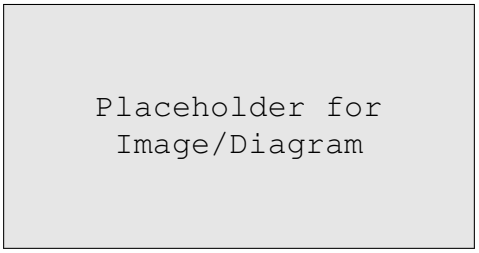
Software Requirements Specification

for



EventHive

Version 1.0

Prepared by: Gemini
Authored for: Aditya Bhardwaj, Yogesh Rana, Rohit Tripathy, Nikshay Kataria

September 19, 2025

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a complete and comprehensive description of the intended purpose, features, and behavior of the EventHive platform, Version 1.0. Its primary goal is to establish a clear, unambiguous, and verifiable agreement between the development team and all stakeholders on the exact functionalities and constraints of the software product.

This document rigorously details the functional requirements, non-functional requirements, external interfaces, and design constraints of the system. It is intended to serve as the single source of truth and foundational guide for project managers, system architects, designers, developers, and quality assurance teams throughout the entire software development lifecycle—from architectural design and implementation to system testing and future enhancements. By meticulously defining the scope and specifications of EventHive, this SRS aims to minimize development ambiguity, reduce rework, provide a solid basis for cost and schedule estimation, and establish a formal baseline for validation and verification activities.

1.2 Document Conventions

This document adheres to the IEEE Std 830-1998 standard for Software Requirements Specifications. All requirements are stated in a clear, concise, and testable manner.

- **Requirement Language:** All functional and non-functional requirements are expressed using the keyword "shall" to denote a mandatory provision (e.g., "The system shall...").
- **Emphasis: Bold text** is used for emphasis or to introduce key terms defined in the glossary.
- **Requirement Identifiers:** Each requirement is uniquely identified with a tag for traceability throughout the project lifecycle:
  - **[FEAT-XXX]:** A high-level feature or capability of the system.
  - **[FUNC-XXX]:** A specific, verifiable functional requirement derived from a feature.
  - **[NFR-XXX]:** A non-functional requirement, such as performance, security, or reliability.
  - **[IF-XXX]:** A requirement related to an external interface.

1.3 Intended Audience and Reading Suggestions

This SRS is written for a diverse audience. The following reading suggestions are provided to guide different stakeholders:

- **Project Stakeholders & Clients:** Should focus on the Introduction (Section 1) and Overall Description (Section 2) to ensure the system's business goals and scope align with the project vision. A review of the high-level features in Section 3 is also recommended.
- **Project Managers:** Should use this entire document for project planning, scope management, scheduling, and tracking. Sections 2, 3, and 5 are particularly critical for defining project milestones and resource allocation.
- **System Architects & Designers:** Will use the detailed requirements in Section 3 (System Features), Section 4 (External Interfaces), and Section 5 (Nonfunctional Requirements) as the primary input for creating high-level and detailed design specifications.
- **Software Developers:** Will refer to Sections 3, 4, and 5 for the precise details required for implementation, ensuring their code meets the specified functional and non-functional criteria.

- **QA & Testing Teams:** Will use the requirements in Sections 3, 4, and 5 as the basis for developing comprehensive test plans, test cases, and acceptance criteria to formally validate and verify the system.
- **Future Maintenance & Support Teams:** Will use this document as a foundational reference to understand the system's intended functionality, architecture, and constraints during operational support and future enhancement phases.

1.4 Product Scope

EventHive is a first-of-its-kind, comprehensive platform designed specifically for the Indian market that merges interactive, map-based event discovery with a multi-layered, real-time community chat system. The platform's vision is to connect local communities and foster nationwide conversations by providing an immersive, end-to-end event experience.

