PROJECT REPORT

ON

JAVASCRIPT GAME

FOR THE COURSE OF T. Y. B.Sc.

COMPUTER SCIENCE

DESIGNED AND DEVELOPED

BY

ANURAG MACHHINDRA KHOKALE

GUIDED BY

Ms. BHUMIKA NAKUM

DEPARTMENT OF COMPUTER SCIENCE

PARLE TILAK VIDYALAYA ASSOCIATION’S

MULUND COLLEGE OF COMMERCE (AUTONOMOUS)

*(Affiliated to University of Mumbai)*

2024-25

MULUND (WEST), MUMBAI-400080

MAHARASHTRA, INDIA

**ACKNOWLEDGEMENT**

I like to extend my gratitude to Dr. Sonali Pednekar, our Principal and all staff of Mulund College of Commerce for providing us moral support, conducivework environment and the much needed inspiration to complete this project on time.

I also take this opportunity to thank our Course Coordinator Dr. Reena Nagda and all the faculties of Department of Computer Science for giving us the most needed guidance and continuous encouragement throughout the duration of the Programme.

I wish to extend my deepest gratitude and special thanks to my project guide Ms. Bhumika Nakum for giving their generous support, necessary inputs and companionship during my project work.

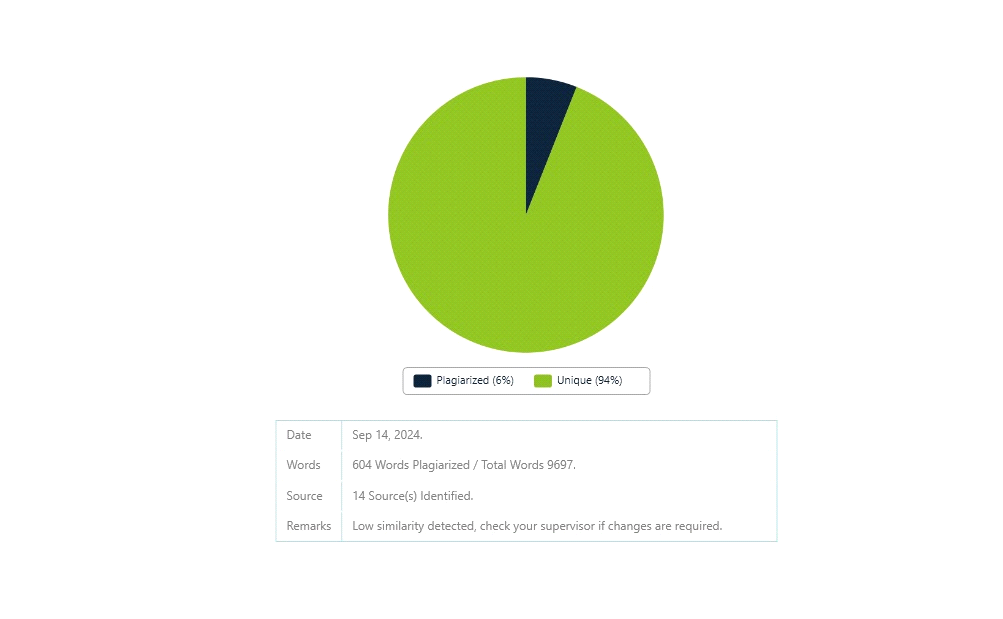
I would like to convey my special thanks to the Management and all the staff of the college for providing the required infrastructure and resource to enable the completion and enrichment of my project.

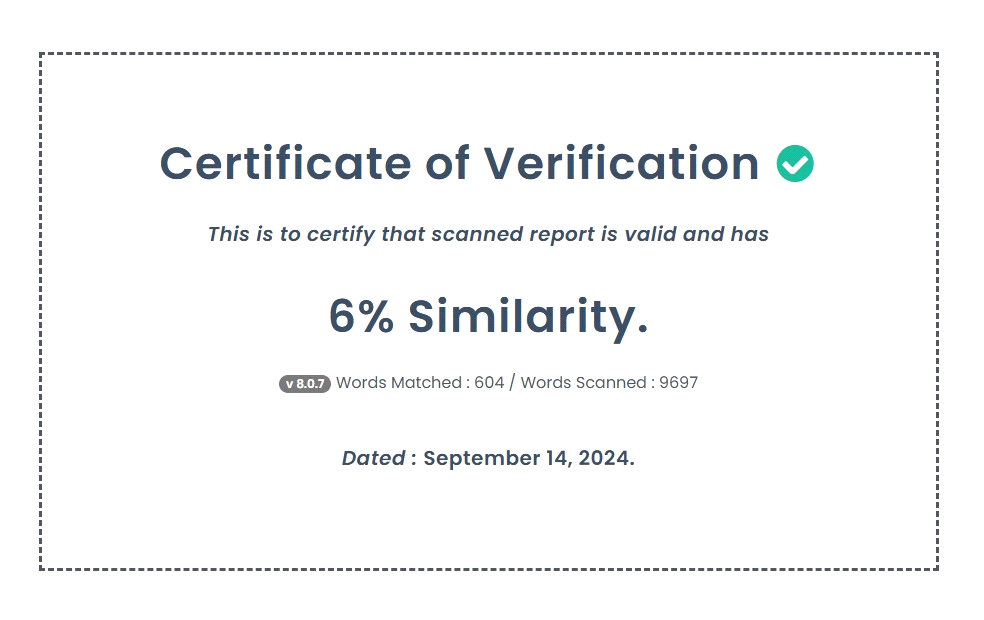
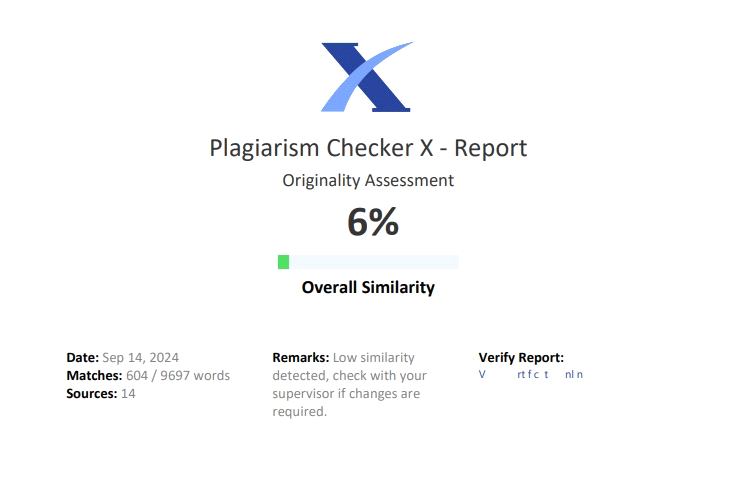
I am extremely grateful to the University of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally I thank all my fellow friends who have directly or indirectly helped me in completing my project.

**Anurag Machhindra Khokale**

**PLAGIARISM REPORT**





**INDEX**

Contents

[1.Title 1](#_Toc177772644)

[1.1 Title of The Project 1](#_Toc177772645)

[1.2 Type of The Project 1](#_Toc177772646)

[1.3 Developed By 1](#_Toc177772647)

[2. Introduction 2](#_Toc177772648)

[2.1 Objective of the project 2](#_Toc177772649)

[2.2 Description of the current system 3](#_Toc177772650)

[2.3 Limitation of the current system 3](#_Toc177772651)

[2.4 Description of the proposed system 4](#_Toc177772652)

[2.5 Advantages of the proposed system 4](#_Toc177772653)

[3.Requirement Specification 6](#_Toc177772654)

[3.1 Software Requirements 6](#_Toc177772655)

[3.2 Hardware Requirements 6](#_Toc177772656)

[3.3 Data Requirements 6](#_Toc177772657)

[3.4 Fact Finding Question 7](#_Toc177772658)

[4.System Design 8](#_Toc177772659)

[4.1 Event Table 8](#_Toc177772660)

[4.2 Class Diagram 10](#_Toc177772661)

[4.3 Use Case Diagram 11](#_Toc177772662)

[4.3.1 Use Case Diagram for User Authentication 11](#_Toc177772663)

[4.3.3 Use Case Diagram for Account Page with Profile and Friends Management 12](#_Toc177772664)

[4.3.4 Use Case Diagram for Game Interaction 12](#_Toc177772665)

[4.4 Sequence Diagram 13](#_Toc177772666)

[4.4.1 SignIn 13](#_Toc177772667)

[4.4.2 SignUp 13](#_Toc177772668)

[4.4.3 Homepage 14](#_Toc177772669)

[4.4.4 account 15](#_Toc177772670)

[4.4.5 game 16](#_Toc177772671)

[4.5 Activity Diagram 17](#_Toc177772672)

[4.5.1 Activity Diagram for SignIn 17](#_Toc177772673)

[4.5.2 Activity Diagram for SignUp 18](#_Toc177772674)

[4.5.3 Activity Diagram forhomepage 18](#_Toc177772675)

[4.5.4 Activity Diagram account page 19](#_Toc177772676)

[4.5.5 Activity Diagram for speedy pup 20](#_Toc177772677)

[4.5.6 Activity Diagram for gorilla clash 21](#_Toc177772678)

[4.5.7 Activity Diagram for samurai 22](#_Toc177772679)

[4.6 State Diagram 23](#_Toc177772680)

[4.6.1 Home State Diagram 23](#_Toc177772681)

[4.6.2 Account State Diagram 23](#_Toc177772682)

[4.6.3 Speedy Pup game State Diagram 24](#_Toc177772683)

[4.6.4 Gorilla Game State Diagram 25](#_Toc177772684)

[4.6.5 Samurai Game State Diagram 26](#_Toc177772685)

[4.7 Package Diagram 27](#_Toc177772686)

[4.8 Component Diagram 27](#_Toc177772687)

[4.9 Deployment Diagram 28](#_Toc177772688)

[4.10 Database Design 29](#_Toc177772689)

[5.System Implementation 31](#_Toc177772690)

[5.1 account.html 31](#_Toc177772691)

[5.2 index.html ( speedy pup game ) 33](#_Toc177772692)

[5.3 index2.html ( gorilla game ) 35](#_Toc177772693)

[5.4 home.html ( samurai game ) 38](#_Toc177772694)

[5.5 home.html 40](#_Toc177772695)

[5.6 login.html 43](#_Toc177772696)

[5.7 signup.html 45](#_Toc177772697)

[5.8 about.html 50](#_Toc177772698)

[5.9 dogInfo.html 54](#_Toc177772699)

[5.10 gorillaInfo.html 55](#_Toc177772700)

[5.11 samuraiInfo.html 56](#_Toc177772701)

[5.12 account.js 57](#_Toc177772702)

[5.13 backgroung.js ( dog game ) 67](#_Toc177772703)

[5.13 enemy.js 68](#_Toc177772704)

[5.14 input.js 70](#_Toc177772705)

[5.15 main.js 71](#_Toc177772706)

[5.16 particale.js 75](#_Toc177772707)

[5.17 player.js 77](#_Toc177772708)

[5.18 playerStates.js 79](#_Toc177772709)

[5.19 index.js ( gorilla game ) 83](#_Toc177772710)

[5.20 index.js ( samurai game ) 97](#_Toc177772711)

[5.21 utils.js 108](#_Toc177772712)

[5.22 about.js , gorrilaGameInfo.js, dogGameinfo.js, samuraiGameinfo.js 109](#_Toc177772713)

[5.23 allowedOrigins.js 110](#_Toc177772714)

[5.24 corsOptions.js 110](#_Toc177772715)

[5.25 authController.js 110](#_Toc177772716)

[5.26 gameController.js 116](#_Toc177772717)

[5.27 friendController.js 117](#_Toc177772718)

[5.28 authMiddleware.js 121](#_Toc177772719)

[5.29 Game.js 122](#_Toc177772720)

[5.30 user.js (model) 123](#_Toc177772721)

[5.31 game.js (model) 124](#_Toc177772722)

[5.32 friend.js (model) 124](#_Toc177772723)

[5.33 auth.js ( route ) 125](#_Toc177772724)

[5.34 game.js ( route ) 126](#_Toc177772725)

[5.35 friendrout.js ( route ) 127](#_Toc177772726)

[5.36 server.js 127](#_Toc177772727)

[6.Results 129](#_Toc177772728)

[6.1 Validation and Naming Conventions 129](#_Toc177772729)

[6.2 Screenshots 130](#_Toc177772730)

[6.2.1 Login Page 130](#_Toc177772731)

[6.2.2 Signup Page 130](#_Toc177772732)

[6.2.3 Home Page 131](#_Toc177772733)

[6.2.4 Game1 Page 131](#_Toc177772734)

[6.2.5 Game2 Page 132](#_Toc177772735)

[6.2.6 Game3 Page 132](#_Toc177772736)

[6.2.7 Accounts Page 133](#_Toc177772737)

[6.2.8 FriendDetail Page 133](#_Toc177772738)

[6.2.6 About Page 134](#_Toc177772739)

[6.2.7 Patches Page 134](#_Toc177772740)

[7. Future Scope and Conclusion 135](#_Toc177772741)

[7.1 Future Enhancement 135](#_Toc177772742)

[7.2 Conclusion 136](#_Toc177772743)

[8.References 137](#_Toc177772744)

[9.Annexure 138](#_Toc177772745)

# 1.Title

1.1 Title of The ProjectWeb Game with Java Script

1.2 Type of The ProjectWebsite

1.3 Developed ByAnurag M Khokale

# 2. Introduction

## 2.1 Objective of the project

* **Diverse Game Collection:** The platform will boast a broad selection of games, ranging from fast-paced action titles to thought-provoking puzzles, ensuring that users of all backgrounds can find something to enjoy. By featuring both single-player and multiplayer options, players will be able to explore solo challenges or dive into collaborative and competitive gameplay experiences with friends or other users.
* **Seamless and Immersive User Experience:** The design will prioritize delivering a seamless and fluid gaming experience, focusing on smooth performance, quick load times, and high-quality graphics. Attention will be given to creating visually engaging environments that captivate users, coupled with intuitive navigation that enables effortless transitions between games. The platform will be fully responsive, ensuring users can access and enjoy games across various devices, including desktops, tablets, and mobile phones.
* **Real-Time Score Tracking and Leaderboards:** Players will be able to track their scores in real-time, both within individual games and across the entire platform. A comprehensive leaderboard system will showcase top players, fostering a sense of healthy competition and motivation. Achievements and milestones will be implemented to reward players for their progress, adding an extra layer of challenge and accomplishment.
* **Multiplayer and Community Features:** The platform will emphasize social and community-driven aspects, allowing players to connect and compete with friends or meet new opponents in real-time multiplayer matches. To further promote interaction, features such as chat rooms, user profiles, and community-driven events will be introduced, nurturing a vibrant community of gamers. Whether through cooperative gameplay or head-to-head challenges, players will be encouraged to engage with others in dynamic and competitive ways.
* **User Profiles, Achievements, and Gamification:** Each user will have a customizable profile, where they can showcase their achievements, high scores, and gaming history. Gamification elements, such as badges, trophies, and level-up systems, will incentivize players to explore more games, master their favorite titles, and unlock unique rewards. These elements will create a sense of personal progression and keep players engaged over time.
* **High-Quality Graphics and Performance Optimization:** The platform will emphasize not only the quantity of games but also the quality of the gaming experience. Games will feature sharp, high-resolution graphics with smooth animations, and performance optimization will ensure low latency and minimal disruptions during gameplay. The platform’s design will be sleek, modern, and user-friendly, ensuring that players can easily navigate between different sections without hassle.
* **Regular Updates and Expanding Game Library:** To keep the platform fresh and exciting, new games will be added regularly, along with periodic updates to existing titles. Developers will have the flexibility to introduce new features, game modes, and seasonal events that align with current gaming trends, ensuring the platform remains relevant and engaging over time.

## 2.2 Description of the current system

**User Profile Management**

* **Profile Updates**: Users can update their profiles, including personal information and gaming preferences. This data is stored in a database, ensuring that user details are accessible and up-to-date.
* **Cover Image and Profile Picture**: Users have the ability to upload a cover image and profile picture, allowing for personalization of their profiles and enhancing their gaming identity.

**Friend List Management**

* **Friend List Updates**: Users can send, accept, or reject friend requests, facilitating social interactions within the gaming community. The system keeps track of these changes, ensuring that friend lists are current and accurate.

**Game Score Tracking**

* **Score Updates**: The system allows users to update their game scores in real-time. Scores are recorded in the database, enabling players to track their performance and achievements over time.
* **Score Display**: Users can view their scores and statistics directly on their profiles, promoting competition and engagement among friends.

**4. Data Handling**

* **Database Integration**: The website utilizes a database to manage user profiles, friend lists, and game scores. This ensures efficient data retrieval and storage, enhancing the overall user experience.

## 2.3 Limitation of the current system

**Performance Issues**

* **Slow Load Times**: Many gaming websites experience prolonged load times, frustrating users and leading to increased bounce rates. Delays in loading game assets disrupt the gaming experience, making it less enjoyable.
* **Lag During Gameplay**: Performance lag can occur due to heavy resource demands or inefficient server responses, resulting in choppy gameplay that negatively impacts user satisfaction.
* **Compatibility Problems**: Some platforms may not be optimized for all devices, leading to performance degradation on mobile or older hardware, limiting accessibility for users.

**Limited Interactivity**

* **Underutilization of JavaScript**: Traditional gaming websites often fail to leverage JavaScript's full potential, resulting in static gameplay experiences. This limits the depth of interaction that could enhance user engagement, such as dynamic content updates and real-time gameplay features.
* **Inflexible User Interfaces**: Websites may have rigid interfaces that do not adapt to different user inputs or preferences, making it difficult for players to customize their experiences or access various functionalities easily.

**Outdated Design**

* **Non-Responsive Layouts**: Many existing platforms use outdated design principles that do not scale well across different devices and screen sizes. This leads to poor usability on mobile devices, where a significant portion of gaming traffic originates.
* **Inefficient Navigation**: An outdated user interface can complicate navigation, making it challenging for users to find games or features quickly. An intuitive design is crucial for retaining users and enhancing their overall experience.
* **Lack of Certain Modern Features**
* **Limited Leaderboard Functionality**: While some platforms have leaderboards, many do not provide comprehensive features that encourage social interaction and competition, essential for community building within gaming.
* **Insufficient Real-Time Updates**: Current systems may lack real-time updates on game events or community activities, leading to a disconnect between players and the gaming community.

**User Engagement**

* **Minimal Interaction Opportunities**: Existing platforms often provide limited options for user interaction, such as in-game chat or social features. This can result in a solitary gaming experience, reducing the community aspect that many players seek.
* **Limited Community Building Features**: Although some platforms include friend lists, the absence of messaging systems or collaborative gameplay modes can still lead to a less engaging experience. Players may feel isolated, leading to decreased retention rates.

## 2.4 Description of the proposed system

* WebGame Platform: This is a cutting-edge online gaming website that features games exclusively developed using JavaScript. The platform includes a variety of games, from fast-paced 2D adventures to multiplayer duels, providing diverse gaming experiences.
* Interactive Gameplay: Leveraging JavaScript, the platform ensures smooth performance and highly interactive gameplay, offering players an immersive experience.
* Real-Time Features: Users can track their scores, compete on leaderboards, and experience real-time updates and challenges.
* Modern Design: The website features a responsive and modern design that adapts to different devices, providing a seamless user experience.
* User Engagement: Players can easily access games, view detailed game descriptions, and participate in interactive features to enhance their gaming experience.

## 2.5 Advantages of the proposed system

**1. Enhanced Gaming Experience**

The integration of JavaScript enables rich, interactive gameplay with smooth performance and engaging graphics, creating a more immersive environment for players.

**2. Responsive Design**

The website features a modern design that adapts to various devices and screen sizes. This ensures users can access the platform seamlessly on desktops, tablets, and smartphones, enhancing accessibility.

**3. Real-Time Interaction**

With features like score tracking, users can see their progress instantly. This real-time feedback keeps players engaged and motivated to improve their gaming skills.

**4. Intuitive Navigation**

The platform is designed with user-friendliness in mind, making it easy for users to find and play games. Clear navigation enhances the overall user experience.

**5. Friend List Functionality**

The ability to add friends allows users to easily connect and see who is part of their gaming network. This feature fosters social interaction, enhancing the community aspect of the platform.

**6. Improved Performance**

Utilizing modern web technologies helps reduce load times and lag, ensuring a smoother gaming experience, which is essential for player satisfaction.

