Software Requirements Specification

for

ShoutMe

Prepared by Adeolu Adeogun & Justin Obasuyi

13/11/2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document describes the requirements of Shout-Me and its analysis. It serves as a reference to system design. Its intended audience is the project coordinator, project supervisor and system designers.

1.2 Document Conventions

This SRS follows a consistent set of standards and typographical conventions to ensure clarity and uniformity throughout the document:

1. *Formatting Conventions:*
   Bold text is used to emphasize key terms, requirement labels, and important system behaviours. Italics are used for optional notes or clarifications.

2. *Priority Levels:*
   Each requirement receives its own explicit priority level (High, Medium, or Low). Priorities are not inherited from higher-level features; they must be stated directly for every individual requirement.

3. *Abbreviations*
   Terms have been labelled as IA, SRE to help shorten the document length

1.3 Intended Audience and Reading Suggestions

IA-1: *Project Coordinator*
The project coordinator will use this SRS to verify that the system objectives, scope, and functional requirements match the project proposal and academic expectations. Their focus should be on Section 1 (Introduction) and Section 2 (Overall Description) to understand the system vision and design direction.

IA-2: *Project Supervisor*
The supervisor will review the SRS to assess completeness, correctness, and feasibility. They will focus on Sections 1, 2, 4, and 5 to ensure the system requirements are technically sound, clearly defined, and achievable within the project timeline.

IA-3: *System Designers*
System designers will rely on this SRS to understand how the system should behave, what components must interact, and what constraints must be followed. Their primary focus is Section 3 (Interfaces) and Section 4 (System Features), as these sections specify the architectural and functional requirements needed for design activities.

1.4 Product Scope

The software being specified is a web-based event booking and social networking platform designed to help people connect through shared interests and real-world activities. Its core purpose is to allow users to create, join, and manage a wide range of events, from small casual gatherings such as 5v5 football matches, study groups, and open-mic nights to larger activities like workshops, concerts, or community meetups.

The platform aims to go beyond simple event discovery by introducing a social interaction layer, enabling users to chat, connect, and build friendships with others who participate in similar events or share common interests. This transforms events from isolated experiences into opportunities for forming lasting social connections and fostering a sense of community and belonging.

Unlike mainstream platforms such as Ticketmaster or Eventbrite, which primarily target large-scale or commercial events, this platform emphasizes community-driven, smaller-scale gatherings. It provides users with the freedom to host their own events regardless of size, budget, or formality ensuring that anyone can organize or participate without restrictions. This makes the system accessible to everyday users, local groups, and grassroots organizers who often lack support from commercial event platforms.

By combining event management tools with integrated social networking features, the platform bridges the gap between digital interaction and real-world connection. It encourages people to explore nearby activities, meet others with similar passions, and stay engaged within their local communities. Overall, the platform aligns with modern business strategies centred on community engagement, personalized user recommendations, and scalable digital ecosystems. It represents a sustainable model for long-term user retention, growth, and community-focused value generation.

1.5 References

REF-1: IEEE SRS Standard (IEEE 830-1998)
Author: IEEE Computer Society
Description: Industry guidelines for structuring and writing Software Requirements Specifications.
Source: https://standards.ieee.org
Version: 1998

REF-2: React Official Documentation
Author: Meta Platforms, Inc.
Description: Front-end development framework documentation used to build the user interface.
Source: https://react.dev
Version: React 18 (Accessed 2025)

REF-3: Django Framework Documentation
Author: Django Software Foundation
Description: Backend framework documentation for server logic, authentication, and REST API development.
Source: https://docs.djangoproject.com
Version: Django 4.x (Accessed 2025)

REF-4: Google Maps JavaScript API Documentation
Author: Google LLC
Description: Mapping and geolocation API documentation used for event location features.
Source: https://developers.google.com/maps/documentation/javascript
Accessed: 2025

REF-5: PostgreSQL Documentation
Author: PostgreSQL Global Development Group
Description: SQL database reference for data modelling, querying, and schema design.
Source: https://www.postgresql.org/docs
Version: 15.x (Accessed 2025)

REF-6: Project Vision & Scope Notes (Team Internal Document)
Author: Project Team (Justin & Adeolu)
Description: Outlines the initial concept, goals, and intended user experience for the platform.
Location: Internal team document
Date: 2025

REF-7: IEEE SRS Template (Module-Provided)
Author: Academic Module Material
Description: Official template used as the structural basis for this SRS.
Source: Provided through course materials (uploaded file)
Date: 2025

