Software Requirements Specification

for

Squared

Version 1.5 approved

Prepared by Team Squared

COS420 - Software Engineering

February 20th, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Keijaoh Campbell	02/16/2022	Added the purpose, intended audience and project scope for the SRS.	1.0
Tyler Walker	2/19/2022	Added majority of section 2, as well as contributed software terms to the glossary.	1.1
Shea Keegan	2/19/2022	Added Document conventions (1.2) and finished the first 2 features and their functional requirements	1.2
Shea Keegan	2/20/2022	Added a third feature and its functional requirements.	1.3
Jeffrey Fosgate	2/20/2022	Added non-functional requirements to sections 5.1 - 5.5.	1.35
Jacob Lorenzo	2/20/2022	Added user documentation section and safety requirements 5-1 and 5-2	1.4
Keijaoh Campbell	2/20/2022	Added the Mock Up UI components, Glossary, Hardware and Software interface.	1.5
Shea Keegan	3/4/2022	Fixed FRs in 4.1.3, 4.2.3, fixed NFRs in 5.5, and in 6 from deliverable 1 feedback.	
Shea Keegan	3/6/2022	Finished the rest of the needed revisions	

Jacob Lorenzo	3/22/2022	Revised and updated document, made formatting consistent	1.6

1. Introduction

Squared is a multiplayer competitive game where the objective is to take over as many squares on the 11x11 board as possible. The game will run in the browser, allowing both mobile/desktop users to play simultaneously. In Squared, the players rely on chance (in the form of dice rolls) to determine how many spaces they can move in a turn. Numbers 1-4 will be the number of squares they can move, with 5-6 causing the player to skip their turn. Players will then see an outline of the possible squares they can move to via clicking. However, players cannot move diagonally. Also, players cannot move onto board squares already "taken" by other players.

1.1 Purpose

The purpose of this document is to build a web based board game. Which will allow gamers to sign up and play against other players and achieve the game's win condition while spectators comment and view the scoreboard and the game progression.

1.2 Document Conventions

The typographical conventions followed are pretty typical. We used Times New Roman at size 12 and used bolded titles to show distinction from normal body paragraphs. Priority of requirements are not indicative of their order, but of their individual priority.

1.3 Intended Audience and Reading Suggestions

This application is intended for use by the general public from age 10 and older. However, for the scope of this project. The project will be using a convenience sample of the audience at the UMaine campus. These persons include the students and professors. The academic level will not play a factor in selecting the audience member.

This document is intended to be used by the developers of Squared to use the document as a guide for the implementation of the game and its associated components. Additionally, the suggested reading section for the developers are Product Scope, and Product Functions. Product Scope, will aid the developers to measure the deliverables that can be realistically delivered in the project's timeline based on the scope outlined.

1.4 Product Scope

The project will consist of a game board which will be the main interaction point of the visitors and gamers. There will be chat functionality for gamers to interact with each other. As well as spectators interacting with each other. The project will consist of a scoreboard for individuals to view their game history, as well as a top score board to see the records that have been made. As well as the top players per player type and overall. Lastly, the application will have a store where players can buy outfits for their characters and special abilities as well.

2. Overall Description

2.1 Product Perspective

The project is a standalone web-based board game designed to be played by players of all ages. It is not a follow-on member of a product family. However, it is a re-envisionment of a game concept one of the team members have previously explored and had not brought beyond the idea stages. This original concept was intended for the Windows 10 phone, which no longer exists.

2.2 Product Functions

- Allow two (or more) users to play a game together
- Allow users to chat with each other during the game
- Allow automated matchmaking for ease-of-use
- Provide a help page with game instructions for new players

2.3 User Classes and Characteristics

Children: People with little technical knowledge. Expectations involve providing a simple enough interface along with a simple explanation of the rules.

Competitive players: People with inherent skill and knowledge of the game. They likely are either in or completed highschool. They seek a challenging experience with other players. It may be beneficial to them to add a matchmaking system to pair them up with other players of a similar skillset.

Casual players: People who merely want a good time. For them, providing instant access to the game is all they'd need to be satisfied.

2.4 Operating Environment

The game will run on just about any platform, considering it runs in a web browser. This includes any operating system, device, browser, etc. Provided they can run a web page, they can play the game.

2.5 Design and Implementation Constraints

Since the game is web-based, it will require a server to play. Meaning that it'd be reliant on us running it in order to provide the service. We also plan on incorporating a database, but provided it's MongoDB (meaning we run it on our hardware) there shouldn't be any issue in relying on another server. In terms of security, we'd have to ensure that we fuzz all aspects of the input and validate forms to prevent sql injection.

2.6 User Documentation

The product will have a help page that will provide the user with a video and a written explanation on how to use the product.

2.7 Assumptions and Dependencies

We plan on using a simple VPS to host the game. As a result, we're reliant on them for security patches and uptime. This also applies to Python and MongoDB, provided they will be the backend of our project. We also plan on using Github to host the software repository and have the VPS mirror that. For software libraries, we plan to use Flask, socket.io, and a middleware to access the MongoDb server. Meaning we'll be relying on their maintainers as well to ensure security for the application.

