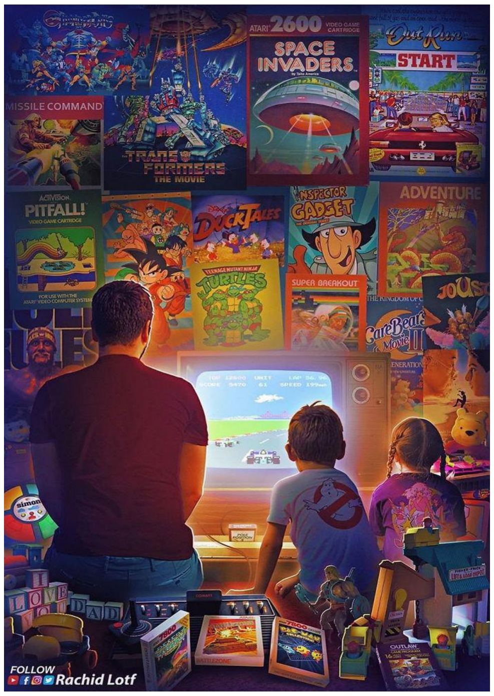
University of Puerto Rico
Mayagüez Campus
Department of Science and Computer Engineering



Team 11 Final Phase

INSO 4101: Introduction to Software Engineering
Section: 080
December 1, 2021

1. Informative Part:

1.1 General Information (Name,Place,Date)

- Human Instrumentality
- Mayagüez, Puerto Rico
- December 1st, 2021

1.2 Development Team

- Samir Y. Ali Rivera
- Kenneth R. Aponte Mendez
- Eithan M. Capella Muñiz
- Maria H. Cotto Nieves
- Jann C. Garcia Pagan
- Ezequiel O. Rosario Sepulveda

1.3 Additional stakeholders

- Janilet Rodríguez Rodríguez: Teaching Assistant
- Ricardo Vélez Acevedo: Teaching Assistant
- Marko Schütz-Schmuck: Consultant Professor
- Gamers
- Non-gamers

1.4 Current Situation

The hobby of playing video games is one that has evolved and has impacted many people's lives in the past few decades. The ability to access and play such games from almost any device nowadays gives non-gamers an entry point to such a fun and entertaining hobby. This leads newcomers to the concept of skill-based games. One can argue that games are meant to be competitive by nature, though such an argument leaves non-gamers with a not so entertaining experience which does not correlate with the main purpose of this “supposed-to-be-fun” hobby. Non-gamers at times feel discouraged or lost as to what they can play with their fellow friends or family members. Commonly available competitive skill-based games make it harder for such individuals as these are hard to learn at first. Needing to constantly play to improve or at least have a chance at winning compounds the problem. This results in inexperienced players who’ve had the need to reallocate time or prioritize other tasks, feeling that they will be unmatched in such high, skill-based games if they play less frequently. Even though player vs. player games are meant to be competitive, competitiveness should not necessarily require high skill.

1.5 Needs

- The team must understand the current situation and fully define the domain.
- The team will need a way to measure progress throughout the development process.
- ***Users need a way to enjoy playing video games without being discouraged from the idea that most games have skill as a significant factor that determines the winner.***
- Have a space for players to personally select and interact with their game sessions.
- Non-gamers should be able to play with friends or family members at any point they want and still have an entertaining and fun experience.
- Users need a way to play such games without the need to download anything or worry about updates that will inhabilitate them from playing.
- Provide users with a ***varied selection of non skill-based games all in one place for ease of access.***

1.6 Ideas

- Gather information about the interests of non-gamers.

- The team should survey the target audience to find interests and game type predilection to broaden the domain knowledge and ensure proper research, including new features that address user preferences.
- By documenting and managing time properly the team can ensure that progress is being made.
- An experience that allows a diverse range of players to interact and enjoy a variety of games on equal footing.
- To provide a wide selection of games, playable regardless of skill, with no need for a definite skill set or experience, and no time requirements to become proficient.
- Host a selection of games that do not require high skills or large time investment.
- Replace the skill-based games with other sets of activities, such as ones based on luck or strategy.
- Provide a fun, web-based environment for non gamers to play and allow them to interact with friends and family members without the need of downloading anything on their system.

1.7 Scope

- After the team assessed the domain requirements, features to be implemented were chosen that would create a playable system, while at the same time it would set the basis for potential product escalation. Aware of time constraints, a fully realized system needed an underlying implementation of the following features:
- Games were implemented and added to a library: a list where players can select the games they’d like to play.

    - Some games are fully implemented with support for up to 2 players.
    - Some games only support up to 1 player against a CPU.

- A profile system that allows players to become registered users with the following implementations:
    - Customizable accounts
    - Record database of the user’s statistics
    - Personalized leaderboard
    - Friend list

1.8 Span

- **Frameworks:**
    - **Front End:**
        - HTML, JS, CSS
        - EJS
    - **Back End:**
        - Node.js
        - Express
        - MongoDB
- **Timespan:**
    - August 27, 2021 - Dec 1, 2021 (Approximately 14 weeks)
        - Team topic selection:
            - August 27, 2021
        - Team proposal:
            - September 17, 2021
        - Front end:
            - October 6, 2021
        - Phase 1 report and demo:
            - October 8, 2021
        - Back end:
            - October 27, 2021
        - Phase 2 report and demo:

            - November 3, 2021
        - Final touches:
            - November 20, 2021
        - Final phase report and demo:
            - December 1, 2021

- Development process:
    - Research was conducted to further understand the team's domain and make sure to develop a system that will solve or at the very least help remedy the current situation.
    - The team decided on a particular list of games to be implemented into the system.
        - These will continue to be thoroughly verified and picked depending on the level of skill required, how commonly known these are, entry level skill, et al.
        - They will continue to be tested in a variety of ways to ensure that they correlate to our needs and ideas (i.e. provide a fun, fair, and entertaining experience to non-gamers) and remove any unforeseen irregularities in gameplay.
    - Implementation of the front-end.
        - What the user will see and interact with.
        - Must be understandable as possible for non-gamers and non tech savvy users.
    - Implementation of the back-end.
        - How the system works in the background.
    - Testing to ensure the system works as intended.
    - Deployment of system-to-be for further validation and verification.
    - Updates to the system-to-be depending on user feedback or results from verifications and

testing to ensure the system works and helps remedy the problem as stated at first.

- Specific details:
    - The system will be a web app for easy access.
    - Mostly based on 1 vs. 1 games.
    - Will contain luck based games to provide a fun experience rather than a “frustrating” competitive one.
        - For example, Rock-Paper-Scissors is a game that is purely based on luck. The player has an equal chance of winning or losing as it contains a 50% win/lose ratio.

1.9 Synopsis

The project is to research and develop a domain model for games that require less time investments and are not as skill-based. The domain model is expected to cover a number of phenomena such as (i) the time investment: how much time a player is willing to spend on a certain game. (ii) learning curve or skill: the amount of skill required in order to play these games. As new players, Non gamers tend to have the need for simpler games as they do not have a lot of available time to commit to learning and practicing these games. Hence there is a need to provide a standardization of games that do not require as much skill and time invested. This can be achieved by gathering information regarding the interest of these non gamers, this information can be gathered by surveys or observing the popular type of games among this group. When this information is gained, a platform can be made, one that implements a library of these games and successfully standardizes this market, providing ease of access.

2. Descriptive Part:

2.1 Descriptive Rough Domain Sketches

A college student called Timmy finds himself with a few hours a month available to engage in some casual, entertainment endeavor. Gaming piques his interest but he has a preconceived notion that all gaming requires playing experience and a significant time investment. The prospect of entering some kind of gaming environment is daunting for Timmy, since all gaming websites he knows of are very competitive and he would not be an asset to any team he could join online. He wishes to play with his friend Chelsey, which is experienced in video games, and wonders if there were some type of gaming platform or web based application where everyone participating could be on equal footing from the get go. Also, he would prefer to come back at any time and not feel lost or inadequate as a game participant, with new content to try out, to which being able to provide feedback to share his opinions and give recommendations would approve on its use.

- Brainstorming:
    - Attractive environment for non gamers.
    - Wholesome, family oriented, stress free.
    - Search for a variety of games that are non competitive.
    - Beware of brands, copyright issues.
    - Differentiate from boardgamearena.com
    - How will monetization/revenue impact design requirements down the line? Re: in-game purchases, ad displays, servers.
    - Can include luck based games.

    - To verify the team can check if players that aren’t playing as frequently still win as much as those who don’t.
    - Some of these games can be rock-paper-scissors, tic-tac-toe, connect 4, checkers, and others.
    - Games must contain a section which will include clear instructions as to how these games are played.
    - Most of these games will be 1 vs. 1, but adding games for more than one player can up the fun factor.
    - Tournaments of these games are an option.
    - The players must be able to launch private game sessions with their friends / family members.

2.2 Personas

- **Timmy**
    - Non-gamer, Stakeholder, Consultant
- **Chelsey**
    - Gamer, Stakeholder

2.3 Descriptive Domain Narrative

Let us envision an individual whom we shall call a player. This player has created an account, which hosts their custom information and statistics, after they completed their registration in the platform. The player peruses the game selection, which contains an updated set of games under different categories. To play with other players, the player must create a game lobby, which is a virtual room where players can join, and invite all desired players. Once the invited players join in, the player will choose what game everyone will participate in, and a game session starts in the lobby. During this time, players may have the choice to win or end a game if their opponent is absent for a set amount of time after the game starts. Game statistics for each player will be continually updated and players shall be able to rate the games played.

2.4 Terminology

- Device: Electronic equipment with internet connection capabilities.
- Game (Domain):
    - Software that was created with the purpose of entertainment. Allows its users to give inputs (usually through a controller or keyboard), which allows interaction with the software.
    - Game properties:
        - Genre: Criteria by which games are categorized, they can be searched via filters conforming to the user’s preference. Ex. Amount of players, board games, etc.
        - Game mode: A switch that allows users to pick from either a randomized selection of games, or search a specific game.
        - Session: the time period between when a user logs into the system or registers and until the user logs out.
        - Behavior:
            - Action: Select game mode
            - Event: Game mode is selected
    - Game entities:
        - Player score and statistics: Keeps track of the points each player has obtained during each round.
        - Game session: the time period between when a player selects a game and the game finishes.

        - Other game entities: The interactable pieces that compose the game (Example: Cards, Dice, etc).
        - Game library: A set that contains all the games available to play.
        - Behavior:
            - Action: Update Game library
            - Event: Game library is updated

- Player (Domain):
    - Potential stakeholder who is currently playing games and receiving a score. Depending on their account they have the privilege of having previous games data stored.
    - Player types:
        - Gamer: A person who has a hobby in playing games, and regularly plays games. They are commonly more experienced than the non-gamers.
        - Non-gamer: Individuals who do not spend a lot of time playing games. These stakeholders are the target audience.
        - Player functions:
            - Log in/Log out:User state x credentials -> user state
            - Register: Player info x credentials -> player info
            - Add/Remove Friend: Player x friends list x friend to be added/removed -> friends list
            - Start game: User state x game selected x position in queue -> user state
            - Create/quit game lobby: Player x user state -> user state

- Player State (Domain):
    - Qualities, location and conditions of the player at any given moment during a game session.
    - Player state types:
        - Guest: Anonymous player without an account. They have all the basic privileges, but their data is not stored.
        - Registered user: Player with a customizable account and have access to all privileges.

- Account (Requirement):
    - Hosts the customized information of the user, leaderboard position, statistics, and their profile after they complete their registration.
    - Account elements:
        - Profile: Contains general information about the user, such as the username, picture, game statistics and other features.
        - Avatar: icon or picture that represents an individual player throughout their game session. A specific page is made for players to select their desired avatar.
        - Leaderboard: List of users that are ranked by the players with the highest scores in a set time. Is personalized according to the user's records.
        - Friend List: List of players that the user has befriended. They can be personally invited to game sessions.
        - Notifications: Messages that inform the users if a match has started or ended, has new games added, and leaderboard notifications, as well as friend invitations.

        - Behavior:
            - Action: Create account profile
            - Event: profile was created
        - Behavior:
            - Action: Update user and leaderboard data
            - Event: User and leaderboard statistics are updated.
        - Behavior:
            - Action: Select avatar picture
            - Event: User is associated with a specific picture from the avatar gallery collection.

