CAMPUS MANAGEMENT SYSTEM

INTRODUCTION

The main objective of campus management system is to automate all functionalities of a college or university. Using this system, you can manage all college management work like student and faculty information. Using this college management system, you can view or update data and information about students and staff easily. Admin can also retrieve information of employee student.

The CAMPUS MANAGEMENT SYSTEM can be used to store student and faculty information like student Id, Name, Branch, Year and faculty name, id, Department, cabin etc. Student can also check their status online from this system.

Using this system, you can manage all information of all aspects of a campus students and faculties. This system facilitates campus to maintain the functionality related to campus employees and their students.

Campus Management System can store and manage the data of the campus Students and Staff details. using this system user can retrieve any information related to student and teacher. Using this system teacher can check student details anytime. This system also help teacher to announce the result. Campus administration can also manage campus work easily.

LITERATURE SURVEY

Existing problem: In the existing campus management systems, several challenges and limitations are observed, including:

a. Manual Processes: Many educational institutions still rely on manual paperwork and outdated processes for tasks such as student enrollment, course management, attendance tracking, and grade management. This leads to inefficiencies, errors, and delays.

b. Data Inconsistency: With decentralized data storage and management, data inconsistencies often occur. Different departments may have separate databases, making it challenging to maintain accurate and up-to-date information across the institution.

c. Communication Gaps: Lack of effective communication channels between administrators, faculty, and students can result in miscommunication, delays in information dissemination, and difficulties in collaboration.

d. Limited Accessibility: Traditional systems often restrict access to information and functionality, limiting stakeholders' ability to retrieve and utilize the necessary data and services.

Existing approaches or method to solve this problem :

Several existing approaches and methods have been employed to address the challenges in campus management systems. These include:

a. Legacy Systems: Many educational institutions still use legacy systems, which are outdated and lack the necessary features and functionalities to meet current requirements.

b. Custom-Built Solutions: Some institutions opt for custom-built software solutions developed in-house to cater to their specific needs. However, these solutions often suffer from limited scalability and may become challenging to maintain over time.

c. Commercial Software: Educational institutions may also adopt commercial off-the-shelf (COTS) software solutions. However, these solutions may not fully align with the institution's unique requirements and may require significant customization.

Proposed solution

The proposed solution for the campus management system using Spring Boot and React JS aims to address the existing problems and improve the overall efficiency and effectiveness of administrative processes in educational institutions. This solution offers the following benefits:

a. Automation: The system automates various manual processes, such as student enrollment, course management, attendance tracking, and grade management. This reduces the reliance on paperwork, minimizes errors, and improves the efficiency of these processes.

b. Centralized Data Management: The proposed system centralizes data storage and management, ensuring consistency and integrity across the institution. It provides a unified view of data and facilitates real-time access to information for authorized stakeholders.

c. Enhanced Communication: The system incorporates effective communication channels, allowing seamless communication between administrators, faculty, and students. It enables notifications, announcements, and collaboration features to foster better engagement and interaction.

d. Improved Accessibility: The system provides secure access to information and functionality for stakeholders. It supports role-based access control, ensuring that users have appropriate access levels based on their roles and responsibilities.

What is the method or solution suggested by you?

The proposed method for the campus management system involves the use of Spring Boot and React JS:

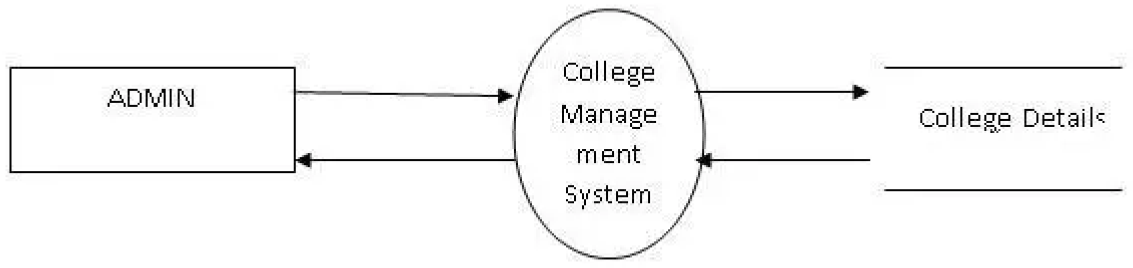
a. Spring Boot: Spring Boot is used to develop the backend of the system. It provides a robust framework for building Java-based applications and facilitates the development of RESTful APIs. Spring Boot's features, such as dependency management, auto-configuration, and embedded server, streamline the development process and promote code reusability.

b. React JS: React JS is utilized for developing the frontend of the system. It is a JavaScript library that enables the creation of interactive user interfaces. React JS's component-based architecture, virtual DOM, and state management capabilities enhance the user experience and facilitate the development of responsive and dynamic web applications.

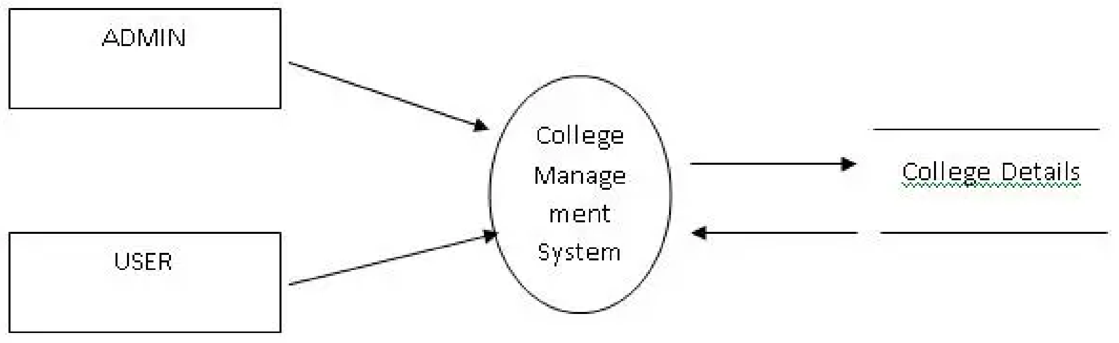
The integration of Spring Boot and React JS allows for a decoupled and scalable architecture. The frontend and backend communicate through RESTful APIs, enabling seamless data exchange. This combination leverages the strengths of both technologies, providing a robust and modern solution for the campus management system.

