CSE 5325, Team 2

EVENT MANAGEMENT SYSTEM

Team Members:

Ashwin Athappan Karuppan Chetty (1002248214)
Harshwardhan Patil (1002224144)
Anirudh Kashyap Ramesh (1002216351)

INDEX

1. Summary

This document outlines a proposal for the development of a comprehensive Event Management System. The platform is designed to streamline every aspect of event organization, from initial planning and promotion to post-event analysis, with a focus on delivering a core, robust system within a two-month timeframe.

The system will provide a centralized solution for administrators to create and manage events, for users to register and attend, and for speakers to engage with the event. This proposal details the project's objectives, a focused scope, functional and non-functional requirements, and a simplified technical architecture.

2. Project Objectives

- **Streamline Event Management:** To create a unified platform that simplifies the entire event lifecycle for administrators.
- **Enhance User Experience:** To provide a seamless and intuitive registration, ticketing, and attendance process for users.
- **Empower Speakers:** To offer a dedicated portal for speakers to manage their participation, upload materials, and communicate effectively.
- **Data-Driven Insights:** To provide robust reporting and analytics tools for administrators to measure event success and gather valuable feedback.

3. Scope & Features

This section details the core features, supplemented with functional and non-functional requirements to ensure a robust and scalable system.

3.1. Core Features (Client Requirements)

- **Event Creation & Management:** A comprehensive admin dashboard to create, update, and manage all event details, including descriptions, dates, venues, and categories.
- **Online Registration System:** A user-friendly interface for attendees to browse events, register, and manage their bookings.
- **Speaker Portal:** A dedicated portal for speakers to accept invitations, manage their profiles, upload presentation slides, and communicate with

event organizers. It will include a dashboard showing attendee counts for their upcoming sessions and a history of their past events on the platform.

- **Digital Ticketing & Validation:** Automated generation of digital tickets with unique QR codes. The system will support a scanning mechanism via web interface to validate ticket authenticity and manage attendee check-ins at the venue.
- **Schedule Management:** Tools for creating and managing detailed event schedules, including multiple tracks, sessions, and speaker assignments.
- **Attendee Tracking:** A system for real-time attendance monitoring and check-ins, powered by the QR code scanning process.
- **Automated Notifications:** An automated system to send email or push notifications regarding event updates, reminders, and schedule changes.
- **Feedback Collection:** A mechanism for attendees to submit post-event feedback and ratings through integrated forms.
- **Reporting & Analytics:** Generation of detailed reports on event attendance, registration numbers, and attendee feedback.

3.2. Functional Requirements

- **User Roles & Permissions:**
- **Admin:** Full access to create, manage, and delete events, users, and speakers. Can view all reports and analytics.
- **User/Attendee:** Can browse events, register, view their tickets, manage their profile, and submit feedback.
- **Speaker:** Can manage their profile, view assigned sessions, upload materials, and communicate with the admin.

3.3. Non-Functional Requirements

- **Security:** The system must implement robust security measures, including data encryption, secure authentication (e.g., JWT), and protection against common web vulnerabilities (XSS [cross-site-scripting], CSRF [cross-site-request-forgery]).
- **Scalability:** The architecture must be designed to handle a moderate volume of concurrent users and data. (Suggested: Microservice Architecture)
- **Performance:** The application must be responsive, with fast page load times and minimal latency in processing user requests.
- **Reliability:** The system should be stable and include mechanisms for data backup and recovery.

- **Usability:** The user interface for all user roles must be intuitive, accessible, and provide a seamless experience across devices (responsive design).

3.4. Core Microservices Breakdown

Here is a breakdown of the essential services, their responsibilities, and how they map to your feature list.

1. Auth Service (or Identity Service)

This service is the gatekeeper for your entire system. It handles who the users are and what they are allowed to do.

- **Core Responsibilities:**
  - ➢ User registration and login (for attendees, speakers, and admins).
  - ➢ Manages user profiles, roles (Attendee, Speaker, Admin), and permissions.
  - ➢ Issues, validates, and refreshes access tokens (e.g., JWT), as mentioned in your security requirements.
  - ➢ Handles third-party authentication (OAuth) if needed.

- **Maps to Requirements:** *User Roles & Permissions*, *Security (JWT)*.

2. Event Service

This service is the source of truth for all information about the events themselves, excluding the detailed schedule.

