Software Requirements Specification

(SRS) Document

PartyPulse

kcxul/PartyPulse: Group-9 Project

12/12/2024

Version 2

Caden Parsons, Y'Vin Kpa, Daisy Ibuoka

1. Product General Description

The goal of the PartyPulse web application is to provide a platform which provides users with the nuisance of finding other gamers to play with.

2. Product Features

The PartyPulse app is designed to make finding gaming partners easier. Below are the key functions and features that define the app:

Game and style preference: Users can select different games that they want to find partners for, which will automatically take them to the respective PartySpace. Users can also select their gaming style like competitive, causal, speedrunner, etc.

Personalized profile creation: All Users can create and design their profiles, and these profiles can be edited with “roles” inside the PartySpaces by Providers based on what style of play, time in the PartySpace, how active the user is inside the PartySpace, etc.

3. Content Creation and Sharing

The PartyPulse application provides a platform for content creators to build their own groups and share their creations and/or videos towards other users who are playing the same game. This will also include integration with other social apps like discord, youtube and twitch.

3. Functional requirements:
- FR0. The app will require all users to make a profile
- FR1: The app will require all users to modify their profile
- FR2: the app will allow the user to filter what PartySpace(s) they join based off of what games they play
-FR3: The app will allow users to talk in a server, or directly to each other.
- FR4: Accounts will be protected with a login page and password
- FR5: The app will provide notifications for users when they receive messages or requests from other users.
-FR6: The app will allow a user to become a provider by creating a “Party Space” for the game they want.
-FR7: The app will allow a provider to ban users from their PartySpace and moderate content
-FR8: The app will allow Admins to ban users, who have been banned from PartySpaces, from using the app if the offense is bad enough.
-FR9: Admins will be assigned.

4. Non-functional requirements:

Customer Non-functional Requirements

1. Usability:
    - The system should allow users to complete key tasks (e.g., subscribing to services) within 3 clicks.
2. Performance:
    - The system should load the service listing page within 2 seconds on a standard broadband connection.
3. Security:
    - User data must be protected with encryption to ensure privacy.

Provider Non-functional Requirements

1. Usability:
    - Providers should be able to create or modify their profiles in under 5 minutes.
2. Performance:
    - The system should process service creation requests within 1 second.
3. Reliability:
    - The system should maintain 99.9% uptime to ensure providers can access their accounts at any time.

SysAdmin Non-functional Requirements

1. Usability:
    - Admin tools should allow user management tasks to be completed within 5 minutes.
2. Performance:
    - The system should allow for moderation of reviews with a response time under 2 seconds.
3. Security:
    - Admin access should be protected by two-factor authentication.

5. Use cases:
    a. Users: Caden Parsons
        I. Creating their profile:
            - Initial assumption: The user has access to the web app and is prompted to create a profile.
            - Normal: The user enters their wanted username and password and the profile is created.
            - What can go wrong: The username is already taken. The site will tell them that the username is already taken.
            - Other activities: There is an “edit” button after they click on their profile picture to go to their profile so users can edit the profile like their picture, username, etc
            - System state on completion: The users profile will be updated
        ii. View games/PartySpaces:
            - Initial assumption: The user has access to the web app and clicks on “Find PartySpaces”.
            - Normal: The user types in the Game for the PartySpace they want to join.
            - What can go wrong: The user can’t find any PartySpaces for the game that they want to join. The web app will show them that there is no PartySpace for the selected game
            - System state on completion: The user be shown either the PartySpace for the game, or shown “there is no PartySpace for this title”.
        iii. Join available PartySpaces/Create a PartySpace:
            - Initial assumption: The user has access to the web app, finds a PartySpace that they want to join and clicks on “Join PartySpace”.
            - Normal: The user clicks on, and joins, the PartySpace that they want to join.
            - What can go wrong: The user either joins the wrong one or can’t find the PartySpace for the game they want. The user will have a pop-up that says “Are you sure you want to join?” And in the case of a PartySpace not being available, they can create the PartySpace and become a provider.

- System state on completion: The user will either be in their selected PartySpace and/or become a Provider for the PartySpace that they created.

iv. Report:

- Initial assumption: The user wants to report a bug that they found.
- Normal: They click on the “Report a bug” button and report the bug
- What can go wrong: The user doesn’t know where they found the bug. This report screen will be in a separate window so they can still see where they are in the web app so they can describe the issue.
- System State on Completion: The user submits the bug report and the window states “You may now close this window”.

Providers - Y'Vin Kpa

I. **Create Party Spaces**

- **Initial Assumption**: The provider has access to the web application, is logged in, and is on the create party space pop-up window.
- **Normal**: The provider has to enter a name for the party. In addition, the user is given the option to upload an icon and write a description for their party, but it is not required. Lastly, there will be a visibility option where you can set it to public or private.
- **What Can Go Wrong**: An error will show up under the name box stating that the party name is longer than 32 characters. When a user uploads an icon for their party, they will be presented with an error that says that the image size is too small. The minimum size is 512x512 pixels. In addition, there is a file size limit of 10 MB. When writing the description, the provider may get an error that the description is longer than 2048 characters.
- **Other Activities**:
- **System State on Completion**: A notification will appear stating that the party space has been created. The notification will also contain the party space name and its unique id.
    - Create Party Space: Your Party Space has been created. ${name} is provided with its ${uniqueID}.