- Game Session/Lobby (Requirement):
    - Refers to a virtual room where players can join, play games together and customize the game experience as per the user's preference.
    - Game session elements:
        - Library: Set of games to be categorized by their genres. Can be updated by the following function:
        - Update library: Add/remove games available.
    - Functions available in game sessions/lobbies:
        - Set timer: Sets an optional timer that allows a user to win or end a game if their opponent does not respond for a set amount of time.
        - Invite players: Player to invite x players in session x game session -> players in session
        - Update statistics: Player's score x (leaderboard -> leaderboard) & (account -> user statistics)

- Game rating: Allows players to rate their experience within the game session to notify the team about the user’s satisfaction.
- Behavior:
    - Action: Create game lobby
    - Event: Game lobby is created
- Behavior:
    - Action: Customize and begin game session
    - Event: A game session is customized and started
- Behavior:
    - Action: Finish game and end session
    - Event: Game session ends
- Behavior:
    - Action: Players rate game session
    - Event: Game session is rated

2.5 Domain Requirements

- The system must be able to provide the users with games that require minimal skill.
- The system must have a game rating system.
- The system must contain a certain number of games that are mostly luck based (e.g. rock-paper-scissors).
- The system will provide an option for private lobbies such that users can play with family and friends.
- The system must provide a means to store the players’ basic profile information.
- The system must provide a means to the player to select a specific game or select a game randomly.
- The system will allow the players to play any of the games in the library randomly.

- The system will provide a way to create and host a game lobby.
- The system must provide a means to notify players about the status of their wins and losses.

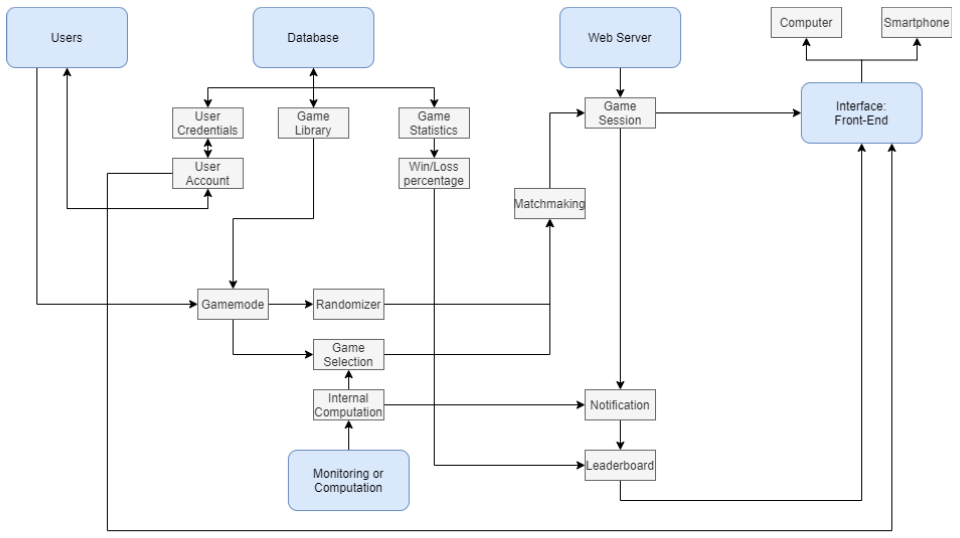
2.6 Interface Requirements

- The system must scale and display properly on any device capable of inputting text to the screen or clicking buttons.
- The system must provide all the appropriate information once the player finishes a game.
- The system must display a list of all the top players; as well as their top games and win/lose ratio.
- The system must provide a means to the player to see their profile information.
- The system will display a set of the available games.
- The system must display the leaderboard to players.

2.7 Machine Requirements

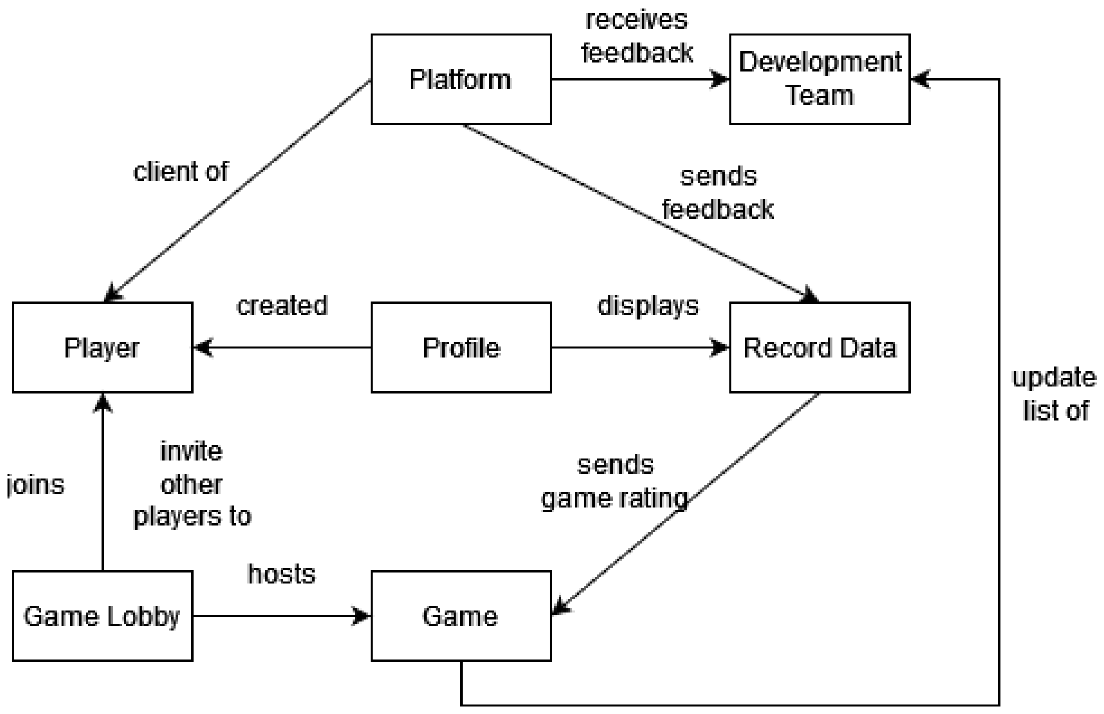
- The system-to-be must have the ability to be used on any web browser without the need of any downloads.
- The system-to-be must be able to handle at least 10,000 users simultaneously.
- The database used must store large amounts of information such as all of the account information from every user with at least 15% of space left.
- The database must be able to send user information and other types of data in at least 500ms.
- The system-to-be must have the ability to allow users from mobile and web browsers with the addition of cross platform.
- System-to-be must be able to connect users in 500 ms or less in 90% of the cases.

2.8 Software Architecture Design



- This diagram represents the projected organization of the system, all the main components and how these interact with each other, and the principles that are used to design the software. As seen, Users mainly interact with the Database and the Webserver. Most internal computations regarding user choices or game-wise will be handled by JavaScript or the computational module. This would be seen as the back-end, after these are handled we send the user their profiles and game session to their device, which will refer to the front-end.

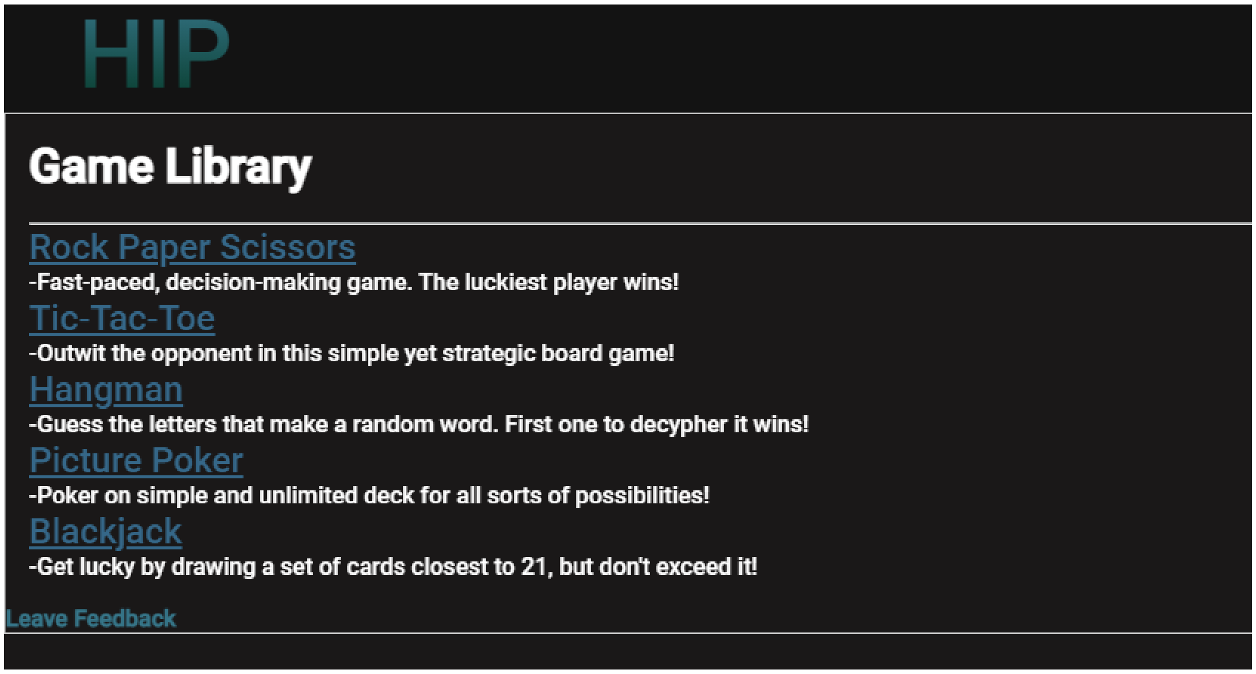
2.9 Software Component Design



- The platform will manage player sessions through game lobbies, where games can be played. Data will be recorded and displayed in the respective player profiles.
- Record data will also be used as feedback for the development team, and will also allow users to recommend more games for the team to add.

2.10 Selected Fragments of Implementation

- Play Page
    - This page contains all of the games that users can (or will be able to) play with other players.


HIP

Game Library

Rock Paper Scissors
-Fast-paced, decision-making game. The luckiest player wins!

Tic-Tac-Toe
-Outwit the opponent in this simple yet strategic board game!

Hangman
-Guess the letters that make a random word. First one to decypher it wins!

Picture Poker
-Poker on simple and unlimited deck for all sorts of possibilities!

Blackjack
-Get lucky by drawing a set of cards closest to 21, but don't exceed it!

Leave Feedback

Figure 1: Play Page

- Login Page
    - A page that gives users the ability to login to our website using their account, created in the registration page. With an account, users will gain access to more features, such as a win-loss tracker, most played games, and much more.
    - Their credentials will be loaded from the database if the input equals that of his/her username and password. Else, one could register a new account on the registration page.
    - This data is properly saved on our database in MongoDB.

