Bilkent University
Department of Computer Engineering

CS491 Senior Design Project

T2435
Eventure

Detailed Design Report

Damla İmre	22001640	damla.imre@ug.bilkent.edu.tr
Gizem Gökçe Işık	21803541	gokce.isik@ug.bilkent.edu.tr
Alper Bozkurt	21802766	alper.bozkurt@ug.bilkent.edu.tr
Ercan Bahri Nazlıoğlu	21903151	bahri.nazlioglu@ug.bilkent.edu.tr
Onur Tanınmış	22003312	onur.taninmis@ug.bilkent.edu.tr

Supervisor: Cevdet Aykanat
Innovation Expert: Muhammed Naci Dalkıran
Course Instructors: Atakan Erdem, Mert Bıçakçı

Table Of Contents

1. Introduction

1.1 Purpose of the System

This system aims to create an innovative event discovery and social engagement platform that allows users to search, register, and share events effortlessly. The application leverages AI-driven recommendations, social networking features, and real-time mapping to enhance the user experience and encourage meaningful event participation.

This system aims to:

- Provide personalized event recommendations based on user interests and location.
- Allow users to create, share, and manage their events.
- Facilitate real-time event mapping where users can discover live events around them.
- Enable social interactions by letting users invite friends, see attendee lists, and join group discussions.
- Ensure user safety by integrating an emergency safety button that alerts authorities if needed.

This application simplifies event discovery and fosters a more connected, interactive community by bridging the gap between event organizers and attendees.

1.2 Design Goals

1.2.1 Usability

- The application must provide an intuitive and user-friendly interface, ensuring smooth navigation for both new and experienced users.
- Users should be able to easily register, search, and participate in events with minimal effort.

1.2.2 Performance

- The system should efficiently handle high traffic, ensuring quick event searches, AI-based recommendations, and real-time updates without delays.
- The backend should be optimized to provide fast API responses and low-latency interactions.
- AI-powered recommendations should be computed efficiently, balancing accuracy and speed.

1.2.3 Reliability

- The system should maintain 99.9% uptime to ensure constant availability.
- Event details and user data should be accurate and up to date.

1.2.4 Scalability

- The system should be able to handle increasing numbers of users, events, and real-time interactions without performance degradation.
- Load balancing techniques should be implemented to distribute traffic efficiently.

1.2.5 Maintainability

- The codebase should follow clean architecture principles, clearly separating backend (Spring Boot) and frontend (React).
- Well-documented code and API endpoints should ensure that future developers can maintain and improve the system with minimal effort.

1.2.6 Flexibility

- The system should allow users to customize preferences, including event filters, notification settings, and privacy options.
- The AI recommendation engine should adapt to changing user behaviors and preferences.
- The UI should support different themes and accessibility options.

1.2.7 Aesthetics

- The application should feature a visually appealing design with a modern and sleek UI.
- Colors, fonts, and layouts should be chosen carefully to maintain a professional and engaging look.
- Animations and transitions should enhance the user experience without causing distractions.

1.2.8 Supportability

- The system should be easily upgradable and allow for the addition of new features.
- Technical support and bug reporting tools should be integrated.
- Third-party service integrations (e.g., maps) should be manageable.

1.3 Definitions, acronyms, and abbreviations

1.3.1 Definitions

- **Eventure** – The proposed mobile application that enables users to discover, join, and create events while integrating AI recommendations, social networking, and safety features.
- **User** – An individual who registers on the platform, browses events, interacts with other users, and participates in event activities.

- **Event** – A scheduled gathering, activity, or occasion for users to create, discover, and register.
- **AI Recommendation System** – An artificial intelligence-based system that provides users personalized event suggestions based on their past preferences, interests, and location.
- **Event Registration** – The process through which a user expresses intent to attend an event within the application.
- **Safety Button** – A security feature that allows users to alert emergency contacts or law enforcement when they feel unsafe.
- **Event Map** – A live, interactive map displaying nearby events, user locations, and friends attending events.
- **Profile Page** – A dedicated section where users can view and manage their personal information, event history, and preferences.
- **Backend** – The server-side logic of the application responsible for handling authentication, event management, AI recommendations, and data storage.
- **Frontend** – The client-side user interface developed in React, where users interact with the system.
- **Database** – A structured data storage system implemented using PostgreSQL to manage user profiles, events, and associated data.
- **API** – A set of protocols that allow communication between the frontend, backend, and external services.

1.3.2 Acronyms and Abbreviations

- **AI** – Artificial Intelligence
- **API** – Application Programming Interface
- **CRUD** – Create, Read, Update, Delete (basic database operations)
- **DB** – Database
- **GPS** – Global Positioning System
- **RBAC** – Role-Based Access Control
- **UI** – User Interface
- **UX** – User Experience
- **JWT** – JSON Web Token (used for authentication)
- **REST** – Representational State Transfer (API architectural style)
- **MVC** – Model-View-Controller (software design pattern)
- **HTTP** – HyperText Transfer Protocol
- **HTTPS** – HyperText Transfer Protocol Secure
- **SQL** – Structured Query Language
- **JSON** – JavaScript Object Notation (data format used for API communication)
- **OAuth** – Open Authorization (authentication framework)
- **2FA** – Two-Factor Authentication

1.4 Overview

This document outlines the design and development of *Eventure*, a mobile application that enhances event discovery, social interaction, and user safety. The application uses AI-driven

recommendations to provide users personalized event suggestions based on their preferences, past interactions, and current location. Users can search, join, and create events, fostering a dynamic community where participation is seamless and engaging.

The system integrates real-time event mapping, allowing users to visualize nearby events and track their friends' attendance. Additionally, *Eventure* prioritizes user safety by incorporating a Safety Button, which enables users to notify emergency contacts or law enforcement in distressing situations.

The backend is implemented using Spring Boot and PostgreSQL, ensuring a scalable, secure, high-performance architecture. The frontend is developed using React, providing an intuitive and responsive user experience. The application follows a modular and flexible design, supporting future enhancements such as ticketing, advanced analytics, and third-party integrations.

2. Current software architecture

The event discovery market features platforms like Eventbrite, Meetup, Swoogo, and River, each with unique strengths and limitations. Eventbrite excels in ticketing and event management but lacks robust social features for attendees to interact before events. Meetup focuses on community building and connecting people with shared interests but is less suited for larger-scale or business-oriented events. Swoogo caters to event organizers with advanced analytics and marketing tools but is often too complex and costly for smaller-scale or casual events [2]. River emphasizes personalized and community-driven experiences but falls short in providing extensive customization or professional event management capabilities [3].

While these platforms address specific event discovery and management aspects, they often fail to integrate personalized recommendations, real-time social interaction, and safety features into a single solution.

Eventure bridges the gaps in the current market by combining personalized event recommendations, enhanced social features, and user safety tools. The app uses AI to analyze users' past preferences, feedback, and location, offering tailored suggestions that resonate with individual interests. Unlike competitors, *Eventure* integrates an interactive map that displays event locations, real-time user avatars, and hot spots highlighting popular events and friend participation, making event exploration more engaging.

3. Proposed software architecture

3.1 Overview

The proposed system, *Eventure*, is a comprehensive mobile application designed to transform how users discover and interact with events by combining advanced technologies

with social and safety features. The system utilizes AI-powered recommendations to provide personalized event suggestions based on users' interests, past feedback, and current location, ensuring relevance and engagement.

Eventure focuses on social integration by enabling users to connect with like-minded individuals, invite friends, and participate in event group chats, fostering community and collaboration. The interactive event map enhances the event exploration process, offering current visualizations of nearby events, popular hotspots, and friends' live locations.

A distinguishing aspect of Eventure is its strong emphasis on user safety. The system includes a discreet Safety Button that users can activate to notify emergency contacts or local law enforcement in critical situations, ensuring peace of mind during event participation.

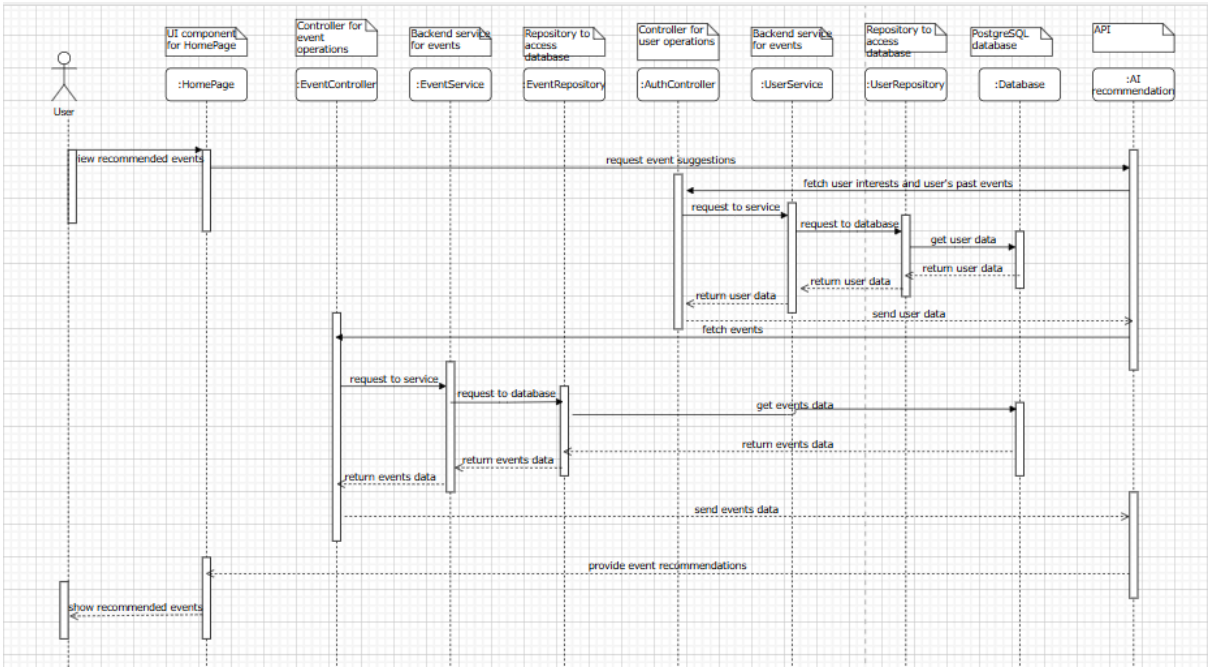
The system further allows users to provide structured feedback after events, contributing to improved recommendations and event quality over time. Users can also create and manage their events, enabling a seamless experience for attendees and organizers.