**Efficient Data Management**

The proposed system incorporates a robust database to manage user profiles and friend connections efficiently. This ensures accurate and real-time updates, enhancing overall functionality.

# 3.Requirement Specification

## 3.1 Software Requirements

Operating System: Microsoft Windows 10 or any other modern operating system (Windows, macOS, Linux).

Application Software:

Front End: HTML, CSS, JavaScript (for game development and interactivity), Bootstrap (for responsive design).

Back End: Node.js (optional, if backend features are needed), Express.js (for server-side logic), and MongoDB (for database, if needed).

Platform: VS Code (or any preferred code editor/IDE).

## 3.2 Hardware Requirements

A computer with:

RAM: 4GB or more.

Processor: 2GHz or higher.

Internet Connection: Stable and reliable for development and testing.

Disk Space: 1GB or more for code and assets.

Keyboard & Mouse: Standard or ergonomic as preferred.

## 3.3 Data Requirements

User Data:

Name: varchar

Username: varchar

Email ID: varchar

DateOfBirth:Varchar

Password: varchar (hashed for security)

High Score: int (for tracking player achievements)

Last Login: datetime

Game Data:

Game ID: varchar

Game Title: varchar

Game Description: Text

Game Image: varchar (URL or file path)

Game Category: varchar

Friends Data

FriendAccepted: bool

FriendRejected: bool

Pending: bool

## 3.4 Fact Finding Question

What are the core features and functionalities needed for the gaming website?

How can the website handle user authentication and game score tracking?

What performance requirements are necessary to ensure smooth gameplay?

How can the website support real-time updates and interactions?

What security measures are needed to protect user data and game assets?

Is the website optimized for different devices and screen sizes?

What tools and technologies are required for ongoing maintenance and updates?

# 4.System Design

## 4.1 Event Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr  N  o. | Event | Trigger | Sour ce | Activity | Response | Destination |
| 1 | User Registration | btndonorregi  ster | user | Filling some  Personal  Details | If successful, redirect to homepage; else show | signup.html |
| 2 | User Login | btnLogin | User | Entering the  registered  Email ID  &  Password | If credentials are correct, log in; else show error | login.html |
| 3 | Start Game | btnStartGame | Admi n | Clicking 'Start Game' | Load the selected game | game.html |
| 4 | User Profile Update | UpdateProfile | User | Updating profile information | f successful, update profile; else show error | profile.html |
| 5 | Forward | 'Right Arrow' key pressed | user | player move forward | forward | Game Screen |
| 6 | Backward | 'Left Arrow' key pressed | user | player move bACKWARD | backward | Game Screen |
| 7 | Jump | 'Space' key pressed | user | player move up | jump | Game Screen |
| 8 | sit | 'Down Arrow' key pressed | user | player | froward | Game Screen |
| 9 | attack | ‘Space Bar;  Key pressed | user | player | attack | Game Screen |
| 10 | Logout | logoutbtn | user | Logout | Takes u to signin page | Signin.html |
| 11 | About | aboutbtn | user | About page | Take you to about page | About.html |

Table Event Table

## 4.2 Class Diagram

A diagram of a computer

Description automatically generated with medium confidence

Figure Class Diagram

## 4.3 Use Case Diagram

### 4.3.1 Use Case Diagram for User Authentication

A close-up of a computer screen

Description automatically generated

Figure User Authentication Use Case Diagram

**4.3.2 Use Case Diagram for Home Page with About Page Management**

A diagram of a company

Description automatically generated

Figure Home Page Use Case Diagram

4.3.3 Use Case Diagram for Account Page with Profile and Friends Management

A screenshot of a chat

Description automatically generated

Figure Account Page Use Case Diagram

### 4.3.4 Use Case Diagram for Game Interaction

A diagram of a diagram

Description automatically generated with medium confidence

Figure Game Interaction Use Case Diagram

## 4.4 Sequence Diagram

### 4.4.1 SignIn

A screen shot of a computer screen

Description automatically generated

Figure Signin Sequence Diagram

### 4.4.2 SignUp

A diagram of a sign up page

Description automatically generated

Figure Signup Sequence Diagram

### 4.4.3 Homepage

A diagram of a computer program

Description automatically generated with medium confidence

Figure Homepage Sequence Diagram

### 4.4.4 account

A screenshot of a computer screen

Description automatically generated

Figure Account Sequence Diagram

### 4.4.5 game

A screen shot of a computer program

Description automatically generated

Figure Game Sequence Diagram Diagram

## 4.5 Activity Diagram

## 

### 4.5.1 Activity Diagram for SignIn

A flowchart of a computer program

Description automatically generated

Figure SignIn Activity Diagram

### 4.5.2 Activity Diagram for SignUp

A diagram of a system

Description automatically generated

Figure SignUp Activity Diagram

### 4.5.3 Activity Diagram forhomepage

A screenshot of a computer

Description automatically generated

Figure Home Page Activity Diagram

### 4.5.4 Activity Diagram account page

A screen shot of a computer screen

Description automatically generated

Figure Account Page Activity Diagram

### 4.5.5 Activity Diagram for speedy pup

A screenshot of a computer screen

Description automatically generated

Figure Speedy Pup Activity Diagram

### 4.5.6 Activity Diagram for gorilla clash

A screenshot of a diagram

Description automatically generated

Figure Gorilla Clash Activity Diagram

### 4.5.7 Activity Diagram for samurai

A screenshot of a computer screen

Description automatically generated

Figure Samurai Activity Diagram

## 4.6 State Diagram

### 4.6.1 Home State Diagram

A diagram of a diagram

Description automatically generated

Figure Home State Diagram

### 4.6.2 Account State Diagram

A diagram of a website

Description automatically generated

Figure Account State Diagram

### 4.6.3 Speedy Pup game State Diagram

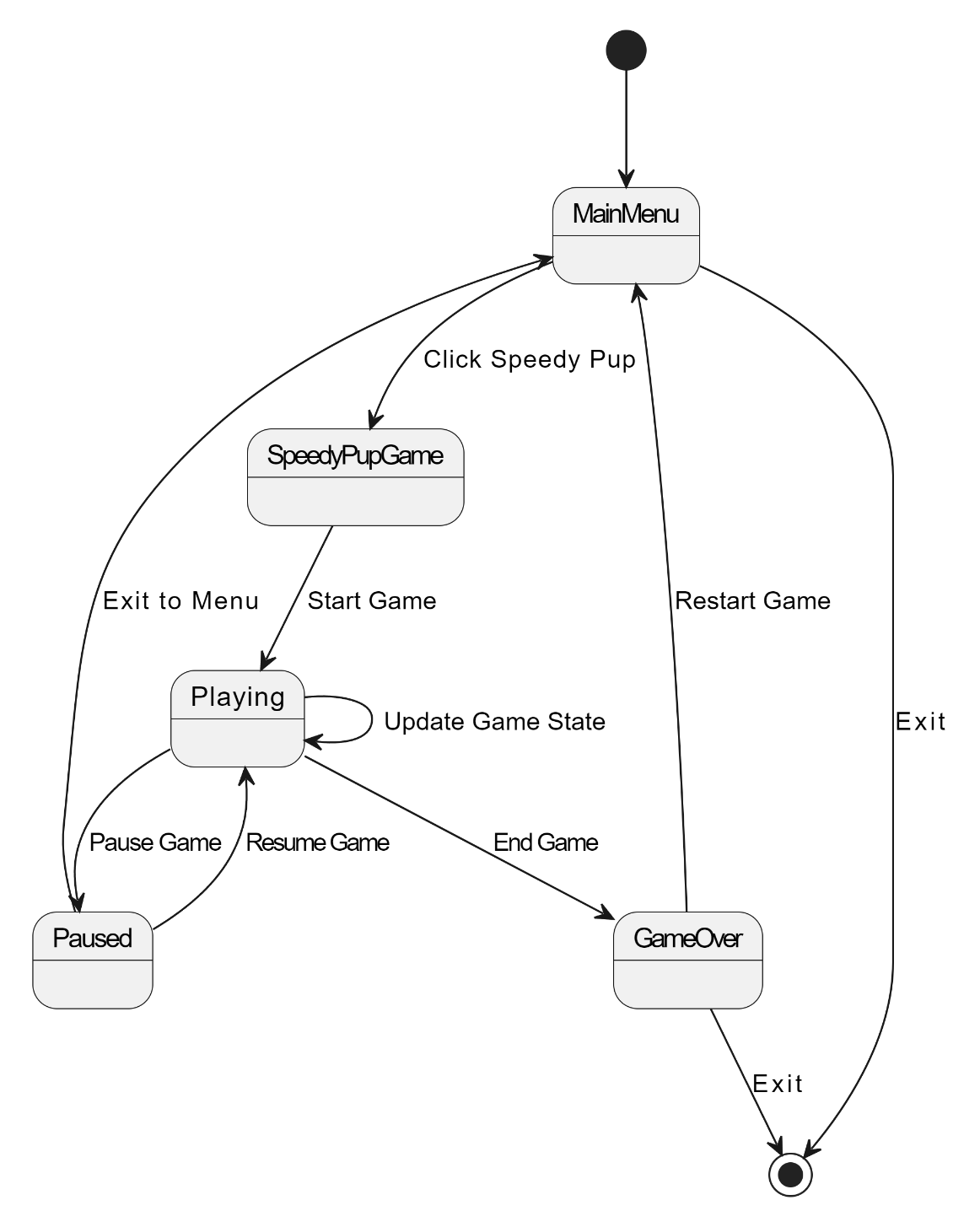


Figure Speedy Pup Game State Diagram

### 4.6.4 Gorilla Game State Diagram

A diagram of a game

Description automatically generated

Figure Gorilla Game State Diagram

### 4.6.5 Samurai Game State Diagram

A diagram of a game

Description automatically generated

Figure Samurai Game State Diagram

## 4.7 Package Diagram

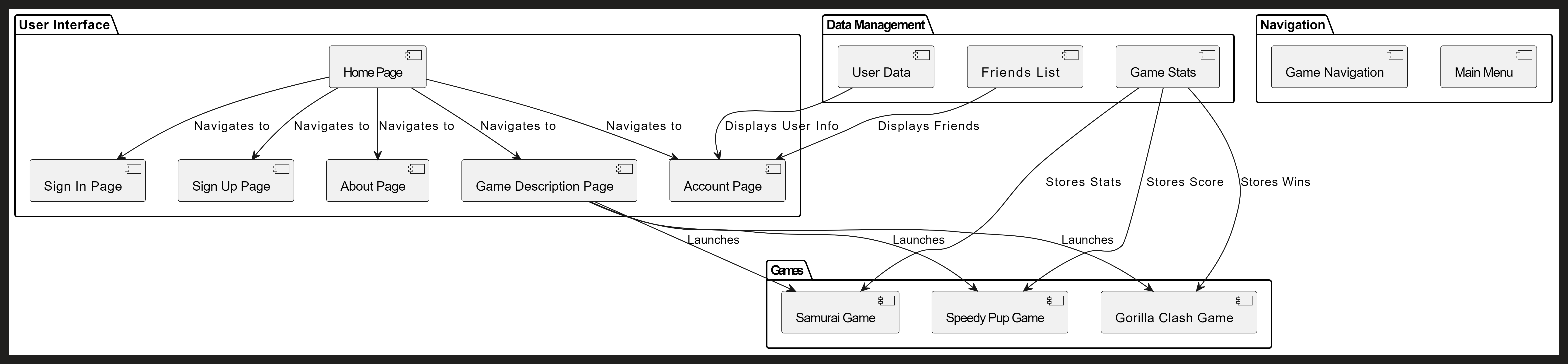


Figure Package Diagram

## 4.8 Component Diagram

A black and white image of a diagram

Description automatically generated with medium confidence

Figure Component Diagram

## 4.9 Deployment Diagram

A screenshot of a computer screen

Description automatically generated

Figure Deployment Diagram

## 4.10 Database Design

A screenshot of a computer

Description automatically generated

Figure Database Design

*A screenshot of a computer

Description automatically generated*

Figure User collection

A screenshot of a computer code

Description automatically generated

Figure frindrequest collection accepted

A computer code with numbers and symbols

Description automatically generated with medium confidence

Figure frindrequest collection rejected

A screenshot of a computer code

Description automatically generated

Figure games collection dogGame

*A screenshot of a computer

Description automatically generated*

Figure games collection gorillaGame

# 5.System Implementation

(FRONTEND)

## 5.1 account.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>GAMES-profile</title>

    <link rel="stylesheet" href="account.css">

</head>

<body>

    <header>

        <div class="logo">

            <div class="logo-left">THE</div>

            <div class="logo-right">BLEEP</div>

        </div>

        <nav class="main-nav">

            <ul>

                <li><a href="../home/home.html">HOME</a></li>

                <li><a href="#">ABOUT</a></li>

                <li><a href="#">UPCOMING</a></li>

            </ul>

            <div class="current-date">

                <span id="date-display"></span> <!-- This is where the date will be displayed -->

            </div>

        </nav>

    </header>

    <main class="container">

        <div class="profile">

            <div id="imagePreview" class="cover" onclick="changeImage1()"></div>

            <input type="file" id="imageInput1" style="display:none;" accept="image/\*" />

            <div id="profile-img" class="pfp" onclick="changeImage2()"></div>

            <input type="file" id="imageInput2" style="display:none;" accept="image/\*" />

            <div class="profile-details">

                <div class="info">

                    <div id="profile-name">ANURAG K</div>

                    <div id="email">example@gmail.com</div>

                    <div id="birthday">01-01-2020</div>

                    <div id="bio">"Always up for a challenge!"</div>

                </div>

                <div class="game">

                    <h1>SPEEDY-PUP</h1>

                    <div id="high-score">High Score: 12000</div><BR>

                    <h1>GORILLA-CLASH</h1>

                    <div id="high-scoreG">High Score: 12000</div><BR>

                </div>

                <!-- <div id="profile-name">ANURAG K</div>

                <div id="high-score">High Score: 12000</div>

                <div id="level">Level: 25</div>

                <div id="achievements">Achievements: 10</div>

                <div id="games-played">Games Played: 150</div>

                <div id="rank">Rank: #42</div>

                 -->

            </div>

        </div>

        <div class="friends">

            <div class="heading">FRIENDS</div>

            <div class="friends-search">

                <input type="search" id="friendName"/><button onclick="sendFriendRequest()">Search</button>

            </div>

            <div id="friends-list"></div>

            <div id="friend-request"></div>

        </div>

    </main>

    <nav class="secondary-nav">

        <ul>

            <li>

                <a class="play" href="#">PLAY</a>

                <div class="popup">

                    <ul>

                        <li class="gc"><a href="../gorillaGame/index2.html">GORILLA CLASH</a></li>

                        <li class="sp"><a href="../dogGame/index.html">SPEEDY PUP</a></li>

                        <li class="su"><a href="../samuriaGame/index (2).html">SAMURAI'S</a></li>

                    </ul>

                </div>

            </li>

            <li class="p"><a href="#">PROFILE</a></li>

            <li class="l"><a onclick="logout()"">LOGOUT</a></li>

        </ul>

    </nav>

    <script src="account.js"></script>

</body>

</html>

## 5.2 index.html ( speedy pup game )

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game</title>

    <link rel="stylesheet" href="style.css"/>

</head>

<body>

    <canvas id="canvas1"></canvas>

    <!-- Game elements -->

    <img id="player" src="assets/player.png">

    <img id="layer1" src="assets/layer-1.png">

    <img id="layer2" src="assets/layer-2.png">

    <img id="layer3" src="assets/layer-3.png">

    <img id="layer4" src="assets/layer-4.png">

    <img id="layer5" src="assets/layer-5.png">

    <img id="enemy\_fly" src="assets/enemy\_fly.png">

    <img id="enemy\_plant" src="assets/enemy\_plant.png">

    <img id="enemy\_spider" src="assets/enemy\_spider\_big.png">

    <img id="fire" src="assets/fire.png">

    <!-- Display for score and username -->

    <div id="scoreDisplay"></div>

    <div id="usernameDisplay"></div> <!-- Element to display the username -->

    <script>

        const token = localStorage.getItem("token");

        document.addEventListener('DOMContentLoaded', () => {

            // Fetch and display the username when the page loads

            displayUsername();

        });

        // Function to display the username

        function displayUsername() {

            const username = localStorage.getItem('username');

            const usernameDisplay = document.getElementById('usernameDisplay');

            if (username) {

                usernameDisplay.innerHTML = `Logged in as: ${username}`;

            } else {

                usernameDisplay.innerHTML = `Not logged in`;

            }

        }

        // Listen for the custom event dispatched from main.js

        window.addEventListener('levelFinished', function(event) {

            const score = event.detail.score;

            const scoreDisplay = document.getElementById('scoreDisplay');

            scoreDisplay.innerHTML = `Score: ${score}`;

            // Call the function to send the score to the database

            sendScoreToDatabase(score);

        });

        // Function to send score to the database

        function sendScoreToDatabase(score) {

            const username = localStorage.getItem('username'); // Fetching username from Local Storage

            const apiUrl = '<http://localhost:5000/api/game/high-score>'; // Replace with your actual endpoint

            if (!username) {

                console.error('Username not found. Please log in.');

                return;

            }

            fetch(apiUrl, {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                    'Authorization': `Bearer ${token}` // Include the token in the request

                },

                body: JSON.stringify({

                    username: username,

                    gameName: 'dogGame',

                    score: score,

                }),

            })

            .then(response => {

                if (!response.ok) {

                    throw new Error(`HTTP error! Status: ${response.status}`);

                }

                return response.json();

            })

            .then(data => {

                console.log('Score saved:', data);

            })

            .catch(error => {

                console.error('Error saving score:', error);

            });

        }

    </script>

    <script type="module" src="main.js"></script>

</body>

</html>

## 5.3 index2.html ( gorilla game )

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>Gorillas</title>

    <link rel="stylesheet" href="index.css" />

    <script src="index.js" defer></script>

  </head>

  <body>

    <canvas id="game"></canvas>

    <svg width="200" height="250" id="windmill">

      <defs>

        <path id="arm" d="M -7 -20 C -7 -10 7 -10 7 -20 L 2 -80 L -2 -80" />

      </defs>

      <g transform="translate(100, 100)">

        <g id="windmill-head">

          <circle r="8"></circle>

          <use href="#arm" />

          <use href="#arm" transform="rotate(+120)" />

          <use href="#arm" transform="rotate(-120)" />

        </g>

      </g>

      <path

        transform="translate(100, 0)"

        d="M -7 250 L 7 250 L 3 115 L -3 115"

      ></path>

    </svg>

    <div id="wind-info">Wind Speed: <span id="wind-speed">0</span></div>

    <div id="info-left">

      <h3><span class="name">Player</span></h3>

      <p>Angle: <span class="angle">0</span>°</p>

      <p>Velocity: <span class="velocity">0</span></p>

    </div>

    <div id="info-right">

      <h3><span class="name">Computer</span></h3>

      <p>Angle: <span class="angle">0</span>°</p>

      <p>Velocity: <span class="velocity">0</span></p>

    </div>

    <div id="instructions">

      <h3 id="game-mode">Player vs. Computer</h3>

      <h1>Drag the bomb to aim!</h1>

    </div>

    <div id="bomb-grab-area"></div>

    <div id="congratulations">

      <h1><span id="winner">?</span> won!</h1>

      <p>

        yeyy!!