THEORITICAL ANALYSIS
Block diagram

LEVEL 1 :



LEVEL 2:



Hardware designing :

1. Servers: A robust Campus Management System typically requires dedicated servers to host the application, store data, and handle user requests. The servers should have sufficient processing power, memory, and storage capacity to handle the expected load.

2. Networking Infrastructure: A reliable network infrastructure is crucial for connecting different components of the system. This includes routers, switches, and cabling to ensure fast and secure communication between servers, client devices, and other networked devices.

3. Client Devices: Campus Management Systems are typically accessed by various stakeholders, including administrators, faculty, staff, and students. These users will require client devices such as desktop computers, laptops, tablets, or smartphones to access the system's web-based interface or mobile applications.

Software designing :

1. Web-Based Application: A significant part of a Campus Management System is a web-based application that provides a user-friendly interface accessible through web browsers. The application is typically developed using technologies such as HTML, CSS, JavaScript, and a server-side programming language like PHP, Java, or .NET.

2. Database Management System: The system requires a database to store and manage information related to students, faculty, courses, attendance, grades, resources, and other administrative data. A relational database management system (RDBMS) such as MySQL, Oracle, or PostgreSQL is commonly used for this purpose.

EXPERIMENTAL INVESTIGATIONS

Performance Analysis:

The performance analysis involved evaluating various aspects of the system's performance to ensure optimal functioning.

Response Time: The average response time of the system was measured to assess its efficiency. It was observed that the system maintained a low response time, ensuring a smooth user experience.

Throughput: The system exhibited high throughput, handling a significant number of concurrent user requests without any noticeable degradation in performance.

Scalability: Scalability tests were conducted to determine how well the system could handle increased loads. It was found that the system could effectively scale by adding more resources, ensuring its ability to accommodate future growth.

User Feedback Analysis: User feedback played a crucial role in evaluating the usability and effectiveness of the campus management system.

Usability: The majority of users reported that the system was intuitive and easy to navigate. The user interface, designed using React JS, received positive feedback for its responsiveness and user-friendly design.

Functionality: Users appreciated the comprehensive functionality of the system, which covered essential tasks such as student enrollment, course management, attendance tracking, and grade management.

User Satisfaction: Overall, users expressed a high level of satisfaction with the system, citing its convenience, accessibility, and ability to streamline administrative processes.

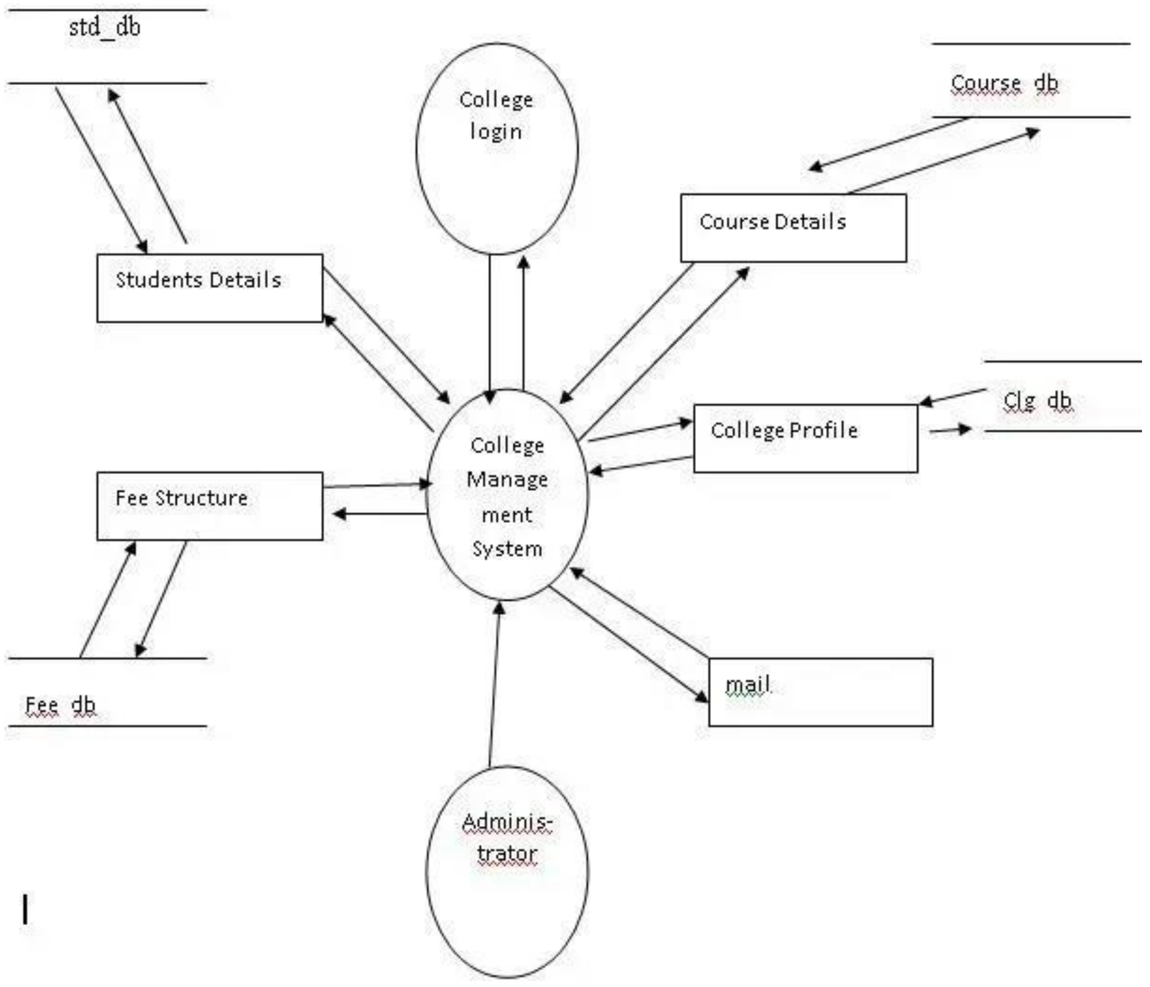
Comparative Analysis**:** A comparative analysis was conducted to assess the advantages of the developed campus management system over existing solutions.