The core functionalities of the product include:

- **Interactive Event Discovery:** Users can visually explore a dynamic map populated with events, utilizing advanced, multi-faceted filters to find activities based on location, category, date, price, and popularity.
- **Community-Driven Event Creation:** Any authenticated user can create and publish events, with a formal verification system in place for trusted organizers such as colleges, NGOs, and clubs to enhance credibility.
- **Integrated Ticketing:** The system provides a seamless, secure ticketing process, supporting both free and paid events with integrated UPI and digital wallet payments native to the Indian financial ecosystem.
- **Multi-Layered Chat System:** The platform features a sophisticated hierarchical chat ecosystem, including temporary, event-specific chatrooms, hyper-local city-level discussion groups, state-wise communities, and a global "All India" chat network, with provisions for user anonymity.
- **Post-Event Engagement:** An "After-Event Hub" facilitates continued, time-limited connections among attendees post-event, fostering a community that lasts beyond the event itself.
- **Gamification and Safety:** The platform will incorporate gamification elements like badges, points, and leaderboards to drive engagement, alongside advanced, proactive safety features including AI-powered content moderation and robust user-reporting tools.

The system is intended to cover the complete event lifecycle, from initial discovery and planning to live engagement and post-event connections, thereby creating a vibrant, self-sustaining community-driven ecosystem.

1.5 References

1. IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*.
2. EventHive Project Vision Document, Version 1.0.
3. Next.js 14 Documentation, https://nextjs.org/docs
4. Google Maps Platform API Documentation, https://developers.google.com/maps/documentation
5. Razorpay API Integration Guide, https://razorpay.com/docs/api/
6. Socket.io Real-time Engine Documentation, https://socket.io/docs/v4/

7. PostgreSQL Official Documentation, https://www.postgresql.org/docs/

2 Overall Description

2.1 Product Perspective

EventHive is a new, self-contained, and independent product designed to operate as a standalone web application. It is not a component of a pre-existing larger software system but is designed to be a central platform within the social-networking and event-management business ecosystem. While self-contained, its functionality is heavily reliant on a suite of external, third-party services with which it must seamlessly integrate.

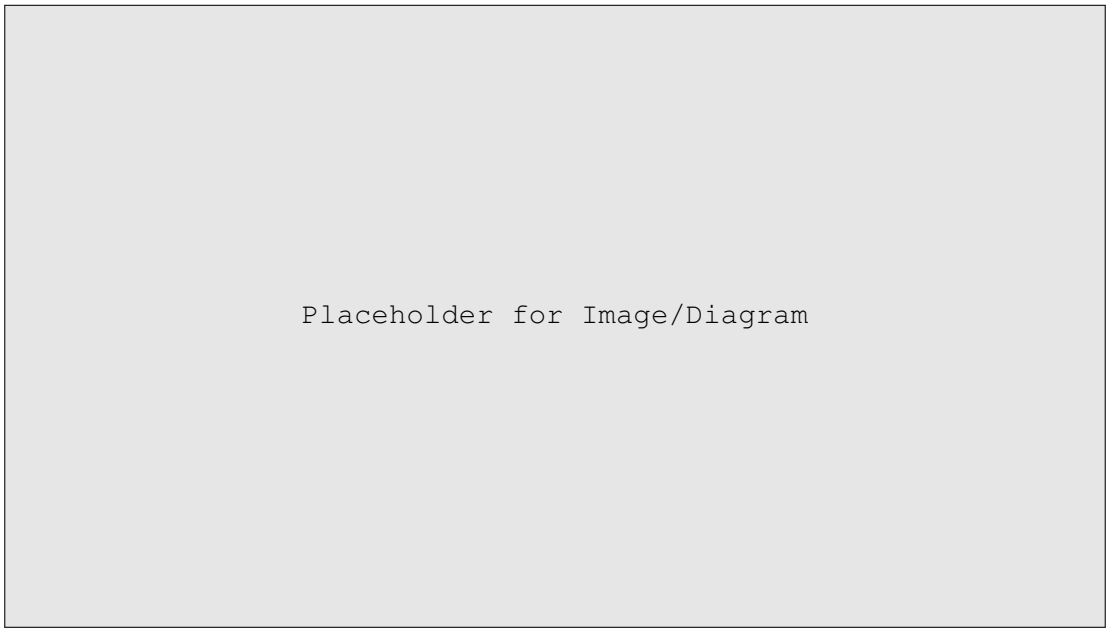


Figure 1: EventHive System Architecture and Interfaces

The system interfaces with:

- **Google Maps/Places API:** For map rendering, event plotting, geocoding, and location-based services.
- **Razorpay/Stripe Payment Gateway:** For processing all UPI and digital wallet payments for event tickets.
- **OAuth 2.0 Providers (Google, Facebook):** For secure, delegated user authentication.
- **Firebase Cloud Messaging:** For dispatching real-time push notifications to users.
- **SendGrid/Twilio:** For transactional email and SMS communications.
- **AWS S3/Cloudinary:** For scalable, secure storage and delivery of user-generated media content.
- **OpenAI API/Custom ML Models:** For proactive, real-time content moderation in chat systems.