2. Overall Description

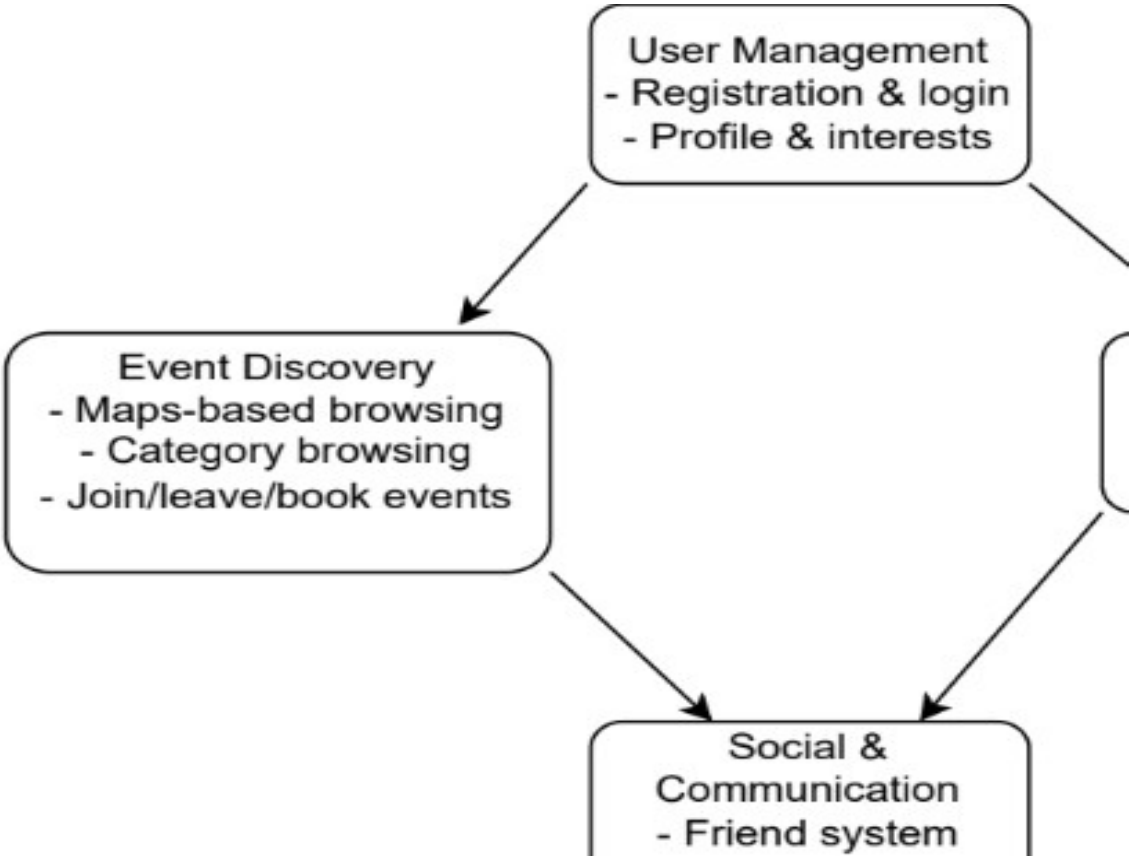
2.1 Product Perspective

*ShoutMe is a **new, standalone web-based platform** designed to combine event booking with social networking features. It is not a modification or extension of an existing system; instead, it is a self-contained solution built specifically to address the limitations of current large-scale event platforms and the lack of community-focused digital tools.*

2.2 Product Functions

High level functions:

- *User registration and login*
- *Create/manage event*
- *Join/leave/book events*
- *Maps based event browsing*
- *User profile and interests*
- *Friend system*
- *Chat and messaging*
- *Browse events by categories*
- *Recommended engine based on interests + location*



2.3 User Classes and Characteristics

User Class	***Description***	***Privileges***
General User	*Regular event participants*	*View/join events,*

User Class	*Description*	*Privileges*
		create events
Event Host	*Users that host or organise events*	*Full CRUD on their events*
Admin	*System managers*	*Remove content, manage users*

2.4 Operating Environment

It is going to be browser based running on either chrome, edge or safari, for the front end its going to be React 18 and for the backend it is going to be Django 14, database is going to be SQL and we will be hosting it from our personal laptops with GitLab deployment

2.5 Design and Implementation Constraints

Technology constraints:

Front-End Framework Requirement:
*The client interface must be implemented using **React**. No alternative front-end frameworks (Angular, Vue, Svelte, etc.) may be substituted.*

Back-End Framework Requirement:
*Server logic, authentication, and API endpoints must be built using **Django** (Python) as specified in the project requirements.*

Database Requirement:
*The system must use a **SQL-based relational database** (PostgreSQL or MySQL). No NoSQL databases (e.g., MongoDB, Firebase Realtime DB) are permitted.*

Mapping & Geolocation:
*The platform must integrate the **Google Maps JavaScript API** for displaying event locations and supporting map-based search.*

Hardware & Environment Constraints
The application must run on standard web browsers and typical consumer hardware, including laptops, tablets, and phones.
The development environment is limited to school-provided equipment (laptops/desktops) and may restrict local hosting options or advanced deployment tools.
The platform must not depend on specialized hardware, sensors, or high-performance GPUs.

