НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський Політехнічний Інститут імені Ігоря Сікорського»

Курсова робота

3 дисципліни «Розробка ігрових застосувань»

Poker

Керівник Виконали студенти

Коротенко А. А. групи ІП-61

Блануца Дмитро та

Данилюк Микола

Київ – 2019

**Зміст**

[Опис гри](#_30j0zll) 3

[Інтерфейс користувача](#_8x8k9u5r5bk9) 4

[Архітектура](#_y7rfo0n9nr5) 5

[Штучний інтелект](#_hx7nc92ri42j) 6

[Проблеми та шляхи їх вирішення](#_fegho2824xh6) 7

[Інтерфейс взаємодії клієнта та сервера](#_fegho2824xh6) 7

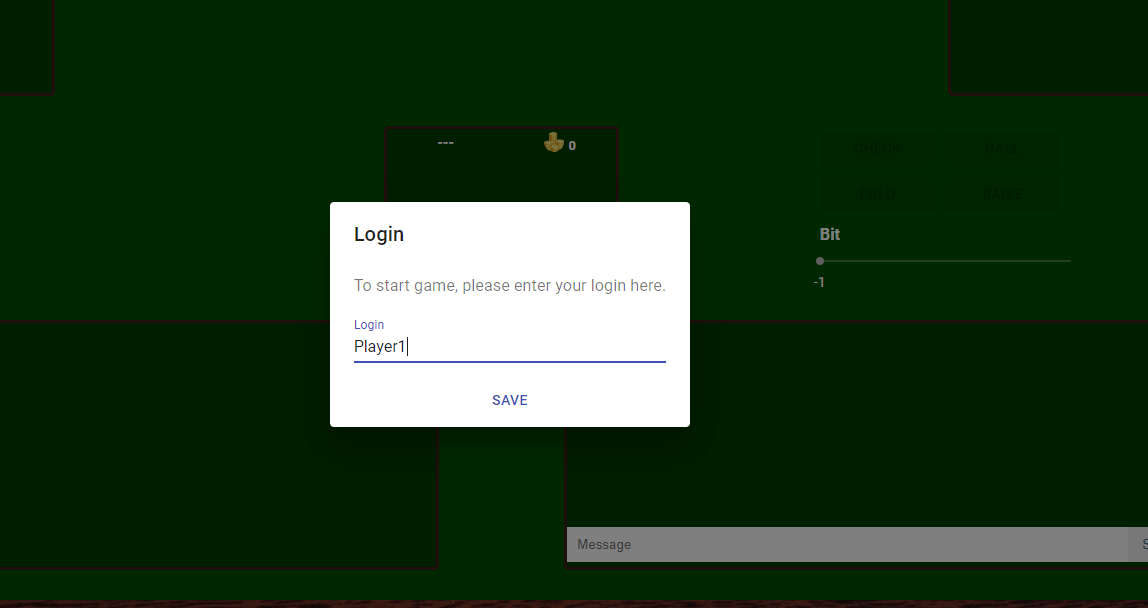
[Модель взаємодії між клієнтом та сервером](#_9o1kmprheydy) 8

# **Опис гри**

Наша команда реалізувала гру “Покер Техаський Холдем”. Обрали реалізацію у вигляді веб-застосунку, бо там зменшуються вимоги до гравця: заходиш на сайт, вводиш бажане ім’я та граєш.

Правила покеру описувати не будемо, вони всі відомі.

При першому завантаженні сторінки з грою, гравець має ввести бажане ім’я, яке перевіряється на унікальність та у разі наявності гравця з таким самим іменем виводиться відповідне повідомлення.



Гравець додається до вже створеного столу або створюється новий. Коли стіл заповнений гра автоматично починається. На початку гри кожному гравцю дається по 1000 монет. Після закінчення гри через певну кількість секунд створюється новий стіл та гра починається знову. Якщо гравець випадково закриє сторінку, а потім знову зайде на сайт то всі дані відновляться. А опоненти отримають повідомлення про відключення та перепідключення гравця.

