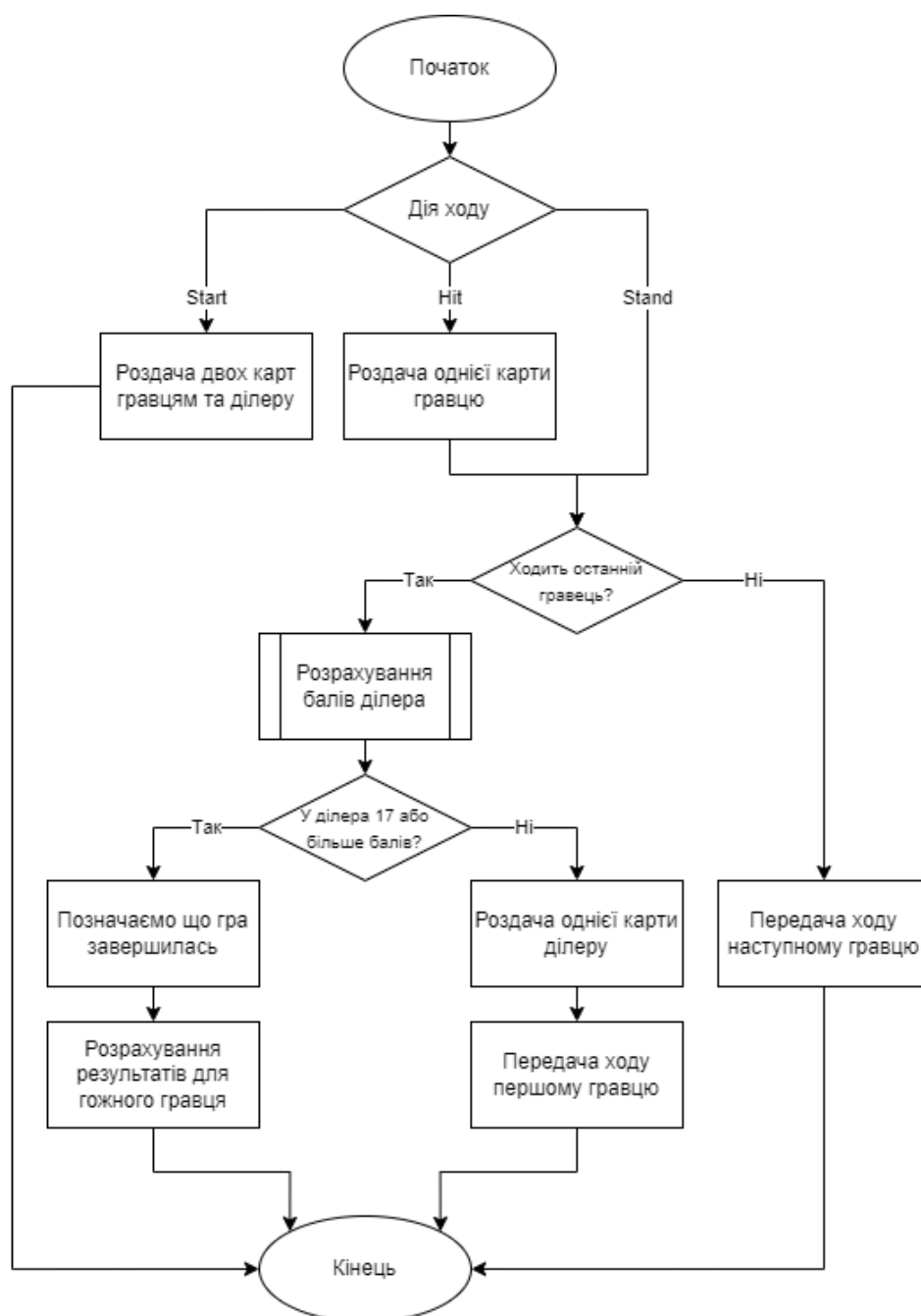


Рисунок 3.8 – Блок-схема для алгоритму процесу ходу

3.2 Вибір шаблону проектування

У розробці архітектури для різноманітних завдань широко використовуються патерни, також відомі як шаблони. Вони сприяють ефективному вирішенню схожих задач, що значно полегшує процес розробки та сприяє економії часу. Коли виникає конкретне завдання у певному

контексті, застосування паттернів дозволяє уникнути зайвих зусиль і скористатися готовими рішеннями [27].

Шаблон описує взаємодію різних елементів та класів, які формують архітектуру програмного рішення, і відповідає вимогам конкретного завдання та проектування. Використання паттернів дозволяє ідентифікувати ключові об'єкти структури та використовувати їх повторно, що спрощує процес побудови архітектури [27].

При проектуванні клієнт-серверної десктопної ігрової програми «Blackjack» було зроблено вибір шаблону проектування «клієнт-сервер».

Даний шаблон використовується для розподілу функцій між клієнтською та серверною частинами програмного забезпечення [28].

Обрання клієнт-серверної архітектури дозволяє ефективно управляти та опрацьовувати даних, забезпечує масштабованість системи, централізоване управління та забезпечує безпеку інформації. Клієнт надсилає запити до сервера, де вони обробляються, і результат повертається клієнту. Сервер може обслуговувати кілька клієнтів одночасно [28].

Цей шаблон проектування дозволяє зручно розподілити функціонал між клієнтом та сервером, що спрощує процес розробки та підтримки програми. Використання клієнт-серверної архітектури також покращує швидкодію та продуктивність програми [28].

У даному проекті для реалізації клієнт-серверної архітектури було використано технологію Windows Communication Foundation (WCF), яка надає зручні засоби для створення сервісів та взаємодії між клієнтом та сервером.

Вибір шаблону проектування «клієнт-сервер» був обґрунтований його відповідністю до потреб проекту, забезпеченням необхідної функціональності та забезпеченням ефективності та безпеки системи. Зображення архітектури можна побачити на рисунку 3.9.

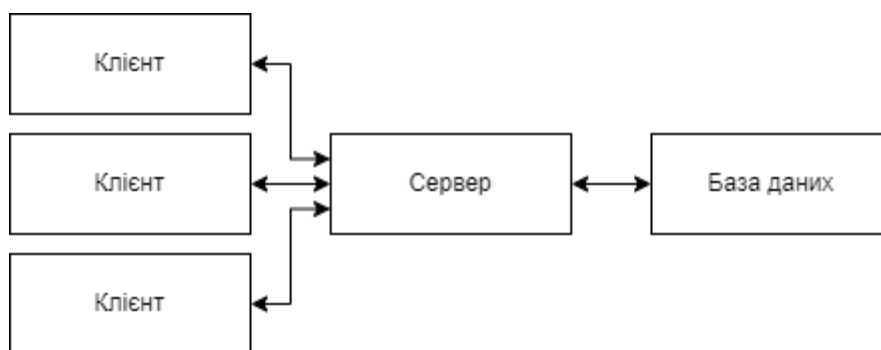


Рисунок 3.9 – Клієнт-серверна архітектура

3.3 Опис модулів розроблюваної системи

В ході роботи для розроблених модулів були утворенні такі класи на клієнтській частині (рисунок 3.10):

- AboutTheProgramWindow;
- AuthorizationWindow;
- BlackjackClientException;
- CreateLobbyWindow;
- EndOfGameWindow;
- GameProcess;
- GameRulesWindow;
- GameWindow;
- LobbySearchWindow;
- LobbyWindow;
- MainWindow;
- RegistrationWindow;
- LobbyMember.

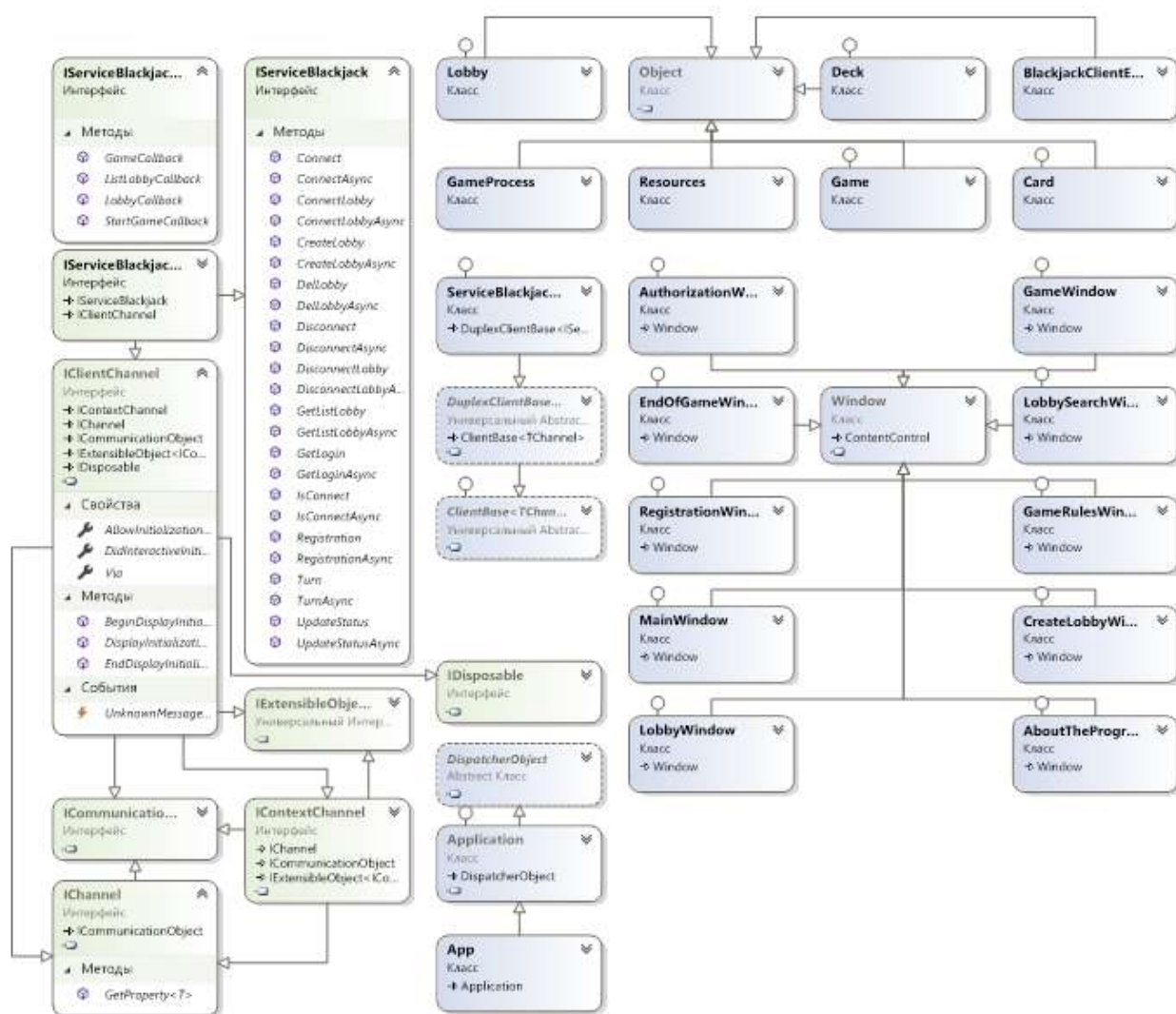


Рисунок 3.10 – Діаграма класів клієнтської частини

В ході роботи для розроблених модулів були утворенні такі класи на серверній частині (рисунок 3.11):

- BlackjackException;
- DataBaseAPI;
- Deck;
- Card;
- Game;
- Lobby;
- ServerUser;
- ServiceBlackjack.

В ході роботи для розроблених модулів були утворенні такі інтерфейси на серверній частині (рисунок 3.11):

- IServiceBlackjack;
- IServiceBlackjackCallback.

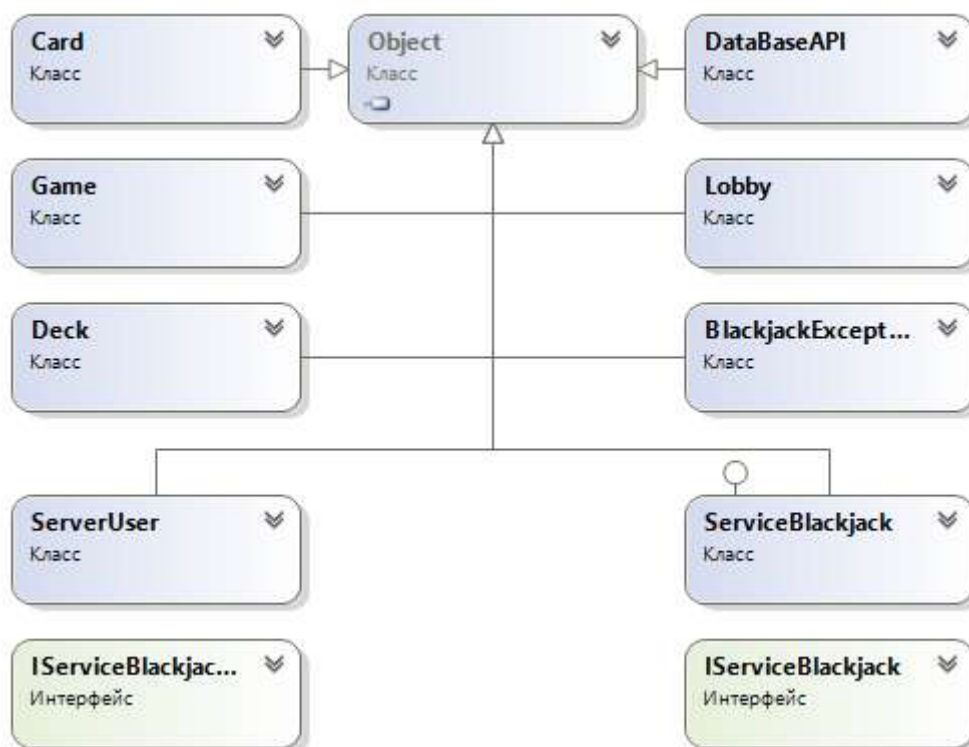


Рисунок 3.11 – Діаграма класів серверної частини

В ході роботи для розроблених модулів були утворенні такий клас на хост-частині (рисунок 3.12):

- Program.

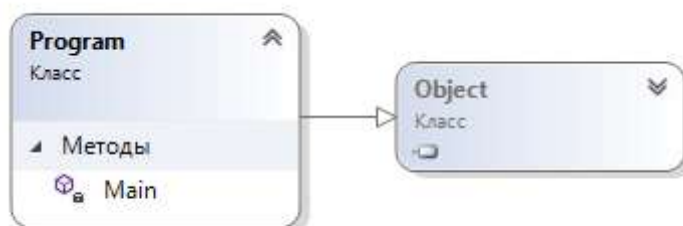


Рисунок 3.12 – Діаграма класів хост-частини

3.3.1 Клас AboutTheProgramWindow

Ціль класу полягає у реалізації вікна, яке буде мати інформацію про розробника. Дані та методи класу наведені у таблиці 3.1.

Таблиця 3.1 – Опис полів та методів класу AboutTheProgramWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
bool StandartStyle	Змінна відповідаюча за стиль вікна
public AboutTheProgramWindow(bool StandartStyle)	Конструктор вікна
private void StyleUpdate()	Встановлення стилю вікна

3.3.2 Клас AuthorizationWindow

Ціль класу полягає у реалізації вікна, яка буде надавати можливість користувачам авторизуватися в свій акаунт.

Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 – Опис полів та методів класу AuthorizationWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public AuthorizationWindow()	Конструктор вікна
public void LobbyCallback(Lobby lobby)	Оновлення активного лобі

Продовження таблиці 3.2

1	2
public void ListLobbyCallback(Lobby[] lobbys)	Оновлення списку лобі
public void StartGameCallback(Game game)	Оновлення початої гри
public void GameCallback(Game game)	Оновлення підчас гри
private:	
MainWindow MainForm	Змінна головного вікна
RegistrationWindow RegistrationWindowForm	Змінна вікна «Реєстрації»
ServiceBlackjackClient client	Змінна для роботи з сервером
int ID	ID користувача
private void ButtonLogin_Click(object sender, RoutedEventArgs e)	Кнопка вхід
private void ButtonSignUp_Click(object sender, RoutedEventArgs e)	Кнопка реєстрація
void ConnectUser()	Підключення користувача до серверу
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.3 Клас **BlackjackClientException**

Ціль класу полягає у обробці виняткових ситуацій у клієнті. Дані та методи класу наведені у таблиці 3.3.

Таблиця 3.3 – Опис полів та методів класу BlackjackClientException

Поля та методи класу	Опис
1	2
public:	
public BlackjackClientException(int code)	Конструктор класу, в якому обирається помилка

3.3.4 Клас CreateLobbyWindow

Ціль класу полягає у реалізації вікна зі створенням лобі. Дані та методи класу наведені у таблиці 3.4.

Таблиця 3.4 – Опис полів та методів класу CreateLobbyWindow

Поля та методи класу	Опис
1	2
public:	
public LobbyWindow LobbyWindowForm	Змінна вікна з лобі
bool StandartStyle	Змінна відповідаюча за стиль вікна
public CreateLobbyWindow(MainWindow MainWindowForm, int ID, bool StandartStyle)	Конструктор вікна
private void StyleUpdate()	Встановлення стилю вікна
private:	
MainWindow MainWindowForm	Змінна головного вікна
ServiceBlackjackClient client	Змінна для роботи з сервером
int ID	ID користувача
private void ButtonBack_Click(object sender, RoutedEventArgs e)	Кнопка назад

Продовження таблиці 3.4

1	2
private void ButtonCrateLobby_Click(object sender, RoutedEventArgs e)	Кнопка створити
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.5 Клас EndOfGameWindow

Ціль класу полягає у відображенні вікна з результатом гри. Дані та методи класу наведені у таблиці 3.5.

Таблиця 3.5 – Опис полів та методів класу EndOfGameWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public EndOfGameWindow(int result, string login, bool StandartStyle)	Конструктор вікна, в якому обирається результат гри

3.3.6 Клас GameProcess

Ціль класу полягає у реалізації алгоритму гри. Дані та методи класу наведені у таблиці 3.6.

Таблиця 3.6 – Опис полів та методів класу GameProcess

Поля та методи класу	Опис
1	2
public:	
public int Score(Card[] Hand)	Розрахунок кількості балів в руці
public Game Turn(Game game, string action)	Процес ходу

3.3.7 Клас GameRulesWindow

Ціль класу полягає у реалізації вікна з правилами гри.

Дані та методи класу наведені у таблиці 3.7.

Таблиця 3.7 – Опис полів та методів класу GameRulesWindow

Поля та методи класу	Опис
1	2
public:	
public GameRulesWindow(MainWindow MainWindowForm, bool StandartStyle)	Конструктор вікна
bool StandartStyle	Змінна відповідаюча за стиль вікна
private void StyleUpdate()	Встановлення стилю вікна
private:	
MainWindow MainWindowForm	Змінна головного вікна
private void ButtonBack_Click(object sender, RoutedEventArgs e)	Кнопка створити

Продовження таблиці 3.7

1	2
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.8 Клас GameWindow

Ціль класу полягає у відображенні ігрового процесу. Дані класу наведені у таблиці 3.8.

Таблиця 3.8 – Опис полів класу GameWindow

Поля структури	Опис
1	2
public:	
public GameWindow(LobbyWindow lobbyWindowForm, int ID, Game game, bool StandartStyle)	Конструктор вікна
public void WindowUpdate(Game game)	Оновлення вікна
bool StandartStyle	Змінна відповідаюча за стиль вікна
private void StyleUpdate()	Встановлення стилю вікна
private:	
LobbyWindow lobbyWindowForm	Змінна вікна з лобі
GameProcess gameProcess	Змінна процесу гри
EndOfGameWindow endOfGameWindowForm	Змінна вікна з результатом
Game game	Інформація про гру

Продовження таблиці 3.8

1	2
ServiceBlackjackClient client	Змінна для роботи з сервером
int ID	ID користувача
private void Window_Loaded(object sender, RoutedEventArgs e)	Відкриття вікна
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
void EndOfGame()	Кінець гри
private void ButtonHit_Click(object sender, RoutedEventArgs e)	Взяти карту
private void ButtonStand_Click(object sender, RoutedEventArgs e)	Не брати карту

3.3.9 Клас LobbySearchWindow

Ціль класу полягає у реалізації та відображення вікна з пошуком лобі, реалізації функції пошуку лобі за назвою та алгоритму підключення до лобі.

Дані та методів класу наведені у таблиці 3.9.

Таблиця 3.9 – Опис полів та методів модулю LobbySearchWindow

Поля та методи модулю	Опис
1	2
public:	
public LobbyWindow LobbyWindowForm	Змінна вікна з лобі
public LobbySearchWindow(MainWindowForm, int ID, bool StandartStyle)	Конструктор вікна

Продовження таблиці 3.9

1	2
public void UpdateLobbyList(Lobby[] lobbys)	Оновлює зміну списку лобі та сам список
bool StandartStyle	Змінна відповідаюча за стиль вікна
private void StyleUpdate()	Встановлення стилю вікна
<i>private:</i>	
MainWindow MainWindowForm	Змінна головного вікна
Lobby[] lobbys	Список лобі
ServiceBlackjackClient client	Змінна для роботи з сервером
int ID	ID користувача
void UpdateLobbyList()	Оновлення списку лобі
private void ButtonBack_Click(object sender, RoutedEventArgs e)	Кнопка назад
private void ButtonEnter_Click(object sender, RoutedEventArgs e)	Кнопка зайти
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
private void Window_Loaded(object sender, RoutedEventArgs e)	Відкриття вікна
private void SearchBox_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)	Зміна тексту пошуку

3.3.10 Клас LobbyWindow

Ціль класу полягає у відображенні та реалізації вікна з лобі. Дані класу наведені у таблиці 3.10.

Таблиця 3.10 – Опис полів класу LobbyWindow

Поля структури	Опис
1	2
<i>public:</i>	
public GameWindow gameWindowForm	Змінна вікна з ігровим процесом
public bool GameClosing	Змінна що позначає що треба закрити вікно коли лобі вже нема на сервері
public LobbyWindow(LobbySearchWindow LobbySearchWindowForm, int ID, Lobby lobby, bool StandartStyle)	Конструктор вікна при відкритті з вікна пошуку
public LobbyWindow(CreateLobbyWindow CreateLobbyWindowForm, int ID, Lobby lobby, bool StandartStyle)	Конструктор вікна при відкритті з вікна створення
public void UpdateLobby(Lobby lobby)	Оновлення інформації про лобі
public void StartGame(Game game)	Запуск гри
bool StandartStyle	Змінна відповідаюча за стиль вікна
private void StyleUpdate()	Встановлення стилю вікна
<i>private:</i>	
LobbySearchWindow LobbySearchWindowForm	Змінна вікна з лобі

Продовження таблиці 3.10

1	2
CreateLobbyWindow CreateLobbyWindowForm	Змінна процесу гри
int ID	ID користувача
ServiceBlackjackClient client	Змінна для роботи з сервером
Lobby lobby	Інформація про лобі
void UpdateLobby()	Оновлення інформації у вікні
private void ButtonExit_Click(object sender, RoutedEventArgs e)	Кнопка вихід
private void ButtonReady_Click(object sender, RoutedEventArgs e)	Кнопка готов
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
private void Window_Loaded(object sender, RoutedEventArgs e)	Завантаження вікна

3.3.11 Клас MainWindow

Ціль класу полягає у відображенні головного вікна. Дані класу наведені у таблиці 3.11.

Таблиця 3.11 – Опис полів класу MainWindow

Поля структури	Опис
1	2
public:	
public CreateLobbyWindow CreateLobbyWindowForm	Змінна вікна з створенням лобі

Продовження таблиці 3.11

1	2
public LobbySearchWindow LobbySearchWindowForm	Змінна вікна з пошуком лобі
public MainWindow(AuthorizationWindow AuthorizationWindowForm, int ID, bool StandartStyle)	Конструктор вікна
bool StandartStyle	Змінна відповідаюча за стиль вікна
private void StyleUpdate()	Встановлення стилю вікна
private:	
GameRulesWindow GameRulesWindowForm	Змінна вікна з правилами гри
AboutTheProgramWindow AboutTheProgramWindow	Змінна вікна з інформацією про розробника
AuthorizationWindow AuthorizationWindowForm	Змінна вікна з авторизацією
int ID	ID користувача
ServiceBlackjackClient client	Змінна для роботи з сервером
private void ButtonNewGame_Click(object sender, RoutedEventArgs e)	Кнопка нова гра
private void ButtonConnectToGame_Click(object sender, RoutedEventArgs e)	Кнопка приєднатися до гри
private void ButtonGameRules_Click(object sender, RoutedEventArgs e)	Кнопка правила гри
private void ButtonAboutTheProgram_Click(object sender, RoutedEventArgs e)	Кнопка про програму

Продовження таблиці 3.11

1	2
private void ButtonExit_Click(object sender, RoutedEventArgs e)	Кнопка вихід
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.12 Клас RegistrationWindow

Ціль класу – реалізація реєстрації. Дані класу наведені у таблиці 3.12.