Architectural Constraints
*The system must follow a **client–server REST architecture**.*
*All communication between the React front-end and Django back-end must use **JSON over HTTPS**.*

Real-time features (such as chat) must follow approved methods only (e.g., WebSockets or polling), depending on what is achievable within time and infrastructure constraints.

2.6 User Documentation

The following user documentation will be delivered alongside the platform to support end users, administrators, and developers. These documents are intended to assist with system setup, platform usage, troubleshooting, and ongoing maintenance.

2.6.1 End-User Documentation

User Guide / Help Pages:
A clear, beginner-friendly guide explaining how to create an account, join events, create events, navigate the map, update profiles, use chat features, and manage bookings.
FAQ Section:
Provides solutions to common issues (e.g., "Why can't I join this event?", "How do I change my password?", "Why is my location not loading?").
Onboarding Tips:
Quick instructional messages or tooltips that appear when new users interact with key features for the first time.

2.6.2 Administrator Documentation

Admin Usage Manual:
Guides administrators on how to manage users, remove inappropriate events, perform moderation, and monitor system activity.
System Maintenance Instructions:
Covers regular backups, clearing logs, updating dependencies, restarting the server, and database management tasks.

2.6.3 Developer Documentation

README File:
Included in the repository, containing instructions for installation, project structure, API endpoints, environment variables, and development standards.
API Reference:
Documentation describing all REST endpoints (request formats, response formats, authentication details, error codes).
UI Component Notes:
A brief guide outlining front-end components, reusable modules, and how to update the interface consistently.

2.6.4 Delivery Format

Documentation will be delivered in the following formats:

- *PDF files for formal guides*
- *Markdown files inside the GitHub repository*
- *Inline comments and code-level documentation*

Additional UI-specific design documents (wireframes, component layouts) will be maintained separately as part of a dedicated UI specification if required.

2.7 Assumptions and Dependencies

The development and operation of the platform rely on several key assumptions and external dependencies. Any changes to these factors may impact system functionality, performance, or project scope.

2.7.1 Assumptions

- ***User Device Capability:***
  Users are assumed to have access to modern browsers capable of running JavaScript, including Chrome, Firefox, Safari, or Edge.
- ***Stable Internet Access:***
  Continuous internet connectivity is assumed for both users and the hosting environment.
  Offline operation is not supported.
- ***Location Permissions:***
  Users are assumed to grant browser permission to access their approximate location for map-based and distance-based event recommendations.
- ***Realistic Event Volume:***
  It is assumed that initial user activity will involve small-to-medium event volumes and not enterprise-level traffic.
- ***User Behaviour:***
  Users are expected to behave responsibly, providing truthful event information and adhering to community guidelines.

2.7.2 External Dependencies

- ***Google Maps API:***
  The system relies on Google Maps for geolocation, marker rendering, and map navigation. Any changes to API pricing, availability, or quota may affect the platform.
- ***Third-Party Libraries:***
  Stability of external packages (React libraries, npm packages, Django modules) is required for system functionality.
- ***Database Engine:***
  The platform depends on PostgreSQL/MySQL for all persistent storage. Any database downtime or corruption will affect event, user, and booking data.
- ***Hosting Environment:***
  The hosting platform must support Python, Django, SQL, SSL (HTTPS), and static file serving.
- ***Email/Notification Services (if used):***
  If registration or password reset involves email notifications, the system depends on a functioning SMTP service or external email provider.

2.7.3 Project Dependencies

- *Successful collaboration between front-end and back-end developers is required for consistent API integration.*
- *Timely delivery of required assets (UI components, endpoints, database schema) is critical to maintain project timeline.*

- *Changes in academic or project requirements may affect scope or prioritisation of features.*

3. External Interface Requirements

3.1 User Interfaces

*The Shout me platform is delivered as a **responsive web application** accessed through a modern browser. The user interface is implemented in **React** and follows a clean, minimal design with consistent components and navigation patterns across all screens.*

- ***Authentication (Login / Registration)***
  Simple forms for email and password input, with validation messages displayed inline beneath invalid fields. Standard buttons: Login, Register, and links to switch between the two pages.

- ***Home / Dashboard***
  A personalized landing page showing upcoming events, recommended events, and shortcuts (e.g., "Create Event", "Find Events Near Me"). The layout typically includes a header with the page title, a main content area with event cards, and optional filter controls.

