**A PROJECT REPORT**

**on**

**Games Sonnet**

***Submitted by***

**Mr. Vikas Naresh Makwana**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF SCIENCE**

**in**

**COMPUTER SCIENCE**

***under the guidance of***

**Prof. Mr. Vipul Saluja,**

**Department of Computer Science**


**R. D. & S.H. National College & S. W. A. Science College**

**Bandra, Mumbai – 400050.**

**(Sem V)**

**(2025 – 2026)**

# R.D. & S.H. NATIONAL COLLEGE & S. W.A. SCIENCE COLLEGE,

**Bandra, Mumbai – 400050.**

**Department of Computer Science**

## PROJECT CERTIFICATE

This is to certify that **Mr.Vikas Naresh Makwana** of **T.Y.B.Sc. (Sem V)** class has satisfactorily completed the Project entitled **Games Sonnet** to be submitted in the partial fulfillment for the award of **Bachelor of Science** in **Computer Science (Semester V)** during the academic year **2025 – 2026**.

It is further certified that the project is an original work and it has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

**Date of Submission:**

**Internal Project Guide**

**Co-ordinator,**
**Department of Computer Science**

**Signature of External Examiner**

**This project is dedicated to Prof. Mr. Vipul Saluja, Teacher Madhavi Tullu, Sir Avinash Kauran, and Teacher Sufiya Ahmed.**

**“A bad website is like a grumpy salesperson.”**

***— Jakob Nielsen*,**

# Acknowledgement

I would like to take this opportunity to acknowledge the dedicated efforts I put into this project. However, the successful completion of this endeavor would not have been possible without the unwavering support and assistance of numerous individuals. I extend my sincere gratitude to all those who played a significant role in this journey.

I am deeply thankful to the Coordinator of the Computer Science Department and our esteemed Project Guide, **Prof. Vipul Saluja**, for their continuous support and invaluable suggestions, which were instrumental in the project's success.

I also wish to express my heartfelt thanks to my friends, whose encouragement and moral support have been a constant source of motivation throughout this process.

Furthermore, I extend my gratitude to all other faculty members and non-teaching staff within our department for their continued support and cooperation.

Last but not least, I would like to acknowledge my peers, who stood by me during the challenges and triumphs of completing this project.

# INDEX

# Preliminary Investigation

## i)Synopsis

**Title:**

Community-Driven Game Recommendation Web Application

**Introduction:**

This project is a web-based platform that allows users to suggest, review, and rate video games, offering a community-oriented space for discovering new games. It aims to solve the common problem of finding quality game recommendations by leveraging peer feedback and interactive features.

**Objectives**

- Apply full-stack development skills to build an interactive platform for sharing game recommendations.
- Gain practical experience in the software development life cycle, including frontend design, backend integration, database management, and authentication.
- Develop a scalable system for real-time data storage and dynamic content updates.
- Practice effective user data handling through secure login and authentication.
- Lay the groundwork for future enhancements, such as auto-generated summaries from user reviews.

**Scope**

- Suggest games by searching or adding new entries.
- Submit and read reviews, rate games (1 to 5 stars), and view top games of the week.
- Use MongoDB for scalable data storage and efficient querying, with support for real-time updates via backend integration.

**Methodology**

1. Requirement Analysis
2. Design
3. Implementation
4. Testing & Debugging
5. Deployment

**Tools and Technologies**

- **Frontend:** React.js
- **Backend:** Node.js with Express.js
- **Database:** MongoDB
- **Authentication:** Custom Auth with bcrypt for password hashing
- **Other Tools:** Git, Visual Studio Code

**Timeline**

- **Week 1–2:** Requirement Analysis & Design
- **Week 3–6:** Implementation (Frontend & Backend)
- **Week 7–8:** Integration & Testing
- **Week 9–10:** Finalization & Documentation

**Resources**

- **Web Hosting Platform:**
    - Netlify (Frontend)
    - Render (Backend)
- **Version Control:** GitHub

**Expected Outcomes**

- A functional and user-friendly game suggestion website where users can discover trending games (from RAWG API) and community-suggested games (from MongoDB Atlas).
- Ability for users to add games even without logging in (saved as anonymous), while logged-in users have their contributions linked to their account.
- Real-time updates: newly added games and reviews instantly appear in the game feed without requiring a page reload.
- Secure authentication system using bcrypt-encrypted passwords and MongoDB Atlas for safe storage.
- Review & rating system where users can submit feedback on games, while guests can still browse and read reviews without login.
- Error handling and fallback pages (e.g., custom 404 page) for better user experience.
- Clear project documentation, including UML diagrams (Activity, Sequence, Collaboration, Statechart, ERD), development process (V-Model), and plans for future upgrades such as:
    - Genre-based filtering & search
    - Collaborative filtering / recommendation engine
    - Advanced community features (e.g., leaderboards, follow system).

**Project Type:**

**Team Size:**

1 member

**Guide's Role**

The assigned guide will provide technical and strategic guidance throughout the project lifecycle, ensuring alignment with academic and industry standards.

## (ii) Organizational Overview

- **Project Title**: *GameSonnet – Community-Driven Game Suggestion and Review Platform*
- **Objective**:
  The system aims to provide a platform where users can explore trending games (from the RAWG API) and suggested games (from MongoDB), contribute new game suggestions, and share reviews. The platform ensures authenticity through an admin approval process before games are added to the official database.
- **Key Stakeholders**:
  1. **End Users (Registered & Guest)**
     - Browse trending and suggested games.
     - View game details and reviews.
     - Registered users can add games, submit reviews, and track their contributions.
  2. **Admin**
     - Manages pending game submissions.
     - Approves, rejects, or edits suggested games.
     - Ensures database reliability by preventing duplicate or invalid entries.
  3. **System**
     - Connects with RAWG API for trending games and game details.
     - Maintains secure database storage in MongoDB for users, games, reviews, and pending suggestions.
     - Enforces authentication and authorization using JWT.

- **Organizational Structure**:
    - **Frontend (React + Vite)**
        - Home Page (Trending + Suggested Games)
        - Add Yours Page (Game suggestions, automatic or manual entry)
        - Your Suggestions Page (Track user contributions)
        - Game Details Page (Description, rating, reviews, external link)
        - Admin Panel (Pending games approval, editing, rejecting)
        - Authentication Pages (Login, Signup)
        - About Page (Website/project information, contact email)
    - **Backend (Express.js + Node.js)**
        - API routes for authentication, game management, review handling, and admin approvals.
        - Integration with RAWG API for game data fetching.
        - Business logic for duplicate entry checking and game approval workflow.
    - **Database (MongoDB Atlas)**
        - Users Collection: Stores user credentials and roles (admin or regular).
        - Games Collection: Approved games with contributors (addedBy).
        - PendingGames Collection: User-submitted games awaiting admin approval.
        - Reviews Collection: Reviews linked to specific games.
- **Working Principle**:
    1. User visits the site and explores trending games (via RAWG API) and suggested games (from MongoDB).
    2. If logged in, the user can suggest a game by entering its name.
        - If found in RAWG API → sent to admin for approval.
        - If not found → manual entry enabled → sent to admin.
    3. Admin approves, rejects, or edits suggestions. Approved games move to the Games collection.
    4. Users can review games, contributing ratings and feedback visible to all.
- **Outcome**:
  A structured, community-driven platform similar to IMDb (but for games), balancing **user collaboration** with **admin moderation** for authenticity and reliability.

## iii) Working Principle

**1. User Visits the Website**

- **Frontend:** Home page loads with navigation bar, trending games section, and suggested games section.
- **Backend:**
    - Trending games → fetched live from **RAWG API**.
    - Suggested games → fetched from **MongoDB (Games collection)**.
- **Database:** Game collection stores approved games that appear in the suggested section.

**2. Browsing Games**

- **Guest User:**
    - Can browse trending and suggested games.
    - Can click on a game card to open the **big card view**, which shows: description, platforms, genres, reviews, and a redirect to the game's website.
    - Can read reviews but **cannot submit reviews**.
- **Logged-In User:**
    - Has all the above access.
    - Additionally can **submit reviews** (rating + comment).
- **Backend:**
    - Fetches reviews from the Review collection.
    - If user submits review → stores it in Review collection.

**3. Suggesting a Game ("Add Yours")**

- **Frontend:**
    - User clicks **Add Yours** in the navbar → redirected to add game page.
    - User enters the **game name**.
- **Backend Logic:**

1)Check in **RAWG API** using the entered name.

2)If found → fetches details → sends to **PendingGames collection** with status = "pending".

3)If not found → manual submission form opens → user enters details (slug, name, description, image, genres, platforms, rating, release date, website).

**i)** Submitted details → saved into **PendingGames collection** with status = "pending".

- **Database:** PendingGame collection stores the submission.

**4. Admin Moderation**

- **Frontend:** Admin logs in → accesses **Admin Panel (/admin)**.
- **Backend:** Admin can:
    - **Approve Game** → moves entry from PendingGame to Game collection (status removed).
    - **Reject Game** → updates status = "rejected" in PendingGame.
    - **Edit Game** → modifies details before approval.
- **Database:**
    - Approved games go into Game collection.
    - Rejected games stay in PendingGame with rejection notes.

**5. User Contributions Tracking**

- **If multiple users suggest the same game:**
    - Their usernames are added to the **addedBy[] field** in the game's record.
    - This ensures collaboration and prevents duplicates.

**6. Reviews System**

- **Logged-in User:** Can submit reviews.
- **Guest User:** Can only view reviews.
- **Backend:** Saves reviews in Review collection linked to the game's ID.
- **Database:** Reviews are retrievable via gameId.

## (iv) Limitations of the Current System

- **Manual Deletion by Admin**: Currently, if a game or review needs to be removed, the admin must manually locate and delete the corresponding document directly from the MongoDB database. This process is time-consuming, error-prone, and lacks a user-friendly interface for moderation**.**
- **Lack of In-App Reporting Mechanism**: Users do not have the ability to report inappropriate or misleading games or reviews directly through the website. Instead, they must send an email to the admin, which delays response time and reduces the efficiency of content moderatio**n.**

## (v) The Proposed System

The proposed system, **GameSonnet**, addresses the limitations of existing platforms (RAWG, Steam, Epic Games, IMDb) by providing a hybrid model where both **API-based data** and **community-driven contributions** are supported with admin moderation.

Key features of the proposed system include:

1. **Admin Panel with Approval Workflow**
    - All user-submitted games are stored in the **PendingGames collection**.

- Admins review, approve, reject, or edit submissions before moving them to the **Games collection**.
- Ensures authenticity, accuracy, and prevents fake entries.

2. **Duplicate Entry Management**
   - Before submission, the backend checks for duplicates:
     - If game exists in **Games collection**, contributor is added to the addedBy field.
     - If game is in **PendingGames**, no duplicate is created.
   - Prevents redundant entries and reduces admin workload.
3. **Manual Submission of Games**
   - If a game is not found in the RAWG API, users can submit details manually.
   - Admin validates and approves before it becomes part of the official catalog.
4. **User Contributions & Collaboration**
   - Multiple contributors can be linked to the same game through the addedBy field.
   - Community-driven approach shows collective efforts in building the catalog.
5. **Reviews System**
   - Logged-in users can submit reviews with ratings.
   - Guests can only view reviews.
   - Reviews are stored in the **Reviews collection**, linked by game ID.
6. **Home Page Features**
   - Trending games (fetched from RAWG API).
   - Suggested games (approved entries from MongoDB).
   - Game details view with description, platforms, genres, redirect link, and reviews section.

**(vi) Advantages of the Proposed System**

1. **User Contribution Enabled**
   - Unlike existing platforms, users can directly suggest games, making the database more community-driven and up-to-date.
2. **Admin Moderation Improves Reliability**
   - Admin approval ensures fake or incorrect data does not enter the main database.
3. **Supports Both API & Manual Data Entry**
   - Hybrid system allows both automatic fetching from RAWG API and manual entry by users when games are missing.
4. **Duplicate Management**

- Backend checks prevent redundancy and ensure clean data storage.

5. **Collaborative Contribution**
    - Multiple users can be linked as contributors to the same game, improving transparency and credit distribution.
6. **Enhanced User Experience**
    - Clear navigation with **Home, Add Yours, Your Suggestions, Admin Panel, About Page**.
    - Reviews and ratings enrich the platform with user opinions.
7. **Scalable & Cost-Effective**
    - Built using **open-source tools** (MERN stack).
    - Free hosting tiers (Render, Netlify) make it economically feasible.
8. **Improved Data Authenticity**
    - Admin validation and rejection notes keep the system transparent and trustworthy.

**(vii) Tools and Technologies to be Used**

**Frontend (UI Layer):**

- **React.js** → for building dynamic and responsive components (Home, Add Yours, Admin Panel, About, Game Cards, Reviews).
- **CSS** → for styling and responsiveness.
- **Axios** → for making API calls to backend and RAWG API.

**Backend (API Layer):**

- **Node.js + Express.js** → REST API handling requests like login, add game, approve/reject, fetch reviews.
- **bcrypt.js** → for password hashing.
- **jsonwebtoken (JWT)** → for secure authentication and role-based access (user/admin).

**Database (Storage Layer):**

- **MongoDB (Atlas)** with following schemas:
    - **Users** → stores user details (username, email, password, isAdmin).
    - **Games** → stores approved games.
    - **PendingGames** → stores games awaiting admin review.
    - **Reviews** → stores user reviews linked by game ID.

**External API Integration:**

- **RAWG API** → fetches trending and detailed game data.

**Hosting & Deployment:**

- **Netlify** → Frontend deployment.
- **Render** → Backend deployment.
- **MongoDB Atlas** → Cloud-based database hosting.

**Version Control & Collaboration:**

- **GitHub** → for source code management.

## viii) Feasibility Study

The feasibility of the system is evaluated across three major dimensions:

**a) Operational Feasibility**

The system is designed to be intuitive and user-friendly, offering:

- A **Home Page** showcasing trending and suggested games.
- An **"Add Yours"** section for users to suggest new games.
- An **Admin Panel** for reviewing and approving submissions.
- A **Game Details Page** displaying reviews and ratings.

Additional operational strengths include:

- **Clear navigation** via a responsive Navbar with distinct User and Admin functionalities.
- **Duplicate check mechanisms** to maintain data integrity and avoid redundancy.
- **Admin workload reduction** through automated filtering of repeated or invalid requests.
- **Support for both automatic API fetching and manual submissions**, making the system flexible and reliable.

**b) Technical Feasibility**

The system is built using modern, scalable technologies:

- **MERN Stack**: MongoDB, Express.js, React.js, and Node.js..
- **JWT-based authentication** ensures secure login and role verification for users and admins.
- **MongoDB** efficiently manages multiple schemas:
    - Users
    - Games
    - PendingGames
    - Reviews
- **Duplicate checks** are implemented using optimized findOne() queries, minimizing overhead by validating entries before insertion.

- **RAWG API integration** enables real-time access to trending game data.
- **Deployment-ready architecture**, compatible with platforms like:
    - **Render** for backend services
    - **Netlify** for frontend hosting

**c) Economic Feasibility**

The system is highly cost-effective due to its use of free and open-source tools:

- **Technology Stack**:
    - React, Node.js, Express, and MongoDB—all open-source.
- **API Usage**:
    - RAWG API offers free access within reasonable usage limits.
- **Hosting**:
    - Netlify and Render provide generous free tiers suitable for MVP and early-stage deployment.
- **Community Contribution Model**:
    - Reduces admin overhead by allowing users to collaboratively enrich the database, improving scalability without increasing operational costs.

# System Analysis

## i)Event Table

Represents all major events (login, game suggestion, admin approval, reviews) with triggers and system responses.
It helps track how GameSonnet reacts to user and admin interactions.

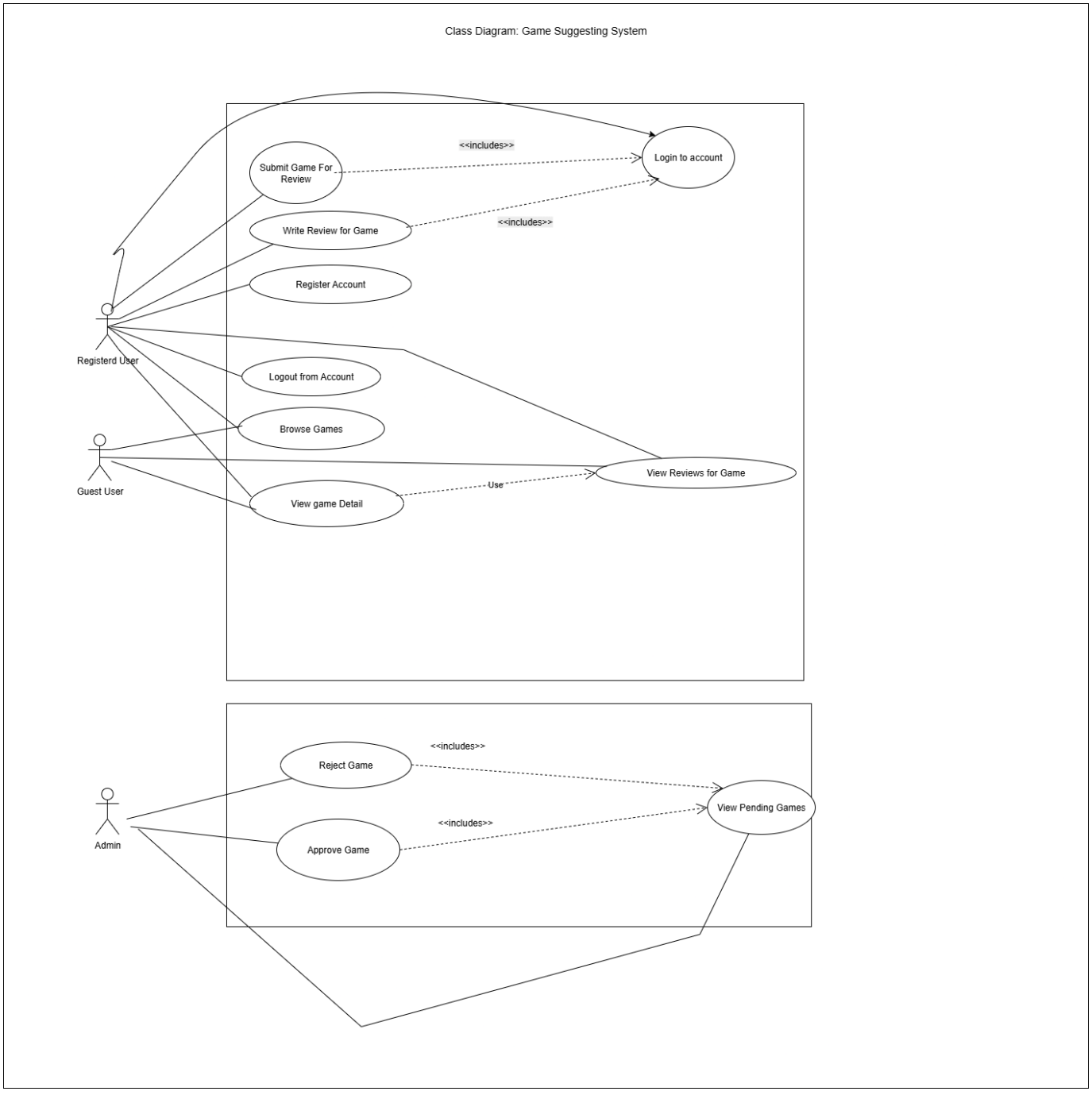
| EVENT NUMBER | DESCRIPTION | TRIGGER | REFLECTED STATUS | SYSTEM RESPONSE |
|---|---|---|---|---|
| 1 | New User Registration | User fills and submits the "Sign Up" form on the page. | The user's account is created and stored in the database | The system validates the input, hashes the password, creates a new User record, and logs the user in by creating a session. |
| 2 | User/Admin Login | A registered user or admin submits the "Sign In" form with their email and password. | The user is authenticated, stored in the browser's session storage. | The system validates credentials against the database. On success, it generates a JSON Web Token (JWT) and redirects the user to the main page |
| 3 | Browse Trending Games | User opens homepage. | Trending games list is displayed from **RAWG API**. | Frontend sends request to RAWG API, backend fetches trending game list, returns results to UI. |
| 4 | Browse Suggested Games | User opens homepage. | Suggested games list (from MongoDB) is displayed under "Suggested Games". | Backend fetches games from **Games collection** and returns to frontend. |
| 5 | Add Game | Logged-in user submits game name in **Add Yours form**. | Game stored in **PendingGames collection** for admin approval. | Backend queries RAWG API. If game found → save in **PendingGames** with status pending. If not found → enable **Manual Submission Form**. |

| | | | | |
|---|---|---|---|---|
| 6 | Manual Game Submission | RAWG API returns no result → user fills manual form with slug, name, description, image URL, genres, platforms, rating, release date, website | Submitted game stored in **PendingGames collection** with status pending | System saves the data with all user-entered fields and user receives a toast message “Game "${gameData.name}" submitted for admin approval! ” |
| 7 | Admin Approves game | Admin visits **/admin page**, selects a pending game, clicks Approve. | Game is added **Games collection**.And status attribute is removed | Backend saves approved game in Games collection and updates status to approved in PendingGames collection |
| 8 | Admin Rejects Game | Admin visits **/admin page**, clicks Reject. | Game remains in **PendingGames** with status = rejected. | System updates status field and saves optional adminNotes. |
| 9 | Admin Edits Game | Admin selects pending game and edits details. | Game record updated in DB. | System updates allowed fields (name, desc, genres, platforms, etc.) and saves changes. |
| 10 | View Game Details | User clicks on a game card (Trending/Suggested). | A **Big Card** opens showing description, genres, platforms, website link, rating, reviews. | If game exists in RAWG → fetch data from RAWG. If not → fetch from **MongoDB**. |
| 11 | Write Review | Logged-in user submits review (rating + comment) on a game. | Review saved in **Reviews collection**. | Backend validates input, saves review with username + timestamp+gameid +reviewText+rating |
| 12 | View Reviews | Any user (logged in or not) clicks on game details | Review list displayed. | System fetches reviews from **Reviews collection** sorted by latest. |
| 13 | Your Suggestions Page | Logged-in user clicks **Your Suggestions** in navbar. | Page shows games suggested by that user. | Backend queries **Games collection** where addedBy = username. |

| | | | | |
|---|---|---|---|---|
| 14 | About Page | User clicks "About" in navbar. | Project description and developer email shown. | Static content rendered from frontend. |
| 15 | Logout | Logged-in user/admin clicks **Logout button**. | Session cleared, UI resets to guest state. | System removes JWT token from browser storage and shows login/register options. |

