# San Francisco State University



Video Game Rating Application

Joystick Journal

SW Engineering CSC 648-848

Andrew Dahlstrom (Product Owner), Nadir Ali (Github Master), Aidan Bayer-Calvert (Back-end Lead), Ryan Flannery (Back-end Lead), Kayla Maa (Front-end Lead), Joyce Fu (Scrum Master), Jason Avina ( Scrum Master)

Milestone 1

10/02/2024

Section 01 Team 02

| History Table |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# Table of Contents

[**Cover Page 1**](#_z6ne0og04bp5)

[**Table of Contents 2**](#_35qsy46zviru)

[**Executive Summary (Ryan) 3**](#_au51mny0sx6)

[**Personas and User Stories(Nadir) 4**](#_3at9u9s4e0vp)

[**Data Definitions (Aidan) 5**](#_9hamacebn51u)

[**Initial List of Functional Requirements(Kayla) 6**](#_76q7xj2qal59)

[**List of Non-Functional Requirements(Jason) 8**](#_kltshzt30o6c)

[**Competitive Analysis(Andrew) 9**](#_e5zocfuhiekm)

[**High-Level System Requirements (Joyce) 10**](#_jfe37igesil)

[**Checklist 12**](#_pso4dkp34aou)

# 

# Executive Summary (Ryan)

The Joystick Journal is a next-generation video game rating and recommendation platform that empowers gamers to discover, review, and evaluate games with ease. As the gaming industry continues to grow exponentially across multiple genres and platforms, players are increasingly overwhelmed by the sheer number of options available. Joystick Journal solves this problem by helping users find video games that match their preferences, enhancing the gaming experience.

The platform offers several unique features designed to make the discovery process more engaging and interactive. Users can easily search for games based on genres, rate games using a multi-metric 5-star rating system, and write detailed reviews. Moreover, the application's standout feature is AI-powered personalized recommendations, which suggest games based on individual player preferences. Users can create profiles to bookmark their favorite titles, track their reviews, and explore trending games across genres. Joystick Journal differentiates itself from competitors like Metacritic and OpenCritic by offering not only these tailored recommendations but also the ability to maintain user accounts, making it a truly personalized platform for gamers.

Our dedicated team, consisting of students passionate about both gaming and technology, brings together expertise in machine learning, user experience design, and software engineering to build a product that addresses real-world gaming challenges. By focusing on community-driven reviews and recommendations, we believe Joystick Journal will become an indispensable tool for gamers seeking meaningful insights and an enjoyable gaming experience.

# Personas and User Stories (Nadir)

### **Persona 1: The Hardcore Gamer**

* **Behavior/Interests:** This user spends a lot of time and money on gaming, focusing on AAA franchise games and high-quality gaming experiences. They often seek the latest games and value top-tier equipment and immersive gameplay.
* **Skills:** Tech-savvy, well-informed about gaming trends, and enjoys experimenting with new games.
* **Pain Points:** Overwhelmed by the number of available games; wants to make informed purchases and ensure they are worth the investment.

**User Story 1:**As a Hardcore Gamer, I want to browse and filter games by their popularity and 5-star ratings so that I can quickly find top-rated games. I would also like to bookmark games I am interested in to revisit them later, and have access to relevant game details such as price, release date, and age rating.

**Priority:** High

### 

### **Persona 2: The Casual Gamer**

* **Behavior/Interests:** Plays occasionally, often offline, and prefers games with good narratives and graphics. They are not overly interested in competitive or multiplayer modes.
* **Skills:** Moderate gaming knowledge, not focused on multiplayer or fast-paced games.
* **Pain Points:** Has no interest in online multiplayer games and prefers to avoid content from users who haven't fully played the games they are reviewing.

**User Story 2:**As a Casual Gamer, I want to filter games to exclude online multiplayer ones and focus on offline story-driven games. I also want reviews from users who have completed the game to ensure I'm reading well-informed opinions. Additionally, I’d like to bookmark games that interest me for future consideration.

**Priority:** Medium

### **Persona 3: The Newcomer**

* **Behavior/Interests:** Recently entered the gaming world, unsure of where to start, and heavily relies on others' recommendations.
* **Skills:** Limited gaming knowledge, looking to explore different genres.
* **Pain Points:** Finds it difficult to decide what to play due to the overwhelming number of choices; values simple, trusted recommendations.

**User Story 3:**As a Newcomer, I want to browse the most popular games by genre so I can easily discover games suited to my interests. Additionally, I would like to filter games by difficulty level to find games that match my skill level. The design and user interface should be intuitive and appealing to build my trust in the reviews and recommendations.