Таблиця 3.12 – Опис полів класу RegistrationWindow

Поля структури	Опис
1	2
public:	
public RegistrationWindow(AuthorizationWindow AuthorizationWindowForm)	Конструктор вікна
private:	
AuthorizationWindow AuthorizationWindowForm	Змінна вікна з авторизацією
ServiceBlackjackClient client	Змінна для роботи з сервером
private void ButtonRegister_Click(object sender, RoutedEventArgs e)	Кнопка зареєструватися
private void ButtonBack_Click(object sender, RoutedEventArgs e)	Кнопка назад
bool LoginValidation(string Login)	Валідація логіну

Продовження таблиці 3.12

1	2
bool PasswordValidation(string Password)	Валідація паролю
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.13 Клас LobbyMember

Ціль класу полягає у збереженні даних про учасників лобі. Дані класу наведені у таблиці 3.13.

Таблиця 3.13 – Опис полів класу LobbyMember

Поля структури	Опис
1	2
public:	
public string Name	Зміна з назвою
public string Status	Зміна з статусом

3.3.14 Клас BlackjackException

Ціль класу полягає у обробці виняткових ситуацій на сервері. Дані та методи класу наведені у таблиці 3.14.

Таблиця 3. 14 – Опис полів та методів класу BlackjackException

Поля та методи класу	Опис
1	2
public:	
public BlackjackException(int code)	Конструктор класу, в якому обирається помилка

3.3.15 Клас DataBaseAPI

Ціль класу – реалізація роботи з БД. Дані класу наведені у таблиці 3.15.

Таблиця 3.15 – Опис полів класу DataBaseAPI

Поля структури	Опис
1	2
<i>public:</i>	
public bool LoginIsExists(string Login)	Конструктор вікна
public bool AddUser(string Login, string Password)	
public void DeleteUser(long User_ID)	
public ServerUser Authorization Check(string Login, string Password)	
public string GetLogin(int ID)	
<i>private:</i>	
string ConnectionString	Змінна з шляхом для підключення до БД

3.3.16 Клас Deck

Ціль класу полягає реалізації об'єкту колоди. Дані та методи класу наведені у таблиці 3.16.

Таблиця 3. 16 – Опис полів та методів класу Deck

Поля та методи класу	Опис
1	2
<i>public:</i>	
public Card[] cards	Змінна з списком карт

3.3.17 Клас Card

Ціль класу полягає у реалізації об'єкту карти. Дані та методи класу наведені у таблиці 3.17.

Таблиця 3. 17 – Опис полів та методів класу Card

Поля та методи класу	Опис
1	2
<i>public:</i>	
public string Name	Змінна назви карти
public string Suit	Змінна масті карти
public int Cost { get; set; }	Змінна ціни карти в балах

3.3.18 Клас Game

Ціль класу полягає у реалізації об'єкту гри. Дані та методи класу наведені у таблиці 3.18.

Таблиця 3. 18 – Опис полів та методів класу Game

Поля та методи класу	Опис
1	2
<i>public:</i>	
public Lobby GameLobby	Змінна з інформацією про лобі
public Deck CardDeck	Змінна з колодою
public int[] Players	Змінна з списком ид гравців
public int[] Result	Змінна з результатами гри
public Card[] Player1Hand	Змінна з рукою гравця номер 1
public Card[] Player2Hand	Змінна з рукою гравця номер 2
public Card[] Player3Hand	Змінна з рукою гравця номер 3

Продовження таблиці 3.18

1	2
public Card[] Player4Hand	Змінна з рукою гравця номер 4
public Card[] DillerHand	Змінна з рукою ділера
public int TurnIndex	Змінна з індексом того хто ходить
public bool EndGame	Флаг що позначає чи треба завершити гру

3.3.19 Клас Lobby

Ціль класу полягає у реалізації об'єкту лобі. Дані та методи класу наведені у таблиці 3.19.

Таблиця 3. 19 – Опис полів та методів класу Lobby

Поля та методи класу	Опис
1	2
public:	
public int MaxNumOfPlayer	Змінна з максимальною кількістю учасників лобі
public int [] Player_ID	Змінна з списком ид учасників лобі
public string[] Player_Status	Змінна з статусом гравців
public string Name	Змінна з назвою лобі
public string Description	Змінна з описом лобі
public string PlayerCountString	Змінна з кількістю гравців

3.3.20 Клас ServerUser

Ціль класу полягає у реалізації об'єкту користувача. Дані та методи класу наведені у таблиці 3.20.

Таблиця 3.20– Опис полів та методів класу ServerUser

Поля та методи класу	Опис
1	2
<i>public:</i>	
public int ID	Змінна з ид користувача
public string Login	Змінна з логіном користувача
public OperationContext operationContext	Змінна з контекстом операції

3.3.21 Клас ServiceBlackjack

Ціль класу полягає у реалізації функціоналу сервера. Дані та методи класу наведені у таблиці 3.21.

Таблиця 3.21– Опис полів та методів класу ServiceBlackjack

Поля структури	Опис
1	2
<i>public:</i>	
public bool IsConnect(string Login)	Перевірка чи вже авторизований користувач з таким Login
public int Connect(string Login, string Password)	Авторизація, повертає ID користувача якщо авторизація успішна, і 0 якщо ні
public void Disconnect(int ID)	Вихід з акаунту
public bool Registration(string Login, string Password)	Реєстрація, додавання акаунту до бази даних
public void CreateLobby(Lobby lobby)	Створення лобі на сервері
public bool ConnectLobby(Lobby lobby, int UserID)	Підключення до лобі

Продовження таблиці 3.21

1	2
public void DisconnectLobby(Lobby lobby, int UserID)	Відключення від лобі
public void DelLobby(Lobby lobby)	Видалення лобі на сервері
public List<Lobby> GetListLobby()	Повернення списку активних лобі
public string GetLogin(int ID)	Повернення Login користувача за його ID
public void UpdateStatus(Lobby lobby)	Оновити статус гравця у лобі (Готов, не готов)
public void Turn(Game game)	Обробка ходу користувача
private:	
DataBaseAPI DataBase	Змінна для роботи з БД
List<ServerUser> users	Змінна що зберігає активних користувачів
List<Lobby> lobbys	Змінна що зберігає активні лобі

3.3.22 Інтерфейс IServiceBlackjack

Ціль інтерфейсу полягає контрактів які має виконувати клас що реалізує функціонал сервера. Дані та методи класу наведені у таблиці 3.22.

Таблиця 3.22– Опис полів та методів класу IServiceBlackjack

Поля структури	Опис
1	2
public:	
bool IsConnect(string Login)	Перевірка чи вже авторизований користувач з таким Login

Продовження таблиці 3.22

1	2
int Connect(string Login, string Password)	Авторизація, повертає ID користувача якщо авторизація успішна, і 0 якщо ні
void Disconnect(int ID)	Вихід з акаунту
bool Registration(string Login, string Password)	Реєстрація, додавання акаунту до бази даних
void CreateLobby(Lobby lobby)	Створення лобі на сервері
bool ConnectLobby(Lobby lobby, int UserID)	Підключення до лобі
void DisconnectLobby(Lobby lobby, int UserID)	Відключення від лобі
void DelLobby(Lobby lobby)	Видалення лобі на сервері
List<Lobby> GetListLobby()	Повернення списку активних лобі
string GetLogin(int ID)	Повернення Login користувача за його ID
void UpdateStatus(Lobby lobby)	Оновити статус гравця у лобі (Готов, не готов)
void Turn(Game game)	Обробка ходу користувача, відправка про нього інформації всім учасникам гри

3.3.23 Інтерфейс IServiceBlackjackCallback

Ціль інтерфейсу полягає контрактів які має виконувати клас що реалізує функціонал сервера. Дані та методи класу наведені у таблиці 3.23.

Таблиця 3.23– Опис полів та методів класу IServiceBlackjackCallback

Поля структури	Опис
1	2
public:	
void LobbyCallback(Lobby lobby)	Оновлення активного лобі
void ListLobbyCallback(List<Lobby> lobbys)	Оновлення списку лобі
void StartGameCallback(Game game)	Оновлення початої гри
void GameCallback(Game game)	Оновлення підчас гри

3.4 Проектування бази даних

Концептуальна схема БД для застосунку «Blackjack» представлена на рисунку 3.13.

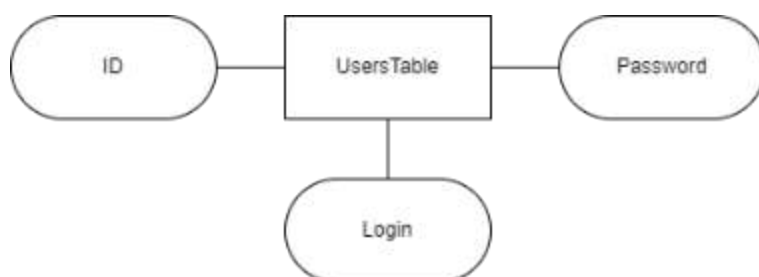


Рисунок 3.13 – Концептуальна модель бази даних

Сутність «UserTable» (ID, Login, Password) – опис полів сутності та їх типу наведено у таблиці 3.24.

Таблиця 3.24 – Опис полів сутності «Recipe»

№	Назва поля	Тип даних	Розмір
1	ID	int	4 байти
2	Login	nchar(20)	40 байтів
3	Password	nchar(50)	100 байтів

Але не всі дані зберігаються у БД, така інформація як списки підключених до серверу користувачів та активних лобі зберігаються у буфері серверу. Зберігання даних у буфері сервера дозволяє зменшити час доступу до них та зменшити навантаження на базу даних, також це підвищує безпеку даних, бо усі тимчасові дані видаляються з буферу при завершенні роботи серверу.

3.5 Основні рішення щодо розробки інтерфейсу

Є кілька принципів, які важливо мати на увазі під час розробки інтерфейсу користувача для дизайну інтерфейсу користувача відеоігор. Ось деякі з ключових принципів ігрового дизайну інтерфейсу користувача [29]:

- **чіткість:** інтерфейс користувача має бути зрозумілим і легким для розуміння. Це означає використання чіткої та лаконічної мови, простої графіки та інтуїтивно зрозумілих значків ігрового інтерфейсу, щоб полегшити гравцям навігацію та розуміння;
- **узгодженість:** інтерфейс користувача має бути узгодженим у всьому ігровому просторі зі стандартним макетом, кольоровою схемою та типографікою;
- **зворотній зв'язок:** інтерфейс користувача повинен надавати гравцеві зворотний зв'язок із зазначенням того, які дії було вжито та які наслідки. Цей зворотній зв'язок може бути наданий за допомогою візуальних або звукових сигналів, таких як звукові ефекти або анімація;
- **чуйність:** інтерфейс користувача має реагувати на дії гравця з мінімальною затримкою. Це допомагає створити безперебійний досвід, який відчуває себе інтуїтивно зрозумілим і простим у використанні;
- **доступність:** інтерфейс користувача має бути доступним для всіх гравців, у тому числі для людей з обмеженими можливостями. Це означає розробку інтерфейсу користувача зі спеціальними можливостями, такими як

параметри високої контрастності, великий текст і альтернативні методи введення;

- простота: інтерфейс користувача має бути простим і оптимізованим, без зайвих елементів або безладу. Це допомагає зробити гру легкою для навчання та гри, а також зменшує зорове та когнітивне перевантаження.

При розробці інтерфейсу програмного рішення «Blackjack» були використані ці принципи.

Намальований від руки 2D ігровий стиль забезпечує баланс між унікальністю та глибиною, звертаючись безпосередньо до емоцій гравців. Стили варіюються від затишних до моторошних, узгоджуючи їх із сюжетом кожної гри. Цей підхід покращує оповідання і додає людський відтінок цифровим сферам, роблячи кожну сцену особливо художньою [30].

Саме цей стиль було використано для однієї з тем інтерфейсу гри «Blackjack». Ручні малюнки додають грі унікальності та відрізняють її від багатьох інших цифрових ігор. Це створює відчуття ручної роботи і художньої виразності, що може бути дуже привабливим для гравців. Ручні малюнки можуть асоціюватися з традиційними картковими іграми, додаючи відчуття автентичності та класичності, що особливо доречно для такої гри, як блекджек. Також такий стиль легко адаптувати до різних тем та змін у дизайні, що робить його гнучким і зручним для оновлень.

Для привернути увагу було використано ретровейв (retrowave) або кіберпанк (cyberpunk) стиль. Ретро-футуристичні елементи та неонові кольори привертають увагу і створюють ефектний вигляд. Це може зробити гру більш помітною та запам'ятовуваною для користувачів. Також Кіберпанк і ретровейв є популярними жанрами в сучасній культурі, включаючи ігри, фільми та музику. Це робить гру привабливою для молодшої та сучасної аудиторії. Але в той же час ретровейв стиль викликає ностальгію за 80-ми роками, що може привабити старших гравців, які пам'ятають ці часи.

Інтерфейс програми реалізовано за допомоги двох фалів ResourceDictionary.xaml та ResourceDictionaryAlternative.xaml. Саме в цих

файлах для кожного елемента інтерфейсу було прописано свій стиль та кольорову палітру. Палітра кольорів для світлої теми програмного застосунку «Blackjack» вказана на рисунку 3.14. Цю тему описано в файлі ResourceDictionary.xaml.

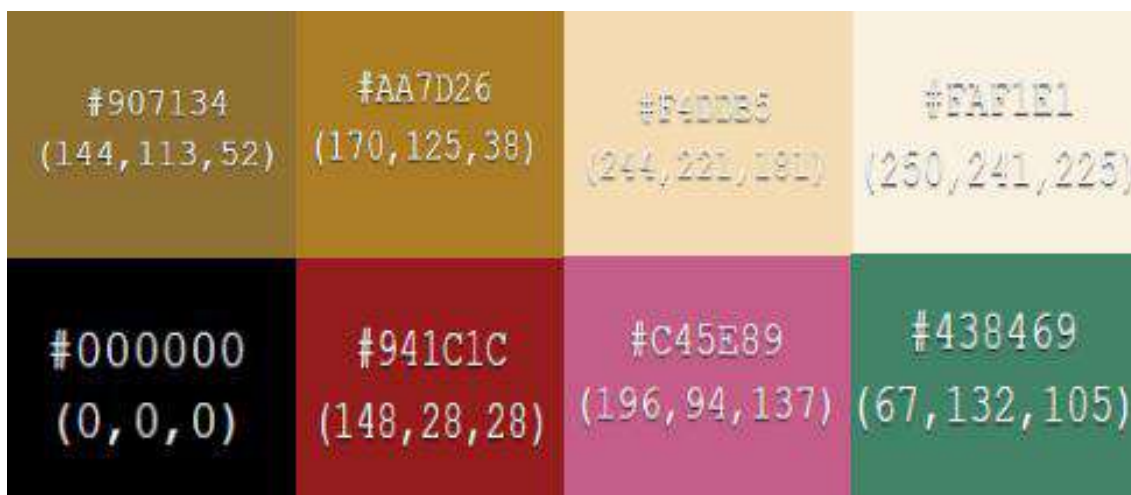


Рисунок 3.14 – Палітра кольорів для світлої теми

Кольорова гама тепла і приємна, що створює дружню атмосферу. Використовуються природні кольори, такі як бежевий, золотий, червоний та чорний.

Кремові відтінки – спокійні, елегантні, чисті. Вони підійдуть для фону. Темно-бежеві відтінки – консервативні, символізують шанобливість та благочестя.

Комусь вони можуть здатися нудними, а когось заспокоюватимуть, тому на них слід зупинити свій вибір, якщо хочеться зробити фон не надто сміливим. Це якраз підходить для гри тому, що основна увага користувачів повинна бути на картах.

Використання кольорів, які нагадують реальні малюнки створює відчуття, що інтерфейс намальований вручну.

Для малювання карт, фішок та фонів використовувалися растрові зображення. Це було обрано через те, що растрові зображення найкращим чином передають відтінки кольорів, їх переходи один в одного і тіні з

максимальною достовірністю [31]. На рисунку 3.15 зображені намальовані карти в сервісі для графічного дизайну «Canva».

Для гри в Blackjack потрібні були такі карти: від 2 до 10, валет, королева, король, туз, джокер. Основними кольорами були: чорний, червоний та білий.

Намальовані карти мають мінімалістичний дизайн з читабельними цифрами та текстом. Також всі вони створені в однаковому стилі, що забезпечує цілісність дизайну.

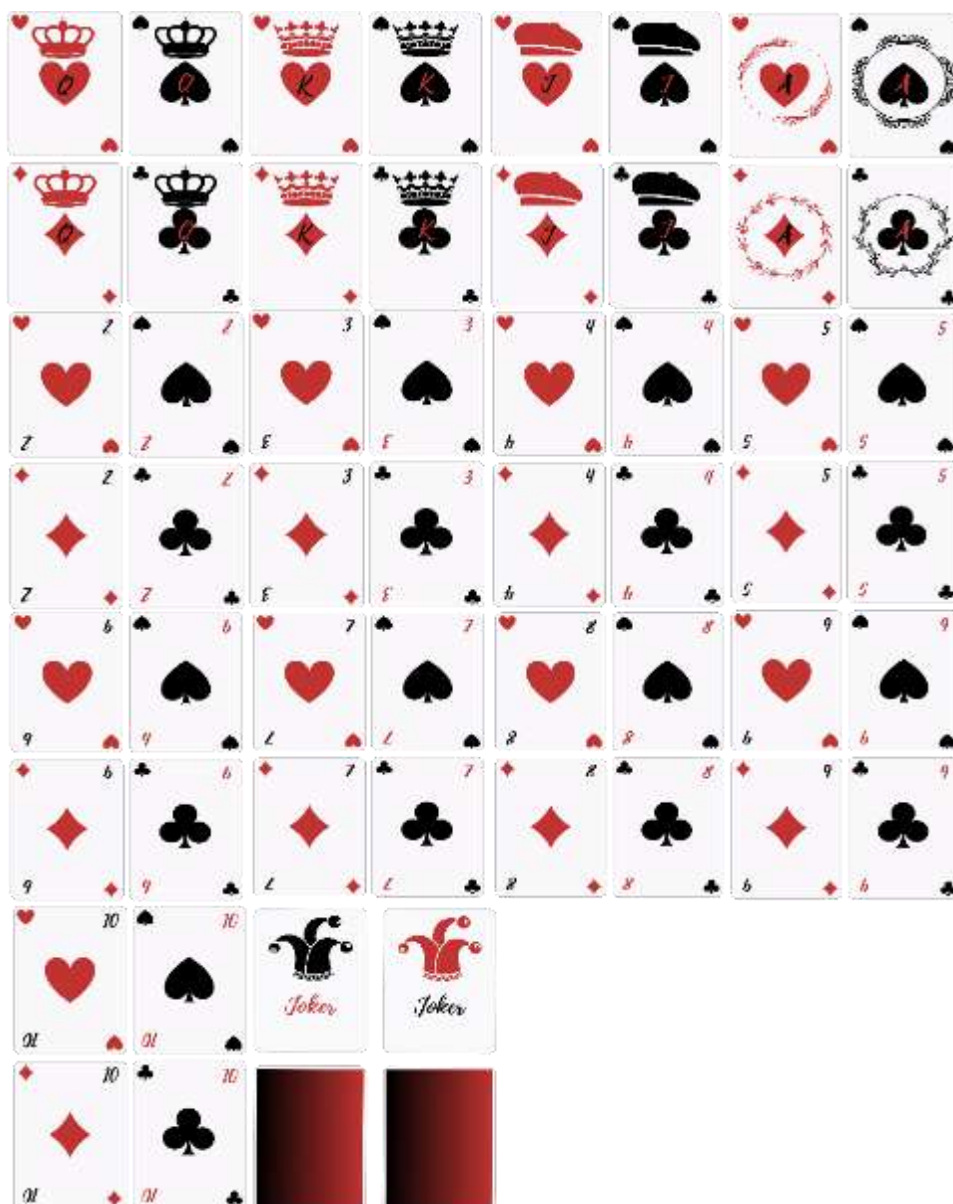


Рисунок 3.15 – Створені карти

На рисунку 3.16 зображено вікно з авторизацією. Фон для цього вікна було створено окремо в сервісі «Canva».

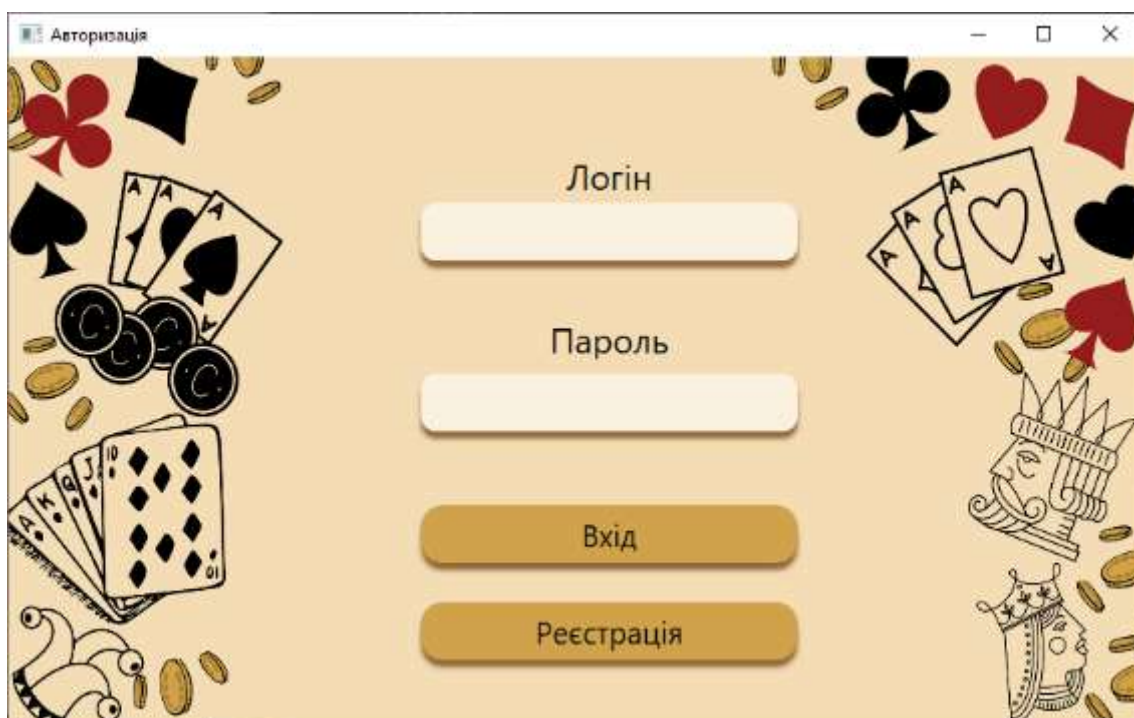


Рисунок 3.16 – Вікно з авторизацією

Для текстових полів використано світлі відтінки в відповідній до фону кольоровій гамі.