By integrating event discovery, social interaction, and safety tools into a single platform, Eventure addresses gaps in existing systems and offers a user-friendly, scalable, and secure solution that meets the needs of modern event-goers.

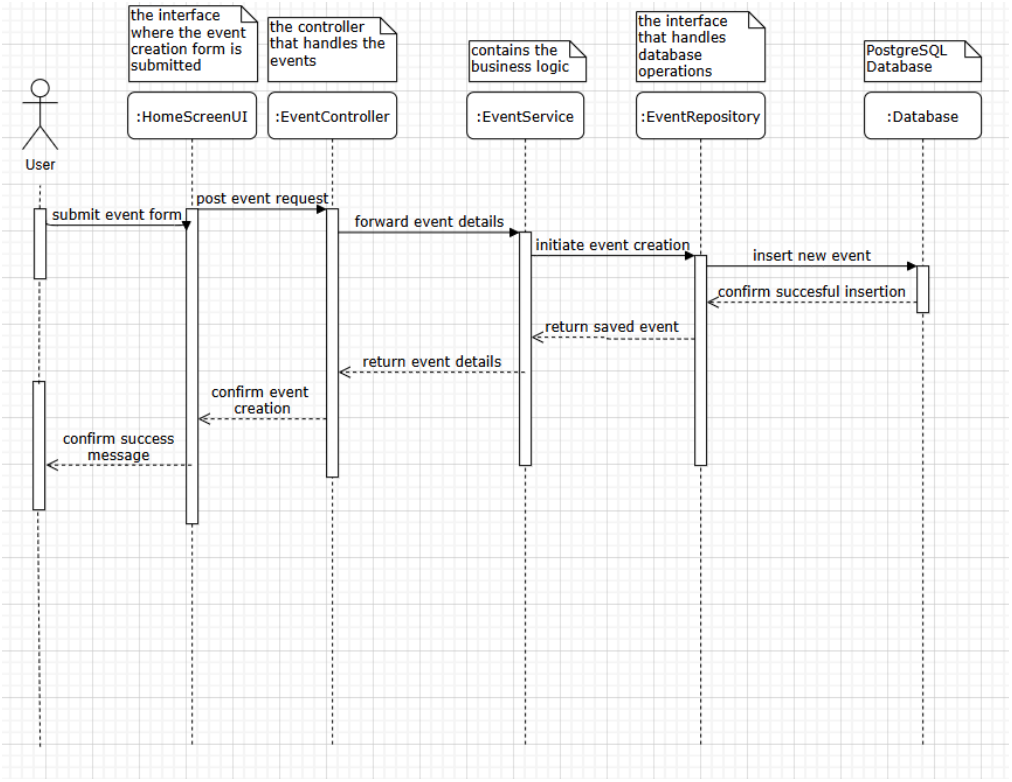
3.2 System Models

3.2.1 Dynamic Models

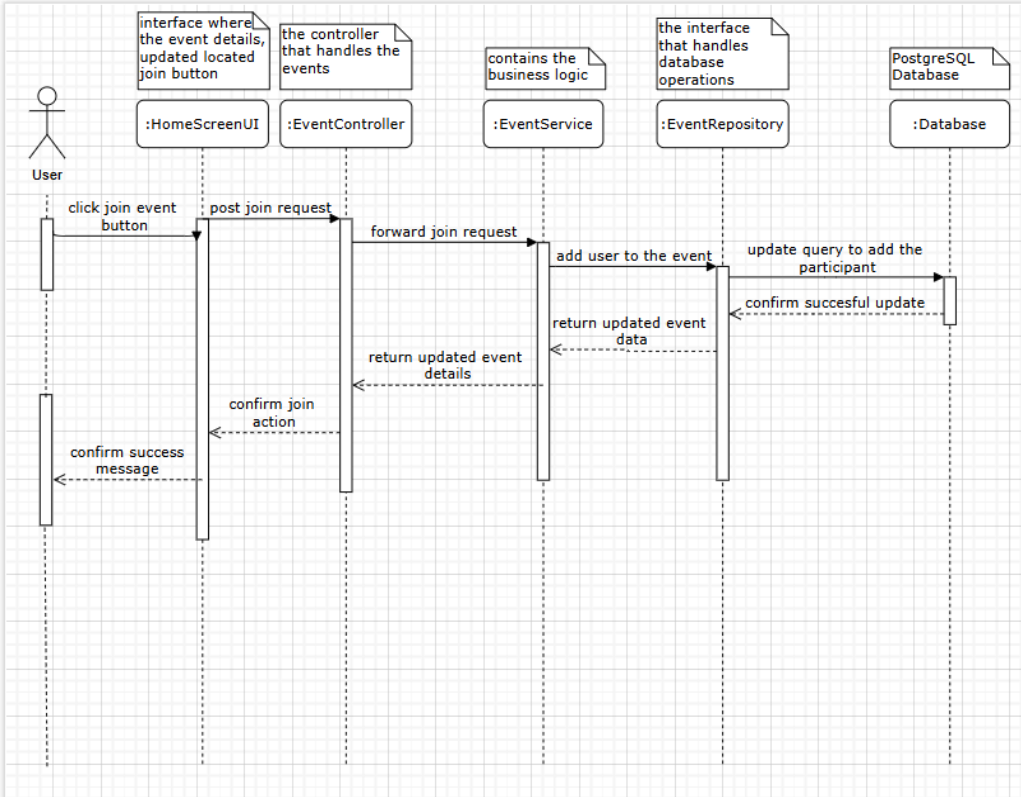
3.2.1.1 View Recommended Events Sequence Diagram



