# Development Plan Document

## 1. Project Overview

The Course Management System is a comprehensive solution aimed at modernizing university operations. It will streamline the management of courses, student enrollment, grading processes, and faculty assignments. The system will include intuitive interfaces, secure authentication mechanisms, and integrated reporting capabilities.

## 2. Core Features

The system will provide the following core features to meet the university's operational requirements:

* - Course Management: Enable administrators to add, edit, and delete courses.
* - Student Management: Facilitate student enrollment, record maintenance, and grade computation.
* - Faculty Management: Manage faculty assignments and course load distribution.
* - Authentication System: Ensure secure access for different user roles (students, faculty, administrators).
* - Grade Recording and Computation: Automate grade calculations and ensure accuracy.
* - Reporting System: Generate detailed reports for grades, class lists, and faculty workload.

## 3. System Architecture

The system will follow the Model-View-Controller (MVC) architecture to ensure modularity and scalability. It will integrate a robust backend, an interactive frontend, and a relational database for seamless data management.

### Technology Stack

The technology stack includes:

* - Frontend: React.js for dynamic user interfaces.
* - Backend: Django (Python) for handling business logic and APIs.
* - Database: PostgreSQL for relational data storage.
* - Authentication: Django's built-in authentication module or OAuth for secure access control.

## 4. Development Phases

### Planning and Setup

* - Create project structure.
* - Initialize Git repository and implement branching strategy.
* - Develop project documentation (e.g., README, development plan).
* - Set up development tools (e.g., VSCode, Postman).

### Authentication System

* - Develop login and registration functionality for all user roles.
* - Implement secure data storage using hashing algorithms (e.g., bcrypt).
* - Add role-based access controls to manage system permissions.

### Core Features Development

* - Course Management: Add, edit, and delete course functionalities.
* - Student Management: Manage enrollment, records, and grades.
* - Faculty Management: Assign courses and manage workloads.
* - Grade Recording and Computation: Automate grade calculations.

### Reporting System Development

* - Generate reports for grades, class lists, and faculty assignments.
* - Enable data export in formats such as PDF and CSV.

### Testing and Deployment

* - Write and execute test cases for all modules (unit, integration, and user testing).
* - Deploy the system on a hosting platform (e.g., AWS, Heroku).

## 5. Implementation Timeline

The development timeline is outlined below:

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Duration | Start Date | End Date |
| Planning and Setup | 1 week | YYYY-MM-DD | YYYY-MM-DD |
| Authentication System | 2 weeks | YYYY-MM-DD | YYYY-MM-DD |
| Core Features Development | 4 weeks | YYYY-MM-DD | YYYY-MM-DD |
| Reporting System Development | 2 weeks | YYYY-MM-DD | YYYY-MM-DD |
| Testing and Deployment | 2 weeks | YYYY-MM-DD | YYYY-MM-DD |