Software Requirements Specification

Version 1.0

FusionFiesta

Theme: College Event Information System

Category: Cross-Platform App Development

Table of Contents

1.1 Background and Necessity for the Cross-Platform App

In colleges and universities, numerous events such as fests, workshops, technical competitions, cultural functions, and seminars are organized regularly. Managing these events manually often leads to miscommunication, resource mismanagement, scheduling conflicts and poor student engagement.

Technological advancements and the widespread use of smartphones offer a promising solution. Cross-platform applications have potential to revolutionize the way services are requested and delivered. They provide rapid, efficient, and user-friendly alternatives to traditional methods.

With the increasing usage of smartphones among students and faculty, a centralized mobile or cross-platform application is essential to streamline the entire event management lifecycle. This includes event creation, registrations, notifications, scheduling, feedback, and post-event documentation.

In short, a cross-platform application is required that can provide:

- **Centralized Access**: One platform for students, faculty, and organizers to access and manage all event-related information.
- **Real-Time Updates**: Automated notifications and updates eliminate dependency on manual communication.
- **Streamlined Scheduling**: Prevents conflicts and overlaps between events through automated scheduling and resource planning.
- **Improved Engagement**: A user-friendly digital interface encourages greater student participation and awareness.
- **Event History**: Enables users to browse past events and helps organizers analyze participation trends and feedback.
- **Role-Based Features**: Different user interfaces and permissions for students, organizers, and administrators to ensure secure and efficient usage.

1.2 Proposed Solution

A cross-platform app titled ‘**FusionFiesta’** should provide a platform for administrators, event coordinators, and students to manage or participate in college events efficiently. The app should support role-based access and allow real-time event updates, digital registrations, announcements, feedback collection, and more.

1.3 Purpose of this Document

This document provides a comprehensive specification of the **FusionFiesta** cross platform application. It outlines the app’s purpose, target users, functional and non-functional requirements, database design, hardware/software requirements, and expected deliverables. It serves as a reference for developers, testers, and stakeholders.

1.4 Scope of Project

The **FusionFiesta** cross-platform application is designed to serve as a centralized platform for managing college events efficiently and engagingly. It should cater to three main user roles – **students**, **event organizers**, and **administrators**–with distinct features tailored to their requirements.

Students should be able to explore upcoming events across categories such as cultural, technical, sports, and register for events digitally. They should be able to receive timely notifications and download e-certificates after participation (based on certificate fee payment).

Event organizers should use the app to propose, create, update, and manage events. They should also be able to monitor participant data, upload results, assign volunteers, and communicate directly with attendees.

The admin role should include oversight of all event activities, including user management, and approval of proposed events. It should also facilitate generation of participation and feedback reports and moderation of content such as feedback and gallery uploads.

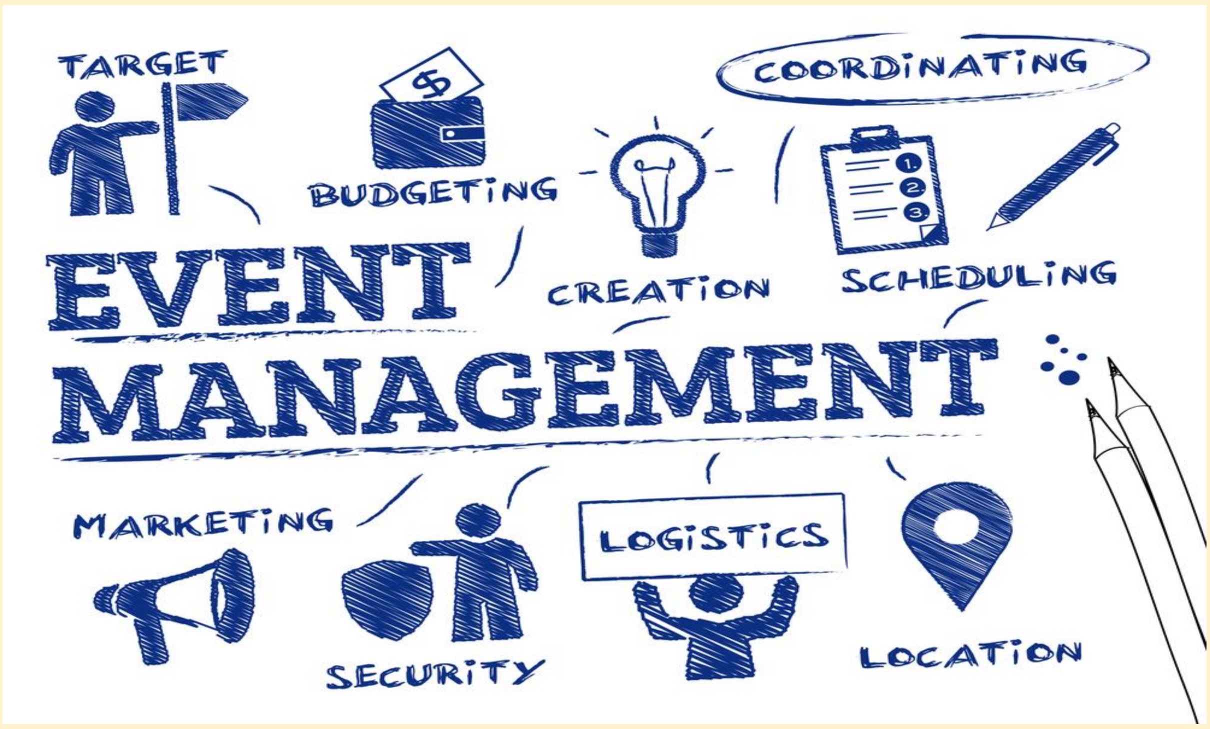
The app aims to digitize the entire event lifecycle–from creation to closure – ensuring greater transparency, accessibility, and efficiency in college event management. It should be a scalable, role-based system capable of supporting real-time updates, secure authentication, and interactive features.