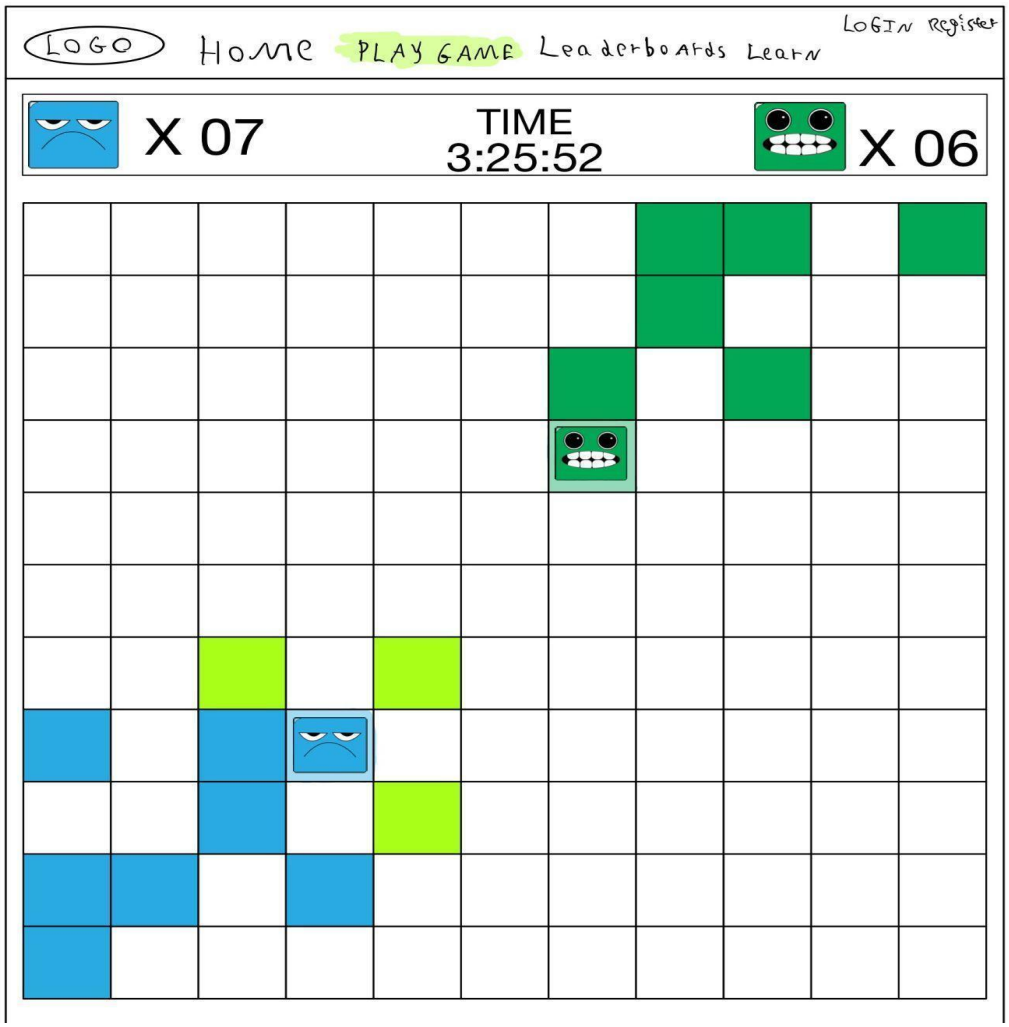
3. External Interface Requirements

3.1 User Interfaces

The following are mock ups of the various screens involved in the web based game. These screens include a Login, Register, Play Game, Leaderboard, Chat and Learn pages.