Legacy Systems: The proposed system outperformed legacy systems in terms of efficiency, user-friendliness, and scalability. Its modern architecture based on Spring Boot and React JS offered significant improvements over outdated technologies.

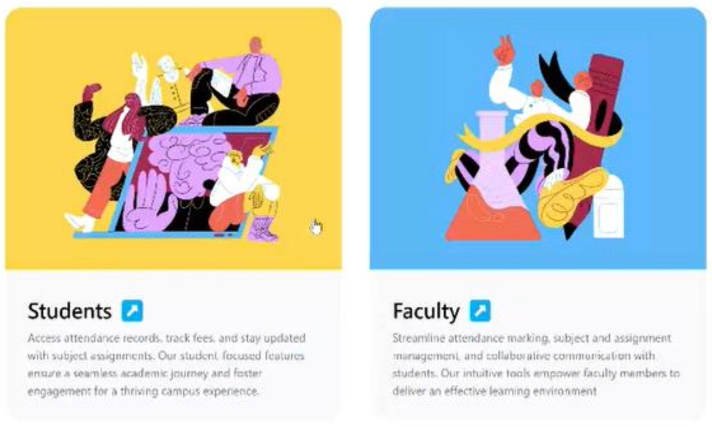
Custom-Built Solutions: The developed system proved to be a viable alternative to custom-built solutions. It provided a comprehensive set of features and functionalities, eliminating the need for costly and time-consuming in-house development.

Commercial Software: The system demonstrated comparable or superior performance to commercial campus management software, while offering the flexibility of customization and cost-effectiveness.

FLOWCHART



RESULT



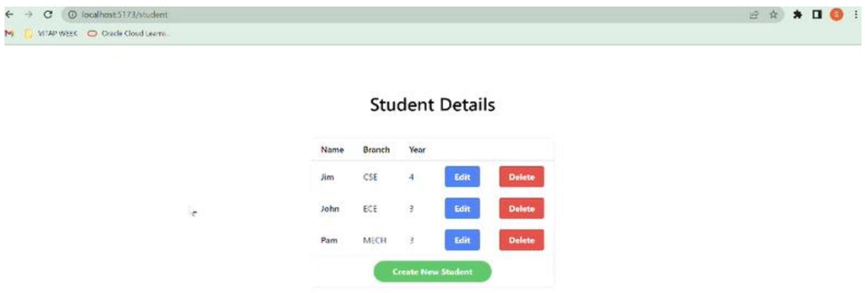
Students

Access attendance records, track fees, and stay updated with subject assignments. Our student-focused features ensure a seamless academic journey and foster engagement for a thriving campus experience.

Faculty

Streamline attendance marking, subject and assignment management, and collaborative communication with students. Our intuitive tools empower faculty members to deliver an effective learning environment

localhost:5173/student

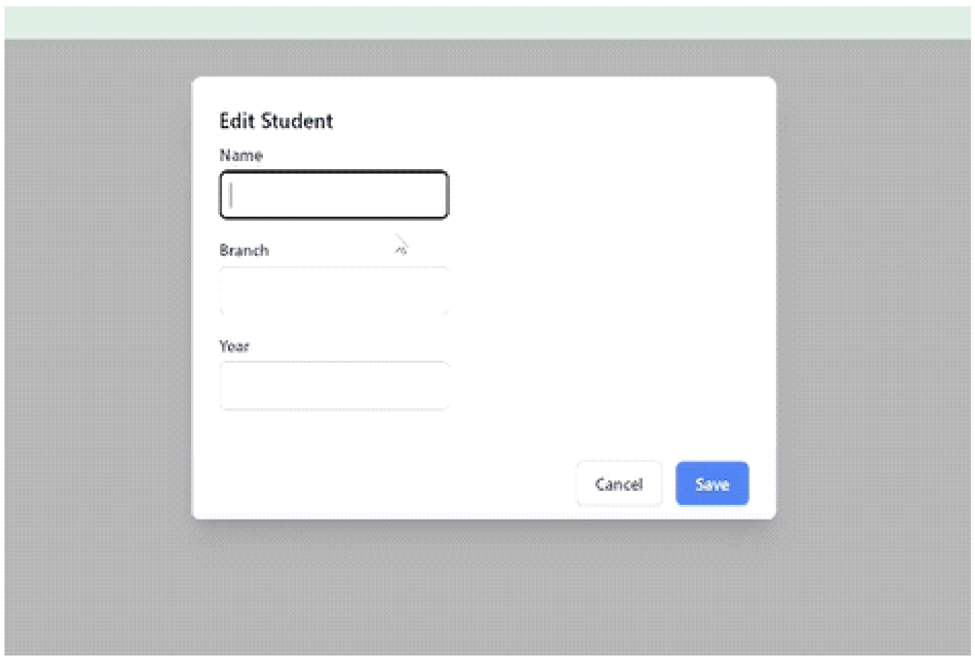


localhost:5173/student

Student Details

Name	Branch	Year		
Jim	CSE	4	Edit	Delete
John	ECE	3	Edit	Delete
Pam	MECH	3	Edit	Delete

Create New Student



Edit Student

Name

Branch

Year

ADVANTAGES & DISADVANTAGES

The benefits of campus management system are the admin can easily retrieve all information related to student and employee. Admin has all the Collective records of students of all the branches. Admin can check all the records of employees of all departments anytime. This system gives easy approach to find the detail information for any student/employee. Using this campus management system, it is very easy to handle all functionality of campus students and employee. This system is beneficial for both students and employees as they can get all information when they need.

APPLICATIONS

Universities and Colleges:

The solution is well-suited for universities and colleges of all sizes, helping them manage their administrative tasks efficiently. It can handle a large volume of student data, course offerings, and administrative processes unique to higher education institutions.

Schools and K-12 Institutions:

The campus management system can also be adapted for use in schools and K-12 institutions. It provides features for managing student records, course enrollment, attendance tracking, and grade management at the K-12 level.

Online Learning Platforms:

The solution can be integrated into online learning platforms to enhance their functionality. It helps manage student enrollment, course offerings, grade tracking, and communication between students and instructors in the online learning environment.