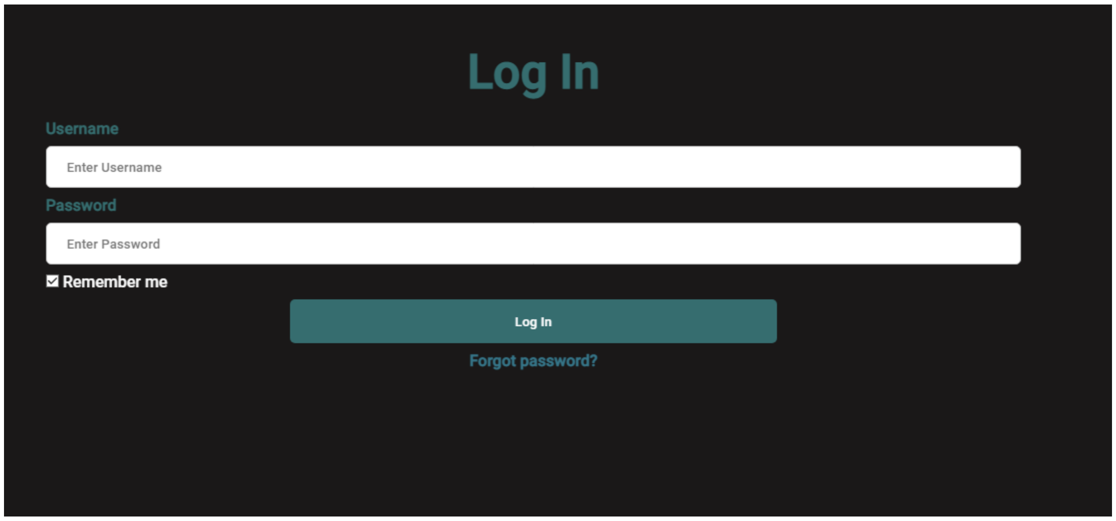
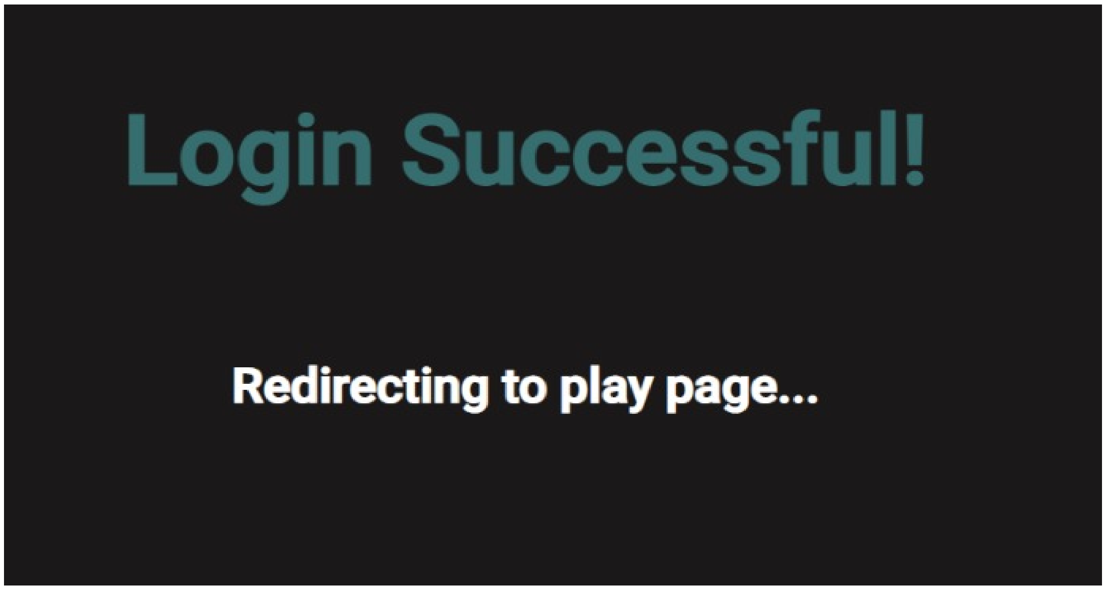


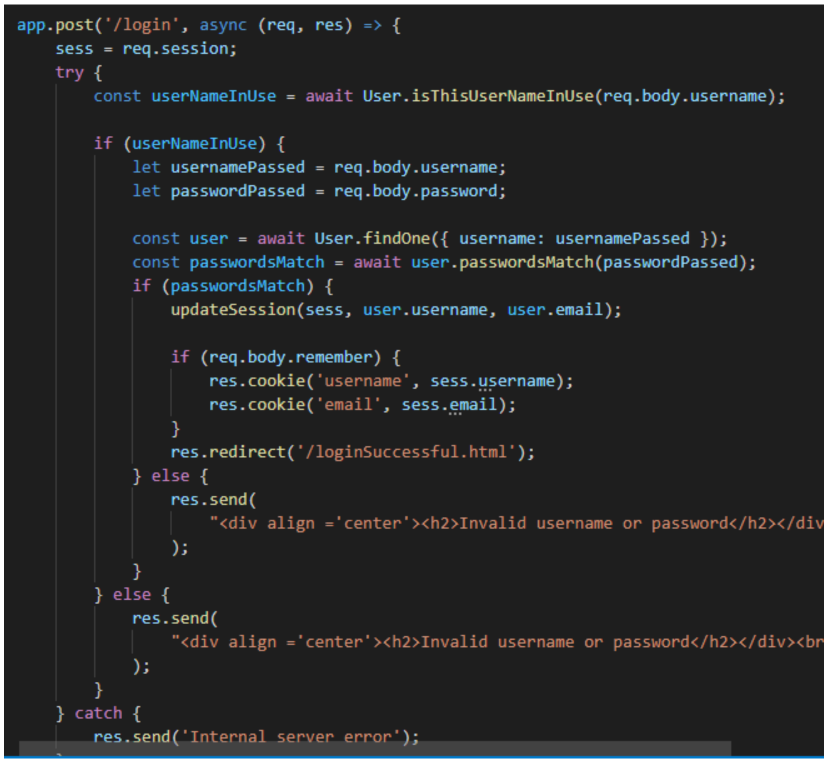
Figure 2: Login Page



Login Successful!

Redirecting to play page...

Figure 3: Login Success



```
app.post('/login', async (req, res) => {
    sess = req.session;
    try {
        const userNameInUse = await User.isThisUserNameInUse(req.body.username);

        if (userNameInUse) {
            let usernamePassed = req.body.username;
            let passwordPassed = req.body.password;

            const user = await User.findOne({ username: usernamePassed });
            const passwordsMatch = await user.passwordsMatch(passwordPassed);
            if (passwordsMatch) {
                updateSession(sess, user.username, user.email);

                if (req.body.remember) {
                    res.cookie('username', sess.username);
                    res.cookie('email', sess.email);
                }
                res.redirect('/loginSuccessful.html');
            } else {
                res.send(
                    "<div align ='center'><h2>Invalid username or password</h2></div
                );
            }
        } else {
            res.send(
                "<div align ='center'><h2>Invalid username or password</h2></div><br
            );
        }
    } catch {
        res.send('Internal server error');
```

Figure 4: Login Validation

- Registration Page
    - On this page, users are able to create an account on our website by using their email address and a password to access their account with.
    - Once an email is registered, it will be added to the MongoDB database and, alongside the username, cannot be registered again.

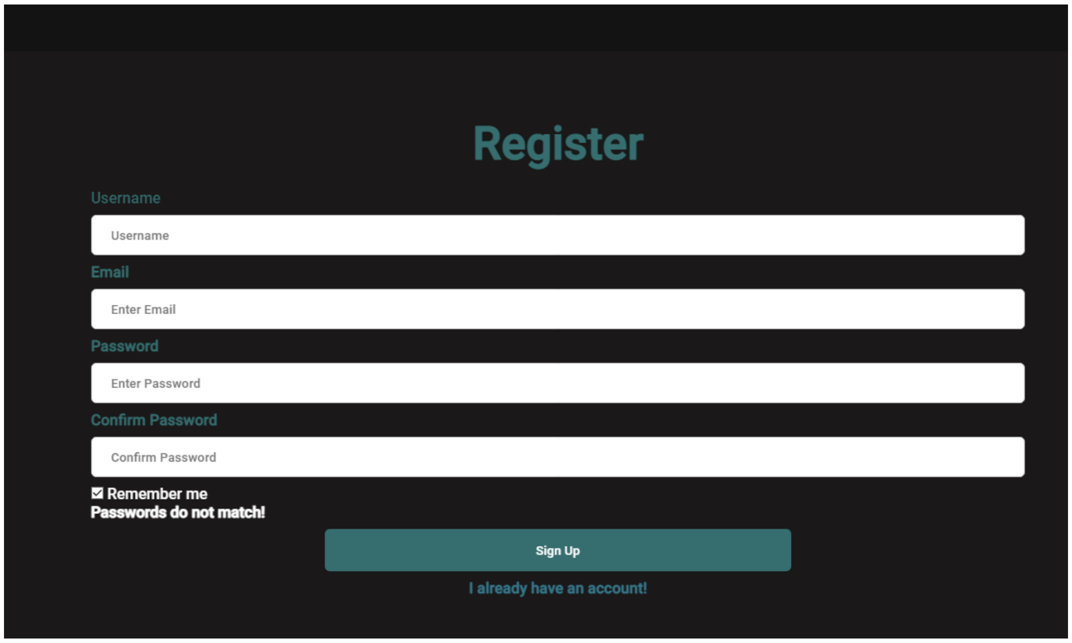
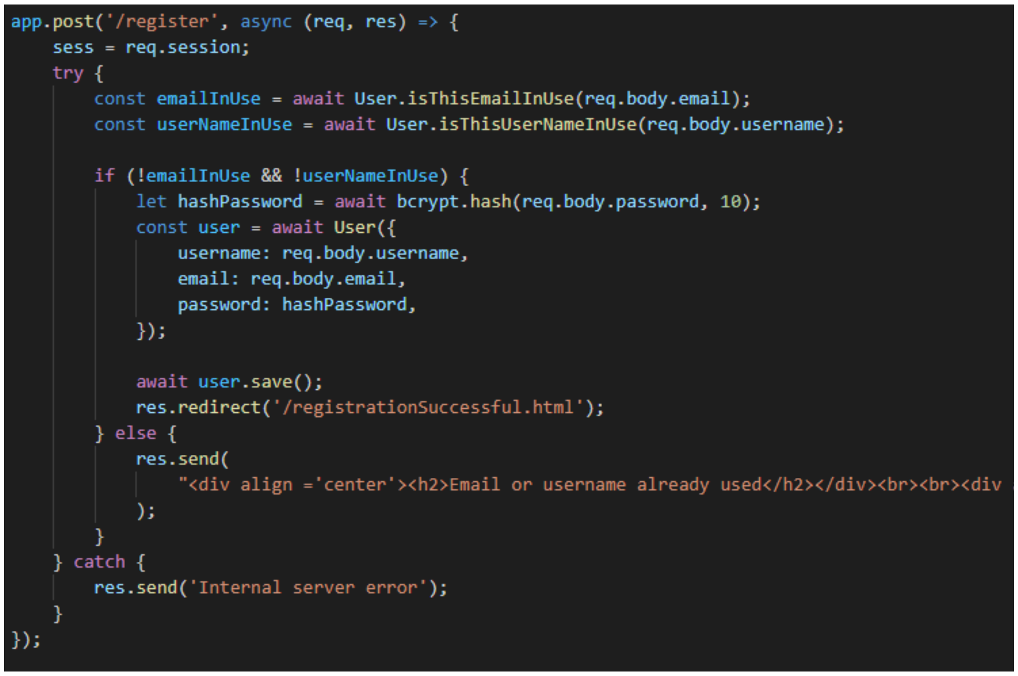


Figure 5: Registration Page



```
app.post('/register', async (req, res) => {
    sess = req.session;
    try {
        const emailInUse = await User.isThisEmailInUse(req.body.email);
        const userNameInUse = await User.isThisUserNameInUse(req.body.username);

        if (!emailInUse && !userNameInUse) {
            let hashPassword = await bcrypt.hash(req.body.password, 10);
            const user = await User({
                username: req.body.username,
                email: req.body.email,
                password: hashPassword,
            });

            await user.save();
            res.redirect('/registrationSuccessful.html');
        } else {
            res.send(
                "<div align ='center'><h2>Email or username already used</h2></div><br><br><div
            );
        }
    } catch {
        res.send('Internal server error');
    }
});
```

Figure 6: Registration Validation

- About Us and FAQ page
    - Web pages containing information about the development team. As development continues and feedback from other stakeholders occurs due to

them, these pages will be updated for the benefits of any visitors.

- Dynamic Navigation Bar
  - When a user is logged in, the login and registration buttons become Profile and Logout buttons, respectively. When a user logs out, the login and registration buttons return to the navigation bar.