      </p>

      <div class="dropdown">

        <button class="dropbtn">New Game</button>

        <div class="dropdown-content">

          <a href="#" class="single-player">Single Player</a>

          <a href="#" class="two-players">Two-Player</a>

          <a href="#" class="auto-play">Autoplay</a>

        </div>

      </div>

    </div>

    <div id="settings">

      <div class="dropdown">

        <button class="dropbtn">New Game</button>

        <div class="dropdown-content">

          <a href="#" class="single-player">Single Player</a>

          <a href="#" class="two-players">Two-Players</a>

          <a href="#" class="auto-play">Autoplay</a>

        </div>

      </div>

      <button id="color-mode">Dark Mode</button>

    </div>

    <button id="fullscreen" onclick="toggleFullscreen()">

      <svg width="30" height="30">

        <path

          id="enter-fullscreen"

          stroke="white"

          stroke-width="3"

          fill="none"

          d="

            M 10, 2 L 2,2 L 2, 10

            M 20, 2 L 28,2 L 28, 10

            M 28, 20 L 28,28 L 20, 28

            M 10, 28 L 2,28 L 2, 20"

        />

        <path

          id="exit-fullscreen"

          stroke="transparent"

          stroke-width="3"

          fill="none"

          d="

            M 10, 2 L 10,10 L 2, 10

            M 20, 2 L 20,10 L 28, 10

            M 28, 20 L 20,20 L 20, 28

            M 10, 28 L 10,20 L 2, 20"

        />

      </svg>

    </button>

  </body>

</html>

## 5.4 home.html ( samurai game )

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="preconnect" href="https://fonts.googleapis.com" />

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

<link

href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap"

rel="stylesheet"

/>

</head>

<body>

<!-- red container div -->

<div style="position: relative; display: inline-block">

<!-- smaller red container div -->

<div

style="

position: absolute;

display: flex;

width: 100%;

align-items: center;

padding: 20px;

"

>

<!-- player health -->

<div

style="

position: relative;

width: 100%;

display: flex;

justify-content: flex-end;

border-top: 4px solid white;

border-left: 4px solid white;

border-bottom: 4px solid white;

"

>

<div style="background-color: red; height: 30px; width: 100%"></div>

<div

id="playerHealth"

style="

position: absolute;

background: #818cf8;

top: 0;

right: 0;

bottom: 0;

width: 100%;

"

></div>

</div>

<!-- timer -->

<div

id="timer"

style="

background-color: black;

width: 100px;

height: 50px;

flex-shrink: 0;

display: flex;

align-items: center;

justify-content: center;

color: white;

border: 4px solid white;

"

>

10

</div>

<!-- enemy health -->

<div

style="

position: relative;

width: 100%;

border-top: 4px solid white;

border-bottom: 4px solid white;

border-right: 4px solid white;

"

>

<div style="background-color: red; height: 30px"></div>

<div

id="enemyHealth"

style="

position: absolute;

background: #818cf8;

top: 0;

right: 0;

bottom: 0;

left: 0;

"

></div>

</div>

<div></div>

</div>

<div

id="displayText"

style="

position: absolute;

color: white;

align-items: center;

justify-content: center;

top: 0;

right: 0;

bottom: 0;

left: 0;

display: none;

"

>

Tie

</div>

<canvas></canvas>

</div>

<script

src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.9.1/gsap.min.js"

integrity="sha512-H6cPm97FAsgIKmlBA4s774vqoN24V5gSQL4yBTDOY2su2DeXZVhQPxFK4P6GPdnZqM9fg1G3cMv5wD7e6cFLZQ=="

crossorigin="anonymous"

referrerpolicy="no-referrer"

></script>

<script src="js/utils.js"></script>

<script src="js/classes.js"></script>

<script src="index.js"></script>

</body>

</html>

## 5.5 home.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>GAMES</title>

    <link rel="stylesheet" href="home.css">

    <script src="home.js" defer></script>

</head>

<body>

    <header>

        <div class="logo">

            <div class="logo-left">THE</div>

            <div class="logo-right">BLEEP</div>

        </div>

        <nav class="main-nav">

            <ul>

                <li><a href="#">HOME</a></li>

                <li><a href="../about/about.html">ABOUT</a></li>

                <li><a href="#">UPCOMING</a></li>

            </ul>

            <div class="current-date">

                <span id="date-display"></span> <!-- This is where the date will be displayed -->

            </div>

        </nav>

        <!-- Note: No change needed in the HTML for secondary-nav -->

    </header>

    <main class="container">

        <div class="left-side">

            <div class="featured-item">

                <div class="topleft-f"></div>

                <div class="topright-f"></div>

                <div class="bottomleft-f"></div>

                <div class="bottomright-f"></div>

                <div class="featured-item-content">

                    <p> -- J A v a S c r i P t    m a d E--</p>

                    <h2> FEATURED – GAMES</h2>

                    <div class="featuredimage">

                        <img src="/assets/speedypup.jpg" alt="Featured Image">

                    </div>

                </div>

            </div>

        </div>

        <div class="right-side">

            <a href="../dogInfo/dogGame.html" class="grid-item-link">

                <div class="grid-item">

                    <div class="topleft-g"></div>

                    <div class="topright-g"></div>

                    <div class="bottomleft-g"></div>

                    <div class="bottomright-g"></div>

                    <div class="grid-item-content" >

                        <p>Endless Game</p>

                        <h3>Speedy Pup – Doggy</h3>

                        <p>Electronics</p>

                        <img src="./assets/leodog1.jpg" alt="Image Description" class="grid-item-image">

                    </div>

                </div>

            </a>

            <a href="../gorillaInfo/gorillainfo.html" class="grid-item-link">

                <div class="grid-item">

                    <div class="topleft-g"></div>

                    <div class="topright-g"></div>

                    <div class="bottomleft-g"></div>

                    <div class="bottomright-g"></div>

                    <div class="grid-item-content">

                        <p>Multiplayer</p>

                        <h3>Gorilla Clash – Monkey</h3>

                        <p>Indie / Punk / Alternative</p>

                        <img src="./assets/gori2.jpg" alt="Image Description" class="grid-item-image">

                    </div>

                </div>

            </a>

            <a href="../samuriaGame/index (2).html" class="grid-item-link">

                <div class="grid-item">

                    <div class="topleft-g"></div>

                    <div class="topright-g"></div>

                    <div class="bottomleft-g"></div>

                    <div class="bottomright-g"></div>

                    <div class="grid-item-content">

                        <p>Two-player</p>

                        <h3>Samurai's</h3>

                        <p>Psych / Kraut / Progressive</p>

                        <img src="./assets/samu.jpg" alt="Image Description" class="grid-item-image">

                    </div>

                </div>

            </a>

            <div class="grid-item">

                <div class="topleft-g"></div>

                <div class="topright-g"></div>

                <div class="bottomleft-g"></div>

                <div class="bottomright-g"></div>

                <div class="grid-item-content">

                    <h3>PATCH NOTES </h3>

                    <p>Spell Blanket</p>

                    <p>UPDATED</p>

                </div>

            </div>

        </div>

    </main>

    <nav class="secondary-nav">

        <ul>

            <li class="playing">

                <a href="#" class="play">PLAY</a>

                <div class="popup">

                    <ul>

                        <li class="gc"><a href="../gorillaGame/index2.html">GORILLA CLASH</a></li>

                        <li class="sp"><a href="../dogGame/index.html">SPEEDY PUP</a></li>

                        <li class="su"><a href="../samuriaGame/index (2).html">SAMURAI'S</a></li>

                    </ul>

                </div>

            </li>

            <li class="profile"><a href="../account/account.html">PROFILE</a></li>

            <li class="logout"><a onclick="logout()">LOGOUT</a></li>

        </ul>

    </nav>

</body>

</html>

## 5.6 login.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="login.css">

    <title>Login</title>

</head>

<body>

    <div class="container">

        <div class="loginbox">

            <div class="heading">Log in to your account</div>

            <div class="login-fields">

                <div class="email">

                    <label for="email">Email</label>

                    <input type="text" id="email" name="email" placeholder="Email">

                </div>

                <div class="password">

                    <label for="password">Password</label>

                    <input type="password" id="password" name="password" placeholder="Password">

                </div>

                <div class="forgot-password">

                    <a href="#" class="forgot-link">Forgot Password?</a>

                </div>

                <div class="sub">

                    <button type="submit" id="login-button">LogIn</button>

                </div>

                <div class="signup-link">

                    Don't have an account? <a href="../signup/signup.html">Sign up</a>

                </div>

            </div>

        </div>

    </div>

    <script>

        document.getElementById('login-button').addEventListener('click', async (event) => {

            event.preventDefault();

            // Get the input values

            const email = document.getElementById('email').value;

            const password = document.getElementById('password').value;

            // Validate input fields

            if (!email || !password) {

                alert('Please fill in all fields.');

                return;

            }

            try {

                // Send login request to the backend

                const response = await fetch('http://localhost:5000/api/auth/login', {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/json',

                    },

                    body: JSON.stringify({ email, password }),

                });

                const data = await response.json();

                if (response.ok) {

                    // Handle successful login, e.g., store token

                    alert('Login successful!');

                    localStorage.setItem('username',data.username);

                    localStorage.setItem('token', data.token);

                    // Redirect to another page if needed

                    window.location.href = '../home/home.html'; // Change this to your desired redirect location

                } else {

                    // Display error message

                    alert(data.message || 'Login failed');

                }

            } catch (error) {

                console.error('Error:', error);

                alert('An error occurred. Please try again later.');

            }

        });

    </script>

</body>

</html>

## 5.7 signup.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="signup.css"> <!-- Assuming you have your CSS in signup.css -->

    <title>Sign Up</title>

</head>

<body>

    <div class="container">

        <div class="signupbox">

            <div class="heading">Sign up and start having fun!</div>

            <div class="birthdate">

                <label for="landing-birthday" class="bday">Birthday</label>

                <div class="selectors">

                    <div class="month">

                        <select class="Lselect" id="MonthDropdown" name="birthdayMonth" required>

                            <option value="" disabled selected>Month</option>

                            <option value="Jan">January</option>

                            <option value="Feb">February</option>

                            <option value="Mar">March</option>

                            <option value="Apr">April</option>

                            <option value="May">May</option>

                            <option value="Jun">June</option>

                            <option value="Jul">July</option>

                            <option value="Aug">August</option>

                            <option value="Sep">September</option>

                            <option value="Oct">October</option>

                            <option value="Nov">November</option>

                            <option value="Dec">December</option>

                        </select>

                    </div>

                    <div class="day">

                        <select class="Mselect" id="DayDropdown" name="birthdayDay" required>

                            <option value="" disabled selected>Day</option>

                            <!-- Days 1 to 31 -->

                            <script>

                                for (let i = 1; i <= 31; i++) {

                                    document.write(`<option value="${String(i).padStart(2, '0')}">${String(i).padStart(2, '0')}</option>`);

                                }

                            </script>

                        </select>

                    </div>

                    <div class="year">

                        <select class="Rselect" id="YearDropdown" name="birthdayYear" required>

                            <option value="" disabled selected>Year</option>

                            <!-- Years 2024 to 2020 -->

                            <script>

                                for (let i = 0; i < 5; i++) {

                                    let year = 2024 - i;

                                    document.write(`<option value="${year}">${year}</option>`);

                                }

                            </script>

                        </select>

                    </div>

                </div>

            </div>

            <div class="ups">

                <div class="email">

                    <label for="email">Email</label>

                    <input type="text" id="email" name="email" placeholder="Email" required>

                </div>

                <div class="uname">

                    <label for="username">Username</label>

                    <input type="text" id="username" name="username" placeholder="Username" required>

                </div>

                <div class="pass">

                    <label for="password">Password</label>

                    <input type="password" id="password" name="password" placeholder="Password" required>

                </div>

                <div class="cpass">

                    <label for="confirmpassword">Confirm Password</label>

                    <input type="password" id="confirmpassword" name="confirmpassword" placeholder="Confirm password" required>

                </div>

                <div class="sub">

                    <button type="submit" id="signup-button">Sign Up</button>

                </div>

            </div>

        </div>

    </div>

    <script>

        // Function to validate form inputs and send the signup request

        document.getElementById('signup-button').addEventListener('click', async (event) => {

            event.preventDefault(); // Prevent the default form submission

            // Get form values

            const email = document.getElementById('email').value.trim();

            const username = document.getElementById('username').value.trim();

            const password = document.getElementById('password').value;

            const confirmPassword = document.getElementById('confirmpassword').value;

            const birthdayMonth = document.getElementById('MonthDropdown').value;

            const birthdayDay = document.getElementById('DayDropdown').value;

            const birthdayYear = document.getElementById('YearDropdown').value;

            // Validation checks

            if (!email || !username || !password || !confirmPassword || !birthdayMonth || !birthdayDay || !birthdayYear) {

                alert('Please fill in all fields.');

                return;

            }

            // Password confirmation check

            if (password !== confirmPassword) {

                alert('Passwords do not match.');

                return;

            }

            // Prepare data to send to the backend

            const data = {

                email,

                username,

                password,

                birthday: `${birthdayDay}-${birthdayMonth}-${birthdayYear}`

            };

            try {

                // Send the signup request to the backend

                const response = await fetch('http://localhost:5000/api/auth/signup', {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/json',

                    },

                    body: JSON.stringify(data),

                });

                const result = await response.json();

                // Handle success or error from the backend

                if (response.ok) {

                    alert(result.message || 'Sign up successful!');

                    window.location.href = '../login/login.html'; // Redirect on success

                } else {

                    alert(result.message || 'Sign up failed. Please try again.');

                }

            } catch (error) {

                console.error('Error during sign up:', error);

                alert('An error occurred. Please try again later.');

            }

        });

    </script>

</body>

</html>

## 5.8 about.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>GAMES</title>

    <link rel="stylesheet" href="about.css">

    <script src="about.js" defer></script>

</head>

<body>

    <header>

        <div class="logo">

            <div class="logo-left">THE</div>

            <div class="logo-right">BLEEP</div>

        </div>

        <nav class="main-nav">

            <ul>

                <li><a href="../home/home.html">HOME</a></li>

                <li><a href="#">ABOUT</a></li>

                <li><a href="#">UPCOMING</a></li>

            </ul>

            <div class="current-date">

                <span id="date-display"></span> <!-- This is where the date will be displayed -->

            </div>

        </nav>

        <!-- Note: No change needed in the HTML for secondary-nav -->

    </header>

    <main class="container">

        <section class="scrolling-text">

            <div class="scrolling-wrapper">

                <div class="scrolling-content">

                    <span>Explore the Adventure</span>

                    <span>Unleash Your Skills</span>

                    <span>Master the Game</span>

                    <span>Compete with Friends</span>

                    <span>Unlock New Levels</span>

                    <span>Challenge the Best</span>

                    <span>Sharpen Your Reflexes</span>

                    <span>Reach the Top</span>

                    <span>Discover Hidden Secrets</span>

                    <span>Take on the Challenge</span>

                    <span>Embrace the Journey</span>

                    <span>Victory Awaits</span>

                </div>

                <!-- Duplicate the content for seamless effect -->

                <div class="scrolling-content">

                    <span>Explore the Adventure</span>

                    <span>Unleash Your Skills</span>

                    <span>Master the Game</span>

                    <span>Compete with Friends</span>

                    <span>Unlock New Levels</span>

                    <span>Challenge the Best</span>

                    <span>Sharpen Your Reflexes</span>

                    <span>Reach the Top</span>

                    <span>Discover Hidden Secrets</span>

                    <span>Take on the Challenge</span>

                    <span>Embrace the Journey</span>

                    <span>Victory Awaits</span>

                </div>

            </div>

        </section>

    </main>

    <section class="secondary-scrolling-text">

        <div class="scrolling-wrapper2">

            <div class="scrolling-content2">

                <span>New Challenges Await</span>

                <span>Play Now and Win</span>

                <span>Join the Adventure</span>

                <span>Level Up</span>

                <span>Test Your Skills</span>

                <span>Become the Champion</span>

                <span>Enter the Arena</span>

                <span>Rise to Victory</span>

            </div>

            <div class="scrolling-content2">

                <span>New Challenges Await</span>

                <span>Play Now and Win</span>

                <span>Join the Adventure</span>

                <span>Level Up</span>

                <span>Test Your Skills</span>

                <span>Become the Champion</span>

                <span>Enter the Arena</span>

                <span>Rise to Victory</span>

            </div>

        </div>

        <section class="about">

            <h1>About Us</h1>

            <p>

                The Bleep is a platform dedicated to simple yet exciting JavaScript-based games. Our collection features fun, interactive games that are perfect for taking a quick break or challenging yourself. With a focus on clean design and smooth gameplay, we aim to provide an enjoyable experience for users of all skill levels. As part of our ongoing development, we're constantly adding new games and enhancing features. Dive into The Bleep and start exploring our growing library of JavaScript games today!

            </p>

            <br><br>

            <p>

                Our current collection features two unique and thrilling games: Gorilla Clash and Speedy Pup.

    In Gorilla Clash, players engage in a fun and strategic battle between two gorillas, testing their skills and accuracy in both single-player and multiplayer modes. Meanwhile, Speedy Pup offers a high-speed adventure, where you guide a spirited dog through challenging levels, earning points and avoiding obstacles.

            </p>

        </section>

    </div>

    <nav class="secondary-nav">

        <ul>

            <li class="playing">

                <a href="#" class="play">PLAY</a>

                <div class="popup">

                    <ul>

                        <li class="gc"><a href="../gorillaGame/index2.html">GORILLA CLASH</a></li>

                        <li class="sp"><a href="../dogGame/index.html">SPEEDY PUP</a></li>

                        <li class="su"><a href="../samuriaGame/index (2).html">SAMURAI'S</a></li>

                    </ul>

                </div>

            </li>

            <li class="profile"><a href="../account/account.html">PROFILE</a></li>

            <li class="logout"><a onclick="logout()">LOGOUT</a></li>

        </ul>

    </nav>

</body>

</html>

## 5.9 dogInfo.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game Info - Gorilla Clash</title>

    <link rel="stylesheet" href="dogGame.css">

</head>

<body>

    <header>

        <div class="logo">

            <div class="logo-left">THE</div>

            <div class="logo-right">BLEEP</div>

        </div>

        <nav class="main-nav">

            <ul>

                <li><a href="../home/home.html">HOME</a></li>

                <li><a href="#">ABOUT</a></li>

                <li><a href="#">UPCOMING</a></li>

            </ul>

            <div class="current-date">

                <span id="date-display"></span>

            </div>

        </nav>

    </header>

    <main class="game-info-container">

        <div class="game-details">

            <div class="gname">

                <p>PC</p><h1 class="game-name">Speedy Pup</h1>

            </div>

            <p class="game-genre">Genre: Endless</p>

            <p class="game-players">Players: 1</p>

            <div class="gameplay-video">

                <video controls>

                    <source src="./assets/vid.mp4" type="video/mp4">

                    Your browser does not support the video tag.

                </video>

            </div>

        </div>

    </main>

    <nav class="secondary-nav">

        <ul>

            <li>

                <a href="../dogGame/index.html">PLAY</a>

            </li>

            <li><a href="../account/account.html">PROFILE</a></li>

            <li><a href="#">LOGOUT</a></li>

        </ul>

    </nav>

    <script src="dogGame.js"></script>

</body>

</html>

## 5.10 gorillaInfo.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game Info - Gorilla Clash</title>

    <link rel="stylesheet" href="gorillaInfo.css">

</head>

<body>

    <header>

        <div class="logo">

            <div class="logo-left">THE</div>

            <div class="logo-right">BLEEP</div>

        </div>

        <nav class="main-nav">

            <ul>

                <li><a href="../home/home.html">HOME</a></li>

                <li><a href="#">ABOUT</a></li>

                <li><a href="#">UPCOMING</a></li>

            </ul>

            <div class="current-date">

                <span id="date-display"></span>

            </div>

        </nav>

    </header>

    <main class="game-info-container">

        <div class="game-details">

            <div class="gname">

                <p>PC</p><h1 class="game-name">Gorilla Clash</h1>

            </div>

            <p class="game-genre">Genre: Action</p>

            <p class="game-players">Players: 1-2</p>

            <div class="gameplay-video">

                <video controls>

                    <source src="./assets/vid.mp4" type="video/mp4">

                    Your browser does not support the video tag.

                </video>

            </div>

        </div>

    </main>

    <nav class="secondary-nav">

        <ul>

            <li>

                <a href="../gorillaGame/index2.html">PLAY</a>

            </li>

            <li><a href="../account/account.html">PROFILE</a></li>

            <li><a href="#">LOGOUT</a></li>

        </ul>

    </nav>

    <script src="gorillaInfo.js"></script>

</body>

</html>

## 5.11 samuraiInfo.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Samurai Info</title>

    <link rel="stylesheet" href="samurai.css">

</head>

<body>

    <header>

        <div class="logo">

            <div class="logo-left">THE</div>

            <div class="logo-right">BLEEP</div>

        </div>

        <nav class="main-nav">

            <ul>

                <li><a href="../home/home.html">HOME</a></li>

                <li><a href="#">ABOUT</a></li>

                <li><a href="#">UPCOMING</a></li>

            </ul>

            <div class="current-date">

                <span id="date-display"></span>

            </div>

        </nav>

    </header>

    <main class="game-info-container">

        <div class="game-details">

            <div class="gname">

                <p>PC</p><h1 class="game-name">Samurai's</h1>

            </div>

            <p class="game-genre">Genre: Fight</p>

            <p class="game-players">Players: 2</p>

            <div class="gameplay-video">

                <video controls>

                    <source src="./assets/vid.mp4" type="video/mp4">

                    Your browser does not support the video tag.