2.2 Product Features

The EventHive platform will provide the following key features:

FEAT-100: Smart Event Discovery & Management: A comprehensive toolset for users to discover events via an interactive map, utilize advanced search filters, and create their own events, with a verification system for official organizers.

FEAT-200: Integrated Ticketing System: A secure and seamless system for handling both free and paid event registrations with integrated UPI payment processing.

FEAT-300: Event-Specific Mini Chats: Each event automatically generates a temporary, dedicated chatroom for registered attendees to facilitate pre-event coordination and live engagement.

FEAT-400: The "After-Event" Hub: For events with significant attendance, a private, time-limited group chat is created to encourage post-event connections and memory sharing.

FEAT-500: Global Indian Chat Network: A multi-level system of persistent public chat groups organized by city, state, and an "All India" global channel, supporting anonymous participation.

FEAT-600: Smart & Secure Onboarding: A streamlined, secure registration and onboarding process using social authentication, with clear prompts for necessary permissions.

FEAT-700: Gamification & Rewards Engine: A sophisticated engagement system featuring badges, points, leaderboards, and tangible rewards for active users and successful event organizers.

FEAT-800: Advanced Safety & Moderation: A suite of proactive and reactive tools, including AI-powered content filtering, a comprehensive user reporting/blocking system, and verified user badges to ensure a safe and trustworthy community environment.

2.3 User Classes and Characteristics

2.4 Operating Environment

- **[OE-01]:** The platform shall be a web-based application accessible through the latest stable versions of major web browsers (Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge) on both desktop and mobile operating systems (Windows 10/11, macOS, Linux, iOS, Android).
- **[OE-02]:** The system's backend infrastructure shall be deployed on a major cloud provider (AWS or Google Cloud Platform) and must be architected for high availability and automated scalability to handle dynamic user loads.
- **[OE-03]:** The system requires a persistent, stable internet connection for all functionalities. No offline capabilities are planned for Version 1.0. The application shall detect connection loss and notify the user gracefully.

2.5 Design and Implementation Constraints

- **[DIC-01]:** The frontend web application shall be developed as a Single Page Application (SPA) using Next.js 14 with the App Router.
- **[DIC-02]:** The entire user interface shall be constructed using Tailwind CSS and the shadcn/ui component library to ensure consistency and rapid development.
- **[DIC-03]:** The backend server shall be implemented using Node.js with the Express.js framework.
- **[DIC-04]:** The primary database for all persistent data shall be PostgreSQL (Version 14+).
- **[DIC-05]:** A Redis (Version 6+) instance shall be utilized for high-speed caching, session management, and message brokering for the real-time chat system.
- **[DIC-06]:** Real-time bi-directional communication for the chat system shall be exclusively implemented using the WebSocket protocol, managed by the Socket.io library.
- **[DIC-07]:** User authentication shall be managed using JSON Web Tokens (JWT) compliant with the RFC 7519 standard.

Table 1: User Classes and Characteristics

User Class	Description & Characteristics	Primary Activities
General User / Attendee	Individuals seeking to discover events and connect with communities. Typically tech-savvy urban youth (18-30), college students, and young professionals. Assumed to be proficient with modern web/mobile applications.	Discovering events, purchasing tickets, joining various chat tiers, participating in discussions, sharing media, managing their user profile and privacy settings.
Event Organizer (Community)	Any authenticated user creating and managing an event. Motivations range from organizing small social meetups to larger workshops. Technical expertise is expected to be low to medium.	Creating and publishing detailed event pages, managing event information, viewing attendee lists, using the organizer's dashboard, and communicating with attendees.
Verified Organizer	Official representatives from established entities (colleges, NGOs, clubs, businesses) who have passed a manual verification process. They represent trusted event sources.	All activities of a Community Organizer, plus managing their verified profile. Their events are granted higher visibility and trust signals (badges).
System Administrator	Internal EventHive technical staff. Requires high technical proficiency in backend systems, database management, and security protocols.	Overseeing user account management, managing the organizer verification queue, responding to escalated reports, configuring system parameters, monitoring platform health and analytics.
Content Moderator	Internal staff or highly trusted community members. Requires training in platform policies and moderation tools, but not deep technical expertise.	Actively reviewing content flagged by the AI system and user reports, issuing warnings, enforcing temporary mutes or permanent bans, and escalating severe violations.

