University Ranking Model

- Garima Choudhary (002104897)
- Priyanka Ahuja (001573684)
- Priya Singh (002951836)

Introduction:

Education plays an extremely important role in structuring a person’s life. Education provided by the schools, colleges, and universities must be top-notch to be able to make a difference in the world. Everyone wants to get admissions in top-ranked schools and universities. It is critical for institutions to be aware of the educational quality they give.

Universities may use software engineering approaches to construct a performance assessment system that allows them to assess the quality of the education they provide their pupils. Feedback from alumni and students is taken into consideration assess the educational system's contributing aspects, such as teachers and courses to their development in the five recent years. This ranking model will be able to help students and their parents to plan based on the rankings of the universities: department and course wise as per the alumni feedback.

Proposed Solution:

Our main goal is to develop a performance measurement system that will allow us to assess the quality of education we provide to students, so that future students will be better able to choose courses based on their interests and relevant university rankings.

This design will include various components as present in the university currently. On the basis, of the current structure the universities will be ranked and displayed for the any incoming student. The design will include a set of characteristics for assessing the quality of education provided by universities, such as graduation rate, relevant courses offered, faculty details, alumni comments, and so on, and institutions will be cumulatively ranked based on these aspects. The dashboard will feature admin login, student login, faculty login and ranking system components, which will allow the admin team to analyze performance for future students based on the alumni details and their employment history.

We will majorly take student’s employment and courses relation into consideration to rank the university performance. We have categorized the students placed in various companies categorically. Also, based on the courses choices famous between the students, the students can rate the courses as good, bad, or excellent based on their experience during the teaching. This feedback can help students to choose a relevant course under particular professor and ease their process in planning. This design will also enable students to decide where they want to go for their studies comparing the growth of the graduates contributed by the faculty and the courses.

We will try to model using the UML diagrams i.e., sequence diagram and class diagram to help explain the solution and the classes to be implemented to build this ranking system efficiently. Various charts based on the students employed and companies they are placed in are shown too. Also, charts showing popular course choices in students is displayed.

Model Assumptions:

1. The elements that influence a course's rating have been highlighted in relation to the university's ranking. Any unstated condition(s) and/or parameters would restrict the scope of the university course ranking model and evaluation.
2. The criteria determining the university model's ranking and course rating are the factors combining the roles and duties of students, faculty, and employers.
3. Alumni students provide complete and accurate details in the student portal to evaluate the performance of the university.

Class and its details:

1. PersonDirectory: This class consists of list of all the Person class which will include all the person details
   a. newPerson(String id): Person – This method takes the person id as an input and will return the type of person i.e., student, employer, faculty.
   b. findPerson(String id): Person : This method will find the person based on the id.
2. Person: This class consists of id as an attribute which will have person id.
   a. getPersonId(): String – This method will return the id of the person
   b. isMatch(String id): Boolean – this method will check if the person is a student, a faculty or an employer based on the id
3. StudentDirectory: This class will include the list of students and is linked to the department. This extends the Person class.
   a. newStudentProfile(Person p): StudentProfile – This method will return new student profile when a student is added.
   b. findStudent(String id): StudentProfile – this function will find a student using the id.
4. StudentProfile: This class will include student details from the Person class, the grades from the transcripts and his employment history
   a. isMatch(String id): boolean – this method will return true if the id passed is a student ID
   b. getTranscript(): Transcript – This method will return the transcript for the particular student
   c. getCourseLoadBySemester(String semester): CourseLoad – This method will return the course details for the student semester wise
   d. getCurrentCourseLoad(): CourseLoad – This function is used to get the current course details for the student
   e. newCourseLoad(String s): CourseLoad – This method will add a new course to the student profile

    f. getCourseList(): ArrayList<SeatAssignment> - This function will get all the course list and return the list of seats assigned

5. FacultyDirectory: This class will include the list of teachers and is linked to the department. This extends the Person class
    a. newFacultyProfile(Person p): FacultyProfile – This method will return new faculty profile when a new teacher is added.
    b. getTopProfessor(): FacultyProfile – This function will get the top professor based on the student ranking by using and comparing with the getProfAverageOverallRating() in the FacultyProfile class
    c. findTeachingFaculty(String id):FacultyProfile - this function will find a teacher using the id.
6. Faculty: This class will include all the faculty details and the list of faculty assignment
    a. getProfAverageOverallRating(): double – This method will return the total rating of the professor extracted based on the faculty assignment
    b. assignAsTeacher(CourseOffer c): FacultyAssignment : this method is assigning the professor to the particular course for the semester
    c. getCourseOffer(Stirng courseid): FacultyProfile : With the help of the course id the faculty teaching that course will be returned
    d. isMatch(String id): Boolean - this method will return true if the id passed is a faculty ID
7. FacultyAssignment: This is relation class between the course and the faculty. This class assigns the faculty to the course. This is where the ranking for the professor will be calculated.
    a. getRating(): double – This function will fetch the rating for the course taught by the professor
    b. setProfRating(double r) - This function will fetch and add new ratings for the course taught by the professor
    c. getFacultyProfile(): FacultyProfile – This method will display the teacher’s profile.
8. EmployerDirectory: This class will include the list of employers and is linked to the department. This extends the Person class
    a. newEmployerProfile(String s): Employer – This method will add a new employer details
    b. findEmployer(String id): Employer – This function will find an employer using the id.
9. Employer: This class consists of the name of the employer and list of all the employment details the employer offers
    a. isMatch(String id): Boolean - this method will return true if the id passed is an employer ID

10. Employment: This class includes attribute such as the profile, employer details of the student, no of offers, etc
    a. isEmployed(String id): Boolean: This method will return true if the student is employed
11. EmploymentHistory: This class will include the list of all the old employers for the student.
    a. nextemployment(String job): Employment – This method will return the employment details when the student gets more offers from different companies
12. Department: This class includes all the catalog details of the person, student, faculty, courses, employer, department name, degree.
    a. addCoreCourse(Course c) – This function is add the details of the core courses only
    b. addElectiveCourse(Course c) – This function is add the details of the core courses only
    c. getPersonDirectory(): PersonDirectory – This method will return all the people under this department
    d. getStudentDirectory(): StudentDirectory – This method will return all the students under this department
    e. newCourseSchedule(String semester): CourseSchedule – This method will add schedule for the course for the semester
    f. getCourseSchedule(String semester): CourseSchedule - This method will return schedule for the course for the semester
    g. getCourseCatalog(): CourseSchedule - This method will return all the courses under this department
    h. calculateRevenuesBySemester(String semester): int – This function will calculate the total revenue for the courses, semester wise
    i. registerForAClass(String studentid, String cn, String semester) – This method will register the student for the course under this department
13. CourseCatalog: This class consists of lists of all the courses
    a. getCourseList(): ArrayList<Course> - This method will return all the courses and its details in the catalog.
    b. getCoursesByNumber(String n): Course - This method will return all the courses and its details in the catalog based on the course number.
    c. newCourse(String n, String nm, int c): Course – This function will add the new course details in the department.
14. Course: This class includes all the course details such as course number, name, price and course credits
    a. getCourseNumber(): String – This will fetch the course number of the course

    b. getCoursePrice(): int - This will fetch the price of the course
    c. getCredits(): int - This will fetch the credit hours of the course
    d. getCourseName(): String - This will fetch the name of the course