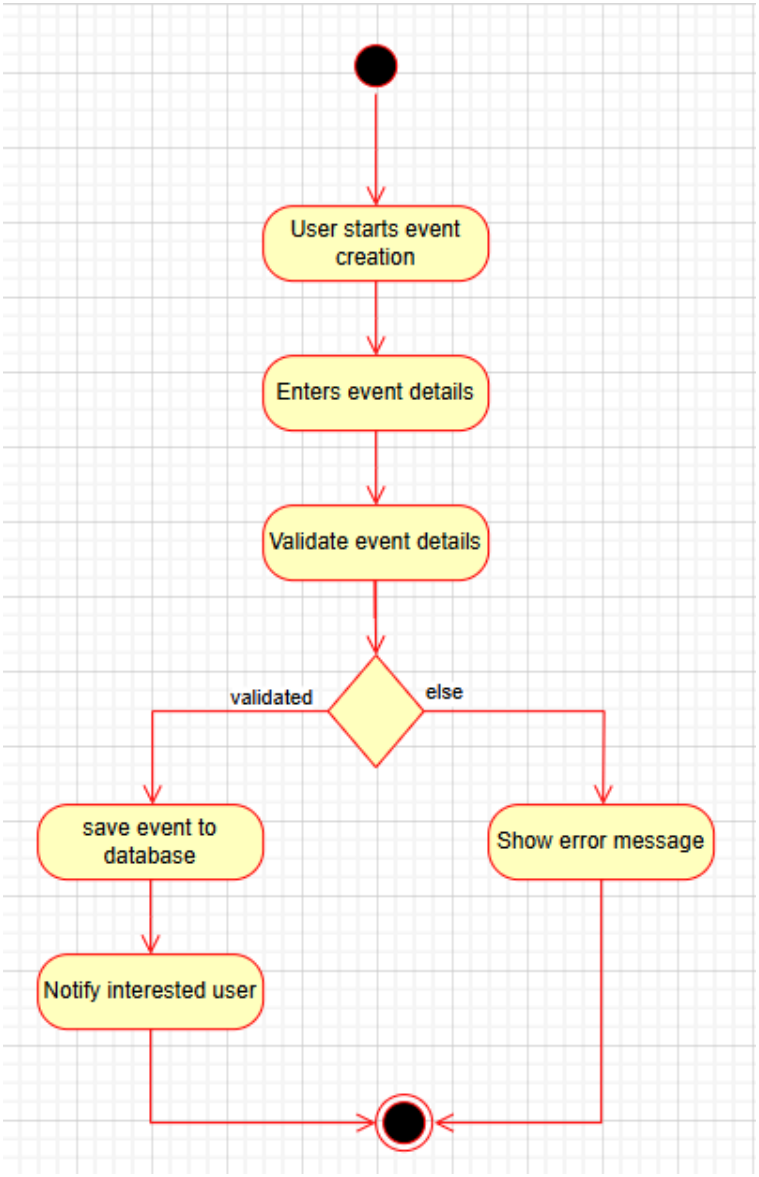
3.2.1.2 Event Creation Sequence Diagram



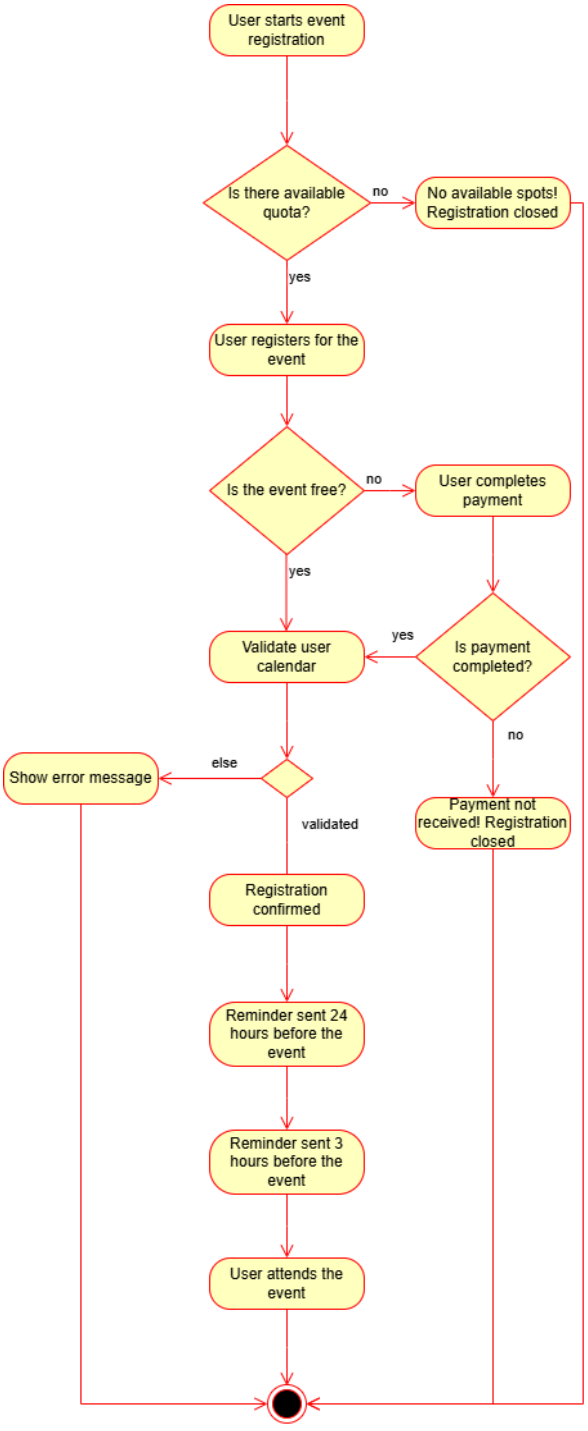
3.2.1.3. Join Event Sequence Diagram



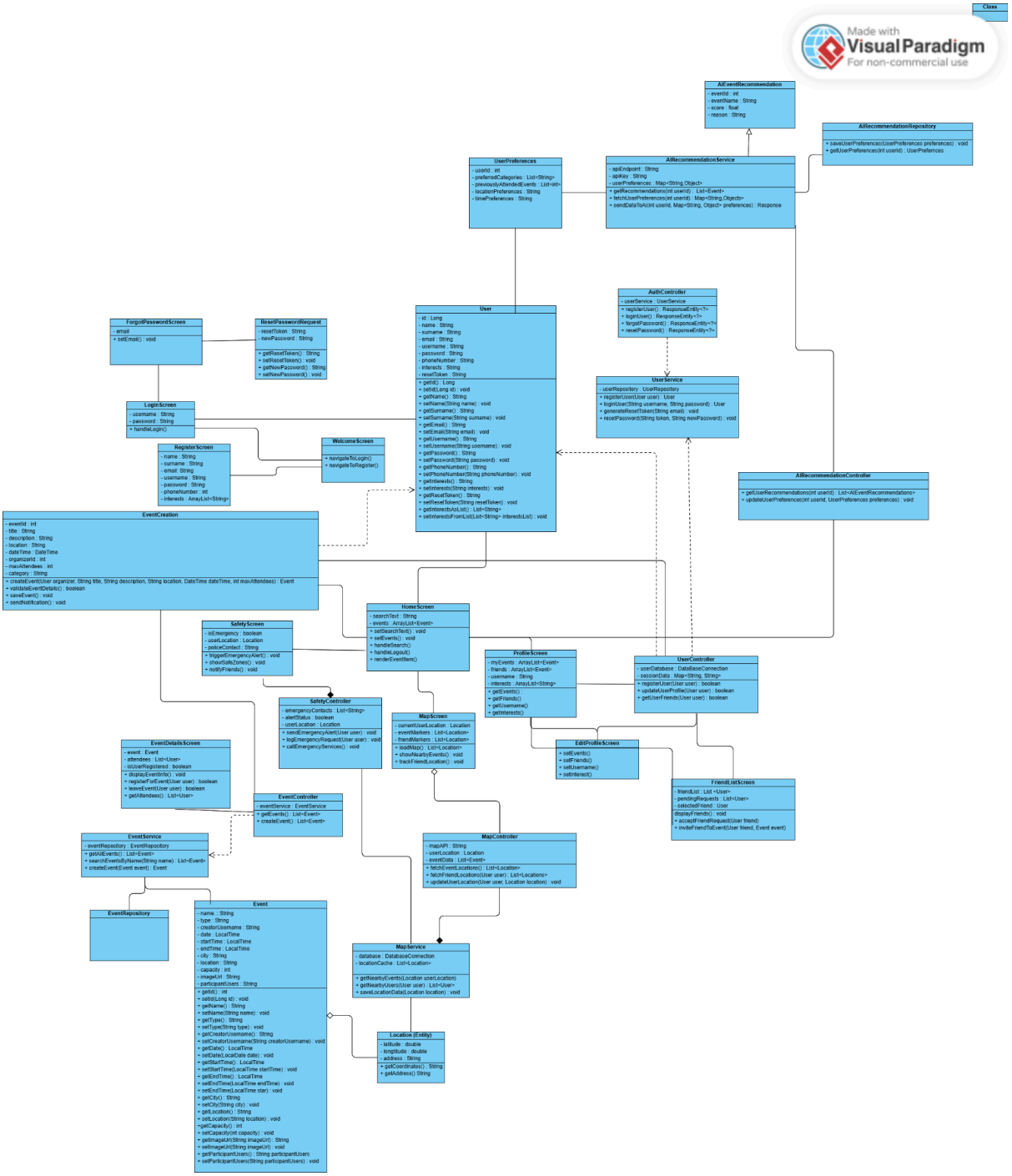
3.2.1.4 Event Creation Activity Diagram



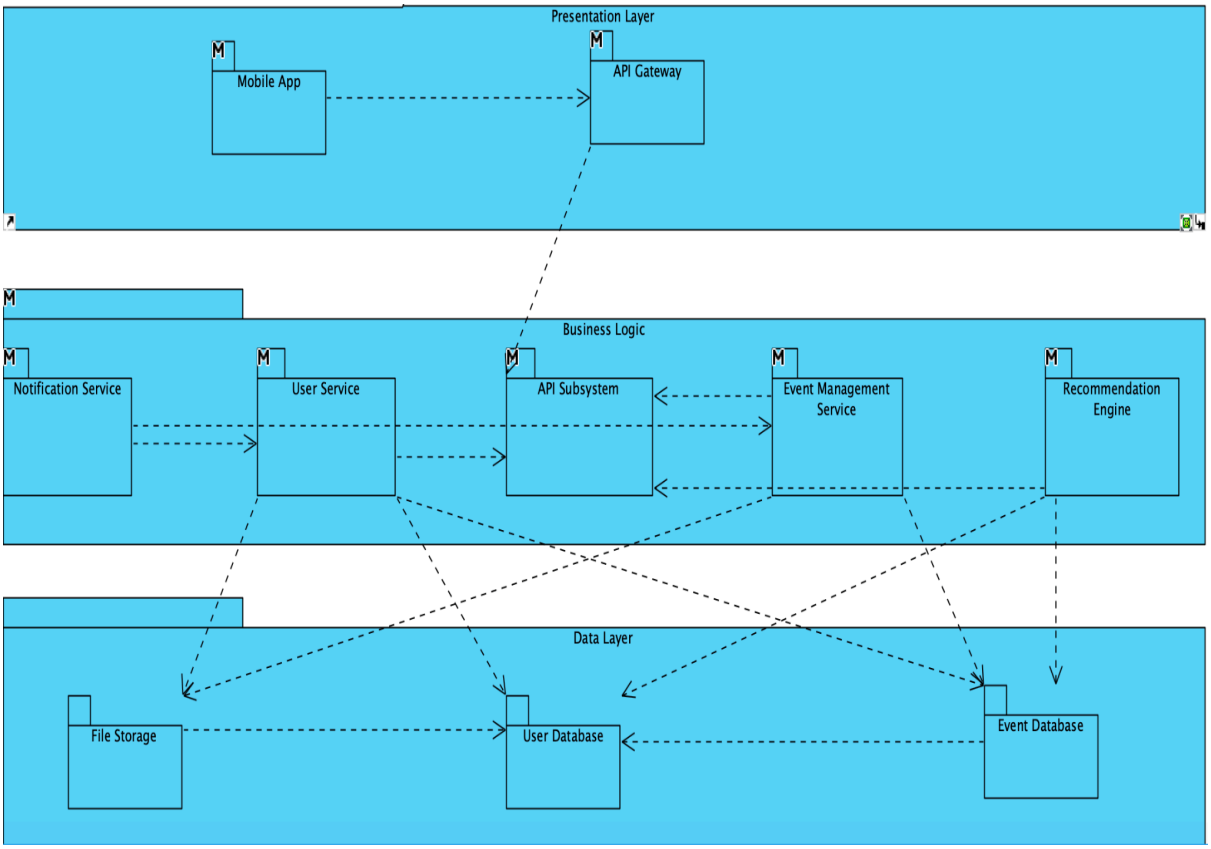
3.2.1.5 Event Registration Activity Diagram



3.2.2 Object and Class Diagram



3.3 Subsystem Decomposition



The three-layer architecture perfectly fits this project as it organizes the system into distinct layers, each with its own responsibilities, making the application easier to develop, maintain, and scale. The Presentation Layer is focused on how the user interacts with the system. It handles all the user-facing components through the mobile app and API Gateway, ensuring smooth communication between the user and the backend. The Business Logic Layer serves as the system's heart, performing essential tasks such as managing user accounts, handling events, providing personalized recommendations, and sending notifications. Centralizing the core functionality here ensures that the system remains flexible and extensible. Finally, the Data Layer manages all data storage, retrieval, and security, including user profiles, event details, and file storage. This layer ensures that all data is easily accessible, well-organized, and protected. The separation of these three layers allows each part of the system to focus on its specific tasks, improving the overall structure and efficiency of the application.

3.4 Access Control and Security

Security is a critical aspect of *Eventure* because it handles user authentication, event registration, and location tracking. This section outlines the access control mechanisms, authentication strategies, and data security practices implemented to ensure a secure and trustworthy platform.