                </video>

            </div>

        </div>

    </main>

    <nav class="secondary-nav">

        <ul>

            <li>

                <a href="../samuriaGame/index (2).html">PLAY</a>

            </li>

            <li><a href="../account/account.html">PROFILE</a></li>

            <li><a href="#">LOGOUT</a></li>

        </ul>

    </nav>

    <script src="samurai.js"></script>

</body>

</html>

## 5.12 account.js

// display date navbar

const dateDisplay = document.getElementById('date-display');

const today = new Date();

const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };

dateDisplay.innerText = today.toLocaleDateString(undefined, options);

// Function to handle image selection and update background

function changeImage1() {

    const input = document.getElementById('imageInput1'); // Assuming you have a file input with id 'imageInput1'

    // Trigger file input dialog

    input.click();

    input.onchange = async function(event) {

        const file = event.target.files[0];

        if (file) {

            const reader = new FileReader();

            // Once the file is loaded, set the preview in the `.cover` element

            reader.onload = function(e) {

                document.querySelector('.cover').style.backgroundImage = `url(${e.target.result})`;

            };

            // Read the selected file as a Data URL to preview it

            reader.readAsDataURL(file);

            // Prepare the file to be uploaded to the backend

            const formData = new FormData();

            formData.append('photo', file);

            // Get the username from localStorage (assuming you store it there)

            const username = localStorage.getItem('username');

            if (username) {

                formData.append('username', username); // Ensure the username is appended

            } else {

                console.error('Username not found in localStorage');

                return; // Stop if the username is not available

            }

            try {

                // Optionally, include a token for authentication if required by the backend

                const token = localStorage.getItem('token'); // Ensure this key matches where you store the token

                // Send the image and username to the server using fetch

                const response = await fetch('http://localhost:5000/api/auth/upload-photo', {

                    method: 'POST',

                    headers: {

                        'Authorization': token ? `Bearer ${token}` : '' // Include token if available

                    },

                    body: formData, // Send the form data (image and username)

                });

                const data = await response.json();

                if (response.ok) {

                    console.log('Image uploaded successfully:', data.message);

                } else {

                    console.error('Image upload failed:', data.message);

                }

            } catch (error) {

                console.error('Error uploading image:', error);

            }

        }

    };

}

// pfp image

function changeImage2() {

    const input = document.getElementById('imageInput2'); // Assuming you have a file input with id 'imageInput1'

    // Trigger file input dialog

    input.click();

    input.onchange = async function(event) {

        const file = event.target.files[0];

        if (file) {

            const reader = new FileReader();

            // Once the file is loaded, set the preview in the `.cover` element

            reader.onload = function(e) {

                document.querySelector('.pfp').style.backgroundImage = `url(${e.target.result})`;

            };

            // Read the selected file as a Data URL to preview it

            reader.readAsDataURL(file);

            // Prepare the file to be uploaded to the backend

            const formData = new FormData();

            formData.append('photop', file);

            // Get the username from localStorage (assuming you store it there)

            const username = localStorage.getItem('username');

            if (username) {

                formData.append('username', username); // Ensure the username is appended

            } else {

                console.error('Username not found in localStorage');

                return; // Stop if the username is not available

            }

            try {

                // Optionally, include a token for authentication if required by the backend

                const token = localStorage.getItem('token'); // Ensure this key matches where you store the token

                // Send the image and username to the server using fetch

                const response = await fetch('http://localhost:5000/api/auth/upload-pfp', {

                    method: 'POST',

                    headers: {

                        'Authorization': token ? `Bearer ${token}` : '' // Include token if available

                    },

                    body: formData, // Send the form data (image and username)

                });

                const data = await response.json();

                if (response.ok) {

                    console.log('Image uploaded successfully:', data.message);

                } else {

                    console.error('Image upload failed:', data.message);

                }

            } catch (error) {

                console.error('Error uploading image:', error);

            }

        }

    };

}

//send friend request

async function sendFriendRequest() {

    const yourToken = localStorage.getItem('token');

    const senderUsername = localStorage.getItem('username');

    const receiverUsername = document.getElementById('friendName').value; // Assuming 'friendName' is the ID of an input field

    try {

      const response = await fetch('http://localhost:5000/api/friend-requests/send', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

          'Authorization': `Bearer ${yourToken}` // Include this if authentication is required

        },

        body: JSON.stringify({ senderUsername, receiverUsername })

      });

      const result = await response.json();

      if (response.ok) {

        console.log(result.message); // Handle success

        alert(result.message); // Optionally show a success message

      } else {

        console.error(result.message); // Handle error

        alert(result.message); // Optionally show an error message

      }

    } catch (error) {

      console.error('Error sending friend request:', error);

      alert('An error occurred while sending the friend request.');

    }

}

// Function to load and display friend requests

async function loadFriendRequests() {

    const token = localStorage.getItem('token');

    const usen = localStorage.getItem('username'); // Retrieve username from local storage

    const apiUrl = `http://localhost:5000/api/friend-requests/requests/${usen}`; // Include username in the URL

    try {

        const response = await fetch(apiUrl, {

            method: 'GET',

            headers: {

                'Content-Type': 'application/json',

                'Authorization': `Bearer ${token}` // Include token in headers

            }

        });

        const requests = await response.json();

        const requestTableBody = document.getElementById('friend-request');

        requestTableBody.innerHTML = ''; // Clear existing content

        if (requests.length === 0) {

            requestTableBody.innerHTML = '<tr><td colspan="2">No friend requests found.</td></tr>';

        } else {

            requests.forEach((request, index) => {

                const row = document.createElement('tr');

                // You can use `request.senderUsername` directly instead of using localStorage

                const senderUsername = request.senderUsername; // Assuming `senderUsername` is populated with an object that has `username`

                const receiverUsername = usen; // Already fetched username from localStorage

                row.innerHTML = `

                    <td>${senderUsername}</td>

                    <td>

                        <button onclick="acceptFriendRequest('${senderUsername}','${receiverUsername}')">Accept</button>

                        <button onclick="rejectFriendRequest('${senderUsername}','${receiverUsername}')">Reject</button>

                    </td>

                `;

                requestTableBody.appendChild(row);

            });

        }

    } catch (error) {

        console.log('Error loading friend requests:', error);

    }

}

//accept req

async function acceptFriendRequest(senderUsername, receiverUsername) {

    const token = localStorage.getItem('token');

    const apiUrl = `http://localhost:5000/api/friend-requests/accept/${senderUsername}/${receiverUsername}`;

    try {

        const response = await fetch(apiUrl, {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

                'Authorization': `Bearer ${token}`

            }

        });

        const data = await response.json();

        if (response.ok) {

            console.log('Friend request accepted:', data.message);

            // Optionally, refresh the list of friend requests

            displayFriends();

            loadFriendRequests();

        } else {

            console.error('Error accepting friend request:', data.message);

        }

    } catch (error) {

        console.error('Error accepting friend request:', error);

    }

}

//reject req

async function rejectFriendRequest(senderUsername, receiverUsername) {

    const token = localStorage.getItem('token');

    const apiUrl = `http://localhost:5000/api/friend-requests/reject/${senderUsername}/${receiverUsername}`;

    try {

        const response = await fetch(apiUrl, {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

                'Authorization': `Bearer ${token}`

            }

        });

        const data = await response.json();

        if (response.ok) {

            console.log('Friend request rejected:', data.message);

            // Optionally, refresh the list of friend requests

            displayFriends();

        } else {

            console.error('Error rejecting friend request:', data.message);

        }

    } catch (error) {

        console.error('Error rejecting friend request:', error);

    }

}

/\*\*display friends  \*/

async function displayFriends() {

    const username = localStorage.getItem('username'); // Ensure username is fetched inside the function or globally

    const res = await fetch(`http://localhost:5000/api/friend-requests/show/${username}`);

    const friends = await res.json();

    console.log(friends);

    const friendsList = document.getElementById('friends-list');

    friendsList.innerHTML = ''; // Clear existing friends list

    friends.friends.forEach(friend => {

        const li = document.createElement('li');

        li.innerText = friend; // Assuming friend is a string or adjust if it’s an object

        li.addEventListener('click', () => fetchFriendGameData(friend)); // Add click event to fetch game data

        friendsList.appendChild(li);

    });

}

// profile display

async function displayProfile(username) {

    try {

        //HIGHSCORE dog GETTER

        const response = await fetch('http://localhost:5000/api/game/top-scores', {

          method: 'POST',

          headers: {

            'Content-Type': 'application/json',

          },

          body: JSON.stringify({ username, gameName: "dogGame" }),

        });

        if (!response.ok) { // Check if the response status is OK (status in the range 200-299)

            console.log('No response from server or response error:', response.statusText);

             // Exit early if response is not OK

            document.getElementById('high-score').innerText = `High Score: not their`;

            return;

        }

        const data = await response.json();

        console.log('Game score:', data.score); // Process the data as needed

        if (data.score !== undefined) {

            document.getElementById('high-score').innerText = `High Score: ${data.score}`;

        } else {

            console.error('Score not found in response data');

        }

        if (username) {

            document.getElementById('profile-name').innerText = `Name: ${username}`;

        } else {

            console.error('Username not found');

        }

        // goRilla score

        const responseG = await fetch('http://localhost:5000/api/game/top-scores', {

            method: 'POST',

            headers: {

              'Content-Type': 'application/json',

            },

            body: JSON.stringify({ username, gameName: "gorillaGame" }),

          });

          if (!responseG.ok) { // Check if the response status is OK (status in the range 200-299)

            console.log('No response from server or response error:', response.statusText);

             // Exit early if response is not OK

            document.getElementById('high-scoreG').innerText = `High Score: not their`;

            return;

        }

        const dataG = await responseG.json();

        console.log('Game scoreG:', dataG.score); // Process the data as needed

        if (dataG.score !== undefined) {

            document.getElementById('high-scoreG').innerText = `Wins: ${dataG.score}`;

        } else {

            console.error('GORILLA Score not found in response data');

        }

        ///end

        //EMAIL AND BIRTHDAY GETTER

        const response2 = await fetch('http://localhost:5000/api/auth/userdetails', {

            method: 'POST',

            headers: {

              'Content-Type': 'application/json',

            },

            body: JSON.stringify({ username}),

          });

          const data2 = await response2.json();

            if (data2.email !== undefined) {

                document.getElementById('email').innerText = `Email: ${data2.email}`;

            }else{

                console.error('Email not found in response data');

            }

            if(data2.birthday !== undefined){

                document.getElementById('birthday').innerText = `Birthday: ${data2.birthday}`;

            }else{

                console.error('Birthday not found in response data');

            }

            if(data2.profileImage !== undefined){

                document.querySelector('.profile .cover').style.backgroundImage = `url('http://localhost:5000/uploads/${data2.profileImage}')`;

            }

            if(data2.coverImage !== undefined){

                document.querySelector('.profile .pfp').style.backgroundImage = `url('http://localhost:5000/uploads/${data2.coverImage}')`;

            }

            console.log(data2)

    } catch (error) {

        console.error('There was a problem with the fetch operation:', error);

    }

}

const username2 = localStorage.getItem('username');

displayProfile(username2);

// logout

function logout(){

    localStorage.clear();

    window.location.href = '../login/login.html';

}

displayFriends();

loadFriendRequests();

async function fetchFriendGameData(friendUsername) {

    try {

        console.log(friendUsername);

        // Fetch data for the "dogGame"

        const response1 = await fetch('http://localhost:5000/api/game/top-scores', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify({ friendUsername, gameName: "dogGame" }),

        });

        if (response1) {

            console.log("first found")

        }else{

            console.log("first not found")

        }

        const data1 = await response1.json();

        console.log('Friend dogGame data:', data1.score);

        // Fetch data for the "gorillaGame"

        const response2 = await fetch('http://localhost:5000/api/game/top-scores', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify({ friendUsername, gameName: "gorillaGame" }),

        });

        if (response2) {

            console.log("second found")

        }else{

            console.log("second not found")

        }

        const data2 = await response2.json();

        console.log('Friend gorillaGame data:', data2);

        // Display data (optional, depending on your HTML structure)

        // const gameDataDisplay = document.getElementById('game-data-display');

        // gameDataDisplay.innerHTML = ''; // Clear existing data

        // const gameData = `

        //     <div><strong>dogGame:</strong> ${data1.score}</div>

        //     <div><strong>gorillaGame:</strong> ${data2.score}</div>

        // `;

        // gameDataDisplay.innerHTML = gameData;

    } catch (error) {

        console.error('Error fetching game data:', error);

    }

}

## 5.13 backgroung.js ( dog game )

class Layer{

    constructor(game, width, height, speedModifier, image){

        this.game = game;

        this.width = width;

        this.height = height;

        this.speedModifier = speedModifier;

        this.image = image;

        this.x = 0;

        this.y = 0;

    }

    update(){

        if (this.x < -this.width) this.x = 0;

        else this.x -= this.game.speed \* this.speedModifier;

    }

    draw(context){

        context.drawImage(this.image, this.x, this.y, this.width, this.height);

        context.drawImage(this.image, this.x + this.width, this.y, this.width, this.height);

    }

}

export class Background{

    constructor(game){

        this.game = game;

        this.width = 1667;

        this.height = 500;

        this.layer1image = document.getElementById('layer1');

        this.layer2image = document.getElementById('layer2');

        this.layer3image = document.getElementById('layer3');

        this.layer4image = document.getElementById('layer4');

        this.layer5image = document.getElementById('layer5');

        this.layer1 = new Layer(this.game, this.width, this.height, 0, this.layer1image);

        this.layer2 = new Layer(this.game, this.width, this.height, 0.2, this.layer2image);

        this.layer3 = new Layer(this.game, this.width, this.height, 0.4, this.layer3image);

        this.layer4 = new Layer(this.game, this.width, this.height, 0.8, this.layer4image);

        this.layer5 = new Layer(this.game, this.width, this.height, 1, this.layer5image);

        this.backgroungLayers = [this.layer1, this.layer2, this.layer3, this.layer4, this.layer5];

    }

    update(){

        this.backgroungLayers.forEach(layer => {

            layer.update();

        })

    }

    draw(context){

        this.backgroungLayers.forEach(layer => {

            layer.draw(context);

        })

    }

}

## 5.13 enemy.js

class Enemy{

    constructor(){

        this.frameX = 0;

        this.frameY = 0;

        this.fps = 20;

        this.frameInterval = 1000/this.fps;

        this.frameTimer = 0;

        this.markedForDeletion = false;

    }

    update(deltaTime){

        //movement

        this.x -= this.speedX + this.game.speed;

        this.y += this.speedY;

        if (this.frameTimer > this.frameInterval){

            this.frameTimer = 0;

            if (this.frameX < this.maxFrame) this.frameX++;

            else this.frameX = 0;

        }

        else{

            this.frameTimer += deltaTime;

        }

        //check if off screen

        if (this.x + this.width < 0) this.markedForDeletion = true;

    }

    draw(context){

        if (this.game.debug) context.strokeRect(this.x, this.y, this.width, this.height);

        context.drawImage(this.image,this.frameX \* this.width, 0 ,this.width, this.height, this.x, this.y, this.width, this.height);

    }

}

export class FlyingEnemy extends Enemy{

    constructor(game){

        super();

        this.game = game;

        this.width = 60;

        this.height = 44;

        this.x = this.game.width + Math.random() \* this.game.width \* 0.5;

        this.y = Math.random() \* this.game.height \* 0.5;

        this.speedX = Math.random() \* 1;

        this.speedY = 0;

        this.maxFrame = 5;

        this.image = document.getElementById('enemy\_fly');

        this.angle = 0;

        this.va = Math.random() \* 0.1 + 0.1

;    }

    update(deltaTime){

        super.update(deltaTime);

        this.angle += this.va;

        this.y += Math.sin(this.angle);

    }

}

export class GroundEnemy extends Enemy{

    constructor(game){

        super();

        this.game = game;

        this.width = 60;

        this.height = 87;

        this.x = this.game.width;

        this.y = this.game.height - this.height -  this.game.groundMargin;

        this.image = document.getElementById('enemy\_plant');

        this.speedX = 0;

        this.speedY = 0;

        this.maxFrame = 1;

        }

}

export class ClimbingEnemy extends Enemy{

    constructor(game){

        super();

        this.game = game;

        this.width = 120;

        this.height = 144;

        this.x = this.game.width;

        this.y = Math.random() \* this.game.height \* 0.5;

        this.image = document.getElementById('enemy\_spider');

        this.speedX = 0;

        this.speedY = Math.random() > 0.5 ? 1 : -1;

        this.maxFrame = 5;

        }

        update(deltaTime){

            super.update(deltaTime);

            if (this.y > this.game.height - this.height - this.game.groundMargin) this.speedY \*= -1;

            if (this.y < -this.height) this.markedForDeletion =true;

        }

        draw(context){

            super.draw(context);

            context.beginPath();

            context.moveTo(this.x + this.width/2, 0);

            context.lineTo(this.x + this.width/2, this.y + 50);

            context.stroke();

        }

}

## 5.14 input.js

export class InputHandler{

    constructor(game){

        this.game = game;

        this.keys = [];

        window.addEventListener('keydown', e => {

            if ((   e.key === 'ArrowDown' ||

                    e.key === 'ArrowUp' ||

                    e.key === 'ArrowLeft' ||

                    e.key === 'ArrowRight' ||

                    e.key === 'Enter'

            ) && this.keys.indexOf(e.key) === -1){

                this.keys.push(e.key);

            }

            else if (e.key === 'd') this.game.debug = !this.game.debug;

        });

        window.addEventListener('keyup', e => {

            if (e.key === 'ArrowDown' ||

                e.key === 'ArrowUp' ||

                e.key === 'ArrowLeft' ||

                e.key === 'ArrowRight' ||

                e.key === 'Enter'

            ){

                this.keys.splice(this.keys.indexOf(e.key), 1);

            }

        });

    }

}

## 5.15 main.js

import { Player } from './player.js';

import { InputHandler } from './input.js';

import { Background } from './backgroung.js';

import { FlyingEnemy, ClimbingEnemy, GroundEnemy } from './enemy.js';

import { UI } from './UI.js';

window.addEventListener('load', function () {

    const canvas = document.getElementById('canvas1');

    const ctx = canvas.getContext('2d');

    canvas.width = 1800;

    canvas.height = 500;

    class Game {

        constructor(width, height) {

            this.width = width;

            this.height = height;

            this.groundMargin = 80;

            this.speed = 0;

            this.maxSpeed = 3;

            this.background = new Background(this);

            this.player = new Player(this);

            this.input = new InputHandler(this);

            this.UI = new UI(this);

            this.enemies = [];

            this.particles = [];

            this.enemyTimer = 0;

            this.enemyInterval = 1000;

            this.debug = true;

            this.score = 0;

            this.fontColor = 'black';

            this.currentLevel = 1;

            this.requiredScore = 10; // Set a reasonable base score requirement

            this.gameFinished = false;

            this.isTransitioning = false; // Flag to prevent multiple transitions

            this.player.currentState = this.player.states[0];

            this.player.currentState.enter();

        }

        update(deltaTime) {

            if (this.gameFinished) return;

            this.background.update();

            this.player.update(this.input.keys, deltaTime);

            if (this.enemyTimer > this.enemyInterval) {

                this.addEnemy();

                this.enemyTimer = 0;

            } else {

                this.enemyTimer += deltaTime;

            }

            this.enemies.forEach(enemy => {

                enemy.update(deltaTime);

                if (enemy.markedForDeletion) this.enemies.splice(this.enemies.indexOf(enemy), 1);

            });

            this.particles.forEach((particle, index) => {

                particle.update();

                if (particle.markedForDeletion) this.particles.splice(index, 1);

            });

            if (this.score >= this.requiredScore) {

                this.finishLevel();

            }

        }

        draw(context) {

            this.background.draw(context);

            this.player.draw(context);

            this.enemies.forEach(enemy => {

                enemy.draw(context);

            });

            this.particles.forEach(particle => {

                particle.draw(context);

            });

            this.UI.draw(context);

            if (this.gameFinished) {

                context.fillStyle = 'rgba(0, 0, 0, 0.5)';

                context.fillRect(0, 0, this.width, this.height);

                context.fillStyle = 'white';

                context.font = '40px Helvetica';

                context.textAlign = 'center';

                context.fillText('Level Complete!', this.width / 2, this.height / 2 - 60);