**Priority:** High

### 

### **Persona 4: The Experienced Reviewer**

* **Behavior/Interests:** Enjoys critiquing and sharing detailed insights about games; values well-crafted reviews and the ability to engage with other gamers.
* **Skills:** High level of expertise in various genres, frequently writes game reviews.
* **Pain Points:** Wants more control over their reviews and a system to track and update them.

**User Story 4:**As an Experienced Reviewer, I want to edit my reviews and view all my past reviews so I can maintain a consistent review history. I also want a seamless and enjoyable user experience while leaving reviews to encourage me to engage regularly with the platform.

**Priority:** Medium

# Data Definitions (Aidan)

**User**

* **Definition:** The individual who interacts with the application. Users can register, log in, and post reviews and ratings for games. Unauthenticated users can view reviews but cannot post them.
* **Attributes:**
  + **user\_id**: (Primary Key, INT, AUTO\_INCREMENT) – A unique identifier assigned to each user.
  + **username**: (VARCHAR) – The user’s chosen display name within the application.
  + **email**: (VARCHAR) – The user’s email address, used for authentication and communication.
  + **password\_hash**: (VARCHAR) – The encrypted version of the user’s password for secure storage.
  + **profile\_picture**: (VARCHAR, optional) – The URL to the user’s profile image, if uploaded.
  + **bio**: (TEXT, optional) – A brief personal description or biography provided by the user.
  + **User preferences**: (One-to-One) – A relationship to the user’s preferences, allowing personalized recommendations.
  + **created\_at**: (TIMESTAMP) – The date and time when the user account was created.
  + **updated\_at**: (TIMESTAMP) – The date and time of the user’s latest account update.

**Game**

* **Definition:** The entity representing a video game that can be reviewed or rated by users.
* **Attributes:**
  + **game\_id**: (Primary Key, INT, AUTO\_INCREMENT) – A unique identifier for each game.
  + **title**: (VARCHAR) – The title or name of the video game.
  + **description**: (TEXT) – A summary or description of the game’s content and features.
  + **genre**: (VARCHAR) – The genre the game belongs to (e.g., RPG, Action, Sports).
  + **reviews**: (LIST of review\_ids) – A list of all the review IDs associated with the game.
  + **release\_date**: (DATE) – The date the game was officially released.
  + **cover\_image**: (VARCHAR, optional) – The URL of the game’s cover image.
  + **created\_at**: (TIMESTAMP) – The date and time when the game was added to the system.
  + **updated\_at**: (TIMESTAMP) – The date and time of the latest update to the game’s data.

**Review**

* **Definition:** Represents the feedback given by a user for a particular game, including both rating and text feedback.
* **Attributes:**
  + **review\_id**: (Primary Key, INT, AUTO\_INCREMENT) – A unique identifier for each review.
  + **user\_id**: (Foreign Key, INT) – The ID of the user who submitted the review.
  + **game\_id**: (Foreign Key, INT) – The ID of the game being reviewed, linking reviews to games.
  + **rating**: (INT) – The user’s rating for the game on a scale of 1-5.
  + **review\_text**: (TEXT, optional) – A textual description or feedback on the game provided by the user.
  + **created\_at**: (TIMESTAMP) – The date and time when the review was created.
  + **updated\_at**: (TIMESTAMP) – The date and time when the review was last updated.

**Preference**

* **Definition:** Captures a user’s preferences for game recommendations, based on genres and keywords.
* **Attributes:**
  + **preference\_id**: (Primary Key, INT, AUTO\_INCREMENT) – A unique identifier for each set of preferences.
  + **user\_id**: (Foreign Key, INT) – The ID of the user whose preferences are being stored.
  + **genre**: (VARCHAR) – The user’s preferred game genre for recommendations.
  + **keywords**: (LIST of strings) – A list of keywords representing specific user preferences (e.g., "open-world", "multiplayer", "story-driven").

# Initial List of Functional Requirements (Kayla)

#### **FR-1001: User Authentication**

* **Description:** Users will be able to log in using their username and password to access personalized features. If credentials are incorrect, the user will be prompted to retry or reset their password.
* **Priority:** High
* **User Story Reference:** As a user (**Hardcore Gamer**, **Casual Gamer**, **Newcomer**, **Experienced Reviewer**), I want to log in with my username and password to access my account. This helps keep my information secure and gives me access to personalized features.