3.4.1 User Authentication and Authorization

Eventure employs secure authentication mechanisms to ensure that only authorized users can access their accounts and interact with the system. The authentication process follows:

- **JWT (JSON Web Token)-based Authentication**:
    - Users log in using their credentials (email/password), and a JWT token is issued upon successful authentication.
    - This token is used for subsequent requests to ensure secure session management.
    - Tokens have a limited lifespan and require periodic renewal to prevent unauthorized long-term access.

For authorization, the system implements Role-Based Access Control (RBAC):

- **Regular Users** – Can view and register for events, create events, and interact with other attendees.
- **Event Organizers** – Have additional permissions to manage events, including modifying details, setting restrictions, and handling registrations.
- **Admin Users** – Have full access to manage users, delete inappropriate events, and handle security-related concerns.

3.4.2 Data Protection and Privacy

Since *Eventure* deals with sensitive user data, including personal details, event participation, and real-time location tracking, the following security measures are implemented:

- **Data Encryption**:
    - Sensitive user data (e.g., emails and location history) is encrypted in the database to prevent unauthorized access in case of data breaches.
- **Privacy Controls**: Users can control their location-sharing settings and opt out of event visibility to enhance personal privacy.

3.4.3 API Security and Access Controls

To prevent unauthorized access and data breaches, *Eventure* implements strict API security measures:

- **Authentication for API Endpoints**:
    - All API requests require authentication via JWT tokens.
    - Public API endpoints (e.g., event browsing) have restricted access to prevent scraping or spam.
- **Rate Limiting**: Prevents abuse by limiting the number of requests per user per minute.
- **CORS (Cross-Origin Resource Sharing) Policies**: Ensures only authorized frontend applications can access backend services.

4. Subsystem Services

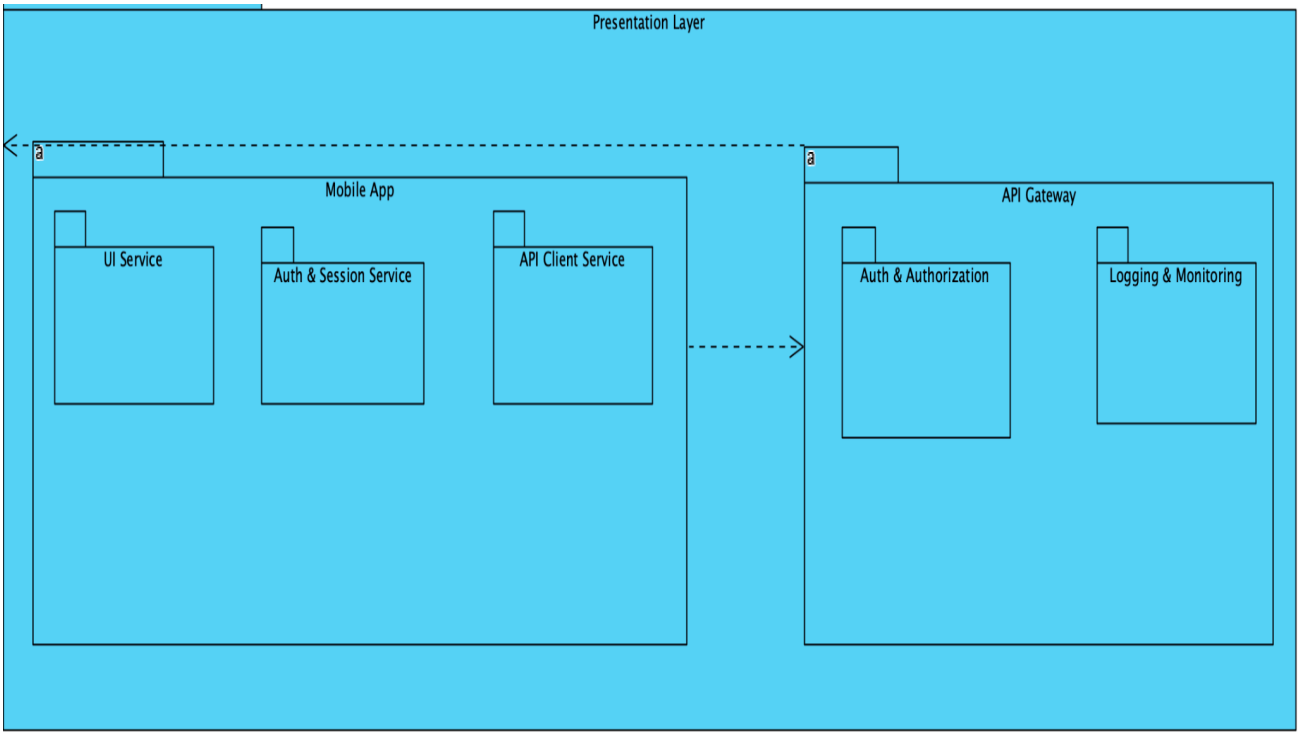
Eventure consists of three main layers: the Presentation Layer, the Business Logic Layer, and the Data Layer. Each layer is critical in ensuring the app's functionality, scalability, and maintainability.

The Presentation Layer is responsible for delivering a seamless user experience. This layer includes the Mobile App and the API Gateway, which handle user interactions, data visualization, and request routing. The Mobile App ensures an intuitive and engaging interface, allowing users to browse events, register, and receive notifications. At the same time, the API Gateway manages communication between clients and backend services, ensuring secure and efficient data exchange.

The Business Logic Layer is where the core functionality of Eventure is implemented. It consists of multiple subsystems, such as the Event Management Subsystem, which handles event creation, updates, and registrations, and the Recommendation Engine, which personalizes event suggestions based on user preferences and historical interactions. Additionally, the Notification Subsystem ensures timely communication, updates, and reminders to users. The API Subsystem, the single REST API, acts as the primary interface for processing business logic requests and managing interactions between the Presentation and Data layers.

The Data Layer is responsible for storing, retrieving, and managing all essential data. It comprises the User Database, which maintains user profiles and authentication details; the Event Database, which stores event information, registrations, and analytics; and the File Storage Subsystem, which handles media uploads, such as event images and user profile pictures. This layer ensures data integrity, security, and efficient retrieval, enabling smooth interactions between the app's components.

4.1 Presentation Layer



The Presentation Layer of Eventure is structured into two main sub-packages: Mobile App and API Gateway. Each sub-package serves a distinct role in handling user interactions and facilitating communication with the backend. Mobile App contains all UI components that users interact with directly. It is responsible for event discovery, registration, user authentication, profile management, and notifications. The app ensures a smooth and engaging user experience by providing intuitive navigation, visually appealing event listings, and real-time updates. It also manages in-app messaging and social features to enhance community engagement. API Gateway acts as an intermediary between the Mobile App and the Business Logic Layer. It is responsible for routing requests, enforcing security policies, and optimizing data exchange. This sub-package ensures efficient communication by managing authentication, load balancing, and request validation before forwarding them to the backend services.

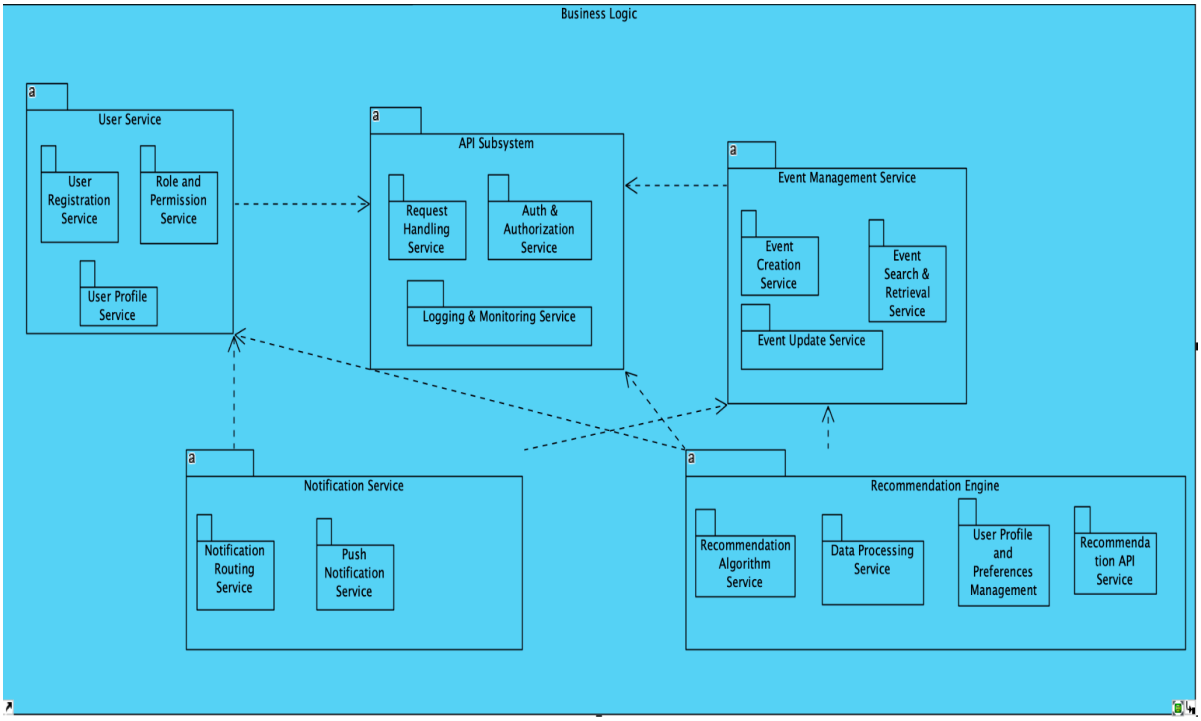
4.1.1 Mobile App

Service Name	Description
Authentication & Session Service	Manages user login, registration, and session management and handles user state while interacting with the Mobile App. Ensures secure access and seamless session persistence.
UI Service	Handles the rendering of UI components and navigation logic within the Mobile App. Ensures a responsive, user-friendly experience by managing the app's screens and user interactions.
API Client Service	Responsible for communicating with the API Gateway, sending requests, and handling responses. It manages API calls for event-related data, user actions, and notifications, ensuring seamless interaction between the frontend and backend.