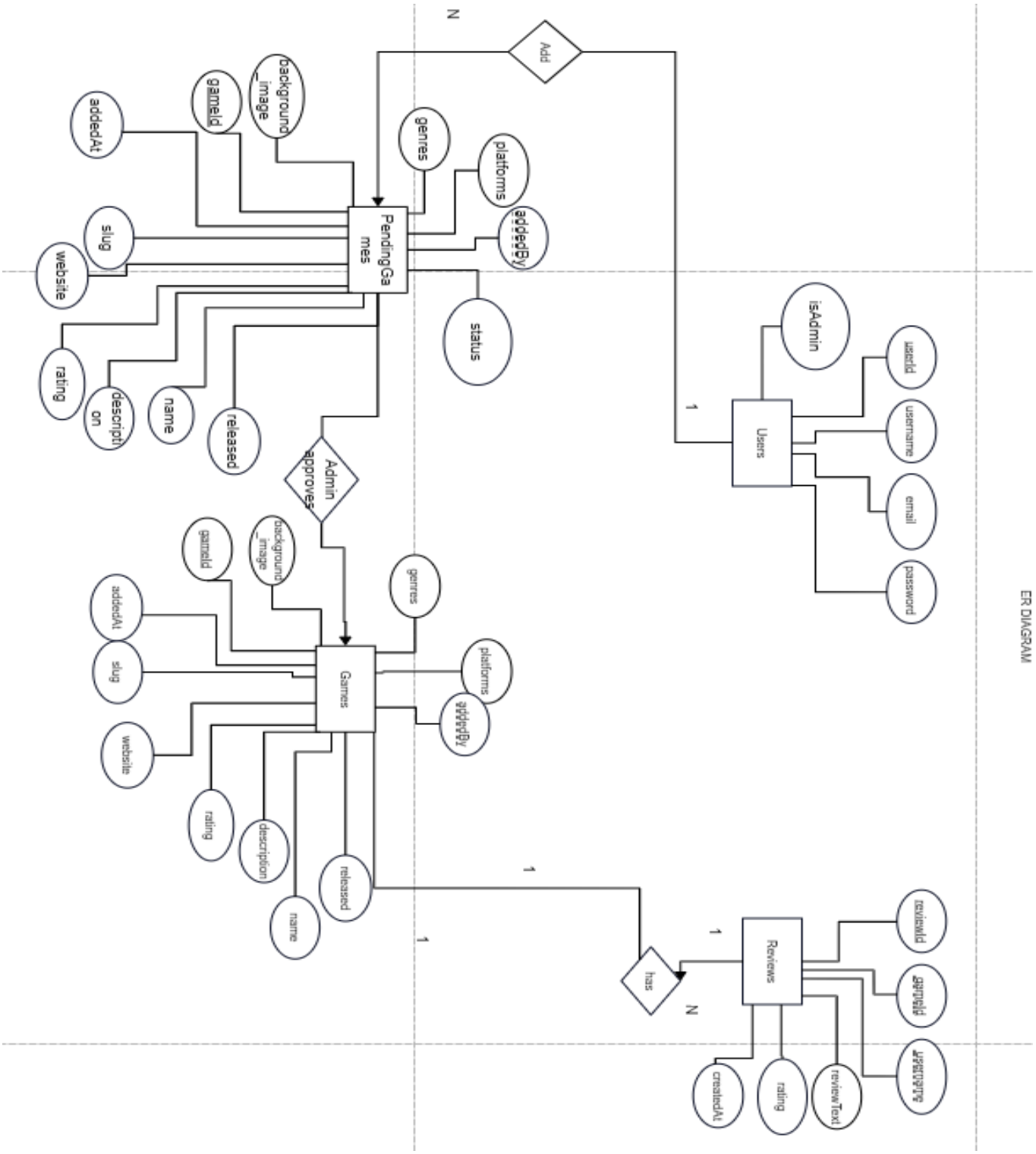
## (ii) Use Case Diagram

Shows how users and admins interact with GameSonnet features like login, suggest games, review, and approve games.
Clarifies functional requirements by highlighting actor–system relationships.



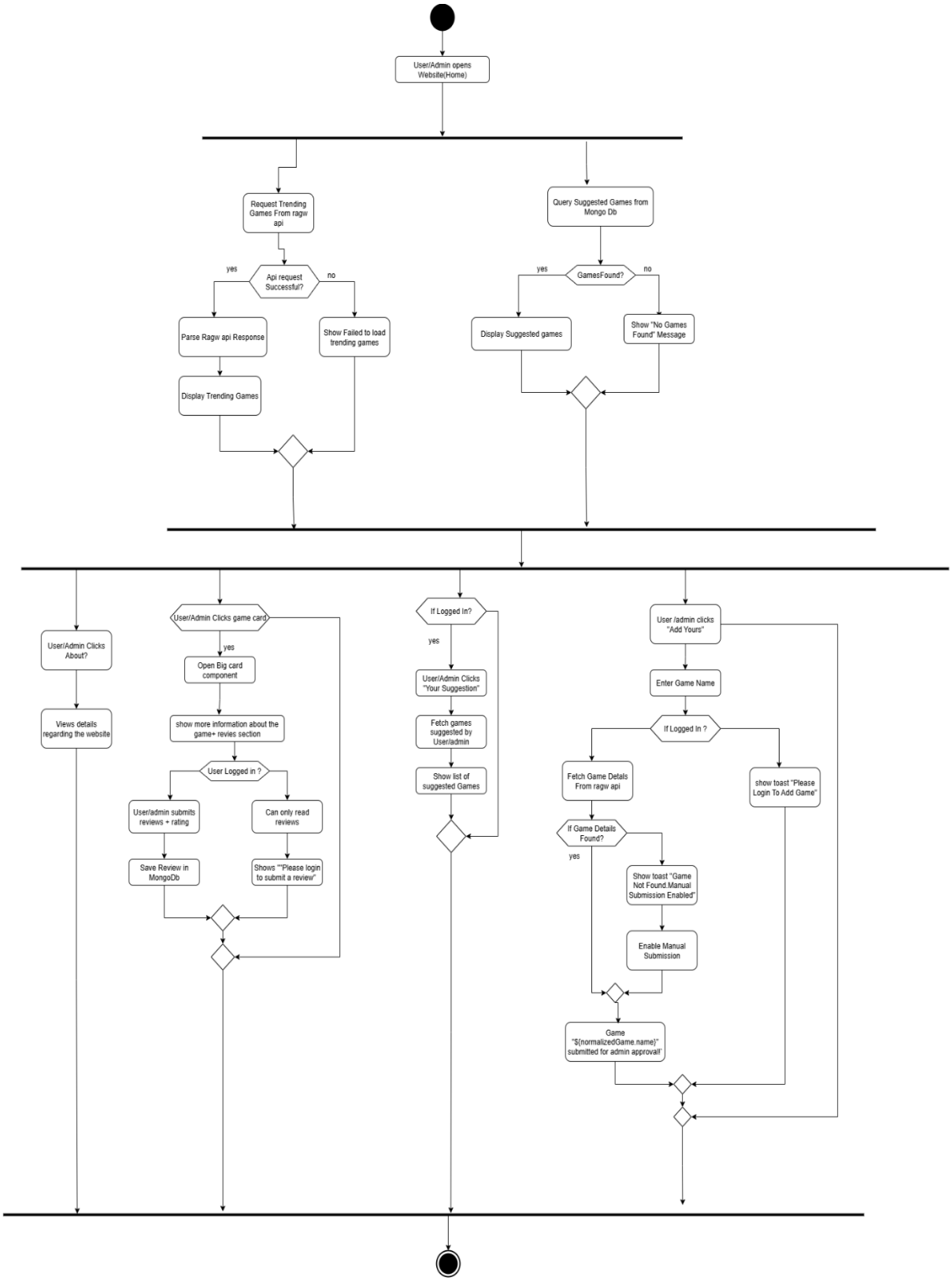
**(iii) ERD**

**Models the database with entities like Users, Games, PendingGames, and Reviews. Shows relationships like one-to-many between Users and Reviews, and Games and Reviews.**



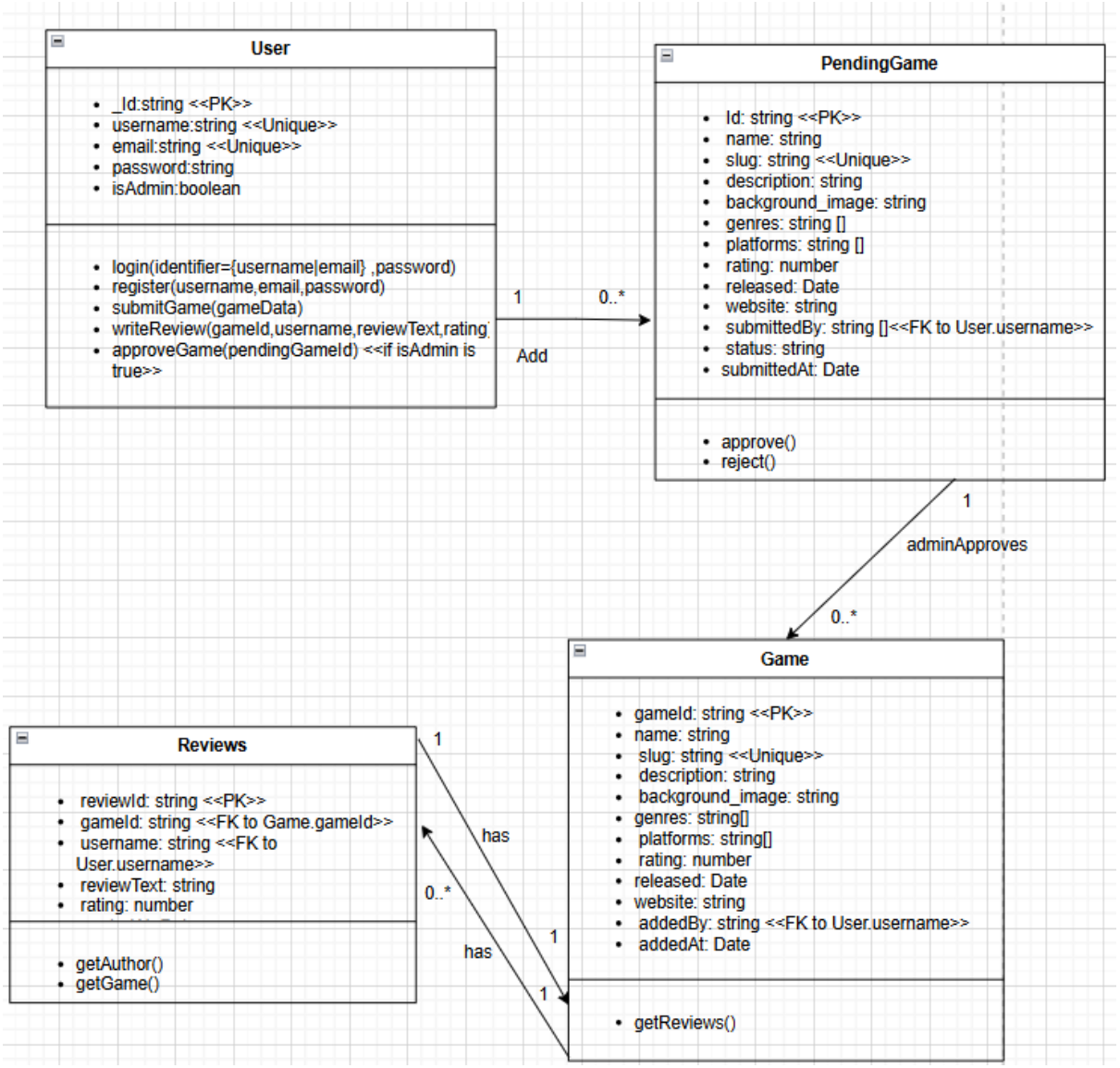
## (iv) Activity Diagram

Represents the workflow of adding games, reviewing. Illustrates sequential and conditional flows within GameSonnet processes.



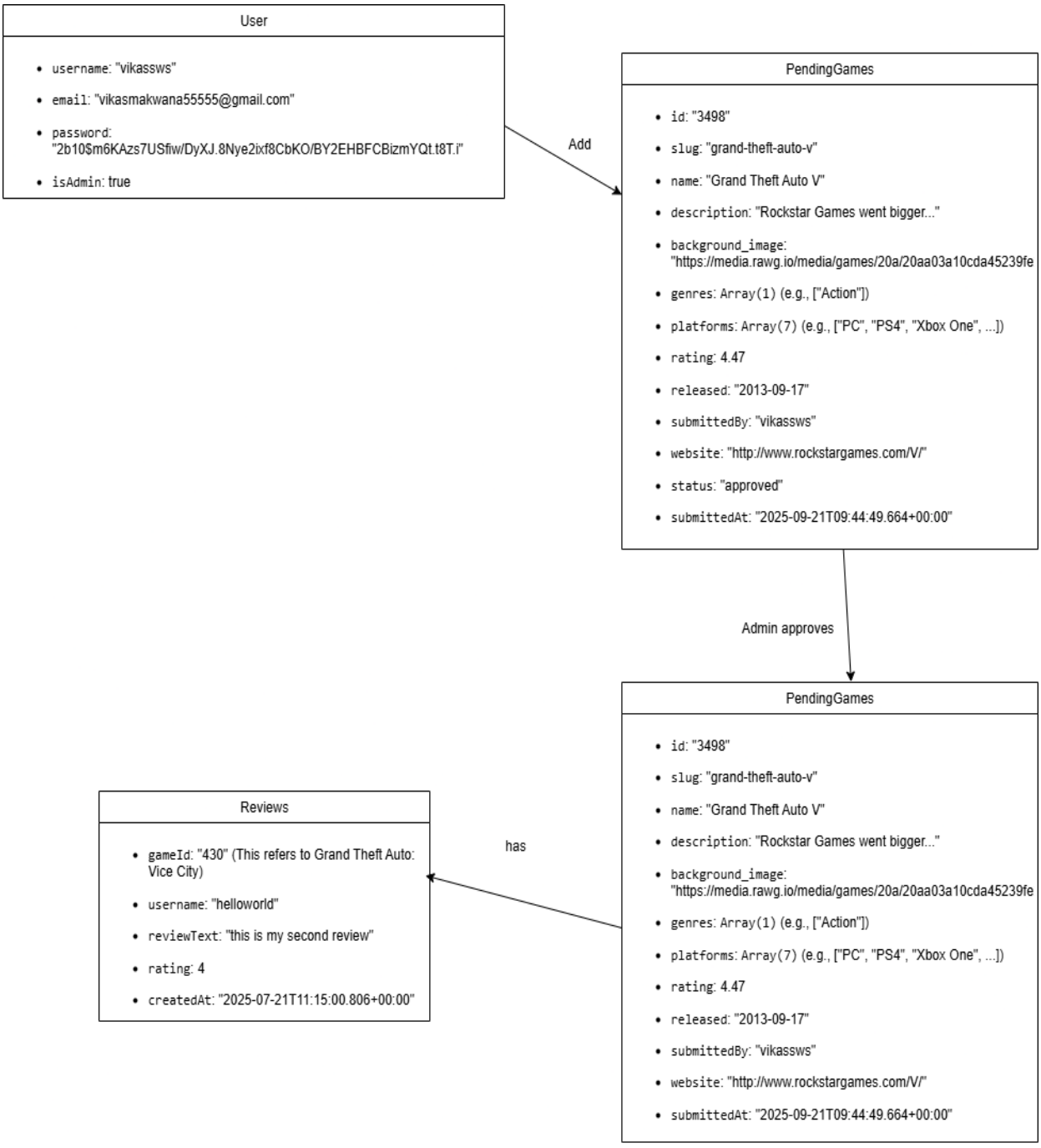
## (v) Class diagram

Defines classes such as User, Game,PendingGame &Review with their attributes and methods. Helps in designing object-oriented structure of the system.



## (vi)Object Diagram

Provides a snapshot of objects (e.g., a logged-in user suggesting a game with reviews).
Represents real instances of classes and their relationships at runtime.

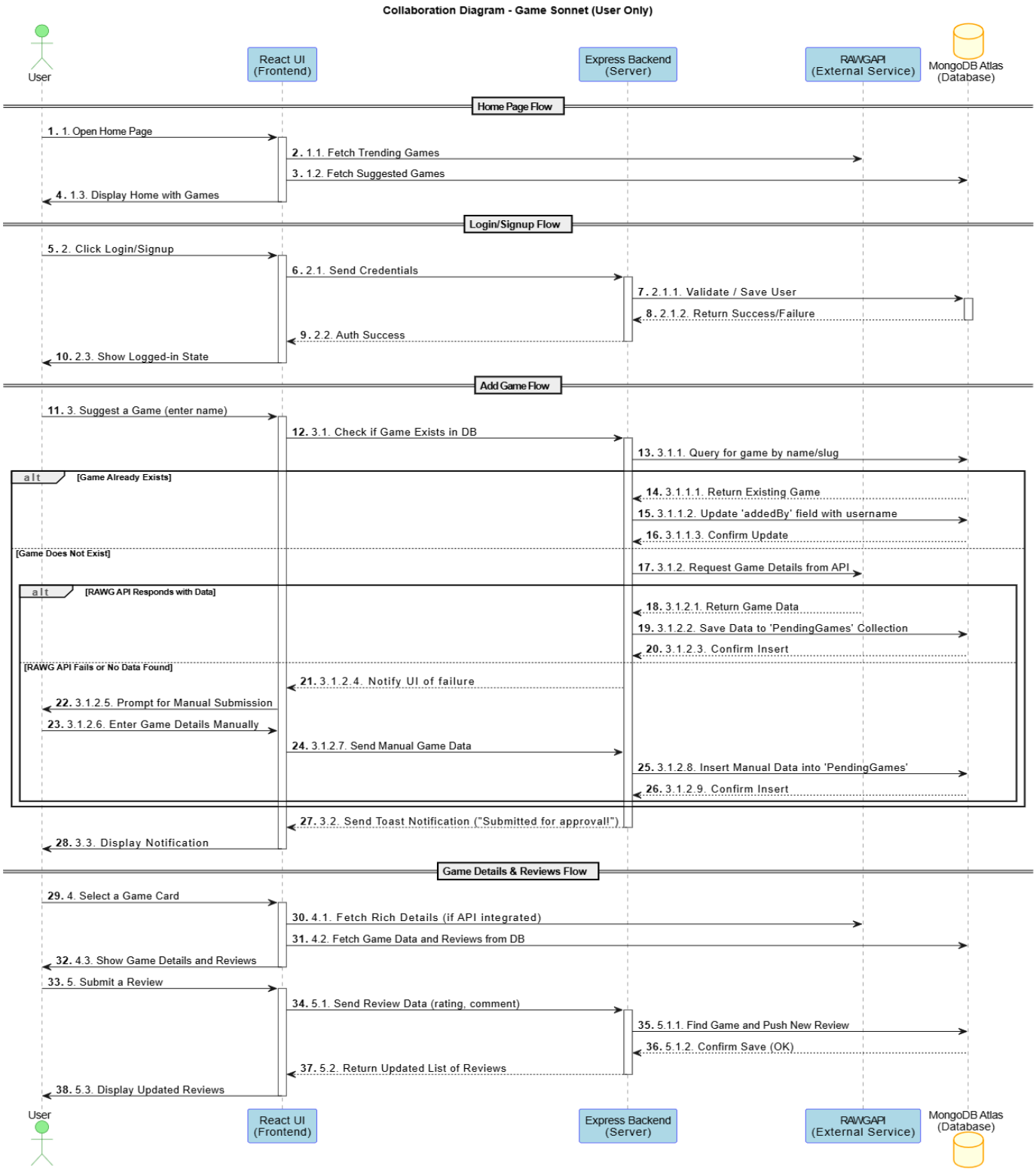


## (vii) Sequence diagram

Emphasizes the time-based message flow between User, System, & Database
Shows order of operations like login, game submission,and review posting.

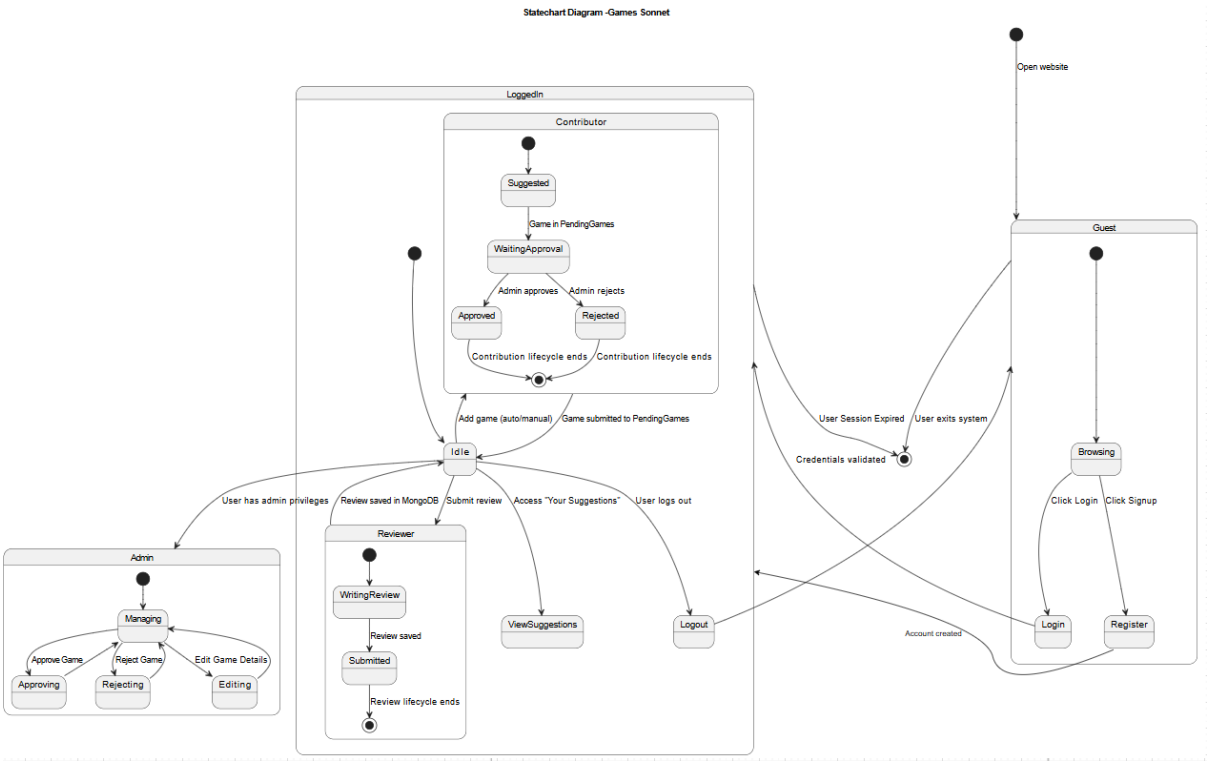
## (viii) Collaboration Diagram

Focuses on the structural organization of objects interacting during game suggestion and review.Highlights how objects communicate with numbering of messages.