Vocational and Training Institutes:

Vocational and training institutes can utilize the campus management system to streamline their administrative processes. It enables them to manage student records, course schedules, attendance, and assessments for various vocational programs.

Continuing Education Programs: Institutions offering continuing education programs can benefit from the campus management system. It helps in managing the enrollment process, course registration, and tracking the progress of participants in continuing education programs.

Professional Training Centers: The system can be applied in professional training centers, such as those offering certifications or specialized skill development programs. It helps manage course registrations, progress tracking, certification processes, and communication with participants.

Language Schools and Institutes: Language schools and institutes can utilize the campus management system to handle student enrollment, course scheduling, attendance tracking, and grade management specific to language programs.

Study Abroad Programs:

Institutions managing study abroad programs can benefit from the campus management system. It helps manage student applications, course transfers, credit transfers, and communication between home and host institutions.

Research Institutions: Research institutions can utilize the system to manage administrative tasks related to research programs and projects. It provides features for managing research grants, collaboration among researchers, and tracking research outputs.

Customized Educational Programs: The campus management system can be adapted to fit the needs of customized educational programs or specialized training initiatives. It offers flexibility for institutions to tailor the system to their specific program requirements and workflows.

CONCLUSION

The project entitled as **Campus Management System** is the system that deals with the issues related to a particular institution.

This project is successfully implemented with all the features mentioned in system requirements specification.

The application provides appropriate information to users according to the chosen service.

The project is designed keeping in view the day-to-day problems faced by a college.
Deployment of our application will certainly help the college to reduce unnecessary wastage of time in personally going to each department for some information.

FUTURE SCOPE

Online examination module would be introduced to conduct online examination.

Scheduling of the staff. i.e., time table setting of the staff

Further, the faculty can upload the videos of their lectures on to this site and students who had missed those classes can view those videos.

BIBILOGRAPHY

1. Basu, S. (2014). Campus Management System: A Review. International Journal of Engineering and Computer Science, 3(11), 8589-8592.

2. Ahuja, N., & Saini, N. (2019). Design and Development of Campus Management System using Agile Methodology. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 5(2), 491-495.

3. Hossain, M., & Rahman, M. (2020). An Efficient Campus Management System for Educational Institutions. International Journal of Computer Science and Information Security, 18(1), 190-198.

APPENDIX

A. SOURCE CODE FOR SPRINGBOOT

campusmanagement

src/main/java

com.project.campusmanagement.controller

FacultyController.java

```
package com.project.campusmanagement.controller;

import com.project.campusmanagement.model.Faculty;
import com.project.campusmanagement.service.FacultyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/faculty")
@CrossOrigin
```

```
public class FacultyController {
    private final FacultyService facultyService;

    @Autowired
    public FacultyController(FacultyService facultyService) {
        this.facultyService = facultyService;
    }

    @GetMapping("/getAll")
    public List<Faculty> getAllFaculties() {
        return facultyService.getAllFaculties();
    }

    @GetMapping("/get/{id}")
    public ResponseEntity<Faculty> getFacultyById(@PathVariable
Long id) {
        Faculty faculty = facultyService.getFacultyById(id);
        if (faculty != null) {
            return ResponseEntity.ok(faculty);
        }
        return ResponseEntity.notFound().build();
    }

    @PostMapping("/create")
    public ResponseEntity<Faculty> createFaculty(@RequestBody
Faculty faculty) {
        Faculty createdFaculty = facultyService.createFaculty(faculty);
        if (createdFaculty != null) {
            return
ResponseEntity.status(HttpStatus.CREATED).body(createdFaculty);
        }
        return ResponseEntity.badRequest().build();
    }

    @PutMapping("/update/{id}")
    public ResponseEntity<Faculty> updateFaculty(@PathVariable
Long id, @RequestBody Faculty faculty) {
        Faculty updatedFaculty = facultyService.updateFaculty(id,
faculty);
        if (updatedFaculty != null) {
            return ResponseEntity.ok(updatedFaculty);
        }
        return ResponseEntity.notFound().build();
    }

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<Void> deleteFaculty(@PathVariable Long
id) {
        facultyService.deleteFaculty(id);
        return ResponseEntity.noContent().build();
    }
}
```

StudentController.java

```
package com.project.campusmanagement.controller;

import com.project.campusmanagement.model.Student;
import com.project.campusmanagement.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
```

```
@RequestMapping("/student")
@CrossOrigin
public class StudentController {
   @Autowired
   private StudentService studentService;

   @PostMapping("/add")
   public String add(@RequestBody Student student){
      studentService.saveStudent(student);
      return "New student is added";
   }

   @GetMapping("/getAll")
   public List<Student> list(){
      return studentService.getAllStudents();
   }

   @PutMapping("/update/{id}")
   public String update(@PathVariable int id, @RequestBody
Student updatedStudent){
      Student student = studentService.getStudentById(id);

      if (student != null) {
         student.setName(updatedStudent.getName());
         student.setBranch(updatedStudent.getBranch());
         student.setYear(updatedStudent.getYear());

         studentService.saveStudent(student);
         return "Student updated successfully";
      } else {
         return "Student not found";
      }
```

```
    }

    @DeleteMapping("/delete/{id}")
    public String delete(@PathVariable int id) {
        Student student = studentService.getStudentById(id);

        if (student != null) {
            studentService.deleteStudent(student);
            return "Student deleted successfully";
        } else {
            return "Student not found";
        }
    }
}
```

com.project.campusmanagement.model

Faculty.java

```
package com.project.campusmanagement.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ElementCollection;
import java.util.List;

@Entity
public class Faculty {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

    @ElementCollection
    private List<String> courses;

    private String department;

    // Constructors, getters, and setters

    // Constructor without ID for creating new faculty
    public Faculty(String name, List<String> courses, String
department) {
        this.name = name;
        this.courses = courses;
        this.department = department;
    }

    // Empty constructor for JPA
    public Faculty() {
    }

    // Getters and setters

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<String> getCourses() {
        return courses;
    }

    public void setCourses(List<String> courses) {
        this.courses = courses;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }
}
```

