EventOnClick Phase 1 Conception

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 (see functional requirements).

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

• Admin Panel: 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 (Scrum-based)

4.1 Scrum Roles

• Product Owner: Responsible for the product vision, backlog prioritization, and stakeholder communication.

• Scrum Master: Facilitates Scrum ceremonies, removes impediments, and ensures adherence to Scrum practices.

• Development Team: Cross-functional team responsible for design, development, testing, and deployment.

4.2 Sprint Planning

Sprint	Duration	Goals / Product Backlog Items
Sprint 1	2 weeks	Project setup, user authentication (register/login), basic UI skeleton, database schema, glossary draft
Sprint 2	2 weeks	Event creation/editing/deletion (backend & frontend), event data model (title, description, date, time, location, category, image, ticket info), admin approval workflow
Sprint 3	2 weeks	Event browsing/search/filter, event details page, notifications, user roles, initial testing
Sprint 4	2 weeks	Admin panel, error handling, non-functional improvements (performance, security), documentation, UML

		component diagram, glossary finalization, testing suite

4.3 Product Backlog (Sample Items)

• As a user, I can register and log in.

• As an event creator, I can create, edit, and delete events with all required information.

• As a public user, I can browse and search for events by city, state, date, and category.

• As an admin, I can approve or reject new event creators and moderate events.

• As a user, I receive email notifications for event updates.

5. Project Organization

• Product Owner: Dhruv

• Scrum Master: Dhruv

• Development Team: Backend Developer, Frontend Developer, Tester, DevOps/Cloud Engineer

6. Functional and Non-Functional Requirements

6.1 Functional Requirements (Specific)

1. User Management: Registration, login, and role assignment (event creator, public user, admin).

2. Event Management:

- Fields: Title, Description, Date, Time, City, State, Category, Image, Ticket Info (URL/price), Organizer.

- CRUD operations for authorized users.

3. Event Discovery:

- Browse, search, and filter events by city, state, date, and category.

- View event details.

4. Admin Panel: Approve/reject event creators, moderate events.

5. Notifications: Email notifications for event creation, approval, and updates.

6. Public Event Page: Shareable event details page.

6.2 Non-Functional Requirements (Measurable)

1. API Response Time: 95% of requests under 500ms.
2. Database Query Time: Complex searches under 200ms.
3. Image Loading: Optimized images load within 2 seconds.
4. Concurrent Users: Supports 500+ concurrent users.

7. Technology Stack (with Rationale)

Service	Purpose	Rationale
Frontend	React.js – Responsive user interface	Popular, component-based, fast development, large ecosystem
Backend	Node.js/Express – API and business logic	Non-blocking, scalable, JavaScript end-to-end
Database	MongoDB – Event and user data storage	Flexible schema, easy to scale, JSON-like documents
Version Control	GitHub – Source code management	Collaboration, versioning, portfolio visibility
Tools	Docker, Postman, Trello – Dev & testing	Standard for environment, API testing, and agile planning

Cloud Provider	AWS or Google Cloud – Hosting and scaling	Reliable, scalable, global reach
Auth	JWT – Secure authentication	Stateless, widely supported, secure

8. Testing Approach

• Unit Testing: For backend API endpoints and frontend components.

• Integration Testing: End-to-end flows (e.g., event creation to public display).

• Manual Testing: Usability and exploratory testing by team members.

• Performance Testing: Simulate concurrent users and measure response times.

• Acceptance Testing: Validate user stories and requirements.

9. Glossary

• Event Creator: Authorized user who can add/manage events.

• Event Viewer: General public user.

• Admin: User who approves event creators and moderates content.

• Product Owner: Scrum role responsible for product vision and backlog.

• Scrum Master: Scrum role facilitating the process.

• Sprint: Time-boxed development cycle in Scrum.

• Backlog: Prioritized list of features and tasks.

• JWT: JSON Web Token, used for secure authentication.

10. 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: (To be created in the next phase; will show actors, main components, and data flow.)