## (xii) State diagram

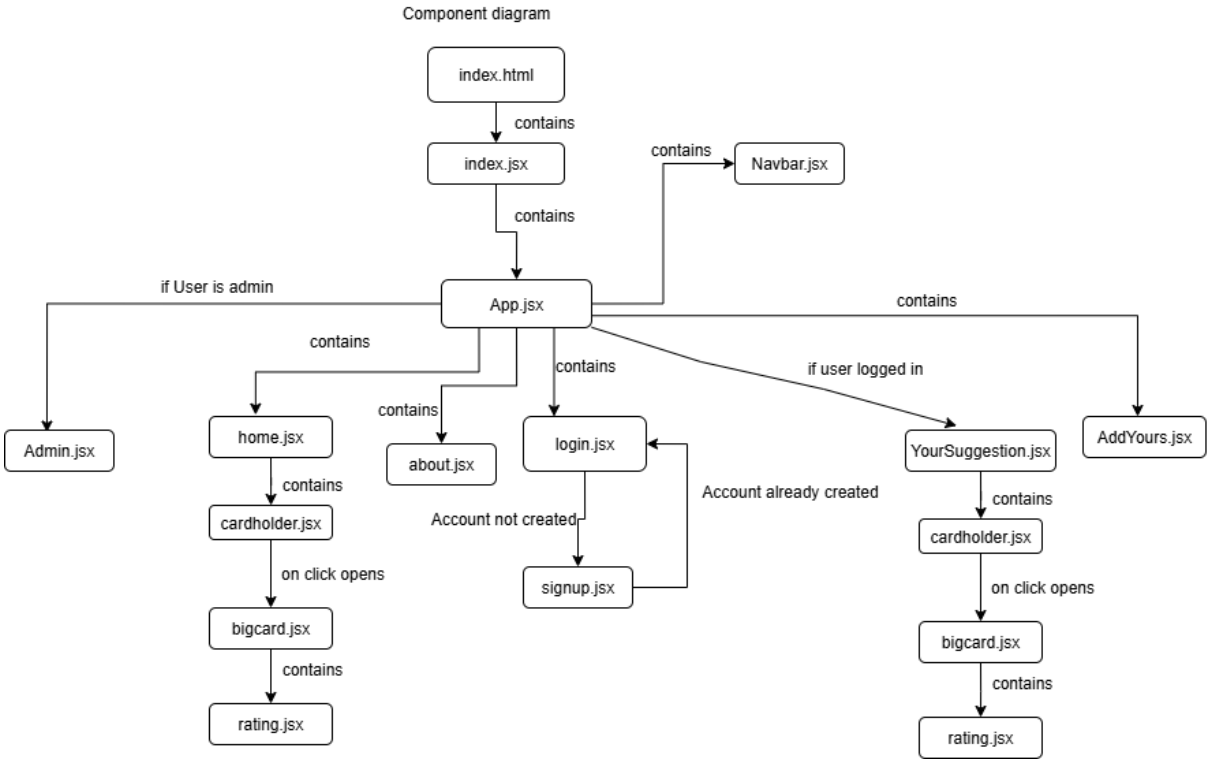
Shows state transitions of a user from Guest → LoggedIn → Contributor/Reviewer → Logout.
Also tracks game states from Pending → Approved/Rejected.



# System Design

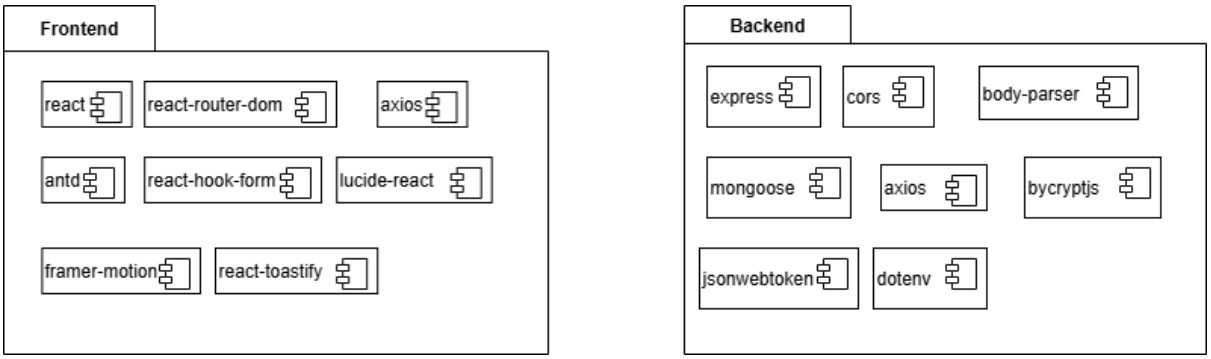
## i)Component Diagram

Depicts physical components like frontend React files
Represents how they connect to deliver system functionality.



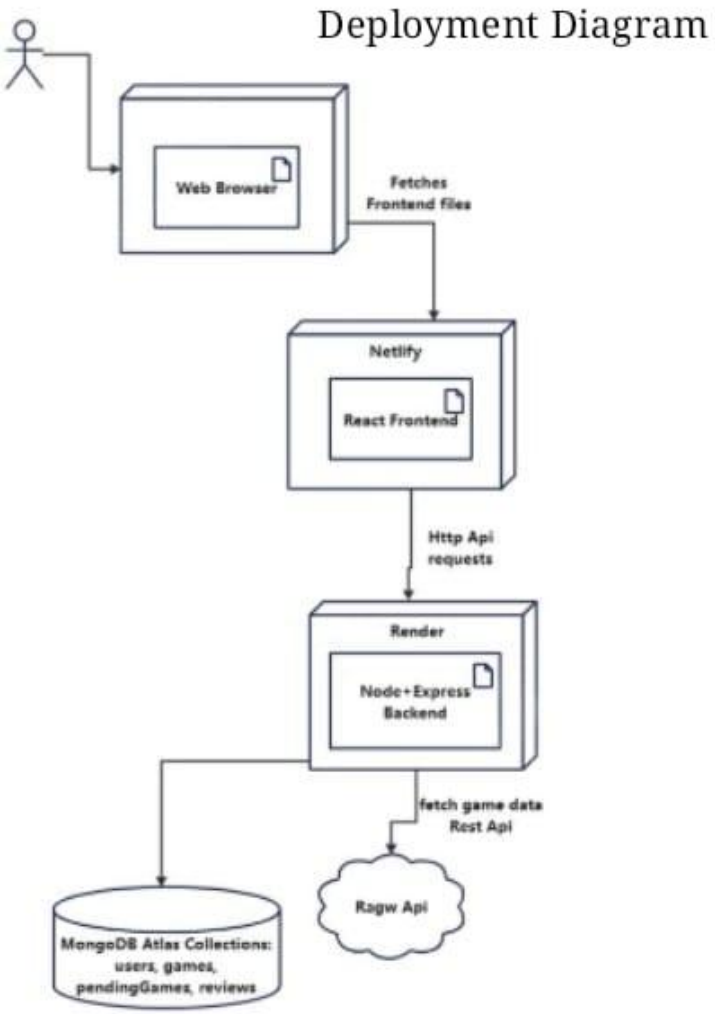
## (ii) Package Diagram

package diagram organizes the tech stack into two logical modules: Frontend (React, Axios, AntD, etc.) for user interface and interaction, and Backend (Express, Mongoose, bcryptjs, etc.) for routing, data handling, and authentication. This modular separation enhances maintainability, scalability, and clarity in your Game Suggesting Website's architecture.



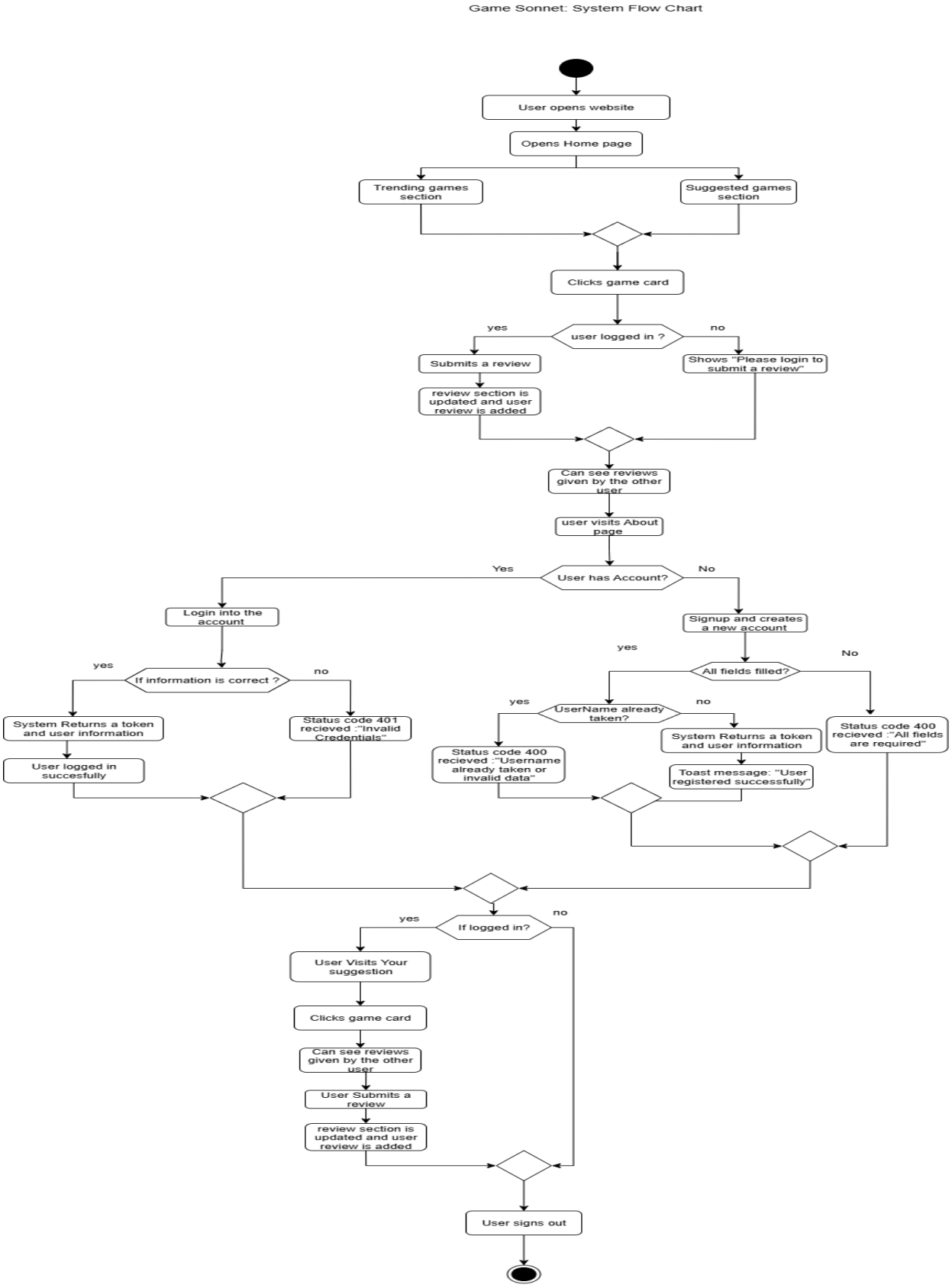
## (iii)Deployment Diagram

Shows system deployment on Netlify (frontend) and Render (backend) with MongoDB Atlas cloud database.
Represents execution environment and physical nodes.



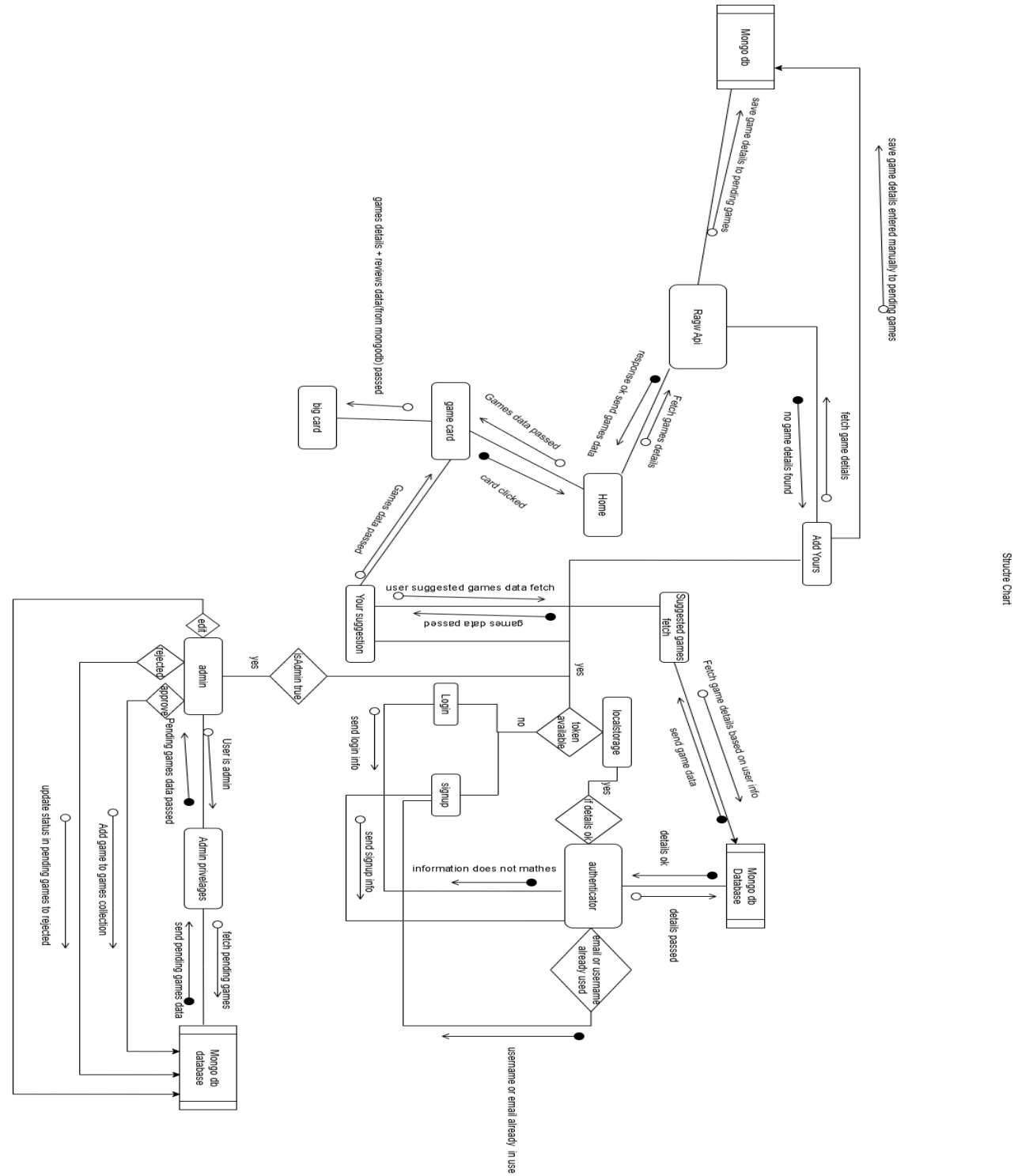
## (iv)System flow chart

Describes the overall logical flow of login, adding games, reviews, and admin approval.Gives a overall view of system operations.



## (v) Structured Chart

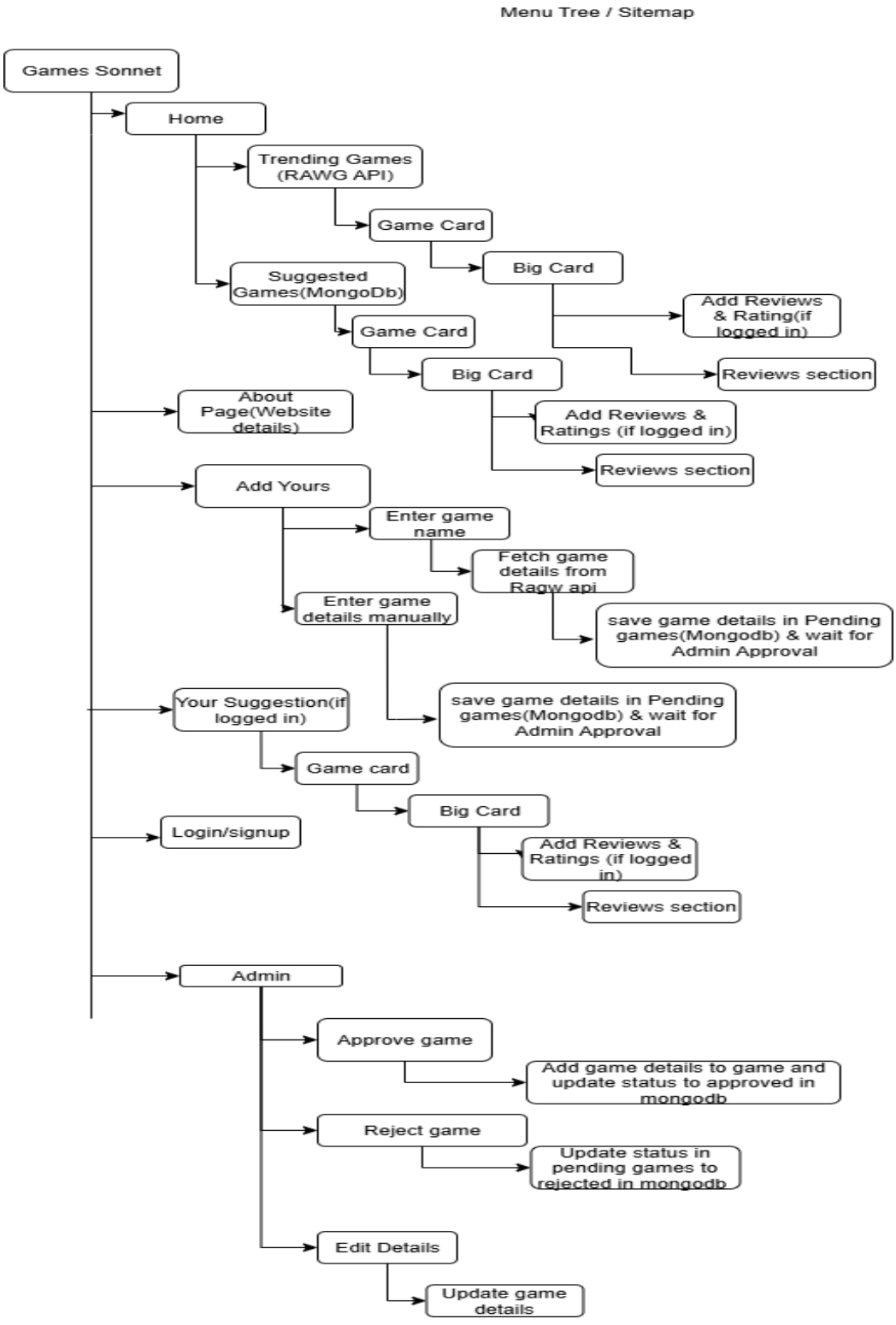
Hierarchical decomposition of GameSonnet into modules like Login, Home, Add Yours, and Admin.Shows relationships and calling structure among components.



# System Coding

## i) Menu Tree / Sitemap

Visual layout of navigation: Home, About, Add Yours, Your Suggestions, Login, Signup, Admin.Ensures clear user navigation across the site.



## ii)Data Dictionary

Defines all data fields like UserID, GameID, ReviewID, AddedBy, Rating, etc.Provides metadata (type, size, description) for consistent database design.