1.5 Constraints

To ensure a smooth and user-friendly experience, the **FusionFiesta** app must meet certain operational constraints. Since the app involves handling event details, user profiles, images, and multimedia content, it is essential that data and high-resolution images are loaded efficiently. This ensures there are negligible delays in screen transitions and improving response time for users across varying network conditions.

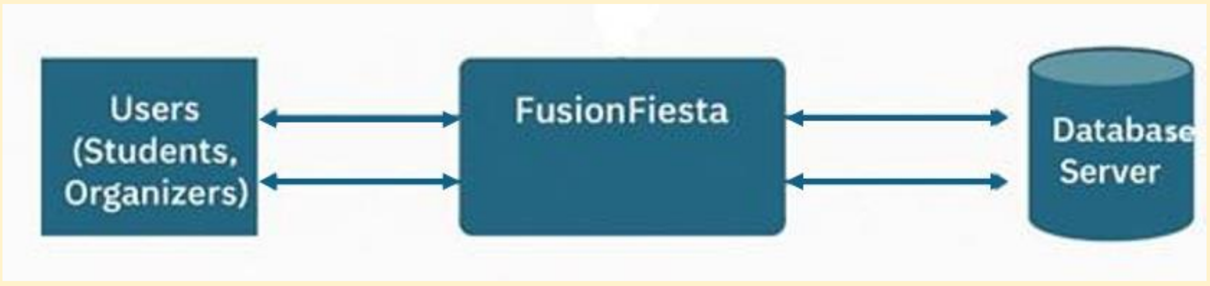
The app should also support real-time synchronization across all user roles – students, organizers, and administrators. For example, if an organizer updates event schedule or uploads new announcements, changes should instantly reflect on student and admin dashboards without requiring manual refresh or re-login. This real-time data consistency is crucial for maintaining transparency and ensuring that all stakeholders are on the same page.

Moreover, user authentication and data privacy must be strictly enforced. Only verified users should be allowed to access the system and role-based access control should prevent unauthorized actions. Sensitive data such as student profiles, event feedback, and certificate records must be securely stored and transmitted using standard encryption. Best practices in mobile/cross-platform security should also be applied.



Architecture Diagram

The broad architecture diagram for the app is depicted as follows:



Various types of users can connect to the **FusionFiesta** cross-platform app which interacts with the database.

1.6 Functional Requirements

The app should be designed as a set of Forms/Pages, Navigation, and Fragments/Screens with menus representing choice of activities to be performed.