- ***Event Listing & Search***
  A list or grid of event "cards" showing title, date/time, location, category, and a short description. A search bar and filters (e.g., category, date, distance) appear at the top or in a sidebar. Each card includes a View Details button.

- ***Event Details & Booking***
  A detail page that displays full event information (host, description, date/time, location with map, capacity, attendees). Main actions: Join/Book Event, Leave Event, and Message Host (if available). For event hosts, additional buttons for Edit and Delete are shown.

- ***Event Creation / Editing***
  A structured form with labelled fields (title, description, category, date, time, location, etc.). The system enforces mandatory fields and valid formats, with error messages displayed near invalid inputs. Standard buttons: Save, Cancel.

- ***Map View (Events on Map)***
  A full-screen map using the Google Maps API, with event markers indicating location. Clicking a marker opens a popup with summary info and a link to the event details page. Map controls include zoom, pan, and optional filters (category, radius).

- ***User Profile & Settings***
  A profile page with user information (name, picture, bio, interests, upcoming/past events, friends). An Edit Profile interface allows updating these fields. Settings (e.g., notification preferences, privacy options) may appear as separate sections or tabs.

- ***Messaging / Chat***
  A messaging interface consisting of a conversation list and a message pane. Users can

view past messages and send new ones via a text input box with a Send button. Optional features include unread indicators and timestamps.
Across all screens, the following GUI standards apply:

- *Global Navigation Bar (visible for logged-in users): app logo/home link, main navigation links (Home, Events, Map, Create Event, Messages, Profile), and a profile menu including Logout.*

- ***Button Conventions:***
  Primary actions (e.g., "Join", "Save", "Create Event") use a distinct primary button style. Destructive actions (e.g., "Delete Event") use a warning style and trigger a confirmation dialog.

- ***Error and Status Messages:***
  Form errors are shown inline beneath the relevant field. Critical system errors or success messages (e.g., "Event created successfully") may appear as banners or toast notifications at the top/bottom of the screen.

- *Keyboard and Accessibility:*
  Forms can be submitted with the Enter key when appropriate.Modal dialogs can be dismissed with the Esc key. All interactive components are keyboard navigable (Tab order), and text labels are associated with form controls.

- *Layout Constraints:*
  The UI is responsive and must adapt to desktop and mobile screen sizes. On smaller screens, the main navigation may collapse into a hamburger menu, and sidebars become stacked sections.

3.2 Hardware Interfaces

The web application will also be accessible on Mobile Devices using iOS and Android smartphones/tablets through mobile browsers (Safari, Chrome)

3.3 Software Interfaces

Client–Server Interface
Client Software:
Web browser (Chrome, Edge, Firefox, Safari, etc.)
Front-end implemented in React

Server Software:
Backend implemented in Django (v4.x)
Exposes a RESTful JSON API

Communication:

Protocol: HTTPS
Format: JSON for all requests/responses
Authentication: JWT (JSON Web Token) sent in Authorization: Bearer <token> header for protected endpoints

Backend–Database Interface
Database Software:
Relational SQL database (e.g., PostgreSQL 15.x or MySQL 8.x)

Google Maps JavaScript API
Purpose:
Display maps and event markers on the front-end.
Optional geocoding of addresses to coordinates.
Interface:
JavaScript API loaded in the browser via script tag.
The front-end sends requests to Google's servers (not directly from the backend).

3.4 Communications Interfaces

*The Shout-Me platform is a web-based system and relies entirely on standard internet communication protocols for all interactions between **users, the application server, and third-party services.***

Message Format
All API requests and responses use JSON as the standard data format.

*Request: **HTTP method (GET/POST/PUT/DELETE)**, URL, headers (e.g. Authorization, Content-Type: application/json), and optional JSON body.*

*Response: **HTTP status code**, headers, JSON body containing requested data or structured error information.*

Security and Encryption
*All traffic must be encrypted using **TLS**. Authentication uses tokens sent in the Authorization header. Sensitive data (e.g. passwords) is only sent in **encrypted request bodies**, never in URLs. Server-side validation and sanitization are required for all incoming data.*

Data Transfers and Synchronization

*No hard real-time guarantees, however, endpoints should be designed to minimize payload size (e.g. pagination for event listings). File uploads (e.g. profile or event images) are limited to reasonable sizes to avoid excessive bandwidth. The Primary interaction model is stateless request–response: the client will fetch updated data via API calls as needed. If real-time chat is used: **Prefer WebSockets (wss://)** for low-latency, bidirectional communication, with JSON messages defining sender, receiver/conversation, content, and timestamp. As a fallback, **HTTP polling or long-polling** may be used. Consistency is managed server-side via **database transactions;** clients refresh their view by re-requesting data.*

Electronic Forms
All user input (registration, login, event creation, profile editing, messaging) is submitted as JSON in the body of HTTPS POST/PUT requests.