## User

| Field Name | Data Type | Description | Constraints & Rules | Example Value |
|---|---|---|---|---|
| **_id** | ObjectId | Unique identifier for the user document. | Primary Key, Required, Autogenerated | ObjectId('...') |
| **username** | String | The user's publicfacing name, used for logging in and on reviews. | Required, Unique | "helloworld" |
| **email** | String | The user's email address for notifications and account recovery. | Required, Unique | "user@example.com" |
| **password** | String | A securely hashed version of the user's password. | Required | "bcr$ypt$h4$hed..." |
| **role** | String | Defines the user's permission level. Can be 'user' or 'admin'. | Required, Enum, Default: 'user' | "user" |

Pending Games

| Field Name | Data Type | Description | Constraints & Rules | Example Value |
|---|---|---|---|---|
| **_id** | Object Id | Unique identifier for the pending game | Primary Key, Required,Auto-generated | ObjectId('68cfc...') |

| | | | | |
|---|---|---|---|---|
| **createdAt** | Date | Timestamp indicating when the user account was created. | Required, Default: Current Date | 2025-07-20T10:00:00Z |
| **id** | Number | Unique identifier from the RAWG API. May be null for manual submissions. | Nullable | 3498 |
| **name** | String | The name of the game as submitted by the user or fetched from the API. | Required | "Grand Theft Auto V" |
| **slug** | String | A URL-friendly version of the game's name. | Unique (if not null), Nullable | "grand-theft-auto-v" |
| **description** | String | Game synopsis. May be empty for manual submissions. | Nullable | "Rockstar Games went..." |
| **background_image** | String | URL for the game's cover image | Nullable | "https://media.rawg.io/..." |

| | | | | |
|---|---|---|---|---|
| **genres** | Array of Objects | List of genres associated with the game. | Nullable | [{"name": "Action"}] |
| **platforms** | Array of Objects | List of platforms the game is available on. | Nullable | [{"platform": {"name": "PC"}}] |
| **rating** | Number | The average rating from the RAWG API. | Nullable | 4.47 |
| **released** | String | The release date from the RAWG API (YYYYMM-DD). | Nullable | "2013-09-17" |
| **website** | String | The official website URL. | Nullable | "https://www.rockstargames.com/V/" |
| **submittedBy** | String | Username of the user who submitted the game for approval. | Required | "vikassws" |
| **status** | String | The current approval status of the submission. | Enum: ('pending', 'approved', 'rejected') | "approved" |

| | | | | |
|---|---|---|---|---|
| **submittedAt** | Date | Timestamp indicating when the game was submitted. | Required, Default: Current Date | 2025-09-21T09:44:49Z |
| **__v** | Number | The internal version key used by Mongoose. | Automanaged by Mongoose | 0 |

## Games

| Field Name | Data Type | Description | Constraints & Rules | Example Value |
|---|---|---|---|---|
| **_id** | ObjectId | Unique identifier for the game document. | Primary Key, Required, Autogenerated | ObjectId('687e...') |
| **id** | Number | Unique identifier from RAWG API. Used as a foreign key by reviews. | Required, Unique | 326292 |
| **slug** | String | A URL-friendly version of the game's name. | Required, Unique | "fall-guys" |
| **name** | String | The official name of the game. | Required | "Fall Guys: Ultimate Knockout" |
| **description** | String | A brief summary or synopsis of the game. | Required | "Fall Guys: Ultimate..." |

| | | | | |
|---|---|---|---|---|
| **background_image** | String | URL for the game's background or cover image. | Required | "https://media.rawg.io/..." |
| **genres** | Array of Objects | List of genres associated with the game. | Required | [{"name": "Action"}] |
| **platforms** | Array of Objects | List of platforms the game is available on. | Required | [{"platform": {"name": "PC"}}] |
| **rating** | Number | The average rating of the game. | Required | 3.73 |
| **released** | String | The official release date of the game (YYYY-MMDD). | Required | "2020-08-04" |
| **addedBy** | String | Username of all people who submitted the game | Required | “Rajesh” |
| **addedAt** | Date | Timestamp when the game was approved and added to this collection. | Required | 2025-07-21T11:15:54Z |

| | | | | |
|---|---|---|---|---|
| **website** | String | The official website URL for the game. | Required | "https://fallguys.com/" |
| **__v** | Number | The internal version key used by Mongoose. | Automanaged by Mongoose | 0 |

## Reviews

| Field Name | Data Type | Description | Constraints & Rules | Example Value |
|---|---|---|---|---|
| **_id** | ObjectId | Primary key, unique identifier for the review. | Primary Key, Required, Autogenerated | ObjectId('687e...') |
| **gameId** | Number | Identifier for the game being reviewed. | Required, Foreign Key (references games.id) | 430 |
| **username** | String | Username of the reviewer. | Required, Foreign Key (references users.username) | "helloworld" |
| **reviewText** | String | The text content of the review. | Required | "this is my second review" |
| **rating** | Number | The numerical score given by the user (e.g ., 15) | Required | 4 |
| **createdAt** | Date | Timestamp indicating when the review was created. | Required, Default: Current Date | 2025-07-21T11:15:00Z |
| **__v** | Number | The internal version key used by Mongoose. | Auto-managed by Mongoose | 0 |

## iii) Design Patterns used

**1. Model–View–Controller (MVC Pattern)**

- **Where used:** In the **backend (Express + MongoDB)**.
- **How it works in your project:**
    - **Model:** MongoDB Schemas (User, Game, PendingGame, Review).
    - **Controller/Service:** Express route handlers (e.g., /login, /save-game, /admin/approve-game).
    - **View:** React frontend (Home, AddYours, Admin Panel).
- **Advantage:** Separation of concerns, making the code cleaner, testable, and easier to maintain.

**2. Factory Method Pattern**

- **Where used:** When creating new **Game** or **PendingGame** objects.
- **How it works in your project:**
    - Depending on context (approved vs pending), a Game or PendingGame document is created.
    - Example: User submission first creates a PendingGame, but when approved → converted into a Game.
- **Advantage:** Encapsulates object creation logic and ensures consistent structure

**3. Singleton Pattern**

- **Where used:**
    - **MongoDB Connection** – mongoose.connect(...) is initialized once and reused across the backend.
    - **JWT Secret & Config** – loaded from .env and used globally.
- **Advantage:** Ensures single instance of database connection/configuration to prevent resource overhead.

**4. Observer Pattern (Event-driven)**

- **Where used:** In **user interactions** (reviews, game suggestions).
- **How it works in your project:**
    - When a **game is added** → notification/feedback (toast message) is triggered.
    - When a **review is submitted** → review list updates dynamically.
- **Advantage:** Decouples frontend components and enables real-time updates.

**5. Strategy Pattern**

- **Where used: Game fetching logic**.
- **How it works in your project:**

    - If game exists in RAWG API → fetch from API.
    - If not → fetch from MongoDB or allow manual submission.
    - This represents different "strategies" for fetching data based on conditions.
- **Advantage:** Improves flexibility by switching between API-fetch and DB-fetch without changing overall logic.

**6. Decorator Pattern (Enhancement of Objects)**

- **Where used: Adding contributors in addedBy field**.
- **How it works in your project:**
    - If another user suggests the same game → system doesn't duplicate the entry, instead it **extends the existing game object by adding new contributor**.
- **Advantage:** Enhances objects dynamically without modifying their structure.

## iv)Source Code

## Home.jsx

```
import React, { useState, useEffect, useCallback } from "react";
import GameCard from "./cardholder";
import MyComponent from "./bigcard";
import axios from "axios";

const Home = ({
  cachedGames,
  setCachedGames,
  cachedSuggestedGames,
  setCachedSuggestedGames,
  newSuggestedGame // <-- added prop
}) => {
  const [games, setGames] = useState([]);
  const [suggestedGames, setSuggestedGames] = useState([]);
  const [selectedGame, setSelectedGame] = useState(null);
  const [startIndex, setStartIndex] = useState(0);
  const [suggestedStartIndex, setSuggestedStartIndex] = useState(0);
  const [cardsPerPage, setCardsPerPage] = useState(3);

  const apiKey = "9030709abcec4c239d8e479434b76ea5";
  const updateCardsPerPage = useCallback(() => {
    const width = window.innerWidth;
    if (width >= 1310) setCardsPerPage(5);
    else if (width >= 1050) setCardsPerPage(4);
    else if (width >= 570) setCardsPerPage(3);
    else if (width >= 300) setCardsPerPage(2);
```

```
  else setCardsPerPage(1);
}, []);

useEffect(() => {
  updateCardsPerPage();
  window.addEventListener("resize", updateCardsPerPage);
  return () => window.removeEventListener("resize", updateCardsPerPage);
}, [updateCardsPerPage]);

useEffect(() => {
  if (cachedGames.length === 0) {
    axios
      .get(`https://api.rawg.io/api/games?ordering=-added&page_size=12&key=${apiKey}`)
      .then((res) => {
        setGames(res.data.results);
        setCachedGames(res.data.results);
      })
      .catch((err) => {
        console.error("Error fetching trending games:", err);
        setGames([]);
      });
  } else {
    setGames(cachedGames);
  }
  if (cachedSuggestedGames.length === 0) {
    axios
      .get("https://gamessonnet.onrender.com/suggested-games")
      .then((res) => {
        setSuggestedGames(res.data);
        setCachedSuggestedGames(res.data);
      })
```

```
      .catch((err) => {
        console.error("Error fetching suggested games:", err);
        setSuggestedGames([]);
      });
  } else {
    setSuggestedGames(cachedSuggestedGames);
  }
}, []);

useEffect(() => {
  if (newSuggestedGame && Object.keys(newSuggestedGame).length !== 0) {
    setSuggestedGames((prev) => [...prev, newSuggestedGame]);
    setCachedSuggestedGames((prev) => [...prev, newSuggestedGame]);
  }
}, [newSuggestedGame, setCachedSuggestedGames]);
const fetchGameDetails = useCallback(
  async (slug) => {
    try {
      const response = await axios.get(
        `https://api.rawg.io/api/games/${slug}?key=${apiKey}`
      );
      setSelectedGame(response.data);
    } catch (apiError) {
      console.warn("RAWG API failed, fetching from DB fallback...", apiError);
      // Fallback to database
      try {
        const dbResponse = await axios.get(
          `http://localhost:3000/game-details/${slug}`
        );
        setSelectedGame(dbResponse.data);
      } catch (dbError) {
```

```
        console.error("DB fallback failed:", dbError);
        alert("Failed to fetch game details from both RAWG API and database.");}}
},
  [apiKey]
 );

 const handleTrendingCardClick = useCallback(
  (game) => {
   fetchGameDetails(game.slug);
  },
  [fetchGameDetails]
 );

 const handleSuggestedCardClick = useCallback((game) => {
  if (!game) return;

  setSelectedGame({
   id: game.id,
   name: game.name,
   description_raw: game.description,
   background_image: game.background_image,
   genres: game.genres,
   platforms: game.platforms,
   website: game.website,
  });
 }, []);

 const handlePrev = (isSuggested) => {
  if (isSuggested) {
   setSuggestedStartIndex((prev) => Math.max(prev - cardsPerPage, 0));
  } else {
```

```
    setStartIndex((prev) => Math.max(prev - cardsPerPage, 0));
  }
};

const handleNext = (isSuggested) => {
  if (isSuggested) {
    setSuggestedStartIndex((prev) =>
      Math.min(prev + cardsPerPage, suggestedGames.length - cardsPerPage)
    );
  } else {
    setStartIndex((prev) =>
      Math.min(prev + cardsPerPage, games.length - cardsPerPage)
    );
  }
};

const visibleGames = games.slice(startIndex, startIndex + cardsPerPage);
const visibleSuggestedGames = suggestedGames.slice(
  suggestedStartIndex,
  suggestedStartIndex + cardsPerPage
);
return (
  <div style={{ position: "relative", width: "100vw" }}>
    {/* Trending & Suggested Sections */}
    <div style={{ padding: window.innerWidth < 768 ? "10px" : "20px", maxWidth: "100%", overflow:
"hidden" }}>
      <div style={{ paddingBottom: "9px" }}>
        <SectionHeader
          style={{ fontSize: "20px" }}
          title={
            <>
```

```
        <img
          src="/trending-up-svgrepo-com.svg"
          alt="Trending"
          style={{
            width: window.innerWidth < 768 ? "20px" : "30px",
            height: window.innerWidth < 768 ? "20px" : "30px",
            marginRight: "8px",
            verticalAlign: "middle",
            filter: "invert(100%)"
          }}
        />
        Trending Games
      </>
    }
    onPrev={() => handlePrev(false)}
    onNext={() => handleNext(false)}
    disabledPrev={startIndex === 0}
    disabledNext={startIndex + cardsPerPage >= games.length}
  />
</div>
<CardList games={visibleGames} onCardClick={handleTrendingCardClick} />

<div style={{ marginTop: window.innerWidth < 768 ? "20px" : "40px" }}>
  <SectionHeader
    title={
      <>
        <img
          src="/glitter-2-svgrepo-com.svg"
          alt="Trending"
          style={{
            width: window.innerWidth < 768 ? "20px" : "30px",
```

```
            height: window.innerWidth < 768 ? "20px" : "30px",
            marginRight: "8px",
            verticalAlign: "middle",
            filter: "invert(100%)"
          }}
        />
        Suggested Games
      </>
    }
    onPrev={() => handlePrev(true)}
    onNext={() => handleNext(true)}
    disabledPrev={suggestedStartIndex === 0}
    disabledNext={suggestedStartIndex + cardsPerPage >= suggestedGames.length}
  />
  <CardList games={visibleSuggestedGames} onCardClick={handleSuggestedCardClick} />
 </div>
</div>
{selectedGame && (
 <div
  style={{
   position: "fixed",
   top: 0,
   left: 0,
   width: "100%",
   height: "100%",
   backgroundColor: "rgba(0, 0, 0, 0.8)",
   display: "flex",
   justifyContent: "center",
   alignItems: "center",
   zIndex: 1000,
   padding: window.innerWidth < 768 ? "10px" : "20px",
```

```
          boxSizing: "border-box"
        }}
        onClick={() => setSelectedGame(null)}
      >
        <div
          style={{
            width: "100%",
            maxWidth: window.innerWidth < 768 ? "100%" : "800px",
            position: "relative",
            maxHeight: "90vh",
            overflow: "auto"
          }}
          onClick={(e) => e.stopPropagation()}
        >
          <MyComponent
            id={selectedGame.id}
            image={selectedGame.background_image}
            title={selectedGame.name}
            description={selectedGame.description_raw || selectedGame.description}
            genre={selectedGame.genres?.map(g => typeof g === "string" ? g : g.name).join(", ") || "N/A"}
            platform={selectedGame.platforms?.map(p =>
              typeof p === "string" ? p : p.platform?.name || p.name
            ).join(", ") || "N/A"}
            website={selectedGame.website}
          />        </div>
      </div>
    )}
  </div>
 );
};
const SectionHeader = ({ title, onPrev, onNext, disabledPrev, disabledNext }) => (
```

```
  <div style={{
    display: "flex",
    alignItems: "center",
    justifyContent: "space-between",
    flexWrap: "wrap",
    gap: "10px"
  }}>
    <div style={{
      fontSize: window.innerWidth < 768 ? "18px" : "24px",
      fontWeight: "bold",
      marginBottom: "15px",
      flex: "1",
      minWidth: "200px"
    }}>
      {title}
    </div>
    <div style={{ display: "flex", gap: window.innerWidth < 768 ? "5px" : "10px" }}>
      <NavButton direction="left" onClick={onPrev} disabled={disabledPrev} />
      <NavButton direction="right" onClick={onNext} disabled={disabledNext} />
    </div>
  </div>
);
const CardList = React.memo(({ games, onCardClick }) => (
  <div style={{ display: "flex", alignItems: "center", gap: "10px" }}>
    <div style={{
      display: "flex",
      gap: window.innerWidth < 768 ? "10px" : "20px",
      overflow: "hidden",
      flex: 1,
      paddingBottom: "10px"
    }}>
```

```
      {games.length === 0 ? (
        <div style={{
          padding: "20px",
          textAlign: "center",
          color: "#999",
          fontSize: window.innerWidth < 768 ? "14px" : "16px"
        }}> No games found.  </div>
      ) : (
        games.filter(game => game && (game.id || game.gameId || game.slug)).map((game, index) => (
          <GameCard
            key={game.id || game.gameId || game.slug || `game-${index}`}
            game={game}
            onClick={() => onCardClick(game)} />))}
    </div>
  </div>
));
const NavButton = ({ direction, onClick, disabled }) => (
  <button
    style={{
      color: "white",
      backgroundColor: "#111B1C",
      border: "none",
      borderBottom: '2px solid gray',
      borderRadius: "50%",
      width: window.innerWidth < 768 ? "40px" : "55px",
      height: window.innerWidth < 768 ? "40px" : "55px",
      display: "flex",
      alignItems: "center",
      justifyContent: "center",
      fontSize: window.innerWidth < 768 ? "14px" : "18px",
      cursor: disabled ? "not-allowed" : "pointer",
```

```
      opacity: disabled ? 0.5 : 1,
      transition: "all 0.2s ease"
    }}
    onClick={onClick}
    disabled={disabled}
  >
    {direction === "left" ? "<" : ">"}
  </button>
);
export default Home;
```

## About.jsx

```
import React from 'react';
import './about.css';
const About = () => {
  return (
    <><div >
      <div className="about-box-shadow">
        This project is all about where you can suggest the games you liked to others
        so that other people can also know about the games and experience them for themselves.
      </div>
      </div>
      <footer
        style={{
          textAlign: 'center',
          padding: '20px',
          fontFamily: "'Segoe UI', 'Roboto', 'Helvetica Neue', sans-serif",
          color: '#9CA3AF',
          fontSize: '2vmax',
          borderTop: '1px solid #444',
```

```
        marginTop: '60px',
        backgroundColor: '#0E0E10',
      }} > Contact us at: <a href="mailto:gamessonnet@gmail.com" style={{ color: '#61dafb', textDecoration: 'none' }}>gamessonnet@gmail.com</a>
    </footer>
  </>);  };
export default About;
```

## AddYours.jsx

```
import React, { useState } from "react";
import axios from "axios";
import "./font.css";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const apiKey = "9030709abcec4c239d8e479434b76ea5";
const AddYours = () => {
  const [gameName, setGameName] = useState("");
  const [isManual, setIsManual] = useState(false);
  const [fetchedGame, setFetchedGame] = useState(null);
  const [manualData, setManualData] = useState({
    slug: "",
    name: "",
    description: "",
    background_image: "",
    genres: "",
    platforms: "",
    rating: "",
    released: "",
    website: "",
  });
```

```
const [showMessage, setShowMessage] = useState(false);
const token = localStorage.getItem("token");
const user = JSON.parse(localStorage.getItem("user")) || {};
const normalizeGameName = (name) =>
  name.trim().toLowerCase().replace(/\s+/g, "-");
const normalizeGameForBackend = (gameData) => {
  return {
    id: gameData.id,
    slug: gameData.slug,
    name: gameData.name,
    description: gameData.description_raw || gameData.description || "",
    background_image: gameData.background_image || "",
    genres: gameData.genres?.map((g) => g.name) || [],
    platforms: gameData.platforms?.map((p) => p.platform?.name || p.name) || [],
    rating: gameData.rating || 0,
    released: gameData.released || "",
    website: gameData.website || "",
  };
};
const fetchGameFromRAWG = async (slug) => {
  try {
    const response = await axios.get(`https://api.rawg.io/api/games/${slug}`, {
      params: { key: apiKey },
    });

    if (response.data.redirect && response.data.slug) {
      const correctedSlug = response.data.slug;
      const correctedResponse = await axios.get(
        `https://api.rawg.io/api/games/${correctedSlug}`,
        { params: { key: apiKey } }
      );
```

```
      return correctedResponse.data;
    }
    return response.data;
  } catch {
    return null;
  }
};
const saveGameToBackend = async (gameData) => {
  try {
    if (!token) {
      toast.error("You must be logged in to add a game.");
      return;
    }
    await axios.post(
      "http://localhost:3000/save-game",
      { gameData },
      { headers: { Authorization: `Bearer ${token}` } }
    );
  } catch (err) {
    toast.error(err.response?.status === 409 ? "Game already exists." : "Failed to save game.");
    console.error("Error saving game:", err);
  }
};
const checkGame = async () => {
  if (!token) {
    toast.error("Please log in to add a game.");
    return;
  }
  if (!gameName) {
    toast.error("Please enter a game name.");
    return;
```

```
  }
  const slug = normalizeGameName(gameName);
  const gameData = await fetchGameFromRAWG(slug);
  if (gameData) {
    const normalizedGame = normalizeGameForBackend(gameData);
    setFetchedGame(normalizedGame);
    await saveGameToBackend(normalizedGame);
    toast.success(`Game "${normalizedGame.name}" submitted for admin approval!`);
  } else {
    toast.info("Game not found. Manual submission enabled.");
    setIsManual(true);
  }
};
const handleManualSubmit = async (e) => {
  e.preventDefault();
  if (!manualData.name || !manualData.slug || !manualData.background_image) {
    toast.error("Please fill all required fields.");
    return;
  }
  const id = generateUniqueId();
  const addedAt = new Date().toISOString();
  const gameData = {
    ...manualData,
    id,
    genres: manualData.genres.split(",").map((g) => g.trim()),
    platforms: manualData.platforms.split(",").map((p) => p.trim()),
    addedBy: user.username || "Anonymous",
    addedAt,
  };
  try {
    await saveGameToBackend(gameData);
```

```
    toast.success(`Game "${gameData.name}" submitted for admin approval!`);
    resetForm();
  } catch (err) {
    console.error("Error saving manual game:", err);
    toast.error("Failed to save game manually.");
  }
};
const resetForm = () => {
  setGameName("");
  setIsManual(false);
  setManualData({
    slug: "",
    name: "",
    description: "",
    background_image: "",
    genres: "",
    platforms: "",
    rating: "",
    released: "",
    website: "",
  });
};
const generateUniqueId = () => Math.floor(1000000000 + Math.random() * 9000000000);
return (
  <div
    style={{
      minHeight: "100vh",
      backgroundColor: "#101014",
      display: "flex",
      justifyContent: "center",
      alignItems: "flex-start",
```

```
    paddingTop: "80px",
    paddingBottom: "50px",
  }}
>
  <ToastContainer />
  <div
    style={{
      width: "100%",
      maxWidth: "600px",
      backgroundColor: "rgba(39, 47, 53, 0.95)",
      borderRadius: "12px",
      padding: "30px",
      boxShadow: "0 0 20px rgba(255,255,255,0.1)",
      position: "relative",  }}>
    <h2 style={{ color: "#fff", textAlign: "center", marginBottom: "20px" }}>
      Add a Game   </h2>
    <input
      type="text"
      placeholder="Enter game name..."
      value={gameName}
      onChange={(e) => setGameName(e.target.value)}
      style={{
        width: "100%",
        padding: "10px",
        borderRadius: "6px",
        marginBottom: "15px",
        border: "none",
        fontSize: "16px",   }} />
    <button
      onClick={checkGame}
      onMouseEnter={() => {
```

```
    if (!token) setShowMessage(true);
   }}
   onMouseLeave={() => setShowMessage(false)}
   style={{
    width: "100%",
    padding: "12px",
    border: "none",
    borderRadius: "8px",
    backgroundColor: token ? "#28a745" : "#6c757d",
    color: "#fff",
    fontWeight: "bold",
    cursor: token ? "pointer" : "not-allowed",
    marginBottom: "20px",
   }} > Check and Add Game </button>
  {!token && showMessage && (
   <div
    style={{
     position: "absolute",
     top: "60px",
     left: "50%",
     transform: "translateX(-50%)",
     backgroundColor: "#333",
     color: "#fff",
     padding: "6px 10px",
     borderRadius: "6px",
     fontSize: "12px",
     whiteSpace: "nowrap",
     zIndex: 10, }}>Please log in to add a game </div>)}
  {isManual && (
   <form
    onSubmit={handleManualSubmit}
```

```
          style={{ display: "flex", flexDirection: "column", gap: "15px" }}
        >{[
          { label: "Slug", key: "slug", type: "text", placeholder: "grand-theft-auto-v" },
          { label: "Name", key: "name", type: "text", placeholder: "Grand Theft Auto V" },
          { label: "Description", key: "description", type: "textarea", placeholder: "Game description..." },
          { label: "Image URL", key: "background_image", type: "url", placeholder:
"https://example.com/game.jpg" },
          { label: "Genres (comma separated)", key: "genres", type: "text", placeholder: "Action, Adventure"
},
          { label: "Platforms (comma separated)", key: "platforms", type: "text", placeholder: "PC, PS4,
Xbox One" },
          { label: "Rating", key: "rating", type: "number", placeholder: "4.5", min: 0, max: 5, step: 0.01 },
          { label: "Released Date", key: "released", type: "date", max: new Date().toISOString().split("T")[0]
},
          { label: "Website", key: "website", type: "url", placeholder: "https://example.com" },
        ].map((field) => (
          <label key={field.key} style={{ color: "#fff", fontWeight: "bold" }}>
            {field.label}
            {field.type === "textarea" ? (
              <textarea
                value={manualData[field.key]}
                onChange={(e) => setManualData({ ...manualData, [field.key]: e.target.value })}
                placeholder={field.placeholder || ""}
                required
                style={{
                  width: "100%",
                  padding: "10px",
                  borderRadius: "6px",
                  border: "none",
                }}
              />
            ) : (
```

```
      <input
        type={field.type}
        value={manualData[field.key]}
        onChange={(e) => setManualData({ ...manualData, [field.key]: e.target.value })}
        placeholder={field.placeholder || ""}
        required
        min={field.min}
        max={field.max}
        step={field.step}
        maxLength={field.maxLength}
        style={{
          width: "100%",
          padding: "10px",
          borderRadius: "6px",
          border: "none",
        }}
      />
    )}
  </label>
))}
<button
  type="submit"
  style={{
    padding: "12px",
    borderRadius: "8px",
    border: "none",
    backgroundColor: "#007bff",
    color: "#fff",
    fontWeight: "bold",
    cursor: "pointer",
  }}>  Submit Manually<button></form>
```

```
      )}
    </div>
  </div>
 );
};
export default AddYours;
```

