Software Design Specifications

Website to Assign Projects for Final Year Students

Document Information

Title: student application project portal
Project Manager: D. Vijay Rao

Document Version No: 1.0
Document Version Date: 2025-04-07
Prepared By: Shaik Naveed Ahmed
Preparation Date: 2025-04-07

Version History

Ver. No	Ver. Date	Revised By	Description	Filename
1.0	2025-04-07	Shaik Naveed Ahmed	Initial Design Draft	FYP_DesignDoc_v1.0.docx

Table of Contents

1 Introduction:

This document provides an in-depth design specification for the Website to Assign Projects for Final Year Students. It outlines the system architecture, component interactions, data models, use case realizations, and system behaviors.

1.1 Purpose:

The purpose of this document is to serve as a comprehensive guide for developers, testers, and stakeholders. It helps ensure consistent implementation of the project modules such as student project applications, faculty project creation, and admin oversight.

1.2 Scope:

This document applies to the full-stack web application for Mahindra University. It includes role-based access, student-project mapping, real-time project status updates, messaging between stakeholders, and secure data handling.

1.3 Definitions, Acronyms, and Abbreviations:

UI – User Interface

DB – Database

HTTPS – HyperText Transfer Protocol Secure

RBAC – Role-Based Access Control

CRUD – Create, Read, Update, Delete

SRS – Software Requirements Specification

1.4 References:

- IEEE 1016-2009 Software Design Documentation
- Project SRS Document
- Express.js, MySQL, HTML/CSS Documentation
- Mahindra University Auth API Guide

2 Use Case View:

This section describes key use cases such as project posting, student application, admin approval, and notification handling. Use cases are derived from functional requirements.

2.1 Use Case:

Use Case: Post Project

Actor: Faculty

Steps:

1. Login
2. Click 'Post Project'
3. Fill project title, description, and deadline

4. Submit → visible to students

Use Case: Apply for Project

Actor: Student

Steps:

1. Login

2. Browse available projects

3. Click 'Apply'

4. Submit CGPA & resume

5. Await faculty approval

3 Design Overview:

The system follows a modular MVC architecture. Each module communicates via secure REST APIs. The frontend is built with JavaScript, the backend with Express.js, and MySQL as the DB.

3.1 Design Goals and Constraints:

Goals: Scalability, usability, security, modularity

Constraints: Semester deadline, use of specific tech stack (HTML/CSS/JS, Express.js, MySQL), limited budget (free-tier cloud)

3.2 Design Assumptions:

All users have internet access. University provides authentication tokens. Admin is the sole authority for approving all project applications. Notifications are app-based only.

3.3 Significant Design Packages:

- Auth Module: Login/logout, session
- Faculty Module: Post/manage projects
- Student Module: Browse/apply projects
- Admin Module: Control, reports
- Messaging Module: Notifications

3.4 Dependent External Interfaces:

- University Auth API
- Google Cloud SMTP (for future email alerts)
- DigitalOcean DB connection

3.5 Implemented Application External Interfaces:

/api/login (Auth)

/api/postProject (Faculty)

/api/apply (Student)

/api/approve (Admin)

/api/notify (Messaging)

4 Logical View:

System divided into AuthController, ProjectController, ApplicationController, NotificationController. Each controller has service and data access layers. All requests handled via route middleware.

4.1 Design Model:

Each module (e.g., Auth, Faculty, Admin) has a main controller with classes for DB interaction, validation, and response formatting.

4.2 Use Case Realization:

Use Case: Student Applies for Project

1. Student logs in
2. Project list is fetched
3. Student selects and applies
4. ApplicationController logs entry

5. Faculty notified

5 Data View:

The system uses MySQL with normalized tables for users, projects, applications, and messages.

5.1 Domain Model:

Entities: User, Project, Application, Notification

Relationships: One faculty posts many projects, one student applies to many projects, admin monitors all.

5.2.1 Data Dictionary:

- user_id: INT PK
- email: VARCHAR(100)
- password: VARCHAR(100)
- role: ENUM('student','faculty','admin')
- project_id: INT PK
- status: ENUM('pending','accepted','rejected')

6 Exception Handling:

- LoginFailureException: Wrong credentials
- UnauthorizedAccess: Wrong role access

- DBConnectionFailure: Retry and alert admin
- ApplicationConflict: Duplicate application

7 Configurable Parameters:

- session.timeout: 30 mins
- max.active.users: 500
- log.level: 'INFO'
- db.host: 'localhost'

8 Quality of Service:

System must have 99.5% uptime, low-latency project listing, and secure encrypted data transmission.

8.1 Availability:

Cloud-hosted with failover support. Minimal downtime planned during semester breaks only.

8.2 Security and Authorization:

Role-based access enforced. Session tokens. Data encrypted using bcrypt and TLS 1.2.

8.3 Load and Performance Implications:

System supports 500 concurrent users, each submitting requests for project browsing and application. Backend supports auto-scaling in production.

8.4 Monitoring and Control:

All logs stored in cloud. Admin has dashboard for user tracking and project stats. Alerts for DB or service failures.