



**Karunya** INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

A CHRISTIAN MINORITY RESIDENTIAL INSTITUTION

AICTE Approved & NAAC Accredited

## **A SKILL BASED EVALUATION REPORT**

**SUBMITTED BY**

**Jerlin Sam RN (URK22AI1094)**

**COURSE CODE**

**20CS2056**

**COURSE NAME**

**WEB TECHNOLOGY**

**APRIL 2024**



**DIVISION OF DIVISION OF DATA SCIENCE AND CYBER  
SECURITY**

**SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY**

# INDUSTRIAL CERTIFICATION



## Statement of Achievement

jerlin sam

has successfully achieved student level credential for completing the JavaScript Essentials 1 course, provided by Cisco Networking Academy in collaboration with OpenEDG JS Institute.

The graduate is able to proficiently:

- Understand the syntax of the core JavaScript language that allows for working with variables, operators, flow control, and functions.
- Understand the basics of the JavaScript data types system, distinguishing between primitive and complex types.
- Think algorithmically and can analyze a problem using a programmatic conceptual apparatus.
- Choose a data type adequate to the problem being solved and use suitable flow control means.
- Design, develop, and improve very simple JavaScript programs.
- Interpret and handle basic exceptions related to errors in program execution.
- Understand a programmer's work in the software development process and the role of fundamental development tools.
- Understand how a program is interpreted and executed in an actual computer environment, local or remote.



Scan to Verify

*Laura Quintana*

Laura Quintana  
Vice President and General Manager  
Cisco Networking Academy

April 11, 2024

# Statement of Achievement

jerlin sam

has successfully achieved student level credential for completing the JavaScript Essentials 2 course, provided by Cisco Networking Academy in collaboration with OpenEDG JS Institute.

The graduate has studied:

- Techniques for constructing and modifying objects, including the use of prototypes and inheritance.
- Methods for defining and encapsulating class properties and managing array data, including JSON conversion.
- Utilization of the Math object and regular expressions for mathematical and string operations.
- Advanced function techniques and asynchronous programming, including callbacks and iterators.
- Problem analysis and program development using algorithmic thinking and object-oriented principles.



Scan to Verify

April 12, 2024

*Laura Quintana*

Laura Quintana  
Vice President and General Manager  
Cisco Networking Academy

**TITLE**  
**PIXELS\_RUSH**

***A REAL TIME APPLICATION REPORT***

***Submitted by***

**ANTO GODWIN AL (URK22AI1076)**

**JERLIN SAM RN (URK22AI1094)**



**DIVISION OF DATA SCIENCE AND CYBER SECURITY**

**KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES  
(Declared as Deemed-to-be-under Sec-3 of the UGC Act,  
1956) Karunya Nagar, Coimbatore - 641 114. INDIA**

**APRIL 2024**

# PROJECT REVIEW FORM



**Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES**

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

School of Computer Science & Technology  
Division of Computer Science & Engineering

## Skill-Based Evaluation - Project Zero Review Form

Course Code & Name : 20CS2056- Web Technology

Academic Year : 2023-24

Semester: Even Semester

#	Reg No	Name	Sign
1	URK22AI1076	Anto Godwin AL	
2	URK22AI1094	Jerlin Sam RN	

Title of the project : PixelRush

Abstract of the project : Our Website is a multiplayer and also an offline GAME

### Functionality:

Players control paintbrushes to cover the game area with their color.

Collision detection prevents trails from intersecting.

Scoring is based on the area covered.

Power-ups may appear for advantages.

Supports multiplayer.

### Validation:

Ensure boundaries are respected.

Detect and handle collisions accurately.

Validate user inputs.

Verify scoring calculation.

Test power-up effects.

### Technology Stack

Front-End: HTML,CSS,Java Script

Back-End:Firebase(google)

Database:Firebase Realtime Database

Approved / ~~Not Approved~~ →

Faculty Signature

## **ABSTRACT**

PIXEL\_RUSH is a multiplayer web-based game inspired by the popular game "paint.io". Developed using HTML, CSS, and JavaScript, with Firebase from Google for database and server integration, PIXEL\_RUSH offers an immersive gaming experience where players compete to claim territory and dominate the canvas. The game features a dynamic pixelated environment where players control colorful avatars and use various tools to paint and expand their territory. Players can collaborate or compete with each other in real-time, strategizing to outmaneuver opponents and secure dominance over the canvas. Key features of PIXEL\_RUSH include real-time multiplayer functionality, customizable avatars, a diverse range of painting tools and effects, and an intuitive user interface designed for seamless gameplay across different devices. Throughout the development process, challenges such as optimizing performance for large numbers of concurrent players and ensuring data integrity in the multiplayer environment were addressed. By leveraging Firebase's robust infrastructure and scalable database solutions, these challenges were effectively managed, resulting in a smooth and engaging gaming experience for players. PIXEL\_RUSH represents an innovative fusion of classic painting mechanics with modern multiplayer gaming, offering players a creative and competitive platform to express themselves and engage with others in a vibrant digital world.