4. System Features

4.1 User Registration & Authentication

4.1.3 Functional Requirements

REQ-1.1: System must allow users to register using email and password.
REQ-1.2: System must hash and store passwords securely.
REQ-1.3: System must authenticate users using JWT or Django authentication.
REQ-1.4: System must prevent login attempts with invalid credentials.
REQ-1.5: System must provide logout functionality that invalidates the active session.
REQ-1.6: System must validate email format and password strength.

4.2 Event Discovery

4.2.1 Description and Priority

Allows users to find events based on interests, categories, distance, and time.
Priority: *High.*

4.2.2 Stimulus/Response Sequences

1. User opens Events page.
2. System retrieves events list based on default filters.
3. User applies filters/search.
4. System returns updated results.

4.2.3 Functional requirements

REQ-2.1: The system shall allow users to browse a list of all public events.
REQ-2.2: The system shall support text-based search.
REQ-2.3: The system shall allow filtering by date, time, location radius, and category.
REQ-2.4: The system shall return results sorted by relevance or date.
REQ-2.5: The system shall paginate or limit large search results.

4.3 Event Details Page

4.2.1 Description and Priority

Displays full information about a specific event.
Priority: High.

4.3.2 Stimulus/Response

1. User selects event.
2. System loads event details, location, host information, and current attendees.
3. UI displays action buttons (Join/Edit/Delete depending on user).

4.3.3 Functional Requirements

REQ-3.1: The system shall display all event fields (title, date, time, location, description, category).
REQ-3.2: The system shall show host info and attendee count.
REQ-3.3: The system shall display a map pinpointing the event location.
REQ-3.4: The system shall show available actions based on user role (host vs participant).

4.4 Event Creation & Management

4.4.1 Description and Priority

Users can create and manage their own events.
***Priority:** High*

4.4.2 Stimulus/Response

1. User clicks "Create Event".
2. System displays event creation form.
3. User submits form.
4. System validates and saves event.
5. Event becomes visible on platform.

4.4.3 Functional Requirements

REQ-4.1: The system shall allow users to create events with required fields.
REQ-4.2: The system shall validate that required fields are completed and dates are valid.

REQ-4.3: The system shall allow event hosts to edit their events.
REQ-4.4: The system shall prevent non-hosts from editing events.
REQ-4.5: The system shall allow event hosts to delete their events.
REQ-4.6: The system shall prompt a confirmation before deleting an event.

4.5 Event Booking (Join / Leave Events)

4.5.1 Description and Priority

Users can join or leave events; the system manages event capacity.
Priority: High.

4.5.2 Stimulus/Response

1. User clicks "Join Event".
2. System checks availability and user eligibility.
3. System confirms booking or returns error.

4.5.3 Functional Requirements

REQ-5.1: The system shall allow users to join public events.
REQ-5.2: The system shall prevent users from joining the same event twice.
REQ-5.3: The system shall prevent hosts from joining their own events.
REQ-5.4: The system shall decrease available capacity when a user joins.
REQ-5.5: The system shall allow users to leave events.
REQ-5.6: The system shall send confirmation messages for booking actions.

4.6 Map-Based Event Discovery

4.6.1 Description and Priority

Users can find events on an interactive map using Google Maps.
Priority: Medium-High.

4.6.2 Stimulus/Response

1. User opens Map tab.
2. System loads events within visible map region.
3. User clicks a marker.
4. System shows event preview popup.

4.6.3 Functional Requirements

REQ-6.1: The system shall display events as markers on a map.
REQ-6.2: The system shall update event markers when the map is moved or zoomed.
REQ-6.3: The system shall display event details in a marker popup.

REQ-6.4: The system shall allow filtering through map interface.
REQ-6.5: The system may request user location (with permission).

4.7 User Profile Management

4.7.1 *Description and Priority*

Users maintain personal information, interests, and profile image.
Priority: Medium-High.

4.7.2 Stimulus/Response

1.User opens Profile page.
2.System loads user's info.
3.User edits fields and saves changes.
4.System validates and updates profile.

4.7.3 Functional Requirements

REQ-7.1: The system shall allow users to view and edit their profile.
REQ-7.2: The system shall allow users to upload or change a profile image.
REQ-7.3: The system shall allow users to add or edit interests.
REQ-7.4: The system shall show user's upcoming and past events.
REQ-7.5: The system shall restrict profile editing to the profile owner.

4.8 Social Features (Friends, Messaging)

4.8.1 *Description and Priority*

Supports social interaction to build community.
Priority: Medium.

4.8.2 Stimulus/Response

1.User sends a friend request.
2.System notifies the recipient.
3.Recipient accepts or rejects.
4.Messaging becomes available if accepted.