- **Core Responsibilities:**
  - ➢ CRUD (Create, Read, Update, Delete) operations for events.
  - ➢ Manages core event details: name, description, date, venue, category, banner images, etc.
  - ➢ Handles event status (e.g., draft, published, cancelled).
- **Maps to Requirements:** *Event Creation & Management*.

3. Schedule Service

While event details are high-level, the schedule is complex enough to warrant its own service. It manages the detailed agenda of an event.

- **Core Responsibilities:**
  - ➢ Manages tracks, sessions, and workshops within an event.
  - ➢ Assigns speakers to specific sessions.
  - ➢ Handles session timing and location (e.g., Room A, Main Hall).
- **Maps to Requirements:** *Schedule Management*.

4. Booking Service (or Registration Service)

This service manages the relationship between attendees and events.

- **Core Responsibilities:**
  - Handles the entire registration flow for an attendee to an event.
  - Manages booking status (e.g., pending, confirmed, cancelled).
  - Keeps track of registration numbers and capacity limits.
  - Publishes an event (e.g., BookingConfirmed) when a registration is successful, which other services will listen to.
- **Maps to Requirements:** *Online Registration System*.

5. Ticket Service

This service is responsible for the tangible proof of registration and is triggered after a successful booking.

- **Core Responsibilities:**
  - Generates a unique digital ticket (with a QR code) upon receiving a BookingConfirmed event.
  - Associates tickets with specific users and events.
  - Provides an endpoint to validate a ticket's authenticity during check-in (the QR code scan).
  - Manages check-in status for attendees.
- **Maps to Requirements:** *Digital Ticketing & Validation*, *Attendee Tracking*.

6. Speaker Service

While speakers are users, their specific needs (profile management, material uploads) justify a dedicated service to keep the Auth Service lean.

- **Core Responsibilities:**
  - Manages detailed speaker profiles (bio, photo, social links).
  - Handles the workflow for session invitations (accept/decline).
  - Manages uploaded materials like presentation slides or documents.
- **Maps to Requirements:** *Speaker Portal*.

7. Notification Service

This is a classic cross-cutting concern, perfect for a dedicated service. It listens for events from all other services and sends communications.

- **Core Responsibilities:**
  - Sends emails, push notifications, or SMS messages.
  - Listens to events like BookingConfirmed, EventUpdated, ScheduleChanged, NewFeedbackRequest.
  - Manages notification templates.
- **Maps to Requirements:** *Automated Notifications*.

8. Feedback Service

This service decouples the feedback collection process from the core event flow.

- **Core Responsibilities:**
  - Creates and manages feedback forms.
  - Collects and stores ratings and comments from attendees for events or specific sessions.
  - Provides data for the Analytics Service.
- **Maps to Requirements:** *Feedback Collection*.

9. Analytics Service

This service is responsible for aggregating data and generating insights. It should be designed for read-heavy workloads and should not be part of the primary transactional flow.

- **Core Responsibilities:**
  - Ingests events from other services (e.g., UserRegistered, AttendeeCheckedIn, FeedbackSubmitted).
  - Aggregates data to generate reports on attendance, registration numbers, and feedback summaries.
  - Provides endpoints for the admin dashboard to visualize analytics.
- **Maps to Requirements:** *Reporting & Analytics*.

High-Level Architecture & Communication

To make this work, you'll need two key patterns that I see you've used in your e-commerce project:

1 **API Gateway**: A single entry point for all client requests. It routes traffic to the appropriate downstream service, handles cross-cutting concerns like rate limiting, and can orchestrate calls. This is where you would enforce authentication by validating JWTs passed from the client.