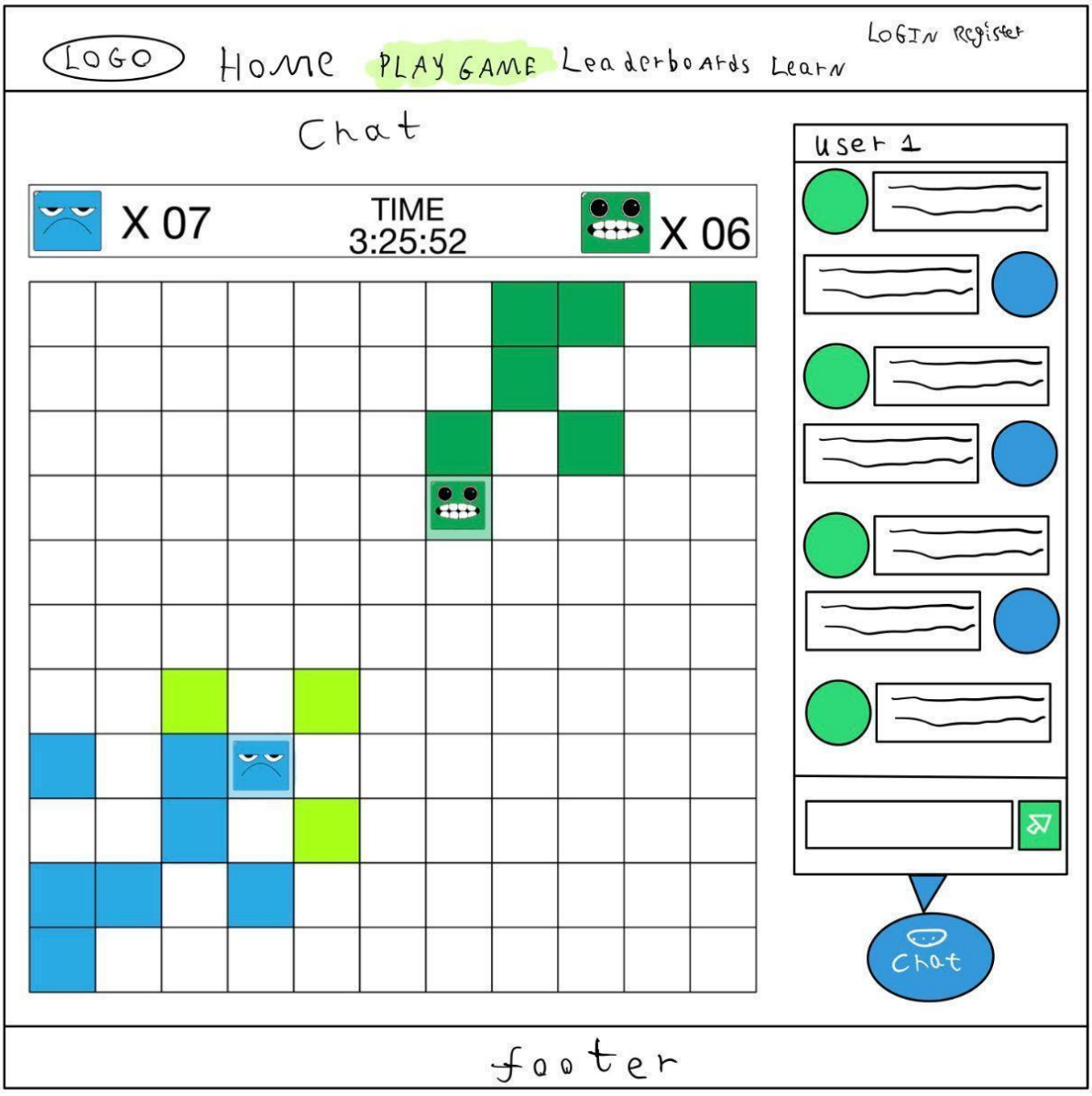
3.1.1 Play Game

This is the main interaction point for the website's visitors. It is where the users with play the game, spectate and use the chat functionality to interact with the other players and spectators of the game.


LOGO
Home
PLAY GAME
Leaderboards
Learn
LOGIN
Register
X 07
TIME
3:25:52
X 06

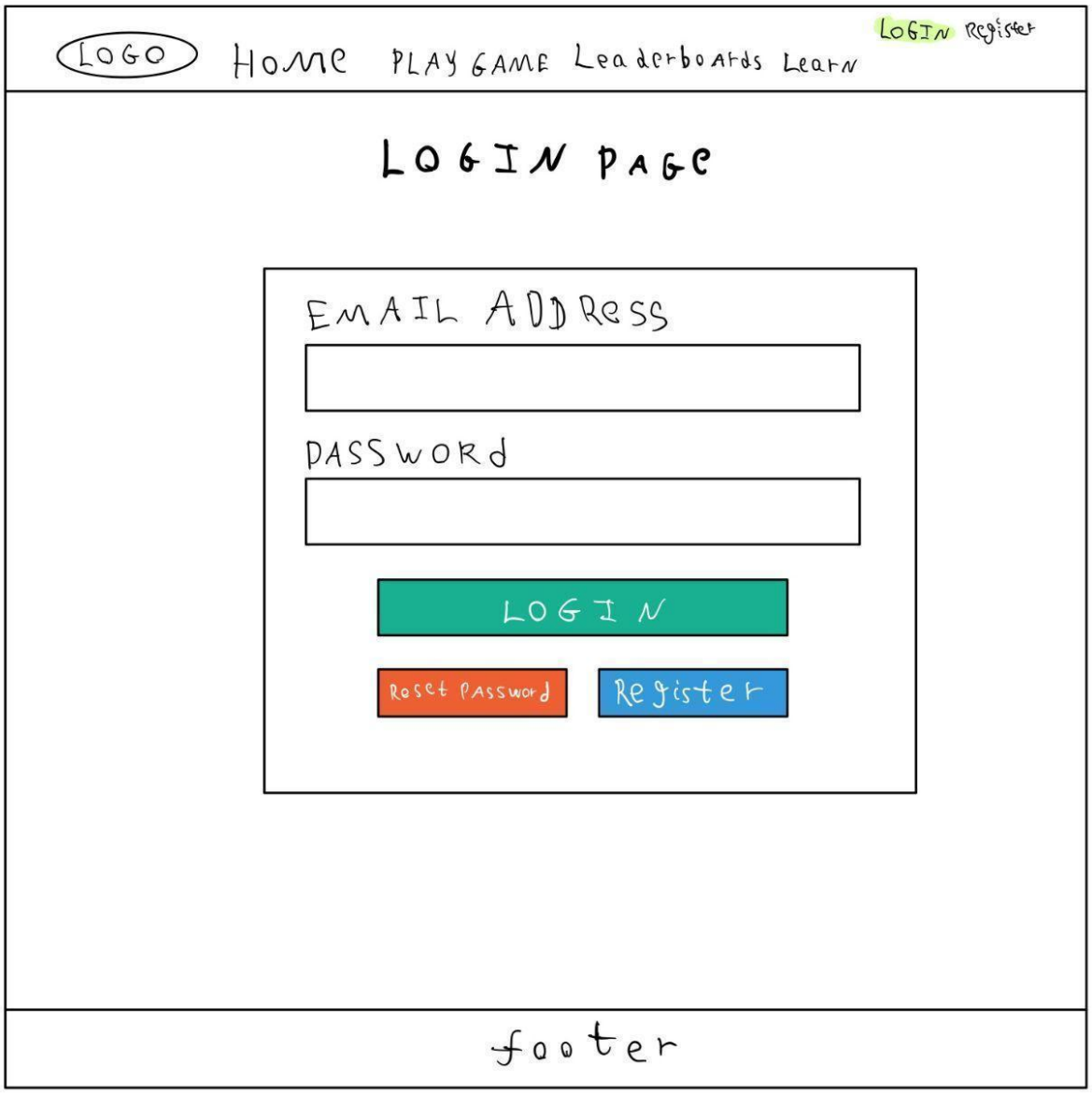
3.1.2 Play Game (Chat Functionality) Page

This mock up showcases the chat functionality of the game and the chat history of "User 1".