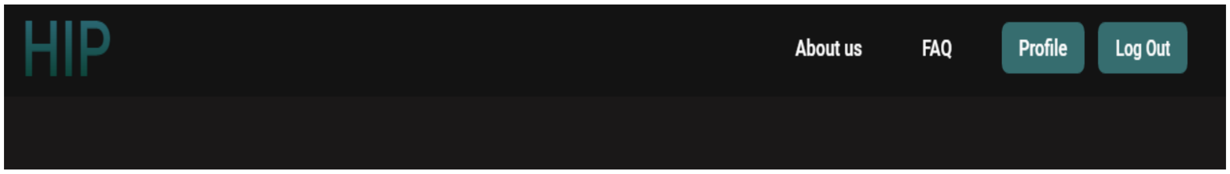
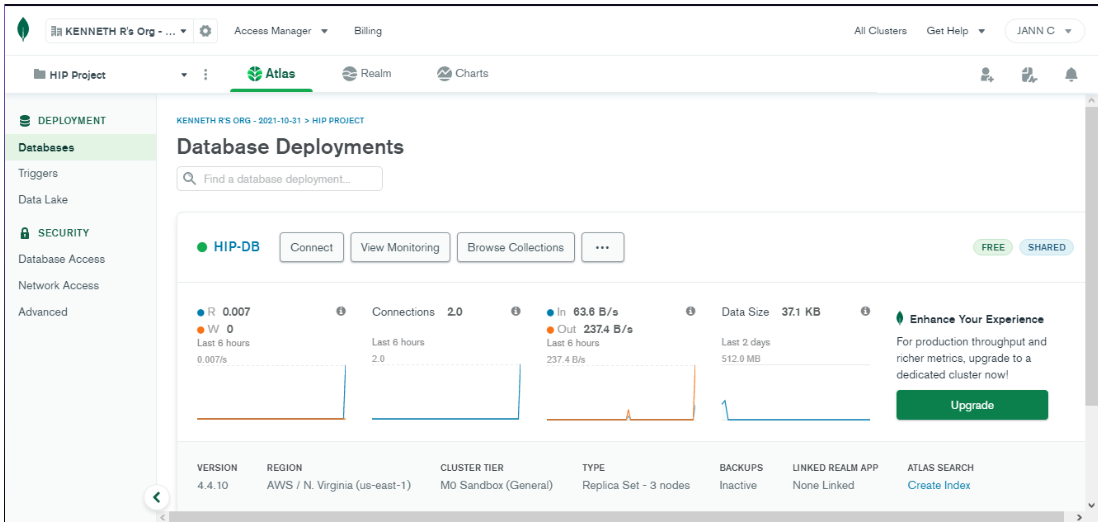


Figure 7: Dynamic Navigation Bar

- User Database
  - User data is stored in our database hosted on the cloud by MongoDB. The users' username, email, and hashed password are saved onto the database. Precautions were taken to avoid saving the unhashed password to the database, to avoid the risk of passwords being compromised. All of the data was stored using schemas, meaning each user profile under the users table is some sort of document itself.



KENNETH R'S ORG - 2021-10-31 > HIP PROJECT

Database Deployments

Find a database deployment...

HIP-DB

R 0.007 W 0 Last 6 hours 0.007/s

Connections 2.0 Last 6 hours 2.0

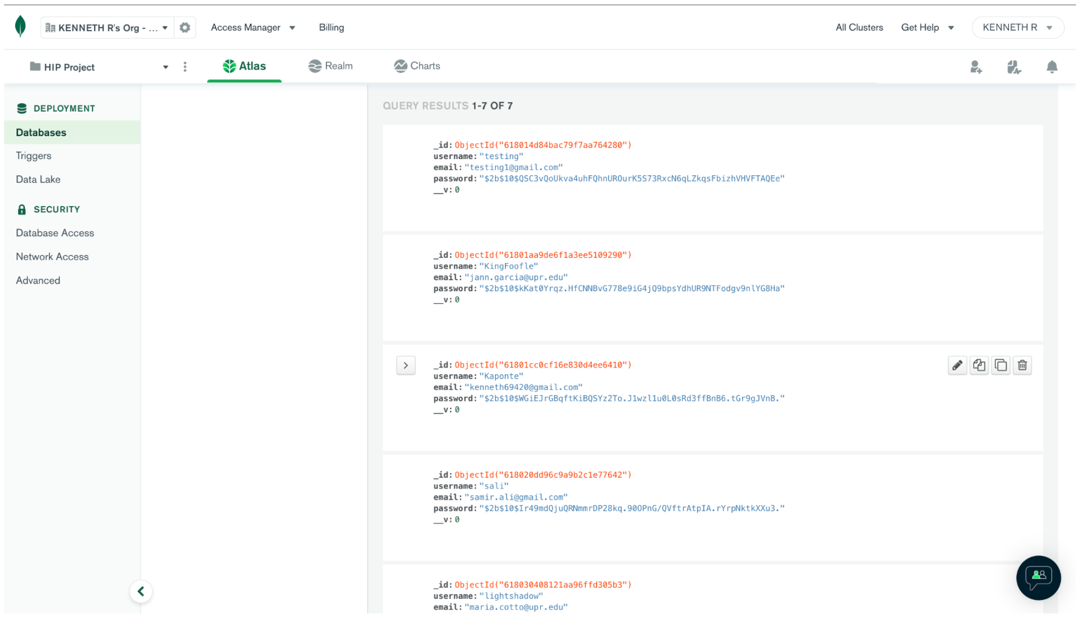
In 63.6 B/s Out 237.4 B/s Last 6 hours 237.4 B/s

Data Size 37.1 KB Last 2 days 512.0 MB

Enhance Your Experience
For production throughput and richer metrics, upgrade to a dedicated cluster now!

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
4.4.10	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

Figure 8: MongoDB Database



QUERY RESULTS 1-7 OF 7

```
_id: ObjectId("618014d84bac79f7aa764280")
username: "testing"
email: "testing1@gmail.com"
password: "$2b$10$QSC3vQoUkva4uhFQhnUROurK5S73RxcN6qLZkqsFbizhVHVFTAQEe"
__v: 0

_id: ObjectId("61801aa9de6f1a3ee5109290")
username: "KingFoofle"
email: "jann.garcia@upr.edu"
password: "$2b$10$kKat0Yrqz.HfCNNBvG778e9iG4jQ9bpsYdhUR9NTFodgv9nlYG8Ha"
__v: 0

_id: ObjectId("61801cc0cf16e830d4ee6410")
username: "Kaponte"
email: "kenneth69420@gmail.com"
password: "$2b$10$WGiEJrGBqftKiBQSYz2To.J1wzl1u0L0sRd3ffBnB6.tGr9gJVnB."
__v: 0

_id: ObjectId("618020dd96c9a9b2c1e77642")
username: "sali"
email: "samir.ali@gmail.com"
password: "$2b$10$Ir49mdQjuQRNmmrDP28kq.90OPnG/QVftrAtpIA.rYrpNktkXXu3."
__v: 0

_id: ObjectId("618030408121aa96ffd305b3")
username: "lightshadow"
email: "maria.cotto@upr.edu"
```

Figure 9: User Database

- Player Profile Page
    - All personalized information about the user will be summarized in the player profile page. The user may upload an image as their avatar, that will be shown throughout the site. This avatar also serves as a shortcut to the player profile page. Some of the information displayed on the

Player Profile page may be customized, for example, the avatar, a personalized phrase, e-mail address and their password. Registration date, game statistics and username are not customizable, however.

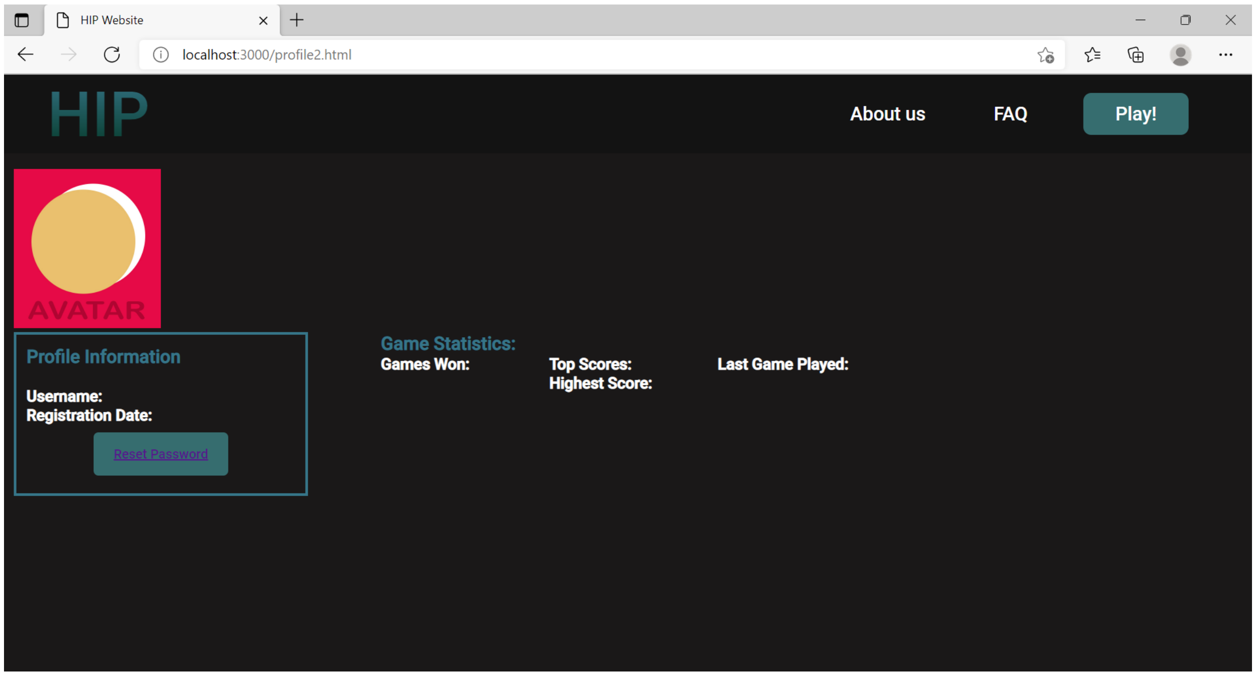


Figure 10: User Profile

- Avatar Selection page:
    - As part of the customizable features, the user will have access to a predetermined selection of pictures that will represent the user throughout the system to be.

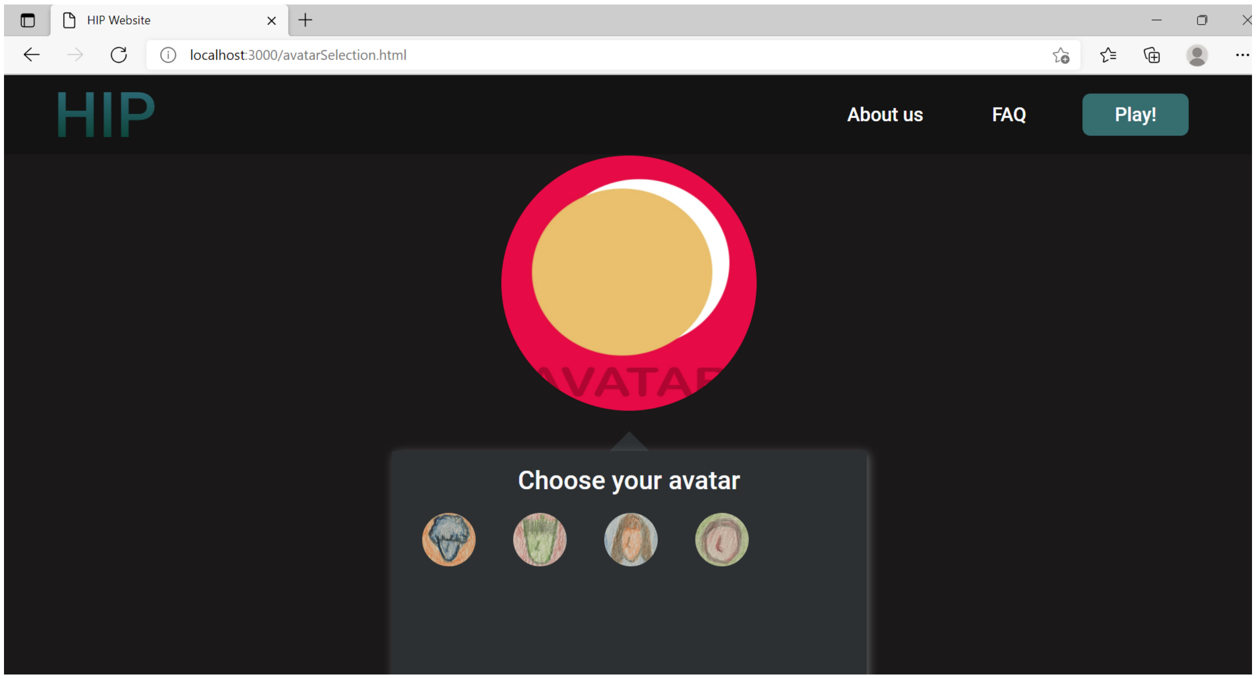


Figure 11: Avatar Selection Screen

- Forgot Password page
  - When a user forgets a password, they will be directed to this page where they should enter the email address they used to register in order to receive a temporary password.