- **[DIC-08]:** All financial transactions shall be processed through the Razorpay payment gateway API. No payment card or bank account information shall be stored on EventHive servers.

2.6 User Documentation

- **[UD-01]:** An online, searchable Help Center/FAQ section shall be accessible within the application, providing detailed guides and troubleshooting steps for all user-facing features.
- **[UD-02]:** A comprehensive "Organizer's Guide" shall be provided in the Help Center, detailing the end-to-end event creation, management, and post-event analysis process.
- **[UD-03]:** Public-facing API documentation, generated using Swagger/OpenAPI specifications, shall be available at a '/docs' endpoint for potential third-party integrations and partnerships.

2.7 Assumptions and Dependencies

- **[AD-01]:** The system assumes users will grant necessary browser location permissions for the map-based features to function optimally. The system must degrade gracefully, allowing manual location input if permission is denied.

- **[AD-02]:** The functionality, accuracy, and rate limits of location services are entirely dependent on the Google Maps and Google Places APIs. The system must handle API failures and rate limiting.
- **[AD-03]:** The success and reliability of all payment transactions are dependent on the uptime and operational status of the Razorpay payment gateway.
- **[AD-04]:** The effectiveness of automated content moderation is dependent on the availability, performance, and accuracy of the external AI service (e.g., OpenAI API).
- **[AD-05]:** It is assumed that the target user base has access to and is familiar with using UPI for digital payments.

3 System Features

This section provides a detailed breakdown of the functional requirements for each feature of the EventHive platform.

3.1 FEAT-100: Smart Event Discovery & Management

This feature allows users to find, view, create, and manage events.

3.1.1 FUNC-101: Interactive Map Display

The system shall display an interactive map as the primary interface element on the homepage.

- **FUNC-101.1:** On initial load, the map shall center on the user's detected geographical location if permission is granted.
- **FUNC-101.2:** If location permission is denied, the map shall default to a central location in India (e.g., Delhi) and prompt the user to search for a city.
- **FUNC-101.3:** The user shall be able to pan and zoom the map. The system shall dynamically fetch and display events for the new visible map area.

3.1.2 FUNC-102: Event Pin Exploration

The system shall display distinct, clickable pins on the map for each event.

- **FUNC-102.1:** Clicking a pin shall open a non-intrusive preview pop-up containing the event's title, date, time, banner image, and price.
- **FUNC-102.2:** The preview pop-up shall contain a "View Details" button that navigates the user to the full event page.

3.1.3 FUNC-103: Advanced Search & Filters

The system shall provide a comprehensive set of tools to refine event discovery.

- **FUNC-103.1:** A text-based search bar shall allow users to search for events by keywords in their title or description.
- **FUNC-103.2:** The system shall provide filter options for: Category (e.g., cultural, educational, music, tech), Date range, Price (free, paid), Distance from user's location (e.g., within 5km, 10km, 25km), and Popularity (trending events).
- **FUNC-103.3:** Applied filters shall update the event pins on the map in real-time.

3.1.4 FUNC-104: Community Event Publishing

Any authenticated user shall be able to create and publish an event.

- **FUNC-104.1:** The system shall provide a multi-step event creation form with fields for: title, description (with rich text support), banner image upload, category, start/end date and time, a location search powered by Google Places API, and ticketing options.
- **FUNC-104.2:** All form fields shall have appropriate validation (e.g., end date must be after start date).

3.1.5 FUNC-105: Verified Organizer System

The platform shall distinguish between community and verified organizers.