## PROBLEM STATEMENT

In today's digital landscape, there is a growing demand for interactive and engaging multiplayer games that provide players with both creative expression and competitive challenges. However, many existing multiplayer games lack the unique blend of painting mechanics and real-time interaction that defines the experience of "paint.io".

To address this gap, our project, Pixel\_rush, aims to develop a multiplayer painting game that combines the addictive gameplay of "paint.io" with innovative features and seamless multiplayer functionality. The game will provide players with a platform to unleash their creativity, collaborate with others, and compete for dominance over the canvas in a dynamic and visually captivating pixelated environment.

Key challenges include:

1. **Real-time Multiplayer Integration:** Implementing robust networking solutions to enable smooth and synchronized gameplay for multiple players simultaneously, ensuring a seamless experience across various devices and network conditions.
2. **Painting Mechanics and Tools:** Designing and implementing a diverse range of painting tools and effects that allow players to express themselves creatively while maintaining balance and fairness in competitive gameplay.
3. **Performance Optimization:** Optimizing game performance to accommodate a large number of concurrent players without sacrificing responsiveness or visual quality, particularly in resource-intensive tasks such as rendering and data synchronization.
4. **Data Management and Security:** Implementing secure data management practices to protect player data and ensure integrity in the multiplayer environment, including user authentication, data validation, and server-side security measures.

By addressing these challenges and leveraging the capabilities of modern web technologies such as HTML, CSS, JavaScript, and Firebase, Pixel\_rush aims to deliver an engaging and immersive gaming experience that captivates players and establishes itself as a standout multiplayer painting game in the digital gaming market.



## **METHODOLOGY / ARCHITECTURE**

### **1. Frontend Development:**

- HTML, CSS, and JavaScript: Utilize these core web technologies to create the user interface and game mechanics, including the canvas for painting, player avatars, and interactive elements.

- Responsive Design: Ensure compatibility across various devices and screen sizes to provide a seamless gaming experience for all players.

### **2. Backend Development:**

- Firebase Integration: Utilize Firebase from Google for backend services including real-time database, authentication, and hosting.

- Real-time Database: Store and synchronize game state, player actions, and other relevant data in real-time to facilitate multiplayer interactions.

- Authentication: Implement user authentication to secure player accounts and ensure data privacy and integrity.

- Cloud Hosting: Host the game application on Firebase Hosting to ensure scalability, reliability, and efficient content delivery to players worldwide.

### **3. Multiplayer Functionality:**

- Networking: Implement WebSocket or Firebase Realtime Database to enable real-time communication between players and the game server.

- Player Interactions: Design game mechanics that allow players to paint on the canvas, collaborate with others, and compete for territory in real-time.

- Server-side Logic: Develop server-side logic to manage game sessions, handle player actions, and enforce game rules to ensure fairness and consistency across all players.

### **4. Painting Mechanics and Tools:**

- Pixelated Environment: Create a pixelated canvas where players can paint and claim territory using a grid-based system.



- Painting Tools: Implement a variety of painting tools such as brushes, erasers, fill tools, and color palettes to allow players to express themselves creatively.

- Effects and Customizations: Introduce special effects, power-ups, and customizable avatars to enhance gameplay and provide players with unique abilities and identities.

## 5. Performance Optimization:

- Client-side Optimization: Optimize client-side rendering and input handling to ensure smooth and responsive gameplay, even on low-end devices.

- Server-side Scalability: Architect the server infrastructure to handle a large number of concurrent players efficiently, including load balancing, caching, and scaling mechanisms.

- Network Optimization: Minimize network latency and bandwidth usage by optimizing data transmission and leveraging Firebase's built-in optimizations for real-time communication.

## 6. Security and Data Management:

- User Authentication: Implement secure user authentication using Firebase Authentication to protect player accounts from unauthorized access.

- Data Validation: Validate user inputs and game state changes on both client and server to prevent cheating and ensure data integrity.

- Privacy Compliance: Ensure compliance with data protection regulations and best practices for handling user data, including encryption, access controls, and data retention policies.

## 7. Testing and Iteration:

- Unit Testing: Test individual components and functions to ensure correctness and reliability.