Student.java

```
package com.project.campusmanagement.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
```

```
}



@Entity
public class Student {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;
   private String name;
   private String branch;

   private int year;

   public Student() {
   }

   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getBranch() {
        return branch;
    }

    public void setBranch(String branch) {
        this.branch = branch;
    }

    public int getYear(){
        return year;
    }

    public void setYear(int year){
        this.year = year;
    }
}
```

com.project.campusmanagement.repository

FacultyRepository.java

```
package com.project.campusmanagement.repository;

import com.project.campusmanagement.model.Faculty;
import org.springframework.data.jpa.repository.JpaRepository;

public interface FacultyRepository extends JpaRepository<Faculty,
Long> {
}
```

StudentRepository.java

```
package com.project.campusmanagement.repository;

import com.project.campusmanagement.model.Student;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;


@Repository
public interface StudentRepository extends
JpaRepository<Student,Integer> {
}
```

com.project.campusmanagement.service
FacultyService.java

```
package com.project.campusmanagement.service;

import com.project.campusmanagement.model.Faculty;

import java.util.List;

public interface FacultyService {
    List<Faculty> getAllFaculties();

    Faculty getFacultyById(Long id);

    Faculty createFaculty(Faculty faculty);

    Faculty updateFaculty(Long id, Faculty faculty);

    void deleteFaculty(Long id);
}
```

FacultyServiceImpl.java

```
package com.project.campusmanagement.service;

import com.project.campusmanagement.model.Faculty;
import com.project.campusmanagement.repository.FacultyRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class FacultyServiceImpl implements FacultyService {
    private final FacultyRepository facultyRepository;

    @Autowired
    public FacultyServiceImpl(FacultyRepository facultyRepository) {
        this.facultyRepository = facultyRepository;
    }

    @Override
    public List<Faculty> getAllFaculties() {
        return facultyRepository.findAll();
    }

    @Override
    public Faculty getFacultyById(Long id) {
        return facultyRepository.findById(id).orElse(null);
    }

    @Override
    public Faculty createFaculty(Faculty faculty) {
        return facultyRepository.save(faculty);
    }

    @Override
    public Faculty updateFaculty(Long id, Faculty faculty) {
        Faculty existingFaculty = facultyRepository.findById(id).orElse(null);
        if (existingFaculty != null) {
            existingFaculty.setName(faculty.getName());
            existingFaculty.setCourses(faculty.getCourses());
            existingFaculty.setDepartment(faculty.getDepartment());
            return facultyRepository.save(existingFaculty);
        }
        return null;
    }

    @Override
    public void deleteFaculty(Long id) {
        facultyRepository.deleteById(id);
    }
}
```

StudentService.java

```
package com.project.campusmanagement.service;

import com.project.campusmanagement.model.Student;

import java.util.List;

public interface StudentService {
    Student saveStudent(Student student);
    List<Student> getAllStudents();
    Student getStudentById(int id);
    void deleteStudent(Student student);
}
```

StudentServiceImpl.java

```
package com.project.campusmanagement.service;

import com.project.campusmanagement.model.Student;
import com.project.campusmanagement.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentRepository studentRepository;

    @Override
    public Student saveStudent(Student student) {
        return studentRepository.save(student);
    }

    @Override
    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    @Override
    public Student getStudentById(int id) {
        Optional<Student> studentOptional = studentRepository.findById(id);
        return studentOptional.orElse(null);
    }

    @Override
    public void deleteStudent(Student student) {
        studentRepository.delete(student);
    }
}
```

src/main/resources
application.properties

```
#configuration
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/campus
spring.datasource.username=root
spring.datasource.password=root@123
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.1.1</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.project</groupId>
	<artifactId>campusmanagement</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>campusmanagement</name>
	<description>full stack campus management application using spring boot and
reactjs</description>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>com.mysql</groupId>
			<artifactId>mysql-connector-j</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

B. SOURCE CODE FOR REACT JS

Campus-management-system
src/pages
Faculty.jsx

```
import React, { useState, useEffect } from "react";
import { Dialog, Transition } from "@headlessui/react";

function Faculty() {
  const [faculties, setFaculties] = useState([]);
  const [isOpen, setIsOpen] = useState(false);
  const [editedFaculty, setEditedFaculty] = useState(null);
  const [name, setName] = useState("");
  const [department, setDepartment] = useState("");
  const [courses, setCourses] = useState([]);

  useEffect(() => {
    fetchFaculties();
  }, []);

  const openModal = (faculty) => {
    setIsOpen(true);
    setEditedFaculty(faculty);
    setName(faculty.name);
    setDepartment(faculty.department);
    setCourses(faculty.courses);
  };

  const closeModal = () => {
```

```
  setIsOpen(false);
  setEditedFaculty(null);
  setName("");
  setDepartment("");
  setCourses([]);
};

const handleFormSubmit = (event) => {
  event.preventDefault();
  if (editedFaculty) {
    updateFaculty();
  } else {
    createFaculty();
  }
};

const createFaculty = () => {
  const newFaculty = { name, department, courses };

  console.log(newFaculty);

  fetch("http://localhost:8080/faculty/create", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(newFaculty),
  })
    .then((response) => response.text())
    .then((data) => {
      console.log(data); // Response from the server
      fetchFaculties();
      closeModal();
    })
    .catch((error) => {
      console.error("Error:", error);
    });
};

const updateFaculty = () => {
    const updatedFaculty = { name, department, courses };

    fetch(`http://localhost:8080/faculty/update/${editedFaculty
.id}`, {
      method: "PUT",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(updatedFaculty),
    })
      .then((response) => response.text())
      .then((data) => {
        console.log(data); // Response from the server
        fetchFaculties();
        closeModal();
      })
      .catch((error) => {
        console.error("Error:", error);
      });
  };

  const fetchFaculties = () => {
    fetch("http://localhost:8080/faculty/getAll")
      .then((response) => response.json())
      .then((data) => {
        setFaculties(data);
      });
  };

  const deleteFaculty = (faculty) => {
    fetch(`http://localhost:8080/faculty/delete/${faculty.id}`,
{
      method: "DELETE",
    })
      .then((response) => response.text())
      .then((data) => {
        console.log(data); // Response from the server
        fetchFaculties();
      })
      .catch((error) => {
        console.error("Error:", error);
      });
  };

