ManageCollege -College Management System

A

RDBMS Lab Project Report

Submitted in the partial fulfillment of the requirement for the award of

Bachelor of Technology

in

Computer and Communication Engineering

By:

Name: Hiya Ajay Gupta
Reg No.: 23FE10CCE00116
Section: A

Under the supervision of:

Ms.Hema Shekhawat
Dr. Jitendra Singh Yadav
Dr. Sandeep Sharma

April 2025

Department of Computer and Communication Engineering
School of Computer Science and Engineering,
Manipal University Jaipur,
VPO. Dehmi Kalan, Jaipur, Rajasthan, India – 303007

Student Declaration

*I hereby declare that this project **ManageCollege** is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the University or other Institute, except where due acknowledgements have been made in the text.*

Place: Manipal University Jaipur

Name of Student: Hiya Ajay Gupta

Date: 21st April 2025

Reg No.: 23FE10CCE00116 — Section: A

B. Tech (CCE) 4th Semester

Acknowledgement

I would like to express my heartfelt gratitude to my project supervisors, **Ms.Hema Shekhawat**, **Dr. Jitendra Singh Yadav** and **Dr. Sandeep Sharma**, for their invaluable guidance, constant support, and encouragement throughout the course of this project. Their insightful suggestions, meticulous feedback, and expert advice helped me refine my ideas and bring the project to fruition. I have greatly benefited from their knowledge and experience.

I would also like to extend my sincere thanks to the faculty members of the **Department of Computer and Communication Engineering, Manipal University Jaipur** for imparting their knowledge and for providing me with the necessary tools and skills to undertake such a project.

A special thanks to my friends and colleagues who have supported me during the entire process. Their constructive discussions and moral support kept me motivated and focused.

Abstract

In today's digital age, managing academic institutions efficiently requires robust and integrated systems. The **ManageCollege** project is a basic College Management System developed as part of the RDBMS Lab to implement some basic functionalities of an academic institution. This system is designed to handle various modules such as student, faculty details, course assignments, department records, and attendance tracking for the administrator.

The backend of the application is built on a relational database model using PostgresSQL for data definition and manipulation. The system employs ER modeling to establish relationships between entities and ensure data integrity. The frontend interface is developed using HTML, CSS, and JavaScript-**Express**, providing a user-friendly platform for administrators to interact with the database.

The project demonstrates key database operations such as the use of DDL (Data Definition Language) and DML (Data Manipulation Language) queries, normalization principles, and the implementation of foreign key constraints. It also focuses on real-time data access, ensuring that all information remains consistent and reliable.

Overall, the ManageCollege project highlights the practical application of relational database concepts in building scalable and functional management systems.

Contents

Student Declaration i

Acknowledgement ii

Abstract iii

List of Figures v

List of Tables vi

1 Introduction 1

2 Motivation 2

3 ER Diagram 3

4 Relational Model 4

5 DDL Query 6
- 5.1 Create Table Queries . . . 6
  - 5.1.1 Create Departments Table . . . 6
  - 5.1.2 Create Students Table . . . 6
  - 5.1.3 Create Teachers Table . . . 7
  - 5.1.4 Create Courses Table . . . 7
  - 5.1.5 Create Attendance Table . . . 7
- 5.2 Constraints and Relationships . . . 7

6 Functionalities with DML SQL queries 8
- 6.1 Sample Data Insertions . . . 8
  - 6.1.1 Departments . . . 8
  - 6.1.2 Students . . . 8
  - 6.1.3 Teachers . . . 8
  - 6.1.4 Courses . . . 8
  - 6.1.5 Attendance . . . 9

7 Front-End UI and Code Snapshots 10
- 7.1 Overview . . . 10
- 7.2 UI Snapshots . . . 10
- 7.3 Code Snippets . . . 15

References 18

List of Figures

Figure No.	Title	Reference
Figure 1.1	ER Diagram of ManageCollege System	ER Diagram
Figure 2.1	Relational Schema Representation	Relational Model
Figure 3.1	Student Page UI	UI Snapshots
Figure 3.2	Faculty Page UI	UI Snapshots
Figure 3.3	Courses Page UI	UI Snapshots
Figure 3.4	Departments Page UI	UI Snapshots
Figure 3.5	Attendance Page UI	UI Snapshots
Figure 3.6	Attendance Dropdown - Student	UI Snapshots
Figure 3.7	Attendance Dropdown - Courses	UI Snapshots
Figure 3.8	Attendance Dropdown - Status	UI Snapshots
Figure 4.1	Home/Dashboard Interface	UI Snapshots

List of Tables

Table No.	Title	Reference
Table 1.1	Entity Attributes in ER Diagram	ER Diagram
Table 2.1	Relational Schema Details	Relational Model
Table 3.1	DDL Queries Used to Create Tables	DDL Queries
Table 4.1	Sample Insert Queries (Students, Courses, etc.)	DML Queries

Table 2: Table List and References

1 Introduction

In today's academic environment, efficient and systematic management of institutional data has become essential for ensuring smooth administrative operations. Educational institutions, especially colleges, handle a vast amount of data related to students, faculty, departments, courses, and attendance on a daily basis. Traditional manual methods of managing such information are prone to errors, time-consuming, and often lack accessibility. To overcome these challenges, robust digital systems are required that can handle, store, and retrieve data reliably and efficiently.

ManageCollege is a college management system developed as part of the RDBMS Lab project. It is designed to manage various academic and administrative tasks such as maintaining student records, managing faculty data, organizing departmental and course details, and recording attendance. The system utilizes a relational database structure to store and query data efficiently and is supported by a user-friendly front-end interface that allows seamless interaction with the system.

This project not only demonstrates the practical implementation of database concepts like data modeling, normalization, and SQL queries (DDL and DML), but also emphasizes the integration of front-end development with backend database operations. The application provides CRUD functionalities (Create, Read, Update, Delete) and enforces data consistency using structured relationships among tables.

The primary goal of ManageCollege is to provide a simple, reliable, and extendable solution for college data management that can be easily scaled or modified as per institutional requirements. This project reflects a real-world implementation of theoretical database principles and demonstrates their application in solving everyday problems faced in academic institutions.

2 Motivation

The management of academic institutions involves handling extensive data related to students, faculty, departments, courses, and daily attendance records. Traditionally, this data has been maintained manually using registers, spreadsheets, or other basic tools. However, these methods often lead to problems such as data redundancy, inconsistency, human error, and inefficient retrieval of information, especially as the volume of data grows.

The motivation behind developing the ManageCollege system stems from the need to address these limitations by building a centralized and structured database-driven system. A college management platform built on relational database principles ensures data integrity, consistency, and easy access to relevant information. Furthermore, with the increasing emphasis on digitization in education, there is a clear demand for platforms that simplify administrative tasks and provide transparency and accountability in academic operations.