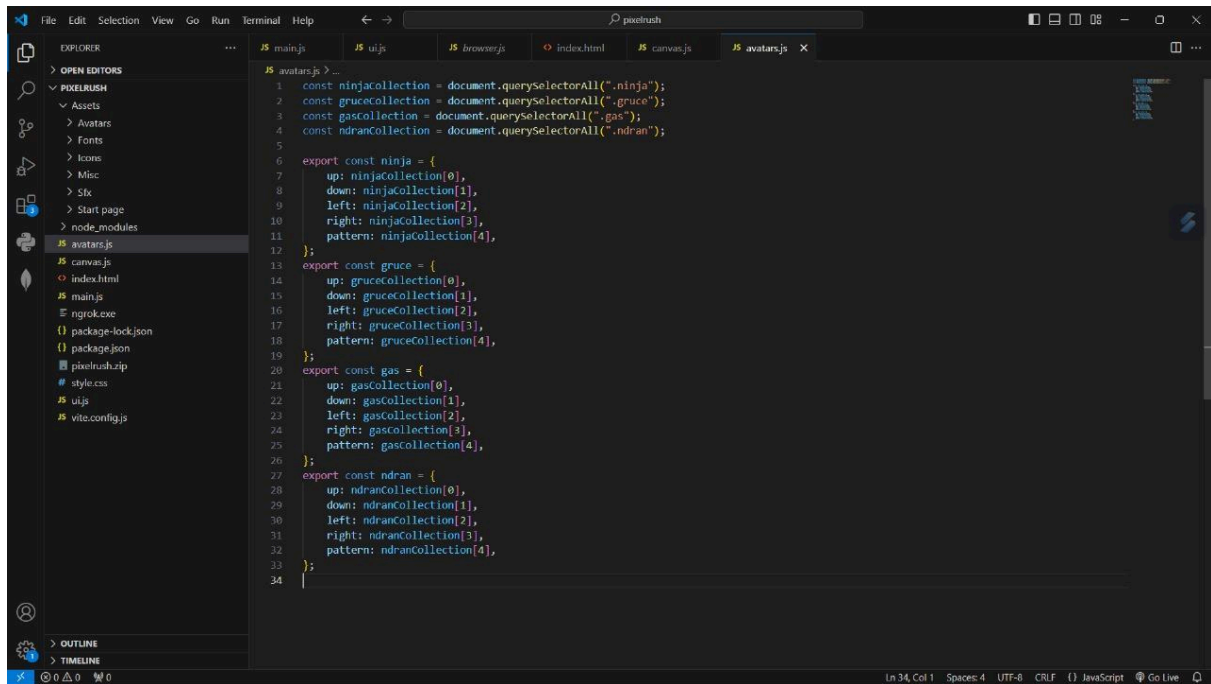
- Integration Testing: Validate the interaction between frontend and backend components, including multiplayer functionality and data synchronization.

- User Testing: Gather feedback from beta testers and early users to identify usability issues, bugs, and potential improvements, and iterate on the game design and features accordingly.

# IMPLEMENTATION – CODING AND OUTPUT SCREENSHOT

Avatar controls:

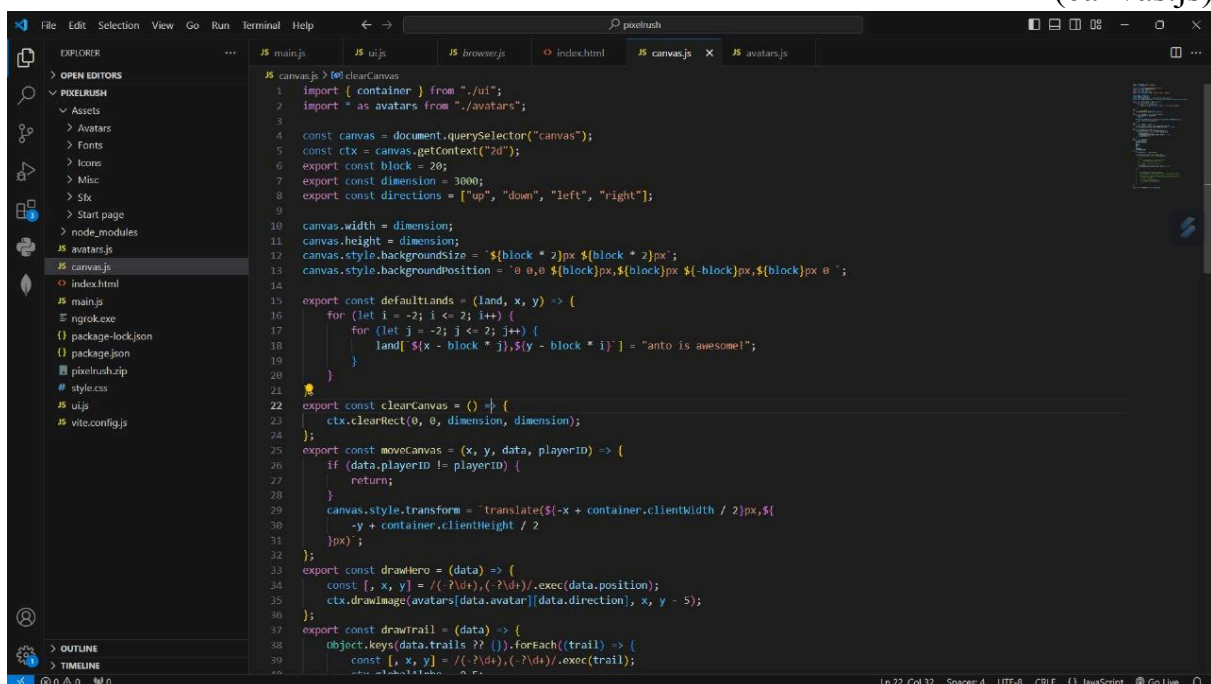
(avatars.js)



```
1 const ninjaCollection = document.querySelectorAll(".ninja");
2 const gruceCollection = document.querySelectorAll(".gruce");
3 const gasCollection = document.querySelectorAll(".gas");
4 const ndranCollection = document.querySelectorAll(".ndran");
5
6 export const ninja = {
7   up: ninjaCollection[0],
8   down: ninjaCollection[1],
9   left: ninjaCollection[2],
10  right: ninjaCollection[3],
11  pattern: ninjaCollection[4],
12 };
13
14 export const gruce = {
15   up: gruceCollection[0],
16   down: gruceCollection[1],
17   left: gruceCollection[2],
18   right: gruceCollection[3],
19   pattern: gruceCollection[4],
20 };
21
22 export const gas = {
23   up: gasCollection[0],
24   down: gasCollection[1],
25   left: gasCollection[2],
26   right: gasCollection[3],
27   pattern: gasCollection[4],
28 };
29
30 export const ndran = {
31   up: ndranCollection[0],
32   down: ndranCollection[1],
33   left: ndranCollection[2],
34   right: ndranCollection[3],
35   pattern: ndranCollection[4],
36 };
37
38
```

In-Game design:

