College Directory Application

Objective:

Develop a College Directory Application to facilitate seamless interaction between students, faculty members, and administrators within a single college. The application will allow users to manage and access personal and academic information efficiently.

User Roles:

1. Student
2. Faculty Member
3. Administrator

Requirements:

1. Login Page

User Interface:

- Simple form with fields for Username, Password, and a role selection dropdown (Student/Faculty Member/Administrator).
- Submit button to log in.

Functionality:

- Validate user credentials against the database.
- Redirect users to their respective dashboards based on their role.
- Implement error handling for incorrect credentials.

2. Student Interface

A. View Personal Profile

- User Interface:
    - Display personal information (name, photo, contact details).
    - Display academic information (courses, grades, attendance).
- Functionality:
    - Fetch and display the student's profile details from the database.

B. Search for Other Students

- User Interface:
    - Search bar with filters for name, department, or year.
    - List view displaying basic details of matching students.
- Functionality:

    - Implement search functionality to query the database.
    - Display search results dynamically.

C. Contact Faculty Advisors

- User Interface:
    - List view of assigned faculty advisors with contact options (email, phone).
- Functionality:
    - Fetch and display advisor details from the database.
    - Provide contact links for email or phone.

3. Faculty Member Interface

A. Manage Class List

- User Interface:
    - List view of students in the faculty member's courses with columns for student names, photos, and contact information.
- Functionality:
    - Fetch and display class list from the database.

B. Update Profile

- User Interface:
    - Form to update office hours, contact email, and phone number.
- Functionality:
    - Validate and save updated profile information to the database.
    - Ensure students can view the updated profile.

4. Administrator Interface

A. Manage Student and Faculty Records

- User Interface:
    - Form to add, update, or remove student and faculty records.
    - List view of records with options for CRUD operations.
- Functionality:
    - Implement CRUD operations for student and faculty records in the database.
    - Ensure changes are reflected immediately in the directory.

B. Dashboard with Graphs

- User Interface:
    - Interface to view graphical representations of key data points (e.g., student enrollment trends, faculty course loads).
    - Include charts and graphs for visual representation of data.
- Functionality:
    - Fetch and aggregate data for graphical display.
    - Provide interactive elements for data visualization.

Technical Requirements:

Frontend:

- Use React or HTML/CSS/JavaScript.
- Implement state management for handling form inputs and API responses.

Backend:

- Use Java with the Spring Boot framework.
- Create RESTful API endpoints for:
    - User authentication and authorization.
    - CRUD operations for profiles and records.
    - Fetching student and faculty details.
    - Dashboard data aggregation.
- Implement data validation and error handling.

Authentication:

- Implement user authentication and authorization using JWT or session-based authentication.
- Protect API endpoints to ensure that only authorized users can access certain functionalities.
- Role-based access control for advanced security.

Database:

- Use PostgreSQL.

Entities

1. User

- Fields:
    - id: Unique identifier for each user.
    - username: Login name for the user.
    - password: Encrypted password for the user.
    - role: Role of the user, defined using an enum (Student, FacultyMember, Administrator).
    - name: Full name of the user.
    - email: Contact email of the user.

    - phone: Contact phone number of the user.
- Explanation: This entity stores the basic login credentials and contact information for all users in the system. The role field, using an enum, helps in differentiating between students, faculty members, and administrators.

2. StudentProfile

- Fields:
    - user_id: Foreign key linking to the User entity.
    - photo: URL or path to the student's profile photo.
    - department_id: Foreign key linking to the Department entity.
    - year: Year of study (e.g., Freshman, Sophomore, etc.).
- Explanation: This entity extends the User entity to include additional information specific to students, such as their academic details, department, and year of study.

3. FacultyProfile

- Fields:
    - user_id: Foreign key linking to the User entity.
    - photo: URL or path to the faculty member's profile photo.
    - department_id: Foreign key linking to the Department entity.
    - office_hours: Office hours for the faculty member.
- Explanation: This entity extends the User entity to include additional information specific to faculty members, such as their department and office hours.

4. AdministratorProfile

- Fields:
    - user_id: Foreign key linking to the User entity.
    - photo: URL or path to the administrator's profile photo.
    - department_id: Foreign key linking to the Department entity.
- Explanation: This entity extends the User entity to include additional information specific to administrators, such as their department.

5. Course

- Fields:
    - id: Unique identifier for each course.
    - title: Title of the course.
    - description: Description of the course content.
    - department_id: Foreign key linking to the Department entity.
    - faculty_id: Foreign key linking to the FacultyProfile entity.