II. **Manage Party Members**

- **Initial Assumption**: The provider has access to the web application, is logged in, and is in the on the party space's settings page.
- **Normal**: Under the member tabs, the provider is presented with a list of current members in the party. The provider can click on the member and they will be presented with a pop-up window with the options to manage roles, ban, or kick. Roles include admin, moderator, and member. On the other hand there is a roles tab where you can update permissions for each role. Permissions include send messages, mute users, kick users, and ban users.
- **What Can Go Wrong**: Providers may accidentally kick or ban a user. To prevent this, the provider or a role with permission will be prompted with a confirmation.
- **Other Activities**: The provider can sort the list of users by date joined, roles, or username.
- **System State on Completion**: A notification pop-up will be presented when certain actions are completed.
    - Roles: User role has been updated.

    - Permissions: Permissions have been updated.
    - Kick: The user has been removed from the party.
    - Ban: The user has been banned from the party.

III. Manage Party Spaces

- **Initial Assumption**: The provider has access to the web application, is logged in, and is on the party space's settings page.
- **Normal**: The provider can access the overview tab in the party settings. When accessed, the provider is presented with the party's name, description, and icon. These can be changed by the provider.
- **What Can Go Wrong**: An error will show up under the name box stating that the party name is longer than 32 characters. When a user uploads an icon for their party, they will be presented with an error that says that the image size is too small. The minimum size is 512x512 pixels. In addition, there is a file size limit of 10 MB. When writing the description, the provider may get an error that the description is longer than 2048 characters.
- **Other Activities**:
- **System State on Completion**: A notification pop-up will be presented when certain actions are completed.
    - Name: Name has been updated.
    - Description: Description has been updated.
    - Icon: Icon has been updated.

IV. Manage Party Chat

- **Initial Assumption**: The provider has access to the web application, is logged in, and is on the party chat page.
- **Normal**: The provider can access the chat settings by pressing a gear icon within the chat header. The provider is then presented with a pop-up window where you enable slow mode, mute users, and filter words.
    - Slow Mode: Sending messages has a cooldown. The cooldown timer can be changed.
    - Mute Users: You can search or select a user from a drop down list to mute from sending messages.
    - Keywords Filter: You can blacklist words that you don't want to appear in the chat. You can empty the keywords by clicking remove filter.
- **What Can Go Wrong**: Provider may accidentally remove all keywords from the filter instead of removing a single keyword. The remove filter action will be prompted with a confirmation message.
- **Other Activities**:
- **System State on Completion**: A notification pop-up will be presented when certain actions are completed.
    - Slow Mode: Slow Mode has been enabled/disabled
    - Cooldown: Slow mode cooldown timer has been set to ${time}.
    - Description: Description has been updated.
    - Icon: Icon has been updated.

SysAdmin: Daisy Ibuoka

I. Manage User Access

- Initial Assumption: The admin logs into the system and navigates to the user management panel.
- Normal Flow: The admin reviews a list of flagged accounts for inappropriate behavior. They identify a customer who has posted unacceptable reviews and decide to ban the user. After confirming the action, the system updates the user's status to banned.
- What Can Go Wrong: If the admin tries to ban a user who is not in the flagged list, the system will show an error message indicating the user is not found.
- System State on Completion: The user's status is updated to banned, and they are removed from the active user list.

II. Moderate Services

- Initial Assumption: The admin accesses the service moderation page to review flagged profiles.
- Normal Flow: The admin selects a flagged profile and examines its content. After determining it violates the content policy, they click "Remove" and confirm the action.
- What Can Go Wrong: If the admin attempts to remove a profile that is not flagged, the system will display a message indicating that no action can be taken.
- System State on Completion: The flagged profile is successfully removed from the system, and the admin receives a notification of completion.

III. Moderate Reviews

- Initial Assumption: The admin navigates to the reviews section to monitor user feedback.

- Normal Flow: The admin identifies a recent review that violates community standards. They select the review and click “Delete.” After confirming the deletion, the review is removed from the system.
- What Can Go Wrong: If the admin tries to delete a review that has already been removed, the system will show an error message stating the review no longer exists.
- System State on Completion: The inappropriate review is successfully deleted, and the admin is notified of the successful action.