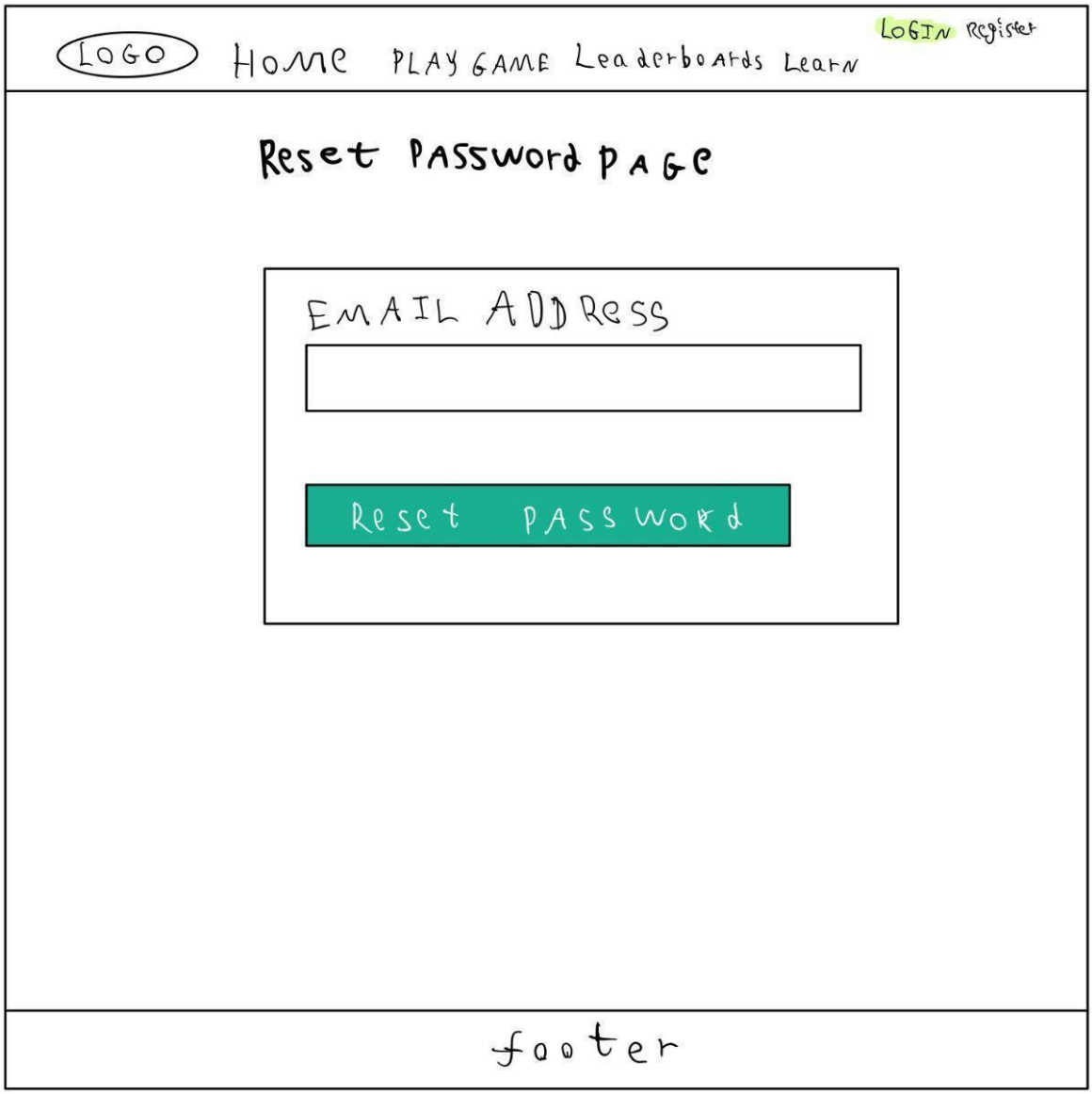
3.1.3 Login Page

The login page which is required for users to join the chat or play the game or to make purchases in the game's store.