15. CourseOffer: This will include the course, faculty and the seat details
    a. AssignAsTeacher(FacultyProfile f) - this method is assigning the professor to the particular course for the semester
    b. getFacultyProfile(): FacultyProfile – This will return the faculty assigned to the course
    c. getCourseNumber(): String - This will fetch the course number of the course
    d. generateSeats(int n) – This method will generate the number of seats as passed in the input
    e. getEmptySeat(): Seat – this method will return the empty seat, if any in the course
    f. assignEmptySeat(CourseLoad c): SeatAssignment – This function will assign the empty seat for the course
    g. getTotalCourseRevenues(): int – This method will return the total price of the courses together
    h. getCreditHours(): int - This will return the credit hours of the course
16. CourseSchedule: This class will return all the courses and its schedules semester wise
    a. newCourseOffer(String n): CourseOffer – This will add new course offered if not present
    b. getCourseOfferByNumber(String n): CourseOffer – This will fetch the course details using the course number
    c. calculateTotalRevenues(): int - This method will return the total price of the courses
17. Transcripts: This class contains metric such as all the grades and the courses of the student
    a. getCourseLoadBySemester(String semester): CourseLoad – This method will return the course details for the student semester wise
    b. getCurrentCourseLoad(): CourseLoad – This function is used to get the current course details for the student
    c. newCourseLoad(String s): CourseLoad – This method will add a new course to the student profile
    d. getStudentTotalScore(): float – This method returns the total score of the student he has received in all the subjects
    e. getCourseList(): ArrayList<SeatAssignment> - This function will get all the course list taken by the student throughout the degree
18. Degree: This class includes list of core courses and elective courses

a. addCoreCourse(Course c): boolean - This will add new core courses offered if not present
b. addElectiveCourse(Course c): boolean - This will add new elective course offered if not present
c. isStudentReadyToGraduate(StudentProfile s): boolean – This method will check if the student has taken the core courses and if yes, will return true
d. validateCoreClasses(ArrayList<SeatAssignment> a):boolean – This method validates if the core classes are being assigned to all the students
e. isCoreSatisfied(ArrayList<SeatAssignment> a, Course c): boolean - This method will check if the student has taken the core courses
f. getTotalElectiveCoursesTaken(ArrayList<SeatAssignment> a): int - This method will return the total elective courses taken by the student
g. isElectiveSatisfied(SeatAssignment): boolean - This method will check if the student has taken the particular elective courses

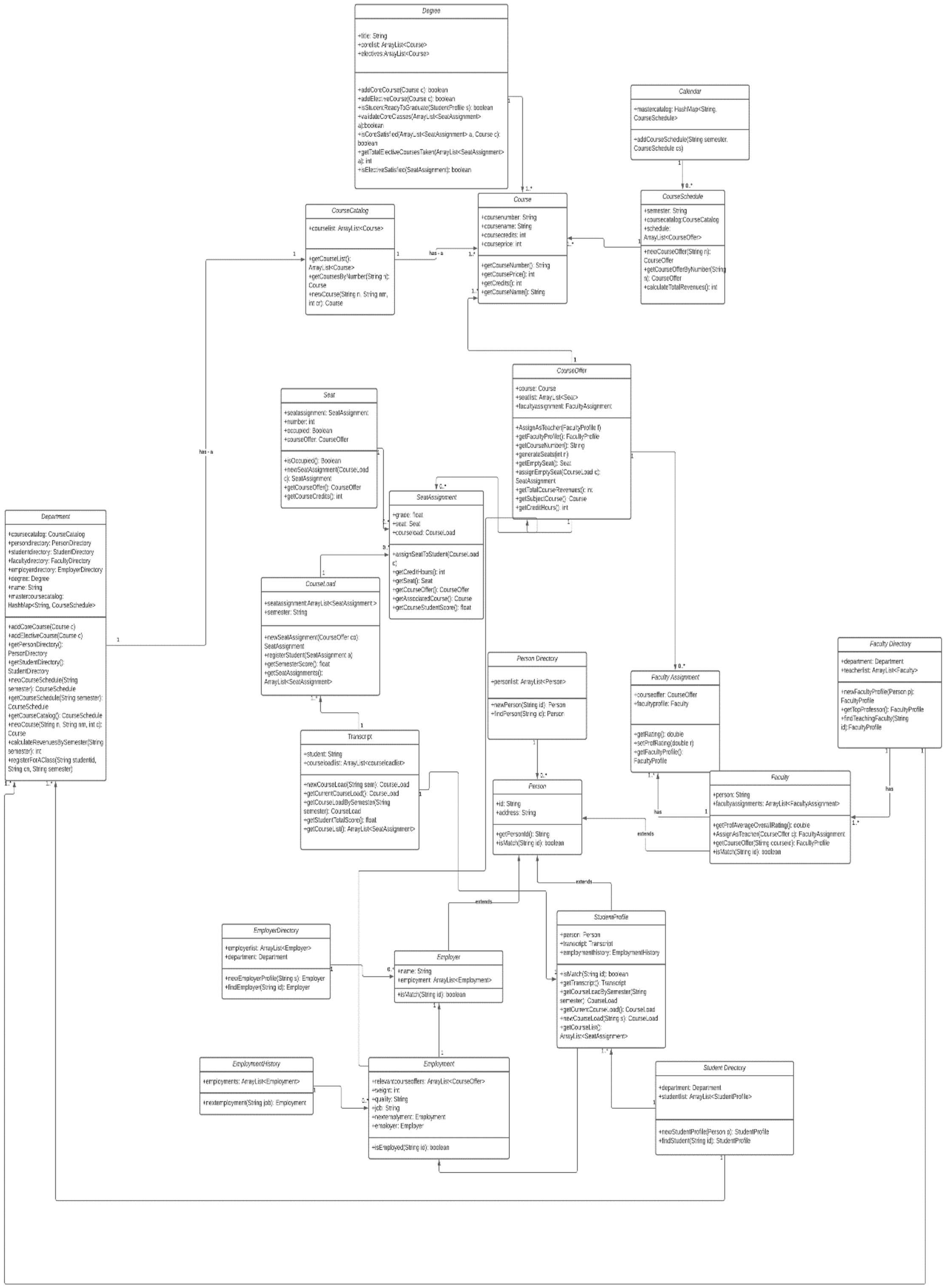
19. Calendar: This class will map all the courses to its schedule
    a. addCourseSchedule(String semester, CourseSchedule cs) – This method will create a new course schedule each semester for the course
20. CourseLoad: This class includes the list of seat assignment details for every semester
    a. newSeatAssignment(CourseOffer co): SeatAssignment – This method will add new seats for the course
    b. registerStudent(SeatAssignment a) – This method will assign a student for a particular course if the seat is available
    c. getSemesterScore(): float – This method will fetch the semester score based on the seat
    d. getSeatAssignments(): ArrayList<SeatAssignment> - This method will fetch all the seats
21. Seat: This class will include course details, availability and seat number
    a. isOccupied(): Boolean – This method will return true if the seat is occupied
    b. newSeatAssignment(CourseLoad c): SeatAssignment – This method will assign new seat for the courses.
    c. getCourseOffer(): CourseOffer – This method will fetch the course details based on the seat
    d. getCourseCredits(): int - This will return the credit hours of the course
22. SeatAssignment: This class includes the Seat details, the course and the grades
    a. assignSeatToStudent(CourseLoad c) – This method will assign the empty seats to the students trying to register
    b. getSeat(): Seat – This method will return the seat details for the seat assignment

c. getAssociatedCourse(): Course - This method will fetch the course details based on the seat assigned