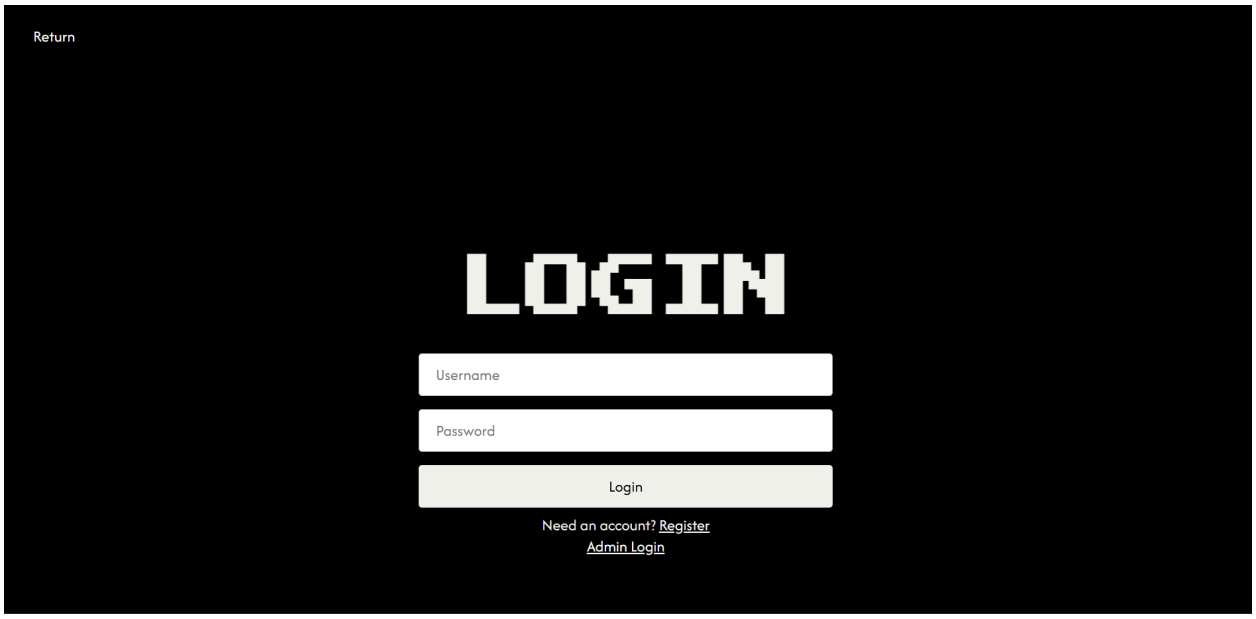
IV. View Usage Statistics

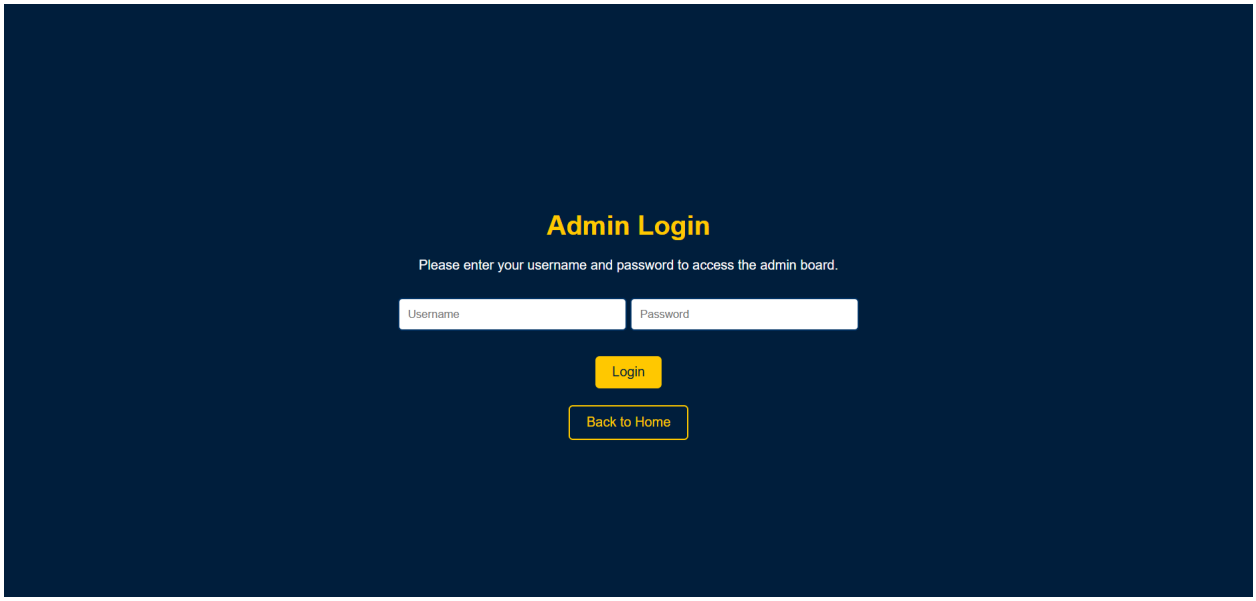
- Initial Assumption: The admin wants to analyze the application’s usage data.
- Normal Flow: The admin navigates to the usage statistics page and reviews various metrics such as active users and engagement over the last month. They decide to export the data for reporting.
- What Can Go Wrong: If the export fails due to connectivity issues, the system will display an error message and suggest retrying the export.
- System State on Completion: The admin successfully exports the data in the chosen format, and a confirmation message is displayed.

Data needed for every use case:
User: Needs list of taken usernames, needs database of partyspaces, names of partyspaces and games, needs database to store profile settings, table for joined partyspaces for each user.