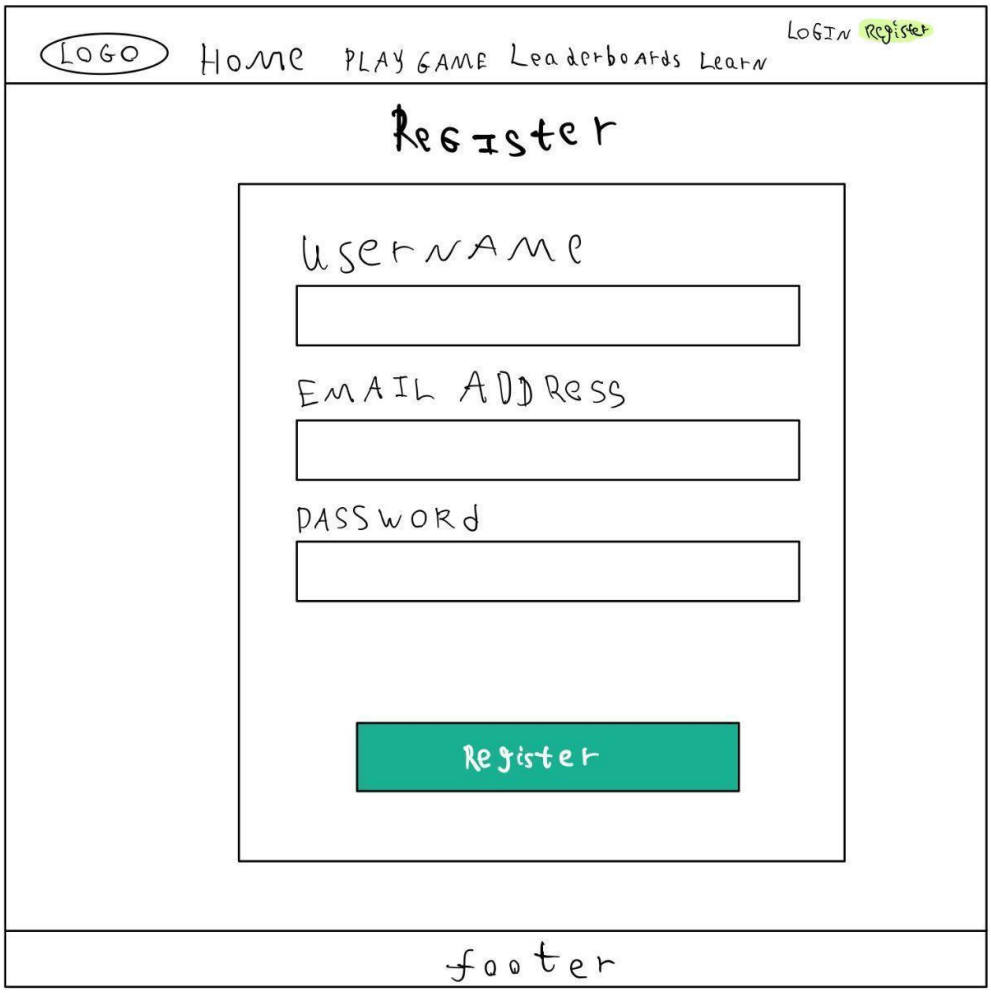
3.1.3 Reset Password Page

This page is intended for use when a user who already has an account and has forgotten their password. A reset password link will be sent to their email address which will have an expiration date and time.



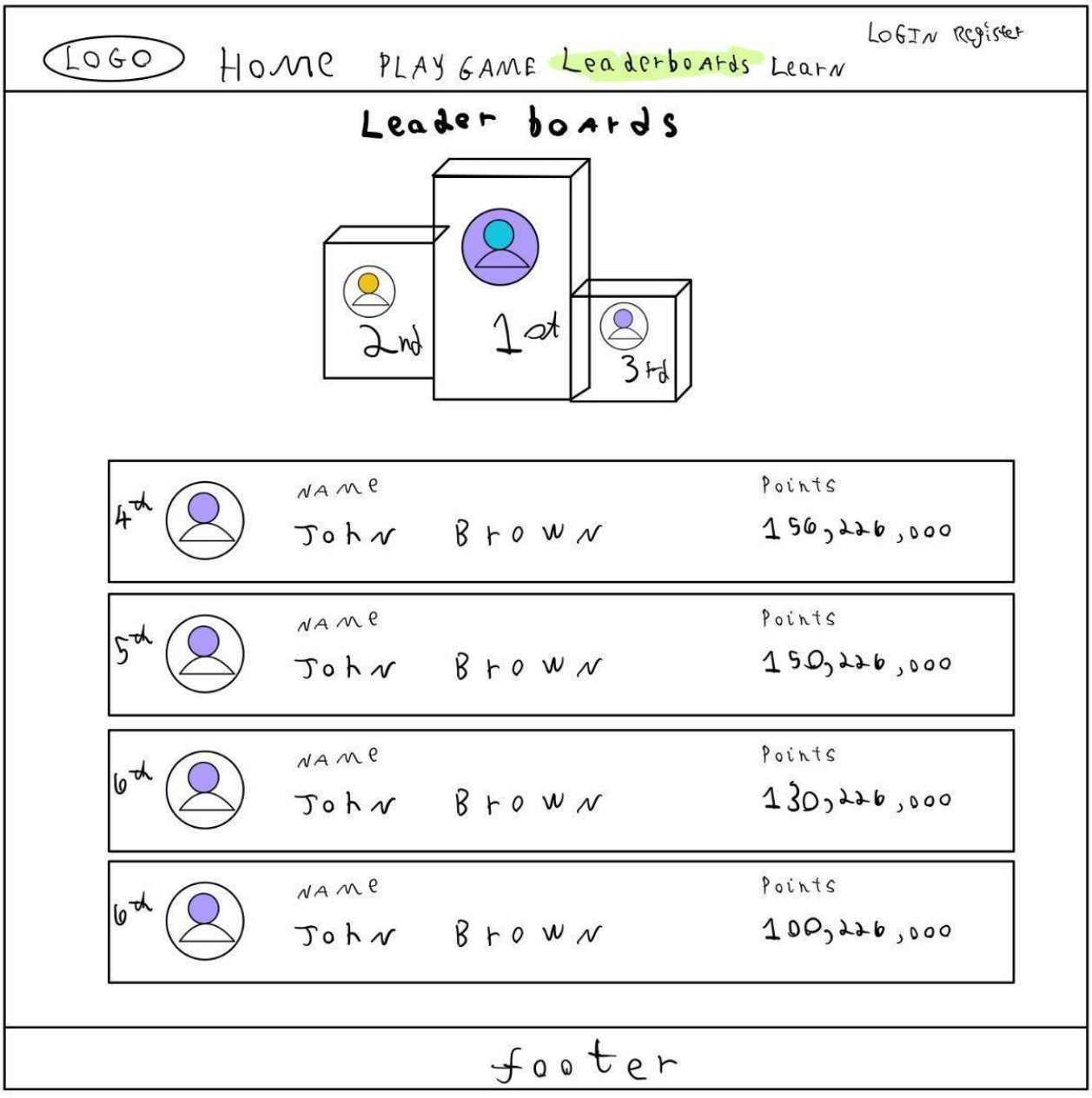
3.1.4 Registration Page

This page allows anonymous users to register to have full access to the game and its functionalities like chat, playing the game and making purchases in the store.



