Software Requirements Specification

for

Arena Connect

Version 1.0

Prepared by

Darshan Kania	**23ITUOZ054**	**23ituoz054@ddu.ac.in**
Hirav Pansuriya	**23CEUOS096**	**23ceuos096@ddu.ac.in**
Manthan Parekh	**23CEUOS098**	**23ceuos098@ddu.ac.in**

Instructor: Mrudang T. Mehta

Course: B.Tech

Lab Section: *Software Engineering Priciple and Practices*

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version	00/00/00

Version	Primary Author(s)	Description of Version	Date Completed
		is being upgraded.	

1 Introduction

1.1 Document Purpose

This document outlines the software requirements for the ArenaConnect platform. It provides a comprehensive description of the system's functionality and intended behavior. This SRS covers all aspects of the platform, including user profile management, game data aggregation, event management, community interaction, and administrative tools.

1.2 Product Scope

ArenaConnect is a centralized gaming platform designed to simplify and enhance the gaming experience for players and organizations. The platform consolidates real-time game data, offers event management tools, and fosters community engagement. Key benefits include:

- Streamlined access to real-time stats and gameplay metrics from popular games.
- Enhanced visibility and participation in gaming events.
- Improved user interaction through social and community features.
- Administrative tools for effective event and user management. ArenaConnect aims to bridge gaps in the gaming ecosystem, promoting growth, engagement, and seamless connectivity.

1.3 Intended Audience and Document Overview

This document is intended for the following audiences:

- **Developers:** To understand the functional and non-functional requirements for system implementation.
- **Project Managers:** To track project scope and deliverables.
- **Marketing Staff:** To identify features for promotional purposes.
- **Testers:** To design and execute test cases.

- **Clients and Professors:** To evaluate the project's vision and ensure alignment with goals.

1.4 Definitions, Acronyms and Abbreviations

- **API:** Application Programming Interface
- **ArenaConnect:** The project name for the centralized gaming platform
- **FIFA:** A popular football simulation video game series
- **PUBG:** PlayerUnknown's Battlegrounds
- **SRS:** Software Requirements Specification
- **UI:** User Interface
- **UX:** User Experience

1.5 Document Conventions

This document adheres to the IEEE standard format for SRS documents. The following conventions are observed:

- **Font:** Arial, size 12 for standard text, and size 16 for headings.
- **Highlighting:** Italics are used for comments and placeholders. Bold text is used for section headings.
- **Spacing:** Single-spaced text with 1" margins.
- **Section Numbering:** Sections and subsections are numbered hierarchically (e.g., 1, 1.1, 1.2).

1.6 References and Acknowledgments

- IEEE Recommended Practice for Software Requirements Specifications (IEEE 830-1998).
- API documentation for supported games such as Call of Duty, Clash of Clans, Chess.com, FIFA, and PUBG.
- Online gaming event management best practices.

- Acknowledgment to professors and mentors for guidance and support in developing the project concept and requirements.

2 Overall Description

2.1 Product Overview

ArenaConnect is a comprehensive platform that integrates gaming data, event management, and community engagement features into a unified system. Designed to address inefficiencies in the gaming industry, it offers tools for gamers and organizations to connect, collaborate, and grow. The platform ensures a secure and user-friendly experience while being accessible across multiple devices.

2.2 Product Functionality

The following are the major functions the ArenaConnect platform must perform:

- **User Profile Management:**
    - Create and manage user profiles.
    - Integrate and display real-time game data.
    - Track and compare performance metrics.
- **Real-Time Game Data Aggregation:**
    - Fetch and display real-time game statistics and leaderboards.
    - Provide a cohesive dashboard for multiple games.
- **Event Management:**
    - Enable event creation, discovery, and participation.
    - Send notifications for event updates and registrations.
- **Community and Interaction Features:**
    - Offer social features such as friend requests and discussion boards.
    - Facilitate communication via live chat and forums.
- **Admin Tools:**
    - Provide tools for event moderation and user management.

- Generate analytics for event performance and user engagement.

2.3 Design and Implementation Constraints

The development of ArenaConnect is subject to the following constraints:

- **Hardware Limitations:**
  - The system must support real-time data updates with minimal latency.
  - Hosting must ensure scalability to handle high traffic during peak gaming events.
- **Interfaces to Other Applications:**
  - Integration with APIs for supported games, including Call of Duty, Clash of Clans, Chess.com, FIFA, and PUBG.
- **Technologies and Tools:**
  - The COMET method for software design.
  - UML modeling language for system diagrams and workflows.
  - Django for backend development and React.js for frontend development.
- **Security Considerations:**
  - Implement robust authentication and authorization mechanisms.
  - Secure API communication using HTTPS and OAuth 2.0 protocols.

