1. Study of Current System

Existing online platforms like **RAWG API, Steam, Epic Games, and IMDb (for movies)** provide structured databases of games or media.

However, these systems have limitations:

1. **No User Contribution** – Users cannot directly contribute new games; all data comes from APIs or official catalogs.

2. **Lack of Moderation** – No admin-based approval process exists, which can lead to data inaccuracy if external submissions were allowed.

3. **No Manual Submission Support** – If a game is missing in the API or catalog, users have no way to add details manually.

4. **No Duplicate Management** – Existing systems do not check if a game has already been suggested by other users, leading to redundancy issues if such a feature were enabled.

5. **Limited Community Collaboration** – No space for users to collectively add or update contributions (e.g., multiple contributors for a single game).

6. **Risk of Outdated or Fake Data** – Since no validation or approval layer is present, reliability may drop if community-driven contributions were allowed without checks.

2. Derivation of Proposed System

The **proposed system (GameSonnet)** improves the current limitations by adding new features:

1) Backend Duplicate Check

- Before submission, system checks MongoDB:
    - **`Games`** collection → If game exists, add contributor to **`addedBy`** field.

- **`PendingGames`** collection → If game is already pending, do not duplicate entry.
- If not found, add new entry into **`PendingGames`** for admin approval

2) **Admin Panel for Game Approval**
   - Games suggested by users are not directly added.
   - Admin only sees **unique, unapproved submissions**.
   - Admin can approve, reject, or edit the new submission.
   -
   - Admins review, approve, reject, or edit details before adding them to the **Games Database**.
   - Ensures accuracy, authenticity, and prevents fake entries.

3) **Manual Submission of Games**
   - If a game is not found in RAWG API, users can **manually enter details**.
   - The manual submission is also reviewed by admins before approval.

4) **User Contributions (AddedBy Field)**
   - Multiple users can contribute the same game.
   - If a game already exists in the database, the new contributor's username is added to the **addedBy field**.
   - This builds a **community-driven contribution model** and shows collective effort.

    - If duplicate → user gets feedback (`"Already exists"` or `"Pending approval"`).
    - If new → toast message **`"Game submitted for admin approval"`.**

5) **Secure Login & User Reviews**
    - Users can log in, suggest games, and leave reviews with ratings.
    - Unregistered users can still view game details and read reviews but cannot post reviews.

6) **Home Page Enhancements**
    - Displays **Trending Games** (fetched from RAWG API).
    - Displays **Suggested Games** (approved by admins from the database).
    - Clicking a card fetches data either from RAWG or from the database.

3. Feasibility Study

The feasibility of the system is evaluated on three major aspects:

a) Operational Feasibility

- The system provides a **user-friendly interface** with:
    - Home Page with trending & suggested games.
    - Add Yours section for suggesting games.
    - Admin Panel for approvals.
    - Game details page with reviews.

- Easy navigation through **Navbar** and clear segregation of **User vs Admin functionalities**.
- Duplicate check improves data quality.
- Prevents admin overload by filtering redundant requests.
- Supports both **automatic API fetching** and **manual submissions**, making it reliable and adaptable.

b) Technical Feasibility

- **Built with MERN Stack (MongoDB, Express, React, Node.js).**
- **Uses JWT authentication** for secure login & admin verification.
- **MongoDB handles multiple schemas:** Users, Games, PendingGames, Reviews.
- **MongoDB duplicate check** is efficient using `findOne()` queries.
- **Adds minimal overhead,** since checks happen before insert.
- **Integration with RAWG API** for trending and real-time game data.
- **Scalable and deployable** on platforms like Render (backend) and Netlify (frontend).

c) Economic Feasibility

- The system is cost-effective:
    - Uses **open-source technologies** (React, Node, Express, MongoDB).
    - API usage of RAWG is free up to certain limits.
    - Minimal hosting costs (Netlify + Render free tiers).
- Community contribution model reduces admin overhead since data is collaboratively enriched.