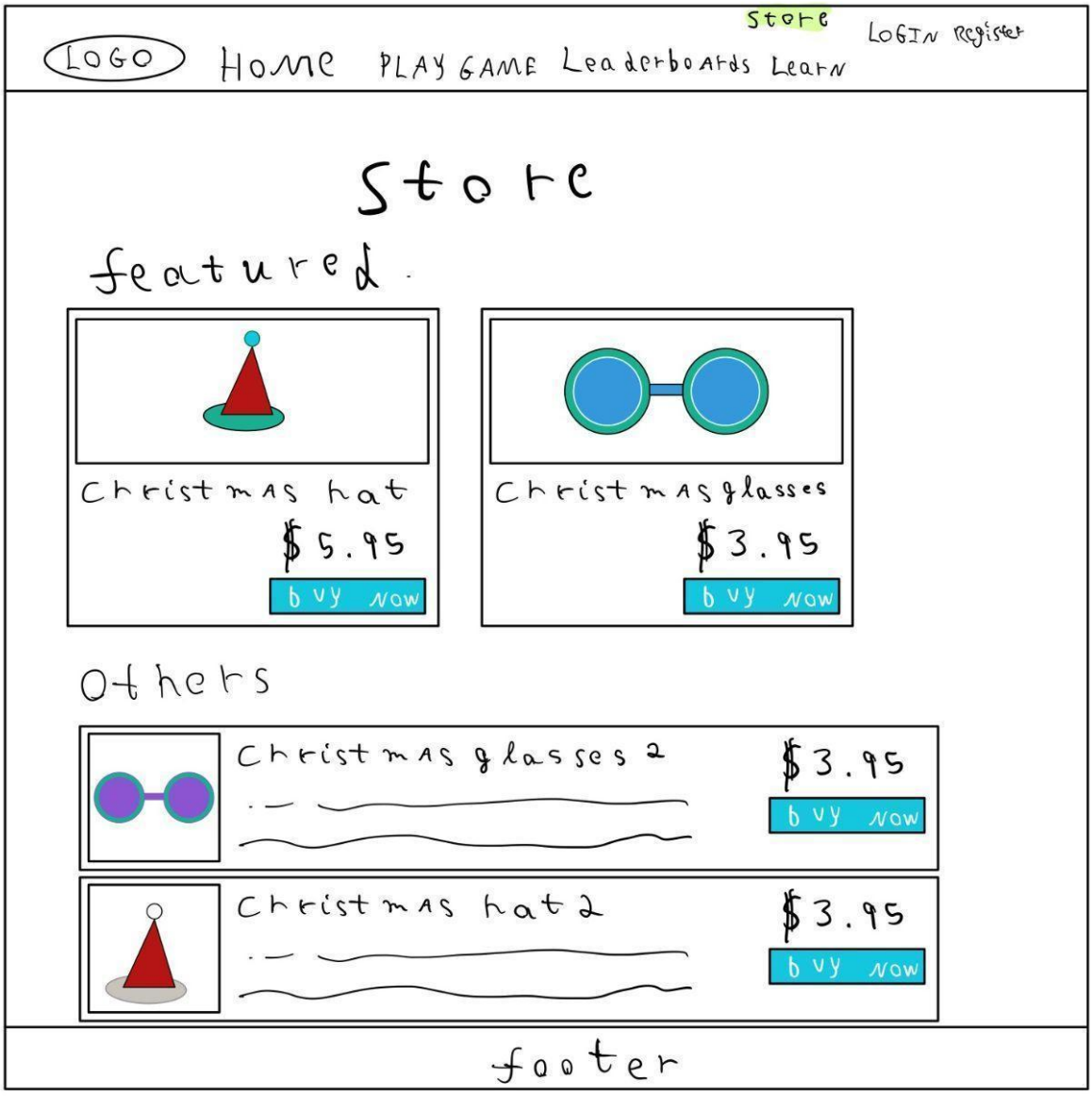
3.1.5 Leaderboard Page

This page will display the top players and records for various game sessions, and achievements.