2.4 Assumptions and Dependencies

The following assumptions and dependencies are critical to the project:

- ***Assumptions:***
  - *All game APIs will provide reliable and timely data.*
  - *Users will have internet access to use the platform effectively.*
  - *Event organizers will adopt the platform for event creation and management.*
- ***Dependencies:***
  - *Availability of third-party game APIs for real-time data integration.*

- *Hosting infrastructure capable of scaling with user demand.*

- *Libraries and frameworks such as Django, React.js, and PostgreSQL for development.*

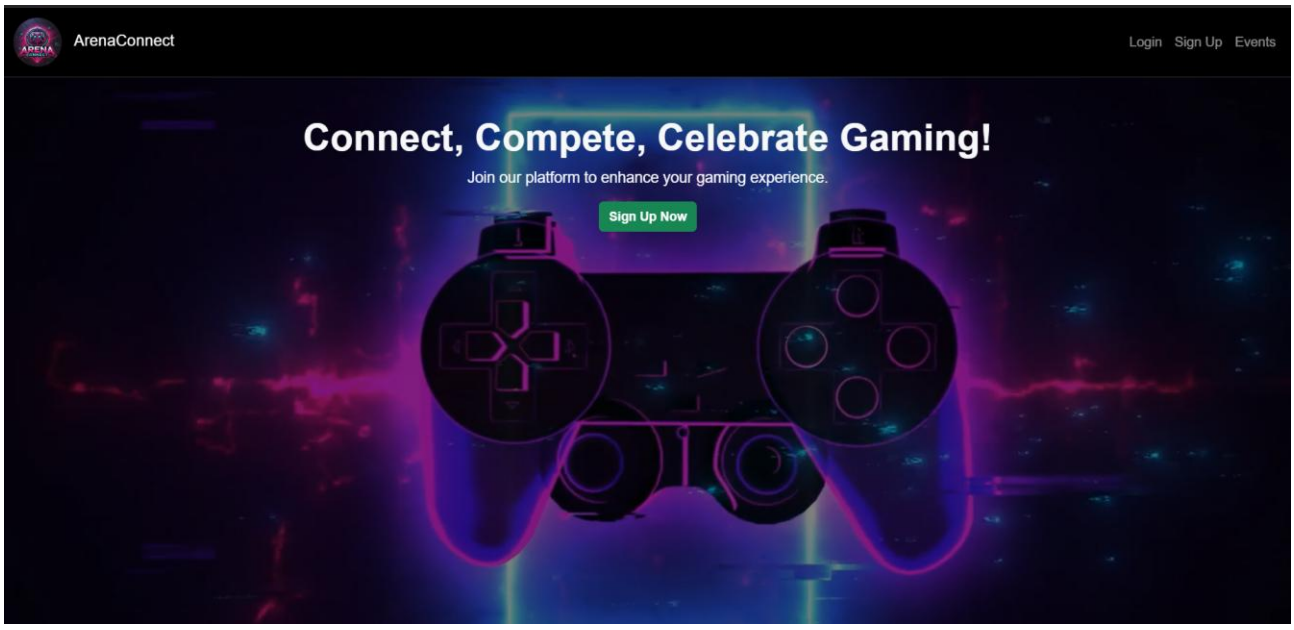
3 Specific Requirements

3.1 External Interface Requirements

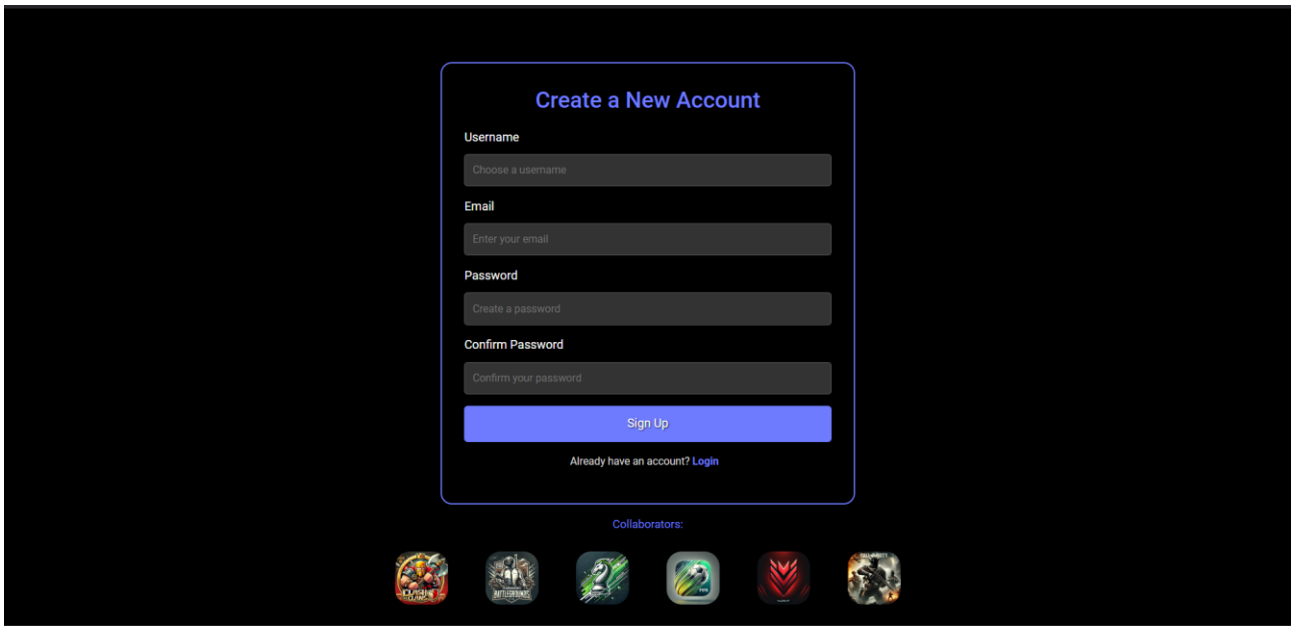
3.1.1 User Interfaces

ArenaConnect will feature a clean and intuitive interface that allows users to easily navigate and interact with the platform. Screenshots of the interface will showcase:

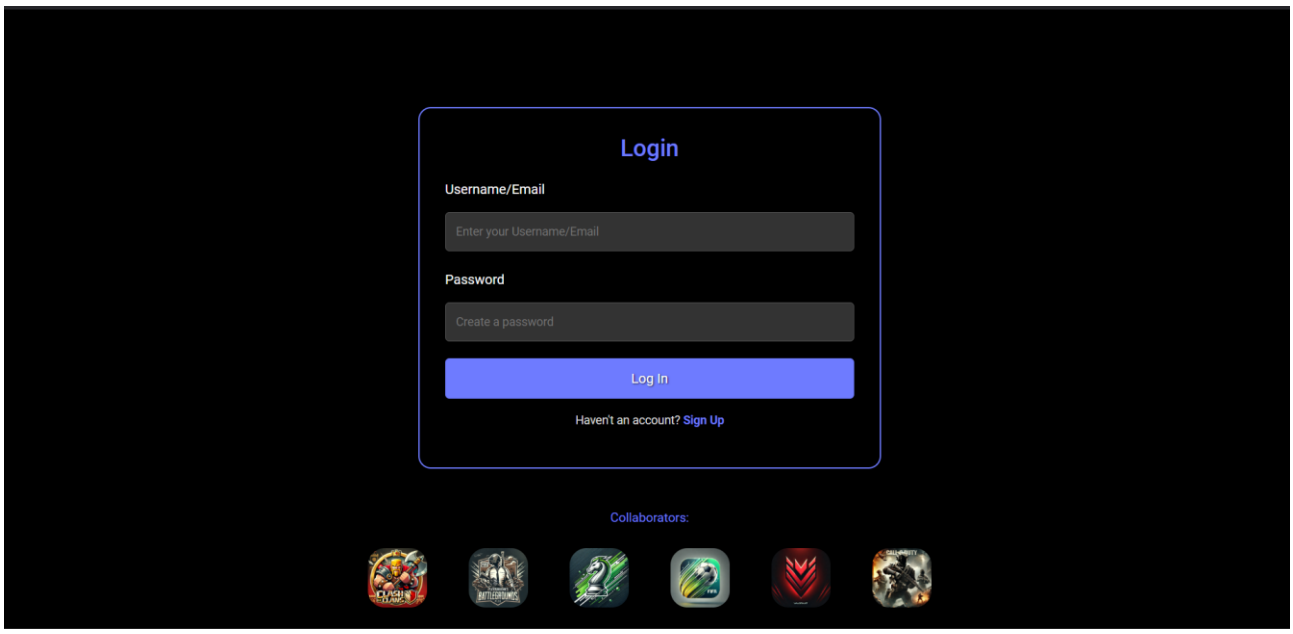
Home Page :