Through this project, we aimed to apply our theoretical understanding of relational database concepts such as ER modeling, schema design, and SQL queries into a practical solution. Additionally, integrating the database with a front-end user interface provided the opportunity to learn full-stack development and create a usable application that mirrors real-world educational software.

Ultimately, the ManageCollege system is motivated by the desire to create a smart, scalable, and efficient solution that can enhance institutional workflows, reduce paperwork, and offer a hands-on application of the concepts learned in the RDBMS Lab.

3 ER Diagram

The Entity-Relationship (ER) Diagram represents the logical structure of the ManageCollege system and outlines how various entities in the system are interrelated. It provides a high-level view of the system's data components and their connections, making it easier to design and implement the database schema.

The key entities identified in the system include:

Student – Contains student-specific details such as student ID, name, email, department, and semester. Faculty – Represents the instructors involved in course delivery, identified by faculty ID, name, and department. Department – Covers department details including department ID and name. Course – Contains information about different courses, their names, and associated departments. Attendance – Records student attendance against specific courses on given dates, including present/absent status. These entities are connected through relationships such as:

A Student belongs to a Department A Faculty belongs to a Department A Course is offered by a Department A Student is enrolled in multiple Courses Attendance is recorded for each Student-Course combination The following figure illustrates the ER Diagram for the ManageCollege system:

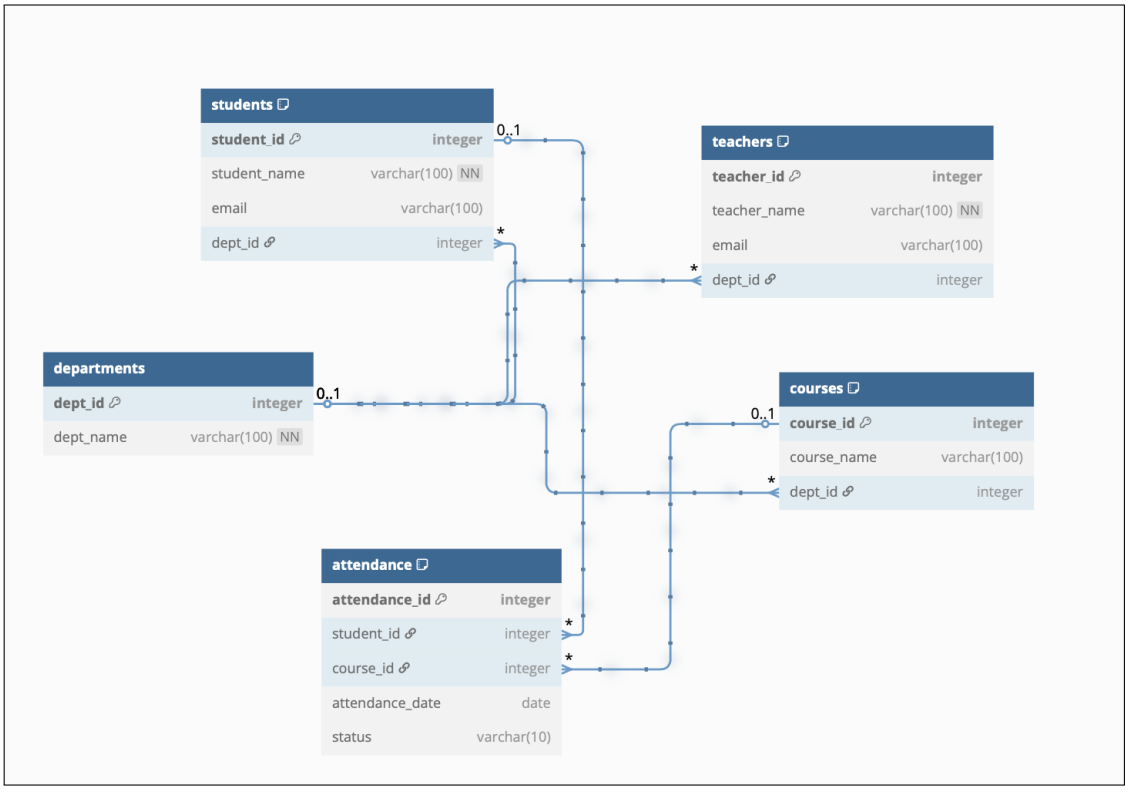

Figure 3.1: Entity-Relationship Diagram

4 Relational Model

The relational model of the *ManageCollege* system translates the ER diagram into a set of normalized relational schemas, where data is organized into well-defined tables with keys and constraints to ensure data integrity. Each table corresponds to an entity or relationship in the ER diagram. Below is a description of the relational schemas:

Table Name	Attributes	Description
Student	`student_id` (PK), `student_name`, `email`, `dept_id` (FK), `semester`	Each student is uniquely identified by `student_id`. `dept_id` references the Department table.
Faculty	`faculty_id` (PK), `faculty_name`, `dept_id` (FK)	Each faculty member is associated with a department through `dept_id`.
Department	`dept_id` (PK), `dept_name`	Represents the various academic departments in the college.
Course	`course_id` (PK), `course_name`, `dept_id` (FK)	Each course is linked to a department.
Attendance	`attendance_id` (PK), `student_id` (FK), `course_id` (FK), `status`, `date`	Tracks student attendance for each course with status (Present/Absent). Foreign keys link to Student and Course tables.