Scenarios with screenshots

User login and admin login





PartyPulse Design Document

Date: 10/22/2024
Version: 2.0
Authors: Caden Parsons, Y'Vin Kpa(we filled in his info), Daisy Ibuoka

Use-Case Model

Actors:

- **User (Gamer)** - Caden Parsons
- **Provider (Party Space Creator)** - Y'Vin Kpa
- **SysAdmin (Administrator)** - Daisy Ibuoka

Use Cases:

User (Gamer)

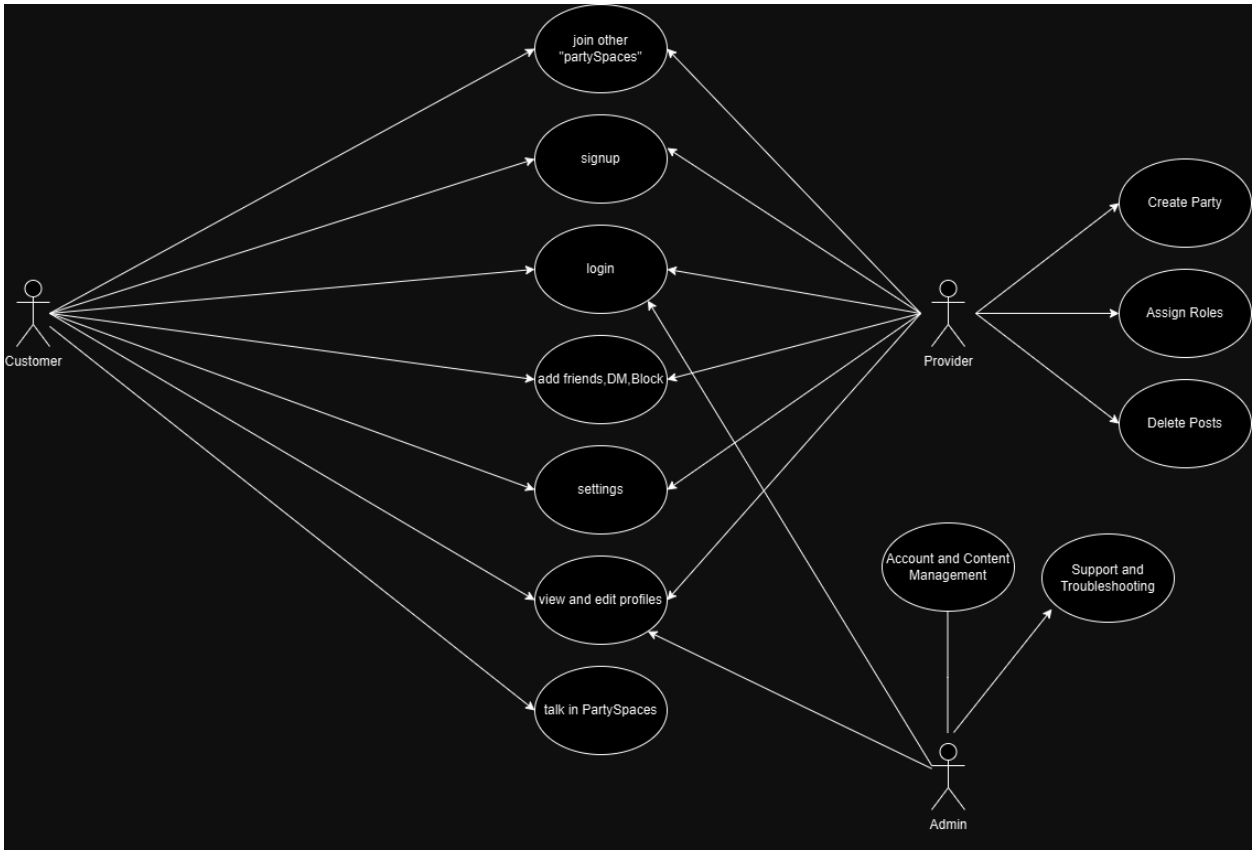
1. **Create Profile**
    - Description: User creates a profile to join the platform.
2. **Modify Profile**
    - Description: User updates their profile information.
3. **Find PartySpaces**
    - Description: User searches for available PartySpaces based on game preferences.
4. **Join PartySpace**
    - Description: User joins a selected PartySpace.
5. **Report Issue**
    - Description: User reports a bug or issue within the app.

Provider (Party Space Creator)

1. **Create Party Space**
    - Description: Provider creates a new PartySpace for a game.
2. **Manage Party Members**
    - Description: Provider manages members in their PartySpace, including banning users.
3. **Manage Party Settings**
    - Description: Provider updates settings for their PartySpace.

SysAdmin (Administrator)

1. **Manage User Access**
    - Description: Admin reviews and bans users for inappropriate behavior.
2. **Moderate Content**
    - Description: Admin removes flagged profiles and inappropriate reviews.
3. **View Analytics**
    - Description: Admin views usage statistics of the application.