                // Draw buttons

                context.font = '30px Helvetica';

                context.fillStyle = 'white';

                context.fillRect(this.width / 2 - 150, this.height / 2 + 50, 100, 50); // Home button

                context.fillRect(this.width / 2 + 50, this.height / 2 + 50, 100, 50); // Next Level button

                context.fillStyle = 'black';

                context.fillText('Home', this.width / 2 - 100, this.height / 2 + 85);

                context.fillText('Next', this.width / 2 + 100, this.height / 2 + 85);

            }

        }

        sendScoreToHtml() {

            // Dispatch a custom event with the score

            const event = new CustomEvent('levelFinished', { detail: { score: this.score } });

            window.dispatchEvent(event);

        }

        finishLevel() {

            if (!this.isTransitioning) {

                this.sendScoreToHtml(); // Send the score when the level is finished

                this.gameFinished = true;

                this.isTransitioning = true;

                window.addEventListener('click', this.handleButtonClick.bind(this));

            }

        }

        handleButtonClick(event) {

            const rect = canvas.getBoundingClientRect();

            const x = event.clientX - rect.left;

            const y = event.clientY - rect.top;

            // Check if click is within Home button

            if (x >= this.width / 2 - 150 && x <= this.width / 2 - 50 &&

                y >= this.height / 2 + 50 && y <= this.height / 2 + 100) {

                this.goHome();

            }

            // Check if click is within Next Level button

            else if (x >= this.width / 2 + 50 && x <= this.width / 2 + 150 &&

                y >= this.height / 2 + 50 && y <= this.height / 2 + 100) {

                this.startNextLevel();

            }

            // Remove event listener after handling

            window.removeEventListener('click', this.handleButtonClick.bind(this));

            this.isTransitioning = false; // Reset flag after handling

        }

        goHome() {

            // Implement navigation to home screen

            window.location.href = '../home/home.html'; // Change this to your home screen URL or implementation

        }

        startNextLevel() {

            if (this.isTransitioning) return; // Ensure only one transition

            this.currentLevel++;

            this.requiredScore = Math.round(this.requiredScore \* 1.2); // Increase required score by 20% for each level

            this.gameFinished = false;

            this.score = 0;

            this.enemies = [];

            this.particles = [];

            this.speed = 0; // Reset speed or adjust as necessary for the new level

        }

        addEnemy() {

            if (this.speed > 0) {

                if (Math.random() < 0.5) this.enemies.push(new GroundEnemy(this));

                else this.enemies.push(new ClimbingEnemy(this));

            }

            this.enemies.push(new FlyingEnemy(this));

        }

    }

    const game = new Game(canvas.width, canvas.height);

    let lastTime = 0;

    function animate(timeStamp) {

        const deltaTime = timeStamp - lastTime;

        lastTime = timeStamp;

        ctx.clearRect(0, 0, canvas.width, canvas.height);

        game.update(deltaTime);

        game.draw(ctx);

        requestAnimationFrame(animate);

    }

    animate(0);

});

## 5.16 particale.js

class Particle {  // Corrected class name capitalization

    constructor(game) {

        this.game = game;

        this.markedForDeletion = false;

    }

    update() {

        this.x -= this.speedX + this.game.speed;

        this.y -= this.speedY;

        this.size \*= 0.97;

        if (this.size < 0.5) this.markedForDeletion = true;

    }

}

export class Dust extends Particle {

    constructor(game, x, y) {

        super(game);

        this.size = Math.random() \* 10 + 10;

        this.x = x;

        this.y = y;

        this.speedX = Math.random();

        this.speedY = Math.random();

        this.color = 'black';

    }

    draw(context) {

        context.beginPath();

        context.arc(this.x, this.y, this.size, 0, Math.PI \* 2);

        context.fillStyle = this.color;

        context.fill();

    }

}

export class Splash extends Particle {

    // Splash implementation remains empty as per your initial code

}

export class Fire extends Particle {

    constructor(game, x, y) {

        super(game);

        this.image = document.getElementById('fire');

        this.size = Math.random() \* 50 + 50;

        this.x = x;

        this.y = y;

        this.speedX = 1;

        this.speedY = 1;

        this.angle = 0;

        this.va = Math.random() \* 0.2 + 0.1;

    }

    update(){

        super.update();

        this.angle += this.va;

        this.x += Math.sin(this.angle \* 5);

    }

    draw(context){

        context.save();

        context.translate(this.x, this.y);

        context.rotate(this.angle);

        context.drawImage(this.image, -this.size \* 0.5, -this.size \* 0.5, this.size, this.size);

        context.restore();

    }

}

## 5.17 player.js

import { Sitting, Running, Jumping, Falling, Rolling } from './playerStates.js';

export class Player{

    constructor(game){

        this.game = game;

        this.width = 100;

        this.height = 91.3;

        this.x = 0;

        this.y = this.game.height - this.height - this.game.groundMargin;

        this.vy = 0; //jump

        this.weight = 1;

        this.image = document.getElementById('player');

        this.frameX = 0;

        this.frameY = 0;

        this.maxFrame;

        this.fps = 20;

        this.frameInterval = 1000/this.fps;

        this.frameTimer = 0;

        this.speed = 0;

        this.maxspeed = 10;

        this.states = [ new Sitting(this.game), new Running(this.game), new Jumping(this.game), new Falling(this.game), new Rolling(this.game)];

    }

    update(input, deltaTime){

        this.checkCollision();

        this.currentState.handleInput(input);

        // horizontal movement

        this.x += this.speed;

        if (input.includes('ArrowRight')) this.speed = this.maxspeed;

        else if (input.includes('ArrowLeft')) this.speed = -this.maxspeed;

        else this.speed = 0;

        //vertical movement

        this.y += this.vy;

        if (!this.onGround()) this.vy += this.weight;

        else this.vy = 0;

        //sprite animation

        if (this.frameTimer > this.frameInterval){

            this.frameTimer = 0;

            if (this.frameX < this.maxFrame) this.frameX++;

            else this.frameX = 0;

        }

        else{

            this.frameTimer += deltaTime;

        }

        //restict game window

        if (this.x < 0) this.x = 0;

        if (this.x > this.game.width - this.width) this.x = this.game.width - this.width;

    }

    draw(context){

        if (this.game.debug) context.strokeRect(this.x, this.y, this.width, this.height);

        context.drawImage(this.image, this.frameX \* this.width, this.frameY \* this.height, this.width, this.height, this.x, this.y, this.width, this.height);

    }

    onGround(){

        return this.y >= this.game.height - this.height - this.game.groundMargin;

    }

    setState(state, speed){

        this.currentState = this.states[state];

        this.game.speed = this.game.maxSpeed \* speed;

        this.currentState.enter();

    }

    checkCollision(){

        this.game.enemies.forEach(enemy => {

            if(

                enemy.x < this.x + this.width &&

                enemy.x + enemy.width > this.x &&

                enemy.y < this.y +this.height &&

                enemy.y + enemy.height > this.y

            ){

                enemy.markedForDeletion = true;

                this.game.score++;

            }else{

            }

        })

    }

}

## 5.18 playerStates.js

import { Dust, Fire } from './particale.js';

const states = {

    SITTING: 0,

    RUNNING: 1,

    JUMPING: 2,

    FALLING: 3,

    ROLLING: 4,

    DIVING: 5,

    HIT: 6,

};

class State {

    constructor(state, game) {

        this.state = state;

        this.game = game;

    }

}

export class Sitting extends State {

    constructor(game) {

        super('SITTING', game);

    }

    enter() {

        this.game.player.frameX = 0;

        this.game.player.frameY = 5;

        this.game.player.maxFrame = 4;

    }

    handleInput(input) {

        if (input.includes('ArrowLeft') || input.includes('ArrowRight')) {

            this.game.player.setState(states.RUNNING, 1);

        } else if (input.includes('ArrowUp')) {

            this.game.player.setState(states.JUMPING, 1);

        } else if (input.includes('Enter')) {

            this.game.player.setState(states.ROLLING, 2);

        }

    }

}

export class Running extends State {

    constructor(game) {

        super('RUNNING', game);

    }

    enter() {

        this.game.player.frameX = 0;

        this.game.player.frameY = 3;

        this.game.player.maxFrame = 6;

    }

    handleInput(input) {

        this.game.particles.push(new Dust(this.game, this.game.player.x + this.game.player.width \* 0.5, this.game.player.y + this.game.player.height));

        if (input.includes('ArrowDown')) {

            this.game.player.setState(states.SITTING, 0);

        } else if (input.includes('ArrowUp')) {

            this.game.player.setState(states.JUMPING, 1);

        } else if (input.includes('Enter')) {

            this.game.player.setState(states.ROLLING, 2);

        }

    }

}

export class Jumping extends State {

    constructor(game) {

        super('JUMPING', game);

    }

    enter() {

        if (this.game.player.onGround()) this.game.player.vy -= 30;

        this.game.player.frameX = 0;

        this.game.player.maxFrame = 6;

        this.game.player.frameY = 1;

    }

    handleInput(input) {

        if (this.game.player.vy > this.game.player.weight) {

            this.game.player.setState(states.FALLING, 1);

        } else if (input.includes('Enter')) {

            this.game.player.setState(states.ROLLING, 2);

        }

    }

}

export class Falling extends State {

    constructor(game) {

        super('FALLING', game);

    }

    enter() {

        this.game.player.frameX = 0;

        this.game.player.maxFrame = 6;

        this.game.player.frameY = 2;

    }

    handleInput(input) {

        if (this.game.player.onGround()) {

            this.game.player.setState(states.RUNNING, 1);

        }

    }

}

export class Rolling extends State {

    constructor(game) {

        super('ROLLING', game);

    }

    enter() {

        this.game.player.frameX = 0;

        this.game.player.maxFrame = 6;

        this.game.player.frameY = 6;

    }

    handleInput(input) {

        this.game.particles.push(new Fire(this.game, this.game.player.x + this.game.player.width \* 0.5, this.game.player.y + this.game.player.height \* 0.5));

        if (!input.includes('Enter') && this.game.player.onGround()) {

            this.game.player.setState(states.RUNNING, 1);

        } else if (!input.includes('Enter') && !this.game.player.onGround()) {

            this.game.player.setState(states.FALLING, 1);

        } else if (input.includes('Enter') && input.includes('ArrowUp') && this.game.player.onGround()) {

            this.game.player.vy -= 27;

        }

    }

}

5.15 UI.js

export class UI {

    constructor(game) {

        this.game = game;

        this.fontSize = 30; // Default font size for other texts

        this.fontFamily = 'Helvetica';

    }

    draw(context) {

        // Draw the score

        context.font = `bold 40px ${this.fontFamily}`; // Increase font size and make it bold for emphasis

        context.textAlign = 'left';

        context.fillStyle = this.game.fontColor;

        context.fillText('Score: ' + this.game.score, 20, 50);

        // Draw the current level

        context.font = `bold 30px ${this.fontFamily}`; // Slightly smaller font for the level

        context.fillText('Level: ' + this.game.currentLevel, 20, 90);

    }

}

## 5.19 index.js ( gorilla game )

// The state of the game

let state = {};

let isDragging = false;

let dragStartX = undefined;

let dragStartY = undefined;

let previousAnimationTimestamp = undefined;

let animationFrameRequestID = undefined;

let delayTimeoutID = undefined;

let simulationMode = false;

let simulationImpact = {};

const darkModeMediaQuery = window.matchMedia("(prefers-color-scheme: dark)");

// Settings

const settings = {

  numberOfPlayers: 1, // 0 means two computers are playing against each other

  mode: darkModeMediaQuery.matches ? "dark" : "light",

};

const blastHoleRadius = 18;

// The main canvas element and its drawing context

const canvas = document.getElementById("game");

canvas.width = window.innerWidth \* window.devicePixelRatio;

canvas.height = window.innerHeight \* window.devicePixelRatio;

canvas.style.width = window.innerWidth + "px";

canvas.style.height = window.innerHeight + "px";

const ctx = canvas.getContext("2d");

// Windmill

const windmillDOM = document.getElementById("windmill");

const windmillHeadDOM = document.getElementById("windmill-head");

  bombGrabAreaDOM.style.left = `${left}px`;

  bombGrabAreaDOM.style.bottom = `${bottom}px`;

}

function initializeWindmillPosition() {

  // Move windmill into position

  const lastBuilding = state.buildings.at(-1);

  let rooftopY = lastBuilding.height \* state.scale;

  let rooftopX =

    (lastBuilding.x + lastBuilding.width / 2) \* state.scale + state.shift;

  windmillDOM.style.bottom = `${rooftopY}px`;

  windmillDOM.style.left = `${rooftopX - 100}px`;

  windmillDOM.style.scale = state.scale;

  windInfoDOM.style.bottom = `${rooftopY}px`;

  windInfoDOM.style.left = `${rooftopX - 50}px`;

}

function draw() {

  ctx.save();

  ctx.scale(window.devicePixelRatio, window.devicePixelRatio);

  drawBackgroundSky();

  // Flip coordinate system upside down

  ctx.translate(0, window.innerHeight);

  ctx.scale(1, -1);

  // Scale and shift view to center

  ctx.translate(state.shift, 0);

  ctx.scale(state.scale, state.scale);

  // Draw scene

  drawBackgroundMoon();

  drawBackgroundBuildings();

  drawBuildingsWithBlastHoles();

  drawGorilla(1);

  drawGorilla(2);

  drawBomb();

  // Eyes

  ctx.fillStyle = "black";

  ctx.beginPath();

  ctx.arc(-3.5, 70, 1.4, 0, 2 \* Math.PI);

  ctx.moveTo(+3.5, 70);

  ctx.arc(+3.5, 70, 1.4, 0, 2 \* Math.PI);

  ctx.fill();

  ctx.strokeStyle = "black";

  ctx.lineWidth = 1.4;

  // Nose

  ctx.beginPath();

  ctx.moveTo(-3.5, 66.5);

  ctx.lineTo(-1.5, 65);

  ctx.moveTo(3.5, 66.5);

  ctx.lineTo(1.5, 65);

  ctx.stroke();

  // Mouth

  ctx.beginPath();

  if (state.phase === "celebrating" && state.currentPlayer === player) {

    ctx.moveTo(-5, 60);

    ctx.quadraticCurveTo(0, 56, 5, 60);

  } else {

    ctx.moveTo(-5, 56);

    ctx.quadraticCurveTo(0, 60, 5, 56);

  }

  ctx.stroke();

}

function drawGorillaThoughtBubbles(player) {

  if (state.phase === "aiming") {

    const currentPlayerIsComputer =

      (settings.numberOfPlayers === 0 &&

        state.currentPlayer === 1 &&

        player === 1) ||

      (settings.numberOfPlayers !== 2 &&

        state.currentPlayer === 2 &&

        player === 2);

    if (currentPlayerIsComputer) {

      ctx.save();

      ctx.scale(1, -1);

      ctx.font = "20px sans-serif";

      ctx.textAlign = "center";

      ctx.fillText("?", 0, -90);

      ctx.font = "10px sans-serif";

      ctx.rotate((5 / 180) \* Math.PI);

      ctx.fillText("?", 0, -90);

      ctx.rotate((-10 / 180) \* Math.PI);

      ctx.fillText("?", 0, -90);

      ctx.restore();

    }

  }

}

function drawBomb() {

  ctx.save();

  ctx.translate(state.bomb.x, state.bomb.y);

  if (state.phase === "aiming") {

    // Move the bomb with the mouse while aiming

    ctx.translate(-state.bomb.velocity.x / 6.25, -state.bomb.velocity.y / 6.25);

    // Draw throwing trajectory

    ctx.strokeStyle = "rgba(255, 255, 255, 0.7)";

    ctx.setLineDash([3, 8]);

    ctx.lineWidth = 3;

    ctx.beginPath();

    ctx.moveTo(0, 0);

    ctx.lineTo(state.bomb.velocity.x, state.bomb.velocity.y);

    ctx.stroke();

    // Draw circle

    ctx.fillStyle = "white";

    ctx.beginPath();

    ctx.arc(0, 0, 6, 0, 2 \* Math.PI);

    ctx.fill();

  } else if (state.phase === "in flight") {

    // Draw rotated banana

    ctx.fillStyle = "white";

    ctx.rotate(state.bomb.rotation);

    ctx.beginPath();

    ctx.moveTo(-8, -2);

    ctx.quadraticCurveTo(0, 12, 8, -2);

    ctx.quadraticCurveTo(0, 2, -8, -2);

    ctx.fill();

  } else {

    // Draw circle

    ctx.fillStyle = "white";

    ctx.beginPath();

    ctx.arc(0, 0, 6, 0, 2 \* Math.PI);

    ctx.fill();

  }

  // Restore transformation

  ctx.restore();

  // Indicator showing if the bomb is above the screen

  if (state.bomb.y > window.innerHeight / state.scale) {

    ctx.beginPath();

    ctx.strokeStyle = "white";

    const distance = state.bomb.y - window.innerHeight / state.scale;

    ctx.moveTo(state.bomb.x, window.innerHeight / state.scale - 10);

    ctx.lineTo(state.bomb.x, window.innerHeight / state.scale - distance);

    ctx.moveTo(state.bomb.x, window.innerHeight / state.scale - 10);

    ctx.lineTo(state.bomb.x - 5, window.innerHeight / state.scale - 15);

    ctx.moveTo(state.bomb.x, window.innerHeight / state.scale - 10);

    ctx.lineTo(state.bomb.x + 5, window.innerHeight / state.scale - 15);

    ctx.stroke();

  }

  // Indicator showing the starting position of the bomb

  if (state.bomb.highlight) {

    ctx.beginPath();

    ctx.strokeStyle = "white";

    ctx.lineWidth = 2;

    ctx.moveTo(state.bomb.x, state.bomb.y + 20);

    ctx.lineTo(state.bomb.x, state.bomb.y + 120);

    ctx.moveTo(state.bomb.x, state.bomb.y + 20);

    ctx.lineTo(state.bomb.x - 5, state.bomb.y + 25);

    ctx.moveTo(state.bomb.x, state.bomb.y + 20);

    ctx.lineTo(state.bomb.x + 5, state.bomb.y + 25);

    ctx.stroke();

  }

}

// Event handlers

bombGrabAreaDOM.addEventListener("mousedown", function (e) {

  hideInstructions();

  if (state.phase === "aiming") {

    isDragging = true;

    dragStartX = e.clientX;

    dragStartY = e.clientY;

    document.body.style.cursor = "grabbing";

  }

});

window.addEventListener("mousemove", function (e) {

  if (isDragging) {

    let deltaX = e.clientX - dragStartX;

    let deltaY = e.clientY - dragStartY;

    state.bomb.velocity.x = -deltaX;

    state.bomb.velocity.y = deltaY;

    setInfo(deltaX, deltaY);

    draw();

  }

});

// Set values on the info panel

function setInfo(deltaX, deltaY) {

  const hypotenuse = Math.sqrt(deltaX \*\* 2 + deltaY \*\* 2);

  const angleInRadians = Math.asin(deltaY / hypotenuse);

  const angleInDegrees = (angleInRadians / Math.PI) \* 180;

  if (state.currentPlayer === 1) {

    angle1DOM.innerText = Math.round(angleInDegrees);

    velocity1DOM.innerText = Math.round(hypotenuse);

  } else {

    angle2DOM.innerText = Math.round(angleInDegrees);

    velocity2DOM.innerText = Math.round(hypotenuse);

  }

}

window.addEventListener("mouseup", function () {

  if (isDragging) {

    isDragging = false;

    document.body.style.cursor = "default";

    throwBomb();

  }

});

function computerThrow() {

  const numberOfSimulations = 2 + state.round \* 3;

  const bestThrow = runSimulations(numberOfSimulations);

  initializeBombPosition();

  state.bomb.velocity.x = bestThrow.velocityX;

  state.bomb.velocity.y = bestThrow.velocityY;

  setInfo(bestThrow.velocityX, bestThrow.velocityY);

  // Draw the aiming gorilla

  draw();

  // Make it look like the computer is thinking for a second

  delayTimeoutID = setTimeout(throwBomb, 1000);

}

// Simulate multiple throws and pick the best

function runSimulations(numberOfSimulations) {

  let bestThrow = {

    velocityX: undefined,

    velocityY: undefined,

    distance: Infinity,

  };

  simulationMode = true;