Приклад коду для налаштування TextBox

```
<Style TargetType="TextBox">
  <Setter Property="FontFamily" Value="Segoe UI" />
  <Setter Property="Foreground" Value="Black" />
  <Setter Property="FontSize" Value="24" />
  <Setter Property="Background" Value="#faf1e1"/>
  <Setter Property="BorderThickness" Value="0" />
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="TextBox">
        <Border Background="{TemplateBinding Background}"
          BorderBrush="{TemplateBinding BorderBrush}"
          BorderThickness="{TemplateBinding BorderThickness}"
          CornerRadius="10">
          <Border.Effect>
            <DropShadowEffect Color="#AA7D26" Direction="270"
              ShadowDepth="5" Opacity="0.7"/>
          </Border.Effect>
          <ScrollViewer x:Name="PART_ContentHost" />
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

```
</Setter>
</Style>
```

Для головного вікна, що на рисунку 3.17 було використано той самий фон. Для кнопок використано закруглення кутів, щоб гра виглядала плавно та приємно для очей користувачів.

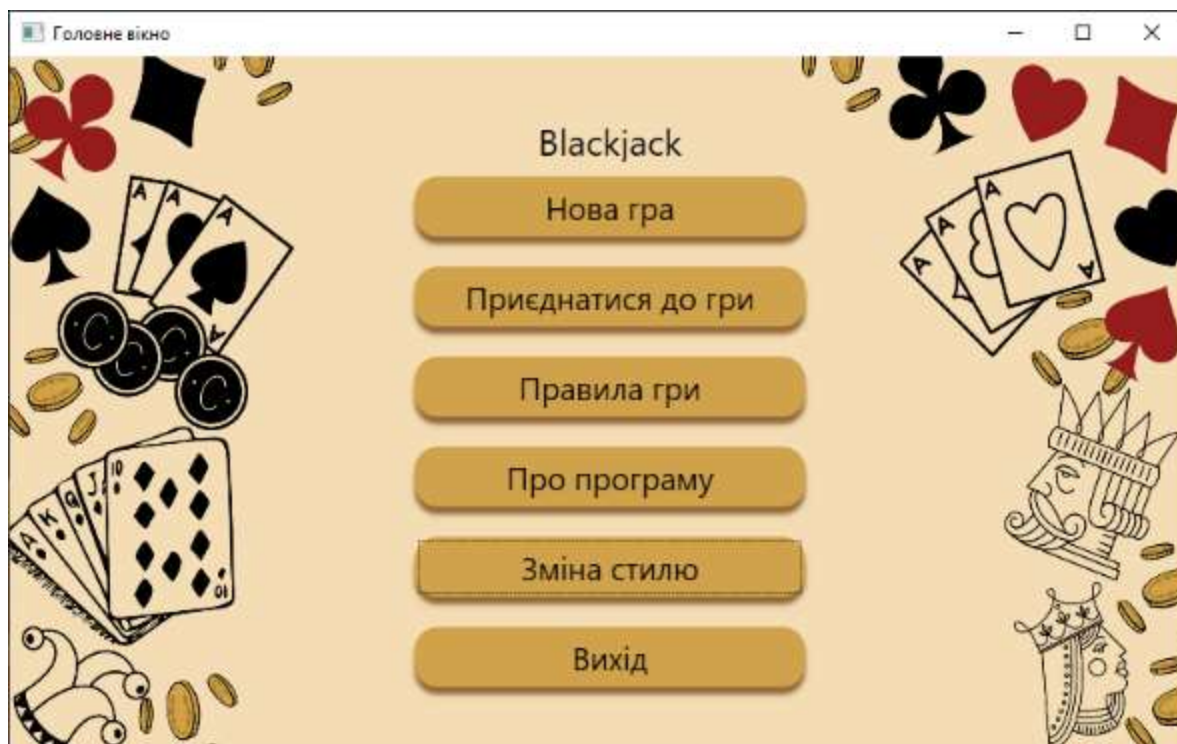


Рисунок 3.17 – Головне вікно

Приклад коду для налаштування закруглення та зміни кольору при наведенні TextBox

```
<Setter Property="Template">
  <Setter.Value>
    <ControlTemplate TargetType="Button">
      <Border Background="{TemplateBinding Background}"
              CornerRadius="15">
        <Border.Effect>
          <DropShadowEffect Color="#AA7D26" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
        </Border.Effect>
        <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
      </Border>
      <ControlTemplate.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
          <Setter Property="Background" Value="#907134"/>
        </Trigger>
      </ControlTemplate.Triggers>
```

```
</ControlTemplate>  
</Setter.Value>  
</Setter>
```

Також створено вікно для реєстрації, що показано на рисунку 3.18.

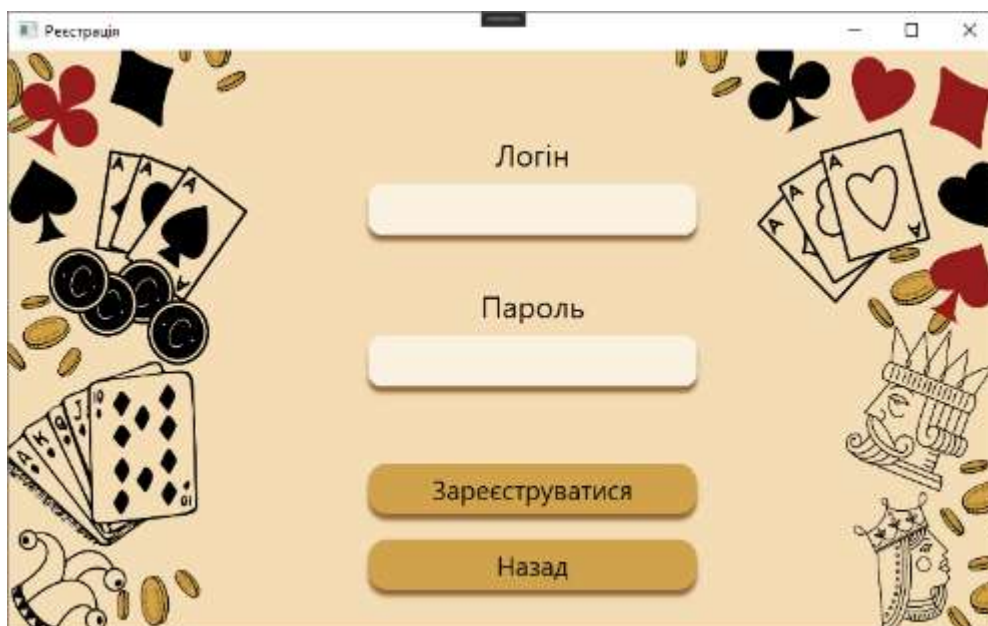


Рисунок 3.18 – Реєстрація

Для створення лобі реалізоване вікно на рисунку 3.19

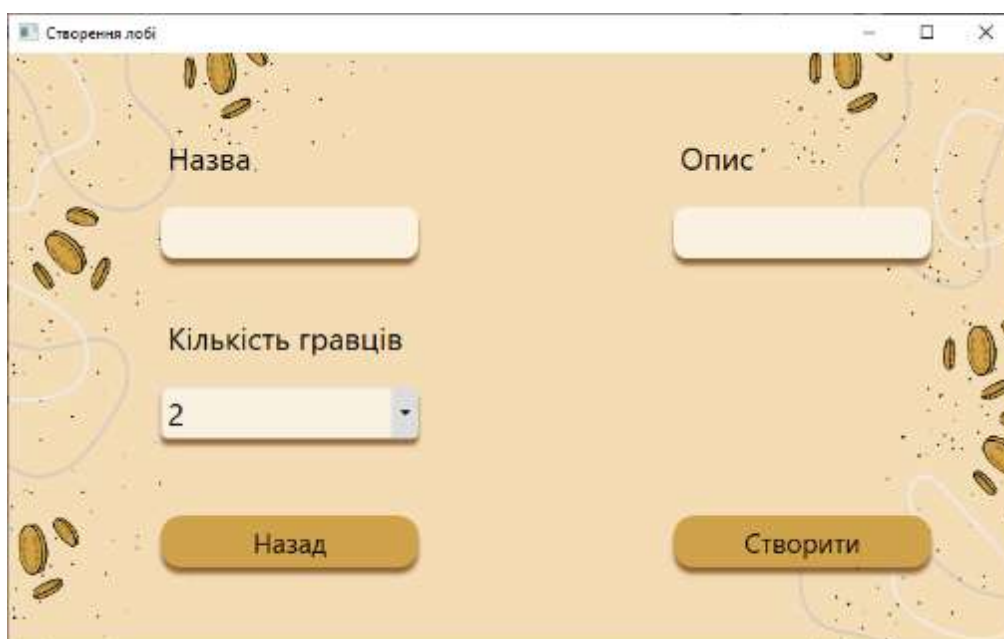


Рисунок 3.19 – Вікно для створення лобі

Приклад коду для налаштування ComboBox

```
<Style TargetType="ComboBoxItem">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#faf1e1"/>
</Style>
```

На рисунку 3.20 зображено вікно для пошуку лобі. Фон для нього та вікна на рисунку 3.19 було створено окремо в сервісі «Canva».



Рисунок 3.20 – Вікно для пошуку лобі

Для лобі, де всі учасники очікують початок гри, створено окреме вікно, що зображено на рисунку 3.21. Це вікно має схожий за кольоровою палітрою фон, але з додатковою картинкою всередині. Цей фон було також намальовано окремо в сервісі «Canva».

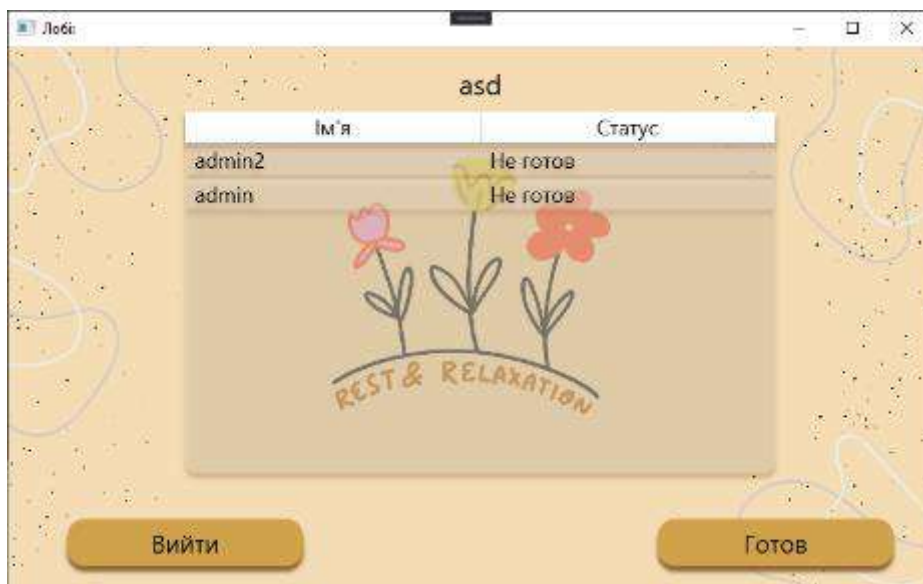


Рисунок 3.21 – Вікно з очікуванням

Для того щоб фонове зображення було видно елементи ListView мають прозорий фон.

Приклад коду з прозорим фоном для ListView:

```
<Setter Property="FontFamily" Value="Segoe UI" />
<Setter Property="Foreground" Value="Black" />
<Setter Property="FontSize" Value="18" />
<Setter Property="Background" Value="#50faf1e1"/>
```

Саме вікно, де відбувається ігровий процес, показано на рисунок 3.22.

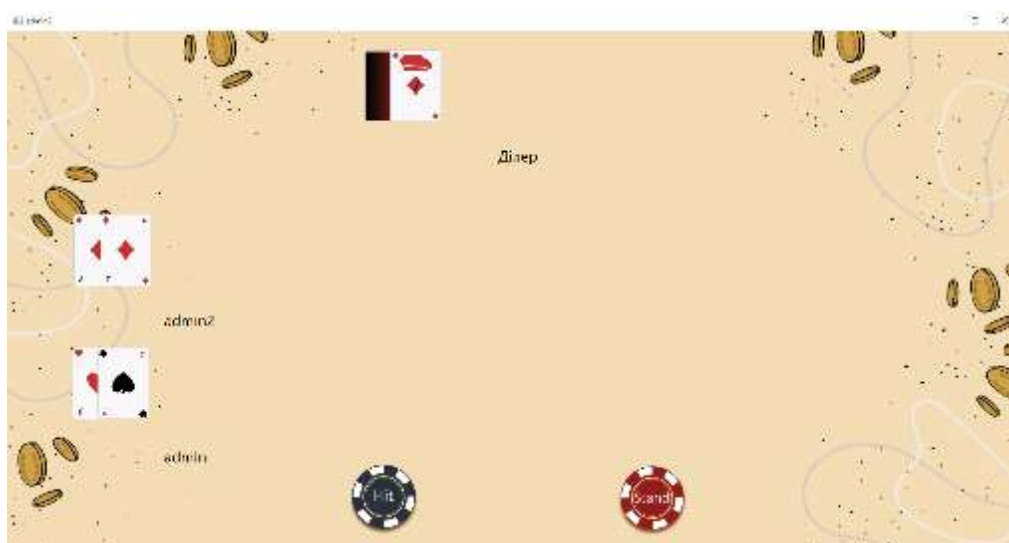


Рисунок 3.22 – Вікно з ігровим процесом

Це вікно має дві кнопки у вигляді фішок: Hit, Stand. Ці кнопки мають свій власний дизайн, що відокремлює їх від інших. Так користувачам буде зрозуміло, що ці кнопки мають лише суто ігровий функціонал.

Приклад для налаштування незвичайних кнопок-фішок

```
<Style x:Key="ButtonRound" TargetType="Button">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="Background" Value="#faf1e1"/>
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="21" />

    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border Background="{TemplateBinding Background}"
                    CornerRadius="90">
                    <Border.Effect>
                        <DropShadowEffect Color="#AA7D26" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
                    </Border.Effect>
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            <ControlTemplate.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="#907134"/>
                </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
```

Після кінця гри користувачу виводиться окреме вікно з результатами. Можливі результати представлені на рисунку 3.23.

Можливі варіанти результату:

- перемога;
- програш;
- нічия.

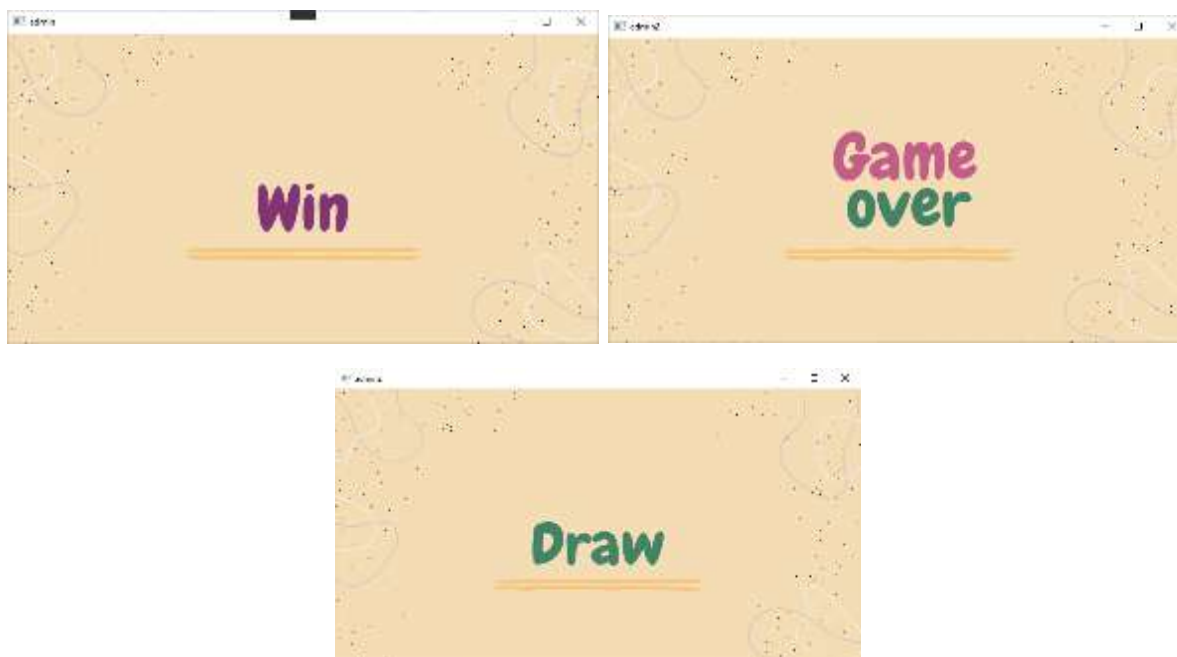


Рисунок 3.22 – Варіанти результату гри

Також було створено ще два довідкових вікно: «Правила гри» та «Інформація про програму». Їх можна побачити на рисунку 3.23.

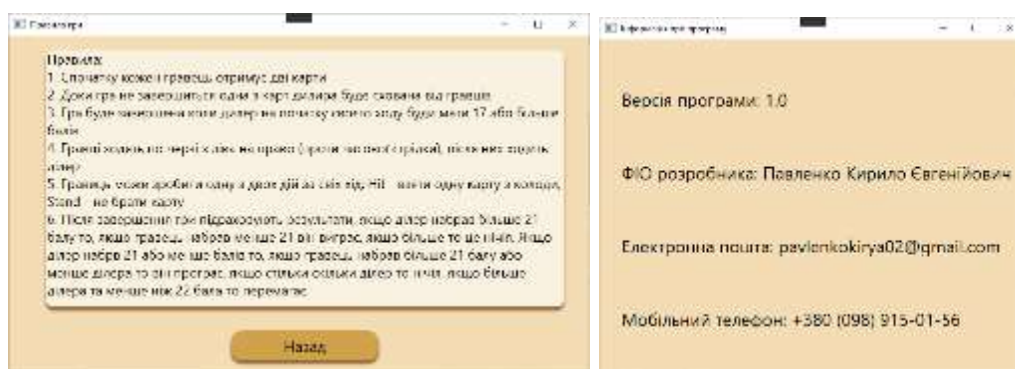


Рисунок 3.23 – Довідкові вікна

Для іншої теми створено окремий файл ResourceDictionaryAlternative.xaml та окрему кольорову палітру. Її можна побачити на рисунку 3.24 [32].



Рисунок 3.24 – Кольорова палітра для темної теми [32]

Приклад коду для зміни теми:

```
private void StyleUpdate()
{
    string newSource;
    if (StandartStyle)
    {
        newSource = "ResourceDictionaryStandart.xaml";
        this.Background = SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFF4DDB5"));
    }
    else
    {
        newSource = "ResourceDictionaryAlternative.xaml";
        this.Background = SolidColorBrush((Color)ColorConverter.ConvertFromString("#2c0540"));
    }

    // Отримуємо поточний ResourceDictionary
    var appResources = this.Resources;
    // Отримуємо MergedDictionaries
    var mergedDictionaries = appResources.MergedDictionaries;
    // Створюємо новий ResourceDictionary
    var newDictionary = new ResourceDictionary();
    newDictionary.Source = new Uri(newSource,
    UriKind.RelativeOrAbsolute);
    // Замінюємо поточний ResourceDictionary на новий
    mergedDictionaries.Clear();
    mergedDictionaries.Add(newDictionary);
}
```

Головне вікно з іншою темою можна побачити на рисунку 3.25. Фон зображає міський пейзаж з неоновими вогнями та сіткою на передньому плані,

що нагадує візуальний стиль 80-х років. Цей фон було створено в сервісі Canva.

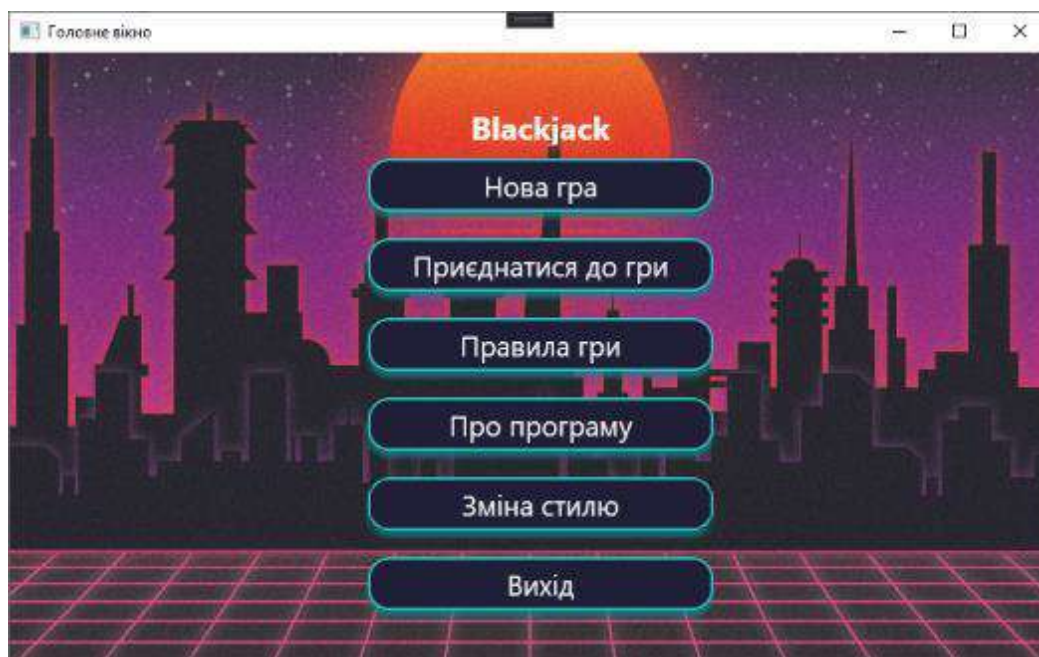


Рисунок 3.25 – Головне вікно з темною темою

Кнопки та текст мають неонові підсвічування, що характерно для ретровейв та кіберпанк стилів. Це додає глибини і підкреслює естетичний вигляд інтерфейсу.

Приклад коду налаштування дизайну елементів Button

```
<Style TargetType="Button">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="Background" Value="#211f38"/>
    <Setter Property="Foreground" Value="#f7f7f7" />
    <Setter Property="FontSize" Value="21" />

    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border x:Name="border"
                    Background="{TemplateBinding Background}"
                    BorderBrush="#2ec2c2"
                    BorderThickness="2"
                    CornerRadius="15">
                    <Border.Effect>
                        <DropShadowEffect Color="#2ec2c2" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
                    </Border.Effect>
                </Border>
                <ContentPresenter
                    HorizontalAlignment="Center"
                    VerticalAlignment="Center"/>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

```

</Border>
<ControlTemplate.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
        <Setter TargetName="border" Property="Background"
Value="#249B9B"/>
    </Trigger>
</ControlTemplate.Triggers>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

```

Вікна для створення лобі та пошуку лобі можна побачити на рисунку 3.26. Ці вікна мають однаковий фон, який було створено в сервісі Canva.

Домінуючі темні кольори створюють контраст з яскравими неоновими елементами, що робить інтерфейс виразним та ефектним.