Following are the functional requirements of the cross-platform app:

1. **User Registration and Authentication:**

   The **FusionFiesta** app should support a robust user registration and authentication system that accommodates different types of users:
   General visitors (students who only browse)
   Participants (students who register for events)
   Staff members (who act as organizers or administrators)

   - ❖ The registration process should allow users to select their role during sign-up: Student Visitor, Student Participant, or Staff (Organizer/Admin).
   - ❖ A normal student can browse event listings and view gallery content, but cannot register for events without upgrading their role to Participant.
   - ❖ A student participant should provide additional details such as enrolment number, department, and college ID proof. The college credentials cannot be verified, however, as that is beyond the scope of the project. This enables students to register for events and download participation certificates upon payment of certificate fees.

- ❖ Staff members, including organizers and admins, should register using institutional email addresses and must be approved by system admin before accessing event management features.
- ❖ Secure login should be implemented using a username/email and password combination. A ‘Forgot Password’ feature should allow users to reset their password via a secure token sent to their email.

2. Role-Based Dashboard:

The **FusionFiesta** cross-platform app should include a role-based dashboard that customizes its content based on whether the user is a student, organizer, or admin. Each dashboard should display information and tools relevant to the user's role, helping them quickly access what they require.

🧑🎓Student Dashboard

- ❖ The dashboard should show a personalized list of upcoming events that the student can register for.
- ❖ It should highlight the events the student has already registered for and show reminders for them.
- ❖ Notifications should appear for any event updates, announcements, or certificates made available.
- ❖ The student should get quick access to view all events, registered events, certificates (provided they have paid the fees for them), and feedback forms.
- ❖ Students can also bookmark or favorite events they like for future reference.

🧑💼Organizer Dashboard

- ❖ Organizers should see statistics such as the number of events they have created and how many students have registered.
- ❖ The dashboard should show each event’s status such as ‘Live’, ‘Upcoming’, or ‘Completed’.
- ❖ Any new participant queries or messages should be visible on the dashboard.
- ❖ Organizers can quickly create new events, manage ongoing ones, upload results and certificates, and check feedback.
- ❖ A calendar view should help them keep track of scheduled and past events.

🛡️ Admin Dashboard

- ❖ Admins should get an overview of system activity, such as how many events are pending approval.
- ❖ They should see summaries such as how many active users exist and how many events each department has organized.
- ❖ The dashboard should alert admins about flagged content or system-related issues.
- ❖ Admins can quickly approve or reject event proposals, manage user accounts, and download reports.
- ❖ They can also moderate feedback and review gallery uploads to maintain content quality.

🔄 Common Dashboard Features

- ❖ All dashboards should include a notification panel that users can filter by unread status or event relevance.
- ❖ The design of the dashboard should adjust to different screen sizes for mobile compatibility.
- ❖ Elements such as charts, status labels, and activity summaries should make information easier to understand.

3. Event Browsing and Filtering:

- ❖ All users should be able to browse a comprehensive list of events, with filters based on category (technical, cultural, and sports), date, department, or popularity.
- ❖ A search bar with auto-suggestions should help users quickly find events by keywords or tags.
- ❖ Events should be displayed with summary cards including title, image, date/time, location, and available slots.
- ❖ Users should be able to sort events by the newest, most popular, or by the ones they are eligible for.
- ❖ Clicking an event should lead to a detailed page with full description, organizer contact, event rules, and event registration button (if applicable based on availability of slots).

4. **Event Registration System:**

   - Only verified Student Participants should be able to register for events.
   - If the user's profile is complete, the registration should be processed with a single click. If any required profile information is missing, the system should prompt the user to update those fields before proceeding with registration.
   - Participants should receive an on-screen confirmation message or an email upon successful registration.
   - Users can view their registered events, cancel registrations before the deadline, and access QR codes or digital passes if required for entry.
   - Organizers can view, approve, or reject participant registrations (if applicable), and communicate with them through internal messages.

5. **Event Management by Organizers:**
   - Organizers should be able to create new events by filling out a structured form with details such as event title, description, date, time, venue, category, registration limit.
   - The event creation form should also allow uploading a banner image, guidelines document (PDF), and select co-organizers or volunteers.
   - Events should remain in a Pending Approval state until reviewed by an admin. Once approved, they should be visible to all students.
   - Organizers can edit event details before the event start date and may also cancel or reschedule events with admin approval.
   - During the event, organizers can track real-time registration status, view participant lists, and mark attendance via QR-code check-in or other options for check-in.
   - After the event, organizers can upload results, add winner details, and attach e-certificates for approved participants provided they have paid the certificate fees.
   - Organizers can also send announcements or updates to all registered users for their event.