## Admin.jsx

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./font.css";
const Admin = () => {
 const [pendingGames, setPendingGames] = useState([]);
 const [loading, setLoading] = useState(true);
 const [error, setError] = useState("");
 const [isAdmin, setIsAdmin] = useState(false);
 const [token, setToken] = useState("");
 const [editingGameId, setEditingGameId] = useState(null);
 const [editedData, setEditedData] = useState({});
 useEffect(() => {
  const user = JSON.parse(localStorage.getItem("user"));
  const storedToken = localStorage.getItem("token");
  if (!user || !storedToken) {
   setError("Please login to access admin panel");
   setLoading(false);
   return;
  }
  if (!user.isAdmin) {
   setError("Admin access required");
```

```
    setLoading(false);
    return;
  }
  setIsAdmin(true);
  setToken(storedToken);
  fetchPendingGames(storedToken);
}, []);

const fetchPendingGames = async (token) => {
  try {
    const response = await axios.get(
      "http://localhost:3000/admin/pending-games",
      { headers: { Authorization: `Bearer ${token}` } }
    );
    setPendingGames(response.data);
    setLoading(false);
  } catch (err) {
    console.error("Error fetching pending games:", err);
    setError(err.response?.data?.error || "Failed to fetch pending games");
    setLoading(false);
  }
};
const handleApprove = async (gameId) => {
  try {
    await axios.post(
      `http://localhost:3000/admin/approve-game/${gameId}`,
      {},
      { headers: { Authorization: `Bearer ${token}` } }
    );
    toast.success("Game approved successfully!");
    setPendingGames(pendingGames.filter((game) => game._id !== gameId));
```

```
  } catch (err) { console.error(err);
    toast.error("Failed to approve game"); } };
const handleReject = async (gameId, notes = "") => {
  try {
    await axios.post(
      `http://localhost:3000/admin/reject-game/${gameId}`,
      { adminNotes: notes },
      { headers: { Authorization: `Bearer ${token}` } }
    );
    toast.success("Game rejected successfully!");
    setPendingGames(pendingGames.filter((game) => game._id !== gameId));
  } catch (err) {
    console.error(err);
    toast.error("Failed to reject game");
  }
};
const handleEditClick = (game) => {
  setEditingGameId(game._id);
  setEditedData({
    name: game.name,
    slug: game.slug,
    description: game.description,
    background_image: game.background_image,
    genres: game.genres
      ?.map((g) => (typeof g === "string" ? g : g.name))
      .join(", "),
    platforms: game.platforms
      ?.map((p) =>
        typeof p === "string" ? p : p.name || p.platform?.name
      )
      .join(", "),
```

```
    rating: game.rating,
    released: game.released?.split("T")[0] || "",
    website: game.website,
  });
};
const handleSaveEdit = async (gameId) => {
  try {
    const updatedGame = {
      ...editedData,
      genres: editedData.genres
        .split(",")
        .map((g) => g.trim())
        .filter(Boolean),
      platforms: editedData.platforms
        .split(",")
        .map((p) => p.trim())
        .filter(Boolean),
    };
    await axios.put(
      `http://localhost:3000/admin/edit-game/${gameId}`,
      { updatedGame },
      { headers: { Authorization: `Bearer ${token}` } }
    );
    toast.success("Game updated successfully!");
    setPendingGames((prev) =>
      prev.map((g) => (g._id === gameId ? { ...g, ...updatedGame } : g))
    );
    setEditingGameId(null);
  } catch (err) {
    console.error(err);
    toast.error("Failed to update game");
```

```
  }
};
const handleChange = (e) => {
  const { name, value } = e.target;
  setEditedData((prev) => ({ ...prev, [name]: value }));
};
const formatDate = (dateString) =>
  new Date(dateString).toLocaleDateString("en-US", {
    year: "numeric",
    month: "short",
    day: "numeric",
    hour: "2-digit",
    minute: "2-digit",
  });
if (loading)
  return (
    <div
      style={{
        minHeight: "100vh",
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
        color: "#fff",
        backgroundColor: "#101014",
      }}
    >
      Loading pending games...
    </div>
  );
if (error)
  return (
```

```
    <div
      style={{
        minHeight: "100vh",
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
        color: "#ff6b6b",
        backgroundColor: "#101014",
      }}
    >
      {error}
    </div>
  );
return (
  <div
    style={{
      minHeight: "100vh",
      padding: "80px 20px",
      backgroundColor: "#101014",
      color: "#fff",
    }}
  >
    <ToastContainer />
    <div style={{ maxWidth: "1200px", margin: "0 auto" }}>
      <h1
        style={{
          textAlign: "center",
          marginBottom: "30px",
          color: "#ABDBF1",
        }}
      >
```

```
  Admin Panel - Pending Games
</h1>
{pendingGames.length === 0 ? (
  <div
    style={{
      textAlign: "center",
      padding: "40px",
      backgroundColor: "#1a1a1a",
      borderRadius: "12px",
      border: "1px solid #333",
    }}
  >
    <h3 style={{ color: "#ABDBF1" }}>No pending games</h3>
    <p style={{ color: "#888" }}>All submissions reviewed.</p>
  </div>
) : (
  <div
    style={{
      display: "flex",
      flexDirection: "column",
      gap: "20px",
    }}
  >
    {pendingGames.map((game) => (
      <div
        key={game._id}
        style={{
          backgroundColor: "#1a1a1a",
          border: "1px solid #333",
          borderRadius: "12px",
          padding: "20px",
```

```
    display: "flex",
    gap: "20px",
    flexWrap: "wrap",
  }}
>
  <div style={{ width: "200px", height: "150px", flexShrink: 0 }}>
    <img
      src={game.background_image || "/placeholder-game.jpg"}
      alt={game.name}
      style={{
        width: "100%",
        height: "100%",
        objectFit: "cover",
        borderRadius: "8px",
      }}
    />
  </div>
  <div style={{ flex: 1, minWidth: "300px" }}>
    {editingGameId === game._id ? (
      <div
        style={{
          display: "flex",
          flexDirection: "column",
          gap: "10px",
        }}
      >
        {[
          { label: "Name", key: "name" },
          { label: "Slug", key: "slug" },
          {
            label: "Description",
```

```
    key: "description",
    type: "textarea",
  },
  { label: "Image URL", key: "background_image" },
  { label: "Genres", key: "genres" },
  { label: "Platforms", key: "platforms" },
  { label: "Rating", key: "rating", type: "number" },
  {
    label: "Released Date",
    key: "released",
    type: "date",
    max: new Date().toISOString().split("T")[0],
  },
  { label: "Website", key: "website" },
].map((field) => (
  <div key={field.key}>
    <label
      style={{
        fontWeight: "bold",
        display: "block",
        marginBottom: "4px",
      }}
    >
      {field.label}
    </label>
    {field.type === "textarea" ? (
      <textarea
        name={field.key}
        value={editedData[field.key]}
        onChange={handleChange}
        style={{
```

```
            width: "100%",
            padding: "6px",
            borderRadius: "6px",
            border: "none",
          }}
        />
      ) : (
        <input
          type={field.type || "text"}
          name={field.key}
          value={editedData[field.key]}
          onChange={handleChange}
          style={{
            width: "100%",
            padding: "6px",
            borderRadius: "6px",
            border: "none",
          }}
          max={field.max || undefined}
          step={field.step || undefined}
        />
      )}
    </div>
  ))}
  <div
    style={{
      display: "flex",
      gap: "10px",
      marginTop: "10px",
    }}
  >
```

```
          <button
            onClick={() => handleSaveEdit(game._id)}
            style={{
              backgroundColor: "#007bff",
              color: "#fff",
              border: "none",
              padding: "8px 16px",
              borderRadius: "6px",
              cursor: "pointer",
            }}
          >
            Save
          </button>
          <button
            onClick={() => setEditingGameId(null)}
            style={{
              backgroundColor: "#6c757d",
              color: "#fff",
              border: "none",
              padding: "8px 16px",
              borderRadius: "6px",
              cursor: "pointer",
            }}
          >
            Cancel
          </button>
        </div>
      </div>
    ) : (
      <div>
        <h3
```

```
  style={{
    margin: "0 0 10px 0",
    color: "#ABDBF1",
    fontSize: "1.5rem",
  }}
>
  {game.name}
</h3>
<p style={{ margin: "0 0 5px 0", color: "#888" }}>
  Submitted by:{" "}
  <span style={{ color: "#fff" }}>
    {game.submittedBy}
  </span>
</p>
<p style={{ margin: "0 0 5px 0", color: "#888" }}>
  Submitted: {formatDate(game.submittedAt)}
</p>
<p style={{ margin: "0 0 5px 0", color: "#888" }}>
  Genres:{" "}
  {game.genres
    ?.map((g) =>
      typeof g === "string" ? g : g.name
    )
    .join(", ")}
</p>
<p style={{ margin: "0 0 5px 0", color: "#888" }}>
  Platforms:{" "}
  {game.platforms
    ?.map((p) =>
      typeof p === "string"
        ? p
```

```
        : p.name || p.platform?.name
    )
    .join(", ")}
</p>
<p style={{ margin: "0 0 10px 0", color: "#ccc" }}>
  {game.description}
</p>
<p style={{ margin: "0 0 10px 0", color: "#ccc" }}>
  Rating: {game.rating}
</p>
<p style={{ margin: "0 0 10px 0", color: "#ccc" }}>
  Released: {game.released?.split("T")[0]}
</p>
<p style={{ margin: "0 0 10px 0", color: "#ccc" }}>
  Website:{" "}
  <a
    href={game.website}
    target="_blank"
    rel="noopener noreferrer"
  >
    {game.website}
  </a>
</p>
<div style={{ display: "flex", gap: "10px" }}>
  <button
    onClick={() => handleApprove(game._id)}
    style={{
      backgroundColor: "#28a745",
      color: "#fff",
      border: "none",
      padding: "10px 20px",
```

```
    borderRadius: "6px",
    cursor: "pointer",
  }}
>
  Approve
</button>
<button
  onClick={() => {
    const notes = prompt(
      "Enter rejection reason (optional):"
    );
    handleReject(game._id, notes);
  }}
  style={{
    backgroundColor: "#dc3545",
    color: "#fff",
    border: "none",
    padding: "10px 20px",
    borderRadius: "6px",
    cursor: "pointer",
  }}
>
  Reject
</button>
<button
  onClick={() => handleEditClick(game)}
  style={{
    backgroundColor: "#ffc107",
    color: "#000",
    border: "none",
    padding: "10px 20px",
```

```
                    borderRadius: "6px",
                    cursor: "pointer",
                  }}
                >
                  Edit
                </button>
              </div>
            </div>
          )}
        </div>
      ))}
    </div>
  )}
 </div>
</div>
);
};
export default Admin;
```

## bigcard.jsx

```
import React, { useState, useEffect } from "react";
import "./bigcard.css";
import Rating from "./rating.jsx";
import axios from "axios";

const MyComponent = (props) => {
  const [isExpanded, setIsExpanded] = useState(false);
  const [reviews, setReviews] = useState([]);
  const [loadingReviews, setLoadingReviews] = useState(false);
```

```
const gameId = props.id;


const countWords = (text) => {
  if (!text) return 0;
  return text.trim().split(/\s+/).length;
};
const getFirstWords = (text, wordCount = 50) => {
  if (!text) return "";
  const words = text.trim().split(/\s+/);
  return words.slice(0, wordCount).join(" ");
};
const description = props.description || "";
const wordCount = countWords(description);
const shouldShowReadMore = wordCount > 50;
const displayDescription =
  isExpanded || !shouldShowReadMore
    ? description
    : getFirstWords(description, 50);


// Fetch reviews for this specific game
const fetchReviews = async () => {
  if (!gameId) return;


  setLoadingReviews(true);
  try {
    const response = await axios.get(`https://gamessonnet.onrender.com/reviews/${gameId}`);
    setReviews(response.data);
  } catch (error) {
    console.error('Error fetching reviews:', error);
    setReviews([]);
  } finally {
```

```
    setLoadingReviews(false);
  }
};
useEffect(() => {
  fetchReviews();
}, [gameId]);

const handleReviewSubmitted = () => {
  fetchReviews();
};
return (
  <div
    style={{
      width: "100%",
      height: window.innerWidth < 768 ? "90vh" : "80vh",
      border: "1px solid #ccc",
      borderRadius: "8px",
      overflow: "auto",
      backgroundColor: "#101014",
      scrollbarWidth: "none", // Firefox
      msOverflowStyle: "none", // IE 10+
    }}
    className="hide-scrollbar"
  >
    {/* Game Image */}
    <img
      src={props.image}
      alt={props.title}
      style={{
        width: "100%",
        height: "40%",
```

```
    objectFit: "contain",
    backgroundColor: "#090B0D",
  }}
/>
<div style={{
  padding: window.innerWidth < 768 ? "15px" : "20px",
  color: "#AABFCB"
}}>
  <h2
    style={{
      fontFamily: "Playfair Display",
      fontSize: window.innerWidth < 768 ? "24px" : "30px",
      marginBottom: window.innerWidth < 768 ? "15px" : "20px",
      lineHeight: "1.2",
      wordBreak: "break-word"
    }}
  >
    {props.title}
  </h2>
  <div
    style={{
      fontSize: "20px",
      marginBottom: "20px",
      fontFamily: "Playfair Display",
    }}
  >
    <div dangerouslySetInnerHTML={{ __html: displayDescription }} />
    {shouldShowReadMore && (
      <span
        onClick={() => setIsExpanded(!isExpanded)}
        style={{
```

```
        color: "#4f46e5",
        cursor: "pointer",
        textDecoration: "underline",
        marginLeft: "5px",
        fontSize: "14px",
      }}
    >
      {isExpanded ? "Read less" : "...Read more"}
    </span>
  )}
</div>
<p
  style={{
    fontSize: "20px",
    marginBottom: "20px",
    fontFamily: "Playfair Display",
  }}
>
  <strong>Genre:</strong> {props.genre}
</p>
<p
  style={{
    fontSize: "20px",
    marginBottom: "20px",
    fontFamily: "Playfair Display",
  }}
>
  <strong>Platform:</strong> {props.platform}
</p>
<a
  href={props.website}
```

```
    target="_blank"
    rel="noopener noreferrer"
    style={{
      display: "inline-block",
      padding: "10px 15px",
      backgroundColor: "#4f46e5",
      color: "white",
      textDecoration: "none",
      borderRadius: "5px",
      marginTop: "10px",
      fontFamily: "Playfair Display",
    }}
  >Redirect to the game website </a> </div>
<hr style={{ border: "1px solid #444747", margin: "0" }} />
<div style={{ padding: "10px", margin: "5px" }}>
  <Rating gameId={gameId} onReviewSubmitted={handleReviewSubmitted} />
</div>
<div style={{ padding: "20px", color: "#AABFCB", fontFamily: "Playfair Display" }}>
  <h3>Reviews ({reviews.length})</h3>
  {loadingReviews ? (
    <p>Loading reviews...</p>
  ) : reviews.length === 0 ? (
    <p>Be first and add your review</p>
  ) : (
    <div>
      {reviews.map((review) => (
        <div
          key={review._id}
          style={{
            marginBottom: "15px",
            padding: "10px",
```

```
              backgroundColor: "rgba(255,255,255,0.05)",
              borderRadius: "8px"
            }}
          >
            <div style={{ display: "flex", justifyContent: "space-between", alignItems: "center",
marginBottom: "5px" }}>
              <div style={{ fontWeight: "bold", color: "#61dafb" }}>
                {review.username}
              </div>
              <div style={{ color: "#FFD700" }}>
                {"★".repeat(review.rating)}{"☆".repeat(5 - review.rating)}
              </div>
            </div>
            <div style={{ marginBottom: "5px" }}>
              {review.reviewText}
            </div>
            <div style={{ fontSize: "12px", color: "#888" }}>
              {new Date(review.createdAt).toLocaleDateString()}
            </div>
          </div>
        ))}
      </div>
    )}
  </div>
 );
};
export default MyComponent;
```

## cardholder.jsx

```
import React, { useState, useEffect } from 'react';
```

```
import './GameCard.css';
import axios from 'axios';

const GameCard = ({ game, onClick }) => {
  const [reviewsCount, setReviewsCount] = useState(0);

  useEffect(() => {
    const fetchReviewsCount = async () => {
      if (game?.id) {
        try {
          const response = await axios.get(`https://gamessonnet.onrender.com/reviews-count/${game.id}`);
          setReviewsCount(response.data.count);
        } catch (error) {
          console.error('Error fetching reviews count:', error);
          setReviewsCount(0);
        }
      }
    };

    fetchReviewsCount();
  }, [game?.id]);
  return (
    <div className="game-card" onClick={onClick}>
      <div className="image-placeholder">
        {game?.background_image ? (
          <img src={game.background_image} alt={game.name} style={{width: '100%', height: '100%',
objectFit: 'cover'}} />
        ) : ('Game Image')}
      </div>
      <div className="card-title">
        <h2>{game?.name || 'Game Title'}</h2>
```

```
    </div>
    <div className="card-details">
      <div className="detail-row">
        <span className="label">Genre:</span>
        <span className="value">{game?.genres?.[0]?.name || 'Action'}</span>
      </div>
      <div className="detail-row">
        <span className="label">Rating:</span>
        <span className="value">{game?.rating || '4.5'}</span>
      </div>
      <div className="detail-row">
        <span className="label">Reviews:</span>
        <span className="value">({reviewsCount})</span> </div>  </div>  </div>
  );
};
export default GameCard;
```

## Login.jsx

```
import React, { useState, useEffect } from "react";
import { useForm } from "react-hook-form";
import { NavLink, useNavigate } from "react-router-dom";

const Login = () => {
  const [data, setData] = useState("");
  const navigate = useNavigate();

  const {
    register,
    handleSubmit,
    formState: { errors, isSubmitting },
  } = useForm();
```

```
useEffect(() => {
  localStorage.removeItem("user");
  localStorage.removeItem("token");
}, []);
const onSubmit = async (formData) => {
  try {
    const response = await fetch("http://localhost:3000/login", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        usernameOrEmail: formData.username,
        password: formData.password,
      }),  });
    if (!response.ok) {
      const error = await response.json();
      alert(error.error || "Login failed");
      return;
    }
    const res = await response.json();
    console.log(res);
    setData(JSON.stringify(res));
    localStorage.setItem("token", res.token);
    localStorage.setItem("user", JSON.stringify(res.user));
    setTimeout(() => {
      navigate("/");
    }, 1000);
  } catch (error) {
    console.error("Login error:", error);
    alert("Something went wrong. Is your backend running?");
  }
};
```

```
return (
  <div
    style={{
      minHeight: "100vh",
      backgroundColor: "#101014",
      backgroundSize: "cover",
      backgroundPosition: "center",
      backgroundRepeat: "no-repeat",
      display: "flex",
      justifyContent: "center",
      alignItems: "flex-start",
      paddingTop: "80px",
      overflow: "hidden",
      position: "relative",
    }}
  >
    <div
      style={{
        width: "100%",
        maxWidth: "400px",
        backgroundColor: "rgba(39, 47, 53, 0.9)",
        borderRadius: "12px",
        padding: "30px",
        boxShadow: "0 0 30px rgba(255, 255, 255, 0.1)",
        backdropFilter: "blur(1px)",
        backgroundSize: "600px",
        backgroundPosition: "center",
      }}
    >
      <h2
        style={{
```

```
    color: "whitesmoke",
    textAlign: "center",
    marginBottom: "24px",
    textShadow: `
      0px 0px 8px red,
      0px 0px 16px red,
      0px 0px 30px maroon,
    `,
  }}
>
  Login
</h2>

<form onSubmit={handleSubmit(onSubmit)}>
  <div style={{ marginBottom: "20px" }}>
    <input
      {...register("username", {
        required: "Username or Email is required",
      })}
      type="text"
      placeholder="Username or Email"
      style={{
        width: "100%",
        padding: "10px",
        backgroundColor: "#1e242a",
        border: "1px solid #444",
        borderRadius: "6px",
        color: "#ABDBF1",
      }}
    />
    {errors.username && (
```

```
    <p style={{ color: "red", fontSize: "14px" }}>
      {errors.username.message}
    </p>
  )}
</div>

<div style={{ marginBottom: "20px" }}>
  <input
    {...register("password", { required: "Password is required" })}
    type="password"
    placeholder="Password"
    style={{
      width: "100%",
      padding: "10px",
      backgroundColor: "#1e242a",
      border: "1px solid #444",
      borderRadius: "6px",
      color: "white",
    }}
  />
  {errors.password && (
    <p style={{ color: "red", fontSize: "14px" }}>
      {errors.password.message}
    </p>
  )}
</div>

<button
  type="submit"
  disabled={isSubmitting}
  style={{
```

```
      width: "100%",
      backgroundColor: "#007bff",
      color: "#fff",
      padding: "10px",
      border: "none",
      borderRadius: "6px",
      fontWeight: "bold",
      cursor: isSubmitting ? "not-allowed" : "pointer",
      opacity: isSubmitting ? 0.6 : 1,
    }}
  >
    {isSubmitting ? "Logging in..." : "Login"}
  </button>
</form>