Table 4.1: Relational Schema Representation

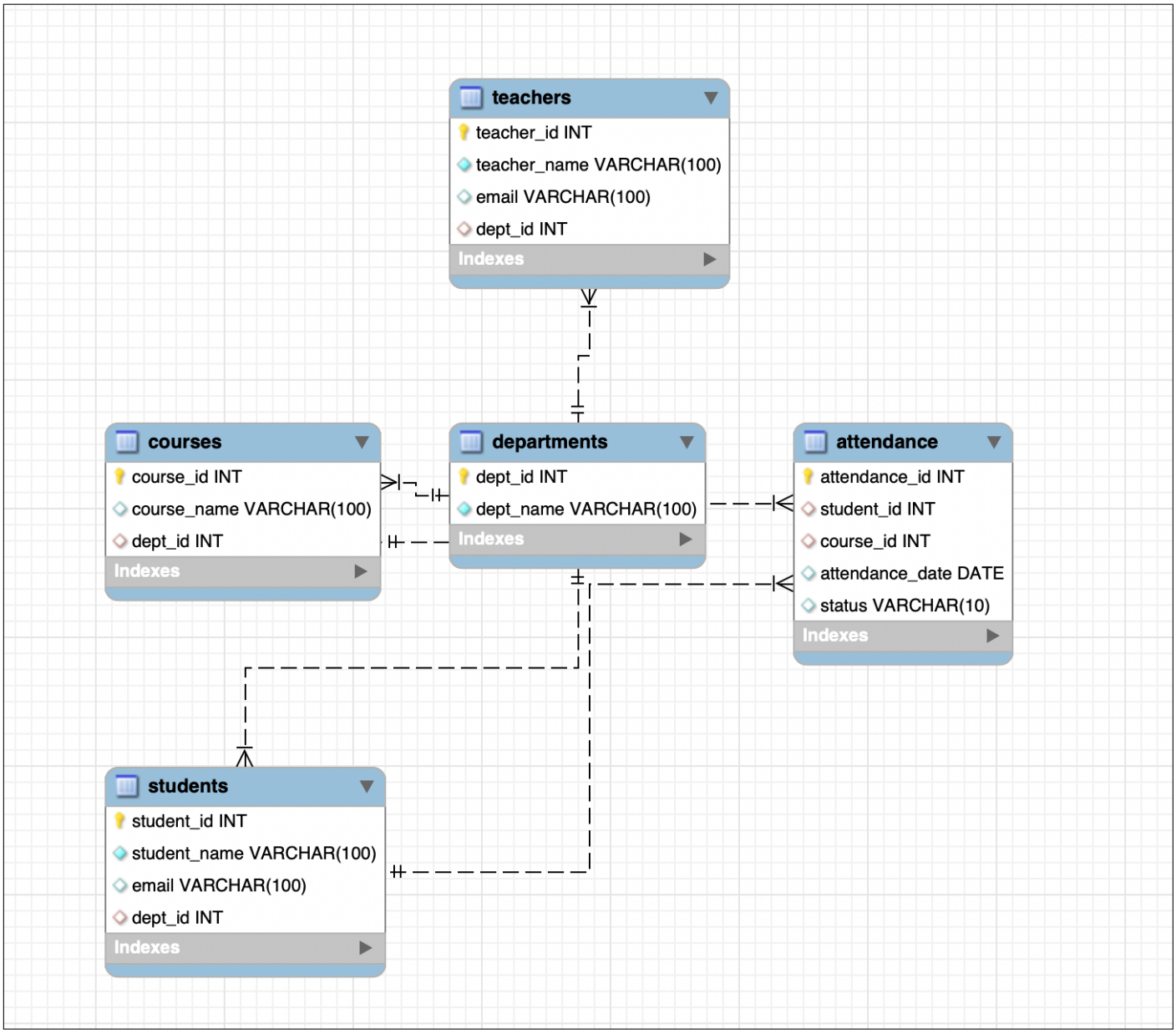

Figure 4.1: Relational Schema Representation

5 DDL Query

Data Definition Language (DDL) queries are used to define the structure of the database. In this chapter, we will describe the DDL queries used to create the tables and enforce the relationships in the *ManageCollege* system.

5.1 Create Table Queries

The following DDL queries were used to create the tables corresponding to the relational schema described earlier:

5.1.1 Create Departments Table

```
CREATE TABLE departments (
    dept_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    dept_name VARCHAR2(100) NOT NULL
);
```

The `departments` table holds the names of all academic departments in the college. The `dept_id` is the primary key and is automatically generated.

5.1.2 Create Students Table

```
CREATE TABLE students (
    student_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    student_name VARCHAR2(100) NOT NULL,
    email VARCHAR2(100),
    dept_id NUMBER,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

The `students` table stores information about the students, including their unique `student_id`, `student_name`, and `email`. The `dept_id` is a foreign key that references the `departments` table.

5.1.3 Create Teachers Table

```
CREATE TABLE teachers (
    teacher_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    teacher_name VARCHAR2(100) NOT NULL,
    email VARCHAR2(100),
    dept_id NUMBER,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

The `teachers` table stores information about the faculty members, with each teacher associated with a department using the `dept_id` foreign key.

5.1.4 Create Courses Table

```
CREATE TABLE courses (
    course_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    course_name VARCHAR2(100),
    dept_id NUMBER,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

The `courses` table defines the courses offered in various departments. Each course is linked to a department using the `dept_id` foreign key.

5.1.5 Create Attendance Table

```
CREATE TABLE attendance (
    attendance_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    student_id NUMBER,
    course_id NUMBER,
    attendance_date DATE DEFAULT SYSDATE,
    status VARCHAR2(10),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

The `attendance` table tracks student attendance for each course on a given date. The `status` field stores whether the student was `Present` or `Absent`. The `attendance_date` is automatically set to the current date using `SYSDATE`.

5.2 Constraints and Relationships

The relationships between the tables are established using foreign keys:

- **Foreign Key Constraints:** Ensure that each student, teacher, and course belongs to an existing department.

6 Functionalities with DML SQL queries

6.1 Sample Data Insertions

The following sample insert statements were used to populate the database with initial data for testing and demonstration purposes.

6.1.1 Departments

```
INSERT INTO departments (dept_name) VALUES ('Computer Science');
INSERT INTO departments (dept_name) VALUES ('Mechanical Engineering');
INSERT INTO departments (dept_name) VALUES ('Electrical Engineering');
```

6.1.2 Students

```
INSERT INTO students (student_name, email, dept_id)
VALUES ('Aarav Gupta', 'aarav@example.com', 1);

INSERT INTO students (student_name, email, dept_id)
VALUES ('Sneha Verma', 'sneha@example.com', 2);
```

6.1.3 Teachers

```
INSERT INTO teachers (teacher_name, email, dept_id)
VALUES ('Dr. Mehta', 'mehta@example.com', 1);

INSERT INTO teachers (teacher_name, email, dept_id)
VALUES ('Prof. Rao', 'rao@example.com', 3);
```

6.1.4 Courses

```
INSERT INTO courses (course_name, dept_id)
VALUES ('Data Structures', 1);

INSERT INTO courses (course_name, dept_id)
VALUES ('Thermodynamics', 2);
```

6.1.5 Attendance

```
INSERT INTO attendance (student_id, course_id, status)
VALUES (1, 1, 'Present');

INSERT INTO attendance (student_id, course_id, status)
VALUES (2, 2, 'Absent');
```

7 Front-End UI and Code Snapshots

7.1 Overview

The front-end of the ManageCollege system is designed using HTML, CSS, and JavaScript. It offers a clean, intuitive user interface that allows users to interact with the database functionalities, including adding and viewing students, faculty, courses, and managing attendance.