- **FUNC-105.1:** The system shall provide a form for organizations to apply for "Verified" status, requiring submission of official documents.
- **FUNC-105.2:** System Administrators shall have an interface to review applications and approve or reject them.
- **FUNC-105.3:** Events created by Verified Organizers shall display a prominent verification badge on the event pin, pop-up, and detail page.

3.1.6 FUNC-106: Integrated Ticketing

The system shall manage the entire ticketing lifecycle.

- **FUNC-106.1:** Organizers shall be able to set ticket prices (including $0 for free events) and quantity limits.
- **FUNC-106.2:** Users shall be able to book tickets for events directly via a checkout process that integrates with the UPI payment gateway.
- **FUNC-106.3:** Upon successful payment, the system shall generate a unique digital ticket (with QR code) for the user, accessible in their profile and sent to their registered email.

3.2 FEAT-300 & 500: Community Ecosystem

This feature set covers all chat and community interaction functionalities.

3.2.1 FUNC-201: Event-Specific Chat

- **FUNC-201.1:** For every event, a temporary, dedicated chatroom shall be automatically created.
- **FUNC-201.2:** Only users who have a confirmed ticket for an event shall be automatically added to and granted access to the corresponding event chatroom.
- **FUNC-201.3:** Event chatrooms shall have a read-only "Announcements" channel where only the event organizer can post messages.
- **FUNC-201.4:** Event chatrooms shall be automatically archived (made read-only) 48 hours after the event concludes.

3.2.2 FUNC-202: The "After-Event" Hub

- **FUNC-202.1:** For events with 10 or more confirmed attendees, the system shall automatically create a private "After-Event Hub" group chat upon the event's conclusion.
- **FUNC-202.2:** This hub shall remain active for 48 hours, allowing attendees to share photos, provide feedback, and exchange contact information.

3.2.3 FUNC-203: Public Chat Network

- **FUNC-203.1:** The system shall host persistent public chat groups for major cities, all states, and a global "All India" channel.
- **FUNC-203.2:** Any authenticated user shall be able to join or leave these public chat groups at will from a central chat directory.
- **FUNC-203.3:** When participating in public chats, users shall have a toggle option to post messages anonymously. Anonymous messages shall be attributed to a generic, non-identifiable "Anonymous User" handle.

3.3 FEAT-600: Smart Onboarding

This feature covers user registration and initial setup.

3.3.1 FUNC-301: Seamless Registration

- **FUNC-301.1:** The system shall allow new users to register via a traditional email/password form or using their Google or Facebook accounts (OAuth 2.0).
- **FUNC-301.2:** Email registration shall require a verification step where the user must click a link sent to their email address.

3.3.2 FUNC-302: Onboarding Process

- **FUNC-302.1:** Upon first login, the system shall guide the user through a brief onboarding process, prompting for location permissions and interest selection to personalize their experience.
- **FUNC-302.2:** The system shall require each user to create a unique username, with real-time validation for availability.

3.4 FEAT-700: Gamification & Rewards

3.4.1 FUNC-401: Organizer Rewards

The system shall automatically issue a digital coupon (e.g., for platform fee discounts on future events) to any Verified Organizer whose event successfully sells over 100 paid tickets.

3.4.2 FUNC-402: Achievement Badges

The system shall define and award users with digital badges for milestones. These badges shall be displayed on the user's public profile. Examples include:

- **FUNC-402.1:** "Explorer" - Attended 5 events.
- **FUNC-402.2:** "Creator" - Organized a successful event (10+ attendees).
- **FUNC-402.3:** "Socialite" - Sent 100+ messages in city chats.

3.4.3 FUNC-403: Community Leaderboards

The system shall display monthly leaderboards for each city and state, ranking users based on a weighted community engagement score derived from messages sent, events attended, and events organized.

3.5 FEAT-800: Advanced Safety Features

3.5.1 FUNC-501: Reporting and Blocking

- **FUNC-501.1:** Within any chat, users shall be able to report a specific message or a user profile. The report shall require selecting a reason (e.g., spam, harassment) and shall be sent to a moderation queue.
- **FUNC-501.2:** Users shall be able to block other users, which will hide all messages from the blocked user and prevent them from initiating private contact.