{data && (
  <div
    style={{
      marginTop: "20px",
      textAlign: "center",
      color: "#28a745",
    }}
  >
    <h4>You have successfully logged in</h4>
  </div>
)}

<div style={{ marginTop: "25px", textAlign: "center" }}>
  <NavLink
    to="/signup"
    style={{
```

```
            color: "#61dafb",
            fontSize: "16px",
            textDecoration: "none",
          }}
        >
          Don't have an account? Sign Up
        </NavLink>
      </div>
    </div>
  </div>
 );
};
export default Login;
```

## Navbar.jsx

```
import React, { useState, useEffect } from 'react';
import { NavLink } from 'react-router-dom';
const Navbar = () => {
 const [user, setUser] = useState(null);
 const [showUserInfo, setShowUserInfo] = useState(false);
 const [isMobile, setIsMobile] = useState(false);
 const [showMobileMenu, setShowMobileMenu] = useState(false);

 useEffect(() => {
  const checkScreenSize = () => {
   const isSmall = window.innerWidth < 990;
   setIsMobile(isSmall);
  };
  checkScreenSize();
  window.addEventListener('resize', checkScreenSize);
  return () => window.removeEventListener('resize', checkScreenSize);
```

```
}, []);


useEffect(() => {
  const checkUser = () => {
    const storedUser = localStorage.getItem('user');
    if (storedUser) {
      setUser(JSON.parse(storedUser));
    }
  };

  checkUser();

  const handleStorageChange = () => {
    checkUser();
  };

  window.addEventListener('storage', handleStorageChange);
  const interval = setInterval(checkUser, 1000);

  return () => {
    window.removeEventListener('storage', handleStorageChange);
    clearInterval(interval);
  };
}, []);

useEffect(() => {
  const handleClickOutside = (event) => {
    if (showUserInfo && !event.target.closest('.user-profile-container')) {
      setShowUserInfo(false);
    }
```

```
    if (showMobileMenu && !event.target.closest('.mobile-menu-container')) {
      setShowMobileMenu(false);
    }
  };

  document.addEventListener('mousedown', handleClickOutside);
  return () => {
    document.removeEventListener('mousedown', handleClickOutside);
  };
}, [showUserInfo, showMobileMenu]);

const handleLogout = () => {
  localStorage.removeItem('user');
  localStorage.removeItem('token')
  setUser(null);
  setShowUserInfo(false);
};

const toggleUserInfo = () => {
  setShowUserInfo(!showUserInfo);
};

const toggleMobileMenu = () => {
  setShowMobileMenu(prev => !prev);
};

const navItems = [
  { path: '/', label: 'Home' },
  { path: '/about', label: 'About' },
  { path: '/addyours', label: 'Add Yours' },
  ...(user ? [{ path: '/yoursuggestion', label: 'Your Suggestions' }] : []),
```

```
  ...(user ? [] : [{ path: '/login', label: 'Login' }]),
];

return (
  <div
    style={{
      display: 'flex',
      justifyContent: 'space-between',
      alignItems: 'center',
      padding: window.innerWidth < 768 ? '15px 20px' : '3vw 40px',
      background: '#1A1C1E',
      backdropFilter: 'blur(20px)',
      WebkitBackdropFilter: 'blur(10px)',
      boxShadow: '0 1px 14px 0',
      position: 'relative',
      zIndex: 1,
      flexWrap: 'wrap',
      minHeight: '60px'
    }}
  >
    <div style={{
      fontSize: '2.1em',
      fontWeight: 'bold',
      color: '#ffffff',
      whiteSpace: 'nowrap',
    fontFamily:'-moz-initial',
    }}>
      Games Sonnet
    </div>

    {!isMobile ? (
```

```
<div style={{ display: 'flex', gap: '30px', alignItems: 'center' }}>
 {navItems.map((item) => (
  <NavLink
   key={item.path}
   to={item.path}
   style={({ isActive }) => ({
    paddingTop: '10px',
    textDecoration: 'none',
    color: isActive ? '#00ffe7' : '#ffffff',
    fontSize: '1.2em',
    fontWeight: '500',
    textShadow: isActive ? '0 0 6px #00ffe7, 0 0 16px #00ffe7' : 'none',
    transform: isActive ? 'scale(1.05)' : 'scale(1)',
   })}
  >
   {item.label}
  </NavLink>
 ))}
 {user && (
  <div className="user-profile-container" style={{ position: 'relative' }}>
   <div  onClick={toggleUserInfo}
    style={{width: '40px',height: '40px',borderRadius: '50%',backgroundColor: '#00ffe7',color: '#1A1C1E',display: 'flex',alignItems: 'center',justifyContent: 'center',fontSize: '18px',fontWeight: 'bold',cursor: 'pointer',transition: 'all 0.3s ease',transform: showUserInfo ? 'scale(1.1)' : 'scale(1)',boxShadow: showUserInfo ? '0 0 15px rgba(0, 255, 231, 0.5)' : 'none',}}  >
    {user.username.charAt(0).toUpperCase()}  </div>
   {showUserInfo && (
    <div    style={{position: 'absolute',top: '50px',right: '0',backgroundColor: 'rgba(39, 47, 53, 0.95)', borderRadius: '12px',padding: '20px',minWidth: '200px',boxShadow: '0 8px 32px rgba(0, 0, 0, 0.3)',backdropFilter: 'blur(10px)',border: '1px solid rgba(255, 255, 255, 0.1)',zIndex: 9999,}}  >
     <div style={{ color: '#ffffff', marginBottom: '10px' }}>
      <strong>Username:</strong> {user.username}
```

```
            </div>
            <div style={{ color: '#ffffff', marginBottom: '15px' }}>
              <strong>Email:</strong> {user.email}
            </div>
            <button onClick={handleLogout}  style={{width: '100%', backgroundColor: '#ff4757', color: '#fff',padding: '8px 12px',border: 'none',borderRadius: '6px',cursor: 'pointer',fontSize: '14px',fontWeight: 'bold', }}> Logout </button>
</div>)}
          </div>)}
       </div>
     ) : (
       <div className="mobile-menu-container" style={{ position: 'relative' }}>
         <div  onClick={toggleMobileMenu}  style={{display: 'flex',flexDirection: 'column',cursor: 'pointer',gap: '3px', }}>
           <div  style={{width: '25px',height: '2px',backgroundColor: '#ffffff',transition: 'all 0.3s ease',transform: showMobileMenu ? 'rotate(45deg) translate(5px, 5px)' : 'none',}}
           />
           <div  style={{ width: '25px',height: '2px',backgroundColor: '#ffffff',transition: 'all 0.3s ease',opacity: showMobileMenu ? '0' : '1',
             }}
           />
           <div  style={{width: '25px',height: '2px',backgroundColor: '#ffffff',transition: 'all 0.3s ease', transform: showMobileMenu ? 'rotate(-45deg) translate(7px, -6px)' : 'none',}}
           />
         </div>
         {showMobileMenu && (
<div  style={{position: 'absolute',top: '30px',right: '0',backgroundColor: 'rgba(39, 47, 53, 0.95)',borderRadius: '12px',padding: '30px',minWidth: '250px',boxShadow: '0 8px 32px rgba(0, 0, 0, 0.3)',backdropFilter: 'blur(10px)',border: '1px solid rgba(255, 255, 255, 0.1)', zIndex: 9999,}}  >
             {navItems.map((item) => (
               <NavLink  key={item.path}  to={item.path}
                 onClick={() => setShowMobileMenu(false)}
```

```
            style={({ isActive }) => ({display: 'block',textDecoration: 'none',color: isActive ? '#00ffe7' : '#ffffff',fontSize: '2.3vmax',fontWeight: '500',padding: '8px 10px',borderBottom: '2px solid gray',textShadow: isActive ? '0 0 6px #00ffe7' : 'none',})}  >
              {item.label}
            </NavLink>
          ))}
          {user && (
            <div style={{ borderTop: '1px solid rgba(255, 255, 255, 0.1)', paddingTop: '10px', marginTop: '10px' }}>
              <div style={{ color: '#ffffff', fontSize: '12px', marginBottom: '5px' }}>
                <strong>{user.username}</strong>
              </div>
              <button  onClick={() => {handleLogout(); setShowMobileMenu(false);}}
                style={{width: '100%',backgroundColor: '#ff4757', color: '#fff',padding: '6px 10px',border: 'none',borderRadius: '4px',cursor: 'pointer',  fontSize: '12px',  fontWeight: 'bold',}} > Logout </button>
 </div> )}  </div> )}</div> )}</div> );
};
export default Navbar;
```

## PageNotFound.jsx

```
import React from "react";
import { Link } from "react-router-dom";
import { motion } from "framer-motion";
import { AlertTriangle } from "lucide-react";
import "./PageNotFound.css"; // Import CSS file
const PageNotFound = () => {
  return (
    <div className="page-not-found">
      <motion.div
        initial={{ scale: 0.5, opacity: 0 }}
        animate={{ scale: 1, opacity: 1 }}
        transition={{ duration: 0.6 }}
```

```
      className="content"
    >
      <div className="icon">
        <AlertTriangle size={80} className="alert-icon" />
      </div>
      <h1 className="title">404</h1>
      <p className="subtitle">
        Oops! The page you're looking for doesn't exist.
      </p>
      <Link to="/" className="home-btn"> Go Back Home</Link>
    </motion.div>
  </div>);  };
export default PageNotFound;
```

## Rating.jsx

```
import React, { useState, useEffect } from 'react';
import { Flex, Rate, Button, message } from 'antd';
import axios from 'axios';
const Rating = ({ gameId, onReviewSubmitted }) => {
  const [stars, setStars] = useState(0);
  const [comment, setComment] = useState('');
  const [loading, setLoading] = useState(false);
  const [token, setToken] = useState(localStorage.getItem('token'));
  useEffect(() => {
    const handleStorageChange = () => {
      setToken(localStorage.getItem('token'));
    };
    window.addEventListener('storage', handleStorageChange);
    const interval = setInterval(() => {
      setToken(localStorage.getItem('token'));
    }, 1000);
```

```
  return () => {
    window.removeEventListener('storage', handleStorageChange);
    clearInterval(interval);
  };
}, []);
const handleSubmit = async () => {
  if (!comment.trim()) {
    message.warning('Please enter a comment.');
    return;
  }
  if (stars === 0) {
    message.warning('Please select a rating.');
    return;
  }
  if (!gameId) {
    message.error('Game ID is required to submit a review.');
    return;
  }
  if (!token) {
    message.error('Please login to submit a review.');
    return;
  }
  setLoading(true);
  try {
    await axios.post(
      'https://gamessonnet.onrender.com/add-review',
      {
        gameId: parseInt(gameId),
        reviewText: comment,
        rating: stars,
      },
```

```
      {
        headers: {
          Authorization: token,},  }
      );
      message.success('Thank you for your review!');
      setStars(0);
      setComment('');
      if (onReviewSubmitted) {onReviewSubmitted();}
    } catch (error) {
      console.error('Error submitting review:', error);
      if (error.response?.status === 401) {
 message.error('Please login to submit a review.');
} else { message.error('Failed to submit review. Please try again.');}
    } finally {
      setLoading(false);}
  };
  return (
    <>
      <style>{`.ant-rate-star-zero svg { color: #999999; }`}</style>
      <Flex gap="middle" vertical>
        <Rate value={stars} onChange={setStars} />
        <textarea
          rows={4}
          placeholder="Tell us what you think..."value={comment}  onChange={(e) =>
setComment(e.target.value)}  style={{ width: '96%', padding: '8px', borderRadius: '4px', resize: 'none' }}  />
        {!token && (
          <p style={{ color: 'red', fontWeight: '500' }}>
            Please login to submit a review.
          </p>)}
        <Button
          type="primary"
```

```
        onClick={handleSubmit}
        loading={loading}
        disabled={!comment.trim() || stars === 0 || !token}>
        Submit Review
      </Button>
    </Flex>
  </>
 );
};
export default Rating;
```

## Signup.jsx

```
import React, { useState } from 'react';
import { useForm } from 'react-hook-form';
import { NavLink, useNavigate } from 'react-router-dom';
const Signup = () => {
 const [data, setData] = useState("");
 const navigate = useNavigate();
 const {
  register,
  handleSubmit,
  formState: { errors, isSubmitting },
 } = useForm();
 const onSubmit = async (formData) => {
  const { username, email, password } = formData;
  try {
   const response = await fetch("https://gamessonnet.onrender.com/register", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username, email, password }),
   });
   if (!response.ok) {
```

```
      const error = await response.json();
      alert(error.error || "Signup failed");
      return;
    }
    const data = await response.json();
    alert("Signup successful!");
  } catch (error) {
    console.error("Signup error:", error);
    alert("Something went wrong during signup.");  };
return (
  <div
    style={{
      minHeight: "100vh",
      backgroundColor: "#101014",
      backgroundSize: "cover",
      backgroundPosition: "center",
      backgroundRepeat: "no-repeat",
      display: "flex",
      justifyContent: "center",
      alignItems: "flex-start",
      paddingTop: "80px",
      position: "relative",
      overflow: "hidden",
    }}
  >
    <div
      style={{
        padding: "2px",
        borderRadius: "16px",
        background: "radial-gradient(circle at center, rgba(255,255,255,0.25), rgba(255,255,255,0) 80%)",
      }}
```

```
    >
<div
        style={{width: "450px",maxWidth: "400px",backgroundColor: "rgba(39, 47, 53, 0.9)",borderRadius: "12px",padding: "30px",boxShadow: "0 0 30px rgba(255, 255, 255, 0.1)",backdropFilter: "blur(10px)",}}
      >
        <h2 style={{ color: "#ffffff", textAlign: "center", marginBottom: "24px" }}>Sign Up</h2>
        <form onSubmit={handleSubmit(onSubmit)}>
          <div style={{ marginBottom: "20px" }}>
            <input
              {...register("username", {
                required: "Username is required",
                pattern: {
                  value: /^[A-Za-z]+$/,
                  message: "Username must contain only letters"
                }
              })}
              type="text"
              placeholder="Username"
              style={{width: "100%",padding: "10px",backgroundColor: "#1e242a",border: "1px solid #444",borderRadius: "6px",color: "#ABDBF1", }}  inputMode="text"  onKeyDown={(e) => {  if (/\d/.test(e.key)) {
                  e.preventDefault();  }  }}
            />
            {errors.username && <p style={{ color: "red", fontSize: "14px" }}>{errors.username.message}</p>}
          </div>
          <div style={{ marginBottom: "20px" }}>
<input  {...register("email", { required: "Email is required" })}  type="email" placeholder="Email"
              style={{width: "100%",padding: "10px",backgroundColor: "#1e242a", border: "1px solid #444",borderRadius: "6px",color: "#ABDBF1",
              }}
            />
```

```
          {errors.email && <p style={{ color: "red", fontSize: "14px" }}>{errors.email.message}</p>}
        </div>
        <div style={{ marginBottom: "20px" }}>
          <input {...register("password", { required: "Password is required" })}
            type="password"
            placeholder="Password"
            style={{width: "100%",padding: "10px",backgroundColor: "#1e242a",border: "1px solid #444",borderRadius: "6px",color: "white",
            }}
          />
          {errors.password && <p style={{ color: "red", fontSize: "14px" }}>{errors.password.message}</p>}
        </div>
        <button  type="submit"  disabled={isSubmitting}
          style={{width: "100%",backgroundColor: "#28a745",color: "#fff",padding: "10px",borderRadius: "6px",fontWeight: "bold",cursor: isSubmitting ? "not-allowed" : "pointer",opacity: isSubmitting ? 0.6 : 1,}}>
          {isSubmitting ? "Creating Account..." : "Sign Up"}
        </button>
      </form>
      {data && (
        <div style={{ marginTop: "20px", textAlign: "center", color: "#28a745" }}>
          <h4>Account created successfully!</h4>
        </div>
      )}
      <div style={{ marginTop: "25px", textAlign: "center" }}>
        <NavLink to="/login" style={{ color: "#61dafb", fontSize: "16px", textDecoration: "none" }}>
          Already have an account? Login
        </NavLink></div> </div> </div>  </div> );  };
export default Signup;
```

## YourSuggestion.jsx

```
import React, { useEffect, useState, useCallback } from "react";
import axios from "axios";
```

```
import CardHolder from "../components/cardholder.jsx";
import MyComponent from "../components/bigcard.jsx";
const Yoursuggestion = () => {
  const [games, setGames] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [selectedGame, setSelectedGame] = useState(null);
  const fetchGameDetails = useCallback(async (slug) => {
    try {
      const response = await axios.get(
        `https://api.rawg.io/api/games/${slug}?key=${apiKey}`
      );
      setSelectedGame(response.data);
    } catch (apiError) {
      console.warn("RAWG API failed, trying DB fallback...", apiError);
      try {
        const dbResponse = await axios.get(
          `http://localhost:3000/game-details/${slug}`
        );
        setSelectedGame(dbResponse.data);
      } catch (dbError) {
        console.error("DB fallback failed:", dbError);
        alert("Failed to fetch game details from both RAWG API and database.");
      }
    }
  }, []);
  const handleSuggestedCardClick = useCallback(
    (game) => {
      if (!game) return;
      if (game.slug) {
        fetchGameDetails(game.slug);
```

```
    } else {
      setSelectedGame({
        id: game.id,
        name: game.name,
        description_raw: game.description || "No description available",
        background_image: game.background_image || "",
        genres: game.genres || [],
        platforms: game.platforms || [],
        website: game.website || "",
      });
    }
  },
  [fetchGameDetails]
);
useEffect(() => {
  const fetchGames = async () => {
    try {
      const token = localStorage.getItem("token")?.trim();
      if (!token) throw new Error("Token missing. Please login first.");
      const response = await axios.get("http://localhost:3000/yoursuggested", {
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "application/json",
        },
        timeout: 10000,
      });
      setGames(response.data);
    } catch (err) {
      console.error("Error fetching games:", err.message);
      setError(
        err.response?.data?.error || err.message || "Failed to fetch games"
```

```
      );
    } finally {
      setLoading(false);
    }
  };
  fetchGames();
}, []);
return (
  <div>
    <h3>Your Suggestions</h3>
    {loading && <p>Loading...</p>}
    {!loading && error && <p style={{ color: "red" }}>{error}</p>}
    {!loading && !error && games.length === 0 && (
      <p>No games suggested yet.</p>
    )}
    {!loading && !error && games.length > 0 && (
      <div
        style={{
          display: "grid",
          gridTemplateColumns: "repeat(2,1fr)",
          padding: "20px",
        }}
      >
        {games.map((game) => (
          <CardHolder
            key={game.id}
            game={game}
            onClick={() => handleSuggestedCardClick(game)}
          />
        ))}
      </div>
```

```
)}
{selectedGame && (
 <div
  style={{
   position: "fixed",
   top: 0,
   left: 0,
   width: "100%",
   height: "100%",
   backgroundColor: "rgba(0, 0, 0, 0.8)",
   display: "flex",
   justifyContent: "center",
   alignItems: "center",
   zIndex: 1000,
   padding: window.innerWidth < 768 ? "10px" : "20px",
   boxSizing: "border-box",
  }}
  onClick={() => setSelectedGame(null)}
 >
  <div
   style={{
    width: "100%",
    maxWidth: window.innerWidth < 768 ? "100%" : "800px",
    position: "relative",
    maxHeight: "90vh",
    overflow: "auto",
    borderRadius: "12px",
    backgroundColor: "#1A1C1E",
   }}
   onClick={(e) => e.stopPropagation()}
  >
```

```
          <MyComponent
            id={selectedGame.id}
            image={selectedGame.background_image}
            title={selectedGame.name}
            description={selectedGame.description_raw}
            genre={
              Array.isArray(selectedGame.genres)
                ? selectedGame.genres
                    .map((g) => (typeof g === "string" ? g : g.name))
                    .join(", ")
                : "N/A"
            }
            platform={
              Array.isArray(selectedGame.platforms)
                ? selectedGame.platforms
                    .map((p) =>
                      typeof p === "string"
                        ? p
                        : p.platform?.name || p.name
                    )
                    .join(", ")
                : "N/A"
            }
            website={selectedGame.website}
          />
        </div>
      </div>
    )}
  </div>
 );
};
```

```
export default Yoursuggestion;
```

## App.jsx

```
import React, { useState } from "react";
import "./App.css";
import { createBrowserRouter, RouterProvider, Outlet } from "react-router-dom";
import Navbar from "../public/components/Navbar";
import Home from "../public/components/home";
import About from "../public/components/about";
import AddYours from "../public/components/AddYours";
import Login from "../public/components/login";
import Signup from "../public/components/signup";
import PageNotFound from "../public/components/PageNotFound";
import Yoursuggestion from "../public/components/Yoursuggestion";
import Admin from "../public/components/Admin";

function RootLayout() {
  return (
    <>
      <Navbar />
      <Outlet /> {/* ✅ children routes render here */}
    </>
  );
}
export default function App() {
  const [cachedGames, setCachedGames] = useState([]);
  const [cachedSuggestedGames, setCachedSuggestedGames] = useState([]);
  const [newSuggestedGame, setNewSuggestedGame] = useState({});

  const router = createBrowserRouter([
    {
```

```
    path: "/",
    element: <RootLayout />,
    errorElement: (
      <>
        <Navbar />
        <PageNotFound />
      </>
    ),
    children: [
      {
        index: true, // default route "/"
        element: (
          <Home
            cachedGames={cachedGames}
            setCachedGames={setCachedGames}
            cachedSuggestedGames={cachedSuggestedGames}
            setCachedSuggestedGames={setCachedSuggestedGames}
            newSuggestedGame={newSuggestedGame}
          />
        ),
      },
      { path: "about", element: <About /> },
      { path: "addyours", element: <AddYours setNewSuggestedGame={setNewSuggestedGame} /> },
      { path: "login", element: <Login /> },
      { path: "signup", element: <Signup /> },
      {path :"yoursuggestion",element:<Yoursuggestion/>},
      {path :"admin",element:<Admin/>},
    ],
  },
]);
```

```
  return <RouterProvider router={router} />;
}
```

### Index.jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'

ReactDOM.createRoot(document.getElementById('root')).render(
                <App />
)
```