4.8.3 Functional Requirements

REQ-8.1: The system shall allow users to send friend requests.
REQ-8.2: Recipients shall be able to accept or reject requests.
REQ-8.3: The system shall allow users to view their friend list.
REQ-8.4: The system shall allow messaging between friends.
REQ-8.5: The system shall support a conversation view with timestamps.
REQ-8.6: The system shall not allow messaging between users who are not connected (unless messaging is event-host only).

4.9 Admin Management & Moderation

4.9.1 Description and Priority

Admins manage platform safety and remove harmful content.
Priority: Medium-Low (depends on project scope).

4.9.2 Stimulus/Response

1.Admin logs into admin panel.
2.Admin views list of users/events.
3.Admin selects item to delete or flag.
4.System updates records.

4.9.3 Functional Requirements

REQ-9.1: The system shall allow admins to remove inappropriate events.
REQ-9.2: The system shall allow admins to deactivate user accounts.
REQ-9.3: The system shall maintain logs of admin actions.
REQ-9.4: The system shall restrict admin panel to admin users only.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

General Responsiveness
PR-1: *For typical user interactions (page navigation, viewing events, opening profiles), the page load time should be ≤* ***2 seconds*** *on a standard broadband connection.*
Rationale: Ensures the application feels responsive and reduces user abandonment.

PR-2: API responses *for common operations (fetching event lists, event details, profile data) should have a server processing time of ≤* ***500 ms*** *under normal load, excluding network latency.*
Rationale: Keeps UI updates smoothly and supports responsive front-end behaviour.

Event Search and Listing
PR-3: *When a user searches for events or applies filters (by category, date, or distance), the system should return the first page of results in ≤* ***1 second*** *of server processing time.*
Rationale: Search and browsing are core actions; delays here directly impact usability.

PR-4: *Event listings should support pagination or incremental loading to limit each response to a manageable number of records (e.g. 20–50 events per page).*
Rationale: Prevents excessively large payloads and reduces both server and client processing time.

Map and Location Features
PR-5: *The* ***map view*** *(with event markers) should become interactive within ≤* ***3 seconds*** *of the user opening the map page, assuming a stable internet connection.*
Rationale: Maps are visually heavier; this sets a realistic but user-friendly expectation.

PR-6: *Adding or updating markers (e.g. after applying filters) should complete in ≤* ***1 second*** *from the time the filter is applied.*
Rationale: Supports fluid exploration of events on the map.

Event Creation, Booking, and Updates
PR-7: *Submitting an event creation or edit form should complete in ≤* ***1 second,*** *and confirmation should be shown immediately after.*
Rationale: Creators must see quick feedback to avoid duplicate submissions.

PR-8: *Booking or joining an event must complete in ≤ 1* ***second*** *of server processing time, and the updated attendance state (e.g. booked/confirmed) must be visible to the user within the same interaction.*
Rationale: Booking is a critical path; slow feedback will cause confusion or double-clicking.

Data Storage and Capacity
PR-9: *The database must support at least:*
10,000 user records
5,000 event records
50,000 booking records
without requiring architectural changes.
Rationale: Ensures the design scales beyond trivial sample data.

PR-10*: Indexes must be used on frequently* ***queried fields*** *(e.g. event date, location, category, user ID) to keep query times within the performance limits defined above.*
Rationale: Prevents performance collapse as data volume increases.

Real-Time / Messaging (If Implemented)
PR-11: *For real-time messaging using* ***WebSockets****, messages should appear in the recipient's chat view within ≤* ***1 second*** *of being sent, under normal network conditions.*
Rationale: Provides a near real-time chat experience suitable for coordinating events.

If any of these performance targets cannot be met due to environment constraints (e.g. very low bandwidth), the system must still behave correctly and display appropriate feedback to users(e.g. loading indicators, retry messages).

5.2 Safety Requirements

SR-1: *Prevention of Accidental Data Loss*
The system must prevent irreversible actions without explicit confirmation. This is done to prevent accidental loss of data due to misclicks or unintended user actions. Examples include: ***Deleting an event, Removing a user account, Cancelling a booking***

Each of these actions must trigger a confirmation dialog:
"Are you sure you want to delete this event? This action cannot be undone."

SR-2: *Protection Against Data Corruption*
All database write operations (event creation, booking, profile updates) must be performed within ***atomic transactions*** *to prevent partial or inconsistent data states.*
In case of system failure during an operation, the database must roll back to a safe, consistent state.
Ensures system integrity and prevents corrupted or invalid records.

.
SR-3: *Safe Event Management*
The system must clearly show ***event location, date, and host information*** *to prevent users from attending events under false assumptions. event hosts must agree to platform policies before publishing events.*

For community safety, events must not be allowed to: Use hateful, dangerous, or illegal descriptions, Promote harmful activities, Admins must have the ability to remove events that violate safety or community guidelines.
This is all done to reduce risk of users attending unsafe or inappropriate events.