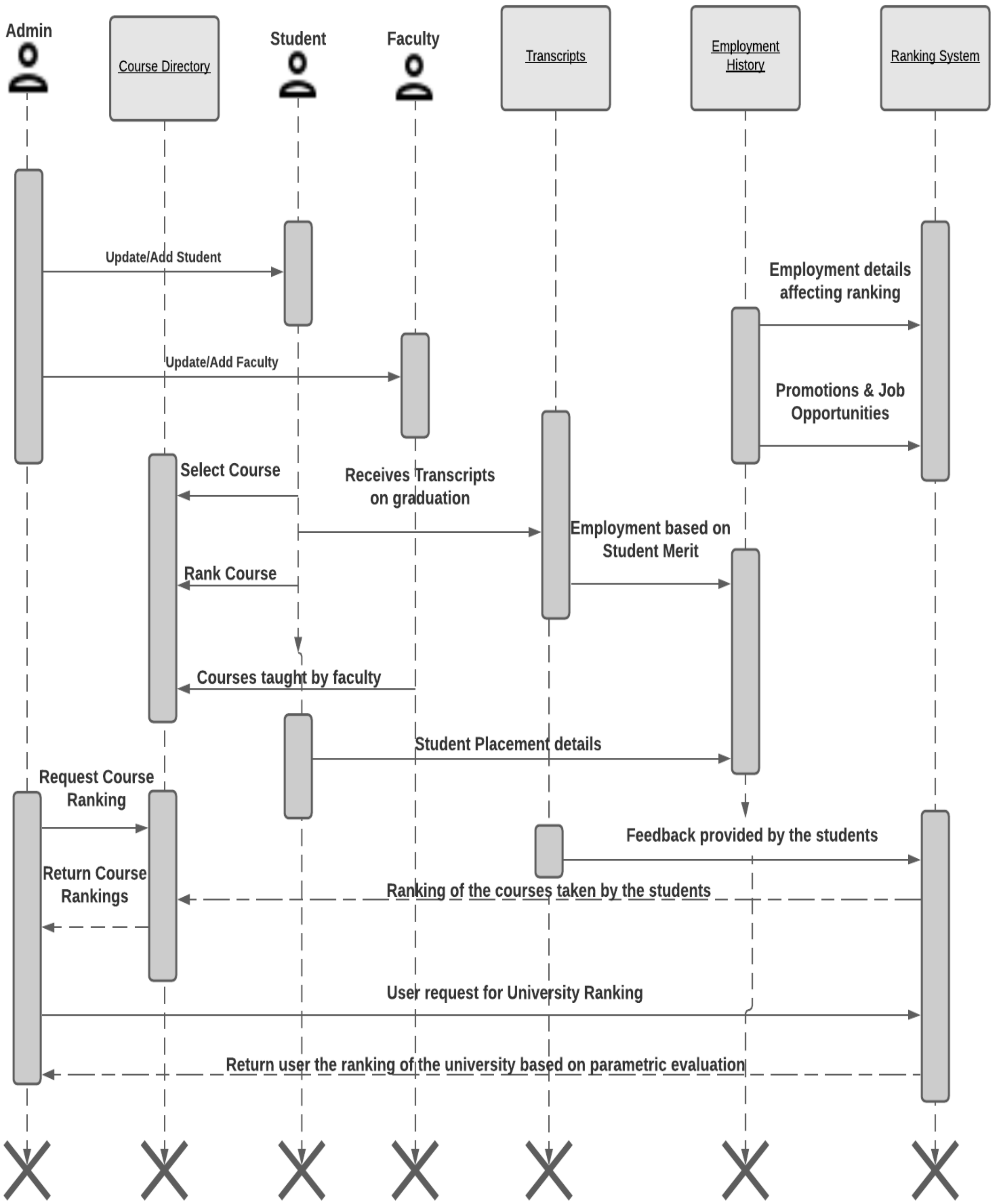
d. getCourseStudentScore(): float – This method will calculate the student score for the course by multiplying the grade and the credit hours.

UML Diagram:

1. Class Diagram

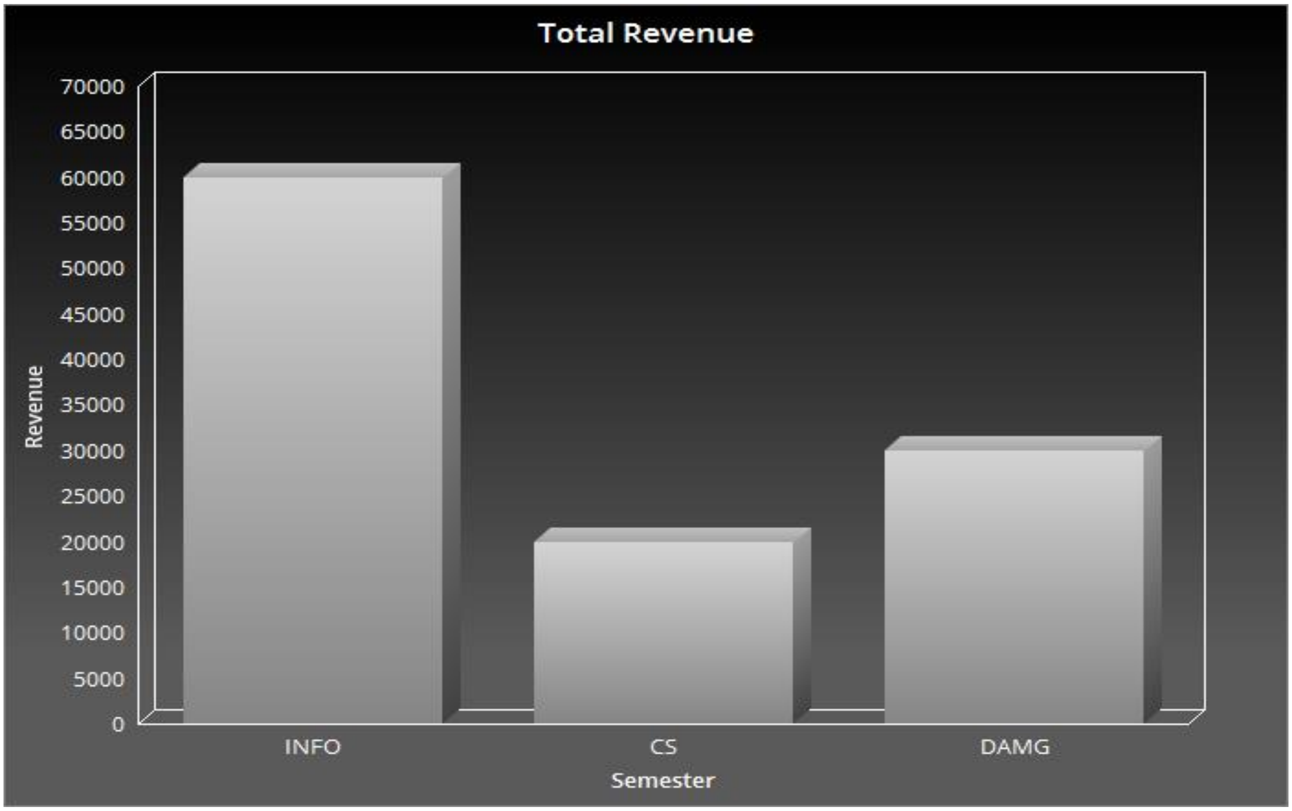

Degree
+title: String
+corelist: ArrayList<Course>
+electives:ArrayList<Course>
+addCoreCourse(Course c): boolean
+addElectiveCourse(Course c): boolean
+isStudentReadyToGraduate(StudentProfile s): boolean
+validateCoreClasses(ArrayList<SeatAssignment> a):boolean
+isCoreSatisfied(ArrayList<SeatAssignment> a, Course c): boolean
+getTotalElectiveCoursesTaken(ArrayList<SeatAssignment> a): int
+isElectiveSatisfied(SeatAssignment): boolean
Calendar
+mastercatalog: HashMap<String, CourseSchedule>
+addCourseSchedule(String semester, CourseSchedule cs)
CourseCatalog
+courselist: ArrayList<Course>
+getCourseList(): ArrayList<Course>
+getCoursesByNumber(String n): Course
+newCourse(String n, String nm, int cr): Course
Course
+coursenumber: String
+coursename: String
+coursecredits: int
+courseprice: int
+getCourseNumber(): String
+getCoursePrice(): int
+getCredits(): int
+getCourseName(): String
CourseSchedule
+semester: String
+coursecatalog:CourseCatalog
+schedule: ArrayList<CourseOffer>
+newCourseOffer(String n): CourseOffer
+getCourseOfferByNumber(String n): CourseOffer
+calculateTotalRevenues(): int
has - a
has - a
1
1..*
0..*
CourseOffer
+course: Course
+seatlist: ArrayList<Seat>
+facultyassignment: FacultyAssignment
+AssignAsTeacher(FacultyProfile f)
+getFacultyProfile(): FacultyProfile
+getCourseNumber(): String
+generateSeats(int n)
+getEmptySeat(): Seat
+assignEmptySeat(CourseLoad c): SeatAssignment
+getTotalCourseRevenues(): int
+getSubjectCourse(): Course
+getCreditHours(): int
Seat
+seatassignment: SeatAssignment
+number: int
+occupied: Boolean
+courseOffer: CourseOffer
+isOccupied(): Boolean
+newSeatAssignment(CourseLoad c): SeatAssignment
+getCourseOffer(): CourseOffer
+getCourseCredits(): int
SeatAssignment
+grade: float
+seat: Seat
+courseload: CourseLoad
+assignSeatToStudent(CourseLoad c)
+getCreditHours(): int
+getSeat(): Seat
+getCourseOffer(): CourseOffer
+getAssociatedCourse(): Course
+getCourseStudentScore(): float
Department
+coursecatalog: CourseCatalog
+persondirectory: PersonDirectory
+studentdirectory: StudentDirectory
+facultydirectory: FacultyDirectory
+employerdirectory: EmployerDirectory
+degree: Degree
+name: String
+mastercoursecatalog: HashMap<String, CourseSchedule>
+addCoreCourse(Course c)
+addElectiveCourse(Course c)
+getPersonDirectory(): PersonDirectory
+getStudentDirectory(): StudentDirectory
+newCourseSchedule(String semester): CourseSchedule
+getCourseSchedule(String semester): CourseSchedule
+getCourseCatalog(): CourseSchedule
+newCourse(String n, String nm, int c): Course
+calculateRevenuesBySemester(String semester): int
+registerForAClass(String studentid, String cn, String semester)
CourseLoad
+seatassignment:ArrayList<SeatAssignment>
+semester: String
+newSeatAssignment(CourseOffer co): SeatAssignment
+registerStudent(SeatAssignment a)
+getSemesterScore(): float
+getSeatAssignments(): ArrayList<SeatAssignment>
Transcript
+student: String
+courseloadlist: ArrayList<courseloadlist>
+newCourseLoad(String sem): CourseLoad
+getCurrentCourseLoad(): CourseLoad
+getCourseLoadBySemester(String semester): CourseLoad
+getStudentTotalScore(): float
+getCourseList(): ArrayList<SeatAssignment>
Person Directory
+personlist: ArrayList<Person>
+newPerson(String id): Person
+findPerson(String id): Person
Faculty Assignment
+courseoffer: CourseOffer
+facultyprofile: Faculty
+getRating(): double
+setProfRating(double r)
+getFacultyProfile(): FacultyProfile
Faculty Directory
+department: Department
+teacherlist: ArrayList<Faculty>
+newFacultyProfile(Person p): FacultyProfile
+getTopProfessor(): FacultyProfile
+findTeachingFaculty(String id):FacultyProfile
Person
+id: String
+address: String
+getPersonId(): String
+isMatch(String id): boolean
Faculty
+person: String
+facultyassignments: ArrayList<FacultyAssignment>
+getProfAverageOverallRating(): double
+AssignAsTeacher(CourseOffer c): FacultyAssignment
+getCourseOffer(String courseid): FacultyProfile
+isMatch(String id): boolean
has
has
extends
extends
extends
EmployerDirectory
+employerlist: ArrayList<Employer>
+department: Department
+newEmployerProfile(String s): Employer
+findEmployer(String id): Employer
Employer
+name: String
+employment: ArrayList<Employment>
+isMatch(String id): boolean
StudentProfile
+person: Person
+transcript: Transcript
+employmenthistory: EmploymentHistory
+isMatch(String id): boolean
+getTranscript(): Transcript
+getCourseLoadBySemester(String semester): CourseLoad
+getCurrentCourseLoad(): CourseLoad
+newCourseLoad(String s): CourseLoad
+getCourseList(): ArrayList<SeatAssignment>
EmploymentHistory
+employments: ArrayList<Employment>
+nextemployment(String job): Employment
Employment
+relevantcourseoffers: ArrayList<CourseOffer>
+weight: int
+quality: String
+job: String
+nextemplyment: Employment
+employer: Employer
+isEmployed(String id): boolean
Student Directory
+department: Department
+studentlist: ArrayList<StudentProfile>
+newStudentProfile(Person p): StudentProfile
+findStudent(String id): StudentProfile

2. Sequence Diagram

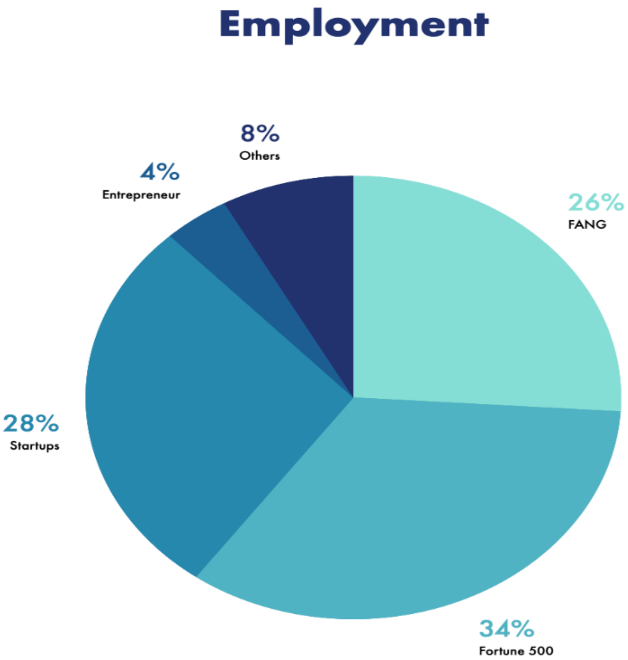


Admin
Course Directory
Student
Faculty
Transcripts
Employment History
Ranking System
Update/Add Student
Employment details affecting ranking
Update/Add Faculty
Promotions & Job Opportunities
Select Course
Receives Transcripts on graduation
Employment based on Student Merit
Rank Course
Courses taught by faculty
Student Placement details
Request Course Ranking
Feedback provided by the students
Return Course Rankings
Ranking of the courses taken by the students
User request for University Ranking
Return user the ranking of the university based on parametric evaluation

Charts:

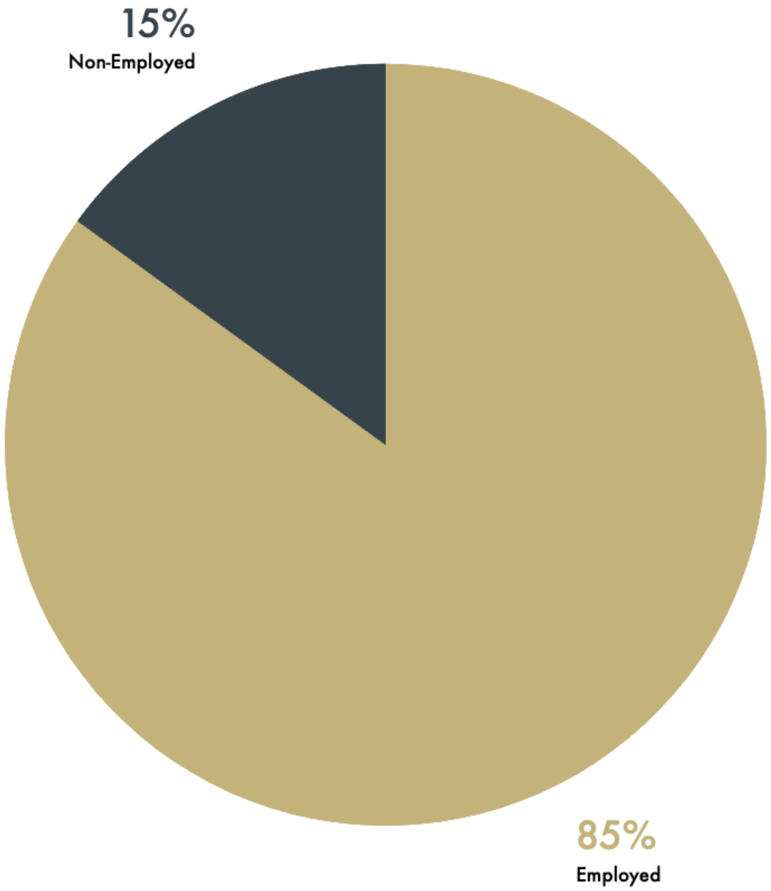


Revenue collected as per course selection



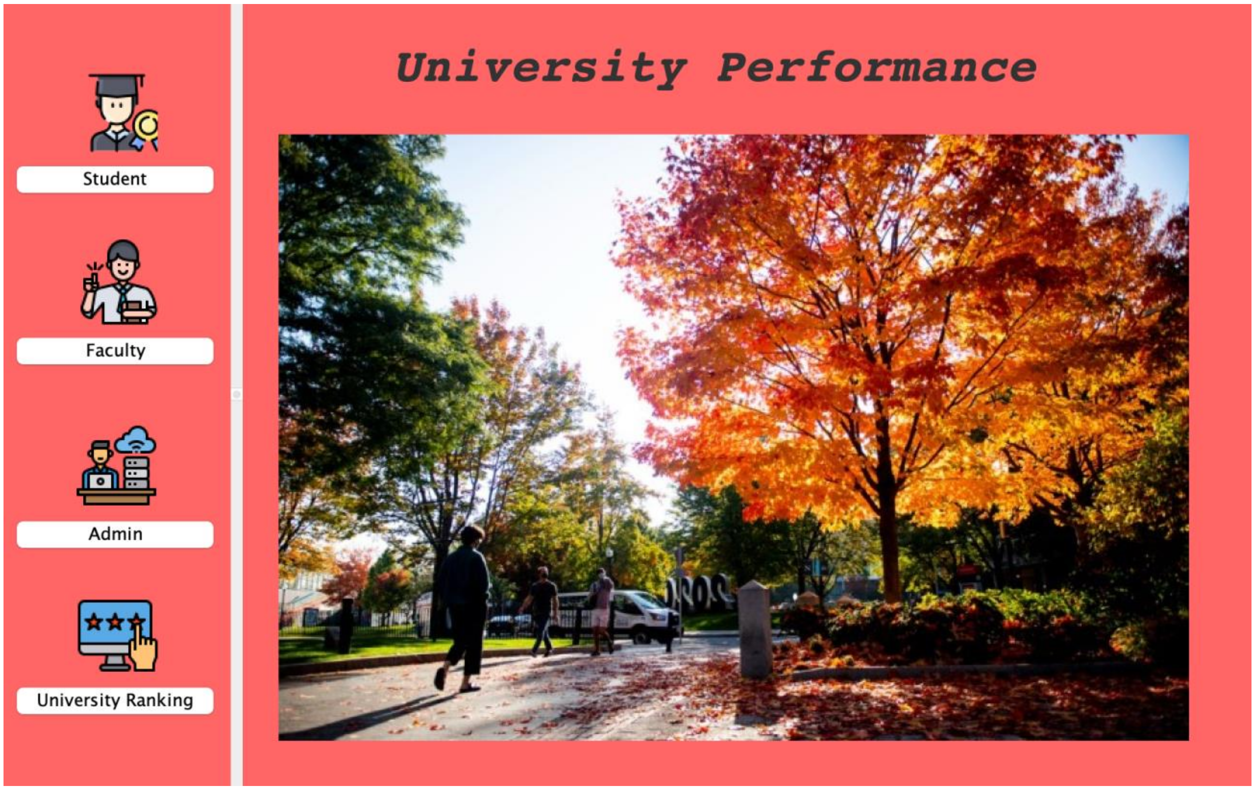
University student employment in company categorically (in percentage)