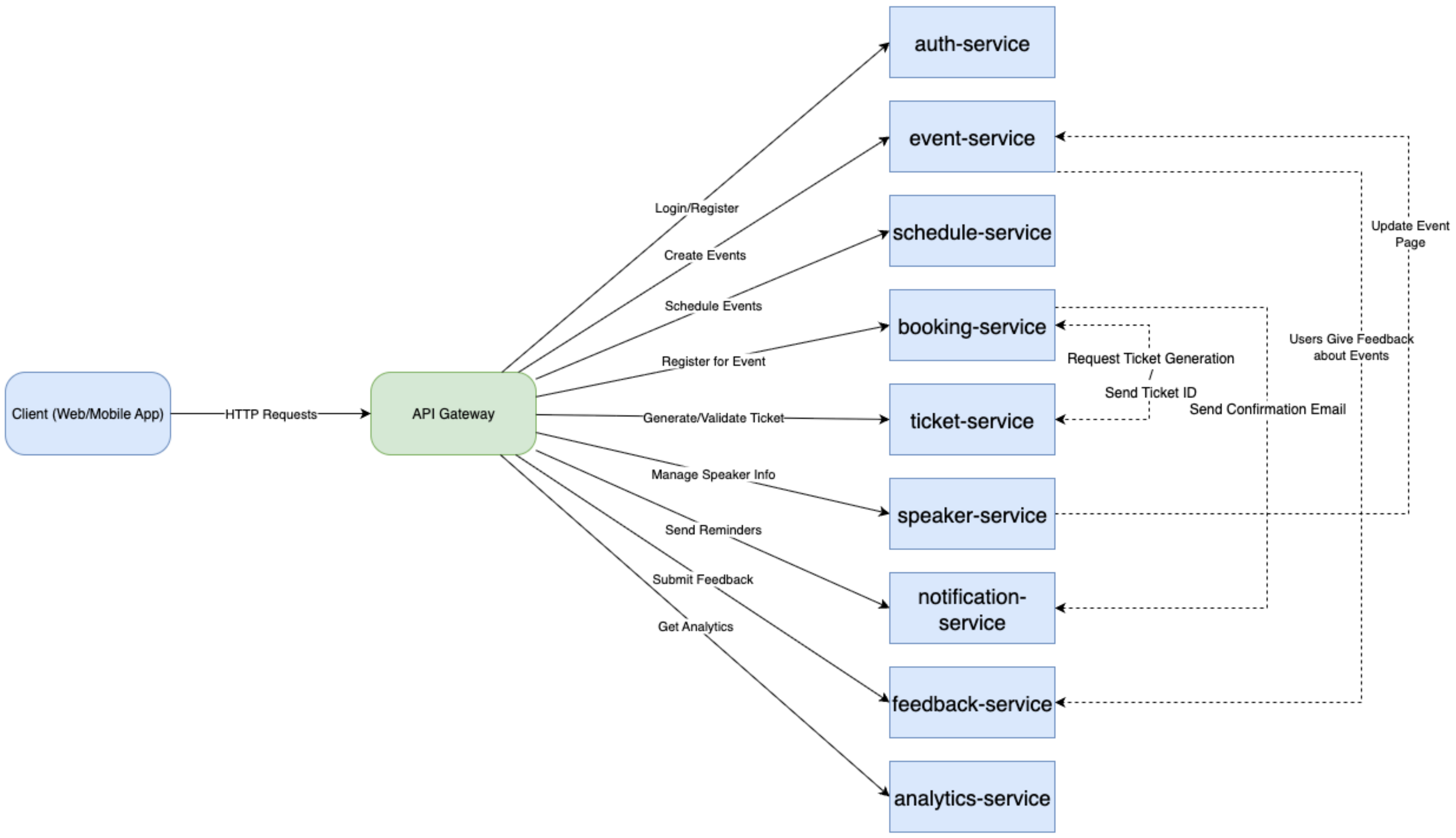
2 **Asynchronous Communication (Event Bus)**: For decoupling, services should communicate asynchronously wherever possible using a message broker (like RabbitMQ, Kafka, or Google Pub/Sub).

Example Flow:

1. A user registers for an event via the **API Gateway**, which routes the request to the **Booking Service**.
2. The **Booking Service** validates the request and creates a booking.
3. It then publishes a BookingConfirmed event to the message bus.
4. Both the **Ticket Service** and the **Notification Service** are subscribed to this event.
5. The **Ticket Service** consumes the event and generates a QR code.
6. The **Notification Service** consumes the event and sends a confirmation email to the user with their ticket details.

This decoupled, event-driven approach ensures that if the Notification Service is temporarily down, the user still gets their ticket, and the notification can be sent once the service recovers. This design directly addresses your non-functional requirements for a **robust and scalable system**.

Below is a high-level view of how the services are related to one another.



4. System Architecture

The proposed system will be built using a modern, 3-tier architecture composed of the following components.