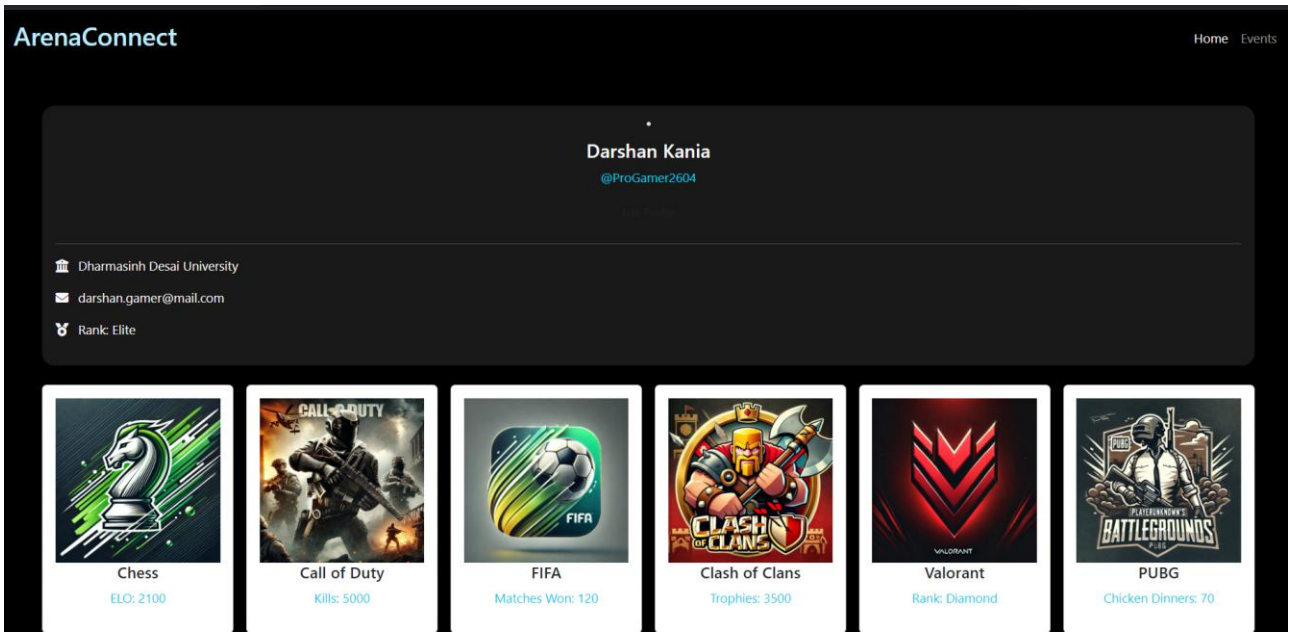
Signup Page :

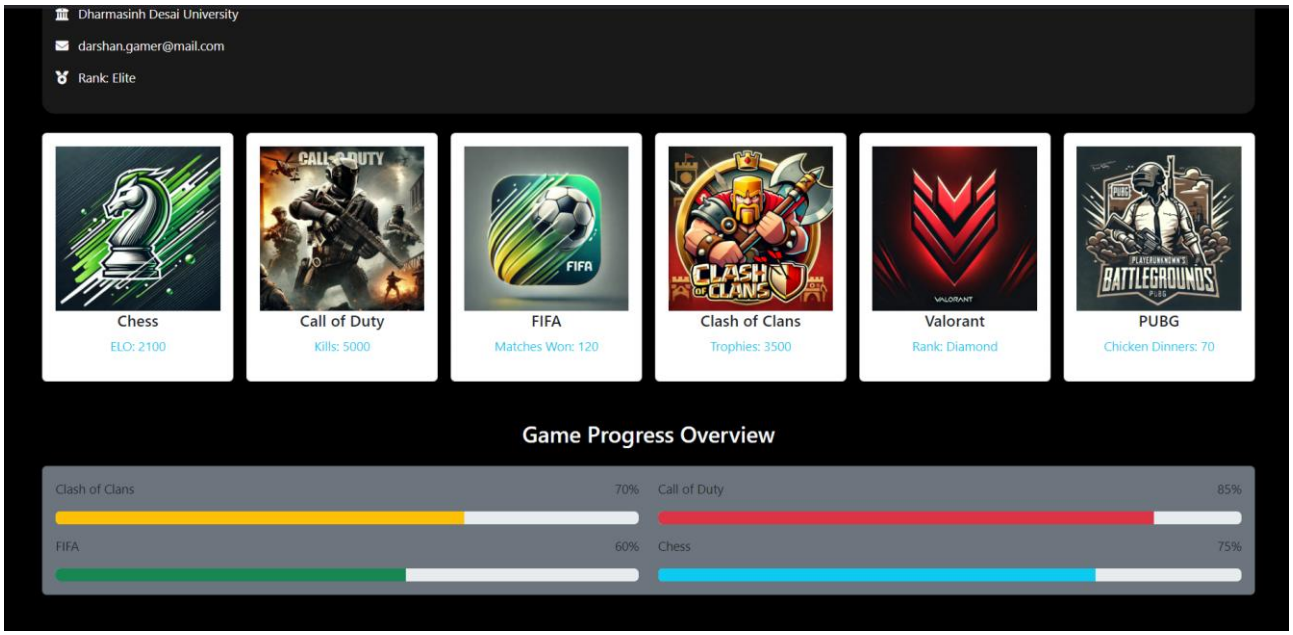


Login Page :