Employed

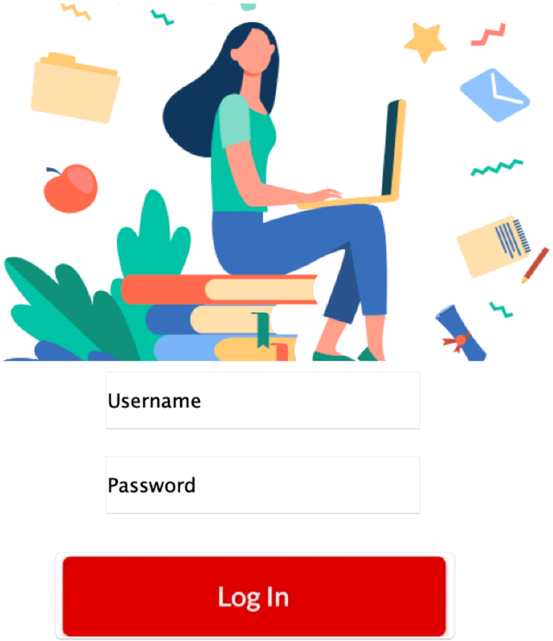


Total percentage of students employed in last 5 years

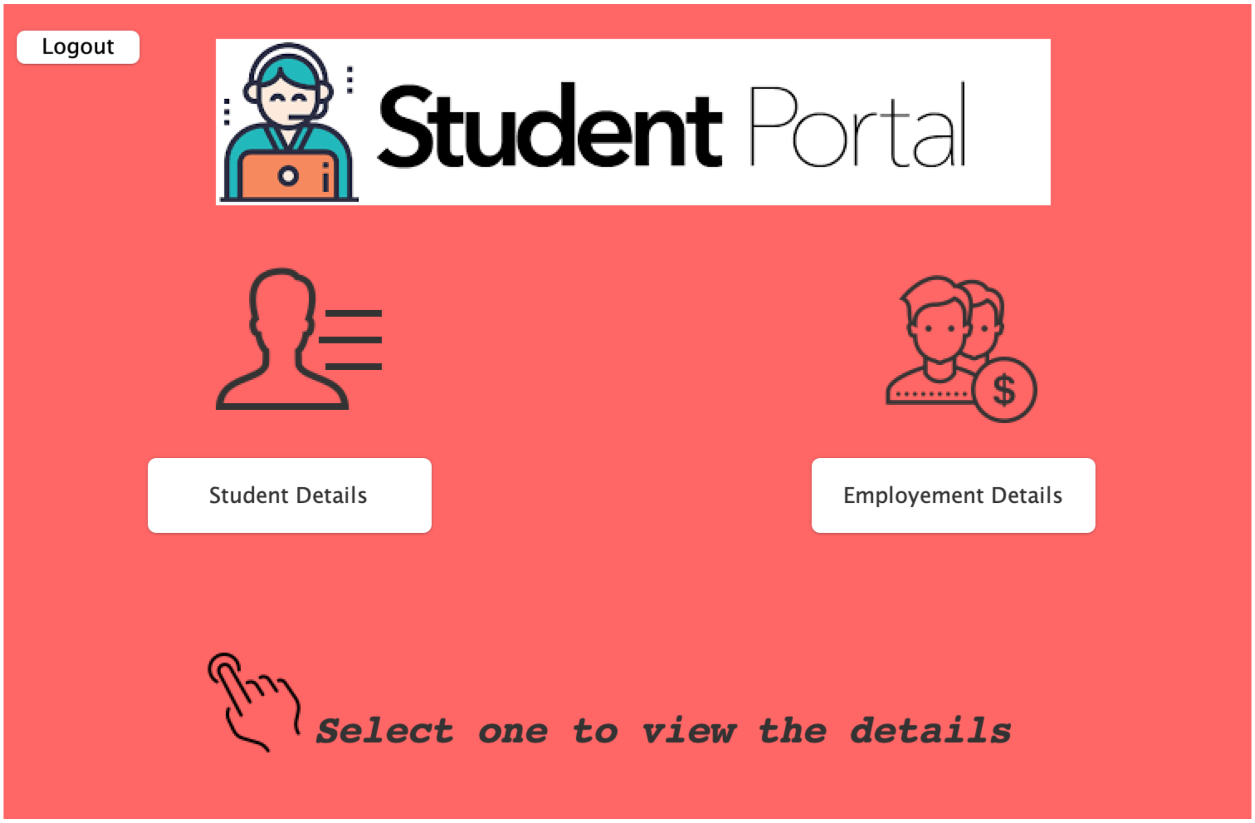
Dashboard Screens:



Main Screen

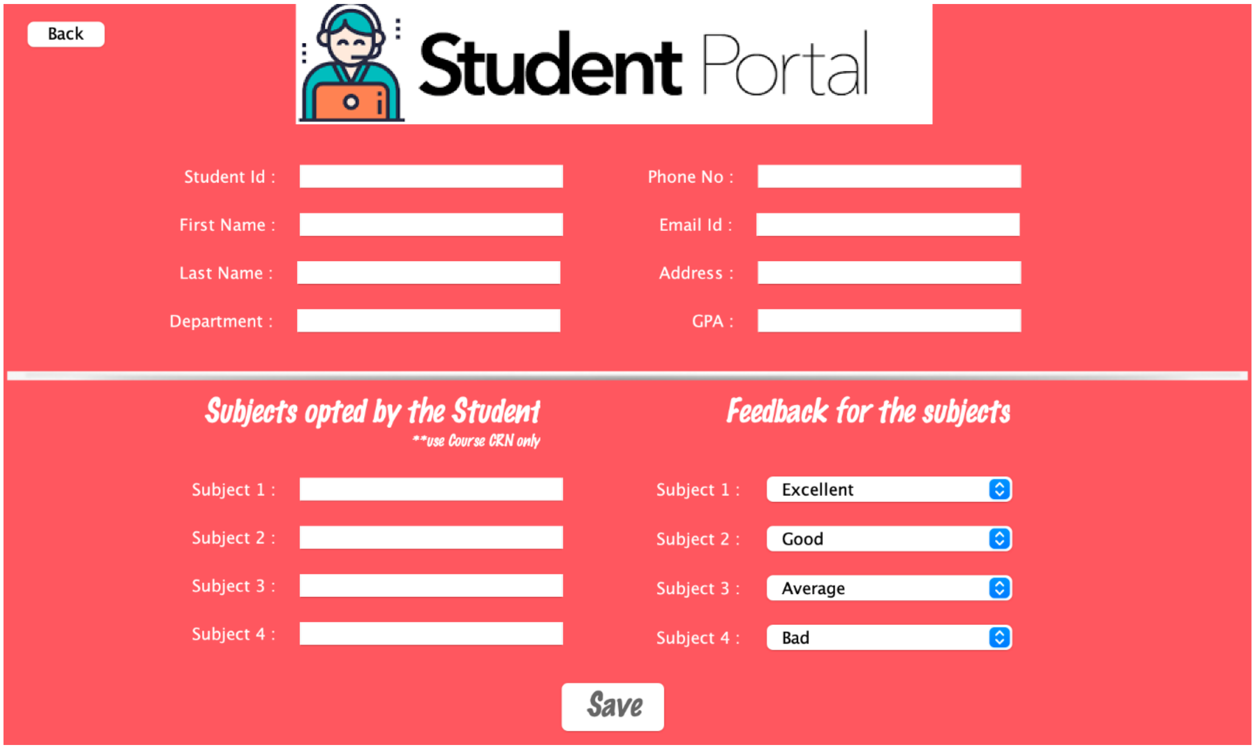


Student Login



Select one to view the details

Student Portal



Student Id :

First Name :

Last Name :

Department :

Phone No :

Email Id :

Address :

GPA :

Subjects opted by the Student

**use Course CRN only

Subject 1 :

Subject 2 :

Subject 3 :

Subject 4 :