5.3 Security Requirements

SRE-1: *Compliance with Data Protection Policies*
The platform must comply with general data safety standards, including GDPR principles for projects operating in the EU environment.

Users must be informed about: ***How their data is stored, how to request deletion of their data, what information is visible to others.***

SRE-2: *Authentication and Access Control Safety*
The system must prevent unauthorized access to: User accounts, Event creation tools, administrative functions, Login attempts must be rate-limited to prevent ***brute-force attacks,*** *Passwords must never be stored or transmitted in* ***plain text****. This is done to protect user identity and prevent harmful account takeovers.*

SRE-3: *Safety Certifications*
As a student project, Shout-Me does not require formal safety certifications, but it must adhere to: ***General web safety and security guidelines Browser and API usage policies such as Google Maps terms of use GDPR-compliant data handling practices*** *to ensures the platform is safe for general use, even without commercial certifications.*

SEC-4: *User Authentication*
Users must authenticate using a valid email and password. Passwords must be stored using

***secure hashing**. The system must use **JWT tokens** for protected API access to ensure only authorised users can interact with private data or features.*

***SEC-5:** Access Control Enforcement*
Users may only access or modify their own data. Event creators can edit or delete only their own events. Administrative functions must be restricted to authorised admin accounts. All sensitive endpoints must include server-side role and permission checks.

***SRE-6:** Data Encryption in Transit*
*All communication between client and server must use **HTTPS/TLS**. No sensitive data may be transmitted over **unencrypted HTTP**. WebSocket communication (if used for messaging) must use WSS to prevent **interception and eavesdropping.***

***SEC-7:** Personal Data Protection*
*Personal information including **account details, event attendance, messages, and profile data** must not be visible to other users unless explicitly allowed. Users must be able to request deletion of their account and associated data to uphold privacy rights.*

5.4 Software Quality Attributes

SQA-1: Usability
*The platform must provide a clean, **intuitive interface** that allows new users to register, browse events, and join events without requiring a tutorial. Core actions (login, event browsing, booking) must require no more than 3 user interactions each to complete. Ease of use is prioritized over advanced customization.*

SQA-2: Reliability
*The system must operate correctly during normal use with minimal failures. Under typical usage (≤ 50 concurrent users), the application must maintain ≥ **99% uptime** during testing periods. Any server error must return a clear, non-technical message without compromising data integrity.*

SQA-3: Availability
The system must be accessible at all times during project demonstrations and testing sessions. Scheduled maintenance tasks must not exceed 5 minutes and must not cause data loss.

SQA-4: Maintainability
The source code must follow modular and component-based design principles. Developers should be able to modify or extend individual features (e.g., event creation, chat, profiles) without affecting unrelated components. All code should be commented and structured for ease of updates.

SQA-5: Adaptability
The platform should support adding new event categories, new profile fields, or new social features without major architectural changes. The front-end must be adaptable to both desktop and mobile layouts using responsive design.

SQA-6: Portability
The application must run on any modern browser (Chrome, Firefox, Edge, Safari) without requiring installation. For deployment, it must run on standard Linux or cloud environments using Python, Django, React, and SQL, without hardware-specific dependencies.

SQA-7: Performance Efficiency
All core pages must load within 2 seconds on a standard broadband connection. The system must handle up to 50 concurrent users without noticeable degradation in responsiveness.

SQA-8: Interoperability
*The system must work seamlessly with integrated APIs such as **Google Maps**. Data exchanged between components must be in consistent JSON format to allow smooth interaction across front-end, back-end, and external services.*

SQA-9: Testability
All major features (authentication, event creation, booking, profile updates) must be testable through automated or manual test cases. API endpoints must return predictable and structured JSON responses to support automated testing.

SQA-10: Robustness
The system must handle invalid input, network interruptions, and unexpected user behaviours without crashing. In all such cases, it must display safe error messages and maintain data consistency.

SQA-11: Reusability
Common components (e.g., event card, profile card, input fields, buttons) must be implemented as reusable React components to reduce duplicated code and simplify future development.

5.5 Business Rules

BR-1: Event Ownership
Only the user who created an event (the event host) may edit or delete that event. Other users cannot modify host-owned events under any circumstances.

BR-2: Booking Permissions
Users may book or join any public event unless the event has reached its maximum capacity (if capacity is set). Hosts cannot book their own events.

BR-3: Role-Based Restrictions
System administrators have elevated privileges such as removing inappropriate events or banning users. Regular users cannot access administrator functions.

BR-4: Profile Visibility
A user's basic profile information (name, picture, interests) may be visible to other users, but private details (email, password, account settings) must remain hidden at all times.