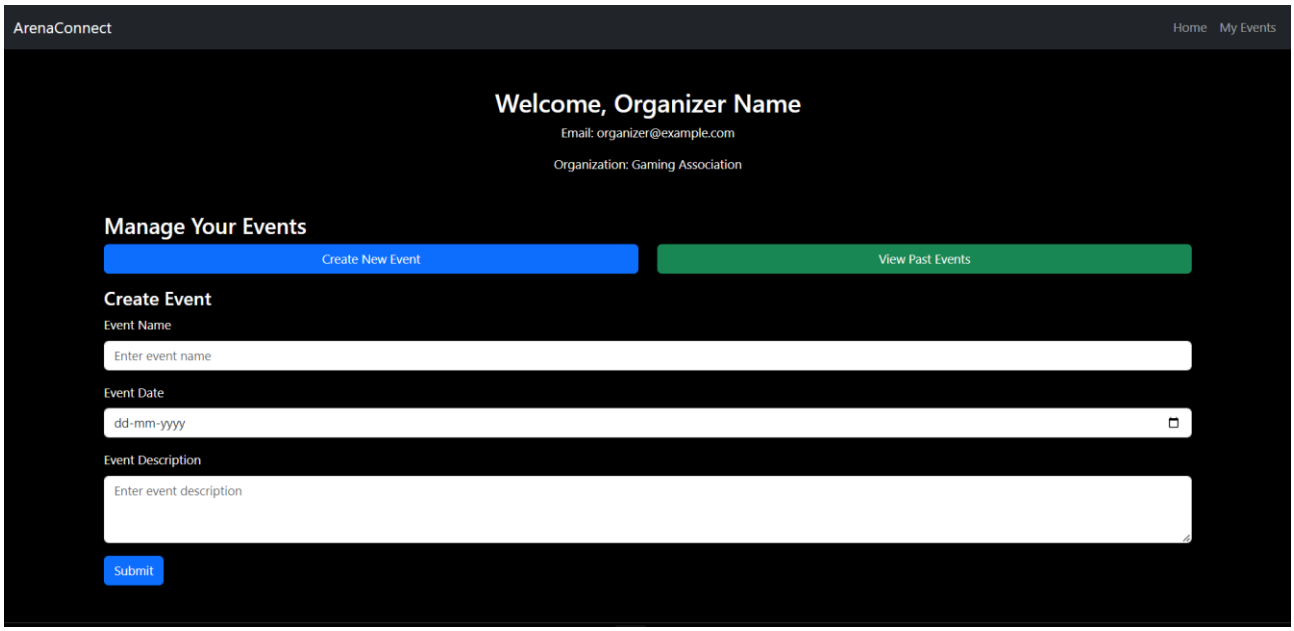


Profile Page :