Рисунок 3.26 – Вікна для створення та пошуку лобі з темною темою

Приклад коду дизайну елементів TextBox

```
<Style TargetType="TextBox">
  <Setter Property="FontFamily" Value="Segoe UI" />
  <Setter Property="Foreground" Value="#f7f7f7" />
  <Setter Property="FontSize" Value="24" />
  <Setter Property="Background" Value="#2c0540"/>
  <Setter Property="BorderThickness" Value="2" />
  <Setter Property="BorderBrush" Value="#ff70c8" />
  <!-- Закруглення -->
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="TextBox">
        <Border Background="{TemplateBinding Background}"
          BorderBrush="{TemplateBinding BorderBrush}"
          BorderThickness="{TemplateBinding BorderThickness}"
          CornerRadius="10">
          <Border.Effect>
            <DropShadowEffect Color="#ff70c8" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
          </Border.Effect>
          <ScrollViewer x:Name="PART_ContentHost" />
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Для вікна з лобі фон відрізняється, але футуристична сітка залишилась, щоб поєднувати дизайн у загальну концепцію стилю темної теми. Фон також було створено у сервісі Canva. Це вікно показано на рисунку 3.27.



Рисунок 3.27 – Вікно з лобі з темною темою

Вікно з грою, яке зображено на рисунку 3.28 має звичайний футуристичний дизайн, де увага гравців прикута до самих карт. Сітка на передньому плані нагадує сцени з науково-фантастичних фільмів або відеоігор, що створює відповідний настрій для гри.

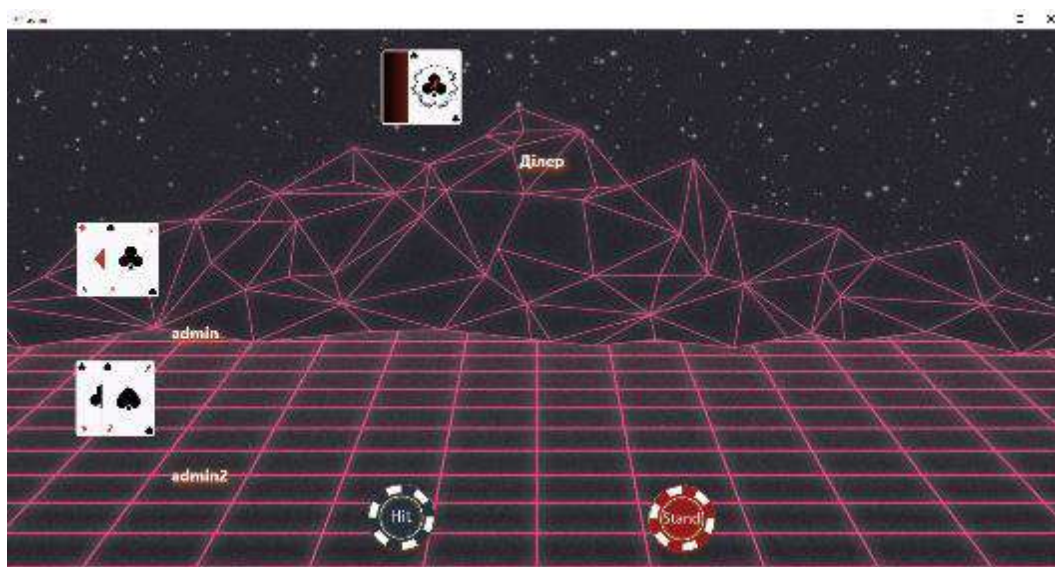


Рисунок 3.28 – Вікно з грою з темною темою

Результати гри з темною темою відображені у трьох вікнах на рисунку 3.29. Це: перемога, нічия та програш.

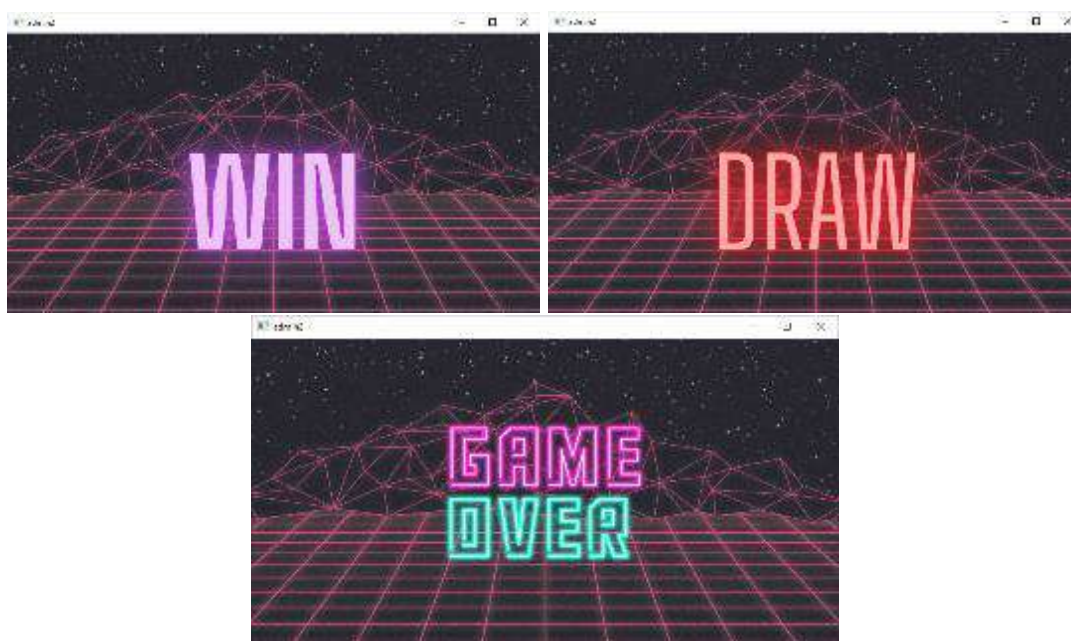


Рисунок 3.29 – Результати гри з темною темою

Довідкові вікна зображено на рисунку 3.30. Їх дизайн є мінімалістичним, щоб не навантажувати користувачів зайвими елементами.

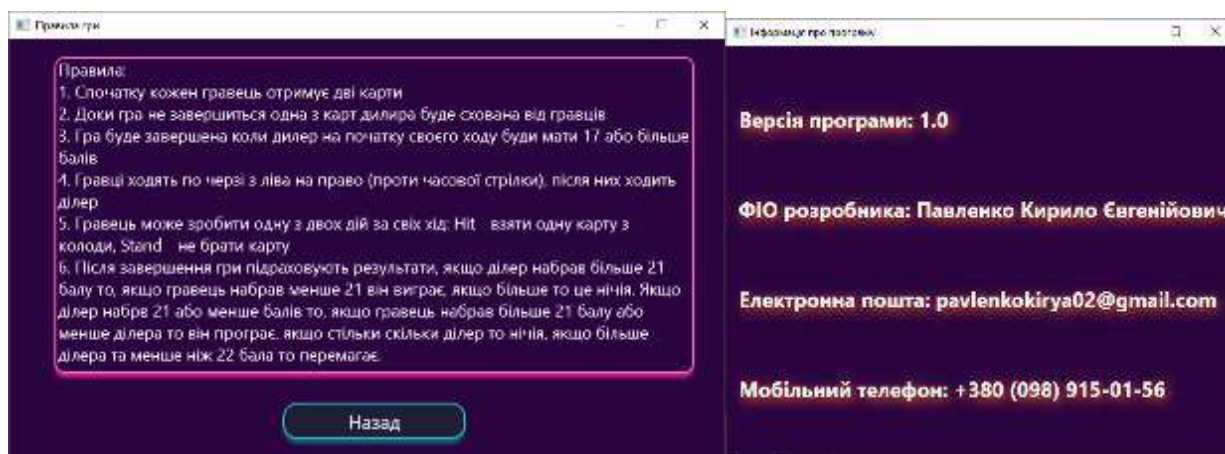


Рисунок 3.30 – Довідкові вікна з темною темою

Приклад коду для дизайну елементів Label