## Backend:-

### Server.js

```
import express from "express";
import cors from "cors";
import bodyParser from "body-parser";
import mongoose from "mongoose";
import dotenv from "dotenv";
import axios from "axios";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";
dotenv.config();
const app = express();
const port = process.env.PORT || 3000;
app.use(cors({
  origin: "*",
  methods: ["GET","POST","PUT","DELETE"],
  allowedHeaders: ["Content-Type","Authorization"],
}));
app.use(bodyParser.json());
```

```
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
  .then(() => console.log("✅ MongoDB connected"))
  .catch(err => console.error("❌ MongoDB connection error:", err));
const UserSchema = new mongoose.Schema({
  username: { type: String, unique: true },
  email: { type: String, unique: true },
  password: String,
  isAdmin: { type: Boolean, default: false },
});
const GameSchema = new mongoose.Schema({
  id: Number,
  slug: String,
  name: String,
  description: String,
  background_image: String,
  genres: Array,
  platforms: Array,
  rating: Number,
  released: String,
  addedBy: [String],  // array of usernames
  addedAt: Date,
  website: String,
});

const ReviewSchema = new mongoose.Schema({
  gameId: { type: Number, required: true },
  username: { type: String, required: true },
  reviewText: { type: String, required: true },
```

```
  rating: { type: Number, required: true, min: 1, max: 5 },
  createdAt: { type: Date, default: Date.now },
});
const PendingGameSchema = new mongoose.Schema({
  id: Number,
  slug: String,
  name: String,
  description: String,
  background_image: String,
  genres: Array,
  platforms: Array,
  rating: Number,
  released: String,
  submittedBy: [String],  // username who submitted
  submittedAt: { type: Date, default: Date.now },
  website: String,
  status: { type: String, enum: ['pending', 'approved', 'rejected'], default: 'pending' },
  adminNotes: String,
});

const User = mongoose.model("User", UserSchema);
const Game = mongoose.model("Game", GameSchema);
const Review = mongoose.model("Review", ReviewSchema);
const PendingGame = mongoose.model("PendingGame", PendingGameSchema);

// JWT Verification Middleware
const verifyToken = (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader) return res.status(401).json({ error: "Token missing" });

  const token = authHeader.split(" ")[1]; // "Bearer <token>"
```

```
  if (!token) return res.status(401).json({ error: "Token missing" });
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    return res.status(401).json({ error: "Invalid token" });
  }
};
// Admin Verification Middleware
const verifyAdmin = (req, res, next) => {
  if (!req.user.isAdmin) {
    return res.status(403).json({ error: "Admin access required" });
  }
  next();
};
app.post("/register", async (req, res) => {
  const { username, email, password } = req.body;
  if (!username || !email || !password)
    return res.status(400).json({ error: "All fields are required" });

  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new User({ username, email, password: hashedPassword });
    await user.save();
    res.status(201).json({ message: "User registered successfully" });
  } catch (err) {
    res.status(400).json({ error: "Username already taken or invalid data" });
  }
});
app.post("/login", async (req, res) => {
```

```
  const { usernameOrEmail, password } = req.body;
  if (!usernameOrEmail || !password)
    return res.status(400).json({ error: "All fields are required" });
  try {
    const user = await User.findOne({
      $or: [{ email: usernameOrEmail }, { username: usernameOrEmail }],
    });
    if (!user || !(await bcrypt.compare(password, user.password)))
      return res.status(401).json({ error: "Invalid credentials" });
    const isAdmin = user.isAdmin || false;
    const token = jwt.sign(
      { id: user._id, username: user.username, email: user.email, isAdmin },
      process.env.JWT_SECRET,
      { expiresIn: "7d" }
    );
    res.json({
      message: "Login successful",
      token,
      user: { username: user.username, email: user.email, isAdmin }
    });
  } catch (err) {
    res.status(500).json({ error: "Login failed" });
  }
});
app.get("/check-user", async (req, res) => {
  const login = req.query.login;
  if (!login) return res.status(400).json({ error: "Login query is required" });
  try {
    const user = await User.findOne({
      $or: [{ email: login }, { username: login }],
    });
```

```
    if (!user) return res.status(404).json({ error: "User not found" });
    res.json({ username: user.username, email: user.email, isAdmin: user.isAdmin });
  } catch (error) {
    res.status(500).json({ error: "Server error" }); }
});
app.post("/save-game", verifyToken, async (req, res) => {
  const { gameData } = req.body;
  if (!gameData) return res.status(400).json({ error: "Game data required" });
  const {
    id,
    slug,
    name,
    description,
    background_image,
    genres = [],
    platforms = [],
    rating = 0,
    released = "",
    website = ""
  } = gameData;
  if (!id || !slug || !name || !background_image) {
    return res.status(400).json({ error: "Missing required game fields" });
  }
  try {
    const existingGame = await Game.findOne({ id });
    const existingPendingGame = await PendingGame.findOne({ id, status: 'pending' });
    if (existingGame) {
      if (!existingGame.addedBy.includes(req.user.username)) {
        existingGame.addedBy.push(req.user.username);
        await existingGame.save();
      }
```

```
      return res.status(200).json({ message: "Game already exists. Your username added as contributor." });
    }
    if (existingPendingGame) {
      return res.status(200).json({ message: "Game is already pending admin approval." });
    }
    const pendingGame = new PendingGame({
      id,
      slug,
      name,
      description: description || "",
      background_image,
      genres,
      platforms,
      rating,
      released,
      submittedBy: req.user.username,
      website,
    });
    await pendingGame.save();
    res.status(201).json({ message: "Game submitted for admin approval" });
  } catch (err) {
    console.error("Save-game error:", err); // log exact error
    res.status(500).json({ error: "Failed to submit game for approval" });
  }
});
app.get("/suggested-games", async (req, res) => {
  try {
    const games = await Game.find().sort({ addedAt: -1 });
    res.json(games);
  } catch (error) {
    res.status(500).json({ error: "Failed to fetch games" });
```

```
  }
});
async function fetchGameDetailsFromBackend(slug) {
  try {
    const response = await axios.get(`https://api.rawg.io/api/games/${slug}`, {
      params: { key: process.env.RAWG_API_KEY }
    });
    return response.data;
  } catch (err) {
    return err;
  }
}
app.post("/addyours", verifyToken, verifyAdmin, async (req, res) => {
  const { gamename } = req.body;
  if (!gamename) return res.status(400).json({ error: "Game name is required" });
  try {
    const gameData = await fetchGameDetailsFromBackend(gamename);
    if (!gameData) return res.status(500).json({ error: "Failed to fetch game" });
    let existingGame = await Game.findOne({ id: gameData.id });
    if (existingGame) {
      if (!existingGame.addedBy.includes(req.user.username)) {
        existingGame.addedBy.push(req.user.username);
        await existingGame.save();
      }
      return res.status(200).json({ message: "Game exists. Username added.", data: existingGame });
    }
const pendingGame = new Game({
      id: pendingGame.id,
      slug: pendingGame.slug,
      name: pendingGame.name,
      description: pendingGame.description,
```

```
      background_image: pendingGame.background_image,
      genres: pendingGame.genres,
      platforms: pendingGame.platforms,
      rating: pendingGame.rating,
      released: pendingGame.released,
      addedBy: pendingGame.submittedBy,
      addedAt: new Date(),
      website: pendingGame.website,
    });
    const newGame = new Game({
      id: gameData.id,
      slug: gameData.slug,
      name: gameData.name,
      description: gameData.description_raw || gameData.description,
      background_image: gameData.background_image,
      genres: gameData.genres,
      platforms: gameData.platforms,
      rating: gameData.rating,
      released: gameData.released,
      addedBy: [req.user.username],
      addedAt: new Date(),
      website: gameData.website,
    });
    await newGame.save();
    res.status(201).json({ message: "Game added successfully!", data: newGame });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Internal server error" });  }
});
app.post("/fetch-game-details", async (req, res) => {
  const { slug } = req.body;
```

```
  if (!slug) return res.status(400).json({ error: "Missing slug" });
  try {
    const response = await axios.get(`https://api.rawg.io/api/games/${slug}`, {
      params: { key: process.env.RAWG_API_KEY },
      validateStatus: () => true, // prevent axios from throwing on 404
    });
    if (response.status === 404) {
      return res.status(404).json({ error: "Game not found" });
    }
    res.json(response.data);
  } catch (err) {
    console.error("Fetch-game-details error:", err.message);
    res.status(500).json({ error: "Failed to fetch game details" });
  }
});
app.put("/admin/edit-game/:id", verifyToken, verifyAdmin, async (req, res) => {
  const gameId = req.params.id;
  const { updatedGame } = req.body;
  if (!updatedGame) {
    return res.status(400).json({ error: "Updated game data is required" });
  }
  try {
    // First check in PendingGames
    let game = await PendingGame.findById(gameId);
    if (!game) {
      // If not pending, check in approved Games collection
      game = await Game.findById(gameId);
    }
    if (!game) {
      return res.status(404).json({ error: "Game not found" });
    }
```

```
    const allowedFields = [
      "name", "slug", "description", "background_image",
      "genres", "platforms", "rating", "released", "website"
    ];
    allowedFields.forEach((field) => {
      if (updatedGame[field] !== undefined) {
        game[field] = updatedGame[field];
      }
    });

    await game.save();
    res.status(200).json({ message: "Game updated successfully", data: game });
  } catch (err) {
    console.error("Edit game error:", err);
    res.status(500).json({ error: "Failed to update game" });  }
});
app.get("/game-details/:slug", async (req, res) => {
  const { slug } = req.params;
  if (!slug) return res.status(400).json({ error: "Slug is required" });
  try {
    // Try finding in the main Game collection first
    const game = await Game.findOne({ slug });
    if (!game) {
      // If not found, try pending games as a fallback
      const pendingGame = await PendingGame.findOne({ slug });
      if (!pendingGame) {
        return res.status(404).json({ error: "Game not found in database" });
      }
      return res.status(200).json(pendingGame);
    }
    res.status(200).json(game);
```

```
  } catch (err) {
    console.error("Fetch game from DB error:", err);
    res.status(500).json({ error: "Failed to fetch game from database" });
  }
});
app.post("/add-review", verifyToken, async (req, res) => {
  const { gameId, reviewText, rating } = req.body;
  if (!gameId || !reviewText || !rating)
    return res.status(400).json({ error: "Game ID, review text, and rating are required" });
  try {
    const review = new Review({
      gameId,
      username: req.user.username,
      reviewText,
      rating,
    });
    await review.save();
    res.status(201).json({ message: "Review submitted successfully" });
  } catch (err) {
    res.status(500).json({ error: "Failed to submit review" });
  }
});
app.get("/reviews/:gameId", async (req, res) => {
  const gameId = parseInt(req.params.gameId);
  if (!gameId) return res.status(400).json({ error: "Invalid game ID" });
  try {
    const reviews = await Review.find({ gameId }).sort({ createdAt: -1 });
    res.status(200).json(reviews);
  } catch (err) {
    res.status(500).json({ error: "Failed to fetch reviews" });
  }
```

```
});
app.get("/reviews-count/:gameId", async (req, res) => {
  const gameId = parseInt(req.params.gameId);
  if (!gameId) return res.status(400).json({ error: "Invalid game ID" });
  try {
    const count = await Review.countDocuments({ gameId });
    res.status(200).json({ count });
  } catch (err) {
    res.status(500).json({ error: "Failed to fetch reviews count" });
  }
});
app.get("/yoursuggested", verifyToken, async (req, res) => {
  try {
    const games = await Game.find({ addedBy: req.user.username }).sort({ addedAt: -1 });
    res.status(200).json(games);
  } catch (err) {
    res.status(500).json({ error: "Failed to fetch your suggested games" });
    console.error(err);
  }
});
app.post("/make-admin", verifyToken, async (req, res) => {
  // Only allow in development environment
  if (process.env.NODE_ENV === 'production') {
    return res.status(403).json({ error: "Not available in production" });
  }
  try {
    const user = await User.findById(req.user.id);
    if (!user) {
      return res.status(404).json({ error: "User not found" });
    }
    user.isAdmin = true;
```

```
    await user.save();
    res.status(200).json({ message: "User is now an admin (DEVELOPMENT ONLY)" });
  } catch (err) {
    res.status(500).json({ error: "Failed to make user admin" });
  } });
app.get("/admin/pending-games", verifyToken, verifyAdmin, async (req, res) => {
  try {
    const pendingGames = await PendingGame.find({ status: 'pending' }).sort({ submittedAt: -1 });
    res.status(200).json(pendingGames);
  } catch (err) {
    res.status(500).json({ error: "Failed to fetch pending games" }); }
});
app.post("/admin/approve-game/:id", verifyToken, verifyAdmin, async (req, res) => {
  try {
    const pendingGame = await PendingGame.findById(req.params.id);
    if (!pendingGame) {
      return res.status(404).json({ error: "Pending game not found" });
    }
    const existingGame = await Game.findOne({ id: pendingGame.id });
    if (existingGame) {
      pendingGame.status = 'rejected';
      pendingGame.adminNotes = 'Game already exists in database';
      await pendingGame.save();
      return res.status(400).json({ error: "Game already exists in database" });
    }


    await newGame.save();
    pendingGame.status = 'approved';
    await pendingGame.save();
    res.status(200).json({ message: "Game approved successfully" });
  } catch (err) {
```

```
    res.status(500).json({ error: "Failed to approve game" });
    console.error(err); }  });
app.post("/admin/reject-game/:id", verifyToken, verifyAdmin, async (req, res) => {
  const { adminNotes } = req.body;
  try {
    const pendingGame = await PendingGame.findById(req.params.id);
    if (!pendingGame) {
      return res.status(404).json({ error: "Pending game not found" });
    }
    pendingGame.status = 'rejected';
    pendingGame.adminNotes = adminNotes || 'Rejected by admin';
    await pendingGame.save();
    res.status(200).json({ message: "Game rejected successfully" });
  } catch (err) {
    res.status(500).json({ error: "Failed to reject game" });
    console.error(err);
  }
});
app.listen(port, "localhost", () => {
  console.log(`🚀 Server running on port ${port}`);
});
```

# System Implementation

## i) Hardware Requirements (End User)

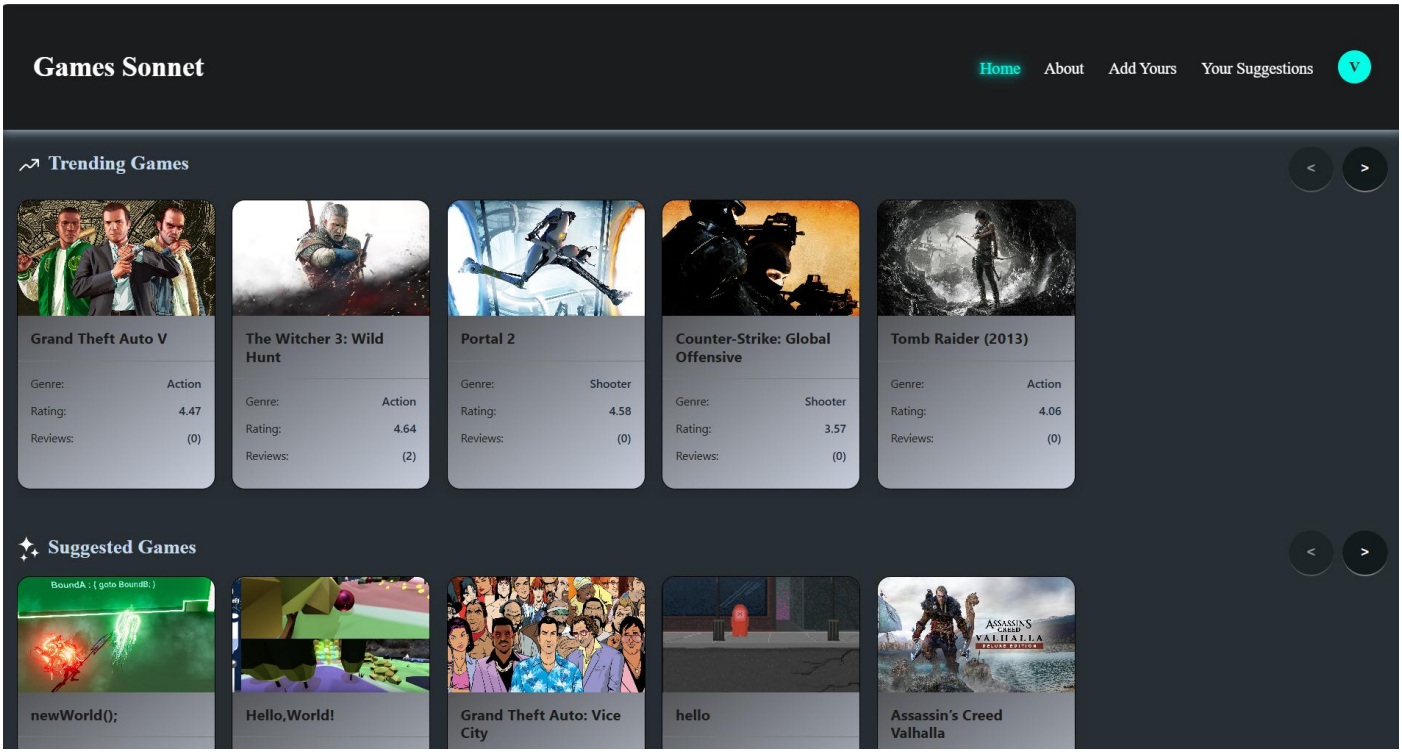
- Device: Any smartphone, tablet, laptop, or desktop with internet access
- Processor: Dual-core or higher (Intel, AMD, ARM—all supported)
- RAM: Minimum 2 GB (4 GB recommended for smoother browsing)
- Storage: No installation required; only browser cache used
- Display: 720p resolution or higher (1366×768 recommended for optimal layout)
- Internet: Stable connection (minimum 1 Mbps) for loading game images and making API calls to RAWG

**Software Requirements (End User)**

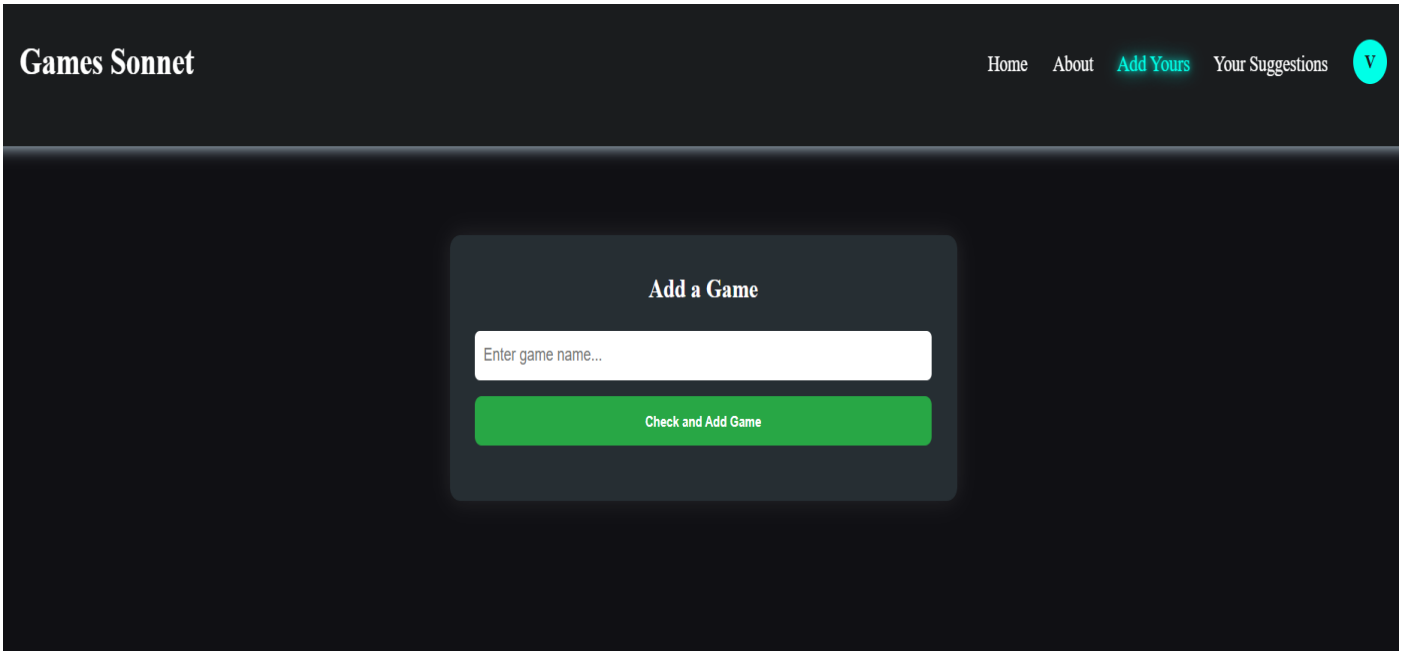
- **Web Browser**:
    - Google Chrome (latest version)
    - Mozilla Firefox
    - Microsoft Edge
    - Safari (for iOS/macOS users)
- **Operating System**:
    - Windows 7 or later
    - macOS
    - Android 8.0+
    - iOS 12+
    - Linux distributions with modern browser support