# **Інтерфейс користувача**

1 - панель користувача;

2 - чат гравців;

3 - change log console - де відображаються всі дії користувачів;

4 - панель ставок та банку гри.

# 

# 

# **Архітектура**

Розподіл обов’язків:

**Блануца Дмитро** - займався клієнтською частиною.

**Данилюк Микола** - реалізовував логіку гри (серверну частину).

Гра була реалізована за шаблоном MVC. Ми не використовували нічого специфічного для ігрової розробки.

Model - це серверна частина, яка відповідала за всю логіку, та збереження даних користувача. Серверна частина була реалізована за допомогою Spring Boot Framework.

View - це частина що відповідає за відображення та реєстрацію дій гравця.

Відображення було реалізоване за допомогою React та Redux.

Controller - це серверна частина, яка передає дії користувача з відображення на модель. Controller реалізований за допомогою WebSocket та MVC контролерів частини Spring Framework.

Прості MVC контролери використовуються для перевірки чи гравець вже зареєстрований та перевірки унікальності імені гравця.

Для всіх інших цілей використовуються WebSocket контролери.

Для ідентифікації користувача на клієнті генерується uuid (якщо в local storage його ще не має) який надсилається на сервер та якщо користувач з таким айдішником зареєстрований, то клієнту надсилаються дані гри або гравець додається до нового столу, інакше виводиться повідомлення, що для мінімальної реєстрації потрібен лише логін користувача.

**Перевага:** простота.

**Недолік:** ідентифікатор може бути видалений після чистки браузера.

# **Штучний інтелект**

# Штучний інтелект не був реалізований. Але ми наведемо короткий опис його реалізації.

# Рішення в покері складні і залежать від безлічі параметрів і атрибутів. Ми можемо візуалізувати процес прийняття рішень у вигляді дерева рішень, де кінцеві вузли - це прийняті рішення, а гілки - різні умови. Ось спрощений приклад такого дерева покерних рішень:

# 

# 

# 

# 

# 

# **Проблеми та шляхи їх вирішення**

## **Інтерфейс взаємодії клієнта та сервера**

## Для реалізації гри треба був простий та надійний інтерфейс спілкування між сервером та клієнтом, ми вирішили використати івенти.

Клієнт з сервером спілкується за допомогою WebSocket.

Це просте та зручне рішення, бо на відображенні дані зберігаються та змінюються за допомогою Redux. Redux - це контейнер в якому всі дані додатку зберігаються в storage. Та всі всі зміни відбуваються за допомогою івентів та редюсера. Редюсер - це чиста функція, яка реалізована у вигляді switch-блоку де оброблюються всі івенти додатку.

Сервер повертає EventGroup - групу івентів які виконуються по черзі через вказаний проміжок часу. (наприклад початок гри та початок першого раунду).

EventGroup - клас, що містить список івентів та час через який вони виконуються.

Всі івенти реалізують інтерфейс Event у якого міститься лише один метод getType, який повертає тип івента. А всі івенти містяться в енамі.

## 

## **Модель взаємодії між клієнтом та сервером**

Важливою частиною роботи було визначення моделі взаємодії між клієнтом та сервером. Мною розглядались 2 варіанти: спілкування за допомогою WebSockets та Restful HTTP. WebSocket це протокол, що призначений для обміну інформацією між браузером та веб-сервером в режимі реального часу. У HTTP зазвичай клієнт надсилає запит і потім сервер відповідає з відповіддю. За допомогою WebSocket і сервер, і клієнт можуть надсилати дані незалежно один від одного (**bi-directional communication**), і одночасно(**full-duplex communication**).

**Переваги:**

* WebSocket чудово працює для випадків, коли комунікація в режимі реального часу є необхідним елементом
* Також він потрібен в сценаріях, коли повідомлення потрібно передати на кілька клієнтів одночасно

**Недоліки:** немає

У онлайн-грі в покер ці 2 аспекти дуже важливі. Саме тому, у нашому застосунку я використовую саме цю модель взаємодії.