3.5.2 FUNC-502: AI Content Moderation

The system shall integrate an AI service to automatically scan all public chat messages in real-time for content violating community guidelines. Messages exceeding a defined toxicity threshold shall be automatically hidden and flagged for moderator review.

3.5.3 FUNC-503: Privacy Controls

The system shall provide users with a dedicated privacy settings page to control the public visibility of their event history, friend list, and other profile details.

4 External Interface Requirements

4.1 User Interfaces

The system's UI shall be clean, modern, and intuitive.

- **[UI-01]: Homepage / Map View:** A full-screen, responsive map view with easily accessible search and filter controls. Event pins must be clearly visible and not overly cluttered. The event detail pop-up shall be a modal overlay.
- **[UI-02]: Event Detail Page:** A dedicated page for each event, showcasing all details, including a prominent "Book Ticket" call-to-action, a map view of the location, organizer information, and an entry point to the event chat (if eligible).
- **[UI-03]: Chat Interface:** A multi-panel chat interface, allowing users to easily switch between their joined event, city, and state chats. The interface shall support text, emojis, and image uploads with previews.
- **[UI-04]: Organizer Dashboard:** A dedicated dashboard for organizers with sections for managing created events, viewing real-time sales analytics, and a tool for sending announcements to attendees.

4.2 Hardware Interfaces

As a web-based application, EventHive has no direct hardware interfaces. It relies on the standard hardware capabilities of the client device (e.g., GPS for location, camera for image uploads) as exposed and permissioned by the web browser.

4.3 Software Interfaces

- **[IF-01] Google Maps API:** The system shall use the Google Maps JavaScript API for map rendering and the Google Places API for location autocomplete in search and event creation forms. API keys must be secured and restricted.
- **[IF-02] Razorpay/Stripe API:** The system shall integrate with the payment gateway's server-side SDK. The integration must securely handle payment intent creation, processing callbacks, and refund webhooks.
- **[IF-03] OAuth 2.0:** The system shall use the OAuth 2.0 protocol to interface with Google and Facebook, requesting only the minimum required scopes (e.g., 'openid', 'email', 'profile').
- **[IF-04] OpenAI API:** The system shall send chat message content to the OpenAI moderation endpoint. It must correctly interpret the classification scores received in the API response to flag content.

4.4 Communications Interfaces

- **[CI-01] HTTPS:** All communication between the client browser and the backend server shall be encrypted and occur exclusively over HTTPS (TLS 1.2 or higher).
- **[CI-02] WebSockets:** Real-time chat communication shall be established via a secure WebSocket (WSS) connection, managed by the Socket.io library.
- **[CI-03] Email/SMS:** The system shall use the REST APIs of SendGrid (for email) and Twilio (for SMS) to send transactional messages. It must handle potential API failures and retries gracefully.

5 Nonfunctional Requirements

5.1 Performance Requirements

- **[NFR-101] Server Response Time:** Under normal load (1,000 concurrent users), the server-side processing time for 95% of all API requests shall be less than 500ms.
- **[NFR-102] Page Load Time:** The Largest Contentful Paint (LCP) for the main map page shall be under 2.5 seconds on a standard 4G mobile connection.
- **[NFR-103] Chat Latency:** Real-time chat messages shall be delivered to all participants in a room (up to 500 users) with an end-to-end latency of less than 250ms.
- **[NFR-104] Scalability:** The system infrastructure must support horizontal scaling. It shall be capable of handling peak loads of up to 10,000 concurrent users with no more than a 15% degradation in response time.

5.2 Reliability and Availability

- **[NFR-201] Availability:** The system shall have a minimum uptime of 99.9% per month, excluding scheduled maintenance.
- **[NFR-202] Fault Tolerance:** In case of a database or server node failure, the system should failover to a replica with minimal disruption to service. The Recovery Time Objective (RTO) is 15 minutes.
- **[NFR-203] Data Backup:** The PostgreSQL database shall be backed up daily, with backups retained for at least 30 days. Point-in-time recovery must be possible.