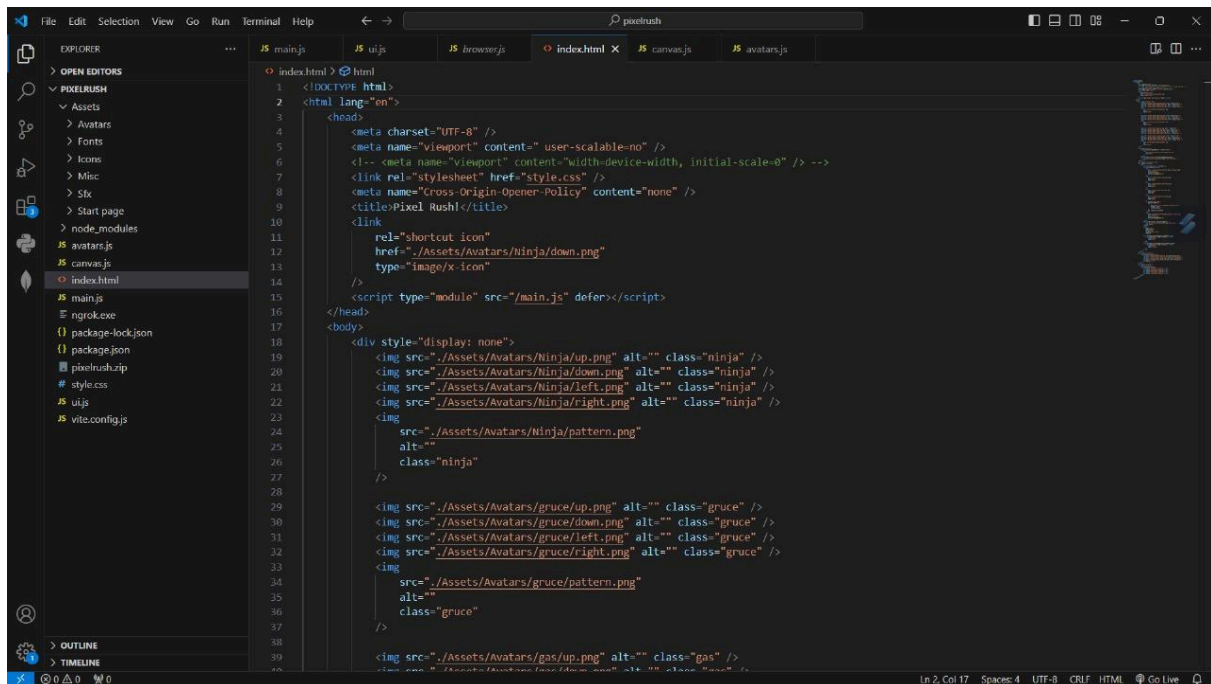
(canvas.js)



```
1 import { container } from "./ui";
2 import * as avatars from "./avatars";
3
4 const canvas = document.querySelector("canvas");
5 const ctx = canvas.getContext("2d");
6 export const block = 20;
7 export const dimension = 3000;
8 export const directions = ["up", "down", "left", "right"];
9
10 canvas.width = dimension;
11 canvas.height = dimension;
12 canvas.style.backgroundColor = `rgb(${block * 2}, ${block * 2}, ${block * 2})`;
13 canvas.style.backgroundPosition = `0 0, 0 ${block}px, ${block}px 0, ${-block}px, ${block}px 0`;
14
15 export const defaultLands = (land, x, y) => {
16   for (let i = -2; i <= 2; i++) {
17     for (let j = -2; j <= 2; j++) {
18       land[`${x - block * j}, ${y - block * i}`] = "anto is awesome!";
19     }
20   }
21 }
22
23 export const clearCanvas = () => {
24   ctx.clearRect(0, 0, dimension, dimension);
25 }
26
27 export const moveCanvas = (x, y, data, playerId) => {
28   if (data.playerID !== playerId) {
29     return;
30   }
31   canvas.style.transform = `translate(${x - container.clientWidth / 2}px, ${y - container.clientHeight / 2}px)`;
32 }
33
34 export const drawHero = (data) => {
35   const [x, y] = /(-?\d+)/.exec(data.position);
36   ctx.drawImage(avatars[data.avatar][data.direction], x, y - 5);
37 }
38
39 export const drawTrail = (data) => {
40   Object.keys(data.trails ?? []).forEach((trail) => {
41     const [x, y] = /(-?\d+)/.exec(trail);
42   });
43 }
44
```

## Lobby Design:

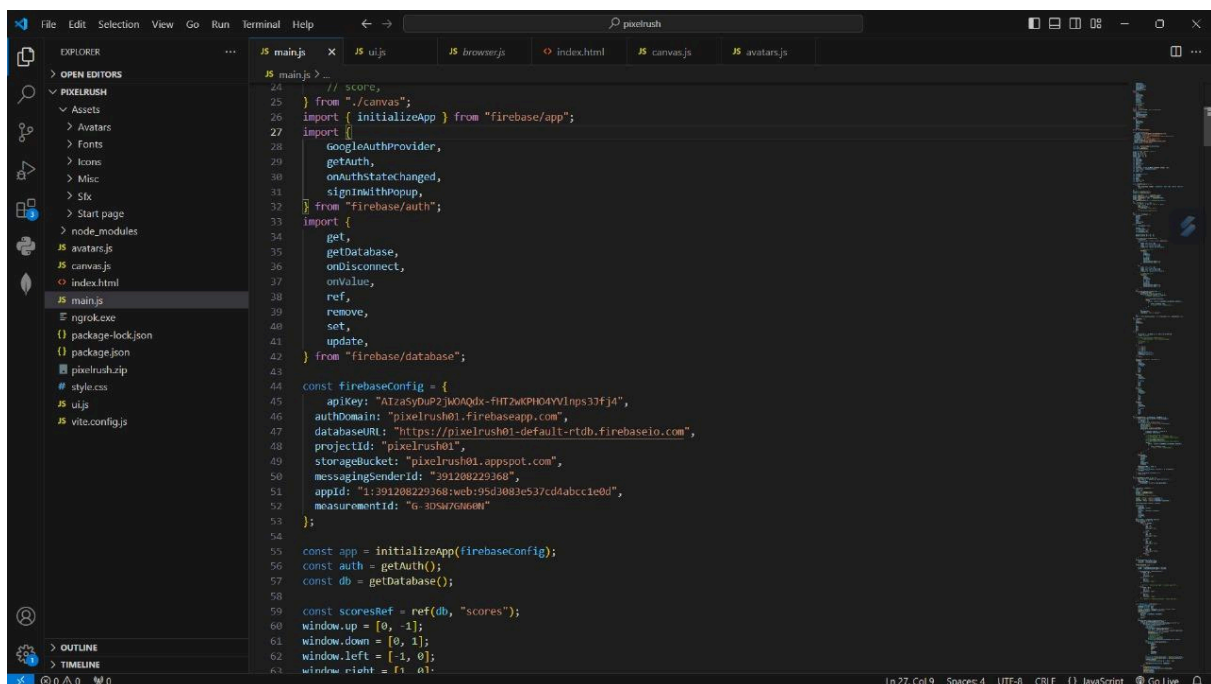
(index.html)