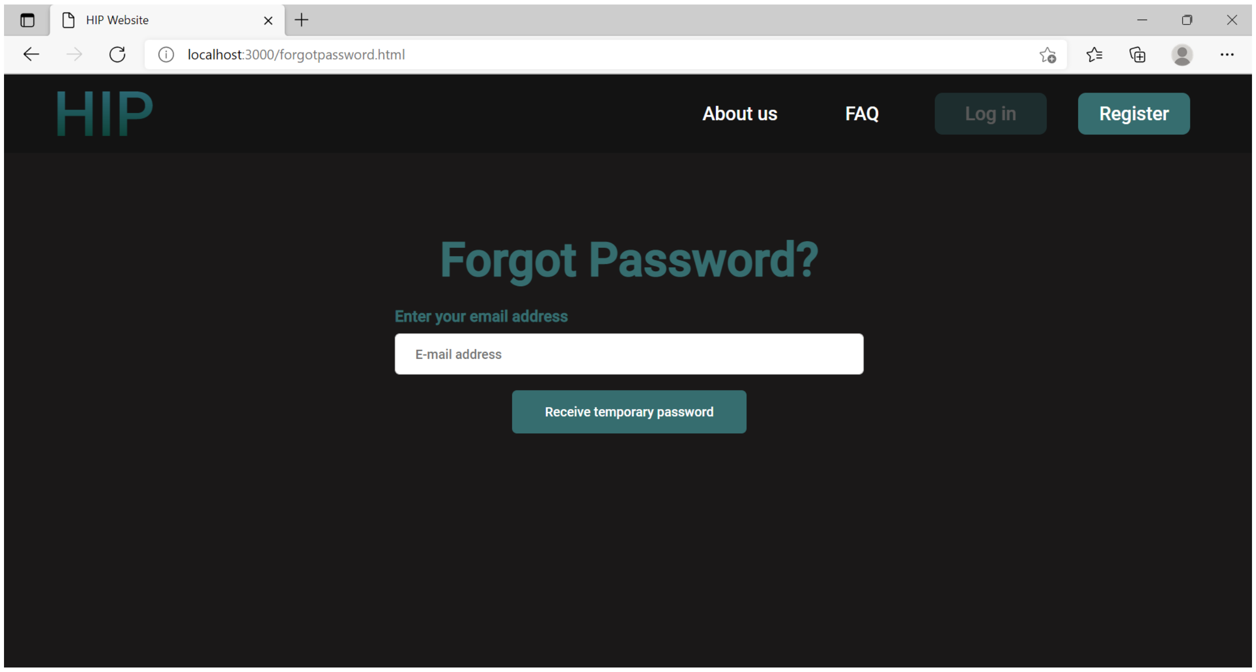


Figure 12: Forgot Password Page

- Reset Password page
    - In case a user has a compromised password, the Reset Password page requires they input the old password and a new password, confirm the new password, and the system shall reset it to the new password.
    - In case a user forgot their password, once they visit the Forgot Password page and receive a temporary password, they will be redirected to the Reset Password page where they can reset the temporary password to a new one.

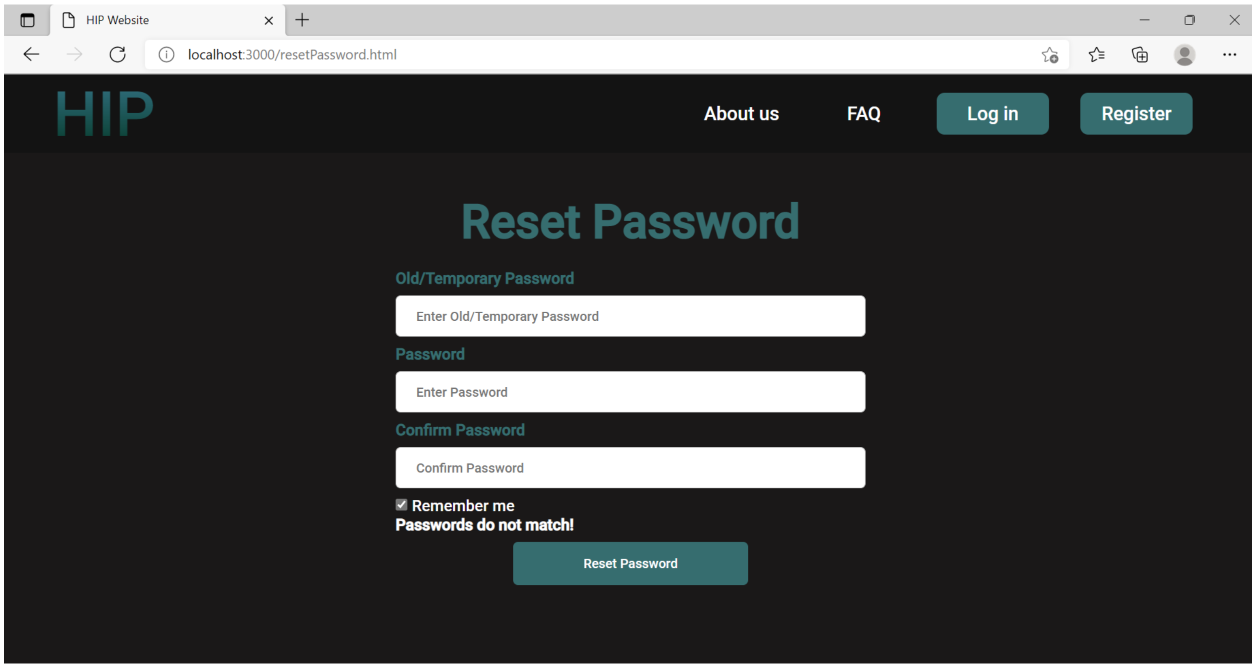


Figure 13: Reset Password Page

- Games:
  The team implemented three initial games that fit the criteria for the system, that is, games that do not require time investment or a particular skill for all players to be matched with each other in a fair way.

The games chosen for the final phase implementation are: rock-paper-scissors, tic tac toe and blackjack.

- Rock-paper-scissors: this is an online version of the classic hand game, where two players take turns to choose from three different shapes: rock, paper or scissors. Once the first player selects their shape, the system allows the opponent to select a shape. The game session ends by awarding a victory to one of the players or, if players choose three times the same shapes both, declaring a tie.
    - Our version of the game allows simultaneous interactive play between two players
    - Once the first player has made their selection, it is unchangeable until their opponent makes their choice and the bout is decided.
    - The winner is declared after three bouts are played.

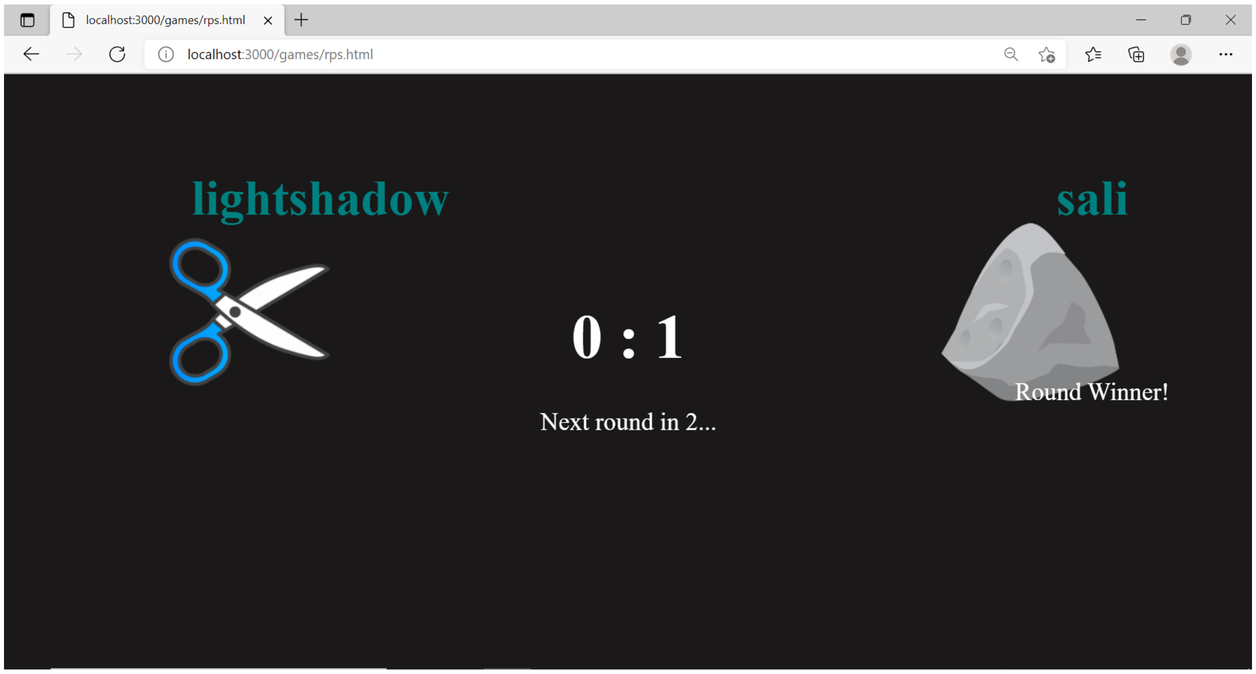


Figure 14: Rock Paper Scissors

- Tic tac toe:
    - A turn based classic game for two players.
    - This online version recreates a 3x3 grid where a player and their opponent take turns until one wins or there is a tie.
    - Players choose between playing as an X mark or as an O mark. The first player to place three of their marks in a horizontal, vertical, or diagonal row is the winner.
    - Our implementation makes use of the lobby feature to match random players who take turns to place their marks in one match.
    - Once a game is finished, it may be reset by any of the players.

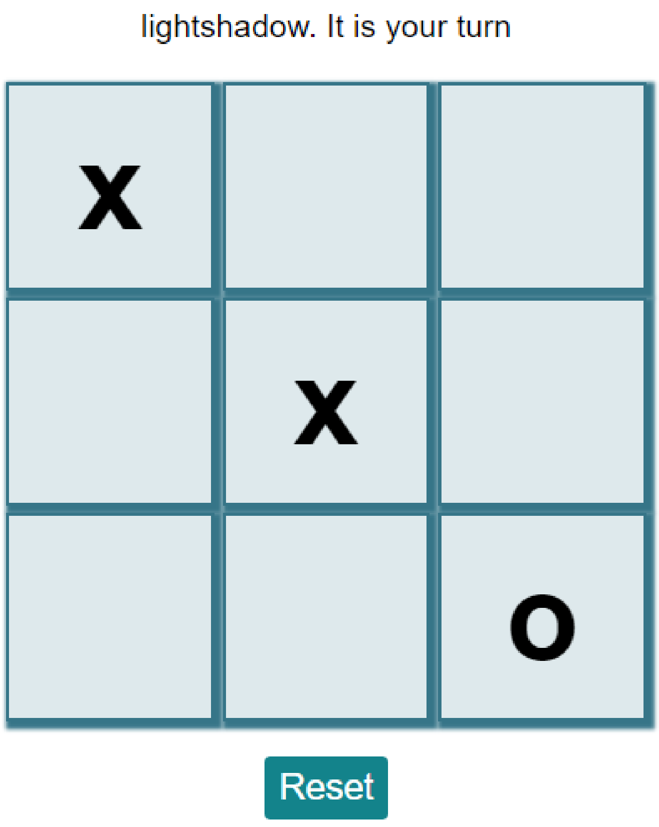

lightshadow. It is your turn

Figure 15: Tic Tac Toe

- Blackjack:
  As a casino style game, blackjack is widely known and much of its playing strategy is dependent on luck.
  - After selecting the blackjack game in the Play Page, the player goes against the dealer and an opponent to get the highest hand of cards that does not surpass 21 in value. The player with the highest number below or equal to 21 wins.
  - This game is currently designed to be played against a computer for singleplayer, as the other games are only playable against other users.
  - Both players begin with a card of a random number. When a player Hits, the table draws a number to add to your card. They stop adding when they Stay at their final result.