4.1.2 API Gateway Services

Service Name	Description
Authentication & Authorization Service	Handles the authentication and authorization of incoming requests. Ensures users are authenticated and authorized to access specific resources, enforcing security policies and role-based access control.
Logging & Monitoring Service	Collects, stores, and analyzes logs for all API requests and system events. It provides insights into the application's performance, usage patterns, and potential errors, supporting debugging and performance optimization.

4.2 Business Logic



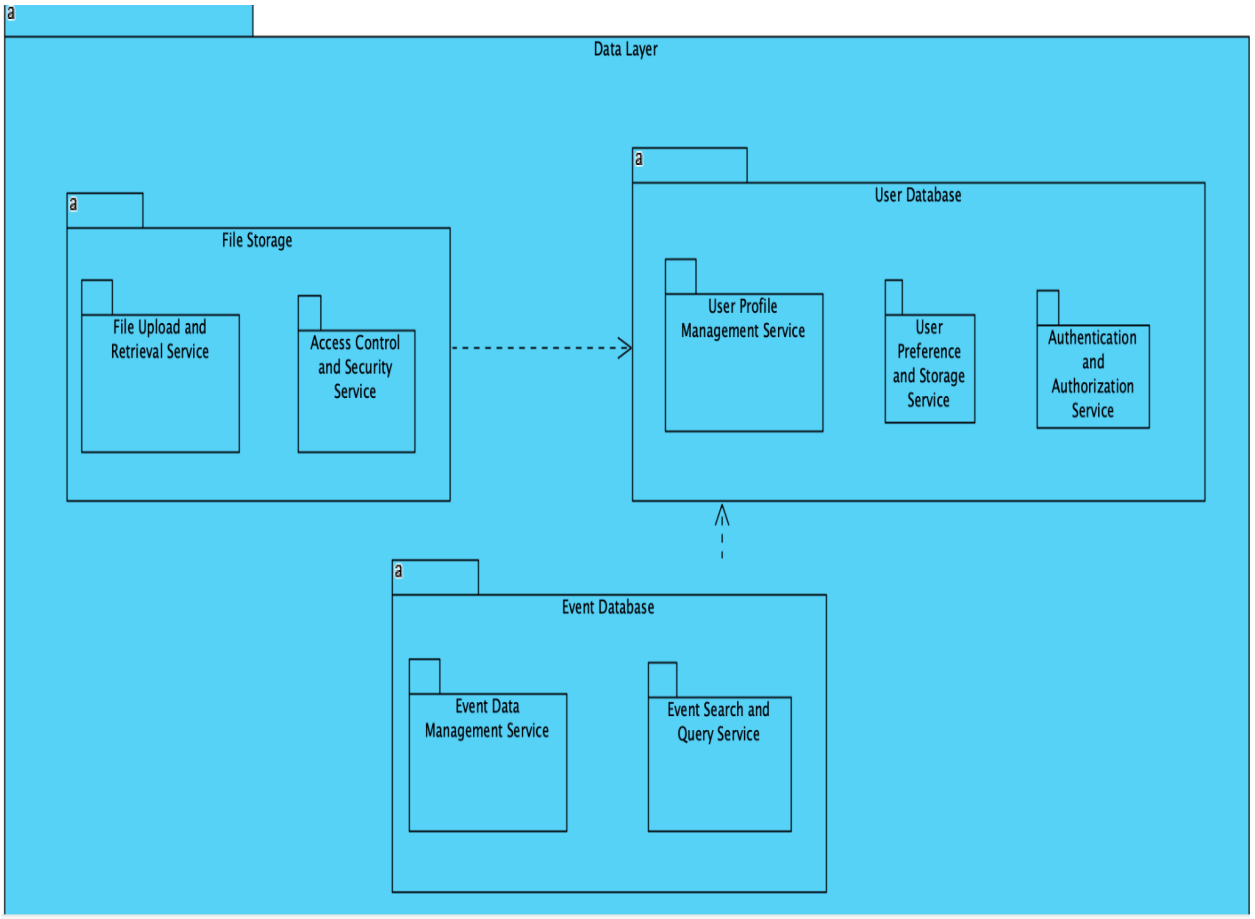
The Business Logic Layer of Eventure processes data, enforces business rules and manages the application's core functionality. It is divided into four key sub-packages: Event Management, Recommendation Engine, Notification System, and API Subsystem. Each sub-package is crucial in ensuring seamless operations and efficient data handling. Event Management handles the creation, modification, and deletion of events. It processes event registrations, manages attendee lists, and ensures that events are updated in real-time. This sub-package enforces business rules such as ticket limits, event status changes, and user access controls. Recommendation Engine analyzes user behavior, event preferences, and historical interactions to provide personalized event recommendations. This sub-package enhances user engagement by suggesting relevant events using machine-learning techniques or rule-based filtering. Notification System manages real-time updates, reminders, and alerts. It ensures that users receive timely notifications about event changes, upcoming events, and personalized suggestions via push notifications, emails, or in-app messages. API Subsystem acts as the single REST API that facilitates communication between the Presentation Layer and the Data Layer. It processes incoming requests, enforces security measures such as authentication and authorization, and ensures efficient data retrieval and modification.

Service Name	Subservices	Description
Event Management Service	Event Creation Service	Responsible for creating new events, including validating data, setting up event details, and storing them in the system.
Event Search & Retrieval Service		Handles searching and retrieving events based on user queries, filters, and preferences.
Event Update Service		Manages modifying existing events, including event details, dates, or ticket availability updates.
Recommendation Service	Recommendation Algorithm Service	Implements algorithms to analyze user preferences and behaviors, generating personalized event recommendations.
Data Processing Service		Processes raw user and event data, preparing it for use in the recommendation system.
User Profile & Preferences Service		Manages user profile information and preferences, which are

		crucial for tailoring recommendations.
Recommendation API Service		Exposes an API endpoint for fetching personalized event recommendations based on the user profile and historical data.
User Profile Service	User Registration Service	Manages the user registration process, including account creation and validation of user information.
Role & Permission Management Service		Handles user roles (e.g., admin, user, guest) and associated permissions, determining access to specific features and data.
User Profile Management Service		Manages the user's personal information, preferences, and settings within the application.
Notification Service	Notification	Routes notifications to the appropriate channels based on user preferences and types of notifications (e.g., push, email, in-app messages).

Push Notification Service		Sends push notifications to users regarding event reminders, updates, or recommendations.
API Service	Logging and Monitoring Service	Monitors API performance, logs requests, responses, and errors, and provides system health and activity tracking.
Authentication & Authorization Service		Handles user authentication, and ensures users are authorized to access specific resources based on roles and permissions.
Request Handling Service		Handles incoming requests, routing them to appropriate services, managing request data, and ensuring correct responses are sent.

4.3 Data Layer



The Data Layer of Eventure is divided into three main sub-packages: User Database, Event Database, and File Storage. Each sub-package is responsible for storing, retrieving, and managing critical data, ensuring efficient and secure data handling across the application. User Database stores and manages user-related information, including authentication credentials, profile details, preferences, and historical interactions. It ensures secure access to user accounts, enabling authentication and authorization mechanisms while maintaining data integrity. Event Database contains all event-related data, including event listings, registrations, participant details, and historical event analytics. This sub-package ensures efficient querying and retrieval of event information, enabling personalized recommendations and seamless event management. File Storage handles media uploads, such as event images, user profile pictures, and other multimedia content. It ensures proper organization, retrieval, and optimization of stored files, allowing smooth integration with different application layers.

Subsystem	Service Name	Description
File Storage Subsystem	File Upload & Retrieval Service	Responsible for handling file uploads, securely storing files, and retrieving them as needed. Ensures proper indexing of files for easy access.
Access Control & Security Service		Ensures that files and storage systems are secure, preventing unauthorized access, implementing encryption, and managing security protocols for stored files. This service manages access rights and permissions for users to interact with stored files.
User Database Subsystem	User Profile Management Service	Manages the storage and retrieval of user profile data, ensuring that user details (e.g., name, contact info) are kept up-to-date.
Authentication & Authorization Service		Handles the management of user authentication (login) and authorization (permissions) by securely storing user credentials and verifying user access levels.

	User Preference Storage Service	Manages and stores user preferences, including settings related to notifications, privacy, and display options.
Event Database Subsystem	Event Data Management Service	Manages the storage and retrieval of event-related data, including event details (name, location, time, etc.), user interaction history, and other related information.
	Event Search & Query Service	Provides efficient searching and querying of events, enabling users to find events based on various filters (e.g., location, date, type, etc.).

5. Test Cases

5.1 Functional Test Cases

5.1.1 Create Account

Test ID: TF001
Summary/Title/Objective: Create Account

The procedure of testing steps:

1. Launch the application.
2. Click on the underlined “Register” text.
3. Fill out the required information: name, email, and password.
4. Click on the "Register" button.

Expected results/Outcome:
The user should be able to create an account successfully.

Priority/Severity: Critical

5.1.2 Login

Test ID: TF002
Summary/Title/Objective: User Login

The procedure of testing steps:

1. Launch the application.
2. Tap “Login” on the welcome screen.
3. Enter a valid username/email and password.
4. Click on the “Submit” button.

Expected results/Outcome:
The user should be successfully logged in and redirected to the home screen.

Priority/Severity: Critical

5.1.3 Reset Password

Test ID: TF003
Summary/Title/Objective: Reset Password

The procedure of testing steps:

1. Launch the application.
2. Tap “Login” on the welcome screen, then click “Forgot Password.”
3. Enter a valid registered email address.
4. Click on “Send Reset Instructions.”
5. Enter the received reset token and a new password on the reset screen.
6. Click on “Reset Password.”

Expected results/Outcome:
The user’s password is updated, and they can log in with the new password.

Priority/Severity: High

5.1.4 Create Event

Test ID: TF004
Summary/Title/Objective: Create a new event

The procedure of testing steps:

1. Log in to the application.
2. Navigate to the “Create Event” form.
3. Fill out event details (name, date, location, etc.).
4. Click on the “Create” or “Submit” button.