## (ii) Screen / Report Layouts

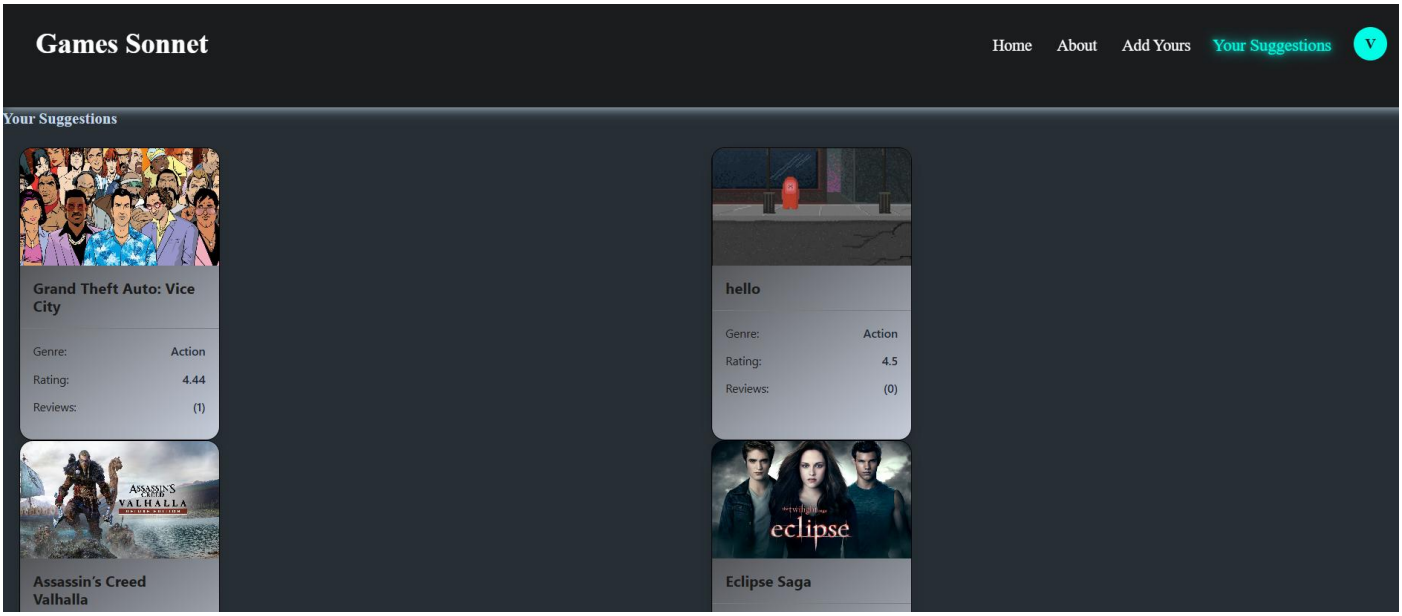
### 1. Home Page



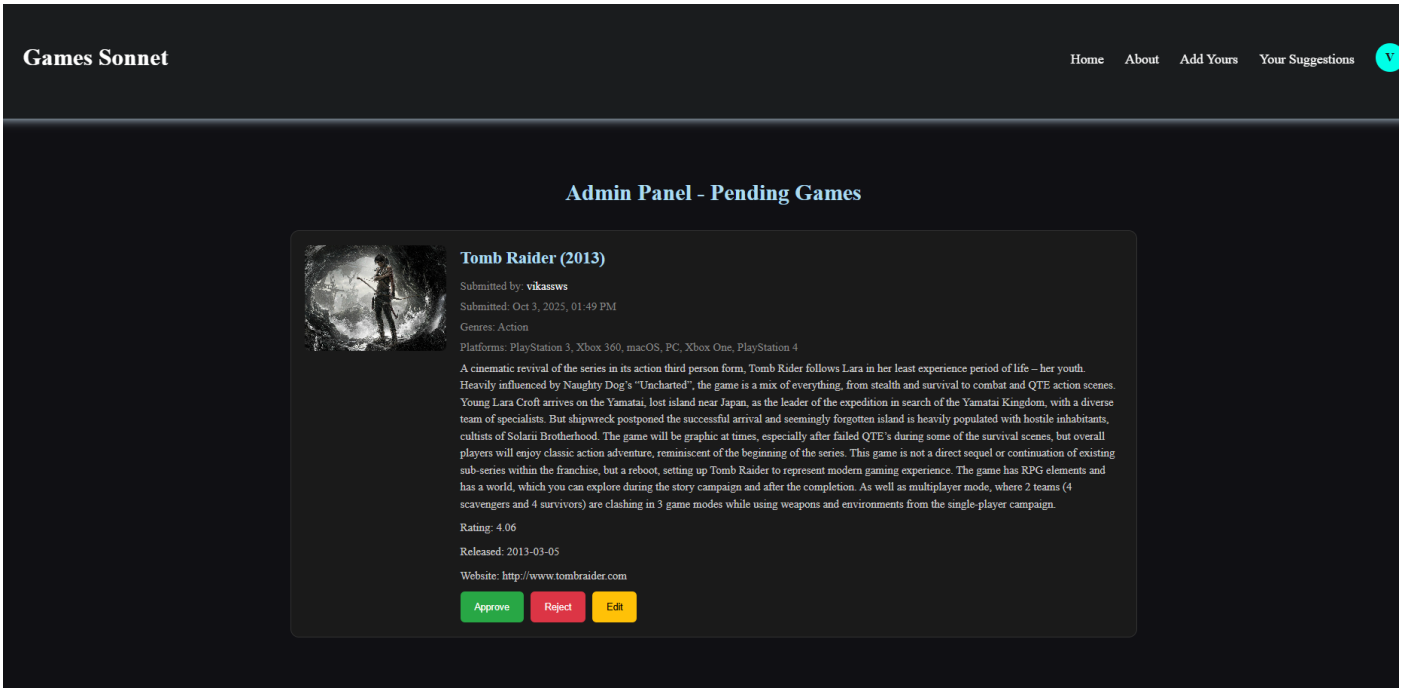

### 2. Add Yours Page



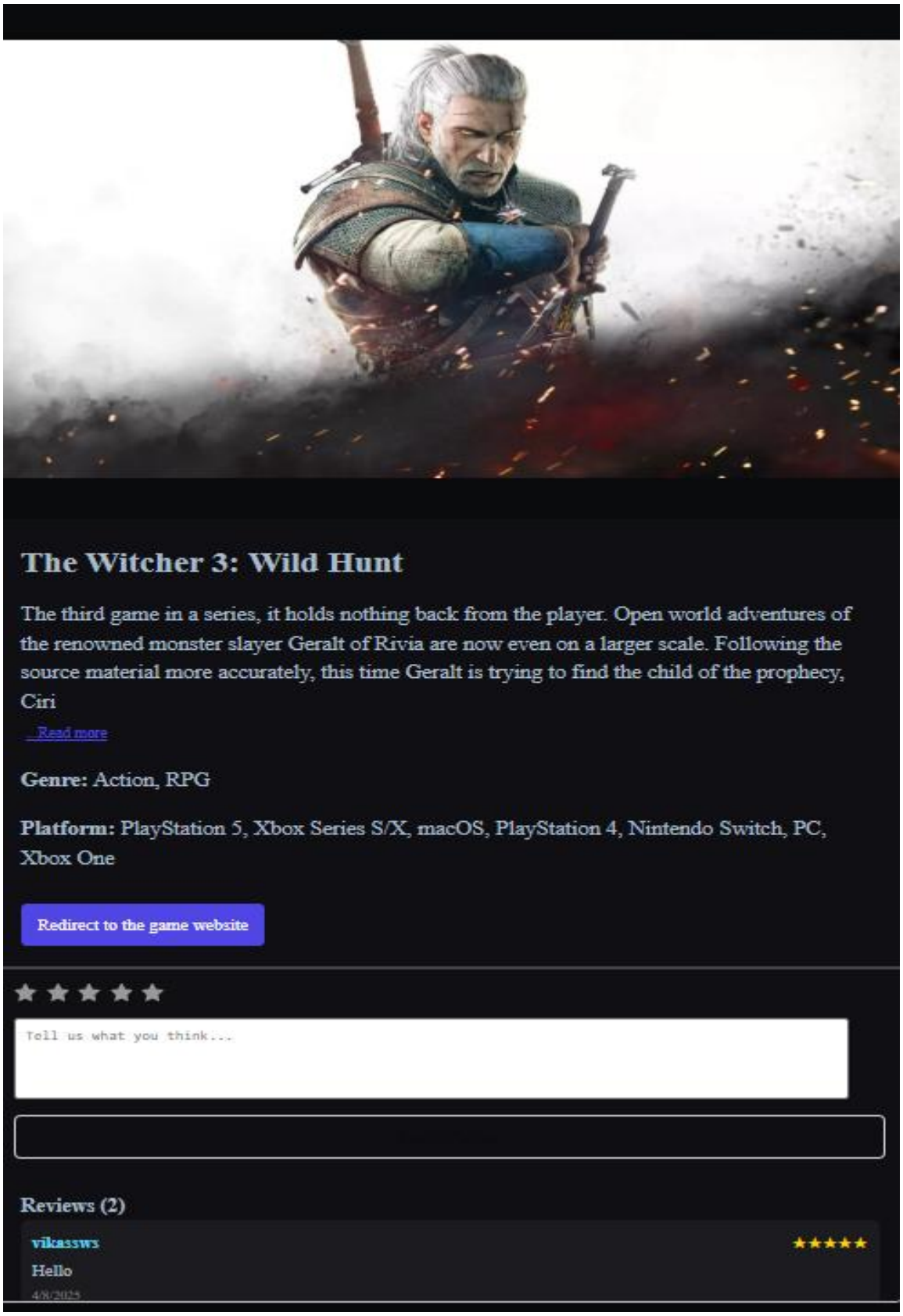
## 3. Your Suggestions Page



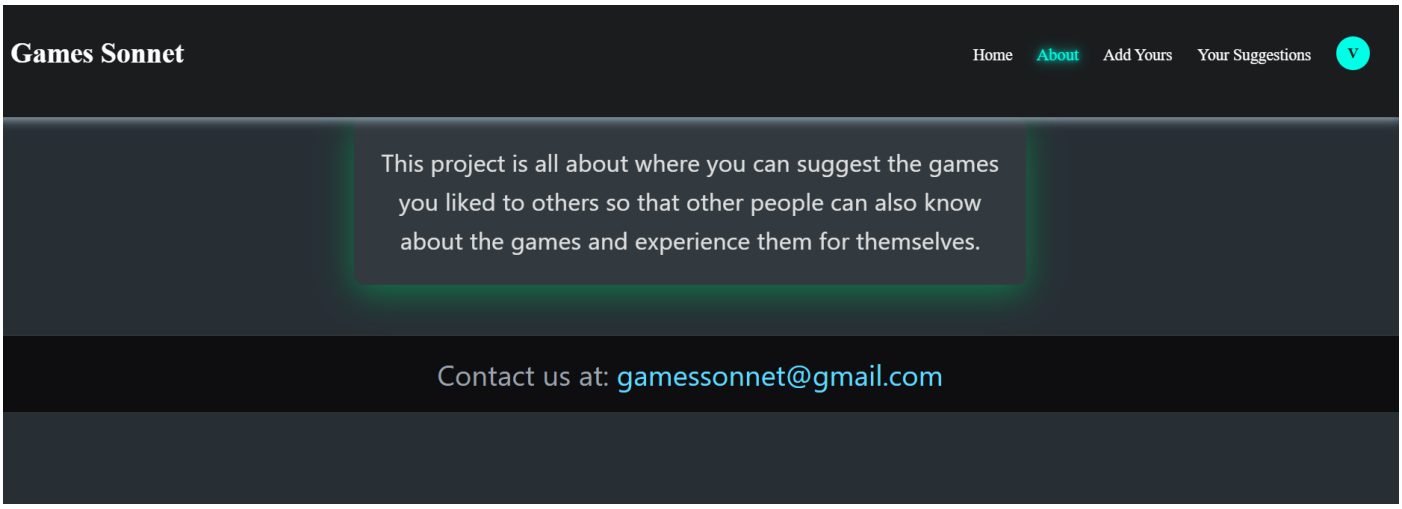

## 4. Admin Panel



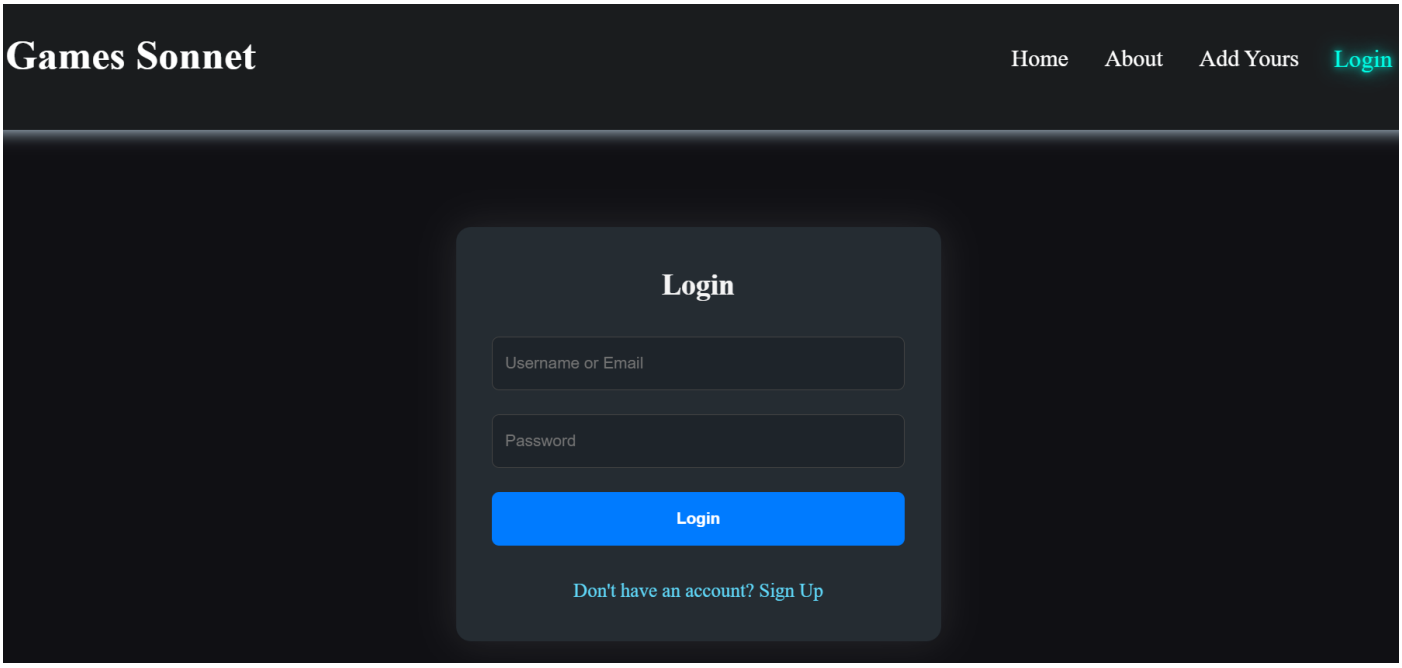
## 6. Game Details Page (Big Card)

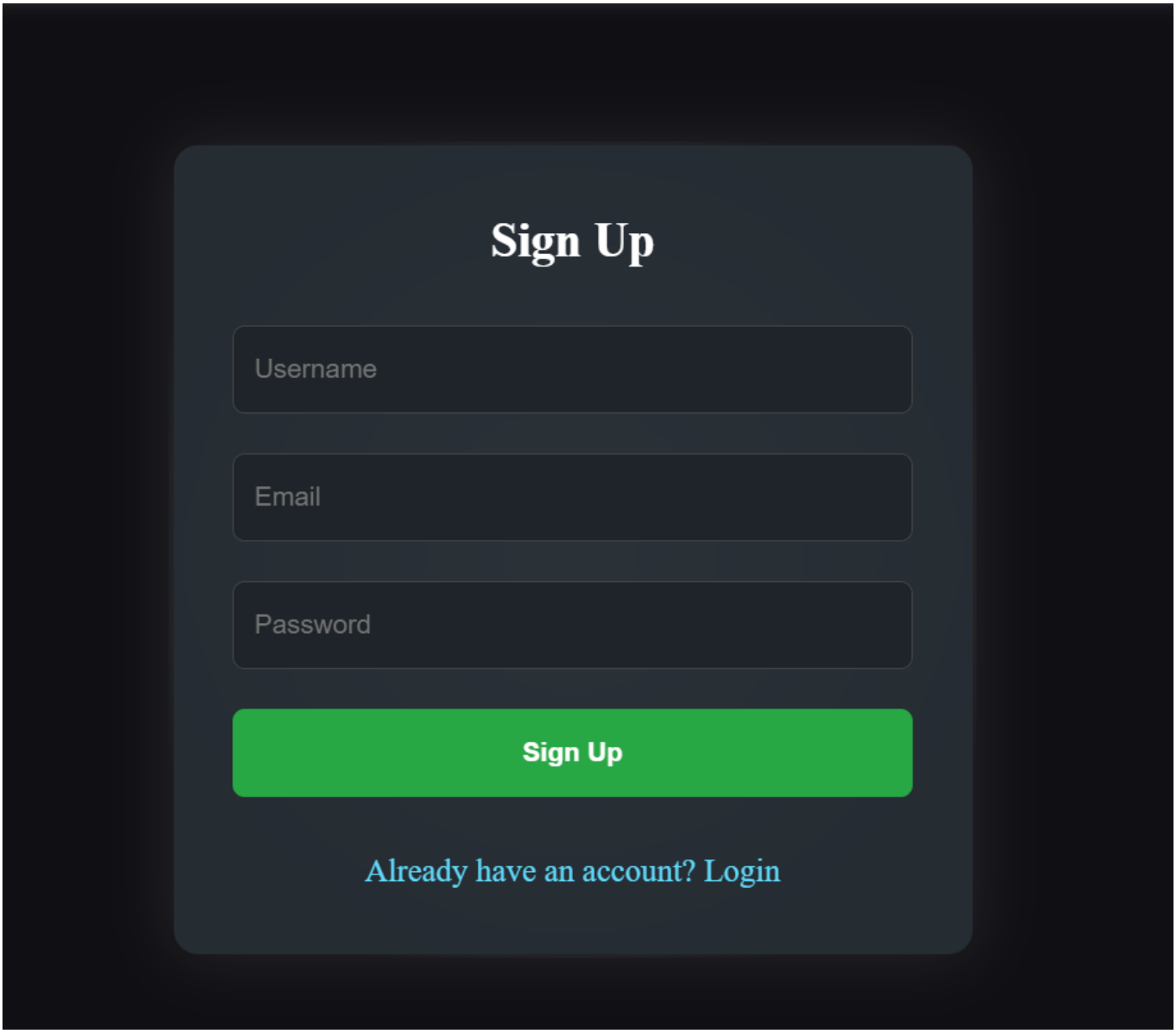


## 6. About Page




## 7. Login/Signup Pages




Sign Up
Username
Email
Password
Sign Up
Already have an account? Login

# Future Enhancements

1. Reporting System – Users can directly report inappropriate reviews or fake game entries instead of emailing admin.
2. Automated Moderation – AI-based content moderation for reviews to detect spam or offensive content.
3. Mobile friendly :- More optimizations for mobile based user’s to enhance their expeirence
4. Recommendation Engine – Personalized game recommendations using Machine Learning (based on user reviews & preferences).
5. Social Features –Likes/upvotes on reviews.
6. Analytics Dashboard for Admin – Insights into most reviewed, trending, or reported games.
7. Email / Push Notifications – Notify users when their suggested game is approved or rejected.
8. Game Wishlist / Favorites – Users can save games to their personal wishlist.
9. Review or Game Removal:- Admin can directly delete a review or game instead of manually finding it in mongodb and deleting it

# References and Bibliography

## Books & Articles


- TutorialsPoint – *UML Diagrams* documentation.
- Geeks for Geeks- UML Daigrams flow/structure


## Websites


- RAWG API Documentation
- MongoDB Documentation
- React Documentation
- Express.js Documentation
- Node.js Documentation


## Videos:


**Codewithharry:-** Sigma web development

**Chai aur code :-** Backend series

# Annexures

**System Proposal**

**Title:** Community-Driven Game Recommendation Web Application

**1. Introduction**

The proposed system is a web-based platform that enables users to discover, suggest, review, and rate video games. It combines community-driven content with external data (via RAWG API) to provide personalized and trending game recommendations. The platform emphasizes ease of use, real-time updates, and secure authentication.

**2. Objectives**

- Develop a full-stack web application with React, Node.js, and MongoDB.
- Implement secure user authentication (bcrypt hashing).
- Allow users (and guests) to suggest games, post reviews, and provide ratings.
- Enable real-time content updates without page reloads.
- Deliver proper documentation and diagrams for academic and industry standards.

**3. Scope**

- **Core Features:** Game suggestions, reviews, star ratings, top weekly games, anonymous submissions.
- **Tech Stack:** React.js (frontend), Node.js + Express (backend), MongoDB Atlas (database).
- **Hosting:** Netlify (frontend), Render (backend).
- **Security:** Encrypted passwords, input validation, error handling.

**4. Methodology**

- Requirement Analysis
- Design (UML diagrams, ERD)
- Implementation (frontend & backend)
- Testing (unit, integration, and user testing)
- Deployment & Documentation

**5. Expected Outcomes**

- A functional game recommendation website.
- Secure login and anonymous contributions.

- Real-time updates for new games and reviews.
- Error handling with custom fallback pages.
- Clear project documentation with UML diagrams.

**6. Timeline (10 Weeks)**

- **Weeks 1–2:** Requirement gathering & design
- **Weeks 3–6:** Implementation (frontend + backend)
- **Weeks 7–8:** Integration & testing
- **Weeks 9–10:** Deployment & documentation

**7. Resources**

- Tools: VS Code, GitHub, MongoDB Atlas, RAWG API
- Platforms: Netlify, Render
- Version Control: GitHub

**8. Team Size**

1 Member

**9. Guide's Role**

The guide will provide technical and strategic advice, ensuring the project aligns with both academic and professional standards.

R. D. & S.H. National College and S.W.A. Science College

T. Y. B.Sc. (Sem V) Computer Science Project

PHASE COMPLETION TABLE (2025 – 2026)

Name of Student: Vikas Naresh Makwana Roll No: CS 23024

Name of Project: GamesSonnet

| Phase | Proposed Date Of Completion | Actual Date Of Completion | Signature of Project Guide | Remarks |
|---|---|---|---|---|
| **PHASE – I: PRELIMINARY INVESTIGATION** | | | | |
| Submission Of Synopsis | 12th July | | | |
| Study Of Current System | 2nd Aug | | | |
| Derivation Of Proposed System | 2nd Aug | | | |
| Feasibility Study | 2nd Aug | | | |
| **PHASE – II: SYSTEM ANALYSIS** | | | | |
| Event Table | 9th Aug | | | |
| Use Case Diagram | 9th Aug | | | |
| ERD | 9th Aug | | | |
| Activity Diagram | 16th Aug | | | |
| Class Diagram | 23rd Aug | | | |
| Object Diagram | 23rd Aug | | | |
| Sequence Diagram | 6th Sept | | | |
| Collaboration Diagram | 6th Sept | | | |
| State Diagram | 6th Sept | | | |
| **PHASE – III: SYSTEM DESIGN** | | | | |
| Component Diagram | 13th Sept | | | |
| Package Diagram | 13th Sept | | | |
| Deployment Diagram | 20th Sept | | | |
| System flow chart | 20th Sept | | | |
| Structured Chart | 20th Sept | | | |
| **PHASE – IV: SYSTEM CODING** | | | | |
| Menu Tree/Sitemap | 9th Aug | | | |
| Data Dictionary | 9th Aug | | | |
| Design Patterns used | 9th Aug | | | |
| Source Code | 13th Sept | | | |
| **PHASE – V: SYSTEM IMPLEMENTATION** | | | | |
| Final Project Submission | 20th Sept | | | |
| Documentation Soft Copy Submission | 27th Sept | | | |

| Phase | Proposed Date Of Completion | Actual Date Of Completion | Signature of Project Guide | Remarks |
|---|---|---|---|---|
| Documentation Hard Copy Submission | 30th Sept | | | |

## GANTT CHART (2025 – 2026)