2. State Machine Diagram

User (Gamer) State Diagram

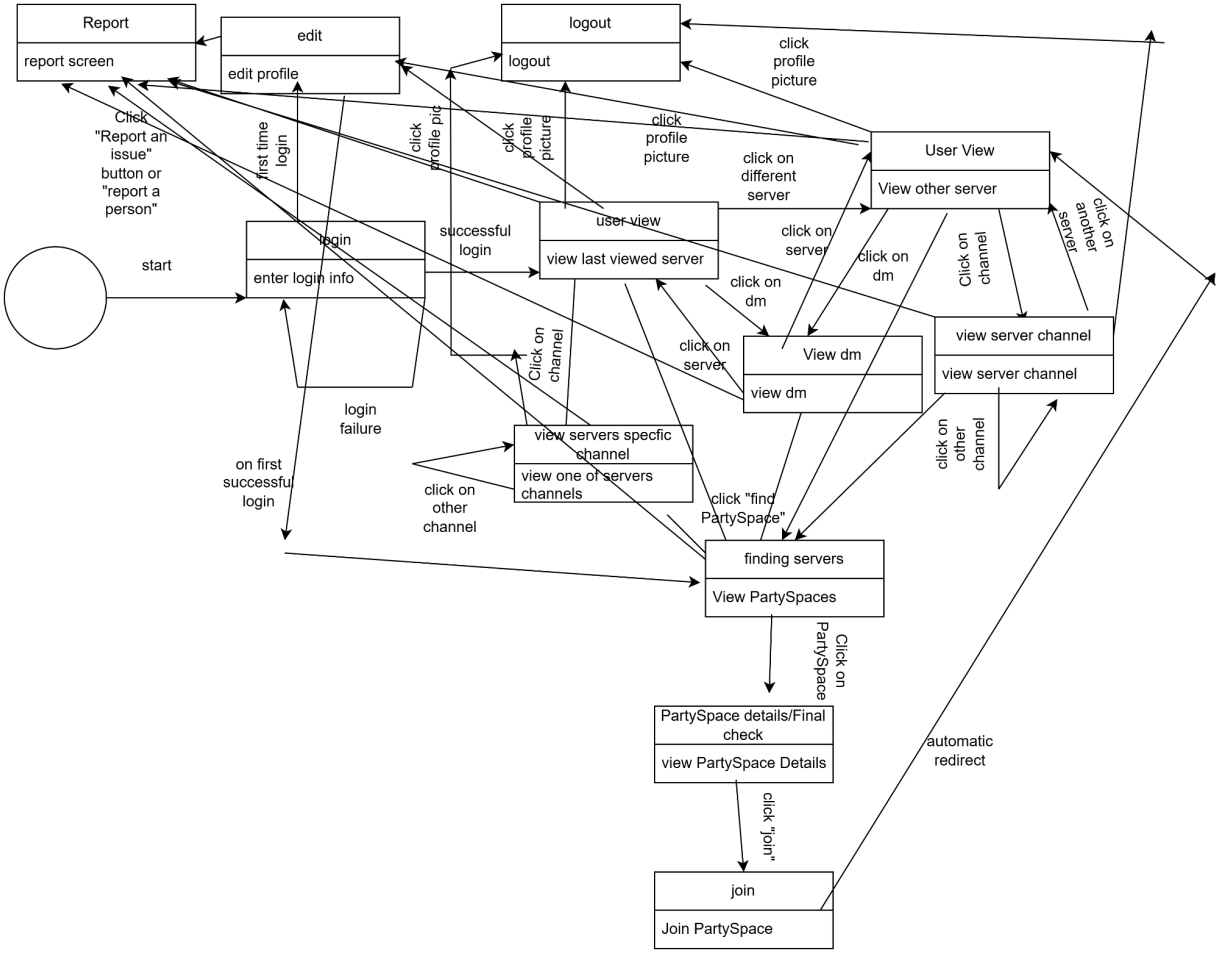
States:

1. **Login**
2. **Profile Creation**
3. **Profile Editing**
4. **PartySpace Search**
5. **Join Confirmation**
6. **Report Issue**

Transitions:

- **Login** → **Profile Creation**: User logs in and is prompted to create a profile.
- **Profile Creation** → **Profile Editing**: User completes profile creation and navigates to edit their profile.
- **Profile Editing** → **PartySpace Search**: User saves changes and searches for PartySpaces.

- **PartySpace Search → Join Confirmation**: User selects a PartySpace to join.
- **Report Issue → PartySpace Search**: User reports an issue and returns to PartySpace search.