```
<Style TargetType="Label">
    <Setter Property="Foreground" Value="#f7f7f7"/>
    <Setter Property="FontFamily" Value="Segoe UI"/>
    <Setter Property="FontSize" Value="20"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="Effect">
        <Setter.Value>
            <DropShadowEffect BlurRadius="20" Color="#ffa05c"
ShadowDepth="5"/>
        </Setter.Value>
    </Setter>
</Style>
```

3.6 Висновки за розділом 3

Отже в результаті були прийняті основні рішення що до алгоритмів, модулів, створених для їх реалізації класів, основних сутностей в БД, шаблону проєктування та інтерфейсу.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

В даному розділі розглянуто призначення програми, її умови застосування, функції, характеристика, структура та засоби звертання й вхідні та вихідні дані, а також повідомлення які може відобразити застосунок програмісту.

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Програма призначена для надання користувачу ігрового процесу гри, за правилами «Blackjack», шляхом створення мультиплеєрної ігрової сесії.

4.1.2 Функції програми

Функції, що виконує програма:

- реєстрація та авторизація користувачів;
- створення та налаштування ігрового лобі;
- пошук та фільтрація за назвою ігрових лобі;
- підключення до лобі, реалізація лобі та системи готовності гравців;
- реалізація ігрової сесії, ігрового процесу, підрахування результатів гри «Blackjack»;
- надання доступу до правил гри;
- надання доступу до інформації про програмне забезпечення та розробника.

4.1.3 Умови застосування програми для клієнтської та серверної частини

Для коректної роботи з готовим програмним забезпеченням потрібно, щоб пристрій, який буде використовуватися, відповідав мінімальним технічним вимогам:

- потрібна оперативна пам'ять: 4 Гб для 64х та 2 Гб для 32х;
- потрібна місце на жорсткому диску: 400 Мб;
- операційні системи родини Windows 7/8.x/10.

Для компіляції та внесення змін до програми рекомендується використовувати середовище Visual Studio 2019 та платформу .NET Framework 4.7.2.

4.2 Характеристика програми

Програма створена мовою C# у середовищі Microsoft Visual Studio 2019 з підсистемами WPF та WCF.

Застосунок має головне меню з якого можливо перейти у всі інші вікна. Гру можливо розпочати створивши лобі, або підключившись до існуючого.

4.2.1 Структура програми

Щоб працювати з клієнтською частиною програми, потрібно щоб працювала хост-частина. Після запуску хоста користувачі можуть запускати клієнтську частину.

При запуску програми клієнтської частини завантажується вікно з авторизацією.

Загальна структура програми складається з: файлів програмного коду .cs, бібліотек, файлів *.xml та *.config. Структура програми наведена на рисунку 4.1

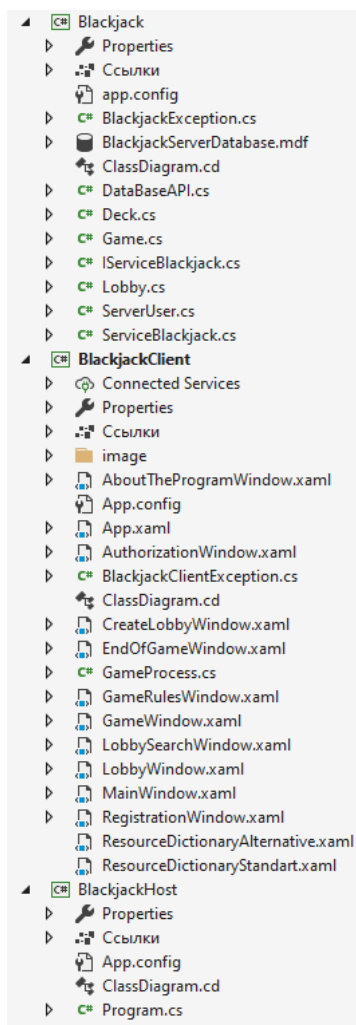


Рисунок 4.1 – Структура програми

4.3 Звертання до програми

Програма має такі методи звертання до неї:

- ліва клавіша комп'ютерної миші (кнопки);
- клавіші клавіатури (ввід назви, опису лобі).

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Програма отримує такі вхідні дані як:

- сигнали про натискання клавіш миші та клавіатури;
- інформацію про обрані налаштування при створенні лобі;

- індекс обраного користувачем елементу списку;
- базу даних;
- зображення карт, кнопок та фонів інтерфейсу.

4.4.2 Вихідні дані

Програма видає такі вихідні дані як:

- інформація про активні лобі;
- інформація про правила гри;
- інформація про лобі в якому знаходиться користувач;
- інформація про ігрову сесію;
- інформація про результат гри.

4.5 Повідомлення

Програма містить такі повідомлення як:

- «LoginIsExists problem» – означає, що виникла помилка при перевірці наявності такого логіну у базі даних;
- «AddUser problem» – означає, що виникла помилка при створенні нового акаунту та внесенні інформації про нього у БД;
- «DeleteUser problem» – означає, що виникла помилка при видаленні акаунту користувача;
- «AuthorizationCheck problem» – означає, що виникла помилка при процесі авторизації;
- «GetLogin problem» – означає, що виникла помилка при отриманні логіну користувача за його ID.

5 КЕРІВНИЦТВО ОПЕРАТОРА

В даному розділі розглянуто призначення програми, умови її виконання й процес її виконання та повідомлення які може відобразити застосунок користувачу.

5.1 Призначення програми

Програма призначена для реалізації ігрового процесу гри, за правилами «Blackjack», шляхом створення мультиплеєрної ігрової сесії.

5.2 Умови виконання програми

5.2.1 Апаратні вимоги програми

Для коректної роботи з готовим програмним забезпеченням потрібно, щоб пристрій, який буде використовуватися, відповідав мінімальним технічним вимогам:

- потрібна оперативна пам'ять: 4 Гб для 64х та 2 Гб для 32х;
- потрібна місце на жорсткому диску: 400 Мб;
- операційні системи родини Windows 7/8.x/10.

5.2.2 Вимоги до операторів

Для комфортної роботи з програмою від користувача потрібні:

- знання та володіння українською мовою;
- навички користування клавіатурою та комп'ютерною мишею;
- ознайомлення з правилами гри.

5.3 Виконання програми

5.3.1 Запуск програми

Програма запускається за допомогою виконуваного файлу BlackjackClient.exe та здійснюється подвійним натисканням лівої клавіші комп'ютерної миші по ньому або його ярлику.

5.3.2 Виконання роботи з програмою

При запуску програми користувач бачить вікно на рисунку 5.1. Для продовження користування програмою треба увійти в свій акаунт.

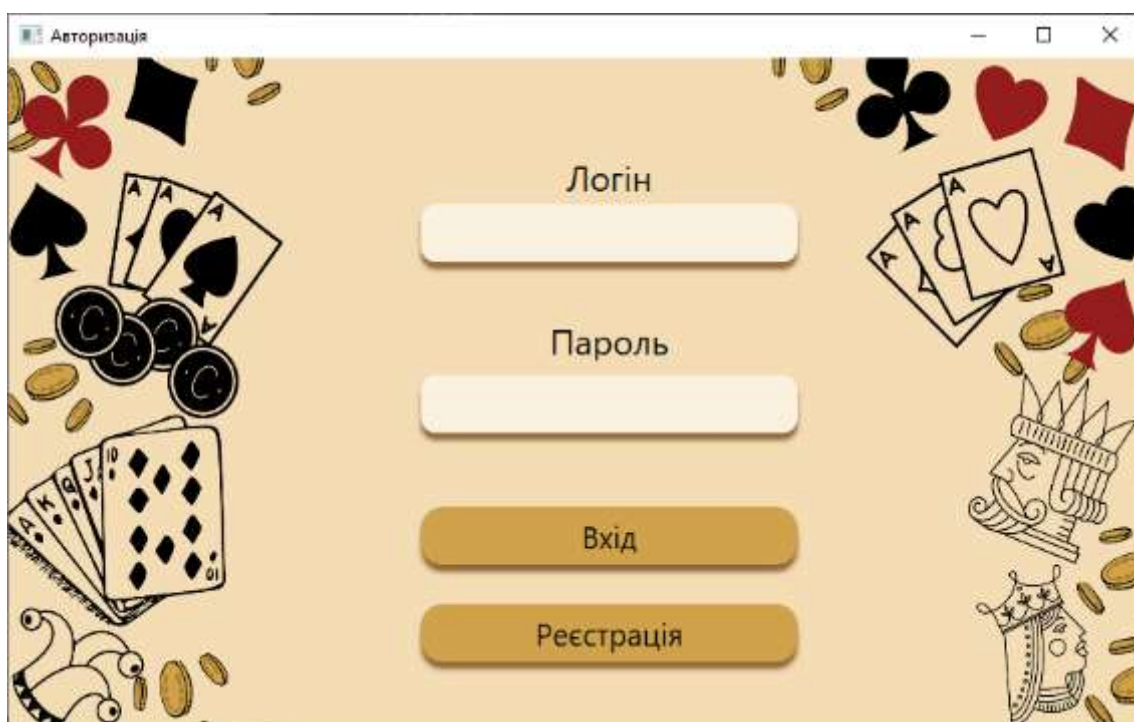


Рисунок 5.1 – Вікно з авторизацією

Якщо користувач немає власного акаунту, то можна його створити. Для цього необхідно натиснути кнопку «Реєстрація». Після чого відобразиться вікно на рисунку 5.2.

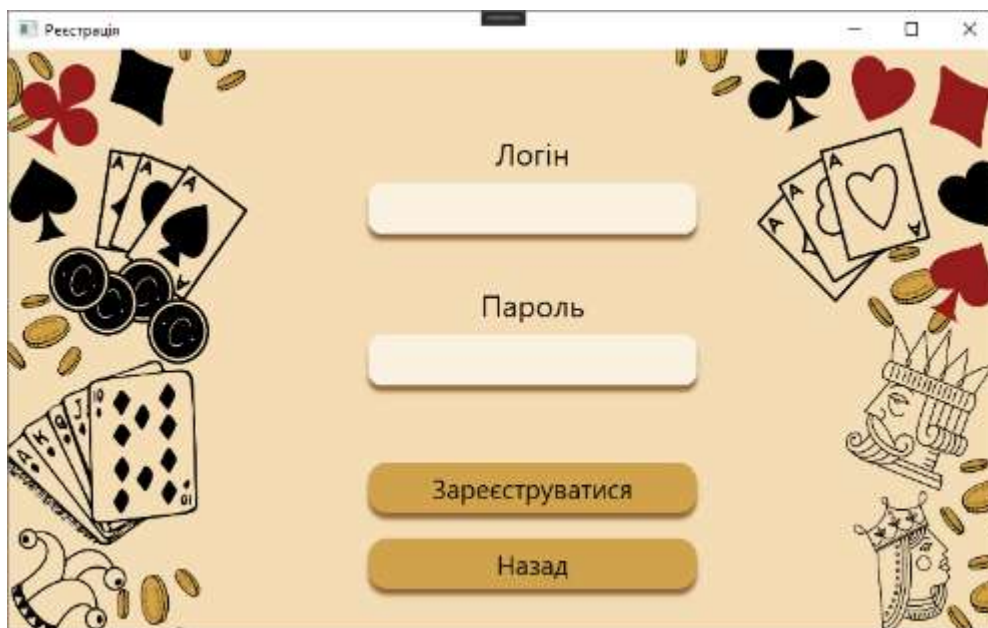


Рисунок 5.2 – Головне вікно

Логін може бути довжиною від 4 до 25 символів та мати у собі цифри та латинські великі та маленькі літери, символи «_» та «.».

Пароль може бути довжиною від 4 до 25 символів та мати у собі цифри та латинські великі та маленькі літери, символи «!";%?'()*+,-. *.».

Після успішної авторизації у акаунт користувач побачить вікно на рисунку 5.3.

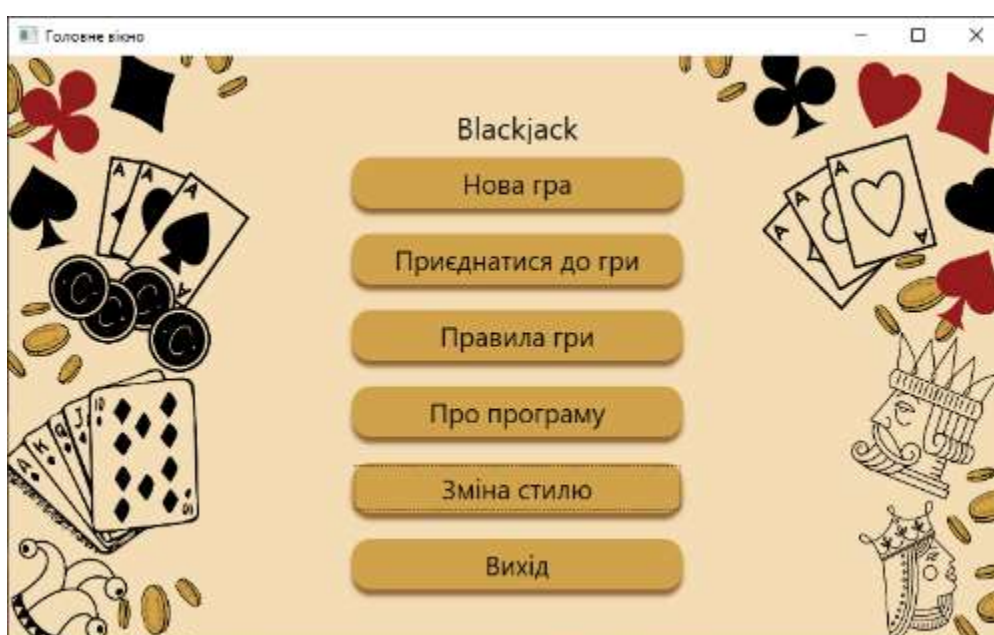


Рисунок 5.3 – Головне вікно

Якщо користувач хоче змінити тему, то треба на головному вікні натиснути на кнопку «Зміна стилю». Після чого головне вікно буде мати вигляд як на рисунку 5.4.



Рисунок 5.4 – Головне вікно з темною темою

Якщо користувач хоче повернути світлу тему, то треба натиснути на кнопку «Зміна стилю». Після чого головне вікно стане як раніше – рисунок 5.3

Для ознайомлення з програмою та його розробником користувач може натиснути кнопку «Про програму». Після чого він побачить вікно на рисунку 5.5

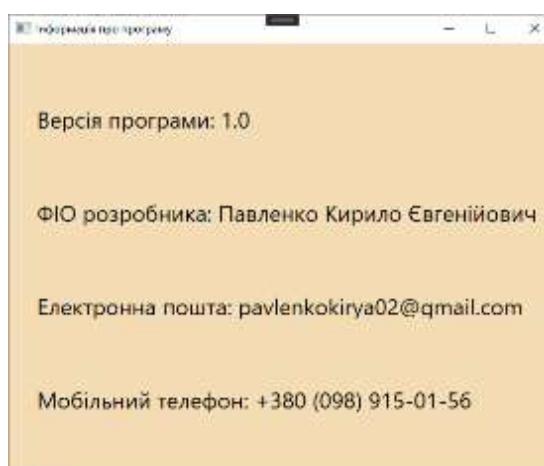


Рисунок 5.5 – Вікно «Про програму»

Якщо користувач не знає правил гри, то може натиснути кнопку «Правила гри», після чого побачить вікно на рисунку 5.6.

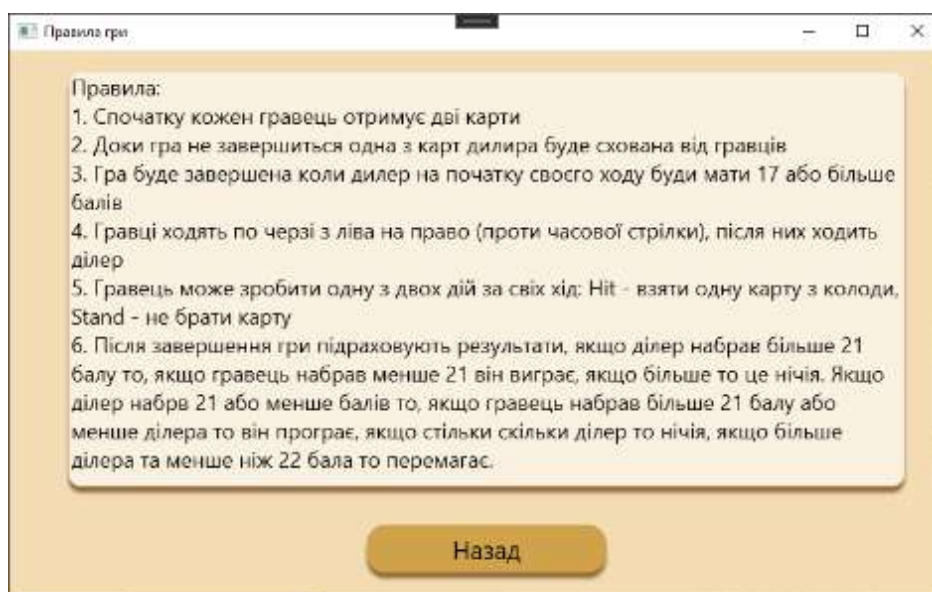


Рисунок 5.6 – Вікно з правилами гри

Є два варіанти як почати гру:

- 1 варіант – створити власне лобі для гри, натиснувши кнопку «Нова гра» на головному вікні;
- 2 варіант – приєднатися до вже існуючого лобі, натиснувши кнопку «Приєднатися до гри» на головному вікні.

При 1 варіанті користувач побачить вікно на рисунку 5.7.

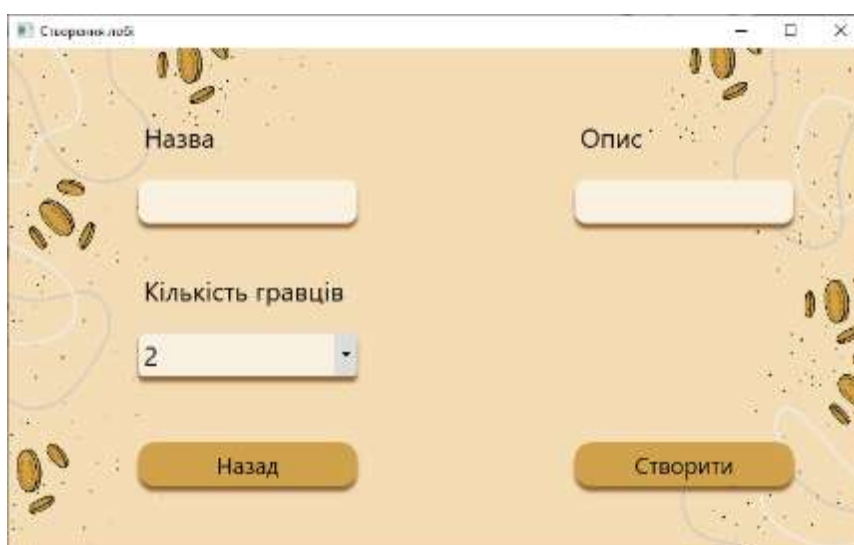


Рисунок 5.7 – Вікно для створення лобі

Тут користувач повинен назвати лобі та описати його декількома словами, також обов'язково повинен вказати кількість гравців від 2 до 4 осіб.

При 2 варіанті користувач побачить вікно на рисунку 5.8.



Рисунок 5.8 – Вікно для пошуку лобі

Тут користувач може шукати лобі по назві або обрати звичайним способом.

Обидва варіанта приводять до вікна на рисунку 5.9.

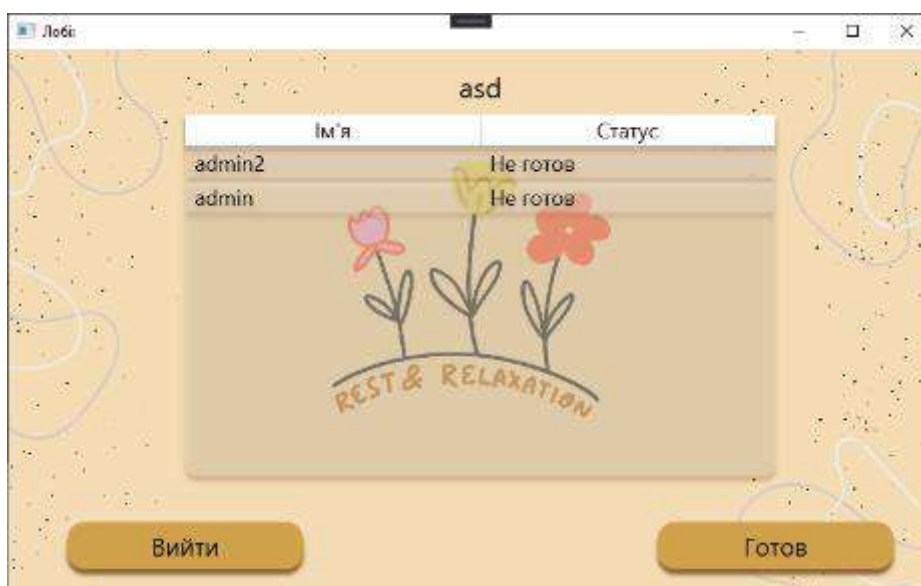


Рисунок 5.9 – Вікно з очікуванням

Тут учасники гри очікують початку гри. Гра почнеться коли обидва учасника натиснуть кнопку «Готов». Після чого гра починається у вікні на рисунку 5.10.

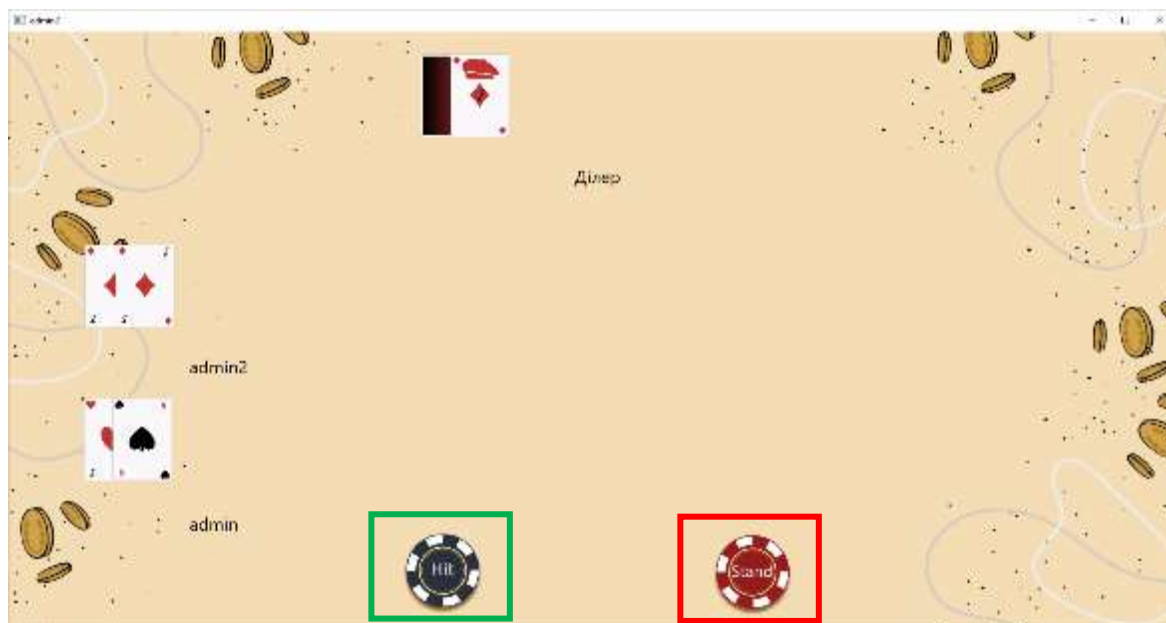


Рисунок 5.10 – Вікно з ігровим процесом

В цьому вікні всі учасники отримують по дві карти. Після цього вони можуть взяти карту, натиснувши на кнопку виділену зеленим прямокутником на рисунку 5.10. А можливо не брати карту, натиснувши на кнопку виділену зеленим прямокутником на рисунку 5.10.

Після кінці гри користувач побачить вікно з результатами гри, що показано на рисунку 5.11.

Можливі варіанти результату гри:

- перемога;
- програш;
- нічия.

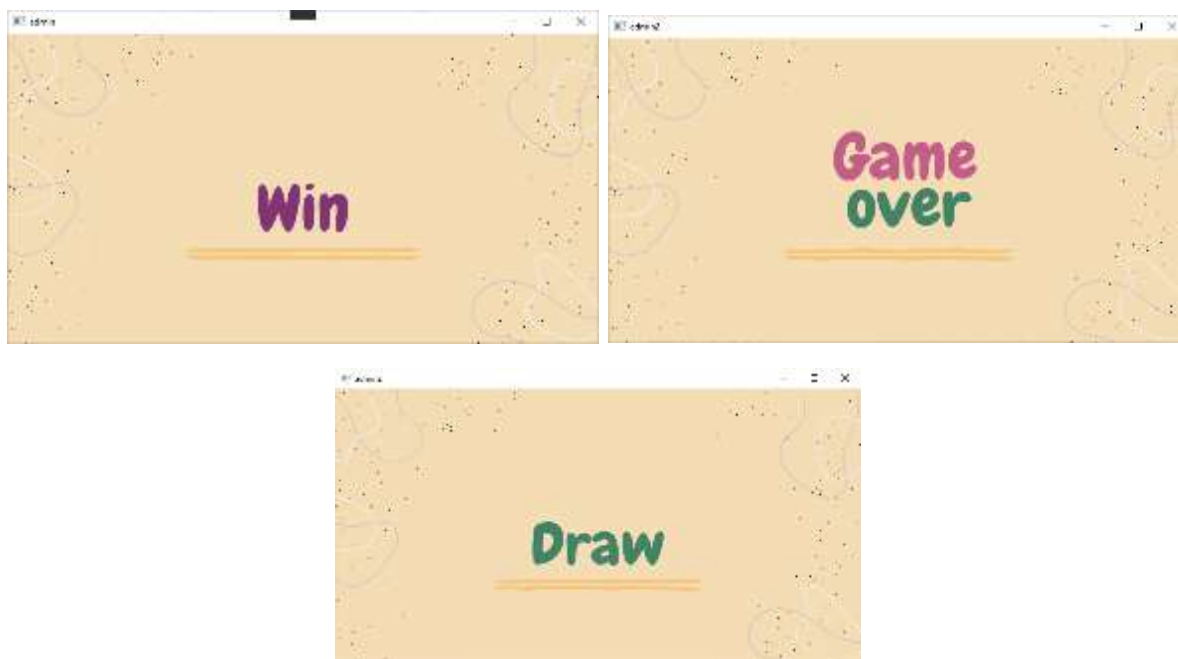


Рисунок 5.11 – Варіанти результату гри

5.4 Повідомлення оператора

Програма містить такі повідомлення як:

- «Помилка авторизації. Нема зв'язку з сервером.» – означає, що вікно авторизації не може зв'язатися з сервером;
- «Помилка реєстрації. Нема зв'язку з сервером.» – означає, що вікно реєстрації не може зв'язатися з сервером;
- «Помилка головного вікна. Нема зв'язку з сервером.» – означає, що вікно з головним меню не може зв'язатися з сервером;
- «Помилка створення лобі. Нема зв'язку з сервером.» – означає, що вікно створення лобі не може зв'язатися з сервером;
- «Помилка створення лобі. Назва зайнята.» – означає, що вікно створення лобі не може створити таке лобі, бо лобі з такою назвою вже існує;
- «Помилка лобі. Нема зв'язку з сервером.» – означає, що вікно лобі не може зв'язатися з сервером;
- «Помилка пошуку лобі. Нема зв'язку з сервером.» – означає, що вікно пошуку лобі не може зв'язатися з сервером;

- «Вхід у лобі невдалий. Лобі заповнене.» – означає, що підключення до лобі не сталося бо в ньому нема вільних місць;
- «Помилка гри. Нема зв'язку з сервером.» – означає, що вікно гри не може зв'язатися з сервером;
- «Помилка кінця гри. Незнайдено файли.» – означає, що вікно створення кінця гри не може знайти необхідні файли;
- «Акаунт створено. Реєстрація вдала.» – означає, що реєстрація пройшла успішно;
- «Логін зайнято. Реєстрація невдала.» – о означає, що реєстрація пройшла не успішно через те, що такий логін вже існує;
- «Некоректний логін або пароль. Реєстрація невдала.» – означає, що реєстрація пройшла не успішно бо логін або пароль не пройшли валідацію;
- «Авторизація невдала. Неправильний логін або пароль.» – означає, що авторизація пройшла не успішно через те що введено неправильні дані.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Тестування буде проведене за таким сценарієм:

а) авторизація в застосунок:

- 1) невведений логін або пароль;
- 2) введено неіснуючий логін;
- 3) введено неправильний пароль;
- 4) введено існуючий логін та правильний пароль;

б) реєстрація акаунту:

- 1) введено зайнятий логін;
- 2) введено некоректно логін;
- 3) введено некоректно пароль;
- 4) введено не зайнятий та коректний логін та коректний пароль;

в) головне вікно (вікно з головним меню):

- 1) перехід на інші сторінки;

г) вікно створення лобі:

- 1) створення лобі з незайнятою назвою;
- 2) створення лобі з зайнятою назвою;
- 3) створення лобі з різною кількістю гравців;

д) вікно пошуку лобі:

- 1) відображення лобі;
- 2) пошук лобі за назвою;
- 3) підключення до повного лобі;
- 4) підключення до не повного лобі;

е) вікно гри:

- 1) відображення ігрового столу;
- 2) відображення рук гравців;
- 3) відображення логіну гравця що ходить;
- 4) дія Hit;

- 5) дія Stand;
- 6) завершення гри;
- ж) вікно з результатом гри:
 - 1) відображення результату гри;

Для коректного тестування потрібно, щоб пристрій, який буде використовуватися, відповідав мінімальним технічним вимогам:

- потрібна оперативна пам'ять: 4 Гб для 64х та 2 Гб для 32х;
- потрібна місце на жорсткому диску: 400 Мб;
- операційні системи родини Windows 7/8.x/10.

6.2 Результати використання розробленого програмного забезпечення

У таблиці 6.1 представлено тестування програмного забезпечення.

Таблиця 6.1 – Тестування ПЗ

№	Назва	Опис результату	Оцінка тесту
Авторизація в застосунок			
1	Невведений логін або пароль	Авторизація не успішна. Отримано повідомлення	Пройшов
2	Введено неіснуючий логін	Авторизація не успішна. Отримано повідомлення	Пройшов
3	Введено неправильний пароль	Авторизація не успішна. Отримано повідомлення	Пройшов
4	Введено існуючий логін та правильний пароль	Авторизація успішна	Пройшов
Реєстрація акаунту			
1	Введено зайнятий логін	Реєстрація не успішна. Отримано повідомлення	Пройшов

Продовження таблиці 6.1

2	Введено некоректно логін	Реєстрація не успішна. Отримано повідомлення	Пройшов
3	Введено некоректно пароль	Реєстрація не успішна. Отримано повідомлення	Пройшов
4	Введено не зайнятий та коректний логін та коректний пароль	Реєстрація успішна	Пройшов
Головне вікно (вікно з головним меню)			
1	Перехід на інші сторінки	Перехід на усі інші сторінки коректний	Пройшов
Вікно створення лобі			
1	Створення лобі з незайнятою назвою	Лобі створено	Пройшов
2	Створення лобі з зайнятою назвою	Лобі не створено. Отримано повідомлення	Пройшов
3	Створення лобі з різною кількістю гравців	Створення з 2, 3 та 4 гравцями коректне	Пройшов
Вікно пошуку лобі			
1	Відображення лобі	Активні лобі відображаються з актуальною інформацією	Пройшов
2	Пошук лобі за назвою	Пошук працює правильно	Пройшов
3	Підключення до повного лобі	Підключення не вдале. Отримано повідомлення	Пройшов
4	Підключення до не повного лобі	Підключення вдале	Пройшов

Продовження таблиці 6.1

Вікно гри			
1	Відображення ігрового столу	Відображення коректне	Пройшов
2	Відображення рук гравців	Коректне відображення актуальної інформації	Пройшов
3	Відображення логіну гравця що ходить	Коректне відображення актуальної інформації	Пройшов
4	Дія Hit	Отримання гравцем карти та перехід ходу коректні	Пройшов
5	Дія Stand	Перехід ходу коректний	Пройшов
6	Завершення гри	Завершення гри та коректний підрахунок результатів у правильний час	Пройшов
Вікно з результатом гри			
1	Відображення результату гри	Зображення з результатом коректне для кожного гравця	Пройшов

ВИСНОВКИ

Під час виконання дипломної роботи було використано середовище Microsoft Visual Studio 2019 та підсистеми WPF та WCF для розробки. Також було проведено аналіз предметної області та описано існуючі методи вирішення завдання. Було порівняно та визначено переваги та недоліки існуючих аналогів таких як «Blackjackist», «Blackjack Gold» і мобільна гра «Blackjack» та обрано за прототип програму «Blackjack Gold».

Основним завданням дипломної роботи стала розробка десктопного клієнт-серверного ігрового застосунку «Blackjack», що дозволило закріпити теоретичні знання в програмуванні застосунків з клієнт-серверною архітектурою.

В першому розділі було проведено аналіз предметної області та обрано прототип проекту. В другому розділі було обрано програмні засоби використані в розробці ПЗ. В третьому розділі проведено опис реалізації програми, наведені створені класи та інтерфейси, описано користувацький інтерфейс та створену БД. В четвертому та п'ятому було надано інструкції користувачу та оператору щодо використання програмного продукту. В шостому розділі описано процес тестування та надано чек-лист виконаних тестів.

В створеному проєкті було реалізовано системи авторизації та реєстрації, створення та налаштування лобі, пошуку та підключення до лобі, готовності гравців лобі. Також було реалізовано мультиплеєрну ігрову сесію, яка надає користувачам ігровий процес за правилами гри «Blackjack», з якими можливо ознайомитися в застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bewersdorff J. Luck, Logic, and White Lies / Jörg Bewersdorff. – Abingdon: Taylor & Francis Group, 2021. – 548 p.
2. Як навчитися грати в блекджек. Кілька простих, але ефективних порад [Електронний ресурс]. – Режим доступу: <https://azart.nv.ua/ukr/reviews/blekdzhek-onlayn-50139538.html>.
3. Client-Server Model [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/client-server-model/>.
4. Kumar S. A review on client-server based applications and research opportunity / Santosh Kumar. // International Journal of Recent Scientific Research. – 2019. – №10. – P. 33857–33862.
5. Client-server [Electronic resource]. – Access mode: <https://www.techtarget.com/searchnetworking/definition/client-server>.
6. Батенко Л. П. Управління проектами: навч. посібник /Л. П. Батенко, О. А. Загородніх, В. В. Ліщинська. – К. КНЕУ, 2005. – 231 с.
7. Що таке веб додаток? Різниця між сайтом, веб-додатком, SPA і PWA [Електронний ресурс]. – Режим доступу: <https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/>.
8. Мобільні застосунки: їх види та особливості [Електронний ресурс]. – Режим доступу: <https://highload.today/uk/mobilni-zastosunki-yih-vidi-ta-osoblivosti/>.
9. Blackjackist Rules [Electronic resource]. – Access mode: <https://www.blackjackist.com/en/rules/>.
10. Classic Blackjack Gold [Electronic resource]. – Access mode: <https://www.casinonewsdaily.com/games/blackjack/classic-blackjack-gold>.
11. Blackjack [Електронний ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=com.tripledot.blackjack>.
12. Skeet J. C# in Depth, Fourth Edition / Jon Skeet. – Shelter Island, NY: Manning Publications, 2019. – 528 p.

13. The State of Developer Ecosystem 2023 [Electronic resource]. – Access mode: <https://www.jetbrains.com/lp/devecosystem-2023/>.
14. C# Named 2023 Programming Language of the Year by TIOBE [Electronic resource]. – Access mode: <https://adtmag.com/Articles/2024/01/08/C-sharp-TIOBE-language-of-the-year>.
15. C# in 2023: The MOST POPULAR Programming Language? [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/c-2023-most-popular-programming-language-cloudhawktech>.
16. Що таке C#? Чи підходить мені ця мова програмування? Чому вона крута? [Електронний ресурс]. – Режим доступу: <https://beetroot.academy/blog/courses/what-is-C>.
17. Як вивчити мову програмування C# та стати .NET розробником [Електронний ресурс]. – Режим доступу: <https://edu.cbsystematics.com/ua/blog/learn-csharp-become-dnet>.
18. Stack Overflow Dev Survey: VS Code, Visual Studio Still Top IDEs 5 Years Running [Electronic resource]. – Access mode: <https://visualstudiomagazine.com/Articles/2023/06/28/so-2023>.
19. 2023 Developer Survey [Electronic resource]. – Access mode: <https://survey.stackoverflow.co/2023/>.
20. What is Visual Studio? [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide>.
21. Невлюдов І.Ш. Автоматизація та приладобудування / І.Ш. Невлюдов // Аналіз сучасних середовищ розробки програмного забезпечення : Всеукраїнська науково-практична конференція, Харків, 25 листопада 2020 р. : тези доповідей. – Харків: ХНУРЕ, 2020. – С. 13–16.
22. Mickey Williams Microsoft Visual C# (Core Reference) / Williams Mickey – Washington: Microsoft Press Redmond, 2002. – 750 p.
23. Що таке WPF? [Електронний ресурс]. – Режим доступу: <https://wpf-tutorial.com/uk/1/про-wpf/що-таке-wpf/>.

24. WPF в порівнянні з WinForms [Електронний ресурс]. – Режим доступу: <https://wpf-tutorial.com/uk/2/про-wpf/wpf-в-порівнянні-з-winforms/>.
25. Database management systems (DBMS) [Electronic resource]. – Access mode: <https://www.techtarget.com/searchdatamanagement/definition/database-management-system>.
26. Різниця між SQL Vs MySQL Vs SQL Server (з прикладами) [Електронний ресурс]. – Режим доступу: <https://uk.myservername.com/difference-between-sql-vs-mysql-vs-sql-server>.
27. Архітектура програмного забезпечення: все що треба знати [Електронний ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya>.
28. Клієнт-серверна архітектура [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/client-server-architecture/>.
29. Game user interface design examples: what you can learn [Electronic resource]. – Access mode: <https://retrostylegames.com/blog/game-user-interface-design-examples/>.
30. Best 2D video game art styles: from pixel art to isometric and realistic games [Electronic resource]. – Access mode: <https://rocketbrush.com/blog/best-2d-video-game-art-styles-from-pixel-art-to-isometric-and-realistic-games>.
31. Растрова та векторна графіка. В чому різниця? [Електронний ресурс]. – Режим доступу: https://teletype.in/@pikiran_academy/MTaaE1ZJ6Oc.
32. ColorMagic [Electronic resource]. – Access mode: <https://colormagic.app/palette/GtLDm5CpdkoFZ8S4Z>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

У сучасному світі комп'ютерні ігри стали невід'ємною частиною нашого життя, надаючи нам можливість насолоджуватися віртуальними пригодами та випробовувати свої навички у різноманітних сценаріях. Однією з найпопулярніших та улюблених гри серед гравців є «Blackjack» – захоплююча карточна гра, яка викликає азарт і розумові здібності.

Головною метою розробки є створення зручного та захоплюючого середовища для гравців, де вони зможуть змагатися один з одним та насолоджуватися неперевершеним досвідом гри.

A.1.1 Назва програми

Назва програми: «Blackjack».

A.1.2 Призначення та область застосування

Призначення ігрової програми «Blackjack» полягає у створенні віртуального середовища, що дозволяє гравцям насолоджуватися захоплюючою грою в «Blackjack» без необхідності фізично перебувати в казино або мати наявність карткового набору.

Ігрова програма «Blackjack» може бути використана для навчання та розвитку навичок гри в «Blackjack». Гравці, незалежно від свого досвіду, зможуть вивчати правила гри, вдосконалювати стратегії та розуміння гральних ситуацій, що допоможе їм поліпшити свої навички.

Область застосування: приватні особи.

А.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка та програмна реалізація десктопної клієнт-серверної гри Blackjack», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

А.3 Призначення розробки

Розробка ігрової програми «Blackjack» має на меті створення популярної карткової гри, відомої як «Блекджек» або «21». Ця програма дозволяє користувачам грати в гру проти комп'ютерного дилера або інших гравців, використовуючи стандартну колоду карт.

Основні цілі розробки ігрової програми «Blackjack»:

- розробка програми, яка забезпечує реалістичний досвід гри в «Блекджек», включаючи правила гри, роздачу карт, вирахування результатів;
- надання можливості налаштування параметрів ігрового лобі для задоволення потреб користувачів;
- реалізація можливості грати в «Блекджек» з іншими гравцями в режимі реального часу. Це може забезпечити соціальний аспект гри, де гравці можуть змагатися один з одним;
- розробка інтелектуального алгоритму дилера, який здійснює логічні рішення згідно з правилами гри «Блекджек»;
- розробка привабливого інтерфейсу користувача з графічними елементами, які відображають картки, столи і інші елементи гри.

В цілому, розробка ігрової програми «Blackjack» спрямована на створення захоплюючого віртуального досвіду гри в «Блекджек» для користувачів, незалежно від їхнього рівня гри, та надання можливостей для налаштування і соціальної взаємодії.

А.4 Вимоги до програми

А.4.1 Вимоги до функціональних характеристик

Основні функціональні характеристики клієнт-серверної десктопної ігрової програми «Blackjack»:

- реєстрація та авторизація користувачів. можливість створення облікового запису користувача з унікальними обліковими даними (логін, пароль) та можливість авторизуватися в програмі для доступу до гри;
- режим гри проти комп'ютерного дилера можливість грати в «Блекджек» проти комп'ютерного дилера згідно з правилами гри;
- мультиплеєрність, можливість грати в «Блекджек» з іншими гравцями в режимі реального часу;
- налаштування гри: можливість створювати та налаштовувати ігрове лобі;
- привабливий інтерфейс користувача з графічними елементами, які відображають картки, столи і інші елементи гри;
- можливість переглядати існуючі лобі, та підключатися до них;
- забезпечення взаємодії між клієнтом та сервером;
- можливість переглядати правила гри.

А.4.2 Вимоги до надійності

Надійність є важливою характеристикою будь-якої клієнт-серверної десктопної ігрової програми, такої як “Blackjack”. Основні вимоги до надійності:

- стабільність з'єднання: забезпечення стабільного з'єднання між клієнтом і сервером;
- обробка помилок: ефективна обробка помилок і виключних ситуацій;

- захист даних: забезпечення конфіденційності та цілісності даних користувачів;
- тестування та відладка: проведення комплексного тестування програми для виявлення та усунення помилок, а також виявлення потенційних проблем з надійністю.

A.4.3 Вимоги до технічних та програмних засобів

Для коректної роботи з готовим програмним забезпеченням потрібно, щоб пристрій, який буде використовуватися, відповідав мінімальним технічним вимогам:

- потрібна оперативна пам'ять: 4 Гб для 64х та 2 Гб для 32х;
- потрібна місце на жорсткому диску: 400 Мб;
- операційні системи родини Windows 10.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- опис програми;
- пояснювальна записка.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Основні стадії розробки:

- аналіз вимог до програмного забезпечення;
- створення технічного завдання;
- проєктування програмного забезпечення;
- розробка програмного забезпечення;

- тестування програмного забезпечення;
- оформлення програмної документації.

A.6.2 Етапи розробки та зміст роботи на кожному етапі

Етапи розробки та зміст роботи на кожному етапі:

- аналіз предметної області – дослідження актуальності теми, ознайомлення з основними поняттями, правилами та стратегіями гри «Blackjack»;
- аналіз існуючих аналогів – огляд існуючих рішень і аналогів, їх порівняння;
- визначення вимог до програмного продукту – опис функцій, характеристик та обмежень, які мають бути враховані під час розробки програми;
- визначення вимог до надійності – встановлення та формулювання вимог, які стосуються надійності програмного продукту. Надійність відноситься до здатності програми працювати без відмов і збоїв, забезпечуючи очікувану функціональність та стабільну продуктивність;
- визначення вимог до технічних та програмних засобів – формулювання вимог щодо необхідних технічних та програмних засобів для розробки та експлуатації програмного продукту;
- визначення вимог до програмної документації – опис складу документації;
- проектування структури клієнтської частини – включає створення дизайну і архітектури клієнтського інтерфейсу, визначення компонентів, модулів та їх взаємозв'язків;
- проектування структури серверної частини – визначення архітектури, компонентів та модулів, які будуть виконувати функції на сервері та забезпечувати необхідну логіку та функціональність програми;

- проектування структури бази даних – визначення схеми БД, таблиць, полів та взаємозв'язків між ними, а також вибір відповідної системи керування базами даних для реалізації цієї структури;

- розробка клієнтської частини – створення дизайну інтерфейсу, включаючи розташування елементів, графічний вигляд, навігацію та інші аспекти, які забезпечують зручну взаємодію з користувачем;

- розробка хоста – розробки програмного середовища або інфраструктури, що забезпечує виконання та надання послуг програмному продукту;

- розробка серверної частини – створення програмної складової, яка забезпечує обробку запитів, бізнес-логіку та зберігання даних на сервері;

- розробка таблиці бази даних та їх наповнення – створення структури бази даних і введення даних в ці таблиці;

- завершення розробки програмного продукту – налаштування з'єднання усіх розроблених частин між собою;

- тестування програмного забезпечення – перевірка і оцінка функціональності, якості, надійності та відповідності вимогам програмного продукту;

- виправлення знайдених проблем – виправлення помилок зафіксованих під час тестування;

- оформлення опису програми – творення документації, яка детально описує функціональність, архітектуру, використання технологій та інші аспекти програмного продукту;

- завершення розробки пояснювальної записки – оформлення усіх пунктів пояснювальної записки.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

«Blackjack» – це захоплюючий десктопний клієнт-серверний ігровий застосунок, який пропонує емоційну гру в популярну карточну гру «Блекджек».

Мета застосунку полягає в тому, щоб надати користувачам захоплюючий інтерактивний досвід гри в Блекджек, де вони зможуть випробувати свою удачу та навички проти віртуального дилера та грати проти інших гравців у мережевій грі.

Застосунок розрахований на широкий спектр кінцевих користувачів, включаючи азартних гравців, фанатів карточних ігор та просто тих, хто бажає розважитись та провести час у захоплюючій грі. Він пропонує зручний інтерфейс, захоплюючу графіку та можливість гри з віртуальним дилером або гру та іншими гравцями в режимі мережі.

Б.2 Функціональне призначення

Функції, що виконує програма:

- реєстрація та авторизація користувачів. можливість створення облікового запису користувача з унікальними обліковими даними (логін, пароль) та можливість авторизуватися в програмі для доступу до гри;
- режим гри проти комп'ютерного дилера можливість грати в «Блекджек» проти комп'ютерного дилера згідно з правилами гри;
- мультиплеєрність, можливість грати в «Блекджек» з іншими гравцями в режимі реального часу;
- налаштування гри: можливість створювати та налаштовувати ігрове лобі;
- привабливий інтерфейс користувача з графічними елементами, які відображають картки, столи і інші елементи гри;

- можливість переглядати існуючі лобі, та підключатися до них;
- забезпечення взаємодії між клієнтом та сервером;
- можливість переглядати правила гри.

Б.3 Опис логічної структури програми

Логічна структура ПЗ показна за допомогою структурної схеми на рисунку Б.1

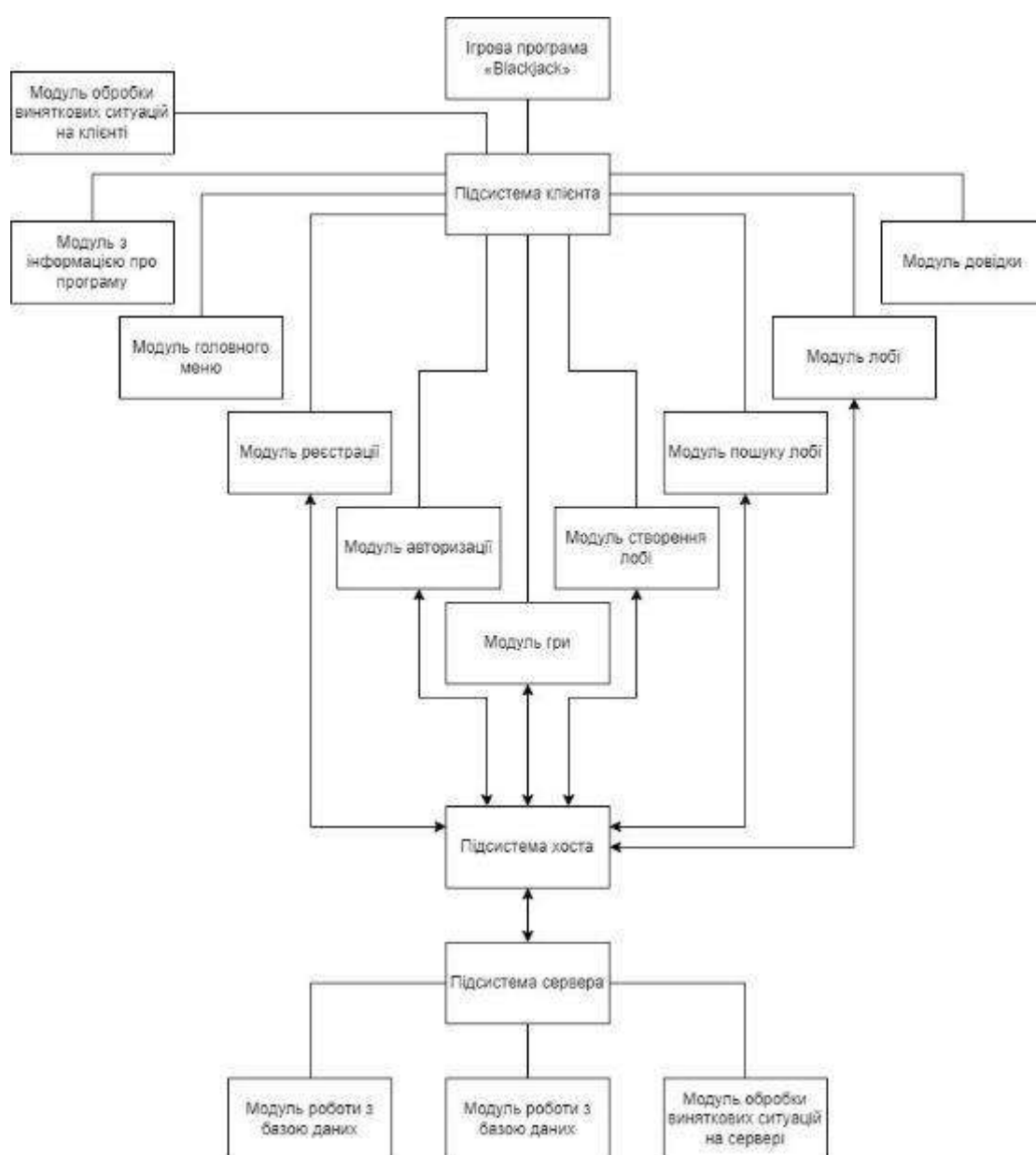


Рисунок Б.1 – Структурна схема проєкту

Б.3.1 Алгоритм процесу авторизації

Цей алгоритм створено для реалізації авторизації у застосунок, під обраним логіном.

Б.3.2 Алгоритм процесу реєстрації

Цей алгоритм створено для реалізації створення акаунту з обраним логіном та паролем, для подальшої авторизації.

Б.3.3 Алгоритм налаштування лобі

Цей алгоритм створено для реалізації внесення змін в налаштування лобі.

Б.3.4 Алгоритм створення лобі

Цей алгоритм створено для реалізації створення лобі, для подальшого пошуку гравців, для ігрової сесії.

Б.3.5 Алгоритм створення лобі

Цей алгоритм створено для реалізації створення лобі, для подальшого пошуку гравців, для ігрової сесії.

Б.3.6 Алгоритм зміна статусу готовності

Цей алгоритм створено для реалізації зміни статусу готовності користувачів, для відстеження моменту, коли треба розпочати ігрову сесію.

Б.3.7 Алгоритм процесу ходу

Цей алгоритм створено для реалізації процесу ходу гравців та дилера, що необхідно для створення ігрового процесу.

Б.4 Використані технічні засоби

Під час виконання дипломної роботи було використано мову програмування C# та середовище розробки Microsoft Visual Studio 2019 з підсистемами WPF та WCF. СКБД використана у розробці Microsoft SQL Server. Архітектуру клієнт-серверна. Через велику кількість переваг, популярність, зручність у використанні, та гарну взаємодію між ними було обрано ці програмні засоби.

Для коректної роботи з готовим програмним забезпеченням потрібно, щоб пристрій, який буде використовуватися, відповідав мінімальним технічним вимогам:

- потрібна оперативна пам'ять: 4 Гб для 64х та 2 Гб для 32х;
- потрібна місце на жорсткому диску: 400 Мб;
- операційні системи родини Windows 7/8.x/10.

Для компіляції та внесення змін до програми рекомендується використовувати середовище Visual Studio 2019 та платформу .NET Framework 4.7.2.

Б.5 Виклик і завантаження

Програма запускається за допомогою виконуваного файлу BlackjackClient.exe та здійснюється подвійним натисканням лівої клавіші комп'ютерної миші по ньому або його ярлику.

Б.6 Початкові дані

Програма отримує такі вхідні дані як:

- сигнали про натискання клавіш миші та клавіатури;
- інформацію про обрані налаштування при створенні лобі;
- індекс обраного користувачем елементу списку;
- базу даних;
- зображення карт, кнопок та фонів інтерфейсу.

Програма видає такі вихідні дані як:

- інформація про активні лобі;
- інформація про правила гри;
- інформація про лобі в якому знаходиться користувач;
- інформація про ігрову сесію;
- інформація про результат гри.

ДОДАТОК В
Текст програми

В.1 Бібліотека з сервером Blackjack

В.1.1 Файл app.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="mexBeh">
          <serviceMetadata httpGetEnabled="true"
httpsGetEnabled="true" />
          <serviceDebug includeExceptionDetailInFaults="false" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service name="Blackjack.ServiceBlackjack"
behaviorConfiguration="mexBeh">
        <endpoint address="" binding="basicHttpBinding"
contract="Blackjack.IServiceBlackjack">
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
        <host>
          <baseAddresses>
            <add
baseAddress="http://localhost:8733/Design_Time_Addresses/Blackjack/ServiceBlackjack/" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

В.1.2 Файл DataBaseAPI.cs