#### **FR-1102: Search for Games**

* **Description:** Users can search for video games by title. Search results will display the game’s cover image, rating, and a link to reviews.
* **Priority:** High
* **User Story Reference:** As a gamer (**Hardcore Gamer**, **Casual Gamer**, **Newcomer**), I want to search for a video game that I am interested in and read reviews and feedback from other gamers to help decide if it's worth playing.

#### **FR-1103: Game Review**

* **Description:** Users can leave a review on any video game in the system, including a written review and rating. The system should allow users to edit their reviews at a later time.
* **Priority:** High
* **User Story Reference:** As a user (**Hardcore Gamer**, **Experienced Reviewer**), I want to leave and edit reviews on video games so I can share my thoughts and help others decide whether the game is worth playing.

#### **FR-1104: Game Rating**

* **Description:** Users can rate video games on a scale of 1 to 5, with the option to rate games across multiple metrics (e.g., gameplay, story, graphics).
* **Priority:** High
* **User Story Reference:** As a user (**Hardcore Gamer**, **Casual Gamer**, **Newcomer**, **Experienced Reviewer**), I want to rate video games on a 5-point scale and by multiple metrics so that I can express my opinion about different aspects of the game and provide guidance for those considering playing it.

#### **FR-1105: Browse for Video Games**

* **Description:** Users can browse for video games by genre and category, such as RPG, action, or sports. Results will show the game title, cover image, and average user rating.
* **Priority:** High
* **User Story Reference:** As a user (**Casual Gamer**, **Newcomer**, **Hardcore Gamer**), I want to browse video games by category so I can quickly discover games that match my interests or explore new ones.

#### **FR-1106: User Profile Page**

* **Description:** Users can view and edit their profile information, such as username, email, and bio. The page will also display a list of their submitted reviews.
* **Priority:** Low
* **User Story Reference:** As a user (**Experienced Reviewer**, **Hardcore Gamer**), I want to edit my profile and view my submitted reviews so I can track my contributions and manage my personal information.

#### **FR-1107: AI Game Recommendation**

* **Description:** The system will recommend video games to users based on their past searches, reviews, and ratings. These suggestions will consider user preferences for genres, keywords, and previously rated games.
* **Priority:** Medium
* **User Story Reference:** As a user (**Hardcore Gamer**, **Casual Gamer**, **Newcomer**), I want the system to recommend games based on my past searches and reviews so I can easily find games that suit my preferences.

# List of Non-Functional Requirements (Jason)

1. **Performance**

The following represent average thresholds considered "smooth" or "good" user experiences:

* The application should have a response time of under 200 milliseconds for API calls.
* Frontend pages should load within 1 sec under normal network conditions due to SSR(Astro)
* The application must efficiently handle up to 1,000 concurrent users.

1. **Storage Space**

* The backend should utilize SQL with a maximum database size limit of 50 GB.
* The application should optimize data storage through proper indexing and data normalization.

1. **Storage Availability**

* The application must ensure a minimum uptime of 99.9% for the database and API endpoints.
* Utilize AWS R2 for object storage with redundancy and backups to prevent data loss.

1. **Usability**

* The application should be user-friendly with a clean, intuitive interface.
* The front end must support multiple input methods, including voice and text input, ensuring accessibility for users with disabilities.
* Provide clear error messages and instructions for user

1. **Security**

* Implement user authentication using secure methods (e.g., JWT or OAuth).
* Ensure data is encrypted in transit (HTTPS) and at rest (for sensitive data in the database).
* Regularly update dependencies to address vulnerabilities.

1. **Fault Tolerance**

* Implement error handling to gracefully manage exceptions without crashing the application.
* The system should recover from failures automatically and continue functioning.

1. **Compatibility With Browsers**

* The front end should be compatible with the latest versions of major browsers (Chrome, Firefox, Safari, Edge).
* Install Tailwind later to ensure the application works well on both desktop and mobile devices.

1. **Usability(voice, text input, etc)**

* Implement voice input functionality that is accurate and responsive for supported browsers.
* Ensure text input fields are intuitive and support auto-suggestions and validations.

1. **Github Management**

* Use GitHub for version control, with a branching strategy defined (e.g., feature branches, main branch).
* Establish code review processes to maintain code quality before merging to the main branch.
* Ensure continuous integration/continuous deployment (CI/CD) practices are in place to automatically deploy changes to Vercel for the front end and EC2 for the back end.

1. **Hosting and Deployment**

* The frontend should be hosted on Vercel with automatic deployments on every merge to the main branch.
* The backend should be hosted on AWS EC2 with proper security groups and instance types chosen for performance.
* The database should be accessible only from the backend server to ensure security.