6. **Certificate Management and Downloads:**
   - Organizers should have a section where they can upload digital certificates (PDF) for event participants or winners (subject to payment of requisite fees).
   - Each certificate should be linked to a participant’s name and event ID and optionally validated with a QR code or certificate ID.

- Student participants should be able to download certificates for events they have attended directly from their dashboard.
- Certificates should be permanently stored in the user's account and can be redownloaded at any time.
- Admins should have access to oversee all uploaded certificates and remove or replace them in case of errors.

7. **Admin Operations and Control Panel:**
   - Admins should manage the entire app through a dedicated control panel with access to system-wide data.
   - They should be responsible for approving/rejecting new events, especially ensuring the event details follow college standards.
   - Admins can create, edit, or deactivate user accounts, including assigning or revoking organizer/admin roles.
   - The admin panel should also allow viewing department-wise statistics, such as number of events conducted and student participation rate.
   - Reports can be generated on demand in PDF or Excel formats, showing data such as total events held, certificates issued, and feedback scores.
   - Admins should also receive system alerts for flagged content, multiple failed login attempts, or errors in the app flow.

8. **Media Gallery:**
   - The app should feature a categorized event gallery where users can browse photos and videos from past college events.
   - Organizers should be able to upload high-quality images and short video clips related to their events after completion.
   - Media files can be tagged with event names, dates, categories (for example, cultural or technical) and departments for easy filtering.
   - Students and visitors can browse the gallery, filter by event type or date, and save favorite images/videos to their profile.
   - Admins should be able to moderate uploaded media, remove inappropriate content, and highlight select gallery items on the home page.
   - The gallery section should be visually appealing, lightweight, and responsive, with options to preview, expand, or share content (if enabled).

9. **Feedback and Ratings System:**
   - After attending an event, students should be prompted to provide feedback, which may include a star rating (1 to 5) and an optional text comment.
   - The feedback form should ask students to rate the event on parameters such as organization, relevance, coordination, and overall experience.
   - Submitted feedback should be visible to the organizers, who can use it to assess and improve future events.
   - Admins should have access to summarized feedback reports, including average ratings, most-used keywords, and suggestions by category.
   - A feedback moderation system should be implemented to flag and review inappropriate language or spam.
   - Events with higher feedback scores may be marked as 'Top Rated' and promoted within the app.

10. **Search, Notifications, and Alerts:**

   - A global search bar should allow users to search for events using event names, departments, or keywords.
   - Search results should support autosuggestions, spelling tolerance (fuzzy matching), and relevance ranking.
   - All users should receive real-time push notifications and in-app alerts for new events, announcements, changes, or certificate availability.
   - Notifications should be role-based, so that students receive participant updates, while organizers receive scheduling alerts or registration counts.
   - Admins should receive system alerts for pending approvals, flagged content, or technical issues detected in the back end.
   - Users can view notification history and optionally turn OFF non-critical alerts through profile settings.

11. User Profile Management:

- ❖ All users should have a dedicated profile section where they can view and update their personal details such as name, email, mobile number, and department.
- ❖ Student participants should also see their enrolment number, registered events, certificates, and saved items.
- ❖ Organizers should have a profile that lists all events they have created, ongoing responsibilities, and communication history with participants.
- ❖ Admins should have profile access that includes control options such as managing roles and user privileges.
- ❖ Users should be able to change their passwords, update contact details, and adjust notification preferences through profile settings.
- ❖ A profile picture upload feature should be included for a more personalized user experience.

12. About Us and Contact Us Pages:

- ❖ The app should include an About Us page that explains the purpose of the app, the team behind it, and the benefits for students and staff.
- ❖ The Contact Us section should provide a contact form where users can submit queries, technical issues, or general feedback.
- ❖ This form should collect the user's name, email, subject, and message, and send it to the admin email or database for follow-up.
- ❖ A Google Maps integration should allow users to view the physical location of the institution, with optional GPS navigation if enabled.
- ❖ Frequently Asked Questions (FAQs) may also be added to help users with common issues or app usage guidance.

13. Common Features Across All Roles:

- ❖ A notification bell or panel should appear for all users, showing real-time updates, announcements, and reminders.
- ❖ The app should have a sitemap or onboarding guide to help new users understand how to navigate and use the app features.
- ❖ A responsive layout should ensure smooth usability across devices (phones, tablets), with adaptive components based on screen size.
- ❖ The system should implement role-based access control, ensuring that each user only sees content relevant to their role.