```
using System.Data;
using System.Data.SqlClient;

namespace Blackjack
{
  class DataBaseAPI
  {
    // Шлях до БД
    string ConnectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;" +
@"AttachDbFilename=E:\c#\Blackjack\Blackjack\BlackjackServerDatabase.mdf;" +
"Integrated Security=True";
```

```

// Чи зайнятий такий логін
public bool LoginIsExists(string Login)
{
    try
    {
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("SELECT ID FROM
dbo.UsersTable WHERE Login = '" + Login + "';", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
        if (Table.Rows.Count != 0)
        {
            return false;
        }
        else
            return true;
    }
    catch
    {
        BlackjackException exception = new BlackjackException(1);
        return false;
    }
}
// Створення акаунту
public bool AddUser(string Login, string Password) // Додавання нового
користувача до БД
{
    try
    {
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("INSERT dbo.UsersTable
(Login, Password) VALUES ('" + Login + "', '" + Password + "');", Connection);
        Connection.Open();
        Command.ExecuteNonQuery();
        Connection.Close();
        ServerUser user = AuthorizationCheck(Login, Password);
        if (user.ID != 0)
            return true;
        else
            return false;
    }
    catch
    {
        BlackjackException exception = new BlackjackException(2);
        return false;
    }
}
// Видалення акаунту
public void DeleteUser(long User_ID) // Видалення користувача
{
    try
    {
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("DELETE FROM
dbo.UsersTable WHERE ID = '" + User_ID + "';");
        Connection.Open();
        Command.ExecuteNonQuery();
    }
}

```

```

        Connection.Close();
    }
    catch
    {
        BlackjackException exception = new BlackjackException(3);
    }
}
// Чи привильні логін пароль для авторизації
public ServerUser AuthorizationCheck(string Login, string Password) //
Перевірка на наявність логіну в БД
{
    try
    {
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(ConnectionString);
        SqlCommand Command = new SqlCommand("SELECT ID FROM
dbo.UsersTable WHERE Login = '" + Login + "' AND Password = '" + Password +
"'";", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
        if (Table.Rows.Count != 0)
        {
            ServerUser user = new ServerUser();
            user.ID = Table.Rows[0].Field<int>("ID");
            user.Login = Login;
            return user;
        }
        else
            return null;
    }
    catch
    {
        BlackjackException exception = new BlackjackException(4);
        return null;
    }
}
// Отримати логін за ІД
public string GetLogin(int ID)
{
    try
    {
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(ConnectionString);
        SqlCommand Command = new SqlCommand("SELECT Login FROM
dbo.UsersTable WHERE ID = '" + ID + "'";", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
        return Table.Rows[0].Field<string>("Login"); ;
    }
    catch
    {
        BlackjackException exception = new BlackjackException(5);
        return "";
    }
}
}
}
}

```

B.1.3 Файл IServiceBlackjack.cs

```

using System.Collections.Generic;
using System.ServiceModel;

namespace Blackjack
{
    [ServiceContract (CallbackContract = typeof(IServiceBlackjackCallback))]
    public interface IServiceBlackjack
    {
        [OperationContract]
        bool IsConnect(string Login);

        [OperationContract]
        int Connect(string Login, string Password);

        [OperationContract]
        void Disconnect(int id);

        [OperationContract]
        bool Registration(string Login, string Password);

        [OperationContract]
        bool CreateLobby(Lobby lobby);

        [OperationContract]
        bool ConnectLobby(Lobby lobby, int UserID);

        [OperationContract]
        void DisconnectLobby(Lobby lobby, int UserID);

        [OperationContract]
        void DelLobby(Lobby lobby);

        [OperationContract]
        List<Lobby> GetListLobby();

        [OperationContract]
        string GetLogin(int ID);

        [OperationContract]
        void UpdateStatus(Lobby lobby);

        [OperationContract]
        void Turn(Game game);
    }

    public interface IServiceBlackjackCallback
    {
        [OperationContract(IsOneWay = true)]
        void LobbyCallback(Lobby lobby);

        [OperationContract(IsOneWay = true)]
        void ListLobbyCallback(List<Lobby> lobbys);

        [OperationContract(IsOneWay = true)]
        void StartGameCallback(Game game);

        [OperationContract(IsOneWay = true)]
        void GameCallback(Game game);
    }
}

```

```
    }
}
```

B.1.4 Файл ServiceBlackjack.cs

```
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;

namespace Blackjack
{
    [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
    public class ServiceBlackjack : IServiceBlackjack
    {
        DataBaseAPI DataBase = new DataBaseAPI();
        List<ServerUser> users = new List<ServerUser>();
        List<Lobby> lobbys = new List<Lobby>();

        // Перевірка чи вже авторизований користувач з таким Login
        public bool IsConnect(string Login)
        {
            for (int i = 0; i < users.Count; i++)
            {
                if (users[i].Login == Login)
                    return true;
            }
            return false;
        }
        // Авторизація, повертає ID користувача якщо авторизація успішна, і 0
        якщо ні
        public int Connect(string Login, string Password)
        {
            ServerUser user = DataBase.AuthorizationCheck(Login, Password);
            if (user != null)
            {
                user.operationContext = OperationContext.Current;
                users.Add(user);
                return user.ID;
            }
            else
                return 0;
        }
        // Вихід з акаунту
        public void Disconnect(int ID)
        {
            var user = users.FirstOrDefault(i => i.ID == ID);
            if (user != null)
            {
                users.Remove(user);
            }
        }
        // Реєстрація, додавання акаунту до бази даних
        public bool Registration(string Login, string Password)
        {
            if (DataBase.LoginIsExists(Login) == false)
            {
                return DataBase.AddUser(Login, Password);
            }
            return false;
        }
    }
}
```

```

    }
    // Створення лобі на сервері
    public bool CreateLobby(Lobby lobby)
    {
        for (int i = 0; i < lobbys.Count; i++)
        {
            if (lobbys[i].Name == lobby.Name)
                return false;
        }
        lobbys.Add(lobby);
        return true;
    }
    // Підключення до лобі
    public bool ConnectLobby(Lobby lobby, int UserID)
    {
        int index = -1;
        for (int i = 0; i < lobbys.Count; i++)
        {
            if (lobbys[i].Name == lobby.Name)
                index = i;
        }
        int countOfPlayer = 0;
        for (int i = 0; i < lobbys[index].Player_ID.Length; i++)
        {
            if (lobbys[index].Player_ID[i] == -1)
            {
                lobbys[index].Player_ID[i] = UserID;
                i = lobbys[index].Player_ID.Length;
            }
            else
            {
                countOfPlayer++;
            }
        }
        if (countOfPlayer != lobbys[index].MaxNumOfPlayer)
        {
            foreach (var item in users)
            {
                item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().ListLobbyCallback(lobbys);
                for (int i = 0; i < lobbys[index].Player_ID.Length; i++)
                {
                    if (lobbys[index].Player_ID[i] != -1)

                item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().LobbyCallback(lobbys[index]);
                }
            }
            return true;
        }
        else
            return false;
    }
    // Відключення від лобі
    public void DisconnectLobby(Lobby lobby, int UserID)
    {
        int index = -1;
        for (int i = 0; i < lobbys.Count; i++)
        {
            if (lobbys[i].Name == lobby.Name)
                index = i;

```

```

    }
    for (int i = 0; i < lobbys[index].Player_ID.Length; i++)
    {
        if (lobbys[index].Player_ID[i] == UserID)
        {
            lobbys[index].Player_ID[i] = -1;
            i = lobbys[index].Player_ID.Length;
        }
    }
    int countOfPlayer = 0;
    for (int i = 0; i < lobbys[index].Player_ID.Length; i++)
    {
        if (lobbys[index].Player_ID[i] != -1)
        {
            countOfPlayer++;
        }
    }
    if (countOfPlayer == 0)
    {
        DelLobby(lobbys[index]);
    }
    else
    {
        foreach (var item in users)
        {
            for (int i = 0; i < lobbys[index].Player_ID.Length; i++)
            {
                if (lobbys[index].Player_ID[i] != -1)

item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().LobbyCa
llback(lobbys[index]);
            }
        }
    }
    foreach (var item in users)
    {

item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().ListLob
byCallback(lobbys);
    }
}
// Видалення лобі на сервері
public void DelLobby(Lobby lobby)
{
    lobbys.Remove(lobby);
}
// Повернення списку активних лобі
public List<Lobby> GetListLobby()
{
    return lobbys;
}
// Повернення Login користувача за його ID
public string GetLogin(int ID)
{
    return DataBase.GetLogin(ID);
}
// Оновити статус гравця у лобі (Готов, не готов)
public void UpdateStatus(Lobby lobby)
{
    // Знаходження індексу лобі у списку
    int index = -1;
    for (int i = 0; i < lobbys.Count; i++)

```

```

    {
        if (lobbys[i].Name == lobby.Name)
            index = i;
    }
    // Зміна статусу гравців
    lobbys[index].Player_Status = lobby.Player_Status;
    int ready = 0;
    bool needDelete = true;
    Game game = new Game();
    // Чи треба почати гру
    for (int i = 0; i < lobbys[index].Player_Status.Length; i++)
    {
        if (lobbys[index].Player_Status[i] == "Готов")
            ready++;
    }
    if (ready == lobbys[index].Player_Status.Length)
    {
        game.GameLobby = lobbys[index];
    }
    // Розсилка змін
    foreach (var item in users)
    {
        for (int i = 0; i < lobbys[index].Player_ID.Length; i++)
        {
            if (lobbys[index].Player_ID[i] == item.ID)
            {
                item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().LobbyCa
                llback(lobbys[index]);
                if (ready == lobbys[index].Player_Status.Length)
                {
                    item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().StartGa
                    meCallback(game);
                }
            }
        }
    }
    // Оновлення списку лобі якщо гра почалась
    if (ready == lobbys[index].Player_Status.Length)
    {
        foreach (var item in users)
        {
            if (needDelete == true)
            {
                lobbys.RemoveAt(index);
                needDelete = false;
            }
        }
    }
    item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().ListLob
    byCallback(lobbys);
}
// Обробка ходу користувача, відправка про нього інформації всім
учасникам гри
public void Turn(Game game)
{
    // Розсилка змін
    foreach (var item in users)
    {
        for (int i = 0; i < game.Players.Length; i++)

```

```

        {
            if (game.Players[i] == item.ID)
            {
                item.operationContext.GetCallbackChannel<IServiceBlackjackCallback>().GameCallback(game);
            }
        }
    }
}

```

В.2 Проєкт клієнтської частини BlackjackClient

В.2.1 Файл App.config

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <system.serviceModel>
    <bindings>
      <netTcpBinding>
        <binding name="NetTcpBinding_IServiceBlackjack">
          <security>
            <transport sslProtocols="None" />
          </security>
        </binding>
      </netTcpBinding>
    </bindings>
    <client>
      <endpoint
        address="net.tcp://localhost:8302/"
        binding="netTcpBinding"
        bindingConfiguration="NetTcpBinding_IServiceBlackjack"
        contract="ServiceBlackjack.IServiceBlackjack"
        name="NetTcpBinding_IServiceBlackjack">
        <identity>
          <userPrincipalName value="DESKTOP-SGT4807\Мария" />
        </identity>
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>

```

В.2.2 Файл GameProcess.cs

```

using BlackjackClient.ServiceBlackjack; // Для роботи з сервером
using System;
using System.Collections.Generic;
using System.Linq;

namespace BlackjackClient
{