BR-5: Event Content Guidelines
Events must comply with platform rules and may not contain hateful, illegal, or harmful content. Hosts are responsible for ensuring their events meet these guidelines.

BR-6: Messaging Safety Rules
Users may only message people they are connected with (e.g., friends or event hosts). Spam, harassment, or abusive behavior is prohibited and may lead to account suspension.

BR-7: Age Restrictions
Users must be at least 16 years old to create an account and participate on the platform. Events intended for adults must be clearly labelled.

BR-8: Data Accuracy
Users are responsible for providing accurate information in their profiles and events. Misleading or false data violates platform policies.

BR-9: Cancellation & No-Show Rules
If a user cancels a booking or fails to attend, hosts may remove them from future attendance lists or mark them as inactive (optional future feature).

BR-10: External API Compliance
Any use of third-party APIs (e.g., Google Maps) must follow the provider's terms of service. Violations may result in restricted access or application errors.

6. Other Requirements

This section outlines additional requirements that do not fall under previous categories but are necessary for the correct operation, maintainability, and compliance of the ShoutMe platform.

OR-1: Database Capacity
The system's database must support:
A minimum of 10,000 registered users
At least 5,000 events
At least 50,000 booking records
This ensures the platform can scale beyond small test datasets and operate effectively during peak usage periods.

Appendix A: Glossary

IA *– Intended Audience*

REF *– References*

PR *– Performance Requirements*

SRE *– Safety Requirements*

SEC *– Security Requirements*

SQA *– Software Quality Attributes*

BR *– Business Rules*

.

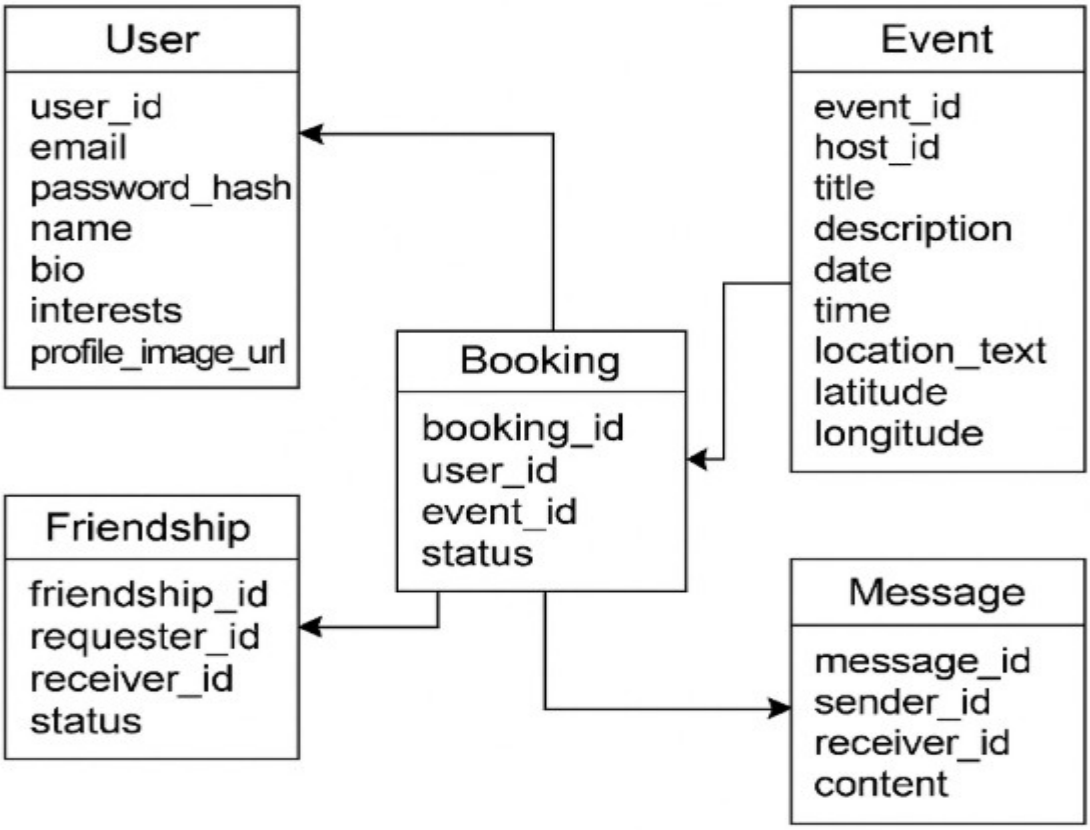
UI *– User Interface*

API *– Application Programming Interface*

JWT *– JSON Web Token*

DB *– Database*

Appendix B: Analysis Models



Appendix C: To Be Determined List

The rest of the other requirements are yet to be decided