PlanItRight - Event Management System

Functional and Non-Functional Requirements Document

Prepared By:
Vivekanand. R
Kiran Kumar Reddy. R

September 3, 2024

Contents

1 Introduction

1.1 Project Overview

The PlanItRight Event Management System is a comprehensive platform designed to simplify and streamline the process of planning, managing, and executing events. The system supports a wide range of event-related activities, including event creation, guest management, task assignment, vendor coordination, and budget tracking. It is designed to meet the needs of event planners, managers, and participants, ensuring a smooth and efficient event experience.

1.2 Purpose of the Document

This document aims to provide a detailed description of the functional and non-functional requirements for the PlanItRight Event Management System. It serves as a guide for developers, testers, and stakeholders to ensure that the system meets all specified needs and performs reliably under expected conditions.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Developers: To understand the functionalities that need to be implemented and the non-functional expectations.
- Testers: To create test cases based on the requirements outlined in this document.
- Project Managers: To track the project's progress and ensure all requirements are being met.
- Stakeholders: To confirm that the system's design aligns with business objectives.

2 Functional Requirements

2.1 Overview

The functional requirements are organized based on the major components of the PlanItRight Event Management System, detailing the specific actions and operations that the system must perform.

2.2 Event Management

1. **Event Creation:** The system shall allow users to create events by specifying the following mandatory details:
    - Event Name
    - Date and Time
    - Location (with address validation)
    - Event Description (including optional attachments)

- Event Category (e.g., Wedding, Conference, Party)

2. **Event Update:** Users shall be able to update the event details after creation. The system should log all changes with timestamps for auditing purposes.

3. **Event Deletion:** Users shall be able to delete events. Deleted events should be archived for auditing but not visible in the active events list.

4. **Event Dashboard:** The system shall provide a real-time dashboard for event organizers to monitor:

   - Upcoming Events
   - Past Events (with summaries)
   - Event Status (Active, Completed, Canceled)

5. **Notifications:** The system shall notify users via email and in-app alerts when:

   - An event is updated or canceled.
   - Important deadlines (e.g., RSVP deadlines) are approaching.

2.3 Guest Management

1. **Guest List Management:** The system shall allow users to import guest lists from CSV, Excel, or through manual entry. The system should validate email addresses and phone numbers during import.

2. **Invitation Sending:** The system shall allow users to send invitations via email, with support for customizable invitation templates.

3. **RSVP Tracking:** Guests shall be able to RSVP directly from their email invitation. The system should automatically update RSVP statuses in the database.

4. **Guest Communication:** The system shall enable users to send follow-up messages and reminders to guests who have not responded or to communicate updates about the event.

5. **Guest Grouping:** The system shall allow users to group guests (e.g., Family, VIPs) and manage these groups for invitation and seating purposes.

6. **Seating Arrangement:** The system should provide an interface to create seating arrangements, considering guest preferences and groupings.

2.4 Task Management

1. **Task Assignment:** The system shall allow event organizers to create tasks associated with an event and assign them to team members. Tasks should include:

   - Task Name and Description
   - Assigned User(s)
   - Due Date

- Priority Level (High, Medium, Low)

2. **Task Tracking:** The system shall provide a task management dashboard that displays all tasks associated with an event, categorized by status:
   - To Do
   - In Progress
   - Completed
   - Overdue
3. **Notifications and Reminders:** The system shall notify users when tasks are assigned, updated, or when deadlines are approaching. Overdue tasks should trigger automatic alerts to both the assignee and the event organizer.
4. **Task Comments:** The system should allow users to add comments or updates to tasks, facilitating collaboration and status updates.
5. **Task Reporting:** The system shall provide reports on task completion rates, overdue tasks, and overall progress towards event readiness.

2.5 Vendor and Budget Management

1. **Vendor Management:** The system shall allow users to manage vendors by storing information such as:
   - Vendor Name and Contact Information
   - Service Provided (e.g., Catering, Venue)
   - Contract Details (with attachment upload)
   - Payment Terms and Status
2. **Budget Tracking:** The system shall allow users to set a budget for each event and track expenses against this budget. The system should categorize expenses and provide real-time budget status.
3. **Payment Management:** The system shall track payments made to vendors, including due dates and payment status. It should generate alerts for upcoming and overdue payments.
4. **Budget Reporting:** The system shall provide detailed budget reports, including:
   - Total Budget Allocated
   - Total Spent
   - Remaining Budget
   - Expense Breakdown by Category
5. **Expense Forecasting:** The system should allow users to project future expenses based on existing contracts and planned activities.

3 Non-Functional Requirements

3.1 Performance

1. The system should be able to handle up to 10,000 concurrent users with minimal performance degradation.
2. The system shall respond to user actions (e.g., page loads, form submissions) within 2 seconds under normal conditions.
3. The system should efficiently handle large datasets, such as guest lists with over 10,000 entries, without significant delays.

3.2 Scalability

1. The system must be horizontally scalable to handle increasing loads, including more users, events, and data.
2. The database architecture should support efficient scaling to accommodate growing data storage requirements.
3. The system should support the addition of new features or modules without significant changes to the existing infrastructure.

3.3 Reliability

1. The system should maintain a 99.9% uptime, excluding scheduled maintenance.
2. The system should include automatic failover mechanisms to ensure high availability in case of server failures.
3. Regular backups should be performed to ensure data integrity and recovery in case of data loss.

3.4 Security

1. The system should implement JWT (JSON Web Token) authentication for secure and stateless user sessions.
2. Data should be encrypted both at rest and in transit using industry-standard encryption techniques (e.g., AES-256).
3. The system should include role-based access control (RBAC) to manage user permissions and restrict access to sensitive information.
4. The system should protect against common web vulnerabilities, including SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
5. The system should log all user activities and security events for audit and review purposes.
6. Regular security audits and penetration testing should be conducted to identify and mitigate potential risks.

3.5 Usability

1. The user interface (UI) should be intuitive and accessible, allowing users with varying technical expertise to navigate the system easily.
2. The system should provide a responsive design, ensuring optimal usability on both desktop and mobile devices.
3. The system should support multiple languages, allowing users to switch between languages as needed.
4. The system should include help documentation and tooltips to assist users in performing tasks.
5. The system should comply with accessibility standards, such as WCAG 2.1, ensuring usability for people with disabilities.

3.6 Maintainability

1. The system's codebase should be well-documented and follow best practices for coding standards to facilitate maintenance and future development.
2. The system should be modular, allowing individual components to be updated or replaced without impacting the overall system.
3. The system should support continuous integration and deployment (CI/CD) processes to streamline updates and patches.
4. The system should include automated testing suites to ensure that updates do not introduce new bugs or regressions.

3.7 Compliance

1. The system should comply with data protection regulations such as GDPR for handling and storing user data.
2. The system should adhere to industry standards for web application security, such as OWASP Top Ten.
3. The system should support data retention and deletion policies to ensure compliance with legal requirements.
4. The system should provide data export functionality to comply with data portability requirements.

4 Conclusion

This document outlines the comprehensive functional and non-functional requirements for the PlanItRight Event Management System. By adhering to these requirements, the system will provide a robust, scalable, and user-friendly platform capable of meeting the diverse needs of event planners and participants. The detailed specifications ensure that all aspects of system performance, security, usability, and maintainability are addressed, paving the way for a successful implementation.