```

```

class GameProcess
{
    // Порахувати кількість балів в руці
    public int Score(Card[] Hand)
    {
        int score = 0;
        for (int i = 0; i < Hand.Length; i++)
        {
            if (Hand[i] != null)
            {
                if (Hand[i].Name != "A")
                {
                    score += Hand[i].Cost;
                }
                else
                {
                    if (score + Hand[i].Cost <= 21)
                    {
                        score += Hand[i].Cost;
                    }
                    else
                    {
                        score += 1;
                    }
                }
            }
        }
        return score;
    }

    // Процес ходу
    public Game Turn(Game game, string action)
    {
        if (action == "Start")
        {
            game.TurnIndex = 0;
            Random random = new Random();
            int randomIndex = 0;
            List<Card> cardsList = game.CardDeck.cards.ToList<Card>();
            // 2 карти ділепу
            List<Card> DillerHand = game.DillerHand.ToList<Card>();
            randomIndex = random.Next(0, cardsList.Count);
            DillerHand.Add(cardsList[randomIndex]);
            cardsList.Remove(cardsList[randomIndex]);
            randomIndex = random.Next(0, cardsList.Count);
            DillerHand.Add(cardsList[randomIndex]);
            cardsList.Remove(cardsList[randomIndex]);
            game.DillerHand = DillerHand.ToArray();
            // 2 карти гравцям
            if (game.Player1Hand != null)
            {
                List<Card> Player1Hand = game.Player1Hand.ToList<Card>();
                randomIndex = random.Next(0, cardsList.Count);
                Player1Hand.Add(cardsList[randomIndex]);
                cardsList.Remove(cardsList[randomIndex]);
                randomIndex = random.Next(0, cardsList.Count);
                Player1Hand.Add(cardsList[randomIndex]);
                cardsList.Remove(cardsList[randomIndex]);
                game.Player1Hand = Player1Hand.ToArray();
            }
            if (game.Player2Hand != null)
            {

```

```

        List<Card> Player2Hand = game.Player2Hand.ToList<Card>();
        randomIndex = random.Next(0, cardsList.Count);
        Player2Hand.Add(cardsList[randomIndex]);
        cardsList.Remove(cardsList[randomIndex]);
        randomIndex = random.Next(0, cardsList.Count);
        Player2Hand.Add(cardsList[randomIndex]);
        cardsList.Remove(cardsList[randomIndex]);
        game.Player2Hand = Player2Hand.ToArray();
    }
    if (game.Player3Hand != null)
    {
        List<Card> Player3Hand = game.Player3Hand.ToList<Card>();
        randomIndex = random.Next(0, cardsList.Count);
        Player3Hand.Add(cardsList[randomIndex]);
        cardsList.Remove(cardsList[randomIndex]);
        randomIndex = random.Next(0, cardsList.Count);
        Player3Hand.Add(cardsList[randomIndex]);
        cardsList.Remove(cardsList[randomIndex]);
        game.Player3Hand = Player3Hand.ToArray();
    }
    if (game.Player4Hand != null)
    {
        List<Card> Player4Hand = game.Player4Hand.ToList<Card>();
        randomIndex = random.Next(0, cardsList.Count);
        Player4Hand.Add(cardsList[randomIndex]);
        cardsList.Remove(cardsList[randomIndex]);
        randomIndex = random.Next(0, cardsList.Count);
        Player4Hand.Add(cardsList[randomIndex]);
        cardsList.Remove(cardsList[randomIndex]);
        game.Player4Hand = Player4Hand.ToArray();
    }
    game.CardDeck.cards = cardsList.ToArray();
}
else
{
    Random random = new Random();
    int randomIndex = 0;
    List<Card> cardsList = game.CardDeck.cards.ToList<Card>();
    // Додавання карт
    if (action == "Hit")
    {
        randomIndex = random.Next(0, cardsList.Count);
        switch (game.TurnIndex)
        {
            case 0:
                List<Card> Player1Hand =
game.Player1Hand.ToList<Card>();
                Player1Hand.Add(cardsList[randomIndex]);
                game.Player1Hand = Player1Hand.ToArray();
                break;
            case 1:
                List<Card> Player2Hand =
game.Player2Hand.ToList<Card>();
                Player2Hand.Add(cardsList[randomIndex]);
                game.Player2Hand = Player2Hand.ToArray();
                break;
            case 2:
                List<Card> Player3Hand =
game.Player3Hand.ToList<Card>();
                Player3Hand.Add(cardsList[randomIndex]);
                game.Player3Hand = Player3Hand.ToArray();
                break;
        }
    }
}

```

```

        case 3:
            List<Card> Player4Hand =
game.Player4Hand.ToList<Card>();
            Player4Hand.Add(cardsList[randomIndex]);
            game.Player4Hand = Player4Hand.ToArray();
            break;
        }
        cardsList.Remove(cardsList[randomIndex]);
    }
    game.TurnIndex++;
    // Чи треба робити хід дилера або завершити гру
    if (game.TurnIndex == game.Players.Length)
    {
        if (Score(game.DillerHand) < 17)
        {
            game.TurnIndex = 0;
            List<Card> DillerHand =
game.DillerHand.ToList<Card>();
            randomIndex = random.Next(0, cardsList.Count);
            DillerHand.Add(cardsList[randomIndex]);
            cardsList.Remove(cardsList[randomIndex]);
            game.DillerHand = DillerHand.ToArray();
        }
        else
        {
            game.EndGame = true;
        }
    }
    game.CardDeck.cards = cardsList.ToArray();
    // Чи треба завершити гру
    if (game.EndGame == true)
    {
        // Якщо дилер набрав більше 21
        if (Score(game.DillerHand) > 21)
        {
            // Гравець 1
            if (game.Player1Hand != null)
            {
                // Вийшов з гри?
                if (game.Result[0] != -1)
                {
                    // Менше дилера - перемога
                    if (Score(game.Player1Hand) <
Score(game.DillerHand))
                    {
                        game.Result[0] = 4;
                    }
                    else
                    {
                        // Більше дилера - нічия
                        if (Score(game.Player1Hand) > 21)
                        {
                            game.Result[0] = 3;
                        }
                    }
                }
            }
        }
        // Гравець 2
        if (game.Player2Hand != null)
        {
            // Вийшов з гри?
            if (game.Result[1] != -1)

```

```

{
    // Менше дилера - перемога
    if (Score(game.Player2Hand) <
Score(game.DillerHand))
    {
        game.Result[1] = 4;
    }
    else
    {
        // Більше дилера - нічия
        if (Score(game.Player2Hand) > 21)
        {
            game.Result[1] = 3;
        }
    }
}
// Гравець 3
if (game.Player3Hand != null)
{
    // Вийшов з гри?
    if (game.Result[2] != -1)
    {
        // Менше дилера - перемога
        if (Score(game.Player3Hand) <
Score(game.DillerHand))
        {
            game.Result[2] = 4;
        }
        else
        {
            // Більше дилера - нічия
            if (Score(game.Player3Hand) > 21)
            {
                game.Result[2] = 3;
            }
        }
    }
}
// Гравець 4
if (game.Player4Hand != null)
{
    // Вийшов з гри?
    if (game.Result[3] != -1)
    {
        // Менше дилера - перемога
        if (Score(game.Player4Hand) <
Score(game.DillerHand))
        {
            game.Result[3] = 4;
        }
        else
        {
            // Більше дилера - нічия
            if (Score(game.Player4Hand) > 21)
            {
                game.Result[3] = 3;
            }
        }
    }
}
}
}

```

```

// Якщо дилер набрав менше 21 або 21
else
{
    // Гравець 1
    if (game.Player1Hand != null)
    {
        // Вийшов з гри?
        if (game.Result[0] != -1)
        {
            // Більше 21 або менше дилера - програш
            if ((Score(game.Player1Hand) > 21) ||
                (Score(game.Player1Hand) < Score(game.DillerHand)))
            {
                game.Result[0] = 2;
            }
            else
            {
                // Стільки скільки у дилера - нічия
                if (Score(game.Player1Hand) ==
                    Score(game.DillerHand))
                {
                    game.Result[0] = 3;
                }
                else
                {
                    // В інших випадках - перемога
                    if (Score(game.Player1Hand) >
                        Score(game.DillerHand))
                    {
                        game.Result[0] = 4;
                    }
                }
            }
        }
    }
    // Гравець 2
    if (game.Player2Hand != null)
    {
        // Вийшов з гри?
        if (game.Result[1] != -1)
        {
            // Більше 21 або менше дилера - програш
            if ((Score(game.Player2Hand) > 21) ||
                (Score(game.Player2Hand) < Score(game.DillerHand)))
            {
                game.Result[1] = 2;
            }
            else
            {
                // Стільки скільки у дилера - нічия
                if (Score(game.Player2Hand) ==
                    Score(game.DillerHand))
                {
                    game.Result[1] = 3;
                }
                else
                {
                    // В інших випадках - перемога
                    if (Score(game.Player2Hand) >
                        Score(game.DillerHand))
                    {
                        game.Result[1] = 4;
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
}
// Гравець 3
if (game.Player3Hand != null)
{
    // Вийшов з гри?
    if (game.Result[2] != -1)
    {
        // Більше 21 або менше дилера - програш
        if ((Score(game.Player3Hand) > 21) ||
            (Score(game.Player3Hand) < Score(game.DillerHand)))
        {
            game.Result[2] = 2;
        }
        else
        {
            // Стільки скільки у дилера - нічия
            if (Score(game.Player3Hand) ==
                Score(game.DillerHand))
            {
                game.Result[2] = 3;
            }
            else
            {
                // В інших випадках - перемога
                if (Score(game.Player3Hand) >
                    Score(game.DillerHand))
                {
                    game.Result[2] = 4;
                }
            }
        }
    }
}
// Гравець 4
if (game.Player4Hand != null)
{
    // Вийшов з гри?
    if (game.Result[3] != -1)
    {
        // Більше 21 або менше дилера - програш
        if ((Score(game.Player4Hand) > 21) ||
            (Score(game.Player4Hand) < Score(game.DillerHand)))
        {
            game.Result[3] = 2;
        }
        else
        {
            // Стільки скільки у дилера - нічия
            if (Score(game.Player4Hand) ==
                Score(game.DillerHand))
            {
                game.Result[3] = 3;
            }
            else
            {
                // В інших випадках - перемога
                if (Score(game.Player4Hand) >
                    Score(game.DillerHand))

```

```

        {
            game.Result[3] = 4;
        }
    }
}
}
}
}
}
// Якщо грв не завершена
else
{
    // Гравець 1
    if (game.Player1Hand != null)
    {
        // Вийшов з гри?
        if (game.Result[0] != -1)
        {
            // 21 - перемога
            if (Score(game.Player1Hand) == 21)
            {
                game.Result[0] = 4;
            }
            else
            {
                // Більше 21 - програш
                if (Score(game.Player1Hand) > 21)
                {
                    game.Result[0] = 2;
                }
            }
        }
    }
}
// Гравець 2
if (game.Player2Hand != null)
{
    // Вийшов з гри?
    if (game.Result[1] != -1)
    {
        // 21 - перемога
        if (Score(game.Player2Hand) ==
Score(game.DillerHand))
        {
            game.Result[1] = 4;
        }
        else
        {
            // Більше 21 - програш
            if (Score(game.Player2Hand) > 21)
            {
                game.Result[1] = 2;
            }
        }
    }
}
// Гравець 3
if (game.Player3Hand != null)
{
    // Вийшов з гри?
    if (game.Result[2] != -1)
    {

```



```

        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border Background="{TemplateBinding Background}"
                    CornerRadius="15">
                    <Border.Effect>
                        <DropShadowEffect Color="#AA7D26" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
                    </Border.Effect>
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
<!-- Налаштування кнопки -->
<Style x:Key="ButtonRound" TargetType="Button">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="Background" Value="#faf1e1"/>
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="21" />

    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border Background="{TemplateBinding Background}"
                    CornerRadius="90">
                    <Border.Effect>
                        <DropShadowEffect Color="#AA7D26" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
                    </Border.Effect>
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- Налаштування текстових полів -->
<Style TargetType="TextBox">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#faf1e1"/>
    <Setter Property="BorderThickness" Value="0" />
    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="TextBox">

```



```

        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <GridViewHeaderRowPresenter Grid.Row="0"
Columns="{Binding Path=View.Columns, RelativeSource={RelativeSource
TemplatedParent}}"/>
        <ScrollView Grid.Row="1"
HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto">
            <ItemsPresenter />
        </ScrollView>
    </Grid>
</Border>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

<Style TargetType="ListViewItem">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="Background" Value="#50faf1e1"/>
    <Style.Triggers>
        <Trigger Property="IsSelected" Value="True">
            <Setter Property="Background" Value="#faf1e1" />
        </Trigger>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Background" Value="#faf1e1" />
        </Trigger>
    </Style.Triggers>
</Style>

<!-- Налаштування ComboBox -->
<Style TargetType="ComboBoxItem">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#faf1e1"/>
</Style>

<Style TargetType="ComboBox">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#faf1e1"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="ComboBox">
                <Grid>
                    <ToggleButton x:Name="ToggleButton" Content=""
IsChecked="{Binding Path=IsDropDownOpen, Mode=TwoWay,
RelativeSource={RelativeSource TemplatedParent}}" ClickMode="Press">
                        <ToggleButton.Template>
                            <ControlTemplate TargetType="ToggleButton">
                                <Grid>
                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition />
                                        <ColumnDefinition Width="20" />
                                    </Grid.ColumnDefinitions>
                                    <Border x:Name="Border"
Grid.ColumnSpan="2" Background="{TemplateBinding Background}"
BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="0"
CornerRadius="5">
                                        <Border.Effect>
                                            <DropShadowEffect
Color="#AA7D26" Direction="270" ShadowDepth="5" Opacity="0.7" />

```

```

        </Border.Effect>
    </Border>
    <Border Grid.Column="0" Margin="1"
BorderBrush="#AA7D26" BorderThickness="0,0,0,0" CornerRadius="5,0,0,5"
Background="#faf1e1"/>
        <Path x:Name="Arrow" Grid.Column="1"
Fill="{TemplateBinding Foreground}" HorizontalAlignment="Center"
VerticalAlignment="Center" Data="M 0 0 L 4 4 L 8 0 Z"/>
    </Grid>
</ControlTemplate>
</ToggleButton.Template>
</ToggleButton>
    <ContentPresenter x:Name="ContentPresenter"
ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"
Content="{TemplateBinding SelectionBoxItem}" HorizontalAlignment="Left"
Margin="5, 0, 0, 0" VerticalAlignment="Center"/>
    <Popup x:Name="Popup" Placement="Bottom"
IsOpen="{TemplateBinding IsDropDownOpen}" AllowsTransparency="True"
Focusable="False" PopupAnimation="Slide">
        <Grid x:Name="DropDown"
SnapsToDevicePixels="True" MinWidth="{TemplateBinding ActualWidth}"
MaxHeight="{TemplateBinding MaxDropDownHeight}">
            <Border x:Name="DropDownBorder"
Background="#faf1e1" BorderThickness="1" BorderBrush="#faf1e1"
CornerRadius="5"/>
            <ScrollViewer Margin="4"
SnapsToDevicePixels="True">
                <ItemsPresenter />
            </ScrollViewer>
        </Grid>
    </Popup>
</Grid>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>
</ResourceDictionary>

```

B.2.4 Файл ResourceDictionaryAlternative.xaml.cs

```

<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <!-- Налаштування кнопки -->
    <Style TargetType="Button">
        <Setter Property="FontFamily" Value="Segoe UI" />
        <Setter Property="Background" Value="#211f38"/>
        <Setter Property="Foreground" Value="#f7f7f7" />
        <Setter Property="FontSize" Value="21" />

        <!-- Закруглення -->
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Border x:Name="border"
Background="{TemplateBinding Background}"
BorderBrush="#2ec2c2"
BorderThickness="2"

```

```

        CornerRadius="15">
        <Border.Effect>
            <DropShadowEffect Color="#2ec2c2" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
        </Border.Effect>
        <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
    </Border>
    <ControlTemplate.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter TargetName="border" Property="Background"
Value="#249B9B"/>
        </Trigger>
    </ControlTemplate.Triggers>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>
<!-- Налаштування кнопки -->
<Style x:Key="ButtonRound" TargetType="Button">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="Background" Value="#faf1e1"/>
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="FontSize" Value="21" />

    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border Background="{TemplateBinding Background}"
                    CornerRadius="90">
                    <Border.Effect>
                        <DropShadowEffect Color="#AA7D26" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
                    </Border.Effect>
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter Property="Background" Value="#907134"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- Налаштування текстових полів -->
<Style TargetType="TextBox">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="Foreground" Value="#f7f7f7" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#2c0540"/>
    <Setter Property="BorderThickness" Value="2" />
    <Setter Property="BorderBrush" Value="#ff70c8" />
    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="TextBox">
                <Border Background="{TemplateBinding Background}"

```

```

        BorderBrush="{TemplateBinding BorderBrush}"
        BorderThickness="{TemplateBinding BorderThickness}"
        CornerRadius="10">
        <Border.Effect>
            <DropShadowEffect Color="#ff70c8" Direction="270"
ShadowDepth="5" Opacity="0.7"/>
        </Border.Effect>
        <ScrollViewer x:Name="PART_ContentHost" />
    </Border>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

```

```

<!-- Налаштування листів -->
<Style TargetType="ListView">
    <Setter Property="FontFamily" Value="Segoe UI" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="Background" Value="#502c0540"/>
    <Setter Property="BorderBrush" Value="#ffa05c"/>
    <Setter Property="BorderThickness" Value="2"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="ListView">
                <Border Background="{TemplateBinding Background}"
                    BorderThickness="{TemplateBinding BorderThickness}"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    CornerRadius="10">
                    <Border.Effect>
                        <DropShadowEffect Color="#AA7D26"
                            Direction="270"
                            ShadowDepth="5"
                            Opacity="0.7"/>
                    </Border.Effect>
                    <Grid>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="Auto"/>
                            <RowDefinition Height="*/>
                        </Grid.RowDefinitions>

                        <GridViewHeaderRowPresenter
                            Grid.Row="0"
Columns="{Binding Path=View.Columns, RelativeSource={RelativeSource
TemplatedParent}}">

                            <GridViewHeaderRowPresenter.Resources>
                                <Style
                                    TargetType="{x:Type
GridViewColumnHeader}">
                                    <Setter
                                        Property="Foreground"
Value="#f7f7f7"/>
                                    <Setter
                                        Property="Background"
Value="#2c0540"/>
                                    <Setter
                                        Property="BorderBrush"
Value="#ffa05c"/>
                                    <Setter
                                        Property="BorderThickness"
Value="1"/>
                                    <Style.Triggers>
                                        <Trigger
                                            Property="IsMouseOver"
Value="True">
                                            <Setter Property="Background"
Value="#411E53" />
                                        </Trigger>
                                </Style>
                            </GridViewHeaderRowPresenter.Resources>
                        </Grid>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

```

```

Value="True">
Value="#411E53" />
        <Trigger      Property="IsPressed"
        <Setter Property="Background"
        </Trigger>
    </Style.Triggers>
</Style>
    </GridViewHeaderRowPresenter.Resources>
</GridViewHeaderRowPresenter>
    <ScrollViewer      Grid.Row="1"
HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto">
    <ItemsPresenter />
</ScrollViewer>
</Grid>
</Border>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

```

```

<Style TargetType="ListViewItem">
    <Setter Property="Foreground" Value="#f7f7f7" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="Background" Value="#502c0540"/>
    <Style.Triggers>
        <Trigger Property="IsSelected" Value="True">
            <Setter Property="Background" Value="#2c0540" />
        </Trigger>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Background" Value="#2c0540" />
        </Trigger>
    </Style.Triggers>
</Style>

```

```

<!-- Налаштування ComboBox -->
<Style TargetType="ComboBoxItem">
    <Setter Property="Foreground" Value="#f7f7f7" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#2c0540"/>
</Style>

```

```

<Style TargetType="ComboBox">
    <Setter Property="Foreground" Value="#f7f7f7" />
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Background" Value="#2c0540"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="ComboBox">
                <Grid>
                    <ToggleButton      x:Name="ToggleButton"      Content=""
IsChecked="{Binding      Path=IsDropDownOpen,      Mode=TwoWay,
RelativeSource={RelativeSource TemplatedParent}}" ClickMode="Press">
                    <ToggleButton.Template>
                        <ControlTemplate TargetType="ToggleButton">
                            <Grid>

```

```

        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition Width="20" />
        </Grid.ColumnDefinitions>
        <Border x:Name="Border"
Grid.ColumnSpan="2" Background="{TemplateBinding Background}"
BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="0"
CornerRadius="5">
            <Border.Effect>
                <DropShadowEffect
Color="#ff70c8" Direction="270" ShadowDepth="5" Opacity="0.7" />
            </Border.Effect>
        </Border>
        <Border Grid.Column="0" Margin="1"
BorderBrush="#ff70c8" BorderThickness="0,0,0,0" CornerRadius="5,0,0,5"
Background="#2c0540"/>
        <Path x:Name="Arrow" Grid.Column="1"
Fill="#2c0540" HorizontalAlignment="Center" VerticalAlignment="Center" Data="M
0 0 L 4 4 L 8 0 Z"/>
    </Grid>
    <ControlTemplate.Triggers>
        <Trigger Property="IsMouseOver"
Value="True">
            <Setter TargetName="Border"
Property="Background" Value="#ff70c8"/>
        </Trigger>
        <Trigger Property="IsMouseOver"
Value="False">
            <Setter TargetName="Border"
Property="Background" Value="#ff70c8"/>
        </Trigger>
    </ControlTemplate.Triggers>
</ControlTemplate>
</ToggleButton.Template>
</ToggleButton>
    <ContentPresenter x:Name="ContentPresenter"
ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"
Content="{TemplateBinding SelectionBoxItem}" HorizontalAlignment="Left"
Margin="5, 0, 0, 0" VerticalAlignment="Center"/>
    <Popup x:Name="Popup" Placement="Bottom"
IsOpen="{TemplateBinding IsDropDownOpen}" AllowsTransparency="True"
Focusable="False" PopupAnimation="Slide">
        <Grid x:Name="DropDown"
SnapsToDevicePixels="True" MinWidth="{TemplateBinding ActualWidth}"
MaxHeight="{TemplateBinding MaxDropDownHeight}">
            <Border x:Name="DropDownBorder"
Background="#2c0540" BorderThickness="1" BorderBrush="#ff70c8"
CornerRadius="5"/>
            <ScrollView Margin="4"
SnapsToDevicePixels="True">
                <ItemsPresenter />
            </ScrollView>
        </Grid>
    </Popup>
</Grid>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

<Style TargetType="Label">

```

```

        <Setter Property="Foreground" Value="#f7f7f7"/>
        <Setter Property="FontFamily" Value="Segoe UI"/>
        <Setter Property="FontSize" Value="20"/>
        <Setter Property="FontWeight" Value="Bold"/>
        <Setter Property="Effect">
            <Setter.Value>
                <DropShadowEffect          BlurRadius="20"          Color="#ffa05c"
ShadowDepth="5"/>
            </Setter.Value>
        </Setter>
    </Style>

</ResourceDictionary>

```

В.3 Проект хост частини BlackjackHost

В.3.1 Файл App.config

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
    <system.serviceModel>
        <behaviors>
            <serviceBehaviors>
                <behavior name="mexBeh">
                    <serviceMetadata          httpGetEnabled="true"
httpsGetEnabled="true" />
                    <serviceDebug
includeExceptionDetailInFaults="false" />
                </behavior>
            </serviceBehaviors>
        </behaviors>
        <services>
            <service name="Blackjack.ServiceBlackjack"
behaviorConfiguration="mexBeh">
                <endpoint address="" binding="netTcpBinding"
contract="Blackjack.IServiceBlackjack" />
                <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
                <host>
                    <baseAddresses>
                        <add baseAddress="http://localhost:8301"
/>
                    <add
baseAddress="net.tcp://localhost:8302" />
                    </baseAddresses>
                </host>
            </service>
        </services>
    </system.serviceModel>
</configuration>

```

В.3.2 Файл Program.cs