  // Calculating the center position of the enemy

  const enemyBuilding =

    state.currentPlayer === 1

      ? state.buildings.at(-2) // Second last building

      : state.buildings.at(1); // Second building

  const enemyX = enemyBuilding.x + enemyBuilding.width / 2;

  const enemyY = enemyBuilding.height + 30;

  for (let i = 0; i < numberOfSimulations; i++) {

    // Pick a random angle and velocity

    const angleInDegrees = -10 + Math.random() \* 100;

    const angleInRadians = (angleInDegrees / 180) \* Math.PI;

    const velocity = 40 + Math.random() \* 130;

    // Calculate the horizontal and vertical velocity

    const direction = state.currentPlayer === 1 ? 1 : -1;

    const velocityX = Math.cos(angleInRadians) \* velocity \* direction;

    const velocityY = Math.sin(angleInRadians) \* velocity;

    initializeBombPosition();

    state.bomb.velocity.x = velocityX;

    state.bomb.velocity.y = velocityY;

    throwBomb();

    // Calculating the distance between the simulated impact and the enemy

    const distance = Math.sqrt(

      (enemyX - simulationImpact.x) \*\* 2 + (enemyY - simulationImpact.y) \*\* 2

    );

    // If the current impact is closer to the enemy

    // than any of the previous simulations then pick this one

    if (distance < bestThrow.distance) {

      bestThrow = { velocityX, velocityY, distance };

    }

  }

  simulationMode = false;

  return bestThrow;

}

function throwBomb() {

  if (simulationMode) {

    previousAnimationTimestamp = 0;

    animate(16);

  } else {

    state.phase = "in flight";

    previousAnimationTimestamp = undefined;

    animationFrameRequestID = requestAnimationFrame(animate);

  }

}

function animate(timestamp) {

  if (previousAnimationTimestamp === undefined) {

    previousAnimationTimestamp = timestamp;

    animationFrameRequestID = requestAnimationFrame(animate);

    return;

  }

  const elapsedTime = timestamp - previousAnimationTimestamp;

  // We break down every animation cycle into 10 tiny movements for greater hit detection precision

  const hitDetectionPrecision = 10;

  for (let i = 0; i < hitDetectionPrecision; i++) {

    moveBomb(elapsedTime / hitDetectionPrecision);

    // Hit detection

    const miss = checkFrameHit() || checkBuildingHit(); // Bomb got off-screen or hit a building

    const hit = checkGorillaHit(); // Bomb hit the enemy

    if (simulationMode && (hit || miss)) {

      simulationImpact = { x: state.bomb.x, y: state.bomb.y };

      return; // Simulation ended, return from the loop

    }

    // Handle the case when we hit a building or the bomb got off-screen

    if (miss) {

      state.currentPlayer = state.currentPlayer === 1 ? 2 : 1; // Switch players

      if (state.currentPlayer === 1) state.round++;

      state.phase = "aiming";

      initializeBombPosition();

      draw();

      const computerThrowsNext =

        settings.numberOfPlayers === 0 ||

        (settings.numberOfPlayers === 1 && state.currentPlayer === 2);

      if (computerThrowsNext) setTimeout(computerThrow, 50);

      return;

    }

    // Handle the case when we hit the enemy

    if (hit) {

      state.phase = "celebrating";

      announceWinner();

      draw();

      return;

    }

  }

  if (!simulationMode) draw();

  // Continue the animation loop

  previousAnimationTimestamp = timestamp;

  if (simulationMode) {

    animate(timestamp + 16);

  } else {

    animationFrameRequestID = requestAnimationFrame(animate);

  }

}

function moveBomb(elapsedTime) {

  const multiplier = elapsedTime / 200;

  // Adjust trajectory by wind

  state.bomb.velocity.x += state.windSpeed \* multiplier;

  // Adjust trajectory by gravity

  state.bomb.velocity.y -= 20 \* multiplier;

  // Calculate new position

  state.bomb.x += state.bomb.velocity.x \* multiplier;

  state.bomb.y += state.bomb.velocity.y \* multiplier;

  // Rotate according to the direction

  const direction = state.currentPlayer === 1 ? -1 : +1;

  state.bomb.rotation += direction \* 5 \* multiplier;

}

function checkFrameHit() {

  // Stop throw animation once the bomb gets out of the left, bottom, or right edge of the screen

  if (

    state.bomb.y < 0 ||

    state.bomb.x < -state.shift / state.scale ||

    state.bomb.x > (window.innerWidth - state.shift) / state.scale

  ) {

    return true; // The bomb is off-screen

  }

}

function checkBuildingHit() {

  for (let i = 0; i < state.buildings.length; i++) {

    const building = state.buildings[i];

    if (

      state.bomb.x + 4 > building.x &&

      state.bomb.x - 4 < building.x + building.width &&

      state.bomb.y - 4 < 0 + building.height

    ) {

      // Check if the bomb is inside the blast hole of a previous impact

      for (let j = 0; j < state.blastHoles.length; j++) {

        const blastHole = state.blastHoles[j];

        // Check how far the bomb is from the center of a previous blast hole

        const horizontalDistance = state.bomb.x - blastHole.x;

        const verticalDistance = state.bomb.y - blastHole.y;

        const distance = Math.sqrt(

          horizontalDistance \*\* 2 + verticalDistance \*\* 2

        );

        if (distance < blastHoleRadius) {

          // The bomb is inside of the rectangle of a building,

          // but a previous bomb already blew off this part of the building

          return false;

        }

      }

      if (!simulationMode) {

        state.blastHoles.push({ x: state.bomb.x, y: state.bomb.y });

      }

      return true; // Building hit

    }

  }

}

function checkGorillaHit() {

  const enemyPlayer = state.currentPlayer === 1 ? 2 : 1;

  const enemyBuilding =

    enemyPlayer === 1

      ? state.buildings.at(1) // Second building

      : state.buildings.at(-2); // Second last building

  ctx.save();

  ctx.translate(

    enemyBuilding.x + enemyBuilding.width / 2,

    enemyBuilding.height

  );

  drawGorillaBody();

  let hit = ctx.isPointInPath(state.bomb.x, state.bomb.y);

  drawGorillaLeftArm(enemyPlayer);

  hit ||= ctx.isPointInStroke(state.bomb.x, state.bomb.y);

  drawGorillaRightArm(enemyPlayer);

  hit ||= ctx.isPointInStroke(state.bomb.x, state.bomb.y);

  ctx.restore();

  return hit;

}

function announceWinner() {

  if (settings.numberOfPlayers === 0) {

    winnerDOM.innerText = `Computer ${state.currentPlayer}`;

  } else if (settings.numberOfPlayers === 1 && state.currentPlayer === 1) {

    winnerDOM.innerText = `You`;

  } else if (settings.numberOfPlayers === 1 && state.currentPlayer === 2) {

    winnerDOM.innerText = `Computer`;

  } else {

    winnerDOM.innerText = `Player ${state.currentPlayer}`;

  }

  showCongratulations();

}

singlePlayerButtonDOM.forEach((button) =>

  button.addEventListener("click", () => {

    settings.numberOfPlayers = 1;

    gameModeDOM.innerHTML = "Player vs. Computer";

    name1DOM.innerText = "Player";

    name2DOM.innerText = "Computer";

    newGame();

  })

);

twoPlayersButtonDOM.forEach((button) =>

  button.addEventListener("click", () => {

    settings.numberOfPlayers = 2;

    gameModeDOM.innerHTML = "Player vs. Player";

    name1DOM.innerText = "Player 1";

    name2DOM.innerText = "Player 2";

    newGame();

  })

);

autoPlayButtonDOM.forEach((button) =>

  button.addEventListener("click", () => {

    settings.numberOfPlayers = 0;

    name1DOM.innerText = "Computer 1";

    name2DOM.innerText = "Computer 2";

    newGame();

  })

);

function generateWindSpeed() {

  // Generate a random number between -10 and +10

  return -10 + Math.random() \* 20;

}

function setWindMillRotation() {

  const rotationSpeed = Math.abs(50 / state.windSpeed);

  windmillHeadDOM.style.animationDirection =

    state.windSpeed > 0 ? "normal" : "reverse";

  windmillHeadDOM.style.animationDuration = `${rotationSpeed}s`;

  windSpeedDOM.innerText = Math.round(state.windSpeed);

}

window.addEventListener("mousemove", function (e) {

  settingsDOM.style.opacity = 1;

  info1DOM.style.opacity = 1;

  info2DOM.style.opacity = 1;

});

const enterFullscreen = document.getElementById("enter-fullscreen");

const exitFullscreen = document.getElementById("exit-fullscreen");

function toggleFullscreen() {

  if (!document.fullscreenElement) {

    document.documentElement.requestFullscreen();

    enterFullscreen.setAttribute("stroke", "transparent");

    exitFullscreen.setAttribute("stroke", "white");

  } else {

    document.exitFullscreen();

    enterFullscreen.setAttribute("stroke", "white");

    exitFullscreen.setAttribute("stroke", "transparent");

  }

}

## 5.20 index.js ( samurai game )

const canvas = document.querySelector('canvas')

const c = canvas.getContext('2d')

canvas.width = 1024

canvas.height = 576

c.fillRect(0, 0, canvas.width, canvas.height)

const gravity = 0.7

const background = new Sprite({

  position: {

    x: 0,

    y: 0

  },

  imageSrc: './img/background.png'

})

const shop = new Sprite({

  position: {

    x: 600,

    y: 128

  },

  imageSrc: './img/shop.png',

  scale: 2.75,

  framesMax: 6

})

const player = new Fighter({

  position: {

    x: 0,

    y: 0

  },

  velocity: {

    x: 0,

    y: 0

  },

  offset: {

    x: 0,

    y: 0

  },

  imageSrc: './img/samuraiMack/Idle.png',

  framesMax: 8,

  scale: 2.5,

  offset: {

    x: 215,

    y: 157

  },

  sprites: {

    idle: {

      imageSrc: './img/samuraiMack/Idle.png',

      framesMax: 8

    },

    run: {

      imageSrc: './img/samuraiMack/Run.png',

      framesMax: 8

    },

    jump: {

      imageSrc: './img/samuraiMack/Jump.png',

      framesMax: 2

    },

    fall: {

      imageSrc: './img/samuraiMack/Fall.png',

      framesMax: 2

    },

    attack1: {

      imageSrc: './img/samuraiMack/Attack1.png',

      framesMax: 6

    },

    takeHit: {

      imageSrc: './img/samuraiMack/Take Hit - white silhouette.png',

      framesMax: 4

    },

    death: {

      imageSrc: './img/samuraiMack/Death.png',

      framesMax: 6

    }

  },

  attackBox: {

    offset: {

      x: 100,

      y: 50

    },

    width: 160,

    height: 50

  }

})

const enemy = new Fighter({

  position: {

    x: 400,

    y: 100

  },

  velocity: {

    x: 0,

    y: 0

  },

  color: 'blue',

  offset: {

    x: -50,

    y: 0

  },

  imageSrc: './img/kenji/Idle.png',

  framesMax: 4,

  scale: 2.5,

  offset: {

    x: 215,

    y: 167

  },

  sprites: {

    idle: {

      imageSrc: './img/kenji/Idle.png',

      framesMax: 4

    },

    run: {

      imageSrc: './img/kenji/Run.png',

      framesMax: 8

    },

    jump: {

      imageSrc: './img/kenji/Jump.png',

      framesMax: 2

    },

    fall: {

      imageSrc: './img/kenji/Fall.png',

      framesMax: 2

    },

    attack1: {

      imageSrc: './img/kenji/Attack1.png',

      framesMax: 4

    },

    takeHit: {

      imageSrc: './img/kenji/Take hit.png',

      framesMax: 3

    },

    death: {

      imageSrc: './img/kenji/Death.png',

      framesMax: 7

    }

  },

  attackBox: {

    offset: {

      x: -170,

      y: 50

    },

    width: 170,

    height: 50

  }

})

console.log(player)

const keys = {

  a: {

    pressed: false

  },

  d: {

    pressed: false

  },

  ArrowRight: {

    pressed: false

  },

  ArrowLeft: {

    pressed: false

  }

}

decreaseTimer()

function animate() {

  window.requestAnimationFrame(animate)

  c.fillStyle = 'black'

  c.fillRect(0, 0, canvas.width, canvas.height)

  background.update()

  shop.update()

  c.fillStyle = 'rgba(255, 255, 255, 0.15)'

  c.fillRect(0, 0, canvas.width, canvas.height)

  player.update()

  enemy.update()

  player.velocity.x = 0

  enemy.velocity.x = 0

  // player movement

  if (keys.a.pressed && player.lastKey === 'a') {

    player.velocity.x = -5

    player.switchSprite('run')

  } else if (keys.d.pressed && player.lastKey === 'd') {

    player.velocity.x = 5

    player.switchSprite('run')

  } else {

    player.switchSprite('idle')

  }

  // jumping

  if (player.velocity.y < 0) {

    player.switchSprite('jump')

  } else if (player.velocity.y > 0) {

    player.switchSprite('fall')

  }

  // Enemy movement

  if (keys.ArrowLeft.pressed && enemy.lastKey === 'ArrowLeft') {

    enemy.velocity.x = -5

    enemy.switchSprite('run')

  } else if (keys.ArrowRight.pressed && enemy.lastKey === 'ArrowRight') {

    enemy.velocity.x = 5

    enemy.switchSprite('run')

  } else {

    enemy.switchSprite('idle')

  }

  // jumping

  if (enemy.velocity.y < 0) {

    enemy.switchSprite('jump')

  } else if (enemy.velocity.y > 0) {

    enemy.switchSprite('fall')

  }

  // detect for collision & enemy gets hit

  if (

    rectangularCollision({

      rectangle1: player,

      rectangle2: enemy

    }) &&

    player.isAttacking &&

    player.framesCurrent === 4

  ) {

    enemy.takeHit()

    player.isAttacking = false

    gsap.to('#enemyHealth', {

      width: enemy.health + '%'

    })

  }

  // if player misses

  if (player.isAttacking && player.framesCurrent === 4) {

    player.isAttacking = false

  }

  // this is where our player gets hit

  if (

    rectangularCollision({

      rectangle1: enemy,

      rectangle2: player

    }) &&

    enemy.isAttacking &&

    enemy.framesCurrent === 2

  ) {

    player.takeHit()

    enemy.isAttacking = false

    gsap.to('#playerHealth', {

      width: player.health + '%'

    })

  }

  // if player misses

  if (enemy.isAttacking && enemy.framesCurrent === 2) {

    enemy.isAttacking = false

  }

  // end game based on health

  if (enemy.health <= 0 || player.health <= 0) {

    determineWinner({ player, enemy, timerId })

  }

}

animate()

window.addEventListener('keydown', (event) => {

  if (!player.dead) {

    switch (event.key) {

      case 'd':

        keys.d.pressed = true

        player.lastKey = 'd'

        break

      case 'a':

        keys.a.pressed = true

        player.lastKey = 'a'

        break

      case 'w':

        player.velocity.y = -20

        break

      case ' ':

        player.attack()

        break

    }

  }

  if (!enemy.dead) {

    switch (event.key) {

      case 'ArrowRight':

        keys.ArrowRight.pressed = true

        enemy.lastKey = 'ArrowRight'

        break

      case 'ArrowLeft':

        keys.ArrowLeft.pressed = true

        enemy.lastKey = 'ArrowLeft'

        break

      case 'ArrowUp':

        enemy.velocity.y = -20

        break

      case 'ArrowDown':

        enemy.attack()

        break

    }

  }

})

window.addEventListener('keyup', (event) => {

  switch (event.key) {

    case 'd':

      keys.d.pressed = false

      break

    case 'a':

      keys.a.pressed = false

      break

  }

  // enemy keys

  switch (event.key) {

    case 'ArrowRight':

      keys.ArrowRight.pressed = false

      break

    case 'ArrowLeft':

      keys.ArrowLeft.pressed = false

      break

  }

})

5.17 classes.js

class Sprite {

  constructor({

    position,

    imageSrc,

    scale = 1,

    framesMax = 1,

    offset = { x: 0, y: 0 }

  }) {

    this.position = position

    this.width = 50

    this.height = 150

    this.image = new Image()

    this.image.src = imageSrc

    this.scale = scale

    this.framesMax = framesMax

    this.framesCurrent = 0

    this.framesElapsed = 0

    this.framesHold = 5

    this.offset = offset

  }

  draw() {

    c.drawImage(

      this.image,

      this.framesCurrent \* (this.image.width / this.framesMax),

      0,

      this.image.width / this.framesMax,

      this.image.height,

      this.position.x - this.offset.x,

      this.position.y - this.offset.y,

      (this.image.width / this.framesMax) \* this.scale,

      this.image.height \* this.scale

    )

  }

  animateFrames() {

    this.framesElapsed++

    if (this.framesElapsed % this.framesHold === 0) {

      if (this.framesCurrent < this.framesMax - 1) {

        this.framesCurrent++

      } else {

        this.framesCurrent = 0

      }

    }

  }

  update() {

    this.draw()

    this.animateFrames()

  }

}

class Fighter extends Sprite {

  constructor({

    position,

    velocity,

    color = 'red',

    imageSrc,

    scale = 1,

    framesMax = 1,

    offset = { x: 0, y: 0 },

    sprites,

    attackBox = { offset: {}, width: undefined, height: undefined }

  }) {

    super({

      position,

      imageSrc,

      scale,

      framesMax,

      offset

    })

    this.velocity = velocity

    this.width = 50

    this.height = 150

    this.lastKey

    this.attackBox = {

      position: {

        x: this.position.x,

        y: this.position.y

      },

      offset: attackBox.offset,

      width: attackBox.width,

      height: attackBox.height

    }

    this.color = color

    this.isAttacking

    this.health = 100

    this.framesCurrent = 0

    this.framesElapsed = 0

    this.framesHold = 5

    this.sprites = sprites

    this.dead = false

    for (const sprite in this.sprites) {

      sprites[sprite].image = new Image()

      sprites[sprite].image.src = sprites[sprite].imageSrc

    }

  }

  update() {

    this.draw()

    if (!this.dead) this.animateFrames()

    // attack boxes

    this.attackBox.position.x = this.position.x + this.attackBox.offset.x

    this.attackBox.position.y = this.position.y + this.attackBox.offset.y

    // draw the attack box

    // c.fillRect(

    //   this.attackBox.position.x,

    //   this.attackBox.position.y,

    //   this.attackBox.width,

    //   this.attackBox.height

    // )

    this.position.x += this.velocity.x

    this.position.y += this.velocity.y

    // gravity function

    if (this.position.y + this.height + this.velocity.y >= canvas.height - 96) {

      this.velocity.y = 0

      this.position.y = 330

    } else this.velocity.y += gravity

  }

  attack() {

    this.switchSprite('attack1')

    this.isAttacking = true

  }

  takeHit() {

    this.health -= 20

    if (this.health <= 0) {

      this.switchSprite('death')

    } else this.switchSprite('takeHit')

  }

  switchSprite(sprite) {

    if (this.image === this.sprites.death.image) {

      if (this.framesCurrent === this.sprites.death.framesMax - 1)

        this.dead = true

      return

    }

    // overriding all other animations with the attack animation

    if (

      this.image === this.sprites.attack1.image &&

      this.framesCurrent < this.sprites.attack1.framesMax - 1

    )

      return

    // override when fighter gets hit

    if (

      this.image === this.sprites.takeHit.image &&

      this.framesCurrent < this.sprites.takeHit.framesMax - 1

    )

      return

    switch (sprite) {

      case 'idle':

        if (this.image !== this.sprites.idle.image) {

          this.image = this.sprites.idle.image

          this.framesMax = this.sprites.idle.framesMax

          this.framesCurrent = 0

        }

        break

      case 'run':

        if (this.image !== this.sprites.run.image) {

          this.image = this.sprites.run.image

          this.framesMax = this.sprites.run.framesMax

          this.framesCurrent = 0

        }

        break

      case 'jump':

        if (this.image !== this.sprites.jump.image) {

          this.image = this.sprites.jump.image

          this.framesMax = this.sprites.jump.framesMax

          this.framesCurrent = 0

        }

        break

      case 'fall':

        if (this.image !== this.sprites.fall.image) {

          this.image = this.sprites.fall.image

          this.framesMax = this.sprites.fall.framesMax

          this.framesCurrent = 0

        }

        break

      case 'attack1':

        if (this.image !== this.sprites.attack1.image) {

          this.image = this.sprites.attack1.image

          this.framesMax = this.sprites.attack1.framesMax

          this.framesCurrent = 0

        }

        break

      case 'takeHit':

        if (this.image !== this.sprites.takeHit.image) {

          this.image = this.sprites.takeHit.image

          this.framesMax = this.sprites.takeHit.framesMax

          this.framesCurrent = 0

        }

        break

      case 'death':

        if (this.image !== this.sprites.death.image) {

          this.image = this.sprites.death.image

          this.framesMax = this.sprites.death.framesMax

          this.framesCurrent = 0

        }

        break

    }

  }

}

## 5.21 utils.js

function rectangularCollision({ rectangle1, rectangle2 }) {

  return (

    rectangle1.attackBox.position.x + rectangle1.attackBox.width >=

      rectangle2.position.x &&

    rectangle1.attackBox.position.x <=

      rectangle2.position.x + rectangle2.width &&

    rectangle1.attackBox.position.y + rectangle1.attackBox.height >=

      rectangle2.position.y &&

    rectangle1.attackBox.position.y <= rectangle2.position.y + rectangle2.height

  )