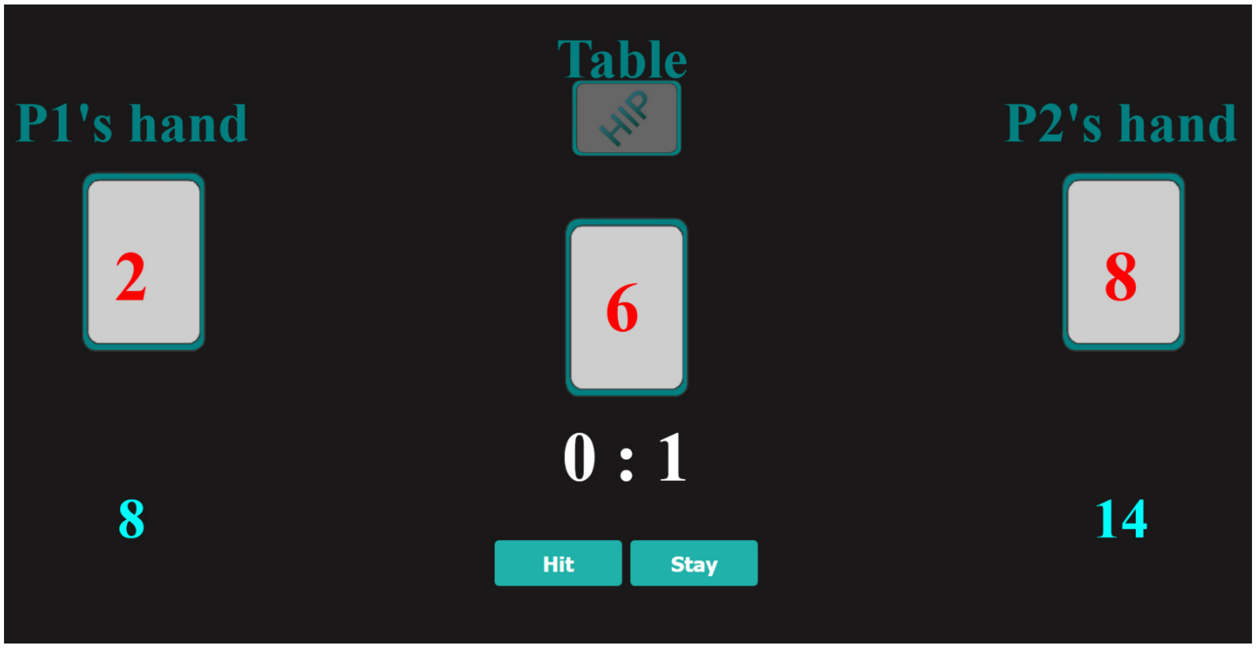


Figure 16: Blackjack

- Connect 4:
  - The team also implemented an online version of the game Connect 4 for two players.
  - Each player is assigned a color and they arrange tiles according to the scheme.
  - The objective is to be the first to form a horizontal, vertical, or diagonal line of that player's colored squares.

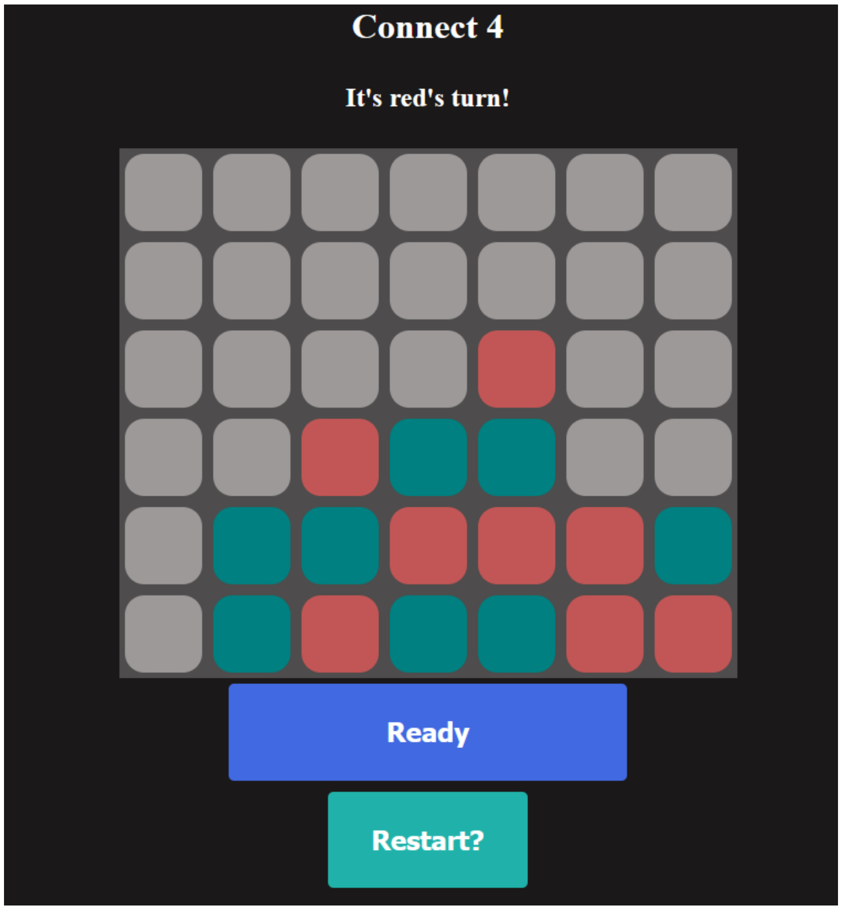


Figure 17: Connect 4

- Matchmaking Support for Select Games
  - Some games have a fully implemented matchmaking system. With that, users are now able to play certain games with other users.

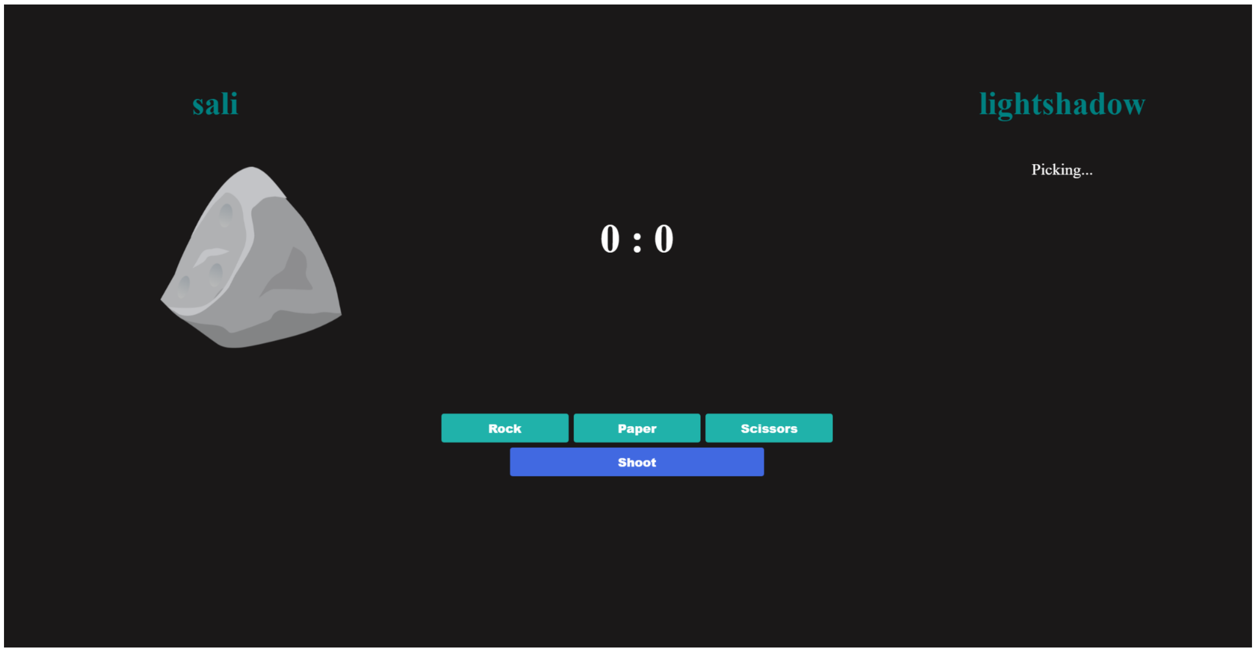


Figure 18: 2-Player Rock Paper Scissors

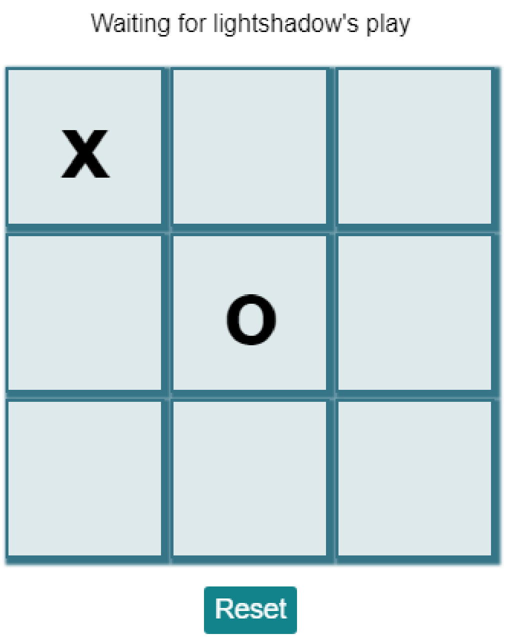


Figure 19: 2-Player Tic-Tac Toe

- Feedback
    - To ensure that our provided games are of expected quality, a feedback system was implemented. Here, users can specify what game they would like to rate, and provide feedback and a 5 star rating.
    - This feedback is sent to our database for later reading.

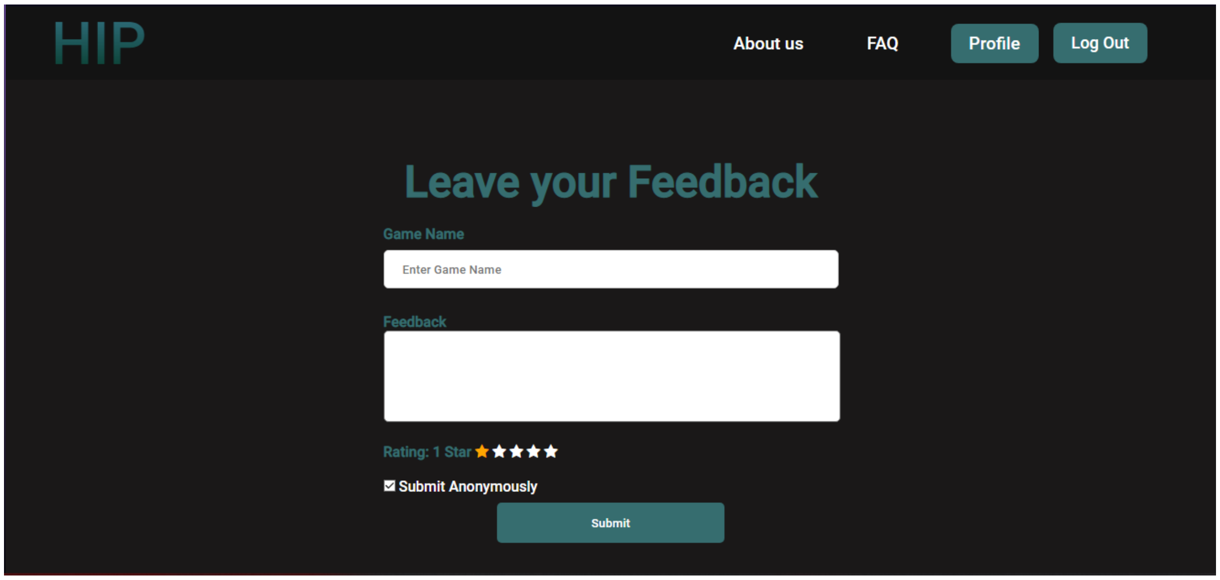
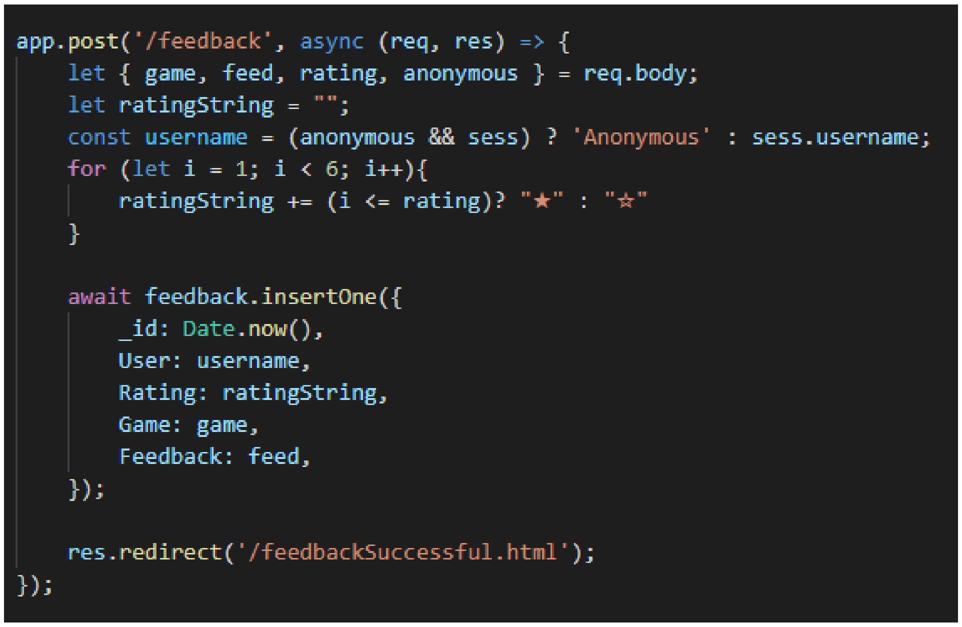


