Poker Game

CS 4540 Final Project Deliverables Report

Ajay Bagali, Jon England & Ryan Furukawa

Abstract

Our project multiplayer version of the classic Texas Hold'em (poker) game that is playable on a web browser. As such, we created a website where users can create an account and enter poker games with other users to earn chips and accumulate statistics. After creating/joining a game, users will largely interact with our game screen that includes a typical pokerboard UI with buttons to fold/call/bet and a messagebox notifying the players of the current game state (i.e the amount of the previous bet, the winner of the previous hand, etc.). These features are beneficial to a “real” user as they present an opportunity for users to play the fun, nostalgic Texas Hold'em game through an engaging, social interface.

URLs

Ajay - ec2-107-23-242-147.compute-1.amazonaws.com

Jon - ec2-54-174-113-51.compute-1.amazonaws.com

Ryan - ec2-54-173-227-116.compute-1.amazonaws.com

Introduction

As mentioned, we created a multiplayer, web version of Texas Hold'em for our project. As such, our interface includes: a home page, a game lobby page, a statistics page, and a game page. Our home page includes a form where users can either start a new poker game or join an existing game. Either way, the player will be directed to the game lobby page where they will see 1) a list of all users in the game and 2) a button to start the game. The list of users is filled through making Ajax requests to the "Data" endpoint, allowing this list of users to update without reloading the page. After selecting the "Start Game," button in the game lobby, no other users will be able to join the game, and all players in the lobby will be redirected to the "Play" screen. The lobby, again, uses an Ajax request to detect that the game has started to redirect users to the Play screen without reloading the lobby page.

The Play screen is where users will play poker games and displays common poker-game UI elements including: a player's hand, the "river" cards, the "pot," buttons to call/bet/fold, and a textbox displaying the most recent game action (mentioned above). These elements are updated through getting the state of the game using Ajax requests to the "Data" endpoint. Players interact with poker games by pressing the bet, call, and fold buttons which make changes to the game model through requests to the "Bet" endpoint. In addition to playing poker, we also created a page to display players' statistics including: the amount of chips they've won, the number of hands they've won, and the number of hands they've won when bluffing (when a player with a better hand folded).

Features

Feature	Scope	Primary Programmer	Spent	File/Function	LoC
Form for users to create and join poker games	Frontend	Jon	2 hours	Home/Index.cshtml	38
Lobby page refreshes using Ajax instead of page loads	Frontend	Jon	1 hours	Lobby.cshtml, lobby.js	50
Display user statistics in statistics page	Frontend	Jon	2 hours	Stats.cshtml	12
Draw state of game every frame using Pixi.js	Frontend	Ajay	20 hours	Play.cshtml, game.js, gameboard.js,	530
Database creation/seeding for user and game objects	Backend	Team	4 hours	SeedUser.cs, Program.cs	120
“Data” endpoint to send state of game to client	Backend	Ryan	1 hour	GamesController.cs (“Data”)	20
Game logic (“Bet” endpoint)	Backend	Ryan	20 hours	GamesController.cs (“Bet”), Game.cs, Player.cs,	730
ASP actions to create and join poker games	Backend	Jon	5 hours	GamesController.cs (“Create”,”Join”)	60

Individual Contributions

Team Member	Time Spent on Project	Lines of Code Committed
Ajay Bagali	30 hours	530
Jon England	25 hours	380
Ryan Furukawa	30 hours	810

Individual Contribution Summary

Ajay

Ajay's contributions were mainly on the frontend section of the game. This includes developing the gameboard UI using Pixi.js which can be found in the game.js, game_board.js, card_images.js, and Play.cshtml. Ajay spent a lot of time learning the Pixi.js library and integrating it with ASP.NET. Additionally, Ajay was responsible for updating the game board as the game was being played.

Jon

Jon's key responsibilities were to create the home, lobby, and statistics pages. As such, Jon didn't commit as many lines of code as his development was more view-heavy and less logic-heavy. As Jon didn't have a single, concrete responsibility like Ajay and Ryan did, Jon played a hand in helping implement the majority of our project, specifically in helping Ajay develop the gameboard UI.

Ryan

Ryan's contribution was that he implemented all of the backend game logic for poker games. As such, Ryan's development was largely in GameController.cs and Game.cs in updating and

displaying the state of a poker game as players made bets. Additionally, Ryan worked on various UI formatting issues and completed the backend logic for creating/joining games with Jon.

Performance Level

We are quite proud of the project we were able to develop. Having no prior experience with creating multiplayer games in web browsers and with PixiJS, we were able to create a stylish, fully functional poker game. Additionally, we're proud that we identified and implemented a practical solution to a new problem; we didn't just expand on a previous project and really took ownership of our product. As such, we believe our team's performance level for this project was superior. In addition to our clean user interface and extenstensive game implementation, we think the polish of our assignment shows in how Ajax requests are used throughout our program to redirect/update pages with having to refresh pages, allowing for a smooth user experience.

Summary

As we've described, we built a robust, multiuser poker game to be played in a web browser. The two primary issues we encountered when completing this project were 1) displaying the game board and 2) retrieving an updated state of the game in the view without user input. Through learning and experimenting with Pixi.js, we were eventually able to draw and manipulate game pieces easily by the time we completed this project. We also found an easy way to use a timer to make Ajax requests without user input, allowing us to update multiple screens throughout our program without reloading the entire page. These mentioned skills likely wouldn't have been addressed in extending the URC project, yet they've given us the ability and confidence to create live, multi user applications and to draw custom interfaces. As such, our project has been especially worthwhile in that it's allowed us to establish new, relevant development skills. As for the utility of our project, we're proud of how our program provides users with the leisurely, recreational experience of the class Texas Holdem game.