5.3 Security Requirements

- **[NFR-301] Data Encryption:** All data in transit shall be encrypted using TLS 1.3. All sensitive user data at rest (e.g., passwords, PII) in the database shall be encrypted. Passwords must be hashed using a strong, modern algorithm (e.g., Argon2).
- **[NFR-302] Authentication and Authorization:** Access to protected API endpoints shall be strictly controlled by a robust JWT-based authentication system. Tokens shall have a short expiration (15 minutes) and be managed with a secure refresh token mechanism. The system shall prevent Insecure Direct Object Reference (IDOR) vulnerabilities.
- **[NFR-303] Input Sanitization:** All user-supplied input shall be rigorously sanitized on the server side to prevent XSS, SQL Injection, and other common web vulnerabilities.
- **[NFR-304] Data Privacy:** The system shall adhere to the principles of India's Digital Personal Data Protection Act (DPDPA).

5.4 Software Quality Attributes

- **[NFR-401] Maintainability:** The codebase shall adhere to a strict style guide (enforced by linters like ESLint/Prettier) and be organized into logical, loosely-coupled modules. A comprehensive suite of unit and integration tests shall be maintained, with a target code coverage of 80%.
- **[NFR-402] Usability:** The UI shall be intuitive, adhering to established design patterns. A new user should be able to find and book a ticket for an event in under 90 seconds from landing on the homepage.

- **[NFR-403] Portability:** The entire application stack (frontend, backend, databases) shall be containerized using Docker and managed with Docker Compose, ensuring consistent environments and easy deployment.

6 Other Requirements

6.1 Data Definition and Database Requirements

- **[DB-01]:** A relational database schema shall be designed to ensure data integrity using primary keys, foreign keys, and constraints.
- **[DB-02]:** The schema must be normalized to at least Third Normal Form (3NF) to reduce data redundancy.
- **[DB-03]:** Database migrations shall be managed using a dedicated tool (e.g., TypeORM migrations, Flyway) to ensure schema changes are version-controlled and repeatable.

6.2 Installation and Acceptance Requirements

- **[IA-01]:** The development team shall provide a fully containerized version of the application using Docker for easy setup in development and staging environments.
- **[IA-02]:** Acceptance testing shall be performed in a staging environment that is a mirror of the production environment.
- **[IA-03]:** A set of acceptance criteria, derived from the functional requirements in this document, shall be defined and signed off by the stakeholders before deployment.

6.3 Legal and Regulatory Requirements

- **[LR-01]:** The platform must have a publicly accessible Privacy Policy and Terms of Service.
- **[LR-02]:** The platform must comply with the Digital Personal Data Protection Act (DPDPA) of India regarding the collection, storage, and processing of user data.
- **[LR-03]:** All financial transactions must comply with the regulations set forth by the Reserve Bank of India (RBI) for payment gateways.