7.2 UI Snapshots

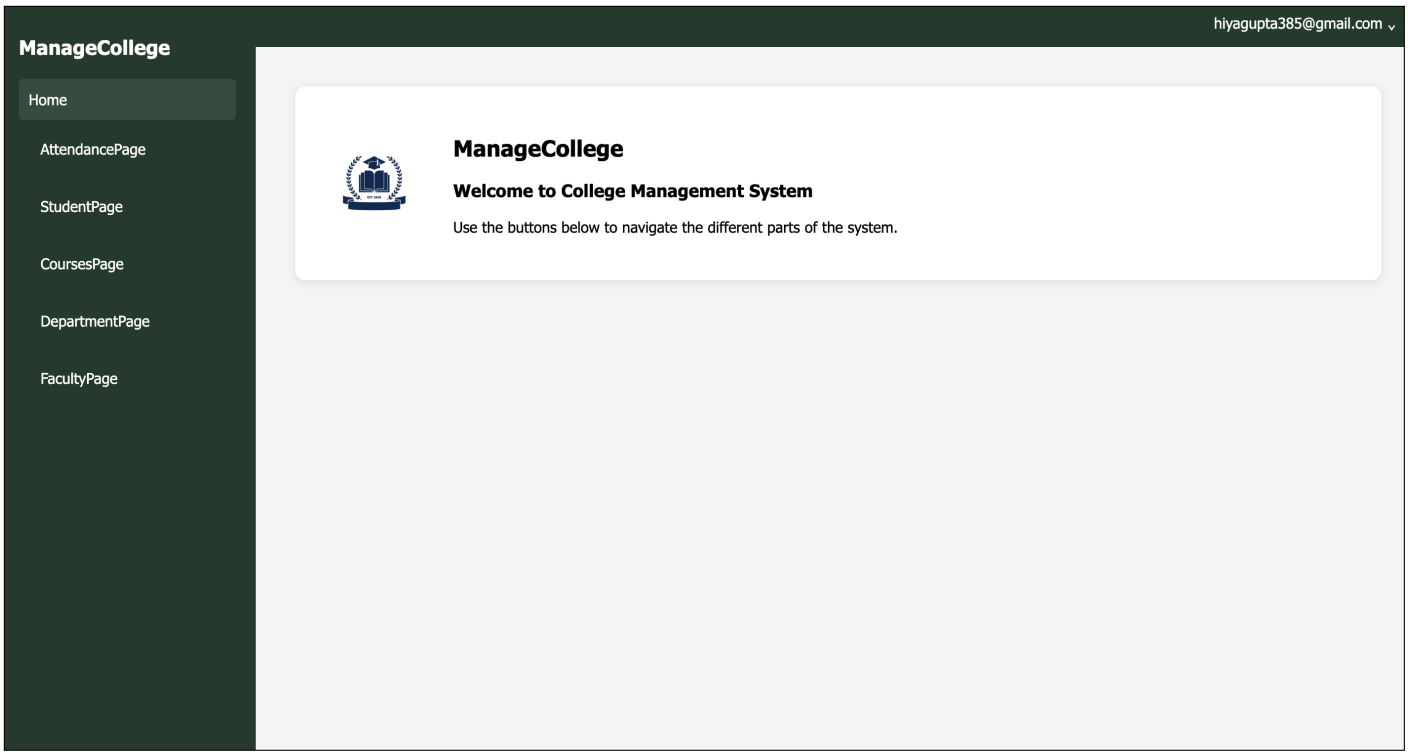

Figure 7.1: Dashboard Interface UI

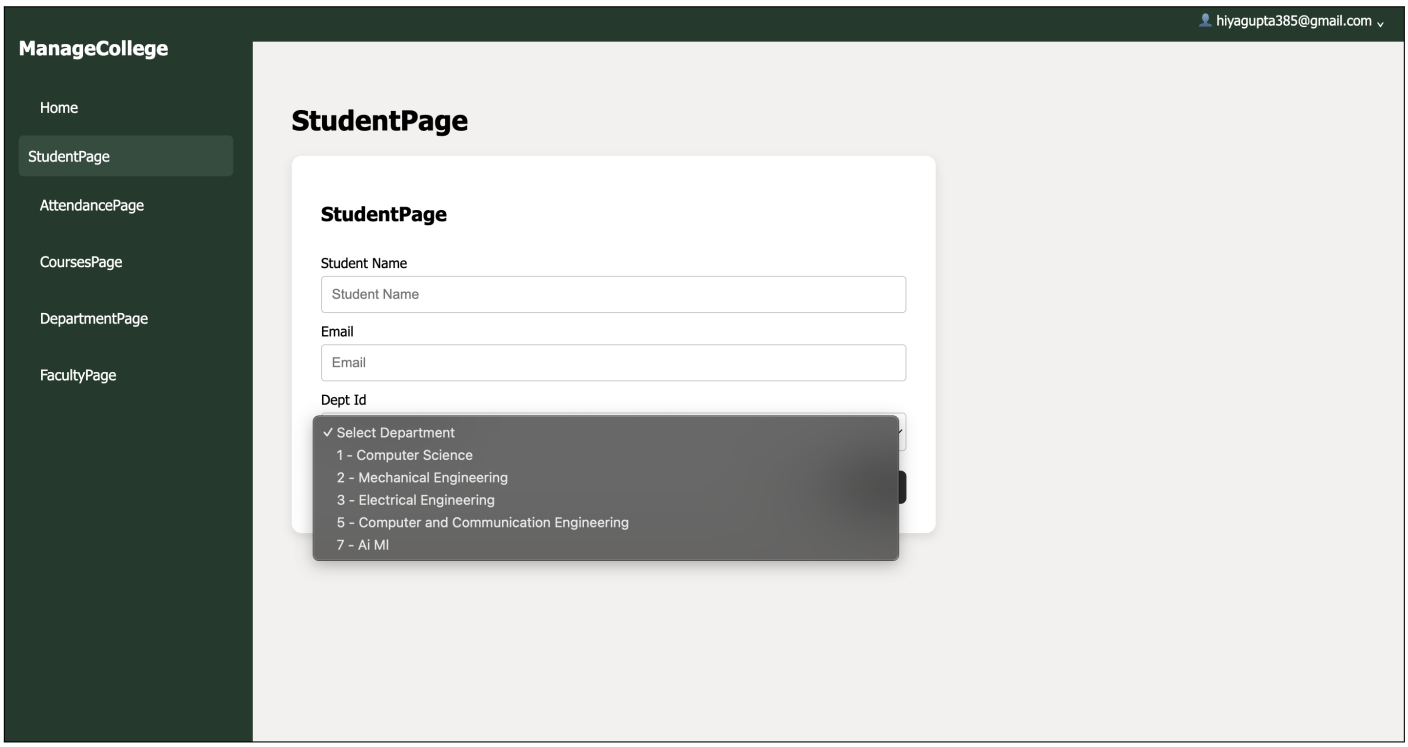

Figure 7.2: Student Page UI

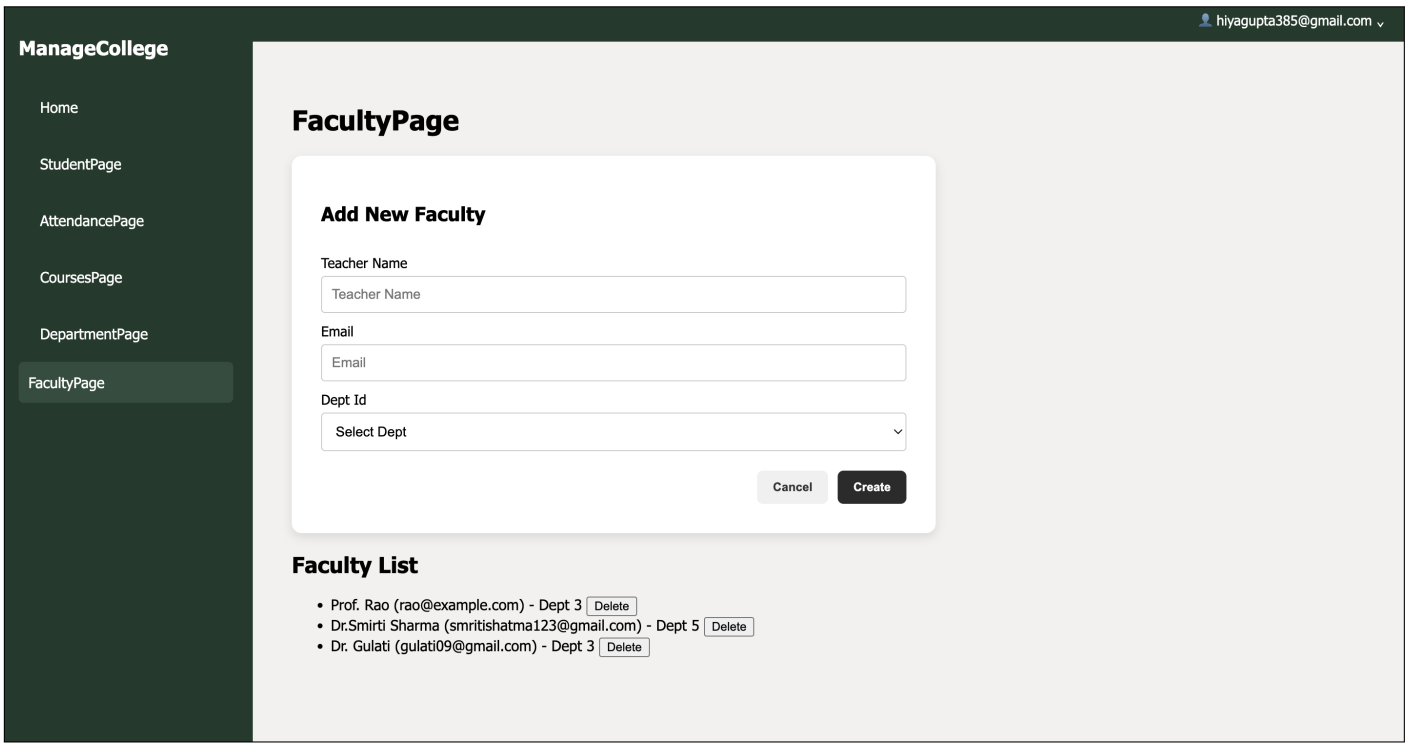

Figure 7.3: Faculty Page UI

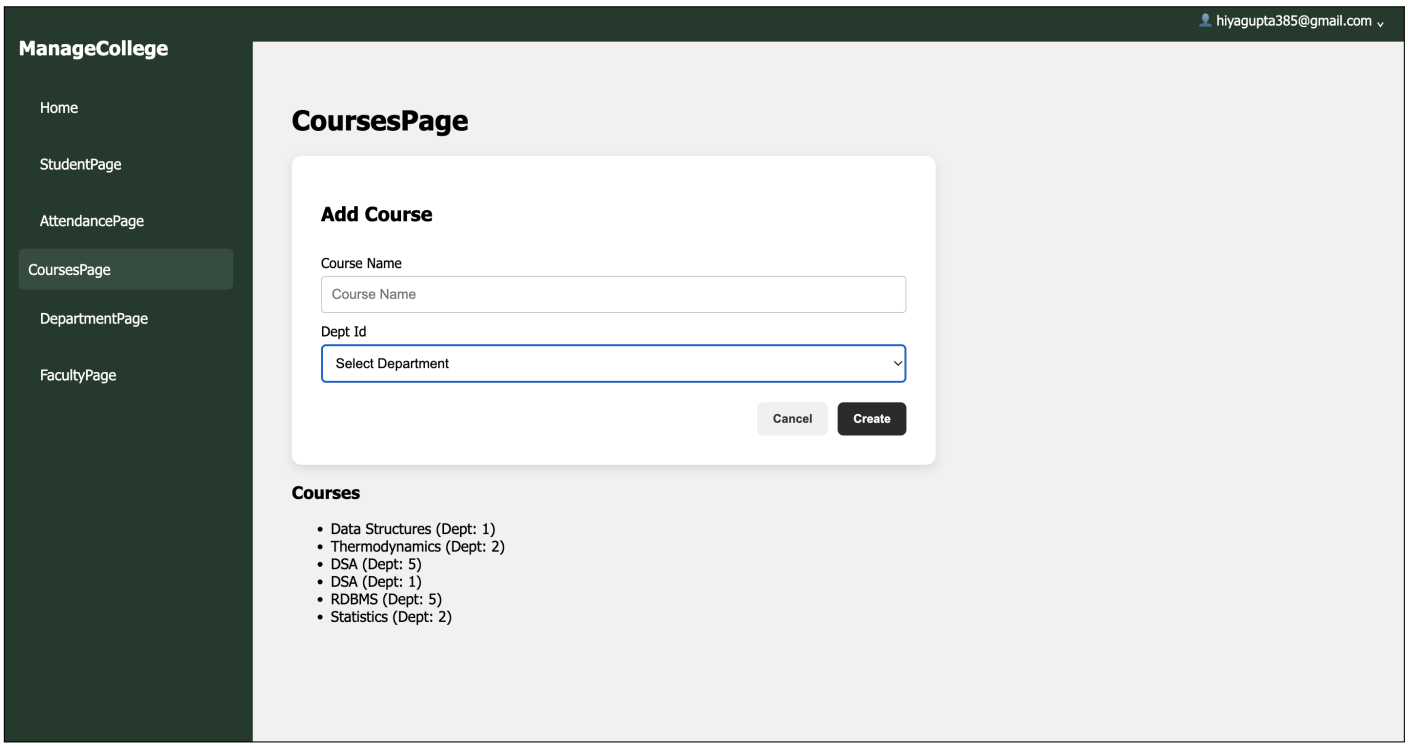

Figure 7.4: Courses Page UI

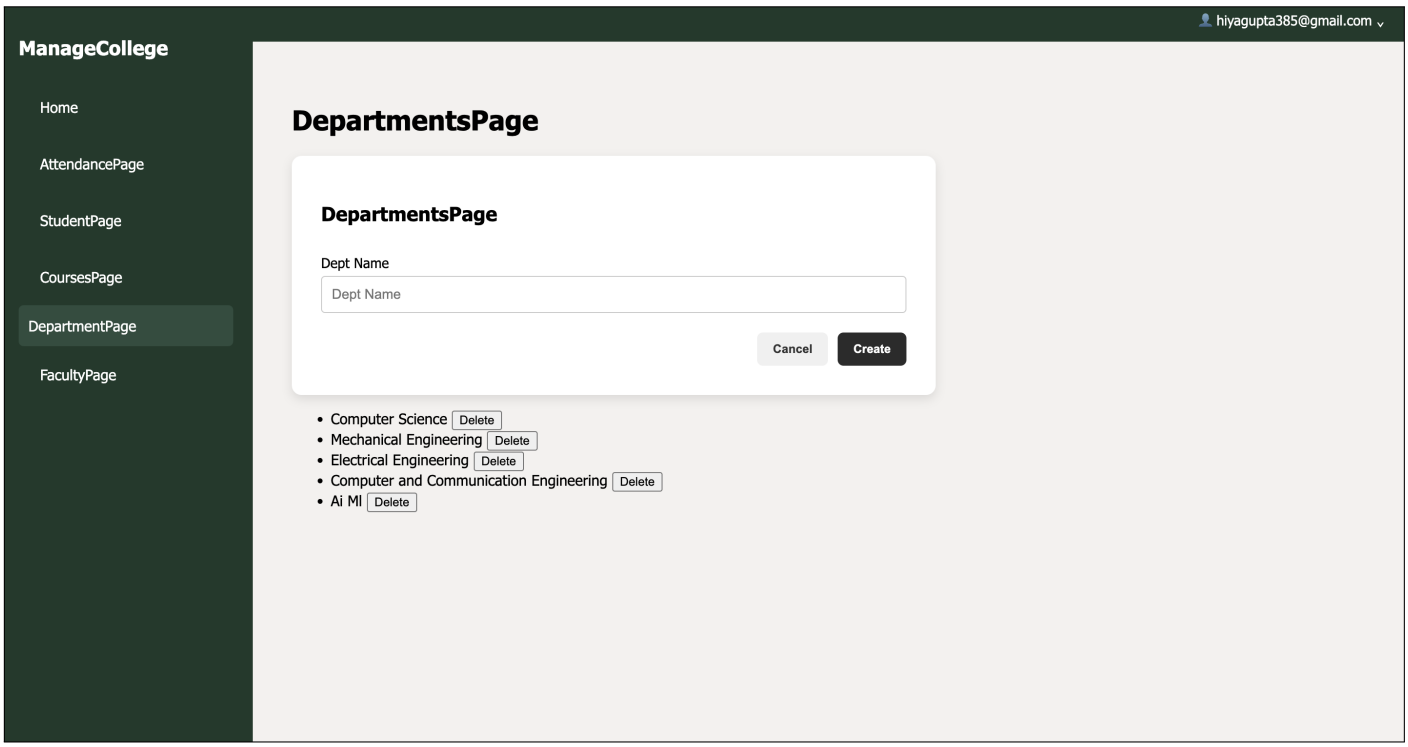

Figure 7.5: Departments Page UI

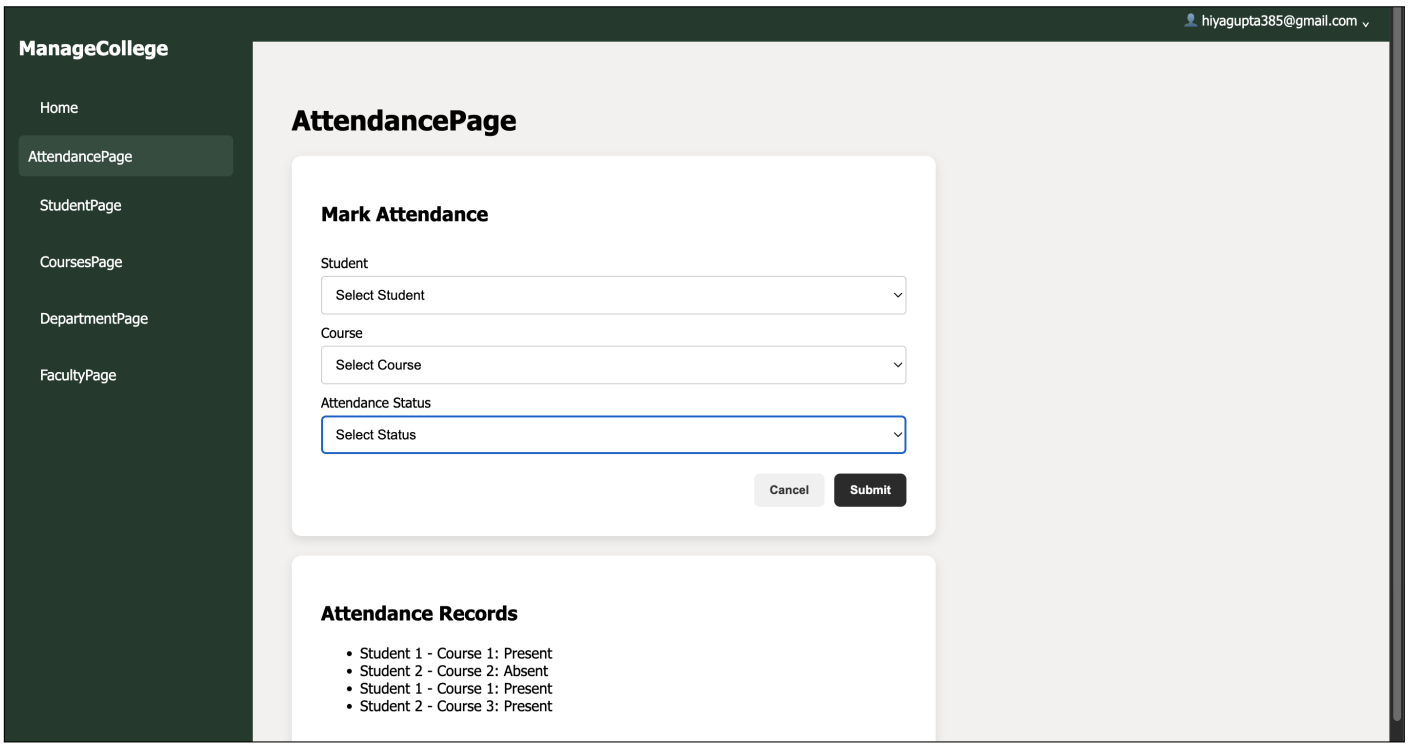

Figure 7.6: Attendance Page UI

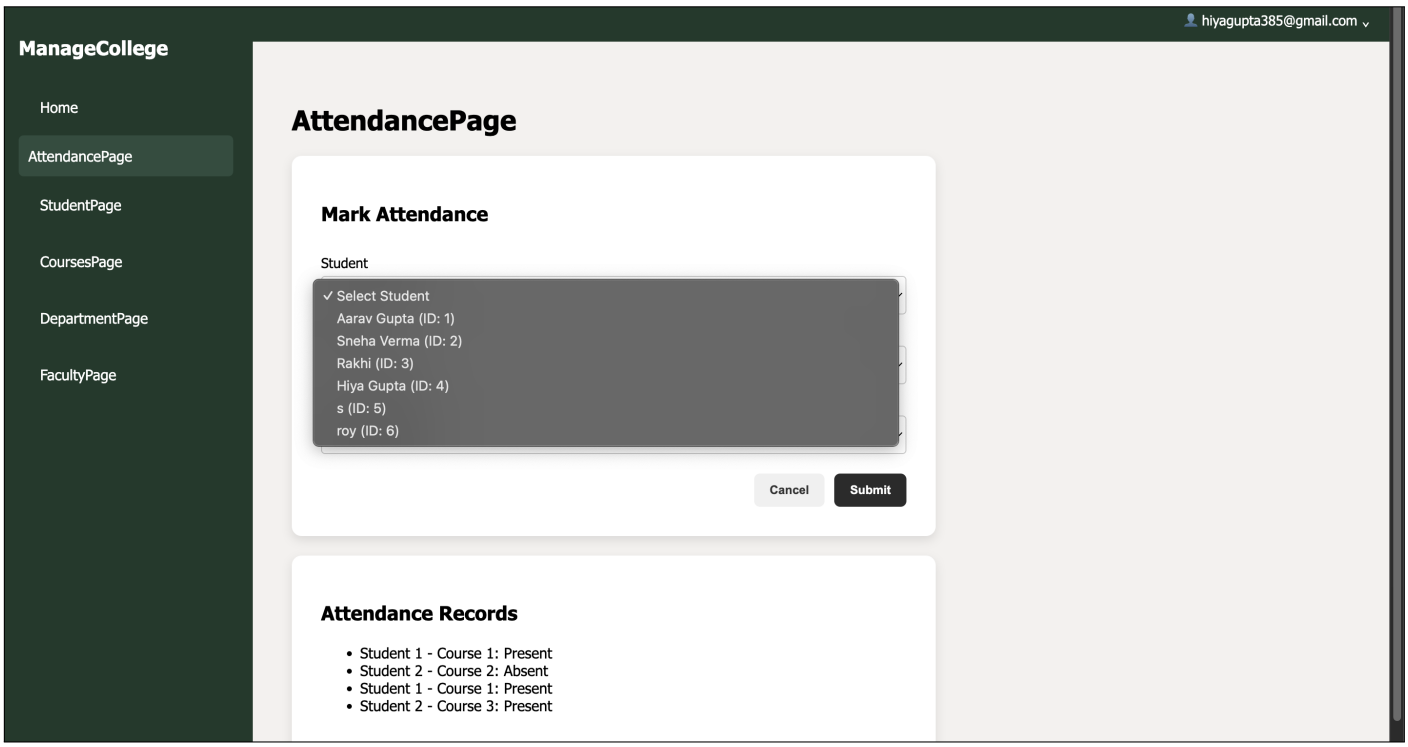

Figure 7.7: Attendance Page-Student Dropdown UI

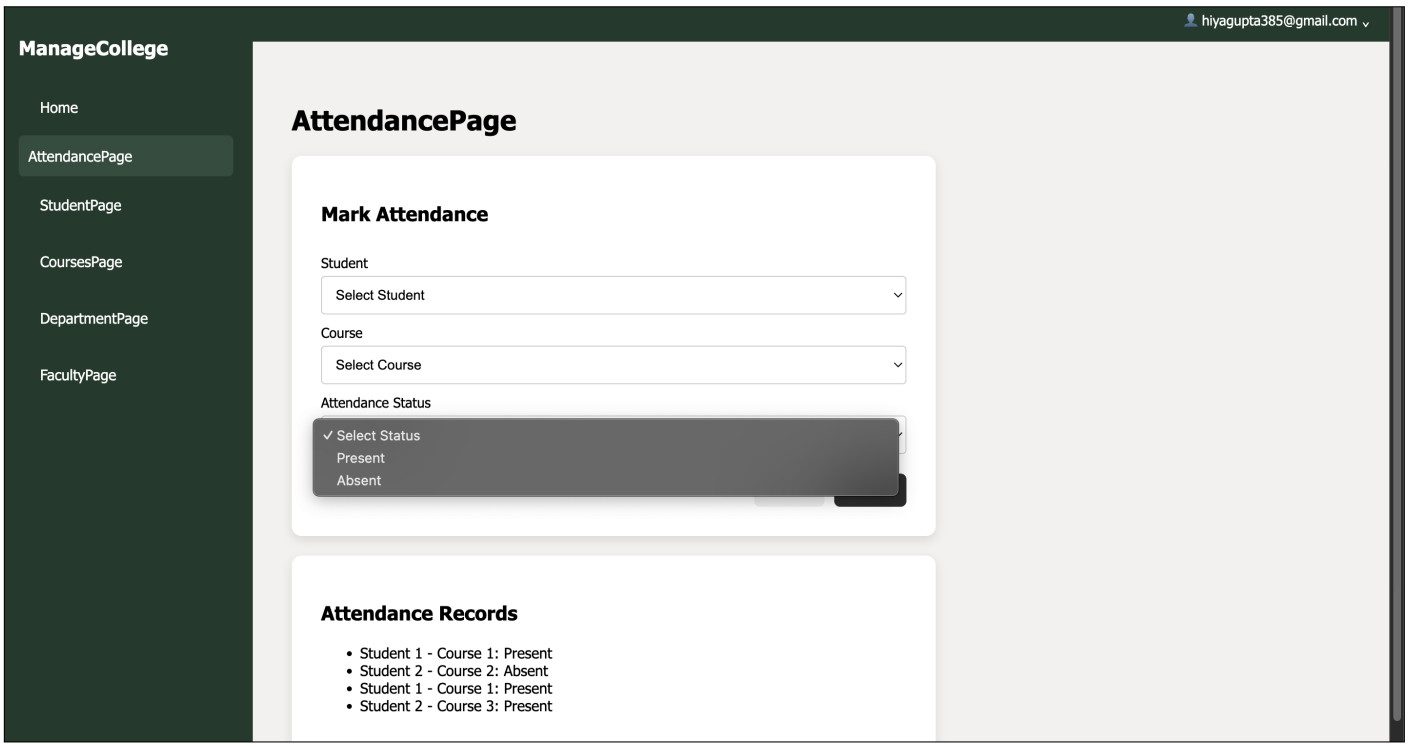

Figure 7.8: Attendance Page-Status Dropdown UI

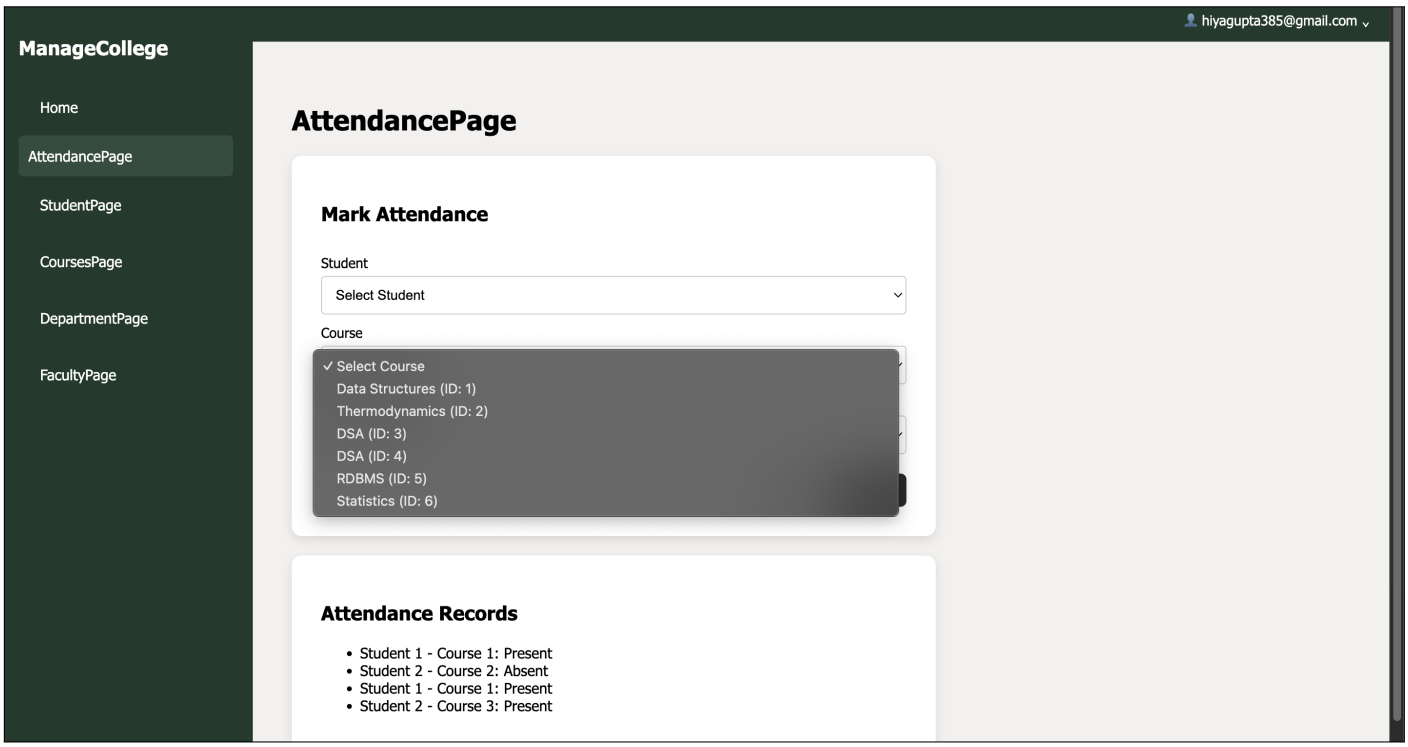

Figure 7.9: Attendance Page-Courses Dropdown UI

7.3 Code Snippets

The main **Server.js** code:

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const path = require("path");

const app = express();
const db = require("./db");

//Middleware
app.use(cors());
app.use(bodyParser.json());

//serve static files
app.use(express.static(path.join(__dirname, 'public')));

