Project:Automation of Academic Section

1. Introduction

1.1 Purpose

The purpose of this project is to demonstrate the components in a Academic system,and establishing communication between the components.Students,Professors and Admins are present in this project.Hostel service is also present which is under Admin's control.

1.2 Scope

1.2 Scope

The system will be a web-based application with:

- Backend REST APIs (Node.js, Express, PostgreSQL/MySQL).
- Frontend (React.js).
- Role-based authentication (professor, student).

  Stakeholders:

- Professors (create & manage assignments, mark attendance)
- Students (attempt assignments, view attendance)
- Administrators (manage students, professors).

2. General Description

2.1 Product Features

- Authentication & Authorization

  1.Professor login, student login.

  2.JWT-based authentication.

- Assignments

  Professors: create assignments questions and can edit anytime.

Students: view assigned work, autosave, submit.
Backend stores answers and grades.

Professors and students can view marks

- Attendance

  Professors: mark students present/absent per course per date.

  Students: view attendance percentage per subject.

- APIs (sample endpoints)

  POST /api/professors/assignments → Create assignment.

  GET /api/student/assignments?student_id=? → Student assignment list.
  POST /api/student/assignments/:id/autosave → Autosave answers.
  POST /api/student/assignments/:id/submit → Submit final answers.
  GET /api/professors/attendance?course_id=...&date=... → Attendance data.

2.2 User Classes

Professor: Creates assignments, evaluates, marks attendance.

Student: Attempts assignments, views submissions & attendance.

Admin (future): Manages users and courses and hostel.

2.3 Operating Environment

Backend: Node.js 18+, Express.js

Database: PostgreSQL/MySQL

Frontend: React.js

Tools: PowerShell/cURL/Postman for testing

3.Functional Requirements

1. Authentication

   System must allow professor/student login with JWT tokens.

2. Assignment Creation

   Professors must create assignments with title, description, due date, questions.

   Professor can edit the questions and answers any time.

   Each question must have options and a correct answer.

3. Assignment Attempt

   Students must view available assignments.

   Students can autosave answers before submission.

   Students are able to change the submissions until the deadline ends and the answers get auto saved at the deadline

4. Attendance Management

   Professors are able to mark students present/absent for a course/date.
   System must lock attendance edits after 30 minutes.

   Students can see percentage attendance per course.

5. System APIs

   APIs must validate JWT tokens.

   Unauthorized requests will return HTTP 403.

   Authorized requests returned HTTP 200

4. Non-Functional Requirements.

Security: Passwords stored as bcrypt hashes. JWT tokens for API access.

Usability: Clean web UI with clear navigation.

5. Database Tables

- professors(professor_id(username), name(full_name), email, password_hash(password), dept)

- students(student_id, name, email, password_hash,program,year)

- courses(course_id, course_title,term,seats_total,seats_available)
- course_responses(id,student_id,course_id,response_status,select_at,updated_at,confirmed_at)
- attendance(courses.course_id,students.student_id,date,status,marked_by(fk from professors),marked_at)

- assignments(assignment_id, title, description, due_date, course_id, created_by)
- assignment_questions(question_id, assignment_id, text, question_text)

- assignment_options(option_id,question_id,label,option_text,is_correct)
- assignment _answers(anwer_id,submission_id,question_id,selected_label,correct(correct/incorrect))

5. External Interface Requirements

Frontend UI
Admin, Professor and Student dashboards.

Backend API
RESTful endpoints returning JSON.

Testing Tools

PowerShell for API validation.