Figure 20: Feedback Page



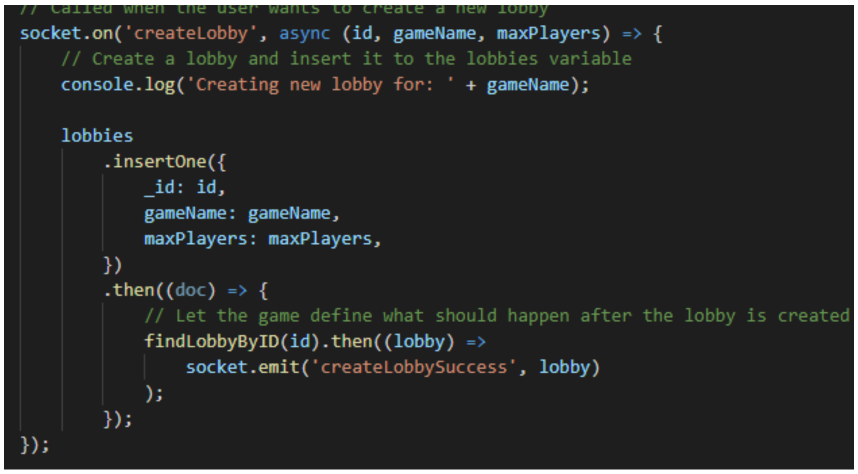
```
app.post('/feedback', async (req, res) => {
    let { game, feed, rating, anonymous } = req.body;
    let ratingString = "";
    const username = (anonymous && sess) ? 'Anonymous' : sess.username;
    for (let i = 1; i < 6; i++){
        ratingString += (i <= rating)? "★" : "☆"
    }

    await feedback.insertOne({
        _id: Date.now(),
        User: username,
        Rating: ratingString,
        Game: game,
        Feedback: feed,
    });

    res.redirect('/feedbackSuccessful.html');
});
```

Figure 21: Submitting Feedback to the Database

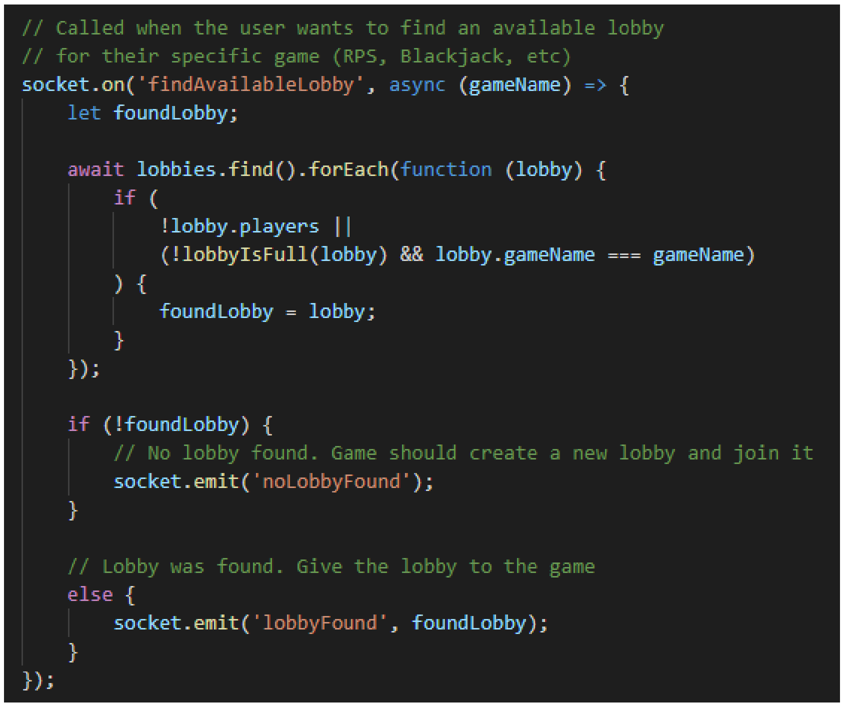
- Lobby Handler
  - To handle lobby creation, specific events were assigned to the socket on connection to any game. These events served as communication between the lobby database and the game itself, and emitted specific events for the game to receive and handle. Some key events are:



```
// Called when the user wants to create a new lobby
socket.on('createLobby', async (id, gameName, maxPlayers) => {
    // Create a lobby and insert it to the lobbies variable
    console.log('Creating new lobby for: ' + gameName);

    lobbies
        .insertOne({
            _id: id,
            gameName: gameName,
            maxPlayers: maxPlayers,
        })
        .then((doc) => {
            // Let the game define what should happen after the lobby is created
            findLobbyByID(id).then((lobby) =>
                socket.emit('createLobbySuccess', lobby)
            );
        });
});
```

Figure 22: Create Lobby



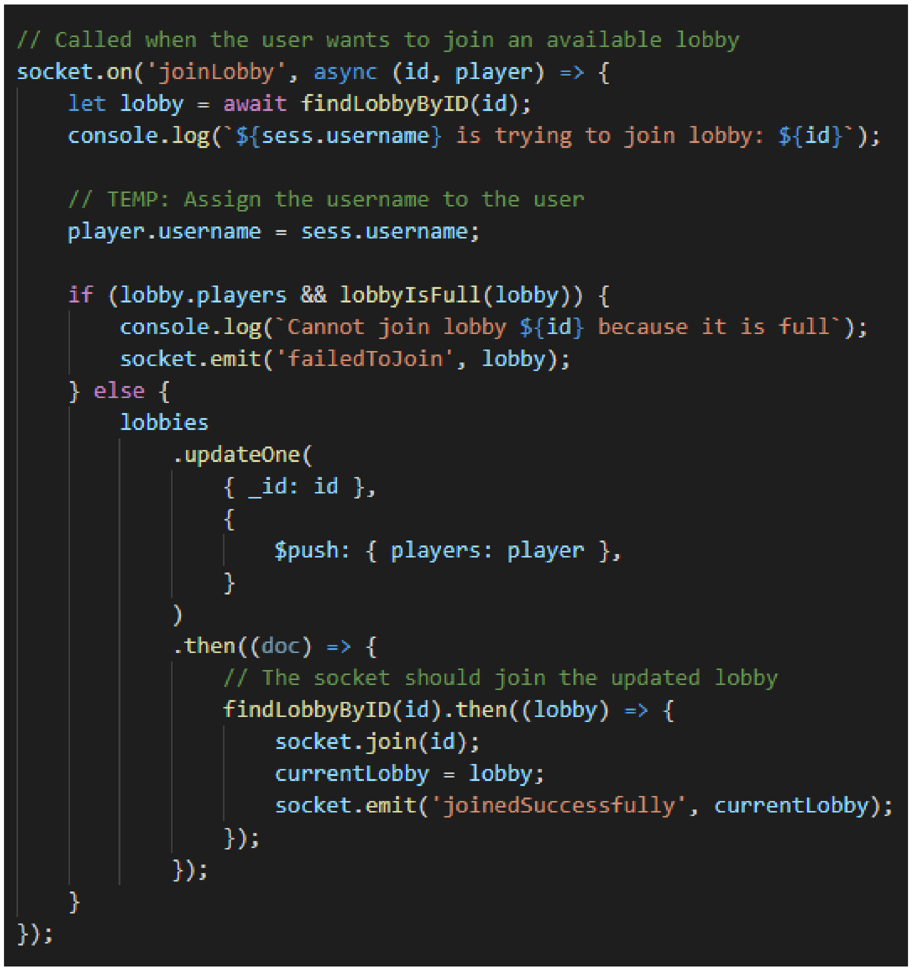
```
// Called when the user wants to find an available lobby
// for their specific game (RPS, Blackjack, etc)
socket.on('findAvailableLobby', async (gameName) => {
    let foundLobby;

    await lobbies.find().forEach(function (lobby) {
        if (
            !lobby.players ||
            (!lobbyIsFull(lobby) && lobby.gameName === gameName)
        ) {
            foundLobby = lobby;
        }
    });

    if (!foundLobby) {
        // No lobby found. Game should create a new lobby and join it
        socket.emit('noLobbyFound');
    }

    // Lobby was found. Give the lobby to the game
    else {
        socket.emit('lobbyFound', foundLobby);
    }
});
```

Figure 23: Find available Lobby



```
// Called when the user wants to join an available lobby
socket.on('joinLobby', async (id, player) => {
    let lobby = await findLobbyByID(id);
    console.log(`${sess.username} is trying to join lobby: ${id}`);

    // TEMP: Assign the username to the user
    player.username = sess.username;

    if (lobby.players && lobbyIsFull(lobby)) {
        console.log(`Cannot join lobby ${id} because it is full`);
        socket.emit('failedToJoin', lobby);
    } else {
        lobbies
            .updateOne(
                { _id: id },
                {
                    $push: { players: player },
                }
            )
            .then((doc) => {
                // The socket should join the updated lobby
                findLobbyByID(id).then((lobby) => {
                    socket.join(id);
                    currentLobby = lobby;
                    socket.emit('joinedSuccessfully', currentLobby);
                });
            });
    }
});
```

Figure 24: Join Lobby

3. **Analytic Part:**

3.1. Concept Analysis

Any players, be them gamers or non-gamers, fall into the category of player for our purposes, since skill is not a characteristic that defines these entities.
One of the main localities a player may find themselves in is a lobby. A game session is a type of virtual room where players are in to play games.

Several actions in the domain are not directly related to the game session: join, invite players, set timer, update statistics and rate game sessions. These could be grouped as administrative actions.

3.2. Validation

Validation of the system was done by all the different parties involved in the project:

- Expert Consultants
    - Provided their expert opinion and thoughts throughout the project's span.
    - The Expert consultants were divided into
        - Marko Schütz-Schmuck (Professor): Reviewed the documentation and proposals and gave his insight.
        - Janilet Rodriguez, Ricardo Velez-Davila(Consultants): Direct consultants on the project, they give feedback on the situation and answer any questions regarding the project.
        - Team members (Project developers): Designers and developers of the project, they compare the finished project relative to the original vision. Validating if it was a success or not.
- Audience
    - The main target audience that were able to give direct feedback on the games and point out any issues regarding the project.

- End Users
    - Every player utilizing the app, through a forum they can address any complaints or successes of the application, validating if the project is working.

3.3. Verification

Throughout the duration of the project, various methods such as unit testing were applied to verify if the project was performing as expected. To be more specific, the methods used were the following:

- Unit Testing: Make sure that all functions and classes work as intended **individually**, considering all edge cases that come to mind. For example: making sure that, when selecting an option in a specific game, both players keep their selection and not be able to change them until the next round.
    - This can be done by writing unit test files to test all back-end functions. Every time a new feature is added, all test files (and any test files added alongside the new feature) should completely be validated.a

- To verify that skill is not a huge factor in determining the winner in any of our games, we verify all players who have played **more** than x amount of games, where x is a number of our choosing, have similar win/loss ratios for the games they have participated in.
- For a game with a **constant** n number of players, the win-loss ratio **should** be (100/n)% for each player.
    - For example, all users who have played Rock-Paper-Scissors **should** have around a 50% win ratio in Rock-Paper-Scissors (n=2)

- For a game with an **average** n number of players, the average win-loss ratio should be 100/n% for each player.
    - For example, **assuming each # of players has an equal chance of happening,** the win-loss

        ratio for Chinese Checkers should be around 25% (average is 4 players)

    - We do not measure W/L ratio for 1-player games, as it does not contribute towards the goal of removing the skill cap from **multiplayer** games since there is no competitiveness if you play on your own.
    - The frequency of user engagement has served as another method of verification, meaning a high frequency would imply that there are users being attracted to this concept.
        - For example, if a certain user creates an account and stops playing, it implies that our concept was not successful (If there are many users who haven’t logged in for more than a specified length of time).