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a file tree for a project named 'PIXELRUSH'. The file 'index.html' is selected and open in the main editor. The code is an HTML document with a head section containing meta tags for charset, viewport, and Cross-Origin-Opener-Policy, and a title 'Pixel Rush!'. The body contains a link to a shortcut icon and a series of image tags for 'ninja' and 'gruce' avatars in various states (up, down, left, right, pattern).

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="user-scalable=no" />
6     <!-- <meta name="viewport" content="width=device-width, initial-scale=0" /> -->
7     <link rel="stylesheet" href="style.css" />
8     <meta name="Cross-Origin-Opener-Policy" content="none" />
9     <title>Pixel Rush!</title>
10    <link
11      rel="shortcut icon"
12      href="./Assets/Avatars/ninja/down.png"
13      type="image/x-icon"
14    />
15    <script type="module" src="/main.js" defer></script>
16  </head>
17  <body>
18    <div style="display: none">
19      
20      
21      
22      
23      
28      
29      
30      
31      
32      
37    </div>
38    
39  </body>
40</html>
```

## Data-Base and other back end codes:

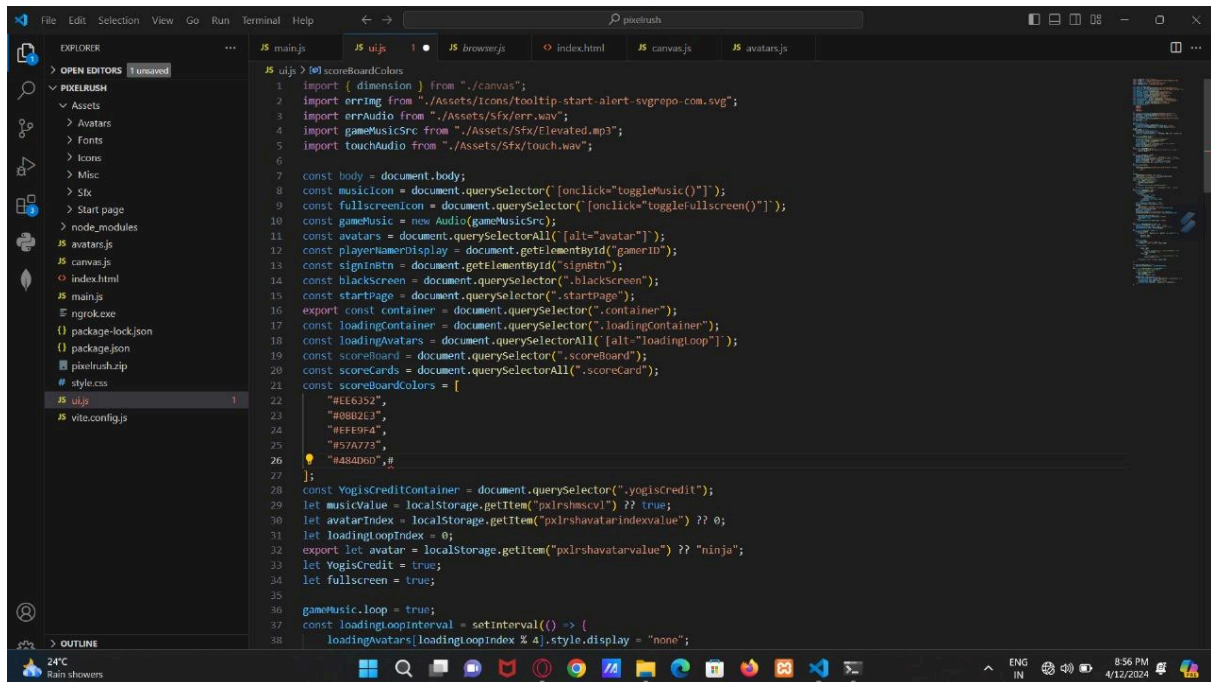
(Main.js)

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows the file tree. The file 'main.js' is selected and open in the main editor. The code is a JavaScript file that initializes Firebase, sets up authentication and database, and defines a 'scores' array.