app.use("/students", require("./routes/students"));
app.use("/teachers", require("./routes/faculty"));
app.use("/departments", require("./routes/departments"));
app.use("/courses", require("./routes/course"));
app.use("/attendance", require("./routes/attendance"));

const PORT = process.env.PORT || 5501;
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`))
app.get('/', (req, res) => {
   res.sendFile(path.join(__dirname, 'views/index.html'));
  });

  app.get('/student-page', (req, res) => {
    res.sendFile(path.join(__dirname, 'views/students.html'));
  });

  app.get('/attendance-page', (req, res) => {
    res.sendFile(path.join(__dirname, 'views/attendance.html'));
  });

  app.get('/courses-page', (req, res) => {
    res.sendFile(path.join(__dirname, 'views/courses.html'));
  });
```

```
  app.get('/department-page', (req, res) => {
    res.sendFile(path.join(__dirname, 'views/department.html'));
  });

  app.get('/faculty-page', (req, res) => {
    res.sendFile(path.join(__dirname, 'views/teachers.html'));
  });
  const { insertStudent } = require('./db'); // Custom DB insert function

  app.use(bodyParser.json());

  app.post('/students', async (req, res) => {
    const { student_name, email, dept_id } = req.body;

    if (!student_name || !email || !dept_id) {
      return res.status(400).json({ message: "All fields are required" });
    }

    try {
      // Insert into the database
      await insertStudent(student_name, email, dept_id);
      res.status(201).send("Student added successfully!");
    } catch (error) {
      console.error("Error adding student:", error);
      res.status(500).json({ message: "Error adding student" });
    }
  });
```

The Database Connector (**Db.js**):

```
const { Pool } = require('pg');
const pool = new Pool({
  user: 'postgres',
  host: 'localhost',
  database: 'college_db',
  port: 5432,
});
module.exports = pool;
```

The HTML code for Index-Page (**Index.html**):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>ManageCollege</title>
  <link rel="stylesheet" href="./styles.css">
</head>
<body>