# Competitive Analysis (Andrew)

| **Feature** | **Metacritic** | **OpenCritic** | **Pocket Gamer** | **Joystick Journal** |
| --- | --- | --- | --- | --- |
| User Account Creation / Log-in | No | No | Yes | Yes |
| Video Game Rating (Multiple Metrics) | Yes | Yes | Yes | Yes |
| User Text Reviews | Yes | Yes | Yes | Yes |
| Genre-based Search | Yes | Yes | Yes | Yes |
| Personalized Recommendations | No | No | No | Yes |
| Social Sharing of Reviews | Yes | No | No | No |

**Comparison Summary**

Our planned app provides key features like user account creation/log-in, which some competitors lack, allowing users to have a more personalized experience by saving their reviews and ratings. Additionally, while many competitors focus on general video game reviews and ratings, our app will support multiple rating metrics, providing users with a more detailed review experience. Moreover, our app will also include personalized recommendations based on past user ratings and reviews, a feature missing in the existing competitors. This combination of personalization and detailed user engagement gives our app an edge over existing platforms like Metacritic and OpenCritic.

# High-Level System Requirements (Joyce)

1. Frameworks and Libraries

**Frontend:**

* **React**: Main library used for building the user interface and managing component-based architecture.
* **ShadCN**: Component library for providing UI components with pre-styled elements.
* **Tailwind CSS**: Utility-first CSS framework for fast and flexible styling.

**Backend:**

* **Node.js**: JavaScript runtime environment for server-side development.
* **Express.js**: Web application framework for creating RESTful APIs and routing functionalities.
* **Python**: Utilized for additional backend functionalities, including scripting and machine learning integration (future enhancement).

**API Integration:**

* **IGDB API**: Used for fetching video game data such as genre, release date, and developer information.
* **Amazon API (Possibly)**: Planned for integrating additional data services, such as game details from Amazon.

1. Development and Deployment Tools

**Version Control:**

* **Git**: Version control system used for managing code changes and team collaboration.
* **GitHub**: Hosting service for code repositories, enabling team members to track changes, review pull requests, and manage branches.

**Integrated Development Environment (IDE):**

* **Visual Studio Code (VS Code)**: Primary IDE used for writing, debugging, and testing both the frontend and backend code.

1. Deployment Platform

**Server Host:**

* **AWS EC2**: Amazon Web Services EC2 instance for hosting the backend server. The application code will run on this cloud server, handling requests and responses.

**Operating System:**

* **Amazon Linux 2**: Operating system for the AWS EC2 instance, providing a secure and stable environment for running the backend.

**Web Server:**

* **Vercel**: Platform used to deploy the frontend (React) application, making the web application accessible to users via a URL.

4. Database Management

**Database:**

* **AWS RDS (MySQL)**: Managed relational database service, handling the storage and management of all user data, reviews, ratings, and game information. MySQL is chosen for its reliability and scalability.

**Database Details:**

* **Database Name:** ratingsdb
* **Host:** ratingsdb.c10o6s8wguyo.us-west-2.rds.amazonaws.com
* **User:** ratingsuser
* **Password:** ratingspass

**Command to Connect:**

mysql -h ratingsdb.c10o6s8wguyo.us-west-2.rds.amazonaws.com -u ratingsuser -p

5. Supported Browsers

The application will be accessible via the following browsers:

* **Google Chrome**
* **Apple Safari**
* **Microsoft Edge**

6. Credentials and Access Information

**Website URL:**

* The frontend of the application will be accessible at:  
  https://csc648-01-fa24-team02.vercel.app/

**Database Table:**

* **Table: users** (contains columns userid, name, password for managing user login and authentication).

# 

# Team Study Guide

Front-End Developers (React, Tailwind CSS, ShadCN): Kayla, Jason, Nadir, Joyce

#### **Week 1: Introduction to React and Basic Components**

* **Objectives**:
  1. Learn the basics of React and how to build components.
  2. Understand JSX, props, and state.
* **Key Topics**:
  1. React Components, JSX, Props, State.
  2. Functional components and hooks.
* **Resources**:
  1. **React Official Tutorial**: This covers the basics of React, component creation, and state management.
  2. **React Docs - Main Concepts**: Explore React's core concepts with examples.