```
1 // score,
2 } from "./canvas";
3 import { initializeApp } from "firebase/app";
4 import {
5   GoogleAuthProvider,
6   getAuth,
7   onAuthStateChanged,
8   signInWithPopup,
9 } from "firebase/auth";
10 import {
11   get,
12   getDatabase,
13   onDisconnect,
14   onValue,
15   ref,
16   remove,
17   set,
18   update,
19 } from "firebase/database";
20
21 const firebaseConfig = {
22   apiKey: "AIzaSyBuP2jW0AQdx-fHT2wKPH04VYlps33fj4",
23   authDomain: "pixelrush01.firebaseio.com",
24   databaseURL: "https://pixelrush01-default-rtdb.firebaseio.com",
25   projectId: "pixelrush01",
26   storageBucket: "pixelrush01.appspot.com",
27   messagingSenderId: "391288229368",
28   appId: "1:391288229368:web:95d3083e537cd4abcc1e0d",
29   measurementId: "G-3D5w7Gw6w8"
30 };
31
32 const app = initializeApp(firebaseConfig);
33 const auth = getAuth();
34 const db = getDatabase();
35
36 const scoresRef = ref(db, "scores");
37 window.up = [0, -1];
38 window.down = [0, 1];
39 window.left = [-1, 0];
40 window.right = [1, 0];
```

## Audio,video and animation:

(ui.js)



## Google SignIn:

```
} from "../canvas";
import { initializeApp } from "firebase/app";
import {
  GoogleAuthProvider,
  getAuth,
  onAuthStateChanged,
  signInWithPopup,

```

```
window.toggleMusic = () => toggleMusic(game);
window.toggleFullscreen = toggleFullscreen;
window.changeAvatar = (key) => changeAvatar(key);
window.signIn = () => {
  signInWithPopup(auth, new GoogleAuthProvider());
};
```

## Firestore:

The screenshot shows the 'Project settings' page for a Firebase project named 'pixelrush01'. The left sidebar contains navigation links for Project Overview, Realtime Database, Authentication, and various product categories like Build, Release & Monitor, Analytics, and Engage. The main content area is titled 'Project settings' and provides instructions on how to install the Firebase SDK using npm. It includes a terminal command and a sample JavaScript configuration file.