3.4. Technology Stack Analysis

- **Front End:**
    - **HTML, JS, CSS:**
        - The 3 essential languages of web development. Because of their versatility and easy learning curve, these 3 languages will be the main foundation for our web application.
    - **EJS:**
        - Middleware that allows easy creation of dynamic HTML pages. It was chosen because of its huge similarity to HTML, but with extra features.
    -
- **Back End:**
    - **Node.js**

        - It was already planned to use JS for front-end development, and because of Node.js' wide utilization, it features extensive documentation, libraries ,and middleware. This framework was a perfect fit for our needs.

    - **Middleware**
        - **Express/Express-Session**
            - Allows easy management of the user session, and allows us to determine if a user is logged in, or out.
        - **Bcrypt**
            - A library that provides ease in the process of hashing and salting passwords.
        - **Body-Parser**
            - Allows ease of handling request bodies under the req.body property.
        - **Cookie-Parser**
            - Parses cookies that are attached to the user request, allows support for signed cookies.

    - **MongoDB**
        - MongoDB is a source-available cross-platform document oriented database. MongoDB allows us to keep upwards of 512MB of space for free which should be enough for the start of our web app in which we only plan to store user data and information from current game sessions. To store data, the mongoose library from node.js is used as it allows the implementation of schemas for storing

        data and user information in a proper document format.

  - **Socket.io**
    - Socket.io is a library that allows for event based communication in real time between the browser and the server. This being a node.js server and the client. Though many other libraries could be used, this one had a plethora of tutorials and documentation, some of which were made for games which proved useful given the scope of the project.

3.5. Risk Analysis

- Risks:
  - User forgets password
    - Consequence: User loses all access to their account, as they cannot login anymore.
    - Solution: provide a way for users to enter their registration email address, so they may receive a temporary key to reset their old password to a new one.
  - Password is stored incorrectly on the MongoDB database
    - Consequence: Passwords can be stolen, and accounts can be hacked in the case of a data breach.
    - Solution: Hashed passwords are saved on the database instead using the bcrypt library from node.js. During authentication, we can compare the input hashed password, to the database hashed password.
  - Session is stored inside of a global variable

        - Consequence: User has to login every time they leave the website
        - Solution: Use a temporary (lasts 24 hours) cookie to identify them and keep them signed in.
    - User utilizes a weak password (e.g. 123)
        - Consequence: The password can be easily guessed, hence the user will eventually get hacked and lose access to their account.
        - Solution: Force the user to create a strong password when registering an account which contains at least 8 characters, 1 capital letter, and a number.

Progress Report Phase 1 (October 2021):

Project Advancements:

Frameworks:

- **Front-End:**
    - HTML,CSS,JS:
        - See *Section 3.4 for more details*

- **Back-End:**
    - Due to the complexity of setting them up, the team currently has not decided what back-end framework would be appropriate for web development. Tested frameworks are listed below:
        - Node.js
        - MySQL
        - Apache
        - Django
        - Flask

Roles:

- Each member of the team was split into either the Front-End, or the Back-End division. Whichever division a member is located under means that it will be their **main** responsibility during the project development. This **does not** mean that it will be their **only** work. Any member can help members of the other divisions, so that no member is left without work, and they are given a chance to develop skills in the other area.

    Front End Members:
    - Samir Y Ali Rivera
    - Maria H. Cotto Nieves
    - Ezequiel O. Rosario Sepulveda

    Back End Members:

- Jann C. Garcia Pagan
- Kenneth R. Aponte Mendez
- Eithan M. Capella Muniz

Front End:

- **Framework Changes:**
    - The team considered using Webflow as our Front-End tool.
        - It is one of the most used no-code tools to develop static websites.
        - However, we chose not to move forward with it for the following reasons:
            - The free plan only allowed a maximum of 2 pages.
            - Only the website owner can edit the page (collaboration was $35/person).

    - The team considered using Silex as our Front-End tool.
        - The web tool is open-source and free.
        - It is 100% online, and provides a drag-and-drop structure, with the addition of being able to code in your own features.
        - With enough manual configuration (via Github), the tool allows multiple collaborators.
        - However, said manual configuration was tedious, and would only get worse with the growth of our project and increasing file count.
        - Another limitation was its lack of documentation (for example, how to collaborate, how to connect back-end).

- The team also considered using Dreamweaver as a Front-End framework tool.
  - It is a powerful, well established, website design program with copious amounts of documentation.
  - The challenges that made the team decide against using Dreamweaver were:
    - Expensive to use, requires a monthly subscription.
    - Steep learning curve.
    - Cumbersome implementation of CSS styling with proprietary constraints.

- The team decided to use Visual Studio Code (VSCode) as our front end tool.
  - VSCode comes with many extensions that simplify the development process.
    - Essential extensions used in VSCode:
      - Live server:
        - Gives a developer the ability to run and make changes to the website in real time on a local host.
      - Prettier:
        - Makes HTML & CSS files easier to read.
  - All of Front-End and Back-End code is in one place
  - Access to the full source code allows for easier interpretation with full flexibility and customizability as opposed to drag and drop tools which provide limited code access.
  - Files and code can be shared throughout the team using optimal commands and functions

from git/github for a better teamwork experience.

- **Member Contributions:**
    - **Ezequiel O. Rosario Sepulveda**
        - Made the Webflow test website
        - Transferred the page to Silex
        - Formatted several pages of the website
        - Created the login and game library form template
    - **Maria H. Cotto Nieves**
        - Made a Dreamweaver test webpage.
        - Formatted several pages of the website
        - Used the login form template to create a registration page
    - **Kenneth R. Aponte Mendez**
        - Made an example website to test VSCode, which ended up being the final option.
        - Made a template that provided aesthetic uniformity to the whole site.
    - **Jann C. Garcia Pagan**
        - Looked over the HTML files to make sure no syntax errors were present, and that everything connected to the right pages.
        - Documented most changes made to frameworks and the implemented components
    - **Samir Y. Ali-Rivera**
        - Helped research different front end tools.
        - Looked over the HTML files to make sure there were no syntax or grammatical errors.

Back End:

- **Framework Changes:**
    - Node.js

        - MySQL
        - Apache
        - Django
        - Flask
- **Member Contributions:**
    - **Eithan M. Capella Muñiz:**
        - Worked with node.js and the Express framework, useful for developing web applications, body parsing requests, etc.
    - **Kenneth R. Aponte Mendez:**
        - Tested various back end alternatives, some of which are stated above.
    - **Jann C. Garcia Pagan:**
        - Tested different back end options, some of these are mentioned above.

Deployment/Hosting:

- **Frameworks:**
    - The team considered using Github Pages to host the application
- **Member Contributions:**
    - **María H. Cotto Nieves**
        - Tested Github pages with an example website

Implemented Components:

- Added Initial webpages:
    - Main Page
    - Login Page
    - Registration Page
    - Play Page

Progress Report Phase 2 (November 2021):

Project Advancements:

Front End:

- **Frameworks:**
    - To make pages more dynamic and less static, the team has commenced usage of EJS files, which are essentially HTML files, but built and rendered at runtime. EJS files not only allows us to extract repeated code, but also allows us to create dynamic HTML pages, a necessity for implementing custom games.
- **Member Contribution:**
    - **Jann C. Garcia Pagan**
        - Formatted all HTML files to improve readability
        - Extracted the Navigation Bar and Headers from most HTML files, to allow more dynamic pages.
    - **Kenneth R Aponte Mendez**
        - Updated login/registration pages to look more consistent with the design of the website.
        - Implemented LoginSuccessful / RegistrationSuccessful pages which redirects a user after 3 seconds to the play or login page after a successful login or register respectively.
    - **Ezequiel O. Rosario Sepúlveda**
        - Added sample games to the Play page, which are all available to play on singleplayer.
    - **Samir Y. Ali Rivera**
        - Helped add sample games to the Play page as well as externally linking them.
    - **María H. Cotto Nieves**

        - Uploaded a profile page where user statistics will be displayed
        - Created pages that will allow the user to reset their password or receive a temporary keyword that will help them log in to the site in case they forget their old password.
        - Included an image collection from which the user may select their avatar to be displayed throughout the site.

Back End:

- **Frameworks:**
    - The team decided to utilize Node.js as our back-end development framework
    - MongoDB was chosen to be our database to store user account information.
- **Member Contribution:**
    - **Eithan M. Capella Muñiz:**
        - Created both main JS files to handle logging in and registration
        - Cookie management to keep a user signed in, even if they close out of their browser.
    - **Jann C. Garcia Pagan:**
        - Addressed bugs present in the registration process which was preventing registration and logging in to succeed locally
        - Improved registration authentication. Users now must present a unique email address and username. Their passwords must also match before registering.
        - Restructured app.js to be more organized, and allow easy development of dynamic HTML pages

        - **Kenneth R Aponte Mendez:**
            - Connected the Node.js backend to the MongoDB database.
            - Changed the login / register methods so that they would now work with MongoDB instead of locally.

Implemented Components:

New:

- Cookies
    - Cookie management was added to keep users logged in even if the page was closed.
- Dynamic HTML Pages
    - Most pages contain a dynamic navigation bar.
- Profile Page
    - Registered Users now contain a Profile within the webpage.
- Rock Paper Scissors
    - Added Game with current support for 1 user against a CPU.

Updated:

- Login/Registration page
    - Users can now login.
    - Updated visuals of the page
    - Player is directed to the play page after logging in.

- Play Page
    - Linked external web pages that allow the user to play each respective game.

Changelog:

- **1.7: Scope**
    - It has been decided that while new games are being implemented, we will temporarily link the users to external websites that contain the game they wish to play. Although we will not be able to determine Win/Loss ratios until these games are implemented, this allows users to get a feel for the new games before they are out.

- **1.8: Span**
    - Due to external issues, phase 2’s due date has been changed from October 29th, 2021, to November 3rd, 2021.
    - Because of the team’s final decision on each framework, a frameworks section was added to finalize the decision.

- **2.4 Terminology**
    - Generalized game entities and removed round counters as not all games have a need for rounds.

- **2.5: Domain requirements**
    - Added new domain requirements which refer to the current situations and needs so that our project idea is properly showcased.
    - Modified several domain requirements to reduce redundancy.

- **2.6: Interface requirements**

    - Removed mentions of skill-based matchmaking since it would defeat the domain's purpose.

- **3.2: Validation**
    - During this phase, the development team has invited family and friends, as potential target audience members, to use the implemented system and communicate whether they feel it accomplishes the goal of being an accessible, non skilled game collection for casual gamers.

- **3.4: Technology Stack Analysis**
    - Several changes of tools and resources have been added to the project that will be vital for the creation of the website, such as Javascript and its Node.js library, multiple parsers for our middleware regarding the back-end functionality, and the addition of an online database to manage user information and security.

- **3.5: Risk Analysis**
    - Analysis of certain risks brought to our web tool is a necessity to avoid having undesired outcomes or consequences

Progress Report Final Phase (December 2021):

Project Advancements:

Frameworks:

- Because of the necessity of enabling matchmaking, a new middleware was included to the project: Socket.io, which is an event-driven middleware that allows users to communicate via WebSockets. This allowed the creation of a Lobby Manager

Member Contributions:

- **Jann C. Garcia Pagan**
  - Extrapolated RPS's lobby logic to allow it to be used as a Lobby Manager.
  - Implemented a User Feedback system.
  - Implemented Tic Tac Toe Lobby System

- **Maria H Cotto Nieves**
  - Implemented tic tac toe game logic
  - Helped test different games.

- **Samir Y Ali Rivera**
  - Helped test the different games.
  - Helped test lobby/matchmaking system.

- **Eithan M Capella Muniz**
  - Implemented Connect 4 using jquery, and helped with some slight testing.

- **Ezequiel Obed Rosario Sepúlveda**
  - Implemented Blackjack.
  - Added CPU logic for Blackjack.
  - Test and support on other games.

- **Kenneth R Aponte Mendez**
  - Implemented Rock-Paper-Scissors.

        - Implemented the lobby system for Rock-Paper-Scissors that works for 2 players which was then refactored and extrapolated so that it could be used with other games.

Implemented Components
New:

- Feedback Page
    - Users are now able to provide feedback that is submitted to our database.

- Tic Tac Toe
    - Added Game with current support for 2 players.

- Blackjack
    - Added Game with current support for 1 user against a CPU.

- Connect 4
    - Added Game with current support for 2 local players.

Updated:

- Play Page
    - Feedback is now accessible from the play page.
    - Rock Paper Scissors, Blackjack, and Tic Tac Toe no longer link to external websites. Instead they link to our custom versions of the game.

- Rock Paper Scissors
    - Supports 2-Player mode.