#### **Week 2: Routing and Front-End Design (React Router, CSS/Tailwind)**

* **Objectives**:
  1. Implement navigation with React Router.
  2. Style components using CSS and Tailwind.
* **Key Topics**:
  1. React Router for handling routes (Login, Profile, Game List, Game Details).
  2. Using Tailwind CSS for responsive design and utility-first CSS.
* **Resources**:
  1. **React Router Tutorial**: Learn how to implement routing in a React app.
  2. [**Tailwind CSS Crash Course**](https://www.youtube.com/watch?v=UBOj6rqRUME): A full crash course on using Tailwind CSS to style your React app.

#### **Week 3: Form Handling and State Management (User Registration, Login, Reviews)**

* **Objectives**:
  1. Handle forms in React for user input (registration, login, reviews).
  2. Understand how to manage state and interact with back-end APIs.
* **Key Topics**:
  1. Controlled forms and input handling.
  2. Using useState and useEffect for managing form data.
* **Resources**:
  1. **Handling Forms in React**: Learn how to handle forms in React, including controlled and uncontrolled components.
  2. **React State Management Tutorial**: A guide to managing state in React applications.

#### **Week 4: API Integration and Data Fetching**

* **Objectives**:
  1. Learn how to fetch data from the back-end API and display it in the front-end.
  2. Work on displaying the list of video games and user reviews.
* **Key Topics**:
  1. Using fetch or Axios to get data from the backend.
  2. Handling API responses and rendering data in components.
* **Resources**:
  1. **React API Calls with Fetch**: This guide covers making API calls in React using fetch.
  2. **Axios HTTP Client**: A comprehensive guide to making HTTP requests with Axios.

Back-End Developers (Node.js, Express, Python, MySQL): Ryan, Aidan, Andrew

#### **Week 1: Introduction to Node.js and Express.js**

* **Objectives**:
  1. Get familiar with Node.js and Express.js.
  2. Learn how to set up a basic Express server.
* **Key Topics**:
  1. Node.js setup, creating a basic Express server.
  2. Handling routes and middleware in Express.
* **Resources**:
  1. **Node.js Official Docs**: A basic guide to understanding and setting up Node.js.
  2. [**Express.js Crash Course**](https://www.youtube.com/watch?v=L72fhGm1tfE): A YouTube tutorial covering Express.js setup and routing.

#### **Week 2: Database Design and MySQL Integration**

* **Objectives**:
  1. Learn how to set up a MySQL database and interact with it using Express.js.
  2. Understand basic database design for user profiles, games, reviews, and ratings.
* **Key Topics**:
  1. MySQL setup, connecting Node.js with MySQL.
  2. Writing queries for user data, games, and reviews.
* **Resources**:
  1. [**MySQL Crash Course**](https://www.youtube.com/watch?v=7S_tz1z_5bA): A YouTube tutorial on MySQL basics and connecting to Node.js.
  2. **Node.js and MySQL Tutorial**: Learn how to perform CRUD operations in Node.js with MySQL.

#### **Week 3: Authentication and JWT**

* **Objectives**:
  1. Implement user registration and login with secure authentication.
  2. Learn how to use JWT (JSON Web Token) for session management.
* **Key Topics**:
  1. User registration, password hashing (bcrypt), and JWT for authentication.
  2. Securing routes with authentication middleware.
* **Resources**:
  1. **JWT Authentication in Node.js**: A guide on implementing JWT-based authentication with Express.js.
  2. [**Bcrypt for Password Hashing**](https://www.npmjs.com/package/bcrypt): Official documentation for using bcrypt to hash passwords.

#### **Week 4: API Development and Testing**

* **Objectives**:
  1. Develop RESTful APIs to handle user profiles, game data, reviews, and ratings.
  2. Write test cases to validate the APIs.
* **Key Topics**:
  1. Building REST APIs in Express.js and handling requests/responses.
  2. Writing unit and integration tests for the APIs.
* **Resources**:
  1. [**RESTful API with Express**](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes): A tutorial on building a RESTful API with Express.
  2. **Jest and Supertest for API Testing**: Learn how to write tests for Node.js APIs using Jest and Supertest.

# Checklist

| Checklist | Status (DONE, ON TRACK, ISSUE) |
| --- | --- |
| Team found a time slot to meet outside of class. | DONE |
| Scrum Master shares meeting minutes with everyone after each meeting. | DONE |
| Github Master chosen. | DONE |
| Everyone sets up their local development environment from the team’s git repo. | DONE |
| Team decided and agreed together on using the listed SW tools and deployment server. | DONE |
| Team ready and able to use the chosen back/front-end frameworks | DONE |
| Team lead ensured that all team members read the final M1 and agree/understand it before submission. | DONE |