EventOnClick Phase 1

1. Project Profile

1.1 Objectives

Build a web platform for discovering and managing local events (fairs, concerts, meet-ups, festivals, etc.) in any city or state.

Allow authorized event organizers to add, edit, and manage events.

Enable the general public to browse, search, and view event details by location and category.

Apply best software engineering practices, cloud deployment, and scalable architecture.

1.2 Scope

In Scope:

• User Management: Registration and login for event creators and general users.

• Event Management: Authorized users can create, edit, and delete events with details (title, description, date, time, location, category, images, ticket info).

• Event Discovery: Public users can browse, search, and filter events by city/state, date, and category.

• Admin Panel: (Optional) For approving event creators and moderating content.

• Notifications: Email notifications for event creation, updates, and approvals.

• Cloud Hosting: Application accessible via web browser, hosted on a cloud platform.

Out of Scope:

• Payment processing for ticketing.

• Social media integration (for now).

• In-depth analytics/dashboard for event engagement.

2. Target Group

Primary Users:

• Event Creators: Authorized individuals or organizations who add and manage events.

• General Public: Users who browse and attend events.

Secondary Stakeholders:

• Local businesses, cultural organizations, and city councils interested in promoting events.

3. Risks and Mitigation Strategies

3.1 Identified Risks

• Technical Risks: Security vulnerabilities in event creation and user authentication. Scalability issues with high user/event volume.

• Operational Risks: Insufficient event data at launch. Low user adoption in early stages.

• External Risks: Inaccurate or inappropriate event postings.

3.2 Mitigation Strategies

• Security: Use JWT authentication and role-based access control.

• Scalability: Design with cloud-native, scalable backend (Node.js, MongoDB, AWS/GCP).

• Content Moderation: Admin approval for new event creators and event moderation.

• User Engagement: Seed platform with initial events and promote via local channels.

4. Project Plan

Major Milestones:

1. Requirements Analysis (1 week): Define all functional and non-functional requirements.

2. System Design (2-3 days): Create architecture diagrams and select tech stack.

3. Prototype Development (3-4 weeks):

   a. Backend and database setup (2 weeks)

   b. Frontend interface and user flows (2 weeks)

4. Testing Phase (1 week): Unit, integration, and user acceptance testing.

5. Final Documentation (1 week): Prepare final reports and system documentation.

Estimated Timeline: 5-6 weeks

5. Project Organization

• Project Lead: Oversees planning, progress, and communication.

• Backend Developer: Implements server-side logic, APIs, and database.

• Frontend Developer: Designs and builds the user interface.

• Cloud/DevOps Engineer: Manages deployment and cloud infrastructure.

• Tester: Validates features and ensures quality.

6. Functional and Non-Functional Requirements

6.1 Functional Requirements

1. User registration and login (event creators and general users).

2. Event creation, editing, and deletion (authorized users only).

3. Event browsing, searching, and filtering (all users).

4. Admin approval for new event creators and event moderation.

5. Email notifications for event status updates.

6. Public event details page with all relevant information.

6.2 Non-Functional Requirements

1. Performance: Fast response times for browsing and searching events.

2. Scalability: Support for growing number of users and events.

3. Usability: Clean, intuitive interface for all user types.

4. Reliability: High uptime and robust error handling.

5. Security: Encrypted passwords, secure authentication, HTTPS.

6. Portability: Accessible on desktop, tablet, and mobile browsers.

7. Software Development Methodology

Development Methodology: Agile Scrum

Scrum Events:

• Sprint Planning: Bi-weekly, define sprint goals and backlog.

• Daily Stand-Up: 15 minutes, sync tasks and resolve blockers.

• Sprint Review: End of sprint, demo completed features.

• Sprint Retrospective: End of sprint, analyze and improve process.

Sprint Structure: Sprint Duration: 2 weeks. Phases: MVP, Feature Enhancements, System Hardening.

8. Technologies

Service	Purpose
Frontend	React.js – Responsive user interface
Backend	Node.js/Express – API and business logic
Database	MongoDB – Event and user data storage
Version Control	GitHub – Source code management
Tools	Docker, Postman, Trello – Dev & testing
Cloud Provider	AWS or Google Cloud – Hosting and scaling
Auth	JWT – Secure authentication

9. System Design

Architecture Overview:

• Frontend: React SPA, communicates with backend via REST API.

• Backend: Node.js/Express, handles business logic and data access.

• Database: MongoDB for storing users and events.

• Authentication: JWT-based, role-based access for creators/admins.

• Cloud Hosting: Deployed on AWS/GCP for scalability and reliability.

UML/Component Diagram:

• Actors: Event Creator, General User, Admin

• Core Components: User Management, Event Management, Notification Service

• Data Flow: User ↔ Frontend ↔ Backend API ↔ Database