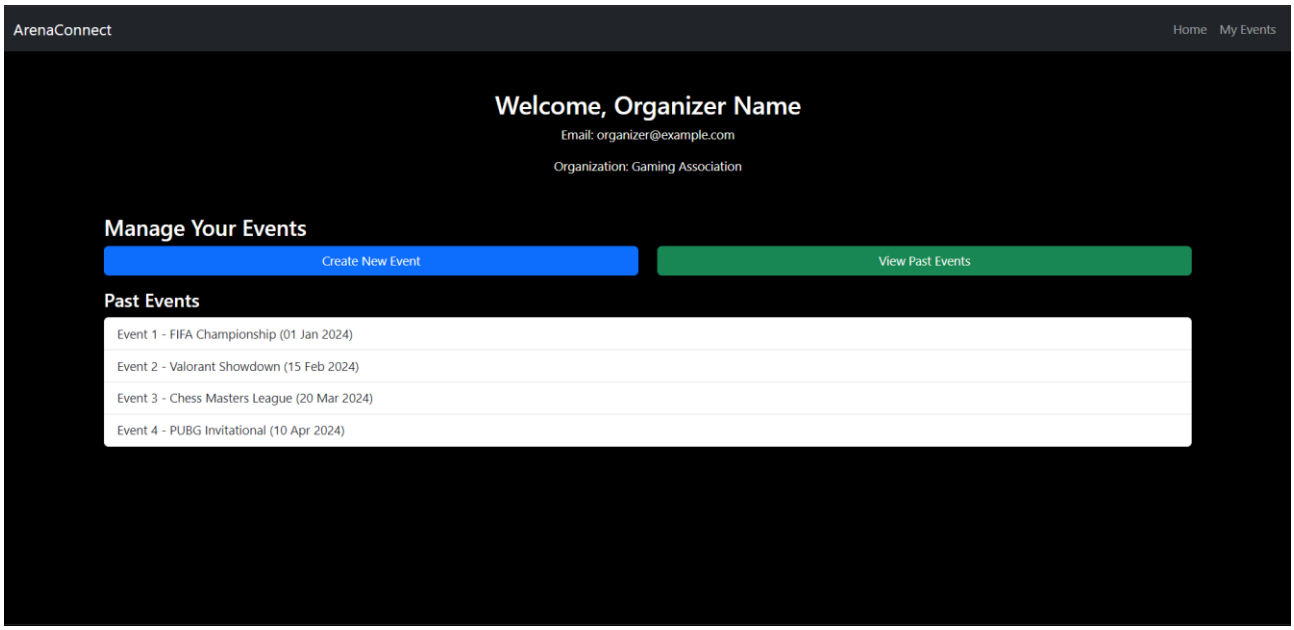




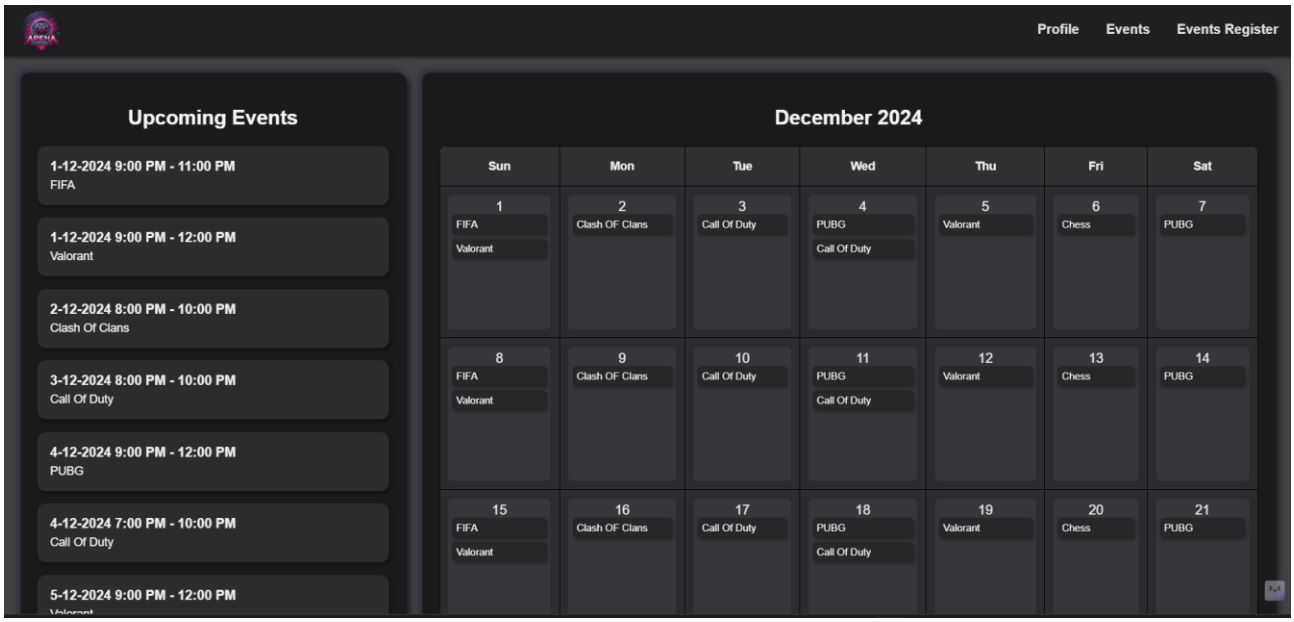
Organizer Create Event :