  return (
    <div className="flex justify-center items-center p-10 flex-
col">
      <h1 className="text-3xl font-semibold m-10">Faculty
Details</h1>
      <div className="max-w-3xl">
        <div className="overflow-x-auto rounded-lg border
border-gray-200">
          <table className="min-w-full divide-y-2 divide-gray-
200 bg-white text-sm">
            {/* Table header */}
            <thead className="ltr:text-left rtl:text-right">
              <tr>
                <th className="whitespace-nowrap px-4 py-2
font-medium text-gray-900 text-left">
                  Name
                </th>
                <th className="whitespace-nowrap px-4 py-2
font-medium text-gray-900 text-left">
                  Department
                </th>
                <th className="whitespace-nowrap px-4 py-2
font-medium text-gray-900">
                  Courses
                </th>
              </tr>
            </thead>
            {/* Table body */}
            <tbody className="divide-y divide-gray-200">
              {faculties.map((faculty, index) => (
                <tr key={index}>
                  <td className="whitespace-nowrap px-4 py-2
font-medium text-gray-900">
                    {faculty.name}
                  </td>
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    {faculty.department}
                  </td>
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    {faculty.courses.join(", ")}
                  </td>
                  {/* Edit and delete buttons */}
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    <button
                      className="bg-blue-500 hover:bg-blue-700
text-white font-bold py-2 px-4 rounded"
                      onClick={() => openModal(faculty)}
                    >
                      Edit
                    </button>
                  </td>
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    <button
                      className="bg-red-500 hover:bg-red-700
text-white font-bold py-2 px-4 rounded"
                      onClick={() => deleteFaculty(faculty)}
                    >
                      Delete
                    </button>
                  </td>
                </tr>
              ))}
              {/* Create new faculty button in a new row */}
              <tr>
                <td
                  className="whitespace-nowrap px-4 py-2 text-
gray-700 text-center"
                  colSpan="5"
                >
                  <button
```

```
          className="bg-green-500 hover:bg-green-700
text-white font-bold py-2 px-8 rounded-full"
          onClick={() => openModal()}
         >
          Create New Faculty
         </button>
        </td>
       </tr>
      </tbody>
     </table>
    </div>
   </div>

   {/* Modal */}
   <Transition.Root show={isOpen} as={React.Fragment}>
    <Dialog
     as="div"
     className="fixed z-10 inset-0 overflow-y-auto"
     onClose={closeModal}
    >
     <div className="flex items-center justify-center min-
h-screen pt-4 px-4 pb-20 text-center sm:block sm:p-0">
      <Transition.Child
       as={React.Fragment}
       enter="ease-out duration-300"
       enterFrom="opacity-0"
       enterTo="opacity-100"
       leave="ease-in duration-200"
       leaveFrom="opacity-100"
       leaveTo="opacity-0"
      >
       <Dialog.Overlay className="fixed inset-0 bg-black
opacity-30" />
      </Transition.Child>

      {/* Modal content */}
      <Transition.Child
       as={React.Fragment}
       enter="ease-out duration-300"
       enterFrom="opacity-0 scale-95"
```

```
              enterTo="opacity-100 scale-100"
              leave="ease-in duration-200"
              leaveFrom="opacity-100 scale-100"
              leaveTo="opacity-0 scale-95"
            >
              <div className="inline-block align-bottom bg-
white rounded-lg text-left overflow-hidden shadow-xl transform
transition-all sm:my-8 sm:align-middle sm:max-w-lg sm:w-full">
                <form onSubmit={handleFormSubmit}>
                  <div className="bg-white px-4 pt-5 pb-4 sm:p-
6 sm:pb-4">
                    <div className="sm:flex sm:items-start">
                      <div className="mt-3 text-center sm:mt-0
sm:text-left">
                        <h3
                          className="text-lg leading-6 font-
medium text-gray-900"
                          id="modal-title"
                        >
                          {editedFaculty ? "Edit Faculty" :
"Create Faculty"}
                        </h3>
                        <div className="mt-2">
                          <div className="mb-4">
                            <label
                              htmlFor="name"
                              className="block text-sm font-
medium text-gray-700"
                            >
                              Name
                            </label>
                            <input
                              type="text"
                              id="name"
                              className="mt-1 p-2 border
border-gray-300 rounded-md w-full"
                              value={name}
                              onChange={(e) =>
setName(e.target.value)}
                            />
                          </div>
                          <div className="mb-4">
                            <label
                              htmlFor="department"
                              className="block text-sm font-
medium text-gray-700"
                            >
                              Department
                            </label>
                            <input
                              type="text"
                              id="department"
                              className="mt-1 p-2 border
border-gray-300 rounded-md w-full"
                              value={department}
                              onChange={(e) =>
setDepartment(e.target.value)}
                            />
                          </div>
                          <div className="mb-4">
                            <label
                              htmlFor="courses"
                              className="block text-sm font-
medium text-gray-700"
                            >
                              Courses (separated by commas)
                            </label>
                            <input
                              type="text"
                              id="courses"
                              className="mt-1 p-2 border
border-gray-300 rounded-md w-full"
                              value={courses.join(", ")}
                              onChange={(e) =>
                                setCourses(e.target.value.split
(", "))
                              }
                            />
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                  <div className="bg-gray-50 px-4 py-3 sm:px-6
sm:flex sm:flex-row-reverse">
                    <button
                      type="submit"
                      className="w-full inline-flex justify-
center rounded-md border border-transparent shadow-sm px-4 py-2
bg-indigo-600 text-base font-medium text-white hover:bg-indigo-
700 focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-indigo-500 sm:ml-3 sm:w-auto sm:text-sm"
                    >
                      Save
                    </button>
                    <button
                      type="button"
                      className="mt-3 w-full inline-flex
justify-center rounded-md border border-gray-300 shadow-sm px-4
py-2 bg-white text-base font-medium text-gray-700 hover:bg-
gray-50 focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-indigo-500 sm:mt-0 sm:ml-3 sm:w-auto sm:text-sm"
                      onClick={closeModal}
                    >
                      Cancel
                    </button>
                  </div>
                </form>
              </div>
            </Transition.Child>
          </div>
        </Dialog>
      </Transition.Root>
    </div>
  );
}