- Explanation: This entity stores information about the courses offered by the college, including the title, description, department, and the faculty member responsible for the course.

6. Enrollment

- Fields:
  - id: Unique identifier for each enrollment record.
  - student_id: Foreign key linking to the StudentProfile entity.
  - course_id: Foreign key linking to the Course entity.
- Explanation: This entity represents the many-to-many relationship between students and courses, indicating which students are enrolled in which courses.

7. Department

- Fields:
  - id: Unique identifier for each department.
  - name: Name of the department.
  - description: Description of the department.
- Explanation: This entity stores information about the various departments within the college. It is referenced by the StudentProfile, FacultyProfile, AdministratorProfile, and Course entities to ensure proper association of users and courses with their respective departments.

Relationships

- User to StudentProfile/FacultyProfile/AdministratorProfile:
  - One-to-One relationship where each user has a corresponding profile in either the StudentProfile, FacultyProfile, or AdministratorProfile entities.
- FacultyProfile to Course:
  - One-to-Many relationship where one faculty member can teach multiple courses.
- Course to StudentProfile (through Enrollment):
  - Many-to-Many relationship facilitated by the Enrollment entity, where students can enroll in multiple courses, and each course can have multiple students.
- Department to StudentProfile/FacultyProfile/AdministratorProfile/Course:
  - One-to-Many relationship where each department can have multiple students, faculty members, administrators, and courses associated with it.

-

SQL

```
-- Enum for Role
CREATE TYPE Role AS ENUM ('STUDENT', 'FACULTY_MEMBER', 'ADMINISTRATOR');

-- User Table
CREATE TABLE User (
   id SERIAL PRIMARY KEY,
   username VARCHAR(50) UNIQUE NOT NULL,
   password VARCHAR(255) NOT NULL,
   role Role NOT NULL,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   phone VARCHAR(15)
);

-- Department Table
CREATE TABLE Department (
   id SERIAL PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   description TEXT
);

-- StudentProfile Table
CREATE TABLE StudentProfile (
   user_id INTEGER PRIMARY KEY,
   photo VARCHAR(255),
   department_id INTEGER NOT NULL,
   year VARCHAR(50),
   FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE,
   FOREIGN KEY (department_id) REFERENCES Department(id) ON DELETE CASCADE
);

-- FacultyProfile Table
CREATE TABLE FacultyProfile (
   user_id INTEGER PRIMARY KEY,
   photo VARCHAR(255),
   department_id INTEGER NOT NULL,
   office_hours VARCHAR(255),
   FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE,
   FOREIGN KEY (department_id) REFERENCES Department(id) ON DELETE CASCADE
);
```

```
-- AdministratorProfile Table
CREATE TABLE AdministratorProfile (
   user_id INTEGER PRIMARY KEY,
   photo VARCHAR(255),
   department_id INTEGER NOT NULL,
   FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE,
   FOREIGN KEY (department_id) REFERENCES Department(id) ON DELETE CASCADE
);

-- Course Table
CREATE TABLE Course (
   id SERIAL PRIMARY KEY,
   title VARCHAR(100) NOT NULL,
   description TEXT,
   department_id INTEGER NOT NULL,
   faculty_id INTEGER NOT NULL,
   FOREIGN KEY (department_id) REFERENCES Department(id) ON DELETE
CASCADE,
   FOREIGN KEY (faculty_id) REFERENCES FacultyProfile(user_id) ON DELETE
CASCADE
);

-- Enrollment Table
CREATE TABLE Enrollment (
   id SERIAL PRIMARY KEY,
   student_id INTEGER NOT NULL,
   course_id INTEGER NOT NULL,
   FOREIGN KEY (student_id) REFERENCES StudentProfile(user_id) ON DELETE
CASCADE,
   FOREIGN KEY (course_id) REFERENCES Course(id) ON DELETE CASCADE
);

-- Insert some data into Department
INSERT INTO Department (name, description) VALUES
('Computer Science', 'Department of Computer Science'),
('Mathematics', 'Department of Mathematics'),
('Physics', 'Department of Physics'),
('Chemistry', 'Department of Chemistry');
```

Evaluation Criteria:

- Functionality: Does the application meet all the specified requirements?
- Code Quality: Is the code well-organized, readable, and maintainable?
- UI/UX Design: Is the user interface intuitive and visually appealing?
- Innovation: Are there any additional features that enhance the application?
- Presentation: How well is the solution presented and demonstrated?