Expected results/Outcome:
A new event has been successfully created and appears on the event list.

Priority/Severity: High

5.1.5 Join Event

Test ID: TF005
Summary/Title/Objective: Join an existing event

The procedure of testing steps:

1. Log in to the application.
2. Navigate to the event details screen of an existing event.
3. Tap the “Join Event” button.
4. Confirm the join action if prompted.

Expected results/Outcome:
The user is successfully added to the event’s participant list.

Priority/Severity: High

5.1.6. Leave Event

Test ID: TF006
Summary/Title/Objective: Leave an event the user has joined

The procedure of testing steps:

1. Log in to the application.
2. Navigate to an event that the user has already joined.
3. Tap the “Leave Event” button.
4. Confirm the leave action if prompted.

Expected results/Outcome:
The user is removed from the participant list of that event.

Priority/Severity: Medium

5.1.7 Search Events

Test ID: TF007
Summary/Title/Objective: Search for events by keyword

The procedure of testing steps:

1. Log in to the application.
2. Navigate to the home screen or search screen.
3. Enter a keyword in the search bar (e.g., “music”).
4. Tap the “Search” button.

Expected results/Outcome:
The system displays a list of events matching the search keyword.

Priority/Severity: Medium

5.1.8 Forgot Password (Invalid Email)

Test ID: TF008
Summary/Title/Objective: Handle “Forgot Password” with an unregistered email

The procedure of testing steps:

1. Launch the application.
2. Tap “Login,” then “Forgot Password.”
3. Enter an invalid or unregistered email address.
4. Click on “Send Reset Instructions.”

Expected results/Outcome:
The system displays an error message indicating the email has not been found.

Priority/Severity: Low

5.1.9 Cancel Event (Organizer)

Test ID: TF009
Summary/Title/Objective: The event organizer cancels an event

The procedure of testing steps:

1. Log in as an organizer who created an event.
2. Go to “My Events” or the event details page.
3. Click on the “Cancel Event” button.
4. Confirm the cancellation action.

Expected results/Outcome:
The event status changes to “canceled,” and participants are notified (if notifications are enabled).

Priority/Severity: High

5.1.10 View Recommended Events

Test ID: TF010
Summary/Title/Objective: Display recommended events based on user interests

The procedure of testing steps:

1. Log in to the application.
2. Navigate to the “Recommended Events” section (or the home screen if it displays recommendations).
3. Observe the list of events presented.

Expected results/Outcome:
The system lists events that align with the user’s stored interests.

Priority/Severity: Medium

5.1.11 View Event Details

Test ID: TF011

Summary/Title/Objective: Verify that users can view the details of an event.

The procedure of Testing Steps:

1. Log in to the application.
2. Navigate to the list of events.
3. Click on an event to view details.

Expected results/Outcome: The system should display the event title, description, location, date, organizer details, capacity, and attendees.

Priority/Severity: Medium

5.1.12 Edit Event (Organizer)

Test ID: TF012

Summary/Title/Objective: Ensure event organizers can edit event details.

The procedure of Testing Steps:

1. Log in as the event organizer.
2. Go to "My Events" and select an event.
3. Click "Edit" and modify details (title, date, location, description, etc.).
4. Save the changes.

Expected results/Outcome: The updated event details should be reflected across the system.

Priority/Severity: High

5.1.13 Remove Event (Admin)

Test ID: TF013

Summary/Title/Objective: Ensure administrators can remove inappropriate events.

The procedure of Testing Steps:

1. Log in as an admin.
2. Navigate to event management.
3. Select an event and click "Remove."
4. Confirm the removal.

Expected results/Outcome: The event should be deleted from the system, and users should not be able to access it.

Priority/Severity: Critical

5.1.14 Invite Friends to Event

Test ID: TF014

Summary/Title/Objective: Allow users to invite their friends to events.

The procedure of Testing Steps:

1. Log in to the application.
2. Open an event’s details page.
3. Click the “Invite Friends” button.
4. Select a friend from the list.
5. Send the invitation.

Expected results/Outcome: The selected friend should receive an invitation notification.

Priority/Severity: Medium

5.1.15 View Friends Attending an Event

Test ID: TF015

Summary/Title/Objective: Ensure users can see which friends are attending an event.

The procedure of Testing Steps:

1. Log in to the application.
2. Open an event's details page.
3. Navigate to the "Attendees" section.

Expected results/Outcome: A list of attending friends should be displayed.

Priority/Severity: Medium

5.1.16 Use the Safety Button

Test ID: TF016

Summary/Title/Objective: Verify that the safety button alerts emergency services.

The procedure of Testing Steps:

1. Log in to the application.
2. Navigate to the safety button feature.
3. Click the button.
4. Confirm the emergency alert.

Expected results/Outcome: An alert should be sent to authorities with the user's location.

Priority/Severity: Critical

5.1.17 Social Media Login Integration

Test ID: TF017

Summary/Title/Objective: Allow users to register or log in using a social media account (e.g., Facebook, Google).

The procedure of Testing Steps:

1. Launch the application.
2. Click the “Log In with Facebook/Google” option.
3. Enter valid social media credentials.
4. Authorize the app if prompted.

Expected Results/Outcome: The user is logged in, and an account is created or linked in the system if it’s their first time logging in with that social media account.

Priority/Severity: High

5.1.18 Direct Messaging Between Event Participants

Test ID: TF018

Summary/Title/Objective: Verify that users can send and receive direct messages within an event group.

The procedure of Testing Steps:

1. Log in to the application.
2. Join or create an event.
3. Navigate to the “Messages” section for that event.
4. Select a user from the participants' list and send a direct message.
5. Observe if the recipient can view and reply to the message.

Expected Results/Outcome:

1. Messages are delivered in real time.
2. Both the sender and the recipient can view the conversation thread.

Priority/Severity: Medium

5.1.19 Event Check-In Feature

Test ID: TF019

Summary/Title/Objective: Ensure users can check in to an event upon arrival to confirm attendance.

The procedure of Testing Steps:

1. Log in to the application.
2. Navigate to an event in which the user is a participant.
3. Tap the “Check In” button when physically at the event location (optionally using GPS validation).
4. Observe the updated attendee status.

Expected Results/Outcome: The user's status changes to "Checked In," and the attendee list is updated accordingly.

Priority/Severity: High

5.1.20 Rate and Review Event

Test ID: TF020

Summary/Title/Objective: Verify users can rate events and leave written reviews.

The procedure of Testing Steps:

1. Log in to the application.
2. Navigate to "My Past Events" or an already concluded event.
3. Select "Rate & Review," enter a star rating (1–5) and a comment, then submit.

Expected Results/Outcome:

1. The review is saved and displayed with the event.
2. Ratings are reflected in the overall average event score.

Priority/Severity: Medium

5.1.21 Push Notification Preferences

Test ID: TF021

Summary/Title/Objective: Allow users to customize which push notifications they receive (e.g., invites, reminders, announcements).

The procedure of Testing Steps:

1. Log in to the application.
2. Go to "Settings" → "Notifications."
3. Toggle specific notifications on or off (e.g., "Event Invites," "Event Reminders," etc.).
4. Trigger an action that typically sends a notification (e.g., event organizer updates an event the user has joined).

Expected Results/Outcome:

1. Only the toggled-on notifications are received.
2. Toggled-off notifications are suppressed.

Priority/Severity: Medium

5.1.22 Update User Profile

Test ID: TF022
Summary/Title/Objective: Update User Profile Information
The procedure of testing steps:

1. Log in to the application.
2. Navigate to the “Profile” or “Account Settings” page.
3. Modify personal details such as name, bio, and contact information.
4. Click “Save” or “Update.”

Expected results/Outcome: The updated profile information is displayed accurately on the user’s profile.
Priority/Severity: Medium

5.1.23 Delete Account

Test ID: TF023
Summary/Title/Objective: Allow users to delete their accounts permanently
The procedure of testing steps:

1. Log in to the application.
2. Navigate to “Account Settings” and select “Delete Account.”
3. Enter the required confirmation (e.g., password or verification code).
4. Confirm the deletion action.

Expected results/Outcome: The user’s account and all associated data are permanently removed from the system.
Priority/Severity: Critical

5.1.24 Two-Factor Authentication Setup

Test ID: TF024
Summary/Title/Objective: Enable users to add an extra layer of security via two-factor authentication (2FA)
The procedure of testing steps:

1. Log in to the application.
2. Navigate to “Security Settings.”

3. Select “Enable Two-Factor Authentication.”
4. Follow the on-screen instructions to link a phone number or authenticator app.

Expected results/Outcome: The user receives a verification code, and 2FA is successfully activated.

Priority/Severity: High

5.1.25 Upload Profile Picture

Test ID: TF025

Summary/Title/Objective: Verify that users can upload and update their profile picture

The procedure of testing steps:

1. Log in to the application.
2. Go to the “Profile” or “Account Settings” page.
3. Click the “Upload Photo” button and select an image file from the device.
4. Crop/adjust the image if prompted, then click “Save.”

Expected results/Outcome: The new profile picture is successfully updated and displayed on the user’s profile.

Priority/Severity: Medium

5.1.26 Report Event

Test ID: TF026

Summary/Title/Objective: Allow users to report or flag an event for inappropriate content

The procedure of testing steps:

1. Log in to the application.
2. Navigate to an event’s detail page.
3. Click the “Report Event” button.
4. Select a reason for reporting from the available options.
5. Submit the report.

Expected results/Outcome: The system acknowledges the report and flags the event for review by moderators.

Priority/Severity: High

5.1.27 RSVP for Event Invitation

Test ID: TF027
Summary/Title/Objective: Verify that users can RSVP to an event invitation
The procedure of testing steps:

1. Log in to the application.
2. Open the event invitation notification or details page.
3. Click the “RSVP” button.
4. Choose the appropriate attendance option (e.g., “Going,” “Maybe,” “Not Going”).

Expected results/Outcome: The RSVP status is updated and visible to the event organizer.
Priority/Severity: Medium

5.1.28 Create Group Chat for Event

Test ID: TF028
Summary/Title/Objective: Enable event participants to create a group chat
The procedure of testing steps:

1. Log in to the application.
2. Navigate to an event where the user is participating.
3. Tap on the “Messages” or “Chat” section.
4. Click “Create Group Chat” and select participants from the attendee list.
5. Initiate the chat conversation.