}

function determineWinner({ player, enemy, timerId }) {

  clearTimeout(timerId)

  document.querySelector('#displayText').style.display = 'flex'

  if (player.health === enemy.health) {

    document.querySelector('#displayText').innerHTML = 'Tie'

  } else if (player.health > enemy.health) {

    document.querySelector('#displayText').innerHTML = 'Player 1 Wins'

  } else if (player.health < enemy.health) {

    document.querySelector('#displayText').innerHTML = 'Player 2 Wins'

  }

}

let timer = 60

let timerId

function decreaseTimer() {

  if (timer > 0) {

    timerId = setTimeout(decreaseTimer, 1000)

    timer--

    document.querySelector('#timer').innerHTML = timer

  }

  if (timer === 0) {

    determineWinner({ player, enemy, timerId })

  }

}

## 5.22 about.js , gorrilaGameInfo.js, dogGameinfo.js, samuraiGameinfo.js

const dateDisplay = document.getElementById('date-display');

const today = new Date();

const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };

dateDisplay.textContent = today.toLocaleDateString(undefined, options);

5.17 home.js

document.addEventListener('DOMContentLoaded', function() {

    const dateDisplay = document.getElementById('date-display');

    const today = new Date();

    const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };

    dateDisplay.textContent = today.toLocaleDateString(undefined, options);

//sliding image

const imageArray = [

    './assets/speedypup.jpg', // Replace with your image URLs

    './assets/gorilla.jpg',

    './assets/speedypup3.jpg',

    './assets/gorilla2.jpg',

    './assets/bg.jpg',

];

let currentImageIndex = 0;

// Get the image element in the featured section

const featuredImage = document.querySelector('.featuredimage img');

// Function to change the image

function changeImage() {

    // Update the image source to the next image in the array

    featuredImage.src = imageArray[currentImageIndex];

    // Increment the index or reset if we've reached the end

    currentImageIndex = (currentImageIndex + 1) % imageArray.length;

}

// Call changeImage every 3 seconds (3000 milliseconds)

setInterval(changeImage, 3000);

// Optional: Call it initially to set the first image

changeImage();

});

function logout(){

    localStorage.clear();

    window.location.href = '../login/login.html';

}

function gorillapage(){

    window.location.href = '../gotillaGame/index2.html';

}

## 5.23 allowedOrigins.js

// allowedOrigins.js

const allowedOrigins = [

    '[http://127.0.0.1:5500](http://127.0.0.1:5500/)', // Your front-end origin

    '[http://localhost:5500](http://localhost:5500/)' // Another front-end origin

];

module.exports = allowedOrigins;

## 5.24 corsOptions.js

// corsOptions.js

const allowedOrigins = require('./allowedOrigins');

const corsOptions = {

    origin: function (origin, callback) {

        // Allow requests with no origin (e.g., mobile apps or curl requests)

        if (allowedOrigins.includes(origin) || !origin) {

            callback(null, true);

        } else {

            callback(new Error('Not allowed by CORS'));

        }

    },

    methods: ['GET', 'POST', 'PUT', 'DELETE'], // Specify allowed methods

    credentials: true, // Allow credentials if needed (cookies, etc.)

};

module.exports = corsOptions;

## 5.25 authController.js

// controllers/authController.js

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const User = require('../models/user');

const fs = require('fs');

const path = require('path');

// Signup Controller

// Signup Controller

const signup = async (req, res) => {

  const { email, username, password, birthday } = req.body;

  try {

    // Check if all fields are provided

    if (!email || !username || !password || !birthday) {

      return res.status(400).json({ message: 'All fields are required.' });

    }

    // Validate email format

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!emailRegex.test(email)) {

      return res.status(400).json({ message: 'Invalid email format.' });

    }

    // Check if passwords meet basic criteria (e.g., length)

    if (password.length < 6) {

      return res.status(400).json({ message: 'Password must be at least 6 characters long.' });

    }

    // Check if the username already exists

    const existingUser = await User.findOne({ username });

    if (existingUser) {

      return res.status(400).json({ message: 'Username already exists.' });

    }

    // Check if the email already exists

    const existingEmail = await User.findOne({ email });

    if (existingEmail) {

      return res.status(400).json({ message: 'Email already in use.' });

    }

    // Hash the password

    const hashedPassword = await bcrypt.hash(password, 10);

    // Create a new user with the validated data

    const user = new User({

      email,

      username,

      password: hashedPassword,

      birthday, // Store birthday as a string or format as needed

    });

    // Save the new user to the database

    await user.save();

    res.status(201).json({ message: 'User created successfully' });

  } catch (error) {

    console.error(error);

    res.status(500).json({ message: 'Server error' });

  }

};

// Login Controller

const login = async (req, res) => {

  const { email, password } = req.body; // Expecting email and password now

  try {

    const user = await User.findOne({ email });

    if (!user) {

      return res.status(400).json({ message: 'Invalid email or password' });

    }

    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {

      return res.status(400).json({ message: 'Invalid email or password' });

    }

    const token = jwt.sign({ id: user.\_id, email: user.email }, process.env.JWT\_SECRET, { expiresIn: '1h' });

    res.json({ token, "username":user.username });

  } catch (error) {

    res.status(500).json({ message: 'Server error' });

  }

};

const userdetails = async (req, res) => {

  const { username } = req.body;

  try {

    if (!username) {

      return res.status(400).json({ message: 'Username is required' });

    }

    // Find user details with projection to only include email and birthday

    const user = await User.findOne({ username });

    if (!user) {

      return res.status(404).json({ message: 'No user found' });

    }

    // Store email and birthday in variables

    const { email, birthday, profileImage, coverImage } = user;

    console.log(user);

    // Respond with email and birthday

    res.status(200).json({ email, birthday, profileImage, coverImage });

  } catch (error) {

    console.error('Error fetching user details:', error);

    res.status(500).json({ message: 'Server error' });

  }

};

// Upload Photo Controller

const uploadPhoto = async (req, res) => {

  const username = req.body.username; // Get username from the request body

  try {

    if (!req.file) {

      return res.status(400).json({ message: 'No file uploaded' });

    }

    // Find user by username

    const user = await User.findOne({ username });

    if (!user) {

      return res.status(404).json({ message: 'User not found' });

    }

    // Log current user profile for debugging

    console.log('User before update:', user);

    // Check if user already has a profile image

    if (user.profileImage) {

      // Construct the full path to the old image file

      const oldImagePath = path.join(\_\_dirname, '../uploads', user.profileImage);

      // Delete the old image file

      fs.unlink(oldImagePath, (err) => {

        if (err) {

          console.error('Error deleting old profile image:', err);

          // Optionally, handle the error (e.g., notify the user or continue)

        } else {

          console.log('Old profile image deleted successfully');

        }

      });

    }

    // Store the new file path in the user's profileImage field

    user.profileImage = req.file.filename; // Use filename to store in the database

    await user.save();

    res.status(200).json({ message: 'Profile photo uploaded successfully', photo: req.file.path });

  } catch (error) {

    console.error('Error uploading profile photo:', error);

    res.status(500).json({ message: 'Server error' });

  }

};

const uploadPfp = async (req, res) => {

  const username = req.body.username; // Get username from the request body

  try {

    if (!req.file) {

      return res.status(400).json({ message: 'No file uploaded' });

    }

    // Find user by username

    const user = await User.findOne({ username });

    if (!user) {

      return res.status(404).json({ message: 'User not found' });

    }

    // Log current user profile for debugging

    console.log('User before update:', user);

    // Check if user already has a profile image

    if (user.coverImage) {

      // Construct the full path to the old image file

      const oldImagePath = path.join(\_\_dirname, '../uploads', user.coverImage);

      // Delete the old image file

      fs.unlink(oldImagePath, (err) => {

        if (err) {

          console.error('Error deleting old profile image:', err);

          // Optionally, handle the error (e.g., notify the user or continue)

        } else {

          console.log('Old profile image deleted successfully');

        }

      });

    }

    // Store the new file path in the user's profileImage field

    user.coverImage = req.file.filename; // Use filename to store in the database

    await user.save();

    res.status(200).json({ message: 'Profile photo uploaded successfully', photo: req.file.path });

  } catch (error) {

    console.error('Error uploading profile photo:', error);

    res.status(500).json({ message: 'Server error' });

  }

};

module.exports = {

  signup,

  login,

  userdetails,

  uploadPhoto,

  uploadPfp,

}

## 5.26 gameController.js

// controllers/gameController.js

const Game = require('../models/Game');

const User = require('../models/user'); // Import the User model

// Add or Update High Score

const addOrUpdateHighScore = async (req, res) => {

  const { username, gameName, score } = req.body;

  try {

    // Check if the username exists in the User collection

    const userExists = await User.findOne({ username });

    if (!userExists) {

      return res.status(404).json({ message: 'Username does not exist' });

    }

    // Check if there's already a high score for this user and game

   if(gameName == "dogGame"){

      const existingScore = await Game.findOne({ username, gameName });

      if (existingScore) {

        // Update the score only if the new score is higher

        if (score > existingScore.score) {

          existingScore.score = score;

          await existingScore.save();

          return res.status(200).json({ message: 'High score updated successfully', score: existingScore });

        } else {

          return res.status(200).json({ message: 'Existing high score is higher, no update made', score: existingScore });

        }

      } else {

        // If no score exists, create a new one

        const newScore = new Game({ username, gameName, score });

        await newScore.save();

        res.status(201).json({ message: 'High score saved successfully', score: newScore });

    }

   }

    else if(gameName == "gorillaGame"){

      const existingScore2 = await Game.findOne({username, gameName});

      if(existingScore2){

        existingScore2.score += 1;

        await existingScore2.save();

        return res.status(200).json({ message: 'High score updated successfully', score: existingScore2 });

      }

      else{

        const newScore2 = new Game({ username, gameName, score });

        await newScore2.save();

        res.status(201).json({ message: 'High score saved successfully', score: newScore2 });

      }

      }

  } catch (error) {

    console.log(error);

    res.status(500).json({ message: 'Server error' });

  }

};

// Get top high scores for a specific game

const getUserHighScore = async (req, res) => {

  const { username, gameName } = req.body; // Retrieve from body

  try {

      // Find the high score for the specified user and game

      let userScore = await Game.findOne({ username, gameName });

      if (!userScore) {

          // If no score found, create and save dummy data with a score of 0

          userScore = new Game({

              username,

              gameName,

              score: 0 // Default score value

          });

          await userScore.save();

          console.log("Since no game played, made dummy data");

      }

      res.status(200).json(userScore);

  } catch (error) {

      console.log(error);

      res.status(500).json({ message: 'Server error' });

  }

};

module.exports = {

  addOrUpdateHighScore,

  getUserHighScore,

};

## 5.27 friendController.js

// controllers/friendRequestController.js

const FriendRequest = require('../models/FriendRequest');

const User = require('../models/user');

// Send a friend request

exports.sendFriendRequest = async (req, res) => {

  try {

    const { senderUsername, receiverUsername } = req.body;

    if (senderUsername === receiverUsername) {

      return res.status(400).json({ message: 'You cannot send a friend request to yourself.' });

    }

    // Check if a friend request already exists

    const existingRequest = await FriendRequest.findOne({

      senderUsername,

      receiverUsername,

      status: 'pending',

    });

    if (existingRequest) {

      return res.status(400).json({ message: 'Friend request already sent.' });

    }

    const friendRequest = new FriendRequest({

      senderUsername,

      receiverUsername,

    });

    await friendRequest.save();

    res.status(200).json({ message: 'Friend request sent.' });

  } catch (err) {

    res.status(500).json({ message: 'Server error.', error: err });

  }

};

// Controller for accepting a friend request

exports.acceptFriendRequest = async (req, res) => {

  try {

    const { senderUsername, receiverUsername } = req.params; // Get usernames from URL parameters

    console.log("Received parameters:", { senderUsername, receiverUsername });

    const sender = await User.findOne({username:senderUsername});

    const receiver = await User.findOne({username:receiverUsername});

    // Find the pending friend request by sender and receiver usernames

    const friendRequest = await FriendRequest.findOne({

      senderUsername,

      receiverUsername,

      status: 'pending'

    });

    if (friendRequest) {

      // Update friend request status to 'accepted'

      friendRequest.status = 'accepted';

      await friendRequest.save();

        const friendRequesttwo = await FriendRequest.findOne({

          senderUsername:receiverUsername,

          receiverUsername:senderUsername,

          status: 'pending'

        });

        if(friendRequesttwo){

          friendRequesttwo.status = 'accepted';

          await friendRequesttwo.save();

        }

        sender.friends.push(receiverUsername);

        receiver.friends.push(senderUsername);

        await sender.save();

        await receiver.save();

      res.status(200).json({ message: 'Friend request accepted' });

    } else {

      console.log("Friend request not found with parameters:", { senderUsername, receiverUsername });

      return res.status(400).json({ message: 'Friend request not found.' });

    }

  } catch (err) {

    console.error('Error accepting friend request:', err);

    res.status(500).json({ message: 'Server error.', error: err });

  }

};

// Reject a friend request

exports.rejectFriendRequest = async (req, res) => {

  try {

    const { senderUsername, receiverUsername } = req.params; // Get usernames from URL parameters

    // Find the pending friend request by sender and receiver usernames

    const friendRequest = await FriendRequest.findOne({

      senderUsername,

      receiverUsername,

      status: 'pending'

    });

    if (!friendRequest) {

      return res.status(400).json({ message: 'Invalid or already processed friend request.' });

    }

    // Update friend request status to 'rejected'

    friendRequest.status = 'rejected';

    await friendRequest.save();

    res.status(200).json({ message: 'Friend request rejected.' });

  } catch (err) {

    res.status(500).json({ message: 'Server error.', error: err });

  }

};

// Get friend requests for a user

// Controller function to get friend requests based on username in the URL parameters

exports.getFriendRequests = async (req, res) => {

  try {

    // Get the username from the request parameters

    const { username } = req.params;

    if (!username) {

      return res.status(400).json({ message: 'Username is required.' });

    }

    // Find the user by username

    const user = await User.findOne({ username });

    if (!user) {

      return res.status(404).json({ message: 'User not found.' });

    }

    // Find friend requests where the receiver is the user and status is 'pending'

    const friendRequests = await FriendRequest.find({ receiverUsername: username, status: 'pending' })

      .populate('senderUsername', 'receiverUsername'); // Populate senderUsername with the actual username

    // Return the list of friend requests as a response

    res.status(200).json(friendRequests);

  } catch (err) {

    // Return a 500 error if any server error occurs

    res.status(500).json({ message: 'Server error.', error: err });

  }

};

//display friend

exports.showFriends = async (req, res) => {

  try {

    const { username } = req.params; // Get username from URL parameters

    if (!username) {

      return res.status(400).json({ message: 'Username is required.' });

    }

    const flist = await User.findOne({username});

    console.log(flist.friends)

    // Return the list of friends as a response

    res.status(200).json({"friends":flist.friends});

  } catch (err) {

    res.status(500).json({ message: 'Server error.', error: err });

  }

};

## 5.28 authMiddleware.js

const jwt = require('jsonwebtoken');

// Middleware to check if the user is authenticated

const verifyToken = (req, res, next) => {

    const authHeader = req.headers['authorization'];

    if (!authHeader || !authHeader.startsWith('Bearer ')) {

        return res.status(403).json({ message: 'Access denied. No token provided or token format is incorrect.' });

    }

    const token = authHeader.split(' ')[1]; // Extract token from "Bearer <token>"

    try {

        // Verify the token

        const decoded = jwt.verify(token, process.env.JWT\_SECRET);

        req.user = decoded; // Attach the decoded user information to the request object

        next();

    } catch (err) {

        console.error('Token verification failed:', err); // Log the error for debugging

        res.status(401).json({ message: 'Invalid token.' });

    }

};

module.exports = verifyToken;

5.23 credentials.js

const allowedOrigins = require('../config/allowedOrigins')

const credentials = (req,res,next) => {

    const origin = req.headers.origin

    if(allowedOrigins.includes(origin)){

        res.header('Access-Control-Allow-Credentials', true)

    }

    next()

}

module.exports = credentials

## 5.29 Game.js

// models/Game.js

const mongoose = require('mongoose');

const gameSchema = new mongoose.Schema({

username: {

type: String,

required: true,

},

gameName: {

type: String,

required: true,

},

score: {

type: Number,

required: true,

},

createdAt: {

type: Date,

default: Date.now,

},

});

module.exports = mongoose.model('Game', gameSchema);

## 5.30 user.js (model)

// models/User.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

  email: {

    type: String,

    required: true,

    unique: true,

    trim: true,

    lowercase: true, // Ensures email is stored in lowercase

  },

  username: {

    type: String,

    required: true,

    unique: true,

    trim: true,

  },

  password: {

    type: String,

    required: true,

  },

  friends:{

    type: Array,

    default: [],

  },

  coverImage: {

    type: String,

    default: '', // Default to an empty string if no image is provided

  },

  profileImage: {

    type: String,

    default: '', // Default to an empty string if no image is provided

  },

  birthday: {

    type: String, // Store birthday as a string in format DD-MMM-YYYY or another format as needed

    required: true,

  },

}, {

  timestamps: true, // Automatically add createdAt and updatedAt fields

});

module.exports = mongoose.model('User', userSchema);

## 5.31 game.js (model)

// models/Game.js

const mongoose = require('mongoose');

const gameSchema = new mongoose.Schema({

  username: {

    type: String,

    required: true,

  },

  gameName: {

    type: String,

    required: true,

  },

  score: {

    type: Number,

    required: true,

    default:0,

  },

  createdAt: {

    type: Date,

    default: Date.now,

  },

});

module.exports = mongoose.model('Game', gameSchema);

## 5.32 friend.js (model)

// models/FriendRequest.js

const mongoose = require('mongoose');

const friendRequestSchema = new mongoose.Schema({

  senderUsername: {

    type: String,

    required: true,

  },

  receiverUsername: {

    type: String,

    required: true,

  },

  status: {

    type: String,

    enum: ['pending', 'accepted', 'rejected'],

    default: 'pending',

  },

  createdAt: {

    type: Date,

    default: Date.now,

  },

});

module.exports = mongoose.model('FriendRequest', friendRequestSchema);

## 5.33 auth.js ( route )

// routes/auth.js

const express = require('express');

const { signup, login, userdetails, uploadPhoto, uploadPfp} = require('../controllers/authController');

const router = express.Router();

const multer = require('multer');

const path = require('path')

//multer lets u store anywhere file

// Configure multer storage

const storage = multer.diskStorage({

  destination: function (req, file, cb) {

    // Resolve the uploads directory relative to the current file

    cb(null, path.join(\_\_dirname, '../uploads'));

  },

  filename: function (req, file, cb) {

    // Ensure username is available and sanitize filename

    const username = req.body.username || 'default\_user'; // Default value if username is not provided

    //const ext = path.extname(file.originalname);  Get file extension

    cb(null, `${username}\_${Date.now()}\_${file.originalname}`); // Unique filename

  }

});

const upload = multer({storage});

// Signup Route

router.post('/signup', signup);

// Login Route

router.post('/login', login);

// details taker

router.post('/userdetails', userdetails);

// Upload profile photo route (requires token authentication)

router.post('/upload-photo', upload.single('photo') ,uploadPhoto);

router.post('/upload-pfp', upload.single('photop') ,uploadPfp);

module.exports = router;

## 5.34 game.js ( route )

// routes/game.js

const express = require('express');

const { addOrUpdateHighScore, getUserHighScore } = require('../controllers/gameController');

const router = express.Router();

// Route to add or update high score

router.post('/high-score', addOrUpdateHighScore);

// Route to get top high scores for a specific game

router.post('/top-scores', getUserHighScore);

module.exports = router;