export default Faculty;
```

Student.jsx

```
import React, { useState, useEffect } from "react";
import { Dialog, Transition } from "@headlessui/react";

function Student() {
  const [students, setStudents] = useState([]);
  const [isOpen, setIsOpen] = useState(false);
  const [editedStudent, setEditedStudent] = useState(null);
  const [name, setName] = useState("");
  const [branch, setBranch] = useState("");
  const [year, setYear] = useState("");
    const [hnew, sethnew] = useState(false);

  useEffect(() => {
    fetchStudents();
  }, []);

  const openModal = (student) => {
    setIsOpen(true);
    setEditedStudent(student);
    setName(student.name);
    setBranch(student.branch);
    setYear(student.year);
  };

  const closeModal = () => {
    setIsOpen(false);
    setEditedStudent(null);
    setName("");
    setBranch("");
    setYear("");
  };

  const handleFormSubmit = (event) => {
    event.preventDefault();
    if (!hnew) {
      updateStudent();
    } else {
      createStudent();
      sethnew(false);
    }
  };

  const createStudent = () => {
    const newStudent = { name, branch, year };

    fetch("http://localhost:8080/student/add", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(newStudent),
    })
      .then((response) => response.text())
      .then((data) => {
        console.log(data); // Response from the server
        fetchStudents();
        closeModal();
      })
      .catch((error) => {
        console.error("Error:", error);
      });
  };

  const updateStudent = () => {
    const updatedStudent = { name: name, branch: branch, year:
year };
    console.log(updatedStudent);

    fetch(`http://localhost:8080/student/update/${editedStudent
.id}`, {
      method: "PUT",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(updatedStudent),
    })
      .then((response) => response.text())
      .then((data) => {
        console.log(data); // Response from the server
        fetchStudents();
        closeModal();
      })
      .catch((error) => {
        console.error("Error:", error);
      });
  };

  const fetchStudents = () => {
    fetch("http://localhost:8080/student/getAll")
      .then((response) => response.json())
      .then((data) => {
        setStudents(data);
      });
  };

  const deleteStudent = (student) => {
    fetch(`http://localhost:8080/student/delete/${student.id}`,
{
      method: "DELETE",
    })
      .then((response) => response.text())
      .then((data) => {
        console.log(data); // Response from the server
        fetchStudents();
      })
      .catch((error) => {
        console.error("Error:", error);
      });
  };


  return (
    <div className="flex justify-center items-center p-10 flex-
col ">
      <h1 className="text-3xl font-semibold m-10">Student
Details</h1>
      <div className="max-w-3xl">
        <div className="overflow-x-auto rounded-lg border
border-gray-200">
```

```
          <table className="min-w-full divide-y-2 divide-gray-
200 bg-white text-sm">
            {/* Table header */}
            <thead className="ltr:text-left rtl:text-right">
              <tr>
                <th className="whitespace-nowrap px-4 py-2
font-medium text-gray-900 text-left">
                  Name
                </th>
                <th className="whitespace-nowrap px-4 py-2
font-medium text-gray-900 text-left">
                  Branch
                </th>
                <th className="whitespace-nowrap px-4 py-2
font-medium text-gray-900">
                  Year
                </th>
              </tr>
            </thead>
            {/* Table body */}
            <tbody className="divide-y divide-gray-200">
              {students.map((student, index) => (
                <tr key={index}>
                  <td className="whitespace-nowrap px-4 py-2
font-medium text-gray-900">
                    {student.name}
                  </td>
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    {student.branch}
                  </td>
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    {student.year}
                  </td>
                  {/* Edit and delete buttons */}
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    <button
```

```
                      className="bg-blue-500 hover:bg-blue-700
text-white font-bold py-2 px-4 rounded"
                      onClick={() => openModal(student)}
                    >
                      Edit
                    </button>
                  </td>
                  <td className="whitespace-nowrap px-4 py-2
text-gray-700">
                    <button
                      className="bg-red-500 hover:bg-red-700
text-white font-bold py-2 px-4 rounded"
                      onClick={() => deleteStudent(student)}
                    >
                      Delete
                    </button>
                  </td>
                </tr>
              ))}
              {/* Create new student button in a new row */}
              <tr>
                <td
                  className="whitespace-nowrap px-4 py-2 text-
gray-700 text-center"
                  colSpan="5"
                >
                  <button
                    className="bg-green-500 hover:bg-green-700
text-white font-bold py-2 px-8 rounded-full"
                    onClick={() =>{
                        sethnew(true);
                        openModal();
                    }}
                  >
                    Create New Student
                  </button>
                </td>
              </tr>
            </tbody>
          </table>
```

```
        </div>
      </div>

      {/* Modal */}
      <Transition.Root show={isOpen} as={React.Fragment}>
        <Dialog
          as="div"
          className="fixed z-10 inset-0 overflow-y-auto"
          onClose={closeModal}
        >
          <div className="flex items-center justify-center min-
h-screen pt-4 px-4 pb-20 text-center sm:block sm:p-0">
            <Transition.Child
              as={React.Fragment}
              enter="ease-out duration-300"
              enterFrom="opacity-0"
              enterTo="opacity-100"
              leave="ease-in duration-200"
              leaveFrom="opacity-100"
              leaveTo="opacity-0"
            >
              <Dialog.Overlay className="fixed inset-0 bg-black
opacity-30" />
            </Transition.Child>

            {/* Modal content */}
            <Transition.Child
              as={React.Fragment}
              enter="ease-out duration-300"
              enterFrom="opacity-0 scale-95"
              enterTo="opacity-100 scale-100"
              leave="ease-in duration-200"
              leaveFrom="opacity-100 scale-100"
              leaveTo="opacity-0 scale-95"
            >
              <div className="inline-block align-bottom bg-
white rounded-lg text-left overflow-hidden shadow-xl transform
transition-all sm:my-8 sm:align-middle sm:max-w-lg sm:w-full">
                <form >
```