  <div class="container">
    <aside class="sidebar">
      <h2>ManageCollege</h2>
      <ul class="nav">
        <li class="active">Home</li>
        <li><a href="/attendance-page" class="nav-link">AttendancePage</a></li>
        <li><a href="/student-page" class="nav-link">StudentPage</a></li>
        <li><a href="/courses-page" class="nav-link">CoursesPage</a></li>
        <li><a href="/department-page" class="nav-link">DepartmentPage</a></li>
        <li><a href="/faculty-page" class="nav-link">FacultyPage</a></li>
      </ul>
    </aside>

    <div class="main">
      <header class="navbar">
        <div></div>
        <div class="user">hiyagupta385@gmail.com </div>
      </header>

      <section class="content">
        <div class="welcome-box">
          <img src="logo.jpg" alt="College Logo" class="logo" />
          <div>
            <h1>ManageCollege</h1>
            <h3>Welcome to College Management System</h3>
            <p>Use the buttons below to navigate the different parts of the syste
          </div>
        </div>
      </section>
    </div>
  </div>
</body>
</html>
```

References

Oracle Corporation, "SQL Language Reference," *Oracle Docs*, 2024. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/index.html

Mozilla Developer Network, "HTML: HyperText Markup Language," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

Mozilla Developer Network, "CSS: Cascading Style Sheets," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS

Mozilla Developer Network, "JavaScript," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Node.js Foundation, "Node.js v20.x Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs

Express.js, "Express - Node.js web application framework," [Online]. Available: https://expressjs.com/

GitHub, "ManageCollege Project Repository," [Online]. Available: https://github.com/

M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.

W3Schools, "HTML, CSS, JavaScript Tutorial," [Online]. Available: https://www.w3schools.com/

C. J. Date, *An Introduction to Database Systems*, 8th ed., Pearson Education, 2003.