Feedback for the subjects

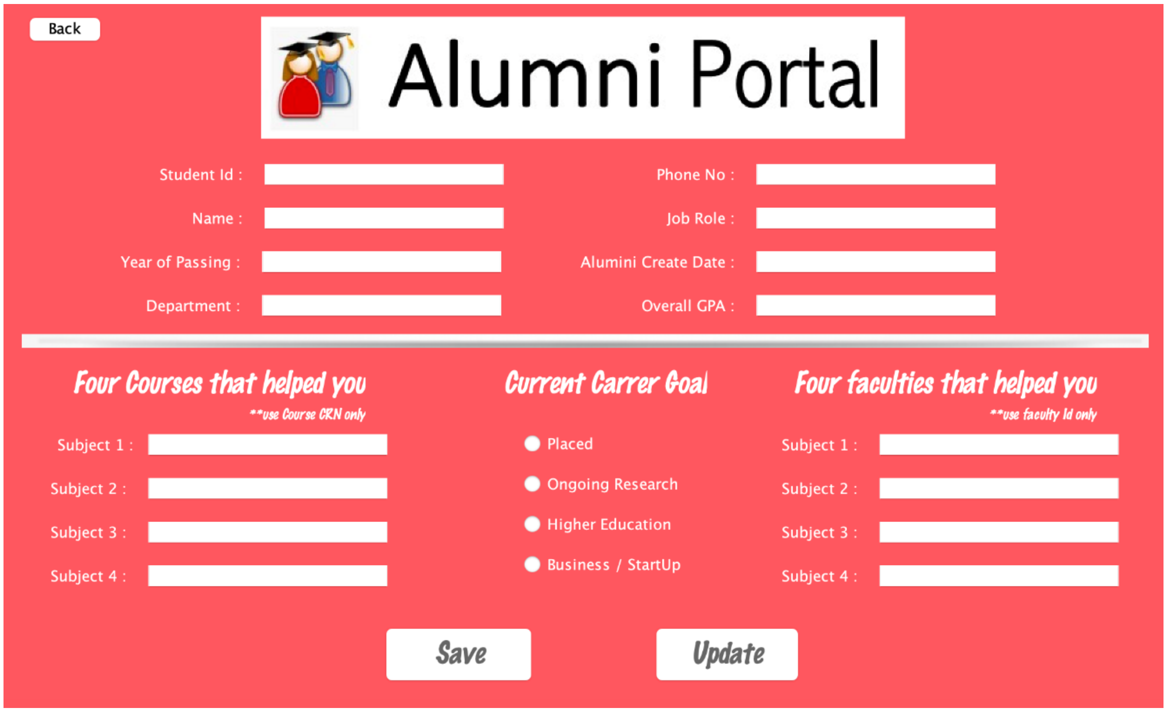
Subject 1 : Excellent

Subject 2 : Good

Subject 3 : Average

Subject 4 : Bad

Add/view/feedback screen for students



Back

Alumni Portal

Student Id :

Name :

Year of Passing :

Department :

Phone No :

Job Role :

Alumini Create Date :

Overall GPA :

Four Courses that helped you

**use Course CRN only

Subject 1 :

Subject 2 :

Subject 3 :

Subject 4 :

Current Carrer Goal

○ Placed

○ Ongoing Research

○ Higher Education

○ Business / StartUp

Four faculties that helped you

**use faculty Id only

Subject 1 :

Subject 2 :

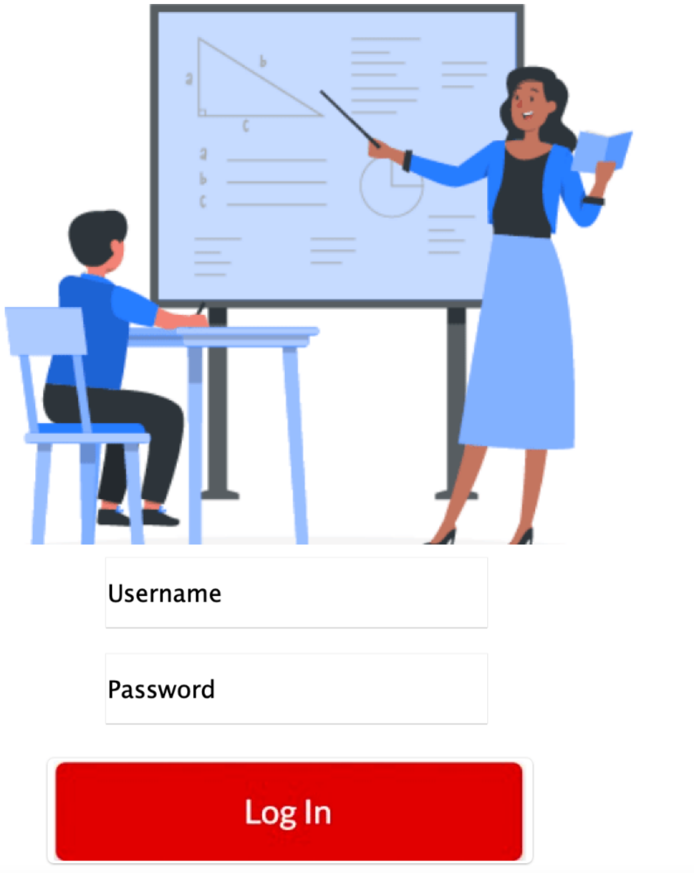
Subject 3 :

Subject 4 :

