CS 342 Project 5 Slapjack

Multiplayer Online Game

Arshad Narmawala – anarma2@uic.edu
Jigar Patel – jpatel218@uic.edu
Clark Chen – schen237@uic.edu
Angela Timonchina – atimoc2@uic.edu
CS 342 Project 5 | Team 6

Table of Contents

FEATURES

GAMES AND OBJECTIVE

- Slapjack, a card game that puts every player's reaction to the test. The object of the game is to win all the cards of the deck that are played at the center by all players. A player can win all the cards put in the center by slapping the center when a jack is put down. In the case of a desktop game, they'd be pressing a button. Each hand is facedown so that players are unable to see their own cards. Otherwise, a player would be aware when they're about to put down a card.

CARD DEALING

- All 52 cards in a deck are dealt to each of the four players. Some players may have more cards than others. Throughout the game players are not allowed to peek at the cards in their hand and remain facedown. Game play

GAME PLAY

- Starting with a random player, every player picks a card from their hand and places it faceup to the center. The fun part of the game is when a jack gets played in the center. The first player to slap their hand on the jack wins all the cards in the center (this is done by pressing the slap button in our game). The winner of those cards adds them to their hand after shuffling them and the game continues. If a player slaps at any card in the center that is not a jack, they must give one card, face down, to the player of that card. When a player has no more cards left, they remain in the game until the next jack is turned. The player may slap at the jack in an effort to get a new pile. If the player fails to win that next pile, they are out of the game.

LANGUAGES AND FRAMEWORKS BEING USED

- The team has decided to develop with node.js which allows us to run JavaScript on the server. Furthermore, it's an event-driven, non-blocking I/O model that makes it lightweight and efficient. Express.js, a web framework that lets us structure a web application to handle multiple different http requests at a specific URL will be used in addition to that. Below are the features and their implementation details.

CLIENT/SERVER RELATIONSHIP

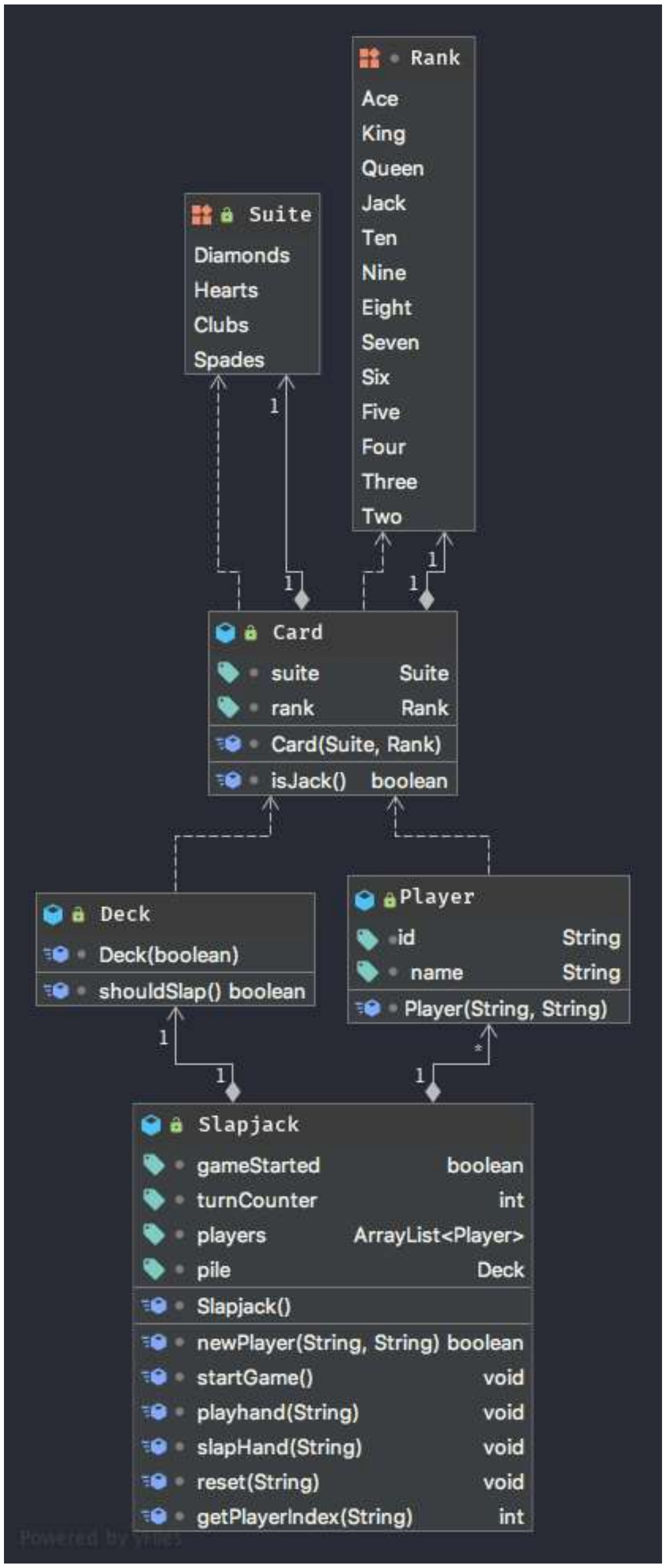
- Node.js will be used to run the JavaScript on the server side. The communication is to be handled by socket.io and their APIs.