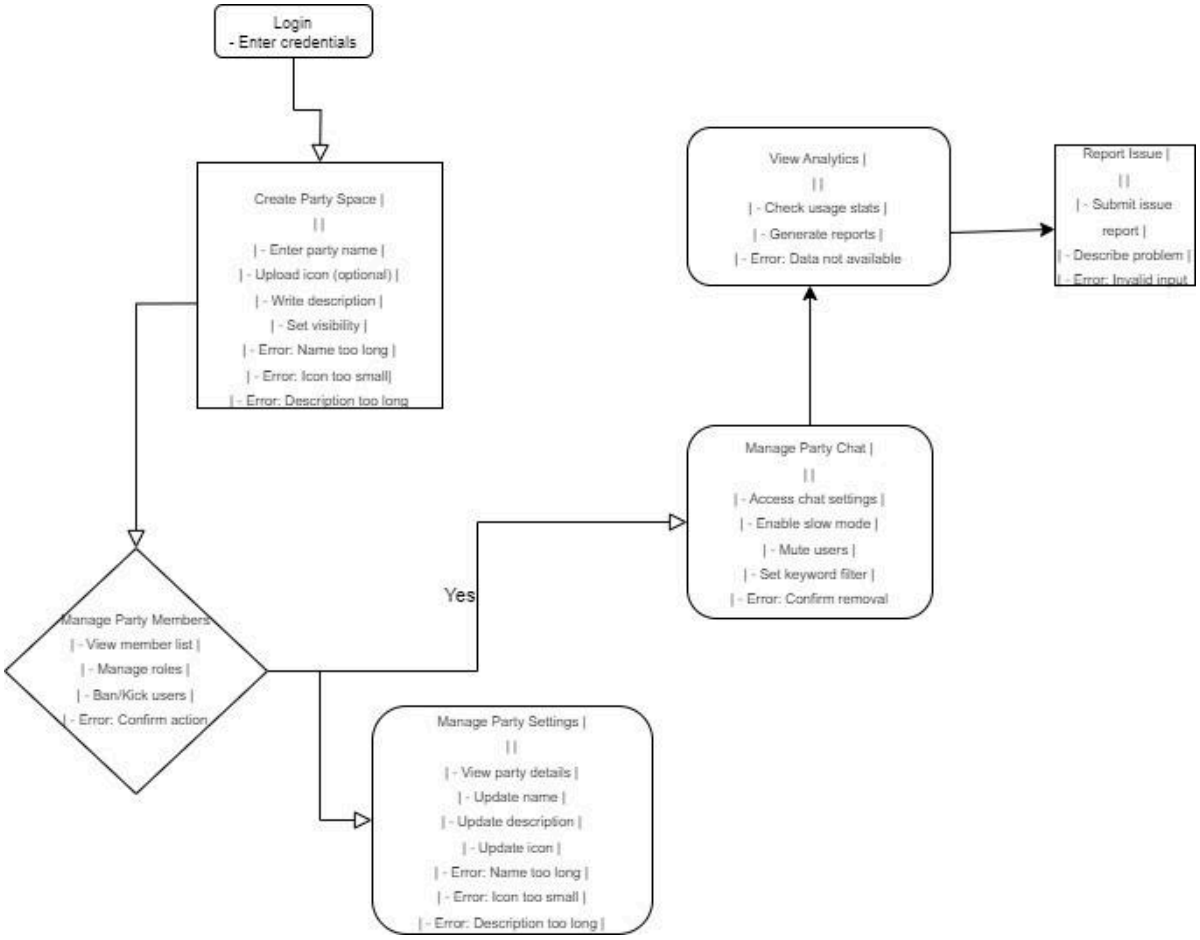
State Machine Diagram for Provider (Party Space Creator)

States:

- **Login**
- **Create Party Space**
- **Manage Party Members**
- **Manage Party Settings**
- **View Analytics**
- **Report Issue**

Transitions:

- Login → Create Party Space: User logs in and creates a new Party Space.
- Create Party Space → Manage Party Members: User creates a Party Space and then manages its members.
- Manage Party Members → Manage Party Settings: User navigates to change settings for the Party Space.
- Manage Party Settings → View Analytics: User checks analytics after updating settings.
- Report Issue → View Analytics: User reports an issue and returns to view analytics.



State Machine Diagram for SysAdmin (Administrator)

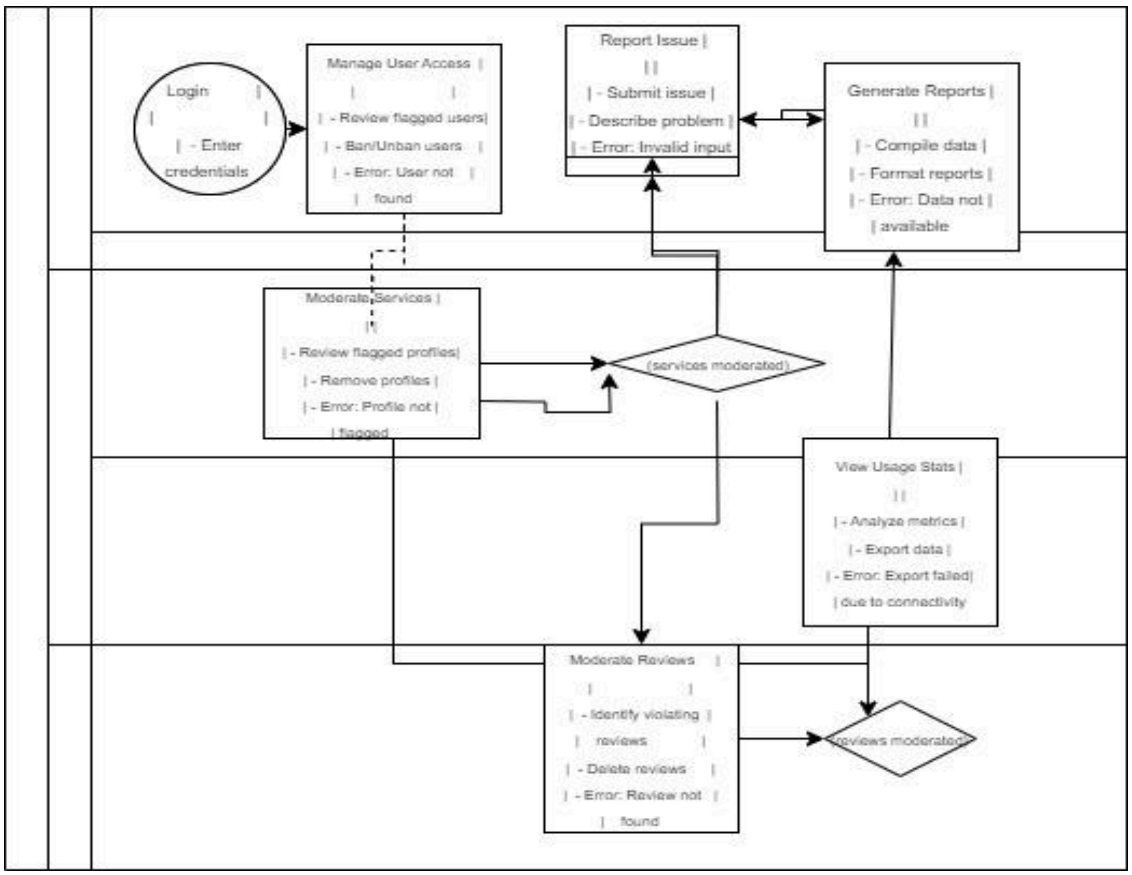
States:

- **Login**
- **Manage User Access**
- **Moderate Content**

- **View Analytics**
- **Generate Reports**
- **Report Issue**

Transitions:

- Login → Manage User Access: Admin logs in and reviews user access.
- Manage User Access → Moderate Content: Admin reviews and moderates content.
- Moderate Content → View Analytics: Admin checks usage statistics after moderation.
- View Analytics → Generate Reports: Admin generates reports based on analytics.
- Report Issue → View Analytics: Admin reports an issue and returns to analytics.



3. Database Schema

Schema Diagram

Tables:

1. **Users**

Field	Type	Constraints
user_id	INT	PK
username	VARCHAR	UNIQUE
password	VARCHAR	
profile_info	TEXT	

2. **PartySpaces**

Field	Type	Constraints
party_id	INT	PK
creator_id	INT	FK for Users(user_id)
game_title	VARCHAR	
description	TEXT	
visibility	ENUM	(‘public’, ‘private’)

3**. Memberships**

Field	Type	Constraints
membership_id	INT	PK
user_id	INT	FK for Users(user_id)
party_id	INT	FK to PartySpaces(party_id)
role	ENUM	(‘member’, ‘admin’, ‘moderator)

4. **Reports**

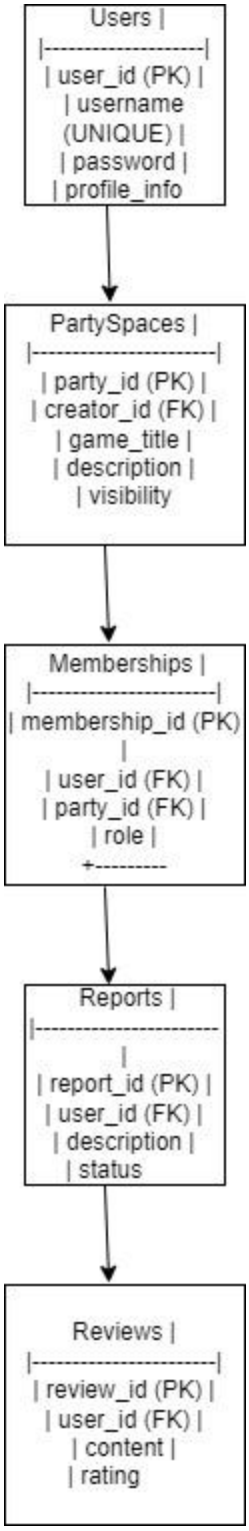
Field	Type	Constraints
report_id	INT	PK
user_id	INT	FK for Users(user_id)
description	TEXT	
status	ENUM	(‘pending’, ‘resolved’)

5. **Reviews**

Field	Type	Constraints
review_id	INT	PK
user_id	INT	FK for Users(user_id)
content	TEXT	
rating	INT	

Relationships

- **Users** can create multiple **PartySpaces**.
- **PartySpaces** can have multiple **Memberships**.
- **Users** can report issues and provide **Reviews**.
- **Admins** can manage **Reports** based on user activity.


Users |
|----------|
| user_id (PK) |
| username (UNIQUE) |
| password |
| profile_info
PartySpaces |
|----------|
| party_id (PK) |
| creator_id (FK) |
| game_title |
| description |
| visibility
Memberships |
|----------|
| membership_id (PK) |
| user_id (FK) |
| party_id (FK) |
| role |
Reports |
|----------|
| report_id (PK) |
| user_id (FK) |
| description |
| status
Reviews |
|----------|
| review_id (PK) |
| user_id (FK) |
| content |
| rating