A Appendix A: Glossary

API Application Programming Interface

AWS Amazon Web Services

DPDPA Digital Personal Data Protection Act

JWT JSON Web Token

LCP Largest Contentful Paint

NGO Non-Governmental Organization

OAuth Open Authorization

PII Personally Identifiable Information

RTO Recovery Time Objective

SPA Single Page Application

SRS Software Requirements Specification

TLS Transport Layer Security

UI User Interface

UPI Unified Payments Interface

UX User Experience

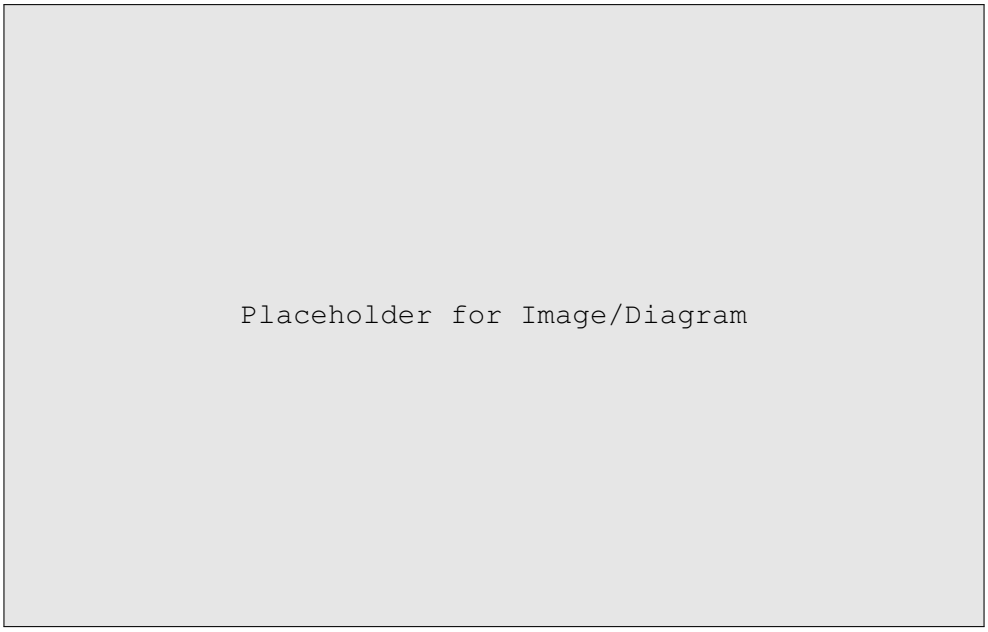
WSS Secure WebSocket

B Appendix B: Analysis Models (Conceptual)

This section provides a conceptual overview of the models that will be developed during the design phase.

B.1 Use Case Diagram

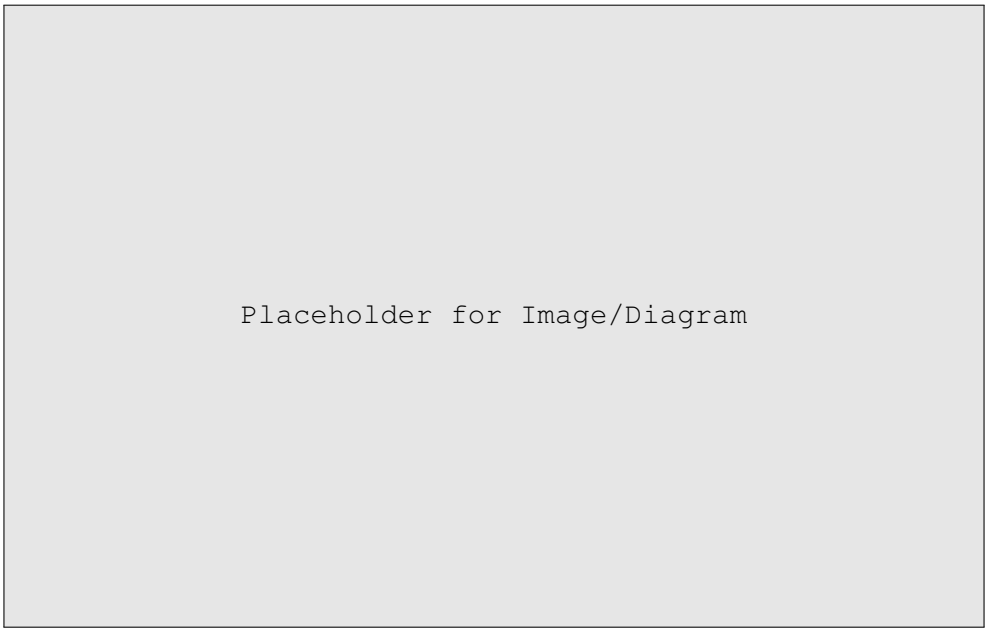
A key use case is "Attend Event." The actors involved are the General User and the Payment Gateway. The process includes: Find Event, View Event Details, Book Ticket, Process Payment, and Join Event Chat.



figureConceptual Use Case Diagram for a General User Attending an Event

B.2 Entity-Relationship Diagram (ERD)

The core entities of the system will be Users, Events, Tickets, Chatrooms, and Messages. Key relationships include: a User can organize many Events; an Event can have many Tickets; a User can have many Tickets; an Event has one Chatroom; a Chatroom has many Messages.



figureConceptual High-Level Entity-Relationship Diagram

C Appendix C: To Be Determined List

This section lists items that are out of scope for this version of the SRS but require future clarification.

- The specific monetary or non-monetary value of the coupon system for Verified Organizers.
- The precise algorithm and weighting for calculating community engagement scores for leaderboards.

- The definitive, exhaustive list of categories for event filtering.
- The final decision on whether to use a third-party AI service for moderation or develop a custom in-house model. This will depend on a cost-benefit analysis.
- The detailed workflow for private messaging between users (deferred to a future version).