```

using System;
using System.ServiceModel;

namespace BlackjackHost
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var host = new
ServiceHost(typeof(Blackjack.ServiceBlackjack)))
            {
                host.Open();
                Console.WriteLine("Host Start");
                Console.ReadLine();
            }
        }
    }
}

```

ДОДАТОК Г
Слайди презентації

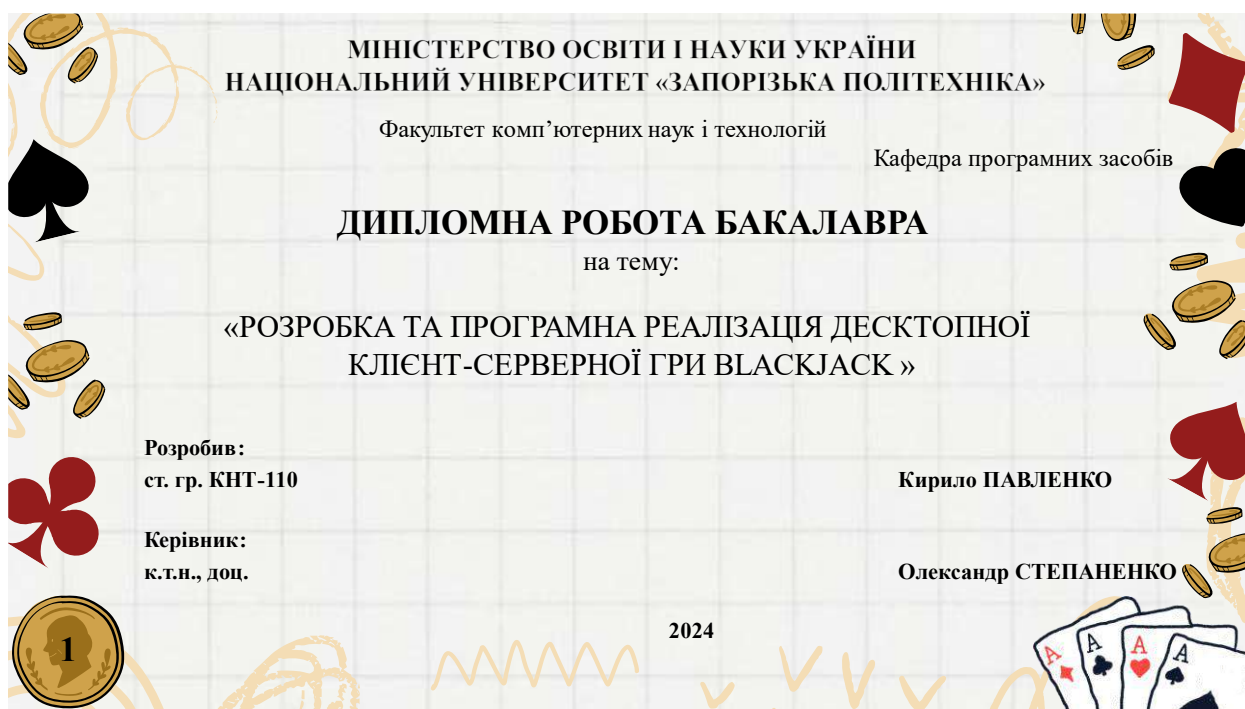


Рисунок Г.1 – Перший слайд



Рисунок Г.2 –Другий слайд



Рисунок Г.3 –Третій слайд



Рисунок Г.4 –Четвертий слайд

Порівняння існуючих аналогів

Можливості	Десктопна клієнт-серверна гра Blackjack	Онлайн гра Blackjackist	Програма Blackjack Gold	Мобільна гра Blackjack
Український інтерфейс	+	-	-	-
Реєстрація та авторизація користувачів	+	+	+	+
Режим гри проти комп'ютерного дилера	+	+	+	+
Мультиплесний режим	+	+	+	Частково
Налаштування гри	-	+	-	+
Система ставок та виграшів	+	+	+	+
Перегляд існуючих лобі	+	Частково	Частково	Частково
Перегляд правил гри	+	+	+	+
Можливість шахрайтсва	-	+	-	-

Рисунок Г.5 –П'ятий слайд

Порівняння мов програмування

Мова програмування	Продуктивність	Доступність ресурсів	Зручність розробки	Підтримка мережевої комунікації
C#	Висока	Висока	Зручна	Широка підтримка через WCF
Java	Висока	Висока	Зручна	Широка підтримка
Python	Середня	Висока	Зручна	Підтримка через різні протоколи

Рисунок Г.6 – Шостий слайд

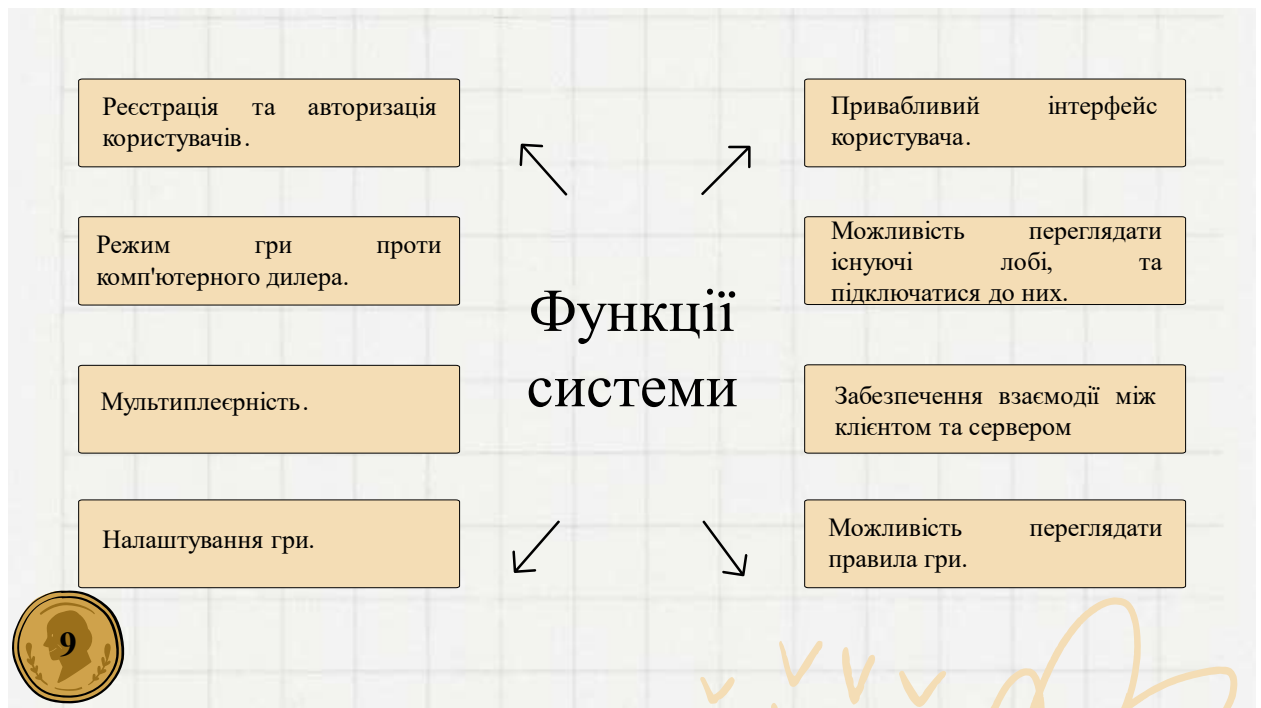


Рисунок Г.9 – Дев'ятий слайд

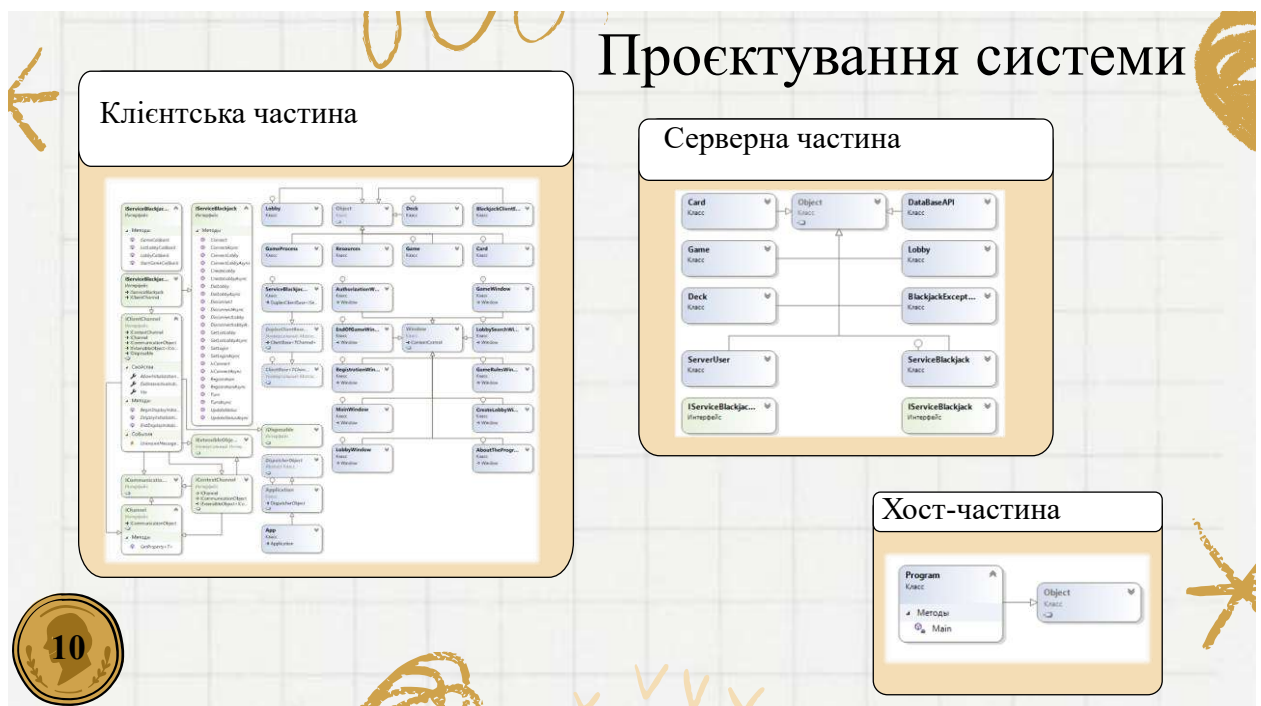


Рисунок Г.10 – Десятий слайд

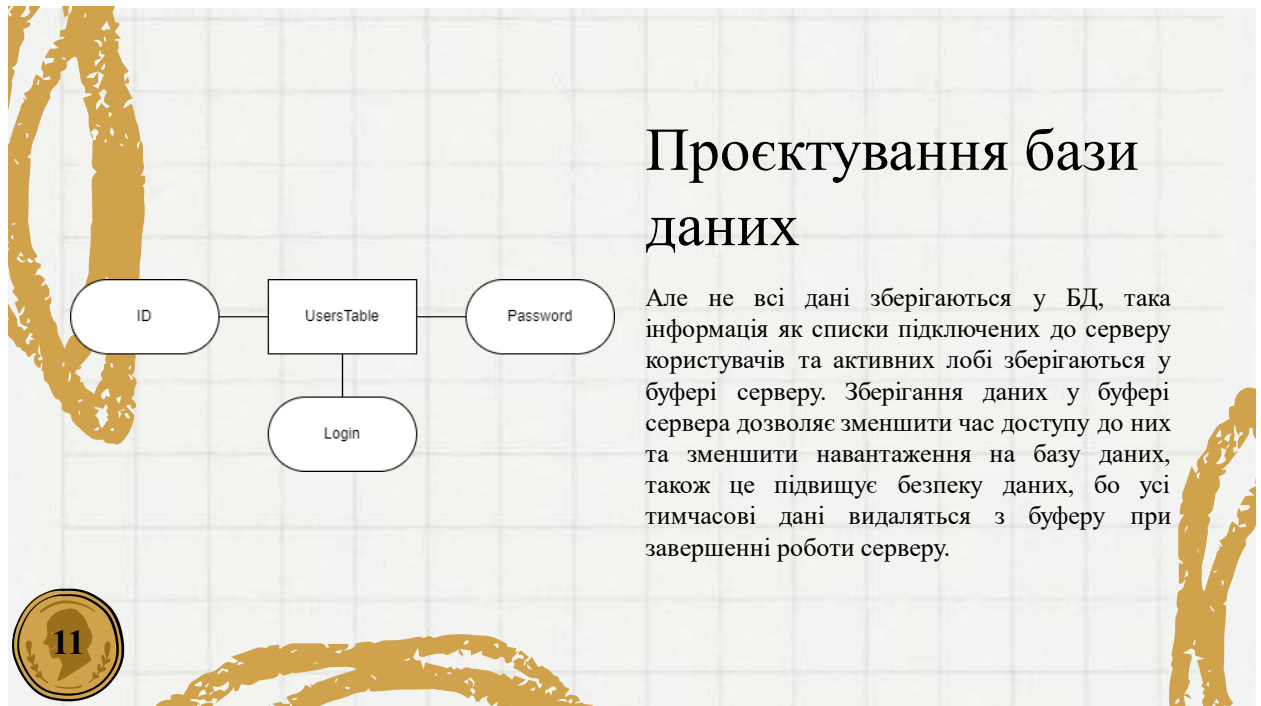


Рисунок Г.11 – Одинадцятий слайд

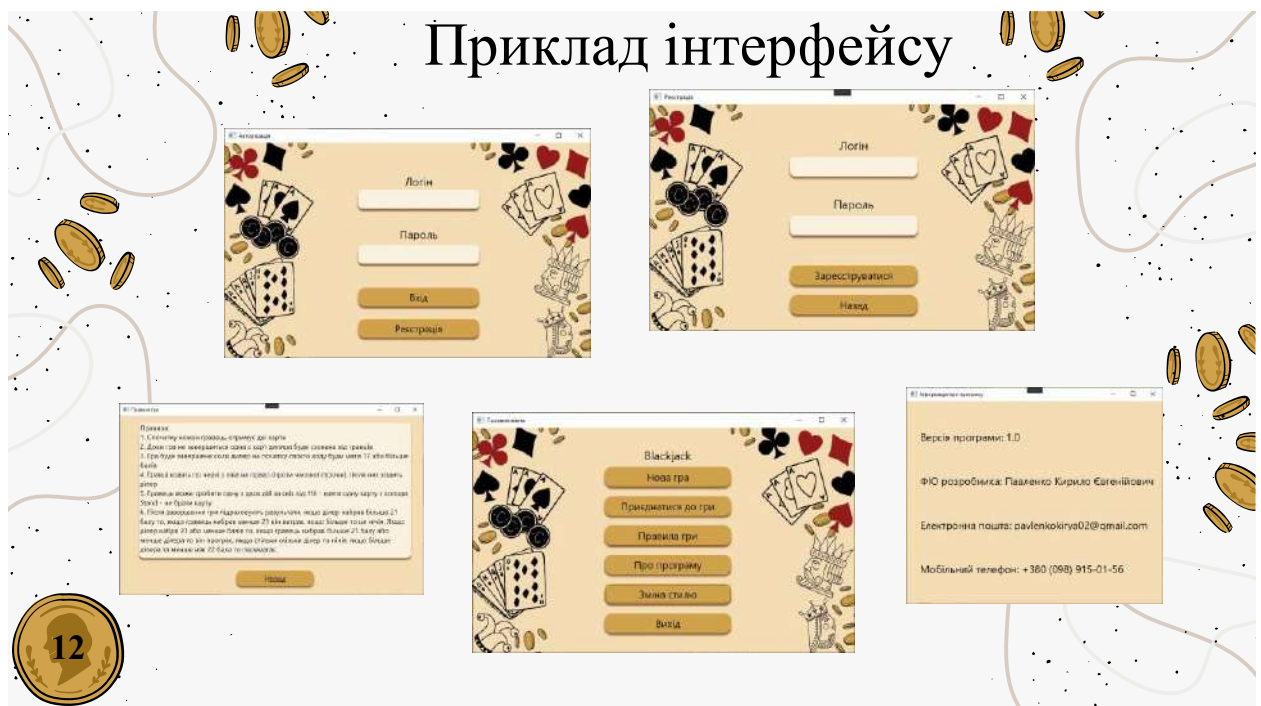


Рисунок Г.12 – Дванадцятий слайд

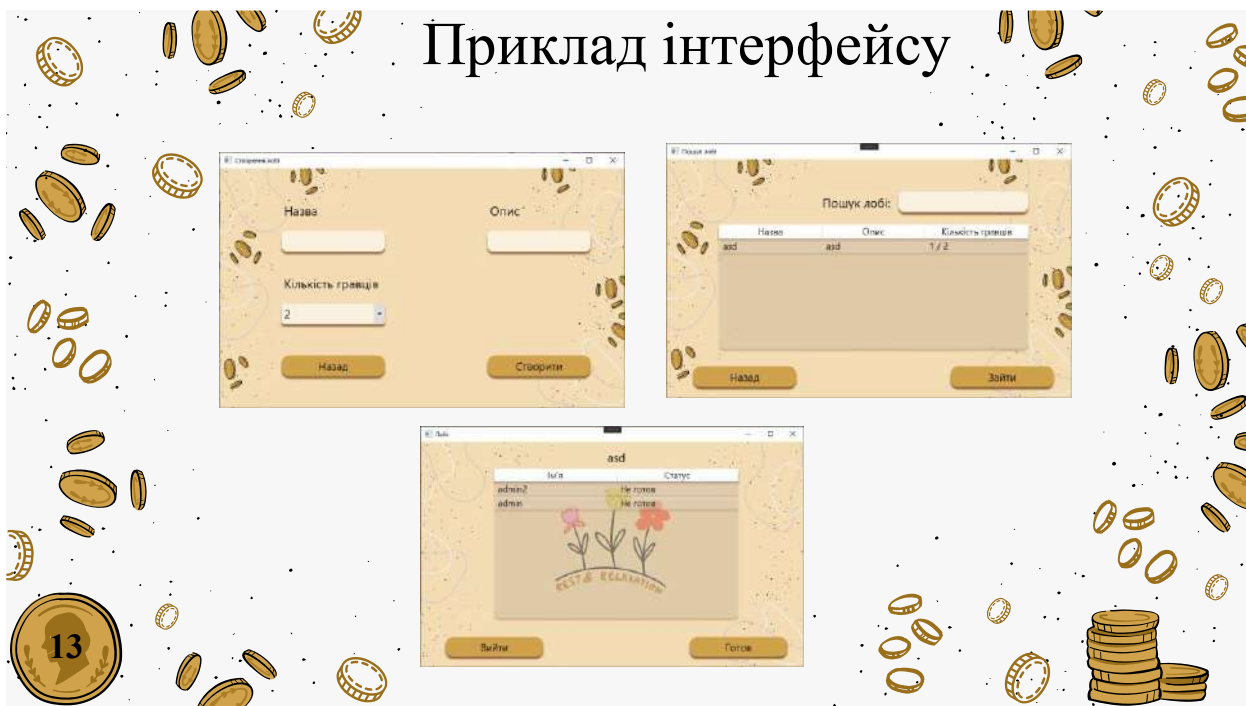


Рисунок Г.13 – Тринадцятий слайд

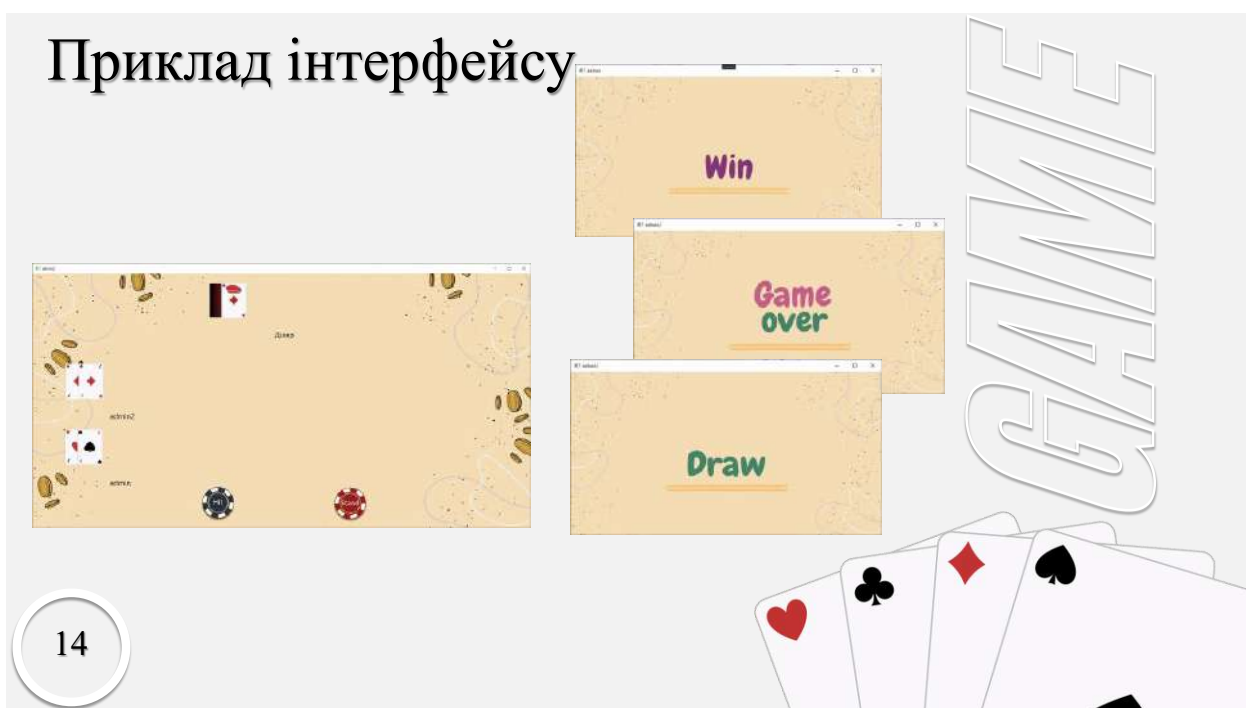


Рисунок Г.14 – Чотирнадцятий слайд

Приклад інтерфейсу



Рисунок Г.15 – П'ятнадцятий слайд

Приклад інтерфейсу



Рисунок Г.16 – Шістнадцятий слайд



Рисунок Г.17 – Сімнадцятий слайд



Рисунок Г.18 – Вісімнадцятий слайд

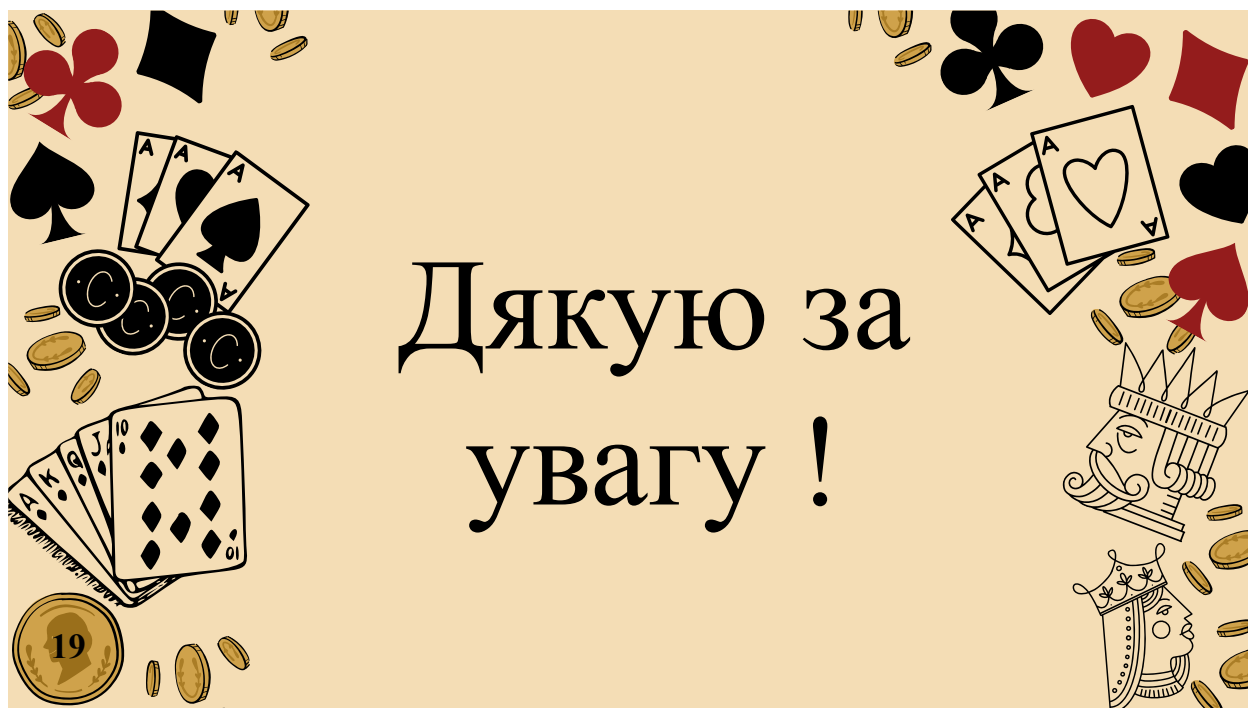


Рисунок Г.19 – Дев'ятнадцятий слайд