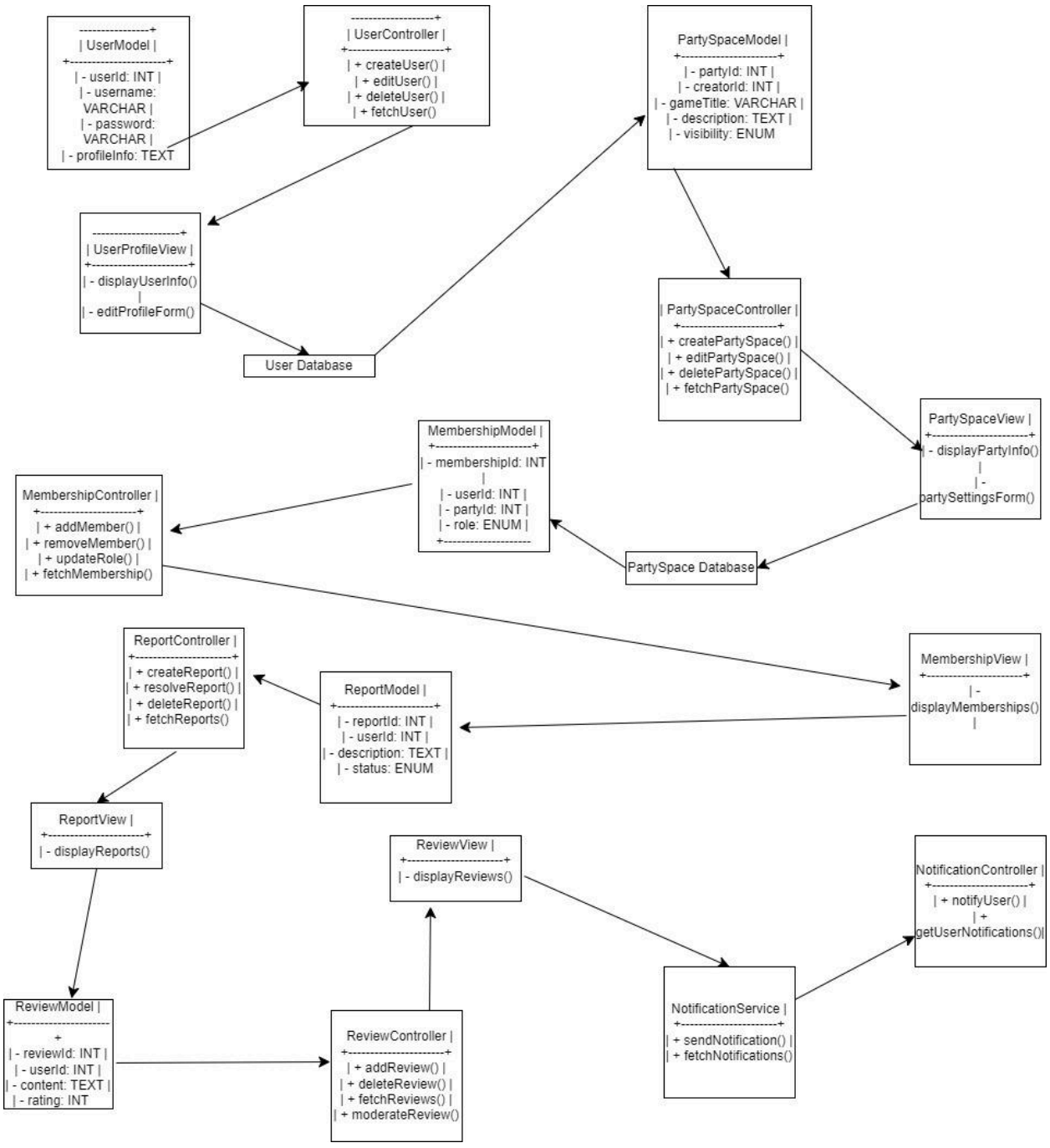
4. Software Architecture

MVC Architecture Diagram

- **Models:**
  - **UserModel**
  - **PartySpaceModel**
  - **MembershipModel**
  - **ReportModel**
  - **ReviewModel**
- **Controllers:**
  - **UserController**
  - **PartySpaceController**
  - **MembershipController**
  - **ReportController**
  - **ReviewController**
- **Views:**
  - **UserProfileView**
  - **PartySpaceView**
  - **MembershipView**
  - **ReportView**
  - **ReviewView**

Relationships

- The **UserController** interacts with **UserModel** to handle user-related actions.
- The **PartySpaceController** manages **PartySpaceModel** for creating and managing PartySpaces.
- **MembershipController** links users to PartySpaces via **MembershipModel**.
- **ReportController** overseas user reports through **ReportModel**.
- **ReviewController** handles reviews through **ReviewModel**.


UserModel
- userId: INT
- username: VARCHAR
- password: VARCHAR
- profileInfo: TEXT
UserController
+ createUser()
+ editUser()
+ deleteUser()
+ fetchUser()
PartySpaceModel
- partyId: INT
- creatorId: INT
- gameTitle: VARCHAR
- description: TEXT
- visibility: ENUM
UserProfileView
- displayUserInfo()
- editProfileForm()
User Database
PartySpaceController
+ createPartySpace()
+ editPartySpace()
+ deletePartySpace()
+ fetchPartySpace()
PartySpaceView
- displayPartyInfo()
- partySettingsForm()
MembershipModel
- membershipId: INT
- userId: INT
- partyId: INT
- role: ENUM
MembershipController
+ addMember()
+ removeMember()
+ updateRole()
+ fetchMembership()
PartySpace Database
ReportController
+ createReport()
+ resolveReport()
+ deleteReport()
+ fetchReports()
ReportModel
- reportId: INT
- userId: INT
- description: TEXT
- status: ENUM
MembershipView
- displayMemberships()
ReportView
- displayReports()
ReviewView
- displayReviews()
NotificationController
+ notifyUser()
+ getUserNotifications()
ReviewModel
- reviewId: INT
- userId: INT
- content: TEXT
- rating: INT
ReviewController
+ addReview()
+ deleteReview()
+ fetchReviews()
+ moderateReview()
NotificationService
+ sendNotification()
+ fetchNotifications()