## 5.35 friendrout.js ( route )

const express = require('express');

const router = express.Router();

const friendRequestController = require('../controllers/friendRequestController');

const verifyToken = require('../middleware/authMiddleware');

// Route to send a friend request

router.post('/send', friendRequestController.sendFriendRequest);

// Route to accept a friend request

router.post('/accept/:senderUsername/:receiverUsername',  friendRequestController.acceptFriendRequest);

// Route to reject a friend request

router.post('/reject/:senderUsername/:receiverUsername',  friendRequestController.rejectFriendRequest);

// Route to get friend requests

router.get('/requests/:username',  friendRequestController.getFriendRequests);

router.get('/show/:username',  friendRequestController.showFriends);

module.exports = router;

## 5.36 server.js

// server.js

const express = require('express');

const mongoose = require('mongoose');

const dotenv = require('dotenv');

const authRoutes = require('./routes/auth');

const gameRoutes = require('./routes/game');

const cors = require('cors');

const credentials = require('./middleware/credentials');

const corsOptions = require('./config/corsOptions');

dotenv.config();

const verifyToken = require('./middleware/authMiddleware');

const app = express();

// Middleware

app.use(credentials); // Apply credentials middleware

app.use(cors(corsOptions)); // Apply CORS with specified options

app.use(express.json()); // Middleware to parse JSON request bodies

// Routes

app.use('/api/auth', authRoutes); // Auth routes (e.g., login, register)

app.use('/api/game', verifyToken, gameRoutes); // Game routes with token verification

// Connect to MongoDB

mongoose.connect(process.env.MONGO\_URI, {

  useNewUrlParser: true,

  useUnifiedTopology: true,

})

  .then(() => console.log('Connected to MongoDB'))

  .catch((error) => console.error('MongoDB connection error:', error));

// Start the server

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

# 6.Results

## 6.1 Validation and Naming Conventions

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No | Control Id | Validation Used | Reason |
| 1 | UserName | RequiredFieldValidator | Name Cannot be Empty. |
| 2 | UserEmail | RequiredFieldValidator | Email cannot be empty |
| 3 | Contact | RequiredFieldValidator | Contact cannot be empty |
| 4 | Password | RequiredFieldValidator | Password cannot be empty |
| 5 | Confirm Password | RequiredFieldValidator | Confirm Password cannot be empty |
| 6 | PlayDogGameBtn | RequiredFieldValidator | Button should be visible and clickable |
| 7 | PlayGorillaGameBtn | RequiredFieldValidator | Button should be visible and clickable |
| 8 | AccountBtn | RequiredFieldValidator | Button should be visible and clickable |
| 9 | GameScoreDisplay | RequiredFieldValidator | Score display should be visible |
| 10 | GameControls | RequiredFieldValidator | Controls should be functional |
| 11 | FinalScoreDisplay | RequiredFieldValidator | Final score should be displayed |

Table Validation and Naming Conventions

## 6.2 Screenshots

### 6.2.1 Login Page

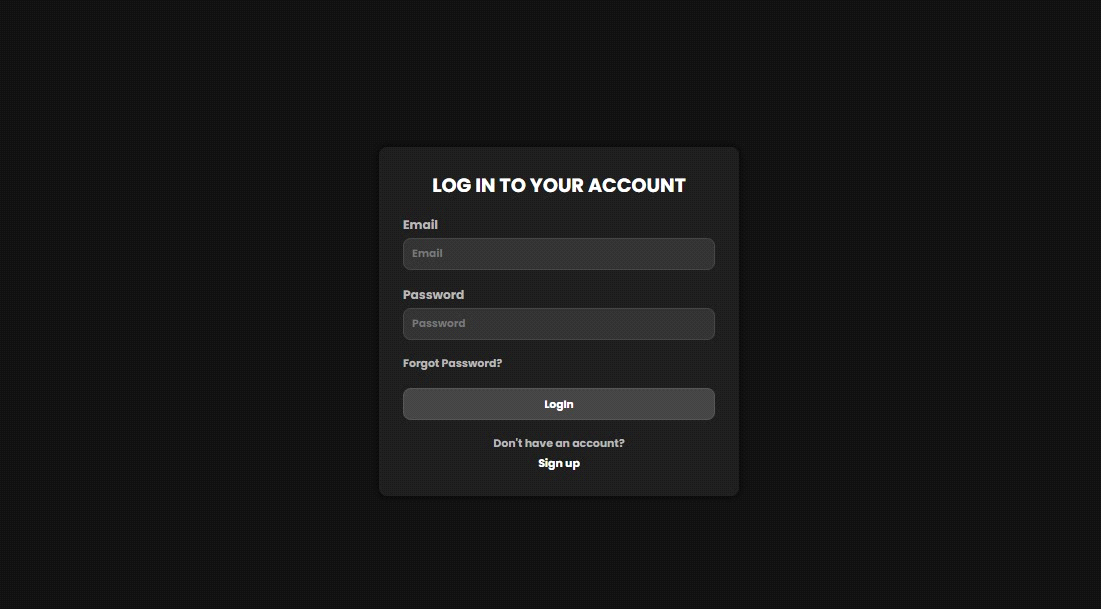


Figure Login Page

### 6.2.2 Signup Page

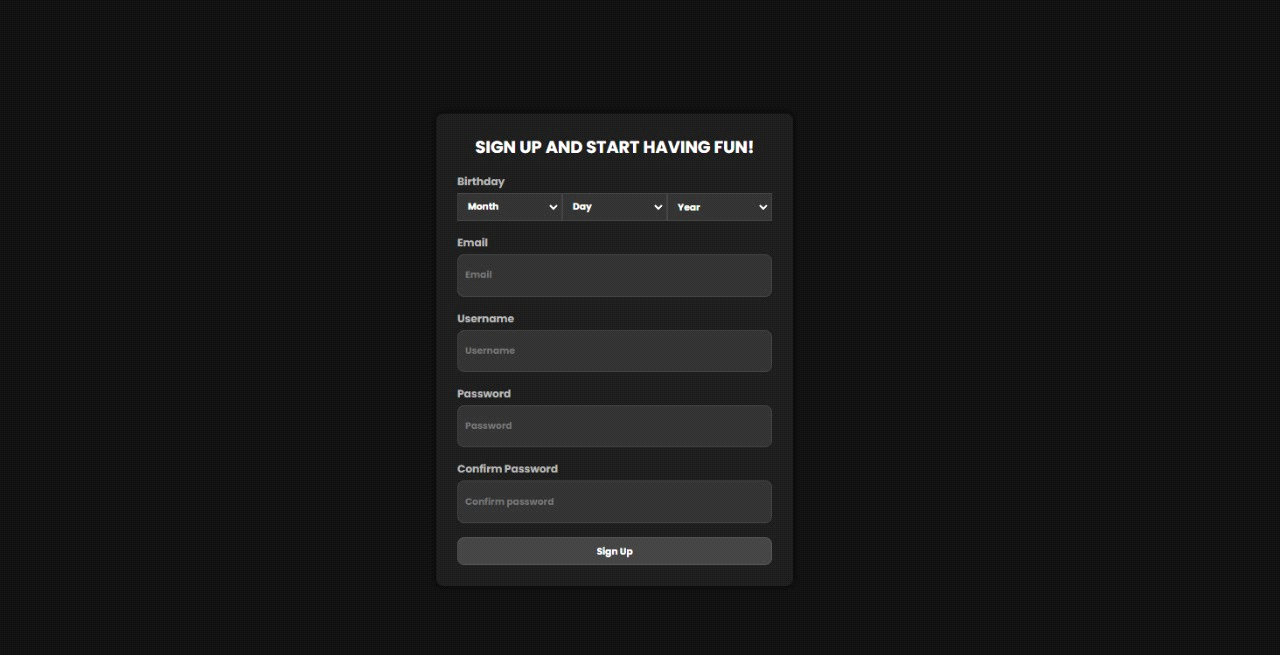


Figure Signup Page

### 6.2.3 Home Page

**A screenshot of a video game

Description automatically generated**

Figure Home Page

### 6.2.4 Game1 Page

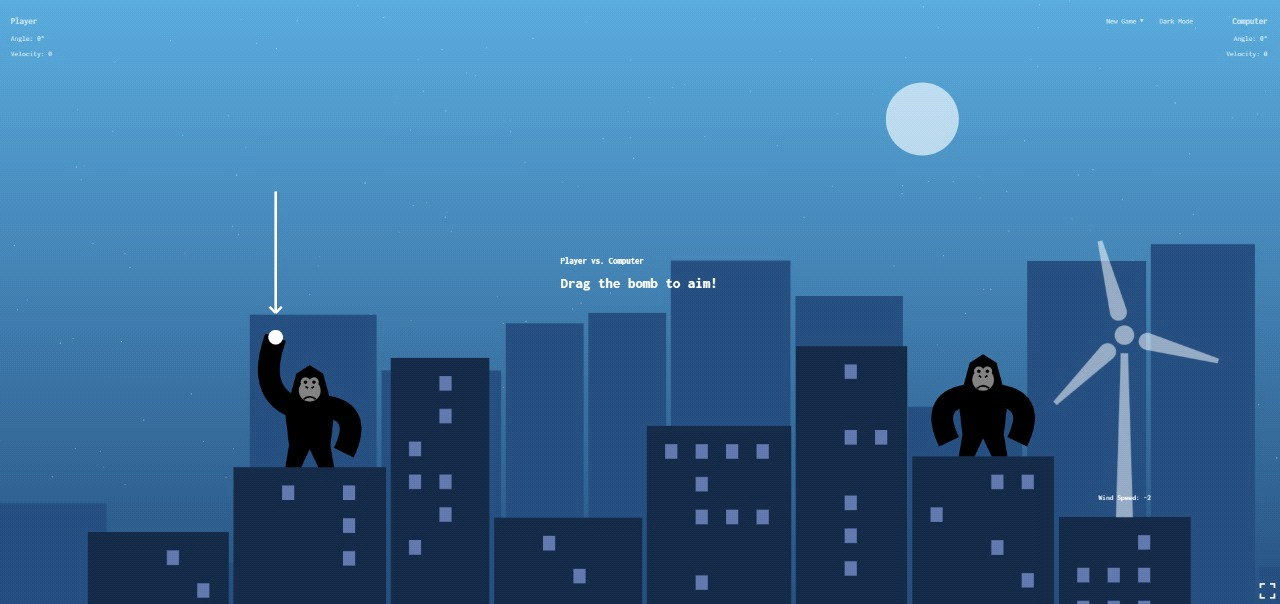


Figure Game1 Page

### 6.2.5 Game2 Page



Figure Game2 Page

### 6.2.6 Game3 Page

A screenshot of a video game

Description automatically generated

Figure Game3 Page

### 6.2.7 Accounts Page

A screenshot of a video game

Description automatically generated

Figure Accounts Page

### 6.2.8 FriendDetail Page

A screenshot of a video game

Description automatically generated

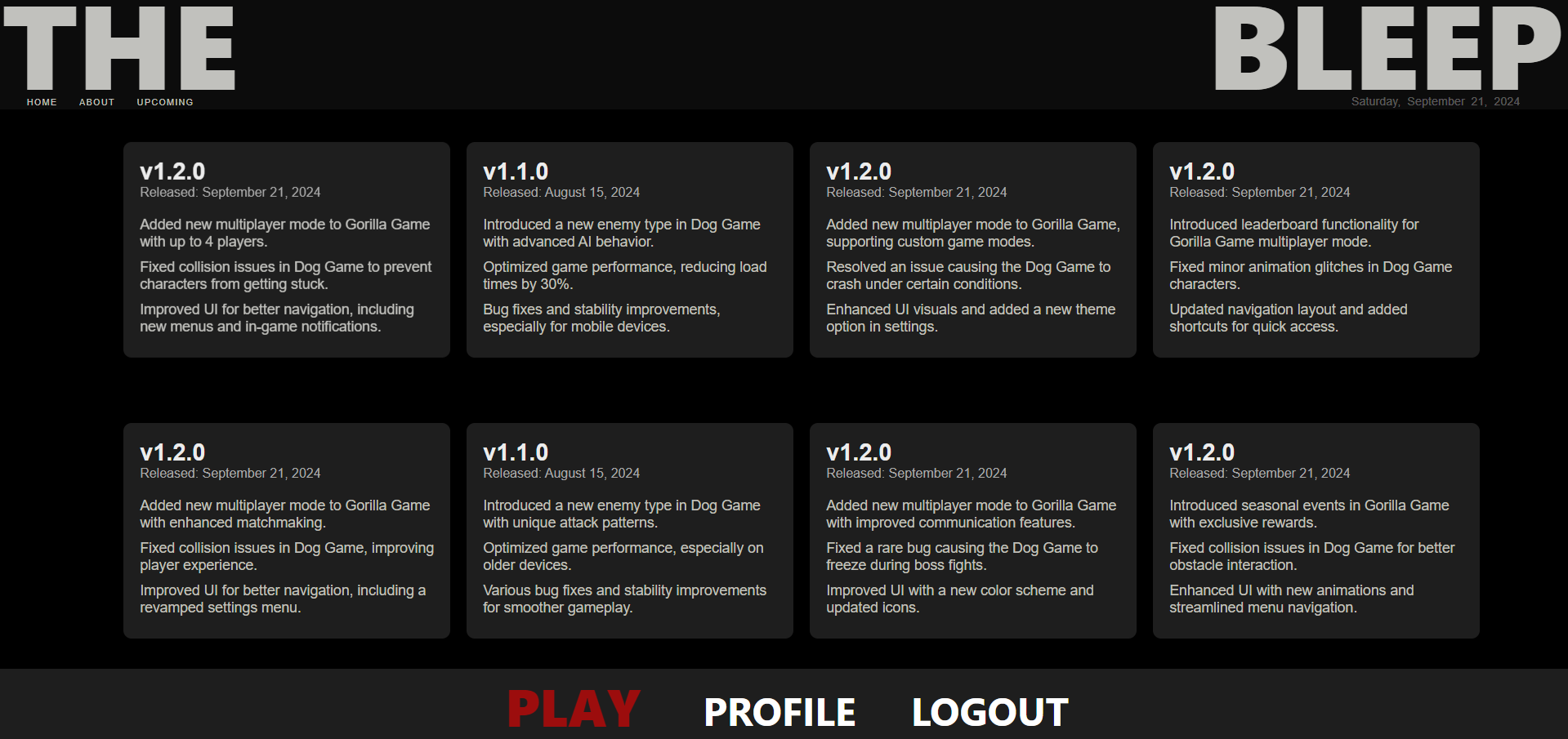
### 6.2.6 About Page

A screenshot of a computer

Description automatically generated

Figure About Page

### 6.2.7 Patches Page



# 7. Future Scope and Conclusion

## 7.1 Future Enhancement

Mobile Integration:

Android and iOS Apps: We plan to extend the website’s functionality by integrating it with dedicated Android and iOS applications. This would allow users to enjoy a seamless gaming experience across multiple devices and enhance user engagement with mobile-specific features.

Live Streaming:

Real-time Gameplay and Updates: Implementing live streaming capabilities for games will enable users to watch live gameplay, view game updates, and participate in live events. This feature can boost interaction and provide a more immersive experience for the community.

Enhanced User Interaction:

In-Game Chat and Social Features: Introducing in-game chat and social features such as friend lists and leaderboards to foster community engagement and competition among players.

Customizable Avatars and In-Game Items: Allow users to personalize their avatars and acquire in-game items through achievements or microtransactions.

Improved Game Analytics:

Detailed Player Statistics: Providing detailed statistics and performance metrics for players to track their progress and improve their gameplay.

Behavioral Analytics: Implementing analytics to understand user behavior and preferences, allowing for tailored content and game updates.

Expanded Game Content:

Additional Game Modes and Levels: Adding new game modes, levels, and challenges to keep the content fresh and engaging for returning players.

Seasonal Events and Updates: Introducing seasonal events and limited-time challenges to keep the game dynamic and encourage continued play.

## 7.2 Conclusion

Project Overview:

The project, titled "Web Game Website", has been developed with a focus on providing an engaging and user-friendly gaming experience. The website features a range of interactive controls and dynamic elements designed to enhance user enjoyment.

User Experience:

The website has been crafted to meet the needs of its target audience, with a design that prioritizes ease of use and accessibility. The intuitive interface and clear navigation ensure that users with basic computer knowledge can effortlessly engage with the site.

Achievement of Objectives:

The project has successfully met all identified objectives, including providing an attractive and functional interface for players. The website's design and features align with the goals set forth, offering a satisfying and interactive gaming experience.

Future Potential:

With planned enhancements and ongoing updates, the website is positioned to continue evolving and adapting to user needs. The integration of mobile apps and live streaming, along with the expansion of game content and social features, will contribute to the site's growth and sustained success.

# 8.References

1. [MongoDB Documentation](https://www.mongodb.com/docs/): Guides on MongoDB setup and management for Game database.
2. [**JavaScript Documentation**:](https://devdocs.io/javascript/) Information on modern JavaScript features and best practices for building dynamic web applications.
3. [Express.js Documentation](https://devdocs.io/express/): Detailed information on building web applications with Express.js, including routing, middleware, and error handling.
4. [**HTML & CSS Documentation**](https://developer.mozilla.org/en-US/docs/Web/CSS): Resources for learning HTML and CSS to build the frontend of your application.
5. [ChatGPT Reference](https://chatgpt.com/c/00ce81f5-3bef-4ecd-a8e1-a53237a477e0): Contains guidance or clarification related to how Game works

# 9.Annexure

**9.1 Figures List**

[Figure 1 Class Diagram 10](#_Toc177738748)

[Figure 2 User Authentication Use Case Diagram 10](#_Toc177738749)

[Figure 3 Home Page Use Case Diagram 11](#_Toc177738750)

[Figure 4 Account Page Use Case Diagram 11](#_Toc177738751)

[Figure 5 Game Interaction Use Case Diagram 12](#_Toc177738752)

[Figure 6 Signin Sequence Diagram 13](#_Toc177738753)

[Figure 7 Signup Sequence Diagram 13](#_Toc177738754)

[Figure 8 Homepage Sequence Diagram 14](#_Toc177738755)

[Figure 9 Account Sequence Diagram 15](#_Toc177738756)

[Figure 10 Game Sequence Diagram Diagram 16](#_Toc177738757)

[Figure 11 SignIn Activity Diagram 17](#_Toc177738758)

[Figure 12 SignUp Activity Diagram 18](#_Toc177738759)

[Figure 13 Home Page Activity Diagram 18](#_Toc177738760)

[Figure 14 Account Page Activity Diagram 19](#_Toc177738761)

[Figure 15 Speedy Pup Activity Diagram 20](#_Toc177738762)

[Figure 16 Gorilla Clash Activity Diagram 21](#_Toc177738763)

[Figure 17 Samurai Activity Diagram 22](#_Toc177738764)

[Figure 18 Home State Diagram 23](#_Toc177738765)

[Figure 19 Account State Diagram 23](#_Toc177738766)

[Figure 20 Speedy Pup Game State Diagram 24](#_Toc177738767)

[Figure 21 Gorilla Game State Diagram 25](#_Toc177738768)

[Figure 22 Samurai Game State Diagram 26](#_Toc177738769)

[Figure 23 Package Diagram 27](#_Toc177738770)

[Figure 24 Component Diagram 27](#_Toc177738771)

[Figure 25 Deployment Diagram 28](#_Toc177738772)

[Figure 26 Database Design 28](#_Toc177738773)

[Figure 27 User collection 29](#_Toc177738774)

[Figure 28 frindrequest collection accepted 29](#_Toc177738775)

[Figure 29 frindrequest collection rejected 29](#_Toc177738776)

[Figure 30 games collection dogGame 29](#_Toc177738777)

[Figure 31 games collection gorillaGame 30](#_Toc177738778)

[Figure 32 Login Page 141](#_Toc177738779)

[Figure 33 Signup Page 141](#_Toc177738780)

[Figure 34 Home Page 142](#_Toc177738781)

[Figure 35 Game1 Page 142](#_Toc177738782)

[Figure 36 Game2 Page 143](#_Toc177738783)

[Figure 37 Game3 Page 143](#_Toc177738784)

[Figure 38 Accounts Page 144](#_Toc177738785)

[Figure 39 About Page 144](#_Toc177738786)

**9.2 Table List**

[Table 1 Event Table 9](#_Toc177738787)

[Table 2 Validation and Naming Conventions 140](#_Toc177738788)