Expected results/Outcome: A new group chat will be created and made accessible to the selected participants.
Priority/Severity: Medium

5.1.29 Block/Unblock User in Chat

Test ID: TF029
Summary/Title/Objective: Ensure that users can block and later unblock other users in event chats
The procedure of testing steps:

1. Log in to the application and open an event chat.

2. Select a participant's profile and choose "Block User."
3. Verify that messages from the blocked user are no longer received.
4. Navigate to settings and select "Unblock" for the user.

Expected results/Outcome: Blocking prevents further messages from the user, and unblocking restores communication.

Priority/Severity: Medium

5.1.30 Event Sharing on Social Media

Test ID: TF030

Summary/Title/Objective: Allow users to share event details on social media platforms

The procedure of testing steps:

1. Log in to the application.
2. Open an event's detail page.
3. Click the "Share" button and select a social media option (e.g., Facebook, Twitter).
4. Confirm the sharing action.

Expected results/Outcome: The event details are successfully shared on the chosen social media platform.

Priority/Severity: Low

5.1.31 Filter Events by Category and Date

Test ID: TF031

Summary/Title/Objective: Enable users to filter event listings based on selected categories and date ranges

The procedure of testing steps:

1. Log in to the application.
2. Navigate to the event search or listing page.
3. Apply filters such as event category (e.g., Music, Sports) and a specific date range.
4. Click the "Apply" or "Search" button.

Expected results/Outcome: The event list updates to display only those events matching the chosen filters.

Priority/Severity: Medium

5.1.32 Calendar Sync Integration

Test ID: TF032
Summary/Title/Objective: Verify that users can sync event details with their device calendar
The procedure of testing steps:

1. Log in to the application.
2. Open an event's detail page.
3. Click the "Add to Calendar" button.
4. Select the preferred calendar application from the device.

Expected results/Outcome: The event details are successfully added to the user's device calendar with correct timing and reminders.
Priority/Severity: Medium

5.1.33 Edit User Profile Information

Test ID: TF033
Summary/Title/Objective: Verify that users can modify additional personal information
The procedure of testing steps:

1. Log in to the application.
2. Access the "Profile" or "Account Settings" page.
3. Update fields such as bio, phone number, and preferences.
4. Click "Save" or "Update."

Expected results/Outcome: The updated details are immediately reflected on the user's profile page.
Priority/Severity: Medium

5.1.34 Logout Functionality

Test ID: TF034

Summary/Title/Objective: Ensure users can securely log out.

The procedure of Testing Steps:

1. Log in to the application.
2. Navigate to the logout option.
3. Click "Logout."
4. Try accessing protected pages after logging out.

Expected results/Outcome: Users should be redirected to the login screen and prevented from accessing secure pages.

Priority/Severity: High

5.2 Non-functional Test Cases

5.2.1 Performance Under Load

Test ID: TNF001

Test Type/Category: Performance

Summary/Title/Objective: Ensure the application remains responsive under high traffic.

The procedure of Testing Steps:

1. Simulate 1000+ concurrent users accessing the system.
2. Measure response time for login, event search, and event registration.

Expected results/Outcome: The application should handle load efficiently without excessive lag.

Priority/Severity: Critical

5.2.2 Security: SQL Injection Prevention

Test ID: TNF002

Test Type/Category: Security

Summary/Title/Objective: Verify that the system is protected against SQL injection.

The procedure of Testing Steps:

1. Attempt SQL injection attacks in input fields.
2. Observe system behavior.

Expected results/Outcome: The system should block any SQL injection attempts.

Priority/Severity: Critical

5.2.3 Event Notifications

Test ID: TNF003

Test Type/Category: Usability

Summary/Title/Objective: Verify that users receive notifications about event updates.

The procedure of Testing Steps:

1. Create an event.
2. Invite another user to the event.
3. Modify the event details.
4. Check if the invited user receives a notification.

Expected results/Outcome: The invited user should receive an update notification.

Priority/Severity: High

5.2.4 Server Response Time for Event Queries

Test ID: TNF004

Test Type/Category: Performance

Summary/Title/Objective: Measure the time taken to fetch and display event listings.

The procedure of Testing Steps:

1. Search for events using different filters.
2. Measure response times under varying load conditions (e.g., 10, 100, 1000 users).

Expected results/Outcome: Event search results should load within 2 seconds under normal conditions and 5 seconds under peak load.

Priority/Severity: High

5.2.5 GDPR Compliance (User Data Deletion)

Test ID: TNF005

Test Type/Category: Compliance

Summary/Title/Objective: Verify that users can delete their personal data as per GDPR.

The procedure of Testing Steps:

1. Log in to the system.
2. Navigate to the account settings.
3. Click on Delete Account and confirm the action.
4. Verify that all personal data (name, email, event history) is removed from the database.

Expected results/Outcome: All personal data should be permanently deleted or anonymized.

Priority/Severity: High

5.2.6 Mobile App Startup Time

Test ID: TNF006

Test Type/Category: Performance

Summary/Title/Objective: Measure how fast the mobile app loads.

The procedure of Testing Steps:

1. Open the mobile app on a device.
2. Measure how long it takes for the home screen to appear.

Expected results/Outcome: The application should start on devices within 5 seconds.

Priority/Severity: Medium

5.2.7 Accessibility Compliance (WCAG)

Test ID: TNF007

Test Type/Category: Accessibility

Summary/Title/Objective: Ensure the application meets accessibility guidelines such as WCAG 2.1 for users with disabilities.

The procedure of Testing Steps:

1. Use screen readers (e.g., VoiceOver, TalkBack) to navigate the application.
2. Check color contrast, font sizes, and labeling of UI elements.
3. Attempt keyboard-only or gesture-only navigation.

Expected Results/Outcome:

1. The application is navigable via assistive technologies.
2. All form fields and interactive elements are correctly labeled for accessibility.

Priority/Severity: High

5.2.8 Cross-Browser Compatibility

Test ID: TNF008

Test Type/Category: Compatibility

Summary/Title/Objective: Verify that the web version of the application works correctly across different browsers (Chrome, Firefox, Safari, Edge).

The procedure of Testing Steps:

1. Launch the application in each supported browser.
2. Perform core actions (login, search events, join an event, etc.).

Expected Results/Outcome: The application layout and functionality are consistent across all tested browsers.

Priority/Severity: High

5.2.9 Data Encryption in Transit

Test ID: TNF009

Test Type/Category: Security

Summary/Title/Objective: Verify that all sensitive data (login, payment details, personal information) is transmitted securely over HTTPS.

The procedure of Testing Steps:

1. Use a network monitoring tool (e.g., Wireshark) while logging in or performing a payment.
2. Observe data packets for encryption.

Expected Results/Outcome: All sensitive data is encrypted using TLS/SSL, and no plain-text credentials are visible in the network trace.

Priority/Severity: Critical

5.2.10 Backup and Restore

Test ID: TNF010

Test Type/Category: Reliability

Summary/Title/Objective: Ensure system data (events, user accounts, messages) can be backed up and restored without data loss.

The procedure of Testing Steps:

1. Perform a scheduled or manual backup of the database.
2. Simulate a data failure or wipe a test environment.
3. Restore the system from the backup.

Expected Results/Outcome: The system is fully restored with accurate event details, user accounts, and messages intact.

Priority/Severity: High

5.2.11 Stress Test for Map and Location-Based Features

Test ID: TNF011

Test Type/Category: Performance

Summary/Title/Objective: Confirm that the interactive event map remains functional with many simultaneous users or events displayed.

The procedure of Testing Steps:

1. Simulate a high volume of events (e.g., 10,000+ events) spread across a map region.
2. Simulate multiple users accessing and zooming/panning the map concurrently.

Expected Results/Outcome:

1. The map feature continues to render smoothly.
2. Filter/search operations are still responsive under heavy load.

Priority/Severity: Critical

5.2.12 Automated Logout after Inactivity

Test ID: TNF012
Test Type/Category: Non-functional

Summary/Title/Objective: Ensure the application automatically logs out users after a period of inactivity for security purposes

The procedure of testing steps:

1. Log in to the application.
2. Leave the application idle for a predefined period (e.g., 10 minutes).
3. Attempt to navigate to a secure page or perform an action.

Expected results/Outcome: The system automatically logs out the user after the idle period, requiring re-authentication.

Priority/Severity: High

5.2.13 Scalability: Database Performance Under High Volume

Test ID: TNF013

Test Type/Category: Performance/Scalability

Summary/Title/Objective: Ensure the database scales effectively when handling a very large volume of data.

The procedure of testing steps:

1. Populate the database with a large dataset (simulate millions of events and user records).
2. Monitor query performance and response times for core operations (e.g., event searches and user logins).
3. Observe the behavior under stress conditions.

Expected results/Outcome: The system continues to operate efficiently without significant performance degradation, even with high data volumes.

Priority/Severity: High

5.2.14 Network Conditions Compatibility

Test ID: TNF014

Test Type/Category: Performance/Compatibility

Summary/Title/Objective: Verify the application's acceptable performance under varying network speeds.

The procedure of testing steps:

1. Use network simulation tools to emulate slow, moderate, and fast network conditions.
2. Test key functionalities (e.g., login, event search, event registration) under each simulated network speed.

Expected results/Outcome: The application remains functional, and user interactions are complete within acceptable time frames, even under slower network conditions.

Priority/Severity: Medium

5.2.15 Crash Recovery and Fault Tolerance

Test ID: TNF015

Test Type/Category: Reliability

Summary/Title/Objective: Ensure the system can recover gracefully from unexpected crashes or failures.

The procedure of testing steps:

1. Forcefully terminate the application during a critical operation (e.g., during event registration).
2. Restart the application and verify data integrity and user session recovery.

Expected results/Outcome: The application recovers without data loss, and users are informed of the recovery process with minimal disruption.

