EventOnClick Phase 2 Development

1. Updated Requirements

1.1 Refined Functional Requirements

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.

1.2 Updated Non-Functional Requirements

• API Response Time: 95% of requests under 500ms.

• Database Query Time: Complex searches under 200ms.

• Image Loading: Optimized images load within 2 seconds.

• Concurrent Users: Supports 500+ concurrent users.

2. Project Plan (Scrum-based)

2.1 Scrum Roles

• Product Owner: Responsible for 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.

2.2 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

2.3 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.

3. Implementation Overview

3.1 Objectives of This Phase

• Begin implementation of the EventOnClick platform as per the refined requirements.

• Set up all frameworks, tools, and version control.

• Ensure the application is maintainable, testable, and ready for further development.

4. Frameworks, Tools, and Version Control

Frontend:

• React.js (Create React App) for a dynamic, responsive user interface.

• React Router for navigation.

• Axios for API communication.

Backend:

• Node.js with Express.js for scalable API development.

• Mongoose for MongoDB object modeling.

• JWT and bcryptjs for authentication and security.

Database:

• MongoDB Atlas for cloud-based, scalable data storage.

Version Control:

• GitHub for source control, using feature-branch workflow and protected main branch.

Other Tools:

• Docker for consistent development and deployment environments.

• Postman for API testing.

• Trello for agile task management.

• Cloudinary for image upload and optimization.

5. Third-Party Libraries

Library/Tool	Purpose	Rationale
React.js	Frontend framework	Modern, component-based UI, large ecosystem
React Router	Client-side routing	Standard for React SPAs, enables seamless navigation
Axios	HTTP client	Promise-based, easy error handling
Node.js	Backend runtime	Non-blocking, scalable, JavaScript end-to-end
Express.js	Backend framework	Minimal, flexible, widely used for REST APIs
Mongoose	MongoDB ODM	Schema validation, query building
MongoDB Atlas	Cloud database	Scalable, managed NoSQL database
bcryptjs	Password hashing	Secure, industry standard
jsonwebtoken	JWT handling	Secure, stateless authentication
multer	File upload	Handles multipart/form-data
nodemailer	Email service	For notifications and verification
joi	Input validation	Ensures data integrity
helmet	Security headers	Secures Express apps
dotenv	Environment variables	Configuration management
jest	Testing framework	Unit/integration testing
socket.io	Real-time communication	For notifications

cloudinary	Image management	Cloud-based, fast delivery
Docker	Containerization	Consistent dev/deployment environments
Postman	API testing	Manual and automated API tests
Trello	Agile task management	Visual project management
GitHub	Version control	Source control, collaboration, CI/CD

6. Implementation & Design Decisions

• Authentication: JWT-based, stateless, scalable, with role-based access.

• Database: MongoDB for flexible event data, Mongoose for schema validation.

• State Management: React Context API for simplicity and performance.

• File Storage: Cloudinary for optimized, fast image delivery.

• API Design: RESTful, consistent error handling and response structure.

7. Architecture Documentation

System Architecture:

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

• Backend: Node.js/Express, business logic, and API endpoints.

• Database: MongoDB for users, events, categories, notifications.

• Authentication: JWT, role-based access.

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

Component Overview:

• Frontend: App.js, Header/Footer, EventCard, EventForm, EventList, AuthComponents, AdminPanel.

• Backend: Auth Service, Event Service, User Service, Notification Service, File Upload Service.

Database Schema:

• Users: _id, email, password, role, profile, createdAt, isVerified

• Events: _id, title, description, date, time, location{city, state}, category, imageUrl, organizer, status, createdAt

• Categories: _id, name, description, icon• Notifications: _id, userId, message, type, read, createdAt