4.1. Frontend

- Description: A responsive and interactive web application where users, speakers, and admins will interact with the system.
- Technology Stack: Next.js with TypeScript. This choice provides server-side rendering for performance and SEO, a rich ecosystem of libraries, and strong typing for maintainability.

4.2. Backend

- Description: A robust API server that will handle business logic, manage data flow between the frontend and the database, and handle user authentication and authorization.

- Technology Stack: Nest.js with TypeScript. This choice provides easy connection between the frontend and the database.

4.3. Database

- Description: The central repository for all application data, including user information, event details, registrations, and schedules.
- Database: PostgreSQL.
- Justification:
    - Relational Integrity: An event management system has highly structured and relational data. PostgreSQL's relational nature and support for ACID transactions ensure data consistency and integrity.
    - Scalability: It is highly scalable and can handle large volumes of data and complex queries efficiently.
    - Ecosystem: It has a mature ecosystem with excellent support in various programming languages and frameworks, including Nest.js.

5. Project Resource Planning

Tech Stack: Next.js, JavaScript/TypeScript, Nest.js, Express.js, Deno, Prisma ORM, Postgres DB,

Project Management Tool: JIRA

Source Control Management: Git

Source Code Repository: GitHub (https://github.com/Buffden/Event-Management-System)

Team Strength: 3

Estimated Completing Time:

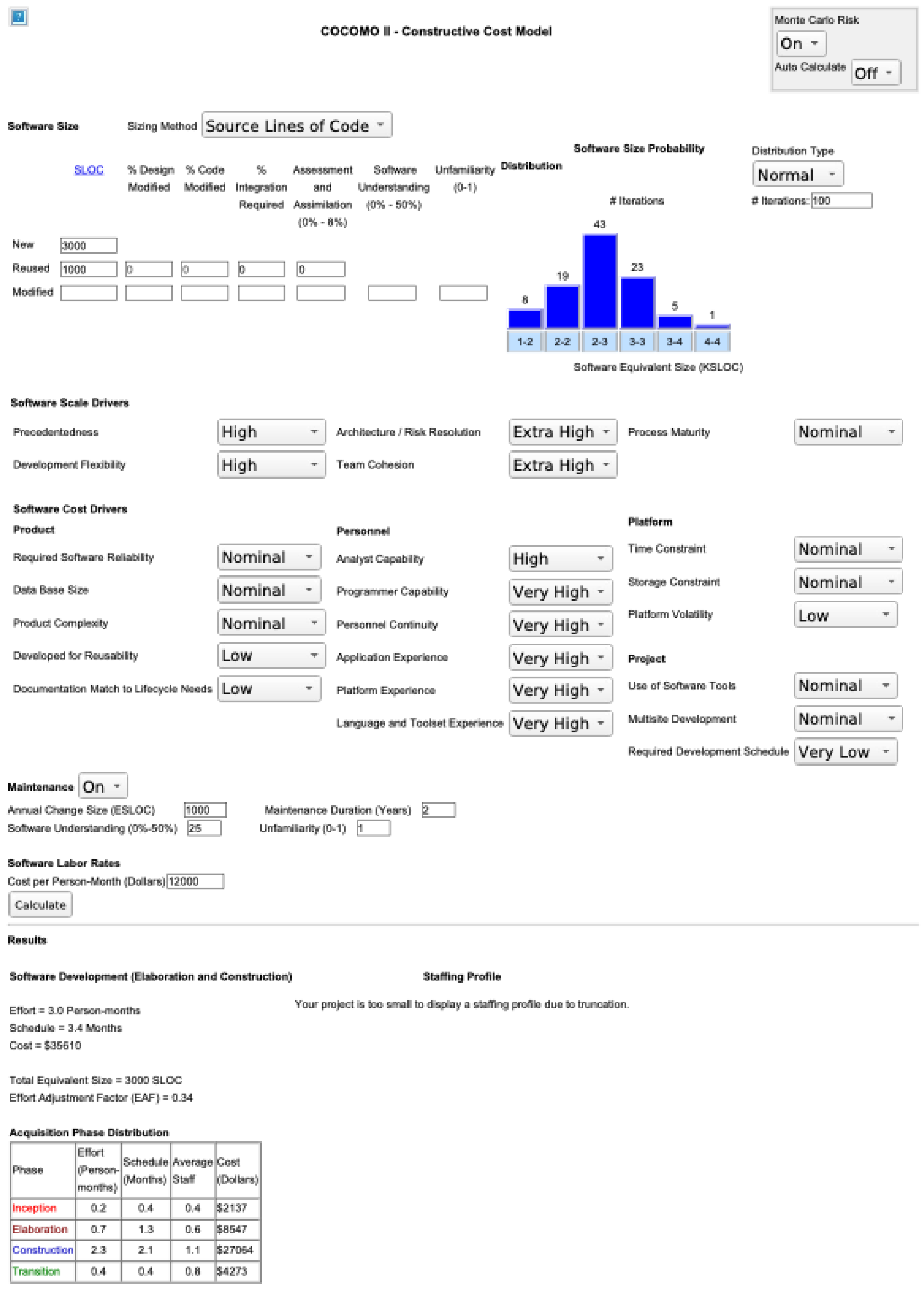
Skill Sets:

Harshwardhan Patil: Next.js, Nodejs, TypeScript, JavaScript, Postgres SQL, Docker, Software Design Patterns, Microservices, Jenkins, UML, Git, Jira

Ashwin Athappan Karuppan Chetty: Next.js, JavaScript/TypeScript, Postgres SQL, Docker, Prisma ORM, Git, JIRA,

Anirudh Kashyap Ramesh: Python, JavaScript, React, Node.js, php, Postgres SQL, docker

Cost Estimation:



Software Size Probability Distribution
# Iterations
8
19
43
23
5
1
1-2
2-2
2-3
3-3
3-4
4-4
Software Equivalent Size (KSLOC)

COCOMO II - Constructive Cost Model

Monte Carlo Risk: On
Auto Calculate: Off

Software Size Sizing Method: Source Lines of Code

Distribution Type: Normal
Iterations: 100

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	3000						
Reused	1000	0	0	0	0		
Modified							

Software Scale Drivers

Precedentedness	High	Architecture / Risk Resolution	Extra High	Process Maturity	Nominal
Development Flexibility	High	Team Cohesion	Extra High		

Software Cost Drivers

Product

Required Software Reliability	Nominal
Data Base Size	Nominal
Product Complexity	Nominal
Developed for Reusability	Low
Documentation Match to Lifecycle Needs	Low

Personnel

Analyst Capability	High
Programmer Capability	Very High
Personnel Continuity	Very High
Application Experience	Very High
Platform Experience	Very High
Language and Toolset Experience	Very High

Platform

Time Constraint	Nominal
Storage Constraint	Nominal
Platform Volatility	Low

Project

Use of Software Tools	Nominal
Multisite Development	Nominal
Required Development Schedule	Very Low

Maintenance On

Annual Change Size (ESLOC): 1000
Maintenance Duration (Years): 2
Software Understanding (0%-50%): 25
Unfamiliarity (0-1): 1

Software Labor Rates

Cost per Person-Month (Dollars): 12000

Calculate

Results

Software Development (Elaboration and Construction)

Effort = 3.0 Person-months
Schedule = 3.4 Months
Cost = $35610

Total Equivalent Size = 3000 SLOC
Effort Adjustment Factor (EAF) = 0.34

Staffing Profile

Your project is too small to display a staffing profile due to truncation.

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.2	0.4	0.4	$2137
Elaboration	0.7	1.3	0.6	$8547
Construction	2.3	2.1	1.1	$27064
Transition	0.4	0.4	0.8	$4273