```
$ npm install firebase
```

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDuP2jW0AQdx-PHT2wKPH04YV1nps3Jf4",
  authDomain: "pixelrush01.firebaseio.com",
  databaseURL: "https://pixelrush01-default-rtdb.firebaseio.com",
  projectId: "pixelrush01",
  storageBucket: "pixelrush01.appspot.com",
  messagingSenderId: "391288229368",
  appId: "1:391288229368:web:95d3083e537cd4abcc1e8d",
  measurementId: "G-3DSW7GN68N"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

## Database:

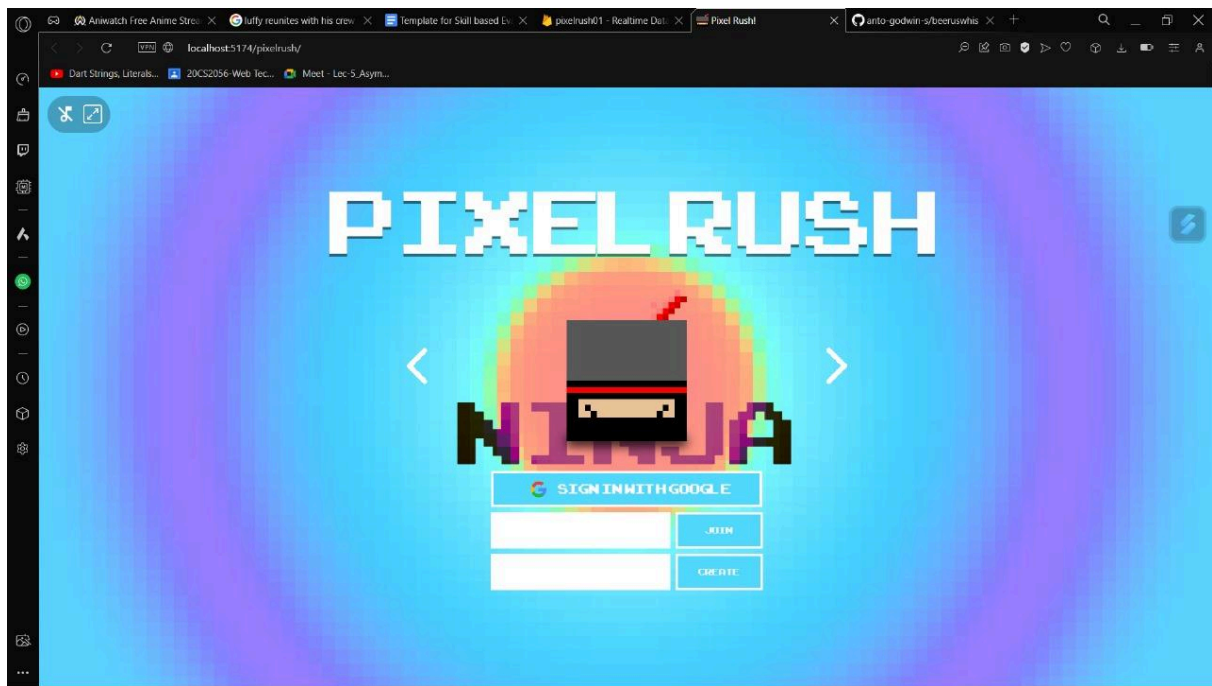
The screenshot shows the 'Authentication' page for a Firebase project named 'pixelrush01'. The left sidebar contains navigation links for Project Overview, Realtime Database, Authentication, and various product categories like Build, Release & Monitor, Analytics, and Engage. The main content area is titled 'Authentication' and shows a list of users. The table below contains the user data.

Identifier	Providers	Created	Signed In	User UID
bharathlax2005@gmail...	Google	Apr 12, 2024	Apr 12, 2024	vGQNoSnP63bQJCZHDUs29i0...
tamilselvamtamet@gm...	Google	Apr 2, 2024	Apr 2, 2024	lVP4ninkReNvaFKH658PbU28...
1216friendsforever@g...	Google	Apr 2, 2024	Apr 2, 2024	PAe08mE1gobFE6xc4zzV4hb...
jerlineam@karunya.edu...	Google	Apr 2, 2024	Apr 2, 2024	2YagovVmiEqRqZabRyyowdIB...
antogodwin@karunya.e...	Google	Apr 2, 2024	Apr 12, 2024	eKZLaBC21xMQHKtgGoC98U...

Rows per page: 50 1 - 5 of 5



Output:



## CONCLUSION

In conclusion, the development of "Pixel Rush" represents a significant undertaking, blending creativity, technical prowess, and a deep understanding of gaming mechanics. By harnessing the power of HTML, CSS, and JavaScript, we have laid the foundation for a visually captivating and responsive frontend interface. Through meticulous design and implementation, we aim to immerse players in a world where strategy meets artistry, echoing the appeal of the beloved "paint.io" game.

Moreover, the integration of Firebase as our backend solution offers a host of advantages. With its real-time database capabilities, we ensure that player actions are instantly reflected across all connected devices, fostering a truly dynamic and engaging multiplayer experience. Firebase's authentication features provide a secure environment for player profiles and interactions, while its hosting services guarantee reliable and scalable performance, even during peak usage periods.

As we navigate the development process, we are mindful of the challenges that lie ahead. From optimizing performance across different devices and browsers to fine-tuning gameplay mechanics for maximum enjoyment, our journey is marked by a commitment to excellence and continuous improvement. Through iterative testing and feedback loops, we endeavor to refine "Pixel Rush" into a polished and compelling gaming experience that captivates audiences worldwide.

Beyond the technical aspects, "Pixel Rush" embodies our team's passion for innovation and our dedication to pushing the boundaries of what's possible in the realm of web-based gaming. It serves as a testament to the collaborative spirit that drives us forward, as we unite our diverse talents and perspectives to realize a shared vision.

In summary, "Pixel Rush" is more than just a project—it's a testament to our creativity, perseverance, and unwavering commitment to excellence. As we embark on this journey, we are excited to see how our efforts will resonate with players and redefine the landscape of online gaming. With determination and ingenuity as our guiding principles, we are confident that "Pixel Rush" will leave a lasting impression and inspire future generations of game developers.



## EVALUATION SHEET

**Reg.No : URK22AI1094**

**Name: Jerlin Sam RN**

**Course code: 20CS2056**

**Course Name: Web Technology**

<b>S.No</b>	<b>Rubrics</b>	<b>Maximum Marks</b>	<b>Marks Obtained</b>
1	Industrial Certification	10	
2	Real – Time Application Design	30	
Total		40	

<b>Rubrics</b>	<b>Excellent</b>	<b>Good</b>	<b>Average</b>	<b>Below Average</b>
<b>Design of website</b>				
<b>Input validation</b>				
<b>Integration of front end and back end</b>				
<b>Presentation and Viva</b>				
<b>GitHub repository</b>				
<b>Report</b>				

**Signature of the Faculty-in-charge**