Priority/Severity: Critical

5.2.16 API Performance Under Concurrent Requests

Test ID: TNF016

Test Type/Category: Performance

Summary/Title/Objective: Evaluate the performance of backend APIs when subjected to simultaneous requests.

The procedure of testing steps:

1. Use a load testing tool to simulate many concurrent API requests (for operations such as login, event creation, and event search).
2. Monitor response times and error rates during the test.

Expected results/Outcome: APIs respond within acceptable time limits, and error rates remain minimal under high load.

Priority/Severity: High

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

Accessibility: Eventure must comply with international accessibility standards like WCAG 2.1, ensuring that all users, including those with disabilities, can navigate and use the application effortlessly, whether through screen readers, keyboard navigation, or other assistive technologies.

Aesthetics: The user interface must be visually appealing and designed to foster user engagement and trust. The color palette, typography, and layout should provide a clean, modern look while maintaining professional integrity suitable for a global audience.

Cost: The development process is constrained by a limited budget, requiring reliance on open-source frameworks like React and Kotlin. Additionally, efforts must avoid paid APIs where feasible and explore free or low-cost alternatives to reduce operational expenses.

Data Privacy: A key constraint is adhering to GDPR and other relevant privacy laws. The system must protect user data, including preferences, location, and interactions, through encryption, secure authentication, and limited data retention policies.

Functionality: The application must support dynamic functionalities such as personalized event recommendations, location-based filtering, and user feedback sharing. These features must be implemented seamlessly, ensuring reliability and ease of use for the end-user.

Interoperability: The system must integrate with multiple external APIs, such as those for events, transportation, and accommodations, to provide up-to-date information. This necessitates maintaining compatibility with diverse data formats and ensuring smooth data exchange.

Maintainability: The application’s codebase and architecture must be modular and well-documented, allowing for future updates, debugging, and integration of additional features without significant overhead.

Scalability: Eventure must be designed to handle increasing users and events efficiently. The backend should support growth through optimized database queries, caching mechanisms, and load balancing to maintain performance.

Sustainability: The app should encourage environmentally responsible behavior, such as prioritizing local event suggestions or optimizing transportation options to minimize carbon emissions and reduce environmental impact.

Usability: The application must prioritize user-friendliness, offering a straightforward, intuitive interface that minimizes the learning curve. This includes clearly labeled navigation, responsive design, and seamless workflows to enhance user satisfaction.

Table 1: Factors that can affect analysis and design

Factors	Effect level	Effect
Public health	3	The app promotes mental well-being by encouraging social interactions, but its effect on physical health is minimal.
Public safety	5	Ensures safe interactions through authentication mechanisms for user connections.
Public welfare	6	Enhances community well-being by fostering connections and providing access to affordable event experiences.
Global factors	7	Caters to a global audience by accommodating diverse user preferences and integrating international APIs.
Cultural factors	7	Ensures inclusivity and adapts recommendations to align with cultural preferences and sensitivities.
Social factors	8	Strengthens community bonds and facilitates meaningful connections through event-based networking.
Environmental factors	4	Encourages nearby events and minimizes transfers in travel to reduce carbon emissions.
Economic factors	6	Offers cost-effective travel and event planning options while supporting local businesses.

6.2 Standards

The following standards are applied in the development of Eventure:

- **ISO/IEC 27001**: To ensure robust data security practices.
- **IEEE 830**: For documenting system requirements.
- **WCAG 2.1**: To provide accessibility compliance.
- **JSON Schema**: Standardizes data exchange between APIs and the application.

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

After discussions, all crucial decisions are made collectively, ensuring each team member is aligned with the project goals. Each member actively contributes to development and report writing.

Damla:

- Works on both frontend and backend, ensuring seamless integration between the client-side and server-side functionalities.
- Implements React components for event discovery, registration, and user profile management, ensuring a responsive and intuitive user experience.
- Develops backend functionalities using Spring Boot, including API endpoints for event management, authentication, and data retrieval.

Onur:

- Works in the backend team, integrating the AI model with the database and API endpoints.
- Develops the AI-powered event recommendation system, researching and implementing appropriate machine learning models to suggest relevant events based on user preferences and activity.
- Works on data analysis and feedback processing, refining the recommendation logic to improve accuracy.

Alper:

- Primarily responsible for designing and implementing the frontend experience. This involves creating the user interface, developing responsive layouts, and ensuring a cohesive visual design.
- Work with the backend team to integrate the React components seamlessly with the Spring Boot APIs, guaranteeing smooth data flow and a user-friendly experience.
- Conducting user interface research, refining usability, and maintaining a consistent brand identity across all pages and features.

Ercan:

- Works on the backend to build the business logic.

Gökçe:

- Primarily responsible for frontend development, focusing on designing and implementing the user interface. Works on integrating location-based features, including interactive maps for event discovery and navigation.
- Develops and refines UI components to create a visually appealing, responsive, and user-friendly experience. Conducts usability testing and UI research to enhance accessibility, consistency, and overall design quality across the application.
- Collaborates with the backend team to ensure seamless data flow between the frontend and server, optimizing performance and user interaction.

7.2 Helping to create a collaborative and inclusive environment

We use Jira to divide roles and tasks in the project. We use our WhatsApp group to reach out to each other regarding project-related matters and arrange meetings. As a team, we primarily focus on our individual tasks. Still, we maintain an open and supportive environment where everyone is willing to help each other regardless of whether the issue falls within their assigned responsibilities. We strongly believe in keeping each other informed about updates, whether they are achievements, challenges, or points of confusion. This approach allows us to stay aligned, take collective action when necessary, and ensure the smooth progress of our project.

Since we are working remotely, most of our meetings and discussions occur online, such as Zoom, where we discuss progress and potential improvements. We also communicate regularly through our WhatsApp group to share updates or ask for quick feedback. Even though we focus on our own tasks, we actively support and guide one another, helping to troubleshoot problems and optimize solutions. Each team member is committed to handling their responsibilities while also stepping in to assist wherever needed, ensuring an inclusive and cooperative work environment.

7.3 Taking the lead role and sharing leadership on the team

At the beginning of the project, we distributed work responsibilities based on individual strengths, previous experience, and areas of interest. While each team member has led in different areas, our leadership roles are flexible and dynamic, shifting as we gain more knowledge and progress through development.

For example, some team members had experience with backend development, so they took the lead in designing the database structure and API endpoints. Others were more comfortable with frontend technologies, taking the initiative in UI/UX design and

implementation. Additionally, when preparing this report, different members stepped up to take ownership of specific sections based on their interests and experience.

Rather than having fixed leadership positions, we adjust our roles based on ongoing challenges and the project's evolving needs. If someone encounters an issue in their assigned work, another team member with relevant expertise steps in to guide them. This dynamic structure ensures that we continuously learn from each other and distribute responsibilities to maximize efficiency and teamwork.

8. Glossary

A

- **Access Control** – A security mechanism that restricts unauthorized users from accessing certain system functionalities or data.
- **AI Recommendation System** – An artificial intelligence-based system that suggests user events based on their interests, past interactions, and location.
- **API (Application Programming Interface)** – A set of rules and protocols allowing different software components to communicate.
- **Authentication** – Verifying a user’s identity before granting access to the system.
- **Authorization** – Determining a usercano's actions within the system based on their role and permissions.

B

- **Backend** – The server-side application responsible for handling business logic, database management, and user authentication.

C

- **CRUD (Create, Read, Update, Delete)** – These are the four basic operations performed on a database.

D

- **Database** – A structured collection of data stored and managed using PostgreSQL in this system.

E

- **Encryption** – A security method used to encode sensitive data to prevent unauthorized access.
- **Event** – A scheduled gathering, activity, or occasion that users can create, explore, or attend.
- **Event Map** – A live, interactive visualization displaying event locations, nearby activities, and attending friends.

- **Event Registration** – The process through which users sign up to attend an event within the application.

F

- **Frontend** – The client-side application (developed in React) that users interact with to access system features.

G

- **GPS (Global Positioning System)** – A navigation system that lets users determine their location and find nearby events.

J

- **JWT (JSON Web Token)** – A security token used for user authentication and session management.

M

- **Machine Learning** – A subset of artificial intelligence that allows the system to improve recommendations based on user behavior and preferences.
- **Microservices Architecture** – A design approach that divides the application into smaller, independent services that communicate via APIs.

O

- **OAuth (Open Authorization)** – A protocol that allows secure authorization between different services without sharing passwords.

P

- **PostgreSQL** – A relational database management system used to store and manage data within the application.

R

- **RBAC (Role-Based Access Control)** – A security model that assigns system permissions based on user roles.
- **React** – A JavaScript library used to build the application's user interface.
- **REST API (Representational State Transfer API)** – A communication standard used for exchanging data between the frontend and backend.

S

- **Scalability** – The ability of the system to handle increased workloads, user traffic, and data storage demands over time.
- **Security Token** – A digital credential used to verify a user's identity during authentication and authorization.

- **Session Management** – The process of managing user sessions to ensure secure and seamless interactions within the system.
- **SQL (Structured Query Language)** – A programming language used for managing relational databases.
- **SSL/TLS (Secure Sockets Layer/Transport Layer Security)** – Encryption protocols that secure communication between the client and server.

U

- **UI/UX (User Interface/User Experience)** – The design and usability aspects of the application that determine how users interact with the system.

W

- **WebSockets** – A protocol used for real-time communication between the server and the client.

9. References

[1] L. Oleshchenko, “CARTOGRAPHIC SOCIAL NETWORK SOFTWARE DEVELOPMENT FOR INTERNET USER EVENTS SEARCHING”, *ResearchGate,* Jan, 2023. [Online]. Available: https://www.researchgate.net/publication/376673562_CARTOGRAPHIC_SOCIAL_NETWORK_SOFTWARE_DEVELOPMENT_FOR_INTERNET_USER_EVENTS_SEARCHING. [Accessed: Dec 15, 2024].

[2] “Event Management Software”, *Swoogo*. [Online]. Available: https://swoogo.events/. [Accessed: Nov 21, 2024].

[3] “Turning Audiences Into Communities”, *River*. [Online]. Available: https://www.getriver.io/. [Accessed: Nov 21, 2024].