6. Project Timeline & Deliverables as Epics

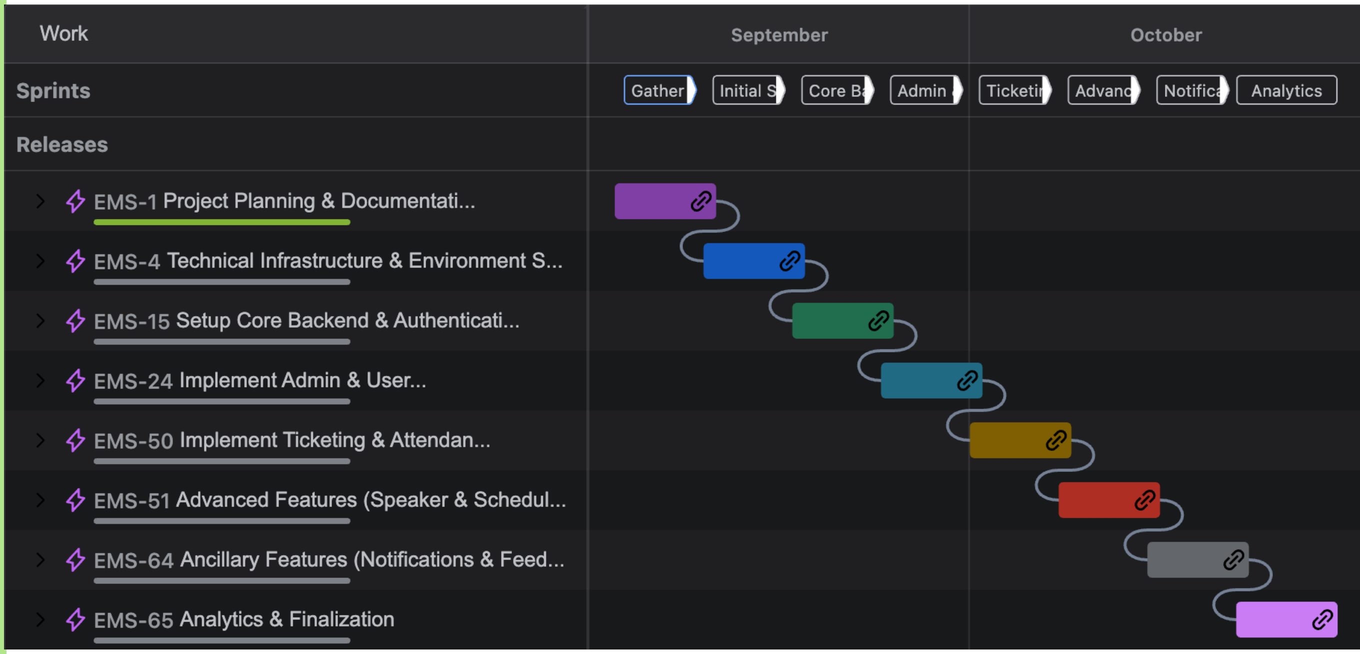
A detailed project plan with specific timelines and milestones will be provided in a subsequent document. High-level phases will include:

- Phase 1: Project Planning & Documentation: This epic covers all foundational planning, requirement finalization, and project management setup tasks needed to ensure the project starts with a clear direction and well-defined goals. Also includes initialization of the project including setup for coordinated development and collaboration.
- Phase 2: Technical Infrastructure & Environment Setup: This epic covers the creation of the project's technical foundation, including code repositories and local development environments.
- Phase 3: Setup Core Backend & Authentication: This epic focuses on building the foundational backend services, including the database models and a secure authentication system.
- Phase 4: Implement Admin & User FE: These epic covers building the essential frontend components for user authentication and the main dashboard for administrators.
- Phase 5: Implement Ticketing & Attendance: This epic focuses on implementing the complete digital ticketing lifecycle, from generation to at-venue validation.
- Phase 6: Advanced Features (Speaker & Schedule): This epic covers the development of specialized portals for speakers and the tools for admins to manage event schedules.
- Phase 7: Ancillary Features (Notifications & Feedback): This epic adds features for automated communication with users and for gathering post-event feedback.
- Phase 8: Analytics & Finalization: This epic is focused on providing data insights to admins and preparing the application for final delivery.

Work Breakdown Structure

EMS-1	Project Planning & Documentation	Done	10-Sep-25	10-Sep-25
EMS-4	Technical Infrastructure & Environment Setup	To Do	10-Sep-25	17-Sep-25
EMS-15	Setup Core Backend & Authentication	To Do	10-Sep-25	24-Sep-25
EMS-24	Implement Admin & User FE	To Do	10-Sep-25	1-Oct-25
EMS-50	Implement Ticketing & Attendance	To Do	10-Sep-25	8-Oct-25
EMS-51	Advanced Features (Speaker & Schedule)	To Do	10-Sep-25	15-Oct-25
EMS-64	Ancillary Features (Notifications & Feedback)	To Do	10-Sep-25	22-Oct-25
EMS-65	Analytics & Finalization	To Do	10-Sep-25	29-Oct-25

Timeline (Gantt Chart):



7. RISK Analysis

TODO: Change Cost Estimation

TODO: Add WBS hierarchy

TODO: Add Individual skills and contributions

TODO: Add Sprint Backlog

8. RISK 2 Analysis