# Software Design Document (SDD)

## Activity Points Management System

# 1. Introduction

## 1.1 Purpose

The Activity Points Management System is designed to streamline the process of tracking student achievements, certificate validation, and activity participation. It automates the allocation of points based on predefined criteria, ensuring a fair and transparent method for recognizing student engagement. The system reduces administrative burden on faculty and provides students with a clear view of their progress.

## 1.2 Project Scope

The system allows students to submit certificates for validation, faculty to review and approve them, and administrators to oversee student records. It features a secure authentication system, real-time notifications, role-based access control, and analytics to monitor student performance. The system is designed to be scalable, ensuring it can support a growing number of students and events.

## 1.3 Intended Audience

This document is intended for software developers, system architects, faculty administrators, and other stakeholders involved in the development, deployment, and management of the Activity Points Management System. It provides an in-depth look at the architecture, data flow, and functional components of the system.

# 2. System Architecture Design

## 2.1 Design

The system is based on a three-tier architecture model, which ensures modularity and separation of concerns. This architecture enhances scalability, security, and maintainability. The layers are as follows:

* Presentation Layer: A responsive front-end developed using React Vite, enabling students, faculty, and admins to interact with the system efficiently.
* Application Layer: The backend, built using Firebase, handles business logic, certificate validation, real-time updates, and notifications.
* Data Layer: A cloud-based Firestore database that securely stores student records, certificates, event details, and activity points.

## 2.2 Purpose

The structured architecture ensures each component functions independently while allowing smooth interaction between layers. The use of Firebase allows real-time synchronization of data, ensuring students and faculty always have access to the latest updates.

# 3. GUI Design

## 3.1 Login and Registration

The login and registration system provides secure authentication via email and password. The system includes role-based access control, ensuring that only authorized users can perform specific actions. Faculty and administrators have different access privileges compared to students.

## 3.2 Home Screen and Dashboard

The dashboard provides an overview of a student’s current activity points, pending certificate approvals, and event participation history. Faculty members see a list of pending validations, while administrators can monitor system-wide trends.

## 3.3 Event and Certificate Management

Students can submit certificates for faculty validation, apply for duty leave, and register for events. Faculty members can review and approve certificates while tracking student participation. Admins can manage event listings and oversee the validation process.

# 4. API Design

## 4.1 API Endpoints

The system provides RESTful API endpoints to facilitate seamless communication between the frontend and backend. Key API endpoints include:

* POST /auth/login - Handles user authentication and session management.
* GET /students/{id} - Retrieves student profile details and activity points.
* POST /certificates/upload - Uploads student certificates for validation.
* PUT /certificates/validate - Faculty members approve or reject certificates.
* GET /events/upcoming - Fetches a list of upcoming events.
* GET /analytics/performance - Provides real-time student performance analytics.

# 5. Database Design

## 5.1 User Class Diagram

The database follows a structured schema to ensure efficient data retrieval and management. It implements role-based access control to define user permissions.

## 5.2 Key Database Tables

* Students - Stores student details, activity points, and participation history.
* Certificates - Contains uploaded certificates, validation status, and faculty approvals.
* Events - Tracks event details, student participation, and organizer information.
* Users - Defines roles and permissions for students, faculty, and administrators.

# 6. Technology Stack

## 6.1 Programming Languages

* Frontend: JavaScript (React Vite) ensures a fast and interactive user experience.
* Backend: Python (Firebase Functions) enables secure and scalable backend operations.
* Database: Firestore (NoSQL) provides efficient data storage and retrieval.

## 6.2 Frameworks and Libraries

* Frontend: React.js and Tailwind CSS for a responsive UI.
* Backend: Firebase Authentication, Firestore, and Cloud Functions for seamless integration.
* Tools: VS Code for development, Postman for API testing, and Figma for UI design.

# 7. Deployment and Maintenance

## 7.1 Deployment Plan

The system will be deployed using Firebase Hosting, ensuring global availability and performance. Backend services will run on Firebase Cloud Functions, reducing the need for dedicated server management. Regular backups will be scheduled to prevent data loss.

## 7.2 Maintenance and Updates

The system will be monitored continuously for performance and security vulnerabilities. Regular updates will be implemented to introduce new features, enhance security, and optimize system performance. An automated error-tracking system will log and resolve issues promptly.

# 8. Conclusion

The Activity Points Management System provides a structured and efficient method for tracking student achievements. By automating the process of certificate validation and point allocation, the system minimizes manual effort while improving transparency. The modular architecture ensures scalability, allowing for future enhancements and integration with additional student services.