Sitemap: To help users understand the flow of app, you should create a Sitemap and add it to the home screen of your app.

Note: Boilerplate or readymade HTML template can be used, provided it is only for design aspect and not for implementing app functionality. Do NOT copy content or code from GPTs or other AI tools, although you are permitted to use images generated by AI tools for any visual representation purposes. It is mandatory to mention such tools used in case you add any AI generated images.

1.7 Non-Functional Requirements

There are several non-functional requirements that should be fulfilled by the cross-platform app. These include:

Safe to use: The cross-platform app should not result in any malicious downloads or unnecessary file downloads.

Accessibility: The cross-platform app should have clear and legible fonts, user-interface elements, and navigation elements.

User-friendliness: The cross-platform app should be easy to navigate with clear menus and other elements and easy to understand.

Operability: The cross-platform app should be reliable and efficient.

Performance: The cross-platform app should demonstrate high value of performance through speed and throughput. In simple terms, the cross-platform app should have minimal load time and smooth page redirection.

Scalability: The cross-platform app architecture and infrastructure should be designed to handle increasing user traffic, data storage, and feature expansions.

Security: The cross-platform app should implement adequate security measures such as authentication. For example, only registered users can access certain features.

Availability: The cross-platform app should be available 24/7 with minimum downtime.

These are the bare minimum expectations from the project. <u>It is a must to implement the FUNCTIONAL and NON-FUNCTIONAL requirements given in this SRS</u>. Once they are complete, you can use your own creativity and imagination to add more features if required.

Database Design

Based on the given specifications, you should define suitable entities, attributes for these entities, and identify relationships between the entities.

For example, some entities along with their attributes can be identified as follows:

Users

Column Name
UserId (PK)
Email (UNIQUE)
Password
Role
CreatedAt

UserDetails

Column Name
DetailId (PK)
UserId (FK)
FullName
Mobile Number
Department
EnrollmentNo
ProfilePic

Events

Column Name
EventId (PK)
Title
Description
Category
Department
Date
Time
Venue
Status
OrganizerId
MaxParticipants
BannerUrl

Registrations

Column Name
RegistrationId (PK)
EventId (FK)
StudentId (FK)
RegisteredOn
Status

Attendance

Column Name
AttendanceId (PK)
EventId (FK)
StudentId (FK)
Attended
MarkedOn

Feedback

Column Name
FeedbackId (PK)
EventId (PK)
StudentId (FK)
Rating
Comments
SubmittedOn

Certificates

Column Name
CertificateId (PK)
EventId (FK)
StudentId (FK)
CertificateUrl
IssuedOn

MediaGallery

Column Name
MediaId (PK)
EventId (FK)
FileType
FileUrl
UploadedBy (FK)
Caption
UploadedOn

Similarly, you can define other entities and relationships between entities and methods representing activities on the entities.

Note: These are just examples, you do not have to adhere to these structures and can design your own table structure with columns.

1.8 Interface Requirements

Hardware

- Intel Core i5/i7 Processor or higher
- 8 GB RAM or higher
- Color SVGA monitor
- 500 GB Hard Disk space
- Mouse
- Keyboard
- Internet access (4G or Wi-Fi)

Software

Programming Software and IDE:
Android Studio IDE with Android 9 or higher with Java, Flutter 1.2 with Dart 2.6 or higher

Database: SQLite or Firebase

Testing

You can test the cross-platform app using a mobile device (Android Smartphone or iPhone) or a suitable emulator.

1.9 Project Deliverables

You are required to design and build the project and submit it along with complete Documentation that includes:

- Problem Definition
- Design Specifications
- Diagrams such as Flowcharts for various Activities, Data Flow Diagrams, and so on
- Database Design
- Test Data Used in the Project
- Project Installation Instructions
- **User Credentials for all Types of Users with Passwords**

Documentation is considered as a very important part of the project. Ensure that documentation is complete and comprehensive. **Documentation should not contain any source code.**

The consolidated project will be submitted as a Source Code zip file with a ReadMe.doc file listing assumptions (if any) made at your end. It should include SQL scripts files (.sql) containing database and table definitions and Android Package File (.apk file) for the cross-platform app.

Submit a video (.mp4 file) demonstrating the working of the cross-platform app, including all the functionalities of the project. This is MANDATORY.

Over and above the given specifications, you can apply your creativity and logic to improve the cross-platform app.

~~ End of the Document ~~