```
                  <div className="bg-white px-4 pt-5 pb-4 sm:p-
6 sm:pb-4">
                    <div className="sm:flex sm:items-start">
                      <div className="mt-3 text-center sm:mt-0
sm:text-left">
                        <h3
                          className="text-lg leading-6 font-
medium text-gray-900"
                          id="modal-title"
                        >
                          Edit Student
                        </h3>
                        <div className="mt-2">
                          <div className="mb-4">
                            <label
                              htmlFor="name"
                              className="block text-sm font-
medium text-gray-700"
                            >
                              Name
                            </label>
                            <input
                              type="text"
                              name="name"
                              id="name"
                              autoComplete="off"
                              defaultValue={editedStudent?.name
}
                              onChange={(e) =>
setName(e.target.value)}
                              className="mt-1 p-2 border
border-gray-300 rounded-md w-full"
                            />
                          </div>
                          <div className="mb-4">
                            <label
                              htmlFor="branch"
                              className="block text-sm font-
medium text-gray-700"
                            >
```

```
                              Branch
                            </label>
                            <input
                              type="text"
                              name="branch"
                              id="branch"
                              autoComplete="off"
                              defaultValue={editedStudent?.bran
ch}
                              onChange={(e) =>
setBranch(e.target.value)}
                              className="mt-1 p-2 border
border-gray-300 rounded-md w-full"
                            />
                          </div>
                          <div className="mb-4">
                            <label
                              htmlFor="year"
                              className="block text-sm font-
medium text-gray-700"
                            >
                              Year
                            </label>
                            <input
                              type="text"
                              name="year"
                              id="year"
                              autoComplete="off"
                              defaultValue={editedStudent?.year
}
                              onChange={(e) =>
setYear(e.target.value)}
                              className="mt-1 p-2 border
border-gray-300 rounded-md w-full"
                            />
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                  <div className="bg-gray-50 px-4 py-3 sm:px-6
sm:flex sm:flex-row-reverse">
                    <button
                      type="submit"
                      onClick={handleFormSubmit}
                      className="w-full inline-flex justify-
center rounded-md border border-transparent shadow-sm px-4 py-2
bg-blue-500 text-base font-medium text-white hover:bg-blue-700
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
blue-500 sm:ml-3 sm:w-auto sm:text-sm"
                    >
                      Save
                    </button>
                    <button
                      type="button"
                      onClick={closeModal}
                      className="mt-3 w-full inline-flex
justify-center rounded-md border border-gray-300 shadow-sm px-4
py-2 bg-white text-base font-medium text-gray-700 hover:bg-
gray-50 focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-indigo-500 sm:mt-0 sm:ml-3 sm:w-auto sm:text-sm"
                    >
                      Cancel
                    </button>
                  </div>
                </form>
              </div>
            </Transition.Child>
          </div>
        </Dialog>
      </Transition.Root>
    </div>
  );
}

export default Student;
```

Home.jsx

```
import React from 'react'
import { Link } from 'react-router-dom';
```

```
function Home() {
  return (
    <div>
      <section className="flex relative bg-[#fff] items-center
justify-center ">
        <div className="relative items-center w-full px-5 py-12
mx-auto lg:px-16 lg:py-32 max-w-7xl md:px-12">
          <div>
            <div className="text-center">
              <span className="w-auto">
                <span className="font-semibold text-[#4354ff]
text-sm uppercase">
                  Campus management system
                </span>
              </span>
              <p className="mt-8 text-3xl font-extrabold
tracking-tight text-black md:text-5xl">
                Welcome to Our
                <br />
                Campus Management System
              </p>
              <p className="max-w-2xl mx-auto mt-4 text-base
lg:text-xl text-slate-500">
                Streamline administrative tasks, enhance
communication, and
                empower campus stakeholders with our
comprehensive Campus
                Management System. From attendance tracking to
fee management,
                we provide a seamless platform for efficient
campus operations.
              </p>
            </div>
          </div>

          <section className="">
            <div className="relative px-4 py-24 mx-auto sm:px-6
md:px-4 lg:max-w-6xl lg:px-20 lg:py-36">
```

```
              <dl className="grid grid-cols-1 gap-10 md:grid-
cols-2 lg:grid-cols-2">
                <Link to="/student">
                  <div className="h-full overflow-hidden bg-
gray-100 rounded-3xl">
                    <div className="">
                      <img
                        src="https://cdn.dribbble.com/users/671
617/screenshots/16792621/media/cad8f9120956f1b984d77effa5f55214
.jpg?compress=1&resize=1600x1200&vertical=center"
                        alt=""
                      />
                    </div>
                    <div className="p-8">
                      <p className="mt-1 text-3xl font-
semibold">
                        Students &#8599;
                      </p>
                      <p className="mt-2 text-gray-500">
                        Access attendance records, track fees,
and stay updated
                        with subject assignments. Our student-
focused features
                        ensure a seamless academic journey and
foster engagement
                        for a thriving campus experience.
                      </p>
                    </div>
                  </div>
                </Link>
                <Link to="/faculty">
                  <div className="h-full overflow-hidden bg-
gray-100 rounded-3xl">
                    <div className="">
                      <img
                        src="https://cdn.dribbble.com/users/671
617/screenshots/16792608/media/d7495ead23b39e366801ac29481b4548
.jpg?compress=1&resize=1600x1200&vertical=center"
                        alt=""
                      />
                    </div>
                    <div className="p-8">
                      <p className="mt-1 text-3xl font-
semibold">
                        Faculty &#8599;
                      </p>
                      <p className="mt-2 text-gray-500">
                        Streamline attendance marking, subject
and assignment
                        management, and collaborative
communication with
                        students. Our intuitive tools empower
faculty members to
                        deliver an effective learning
environment
                      </p>
                    </div>
                  </div>
                </Link>
              </dl>
            </div>
          </section>
        </div>
      </section>
    </div>
  );
}

export default Home
```

App.jsx

```
import { useState } from "react";
import { Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import Faculty from "./pages/Faculty";
import Student from "./pages/Student";

function App() {
  const [count, setCount] = useState(0);
  return (
    <div>
      {/* <Home /> */}
      <Routes>
          <Route path="/" element={<Home />} />
          <Route path="faculty" element={<Faculty />} />
          <Route path="student" element={<Student />} />
      </Routes>
    </div>
  );
}

export default App;
```

main.jsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import "./index.css";
import { BrowserRouter } from "react-router-dom";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```