- Socket.IO is a JavaScript library for Realtime web applications. It enables real time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. The communication between clients and server will be accomplished through strings being passed around.

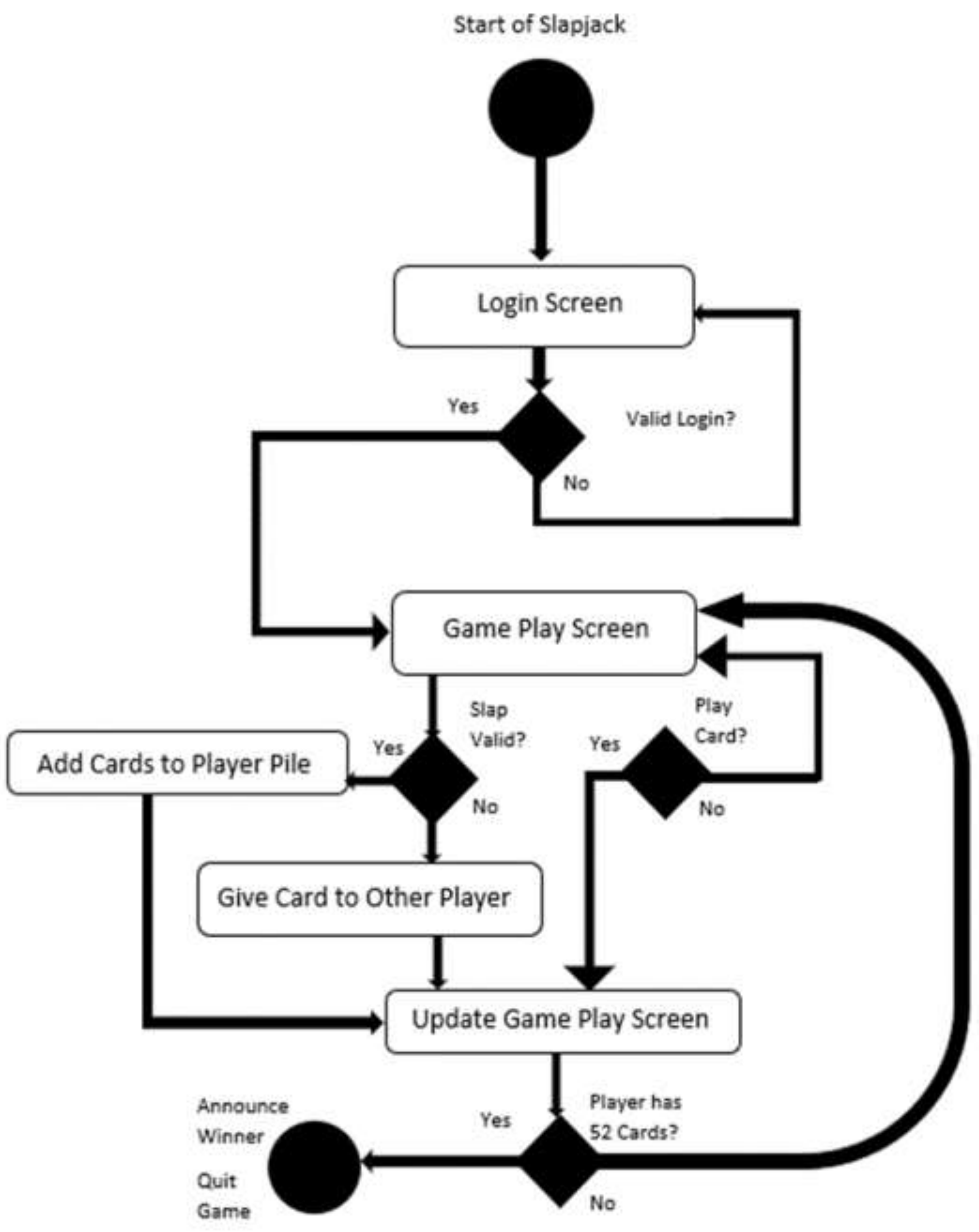
USER INTERFACE

- The user interface will display each players hand of cards facedown, buttons for playing a hand, and slapping center of table. The interface will also display cards faceup played at the center of the table.

UML DIAGRAMS



ACTIVITY DIAGRAMS



RESPONSIBILITIES

Everyone had specified responsibilities, but we helped each other to make sure everything was implemented / tested and finished.

	RESPONSIBILITY
ARSHAD	Client & Server – Set up client-side w/ React & establish connection w/ NodeJS
JIGAR	Server – NodeJS Server & Client communication via strings & accept multiple clients
CLARK	Client UI and server-side testing + project documentation (This doc, and README)
ANGELA	Server – Front end keyboard input and classes for server side

BUILD / COMPILE INSTRUCTIONS

Client setup: go into client folder, and run the following command

- npm install
- npm start

Server setup: go into ts_server folder, and run the following command

- npm install
- npm start