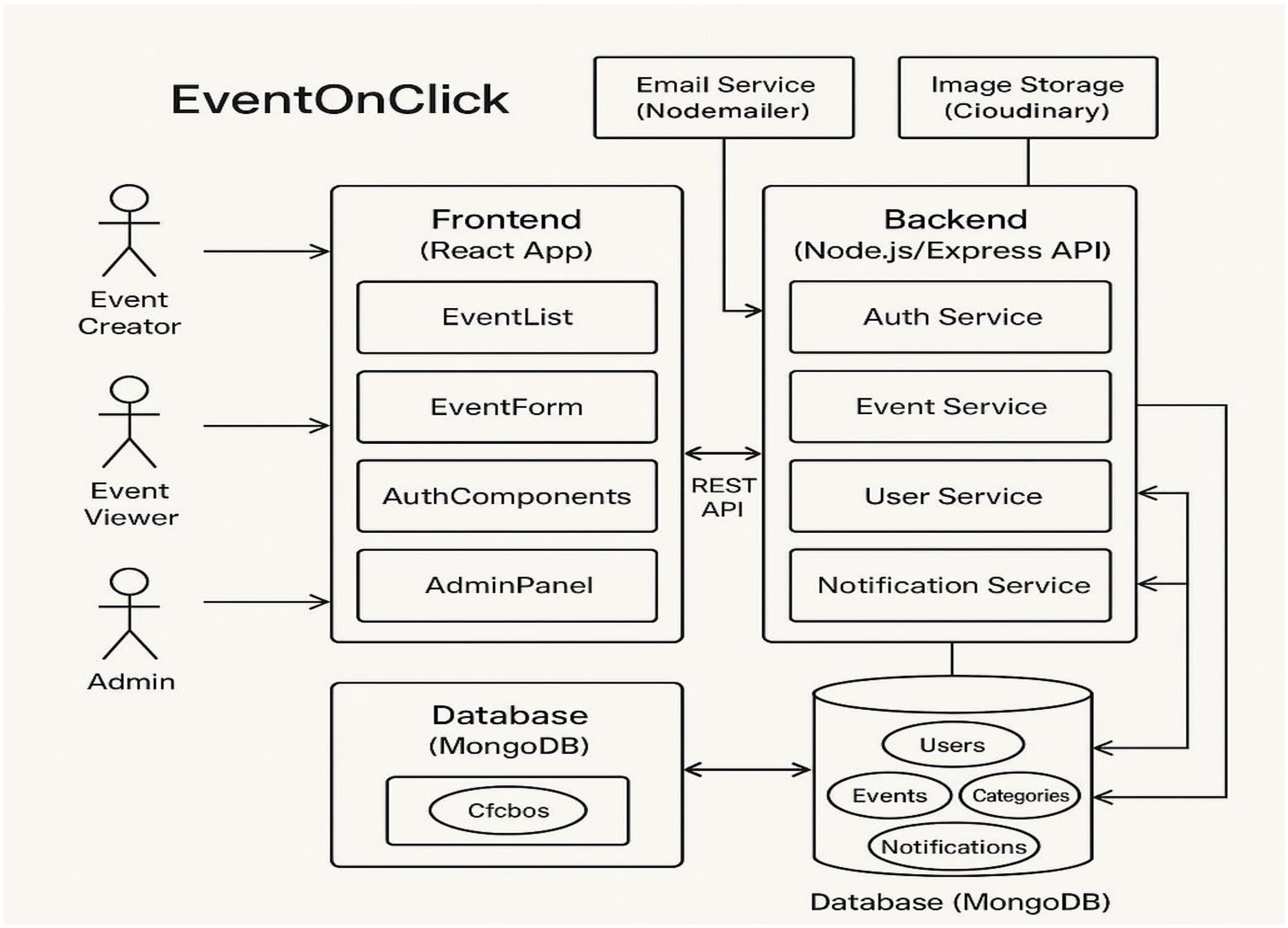

Figure 6.1 Architecture

8. Testing

• Test Data: Seed scripts for users, events, categories; edge cases for validation.

• Testing Levels:

- Unit tests (backend functions, validation logic)

- Integration tests (API endpoints)

- Component tests (React components)

- End-to-end tests (user journeys)

• Coverage: 85% backend, 70% frontend.

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. Documentation & Release

• API Documentation: Swagger/OpenAPI.

• Code Documentation: JSDoc comments.

• README: Installation, setup, deployment.

• Diagrams: Updated component and deployment diagrams.

• User Guide: For event creators and attendees.

• Version Control: Feature-branch workflow, semantic versioning, GitHub Actions for CI/CD.