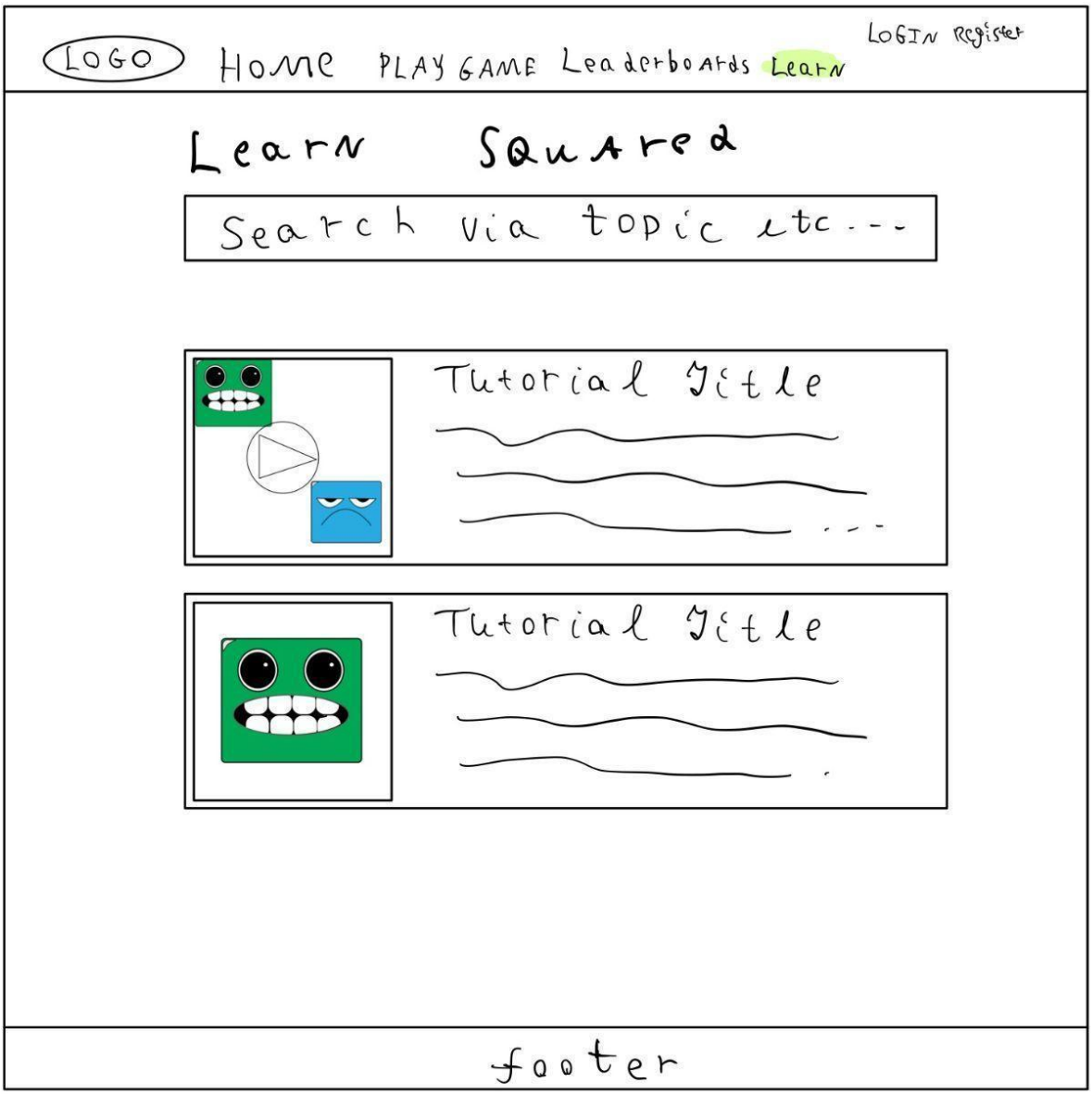
3.1.6 Store Page

This view will allow players to purchase power ups and character skins or special character costumes to differentiate themselves in their game play and gamer profile.



3.1.7 Learn to play Squared Page

This page is for new players or players who would like a refresher on how to play the game and understand the win conditions and reward systems.



3.2 Hardware Interfaces

This game will be hosted on the web, therefore the hardware requirements are either a mobile device, laptop or desktop. Additionally, a web browser that is the latest or close the the most current stable version of the web browser because the game will always be updated to adhere and take advantage of the latest versions of the various web browsers such as Google Chrome. Safari, Microsoft Edge and Firefox.

3.3 Software Interfaces

Python - this is the base programming language that will be used to handle the API endpoints and logic required to interact with the database and methods related to the game logic like win conditions.

VPS - will be used to host the production and mirror environment of the game. This will be where the database, API and website interface will be hosted behind a firewall.

Flask - this is the Python library which will be used to handle the API endpoint's Post, Get, Put and Delete requests. It also handles the routing of requests and authentication of requests by using the extension library JWT. The data will be retrieved from the interface in a JSON format and be de-serialized and processed and the Flask endpoints will be used to serialize the results and return the response in JSON format to the end user or interface.

Socket.io - This Python library will be used to handle functionalities that require the use of sockets like chats and other real time functions. This library will be used within the API for the relevant endpoints that require sockets.

MongoDB - This document based database will be interacted with from the API and will store records like high scores, profile data among others. The data that is retrieved will be in a JSON format from the API.

Github - This repository will be used to back up all the project's source code and assets and facilitate group collaboration on the development of the project.

Stripe - This third party payment platform will be used on the API endpoints that handle transaction verification that the payment was successful among other monetary transactions queries. The data will be sent from the game's API to Stripes API in the form of a POST, GET, PUT or Delete along with an authentication token to verify the developer account, as well as other data which would be dependent on the type of request being made.

4. System Features

Chat Feature
Dice Roll
The Board

4.1 Chat Feature

4.1.1 Description and Priority

This feature will allow the players to communicate with each other during and after the match. It allows easy rematch communication and assures the user that they are not playing against a bot. This is a medium priority feature at the moment because it is not integral to the game, but it is a feature that we all talked about implementing from the very beginning.

4.1.2 Stimulus/Response Sequences

User A will interact with the chat bar at the bottom of the chat box and type a message of any kind in it. When User A feels his message is done, he can hit enter and the chat is sent to a server that all chatters are connected to and then the chat goes from the server to User B and User C.

4.1.3 Functional Requirements

REQ-1-1: The chat system shall show the time the message was sent. Priority medium
REQ-1-2: The chat system shall show the name of the person who sent it. Priority medium

REQ-1-3: The chat system shall have a reminder for when it's your turn. Priority medium

REQ-1-4: The chat system shall allow spectators to chat as well. Priority medium

REQ-1-5: The system shall allow the chat to be minimized. Priority low

REQ-1-6: The system shall make a notification noise for new chats when chat is minimized. Priority low

4.2 Dice Roll

4.2.1 Description and Priority

The dice roll is a simple feature that helps randomly determine a number 1-6 per die. This has high priority because it is integral to our game. Without the dice roll, our game would need to be reworked.

4.2.2 Stimulus/Response Sequences

The user will press the button to roll dice and that will start the program to generate a dice roll. The dice roll will be stored and sent to all players and spectators to see. The user who rolled the dice will then have that number of moves.

4.2.3 Functional Requirements

REQ-2-1: The system shall display a picture of the dice roll that was generated when the user rolls the dice. Priority high

REQ-2-2: The system shall show your dice roll to the other players/spectators. Priority high

REQ-2-3: The system shall keep track of last turns dice roll. Priority medium

REQ-2-4: The system shall activate special conditions for specific dice rolls. Priority low

REQ-2-5: The system shall allow for dice re-colors/re-skins. Priority low

REQ-2-6: The dice shall show an animation before the numbers are revealed. Priority low

REQ-2-7: The system shall play a rolling sound effect as the user rolls the dice. Priority low

4.3 The Board

4.3.1 Description and Priority

The board is the field that our game will be played on. It is 11x11 and starts teams off in opposing corners. This will be the visual representation of where all of the game pieces are and what cells have been occupied already. This is a high priority because it is where the game is played. Squared without the board is like playing football with no yard markers.