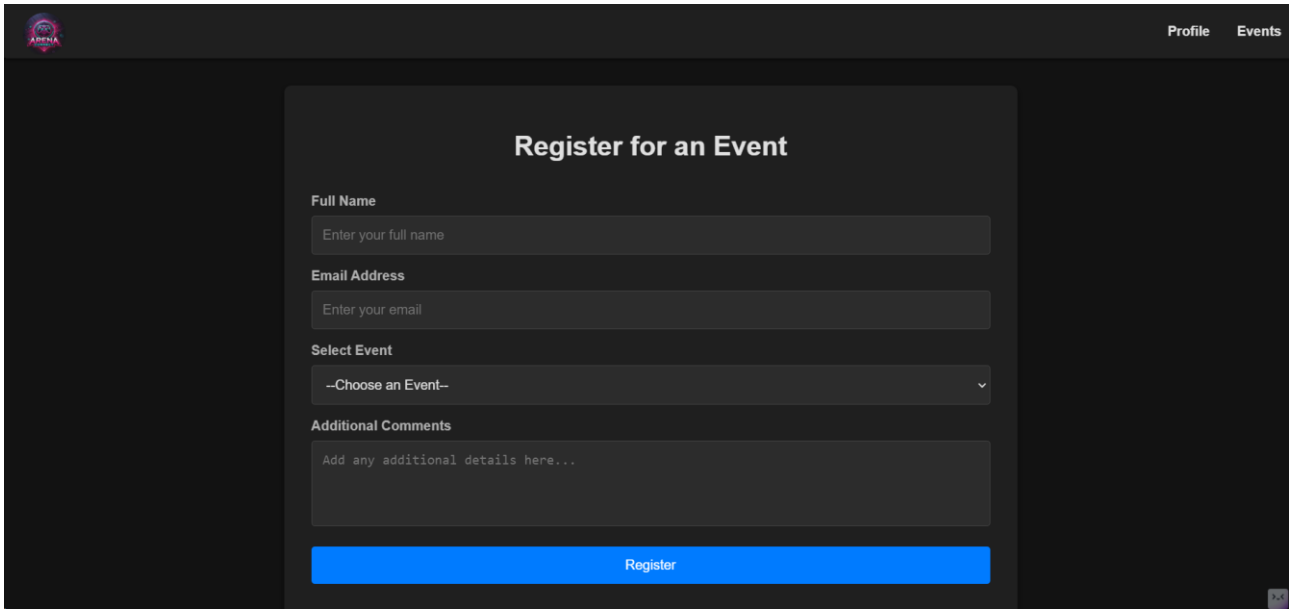
Organizer Past Event :



Upcoming Events :



User Event Registration :



3.1.2 Hardware Interfaces

ArenaConnect interacts with the following hardware components:

- **User Devices:** Desktop, laptop, and mobile devices supporting web browsers.
- **Server Infrastructure:** Hosts the backend, ensuring scalability and reliability.

- **Third-Party Game APIs:** Connected via secure API endpoints.

3.1.3 Software Interfaces

The platform integrates with multiple software components:

- **Game APIs:** Fetches player data from Call of Duty, Clash of Clans, Chess.com, FIFA, and PUBG.
- **Authentication Services:** Ensures secure user login through OAuth 2.0.
- **Messaging System:** Integrated with Discord for live communication.
- **Database Management:** Utilizes PostgreSQL for user and event data.

3.2 Functional Requirements

- **Player Functional Requirements**

1. **Profile Management**

    - **3.2.1 : Players shall create and update their profiles.**
        - **Input:** Player details such as username, email, and preferences.
        - **Processing:** Validate input data, store it in the database, and associate it with a unique user ID.
        - **Output:** Confirmation of successful profile creation or update.
        - **State:** Profile creation mode or update mode.
        - **Precondition:** User must provide valid input data.
        - **Postcondition:** Profile is stored and accessible for future actions.
        - **Exception Scenario:** Invalid input data results in an error message.
    - **3.2.2 : The system shall fetch and display real-time player stats.**
        - **Input:** Player identifier and API request for stats.
        - **Processing:** Query external game APIs for player-specific stats and aggregate the data.
        - **Output:** Display of player stats including rankings, achievements, and progress.
        - **Precondition:** Valid player profile exists.
        - **Postcondition:** Real-time stats are presented on the player dashboard.
        - **Exception Scenario:** API failure or unavailability results in a notification.

2. **Gaming Insights**

    - **3.2.3 : The system shall aggregate data from multiple game APIs into a single dashboard.**
        - **Input:** API requests to multiple game platforms.

- **Processing:** Fetch and standardize data from various APIs for display.
- **Output:** Consolidated gaming stats in a user-friendly format.
- **Precondition:** Supported games are linked to the player profile.
- **Postcondition:** Aggregated stats are accessible in real-time.
- **Exception Scenario:** Incompatible API responses are flagged for troubleshooting.

- **3.2.4 : Players shall view leaderboards and compare rankings.**
  - **Input:** Player selection of leaderboard filters (e.g., global, friends-only).
  - **Processing:** Query leaderboard data and apply selected filters.
  - **Output:** Display of filtered leaderboard data.
  - **Precondition:** Leaderboard data is available.
  - **Postcondition:** Relevant leaderboard data is displayed.
  - **Exception Scenario:** No data available for selected filters.

