## Problem Statement:

The current system for managing universities is out-of-date and deficient in key characteristics that are required to handle the intricate responsibilities involved in managing a contemporary institution. A comprehensive system for administering universities is required so that responsibilities like managing course catalogs, scheduling classes, student registration, employer management, job posting, and student employment history may be managed.

The university requires a solution that makes it simple for students to sign up for classes, for employers to advertise job opportunities and recruit applicants, and for both sides to give feedback on how well students, courses, and instructors are performing. The institution also wishes to keep track of alumni's work histories, including promotions and the influence of their instructor courses on their achievements.

The suggested system should be a desktop Java Swing application with a modular architecture, each module concentrating on a certain use case. Additionally, the system needs to be user-friendly, safe, and scalable in order to support the university's projected future development.

## Solution Proposed:

### Administrator/Department Head:

- Manage the course catalog: add and delete courses

### Manage the course schedule:

- create a new schedule for a new term, add course offers for courses, and assign teachers.
- Manage student course registrations: view and manage student registrations.
- Manage employers: add new employers and view the employer list.
- View and generate reports on student performance, course impact, and professor's role in the success of the job.

### Teacher:

- View and manage course schedule: add course offers for courses and view schedule for the current term.
- View and manage student course registrations: view and manage student registrations for courses they are teaching.
- View and manage employer list: view available jobs posted by employers.
- Provide feedback on student performance, course impact, and employer feedback.

### Student:

- Register for courses every term.
- View available jobs posted by employers.
- Apply for jobs posted by employers and wait for the employer to choose them.
- Provide feedback on courses, professors, and employers.

### Employer:

- Post jobs for students to apply to
- Choose students for the job based on their application.
- Provide feedback on student performance.

### Alumni:

- Update employment history and link to employer, student, and course offer of the highest impact
- Assign a weight to each employment object to capture the value of promotions (e.g., +1 for each promotion)
- Provide feedback on courses, professors, and employers.
- Note: These are just some possible responsibilities for each role based on the given input. The actual responsibilities may vary depending on the specific requirements and constraints of the system being developed.

## Class Design Decisions:

1. **Department class:** This class should contain the department's name, description, and a list of courses offered by the department. It should have methods to add and remove courses from the list.

2. **Course class:** This class should contain the course's name, description, department, and a list of scheduled course offerings. It should have methods to add and remove scheduled course offerings and to assign a teacher to a course offering.

3. **Teacher class:** This class should contain the teacher's name, contact information, and a list of courses they are qualified to teach. It should have methods to add and remove courses from the list.

4. **Student class:** This class should contain the student's name, contact information, and a list of courses they are registered for. It should have methods to add and remove courses from the list and to view the student's registration history.

5. Employer class: This class should contain the employer's name, contact information, and a list of job postings. It should have methods to add and remove job postings.

6. **Job posting class:** This class should contain the job title, description, employer, and a list of applications from students. It should have methods to add and remove applications and to select a student for the job.

7. **Alumni class:** This class should contain the alum's name, contact information, and a list of employment history. It should have methods to add and remove employment history entries and to view the alum's employment history.

8. **Employment History class:** This class should contain the employer, student, course offering, and the number of promotions the alum receive.

## Opinion and Conclusion:

- In conclusion, the problem statement's description of the university management system is a difficult undertaking that needs a thorough solution. A scalable and simple-to-maintain solution can be created by leveraging Java Swing and a modular MVC architecture when creating desktop applications.

- The suggested class design choices above offer a strong framework for controlling the various entities included in the university management system. Furthermore, the proposed approach makes it possible to implement all the use cases mentioned in the problem statement.

- A full university management system will be made available because of the suggested solution, making it simple to manage the course catalogue, schedule classes, register students, manage employers, post jobs, and track students' employment histories. It will increase the effectiveness of university administration, strengthen the relationship between students and employers, and boost user experience overall.