4.3.2 Stimulus/Response Sequences

After a dice roll, the user will be prompted with paths of possible moves. The user needs to click or tap the spot that they wish to move to. Once the user has used all of their moves, their turn will end and it will be the next player's turn.

4.3.3 Functional Requirements

REQ-3-1: The system shall highlight possible moves based on the dice roll. Priority high
REQ-3-2: The system shall update the state of the board each turn. Priority high
REQ-3-3: The system shall allow for board customization. Priority low
REQ-3-4: The system shall make sound effects as the character moves spaces. Priority low

5. Other Nonfunctional Requirements

5.1 Performance Requirements

REQ-4-1: The system’s response time to player input in-game must not exceed three seconds per input, 99% of the time.

REQ-4-2: The system shall process account login requests in no more than five seconds 99% of the time.

REQ-4-3: The system shall create a data server entry for new accounts within five seconds of submitting account data, 95% of the time.

REQ-4-4: The system shall connect players who are waiting to play together and initiate a new game in no more than thirty seconds, 90% of the time.

REQ-4-5: The system shall display chat messages sent to other players within five seconds of the message being submitted, 99% of the time.

5.2 Safety Requirements

REQ-5-1: The system shall give basic users minimal privilege so the chance of operations crashing the application is minimized.

REQ-5-2: The system shall use HTTPS to ensure encryption on the service level.

5.3 Security Requirements

REQ-6-1: The system shall ensure that all account data sent to and from the data server is secure.

REQ-6-2: The system shall have password expiration so that inactive accounts can't be targeted.

REQ-6-3: The system shall require Two-factor authentication in order to sign up so that all accounts are as secure as possible.

REQ-6-4: The system shall have exception management to limit the output to an error code when a failure occurs.

5.4 Software Quality Attributes

REQ-7-1: The system shall have a dynamic UI that will scale to the dimensions of the user's screen.

REQ-7-2: The system shall not have more than a 10-second delay from when one user ends a turn to when the other user receives the information 99% of the time.

REQ-7-3: The system shall provide users with limited previews of themes, so the user can see it before equipping it.

REQ-7-4: The system shall provide users with limited previews of items, so the user can see it before purchasing it.

5.5 Business Rules

REQ-8-1: The website's developers shall be responsible for updating the appearance of *Squared* in accordance with player demands and feedback.

REQ-8-2: The website's developers shall be responsible for managing the functionality of *Squared* in accordance with player demands and feedback.

REQ-8-3: The website's administrators shall be members of the website's development team responsible for managing game moderators and reporting communicating player demands and feedback to other website developers

REQ-8-4: The game moderators shall be members of the *Squared* player base that will be responsible for moderating chat messages and games, and punishing players who attempt to breach the website's safety and security requirements.

REQ-8-5: The game referees shall be members of the *Squared* player base that are able to moderate individual games and serve as an impartial witness to *Squared* games in the event of disputes between players in-game.

REQ-8-6: The website's developers shall be responsible for managing and assessing the website's administrators.

6. Other Requirements

The website's developers shall be responsible for complying with local laws regarding data collection.
(ie: Europe's GDPR, which restricts the collection of user data without explicit consent)

Appendix A: Glossary

API - Application Programming Interface is responsible for routing requests from an application's interface and interacts with the database directly. Separating concerns, so that the application interface doesn't know the database exists and instead interacts with methods exposed to the internet , also known as end points.

VPS - Virtual Private Server. Essentially, a remote server developers can access in order to run the project.

GDPR - General Data Protection Regulation. A European law regarding user data, in in particular, it's collection and removal on user request.

Python - A high level programming language designed for ease of use.

Flask - A Python library to quickly build webpages.

HTML - Hyper-Text Markup Language. A way to create websites

CSS - Cascading Style Sheets. A way to style HTML

Javascript - A high level programming language to run in the browser.

Front-end - The side of the application a user sees. Typically includes HTML/CSS/Javascript.

Back-end - The side of the application that supports the front-end. This includes things like signing users in, running code, and communicating with a database.

MongoDB - Is a cross platform document-oriented databaseIt is also known as a NoSQL database due to the fact that it uses a JSON like document structure with the addition of

optional schemas. In the scope of this project, the database will be used to store player profile information, game statistics and store purchases.

Socket server - A way for the front-end and the back-end of a web application to communicate in a session.

Mirror Environment - is an area where new features are tested in a simulated environment which uses snapshot production data usually a week old. To do all the necessary tests before the source code is published to production.

Stripe - this is a third party payment gateway that will handle the monetary transactions of the system and the securing the customers data during and after the transaction has completed.

JWT - JSON Web Token is an open industry standard (RFC 7519) method used to secure connections between two parties by using authentication tokens to sure API endpoints.

Post Request - This verb is used to describe create methods or endpoints used to create new records in a database.

Get Request - This verb is used to describe retrieval methods or endpoints used to fetch records from a database. This type of endpoint can retrieve a single record, multiple records or multiple records filtered by various attributes.

Put Request - This verb is used to describe update methods, which are used to update an existing record or records.

Delete Request - This verb is used to delete records permanently and as such should be handled with care.

Serialization - It is the process of transforming a data structure into a format that can be

transmitted across the network. In other words from bytes back to an object that can be consumed.

De-serialization - This is the opposite of Serialization, because it takes the transformed data structure and returns it into its original format so that it will be more readable.