3. Event Participation

- **3.2.5 : Players shall register for events directly from the platform.**
  - **Input:** Event selection and player registration details.
  - **Processing:** Validate registration and add the player to the event participant list.
  - **Output:** Confirmation of successful registration.
  - **Precondition:** Event must be open for registration.
  - **Postcondition:** Player is added to the event roster.
  - **Exception Scenario:** Event registration closes during submission.

- **3.2.6 : Notifications shall alert players about event updates.**
  - **Input:** Event update triggers.
  - **Processing:** Generate and send notifications to registered players.
  - **Output:** Notification about event changes (e.g., timing, location).
  - **Precondition:** Player is registered for the event.
  - **Postcondition:** Player is informed of updates.
  - **Exception Scenario:** Notification delivery failure.

• Organizer Functional Requirements

1. Event Management

- **3.2.7 : Organizers shall create public and private events with detailed descriptions.**
  - **Input:** Event details such as name, type, date, time, and rules.
  - **Processing:** Validate and store event information in the system.
  - **Output:** Confirmation of event creation and visibility based on privacy settings.
  - **Precondition:** Organizer is authenticated.
  - **Postcondition:** Event is listed in the system for eligible users.
  - **Exception Scenario:** Duplicate or conflicting event details are flagged.

- **3.2.8 : The system shall provide insights into event participation.**

  - **Input:** Request for event analytics.

- **Processing:** Aggregate participation data and generate insights.
- **Output:** Dashboard with event metrics (e.g., participant count, engagement rate).
- **Precondition:** Event must be active or completed.
- **Postcondition:** Insights are available for organizer review.
- **Exception Scenario:** Insufficient participation data limits insights.

2. **Moderation Tools**

- **3.2.9 : Organizers shall moderate user interactions during events.**

  - **Input:** Reports of inappropriate behavior or rule violations.
  - **Processing:** Review reports, take necessary actions (e.g., warnings, bans).
  - **Output:** Moderation actions and resolution status.
  - **Precondition:** Event must be ongoing.
  - **Postcondition:** User behavior is moderated.
  - **Exception Scenario:** Delayed actions due to insufficient evidence.

- **Admin Functional Requirements**

1. **User Management**

- **3.2.10 : Admins shall manage user accounts, including banning inappropriate users.**

  - **Input:** User reports or system flags.
  - **Processing:** Review cases and apply appropriate actions (e.g., ban, suspension).
  - **Output:** User status updates and notification of actions.
  - **Precondition:** Valid reports or evidence.
  - **Postcondition:** User accounts are managed to ensure platform integrity.
  - **Exception Scenario:** Appeals or challenges to admin actions.

2. **System Analytics**

- **3.2.11 : The system shall generate platform-wide analytics for user engagement.**

  - **Input:** Request for analytics.
  - **Processing:** Aggregate and analyze platform activity data.
  - **Output:** Reports with key metrics (e.g., active users, event participation).
  - **Precondition:** System data is up-to-date.
  - **Postcondition:** Analytics are generated and accessible to admins.
  - **Exception Scenario:** Data discrepancies affect report accuracy.

3.3 Use Case Model

4 Other Non-functional Requirements

4.1 Performance Requirements

P1: The system shall fetch real-time game statistics and update user dashboards within 3 seconds after receiving data from external game APIs.

P2: The platform must support at least 10,000 concurrent users with no more than 5% performance degradation during peak hours.

P3: Notifications about events or in-game milestones must be delivered to users within 5 seconds of the trigger.

P4: The system should render the dashboard and other core pages in under 2 seconds for users on a standard broadband connection (minimum 10 Mbps).

P5: Event registration processing must complete in under 2 seconds, including sending confirmation to the user.

4.2 Safety and Security Requirements

S1: User authentication shall be implemented using secure protocols (e.g., OAuth 2.0) to protect user credentials.

S2: All sensitive user data, including passwords, shall be encrypted both in transit and at rest.

S3: The system shall log all administrative actions for auditing purposes and ensure these logs are tamper-proof.

S4: In the event of unauthorized access attempts, the system shall lock the account after 5 failed login attempts and notify the user.

4.3 Software Quality Attributes

Reliability :

- The system shall maintain a 99.9% uptime, with redundancy mechanisms to handle server failures.
- Scheduled maintenance windows shall not exceed 2 hours per month.

Usability :

- The platform interface shall conform to accessibility standards (e.g., WCAG 2.1 AA) to ensure usability for users with disabilities.
- User feedback mechanisms shall be implemented to capture and address usability issues proactively.

Maintainability :

- The system shall modularize components to simplify updates and modifications.
- Documentation shall be provided for all APIs, ensuring ease of integration and debugging.

5 Other Requirements

*<This section is **Optional**. Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

Appendix A – Data Dictionary

<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>

Appendix B - Group Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>