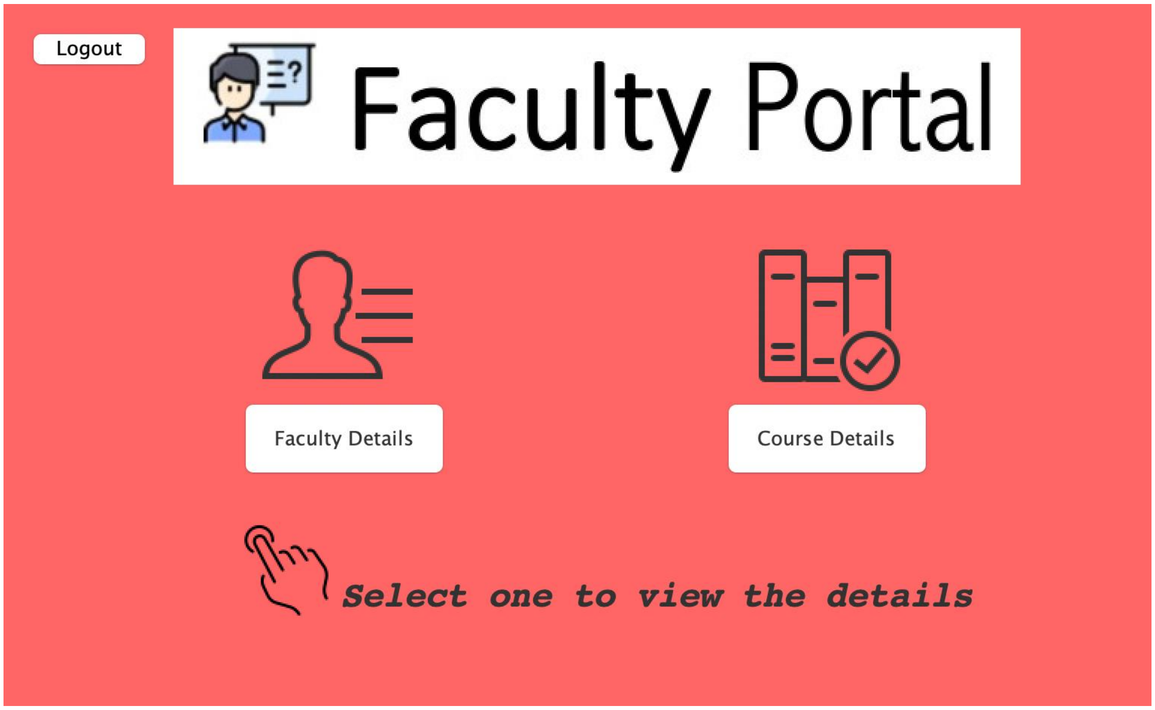
Save

Update

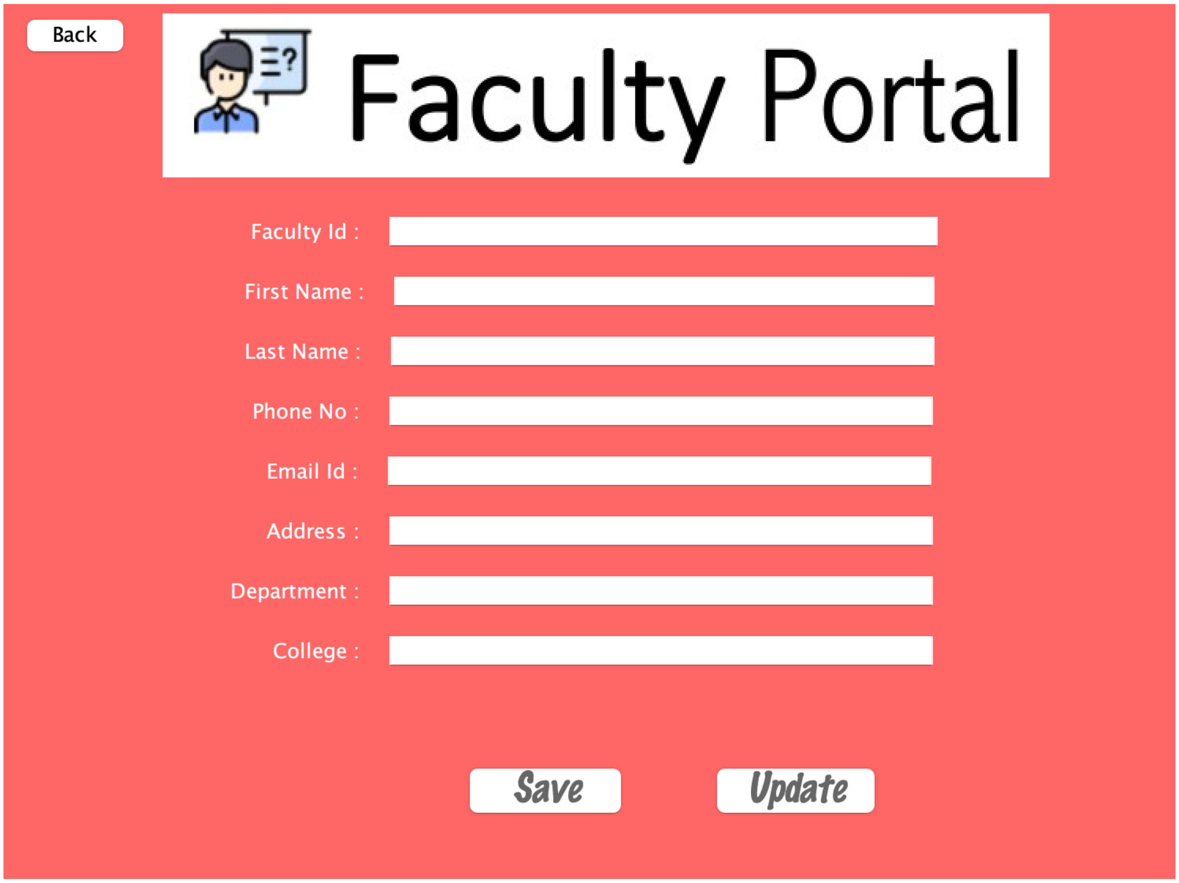
Student Employment Details



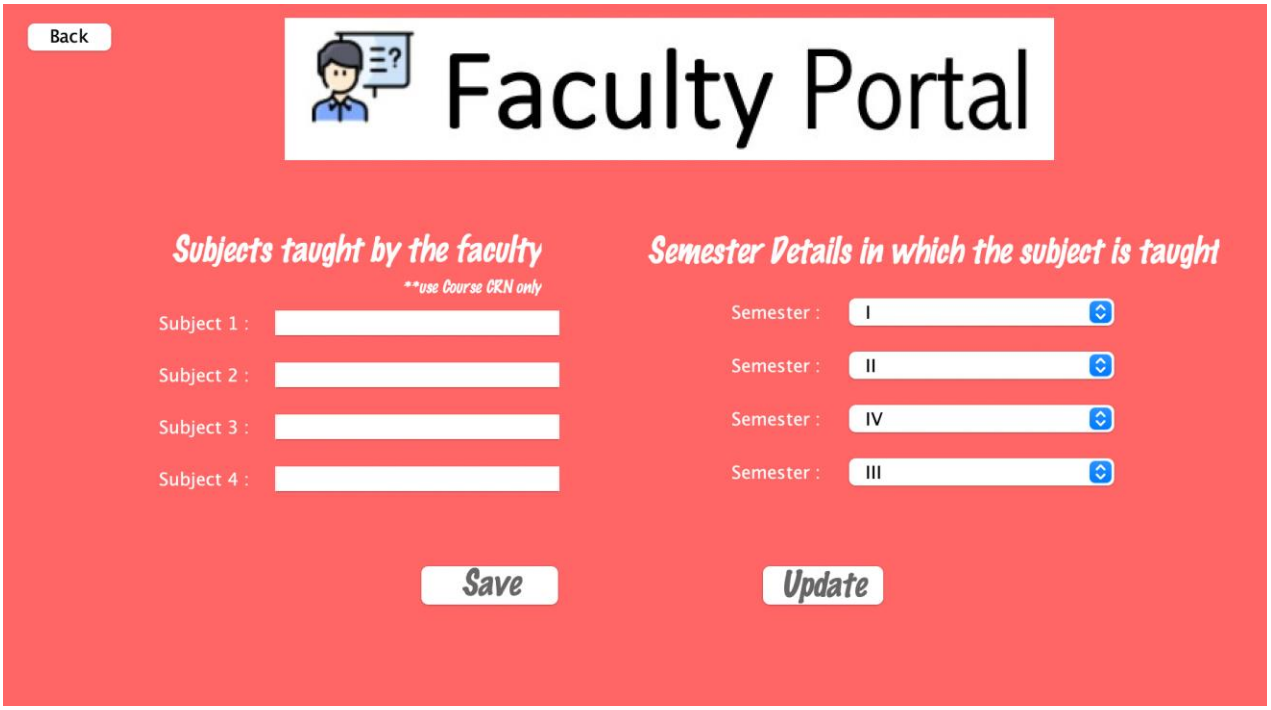
Faculty Login



Faculty Portal



Faculty Details



Subjects taught by the faculty

**use Course CRN only

Subject 1 :

Subject 2 :

Subject 3 :

Subject 4 :

Semester Details in which the subject is taught

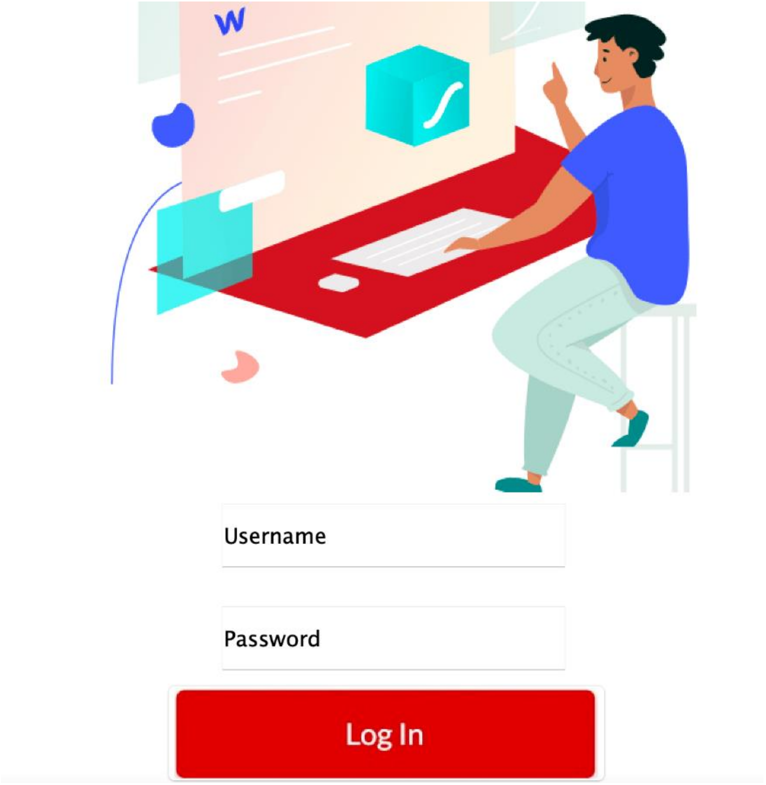
Semester : I

Semester : II

Semester : IV

Semester : III

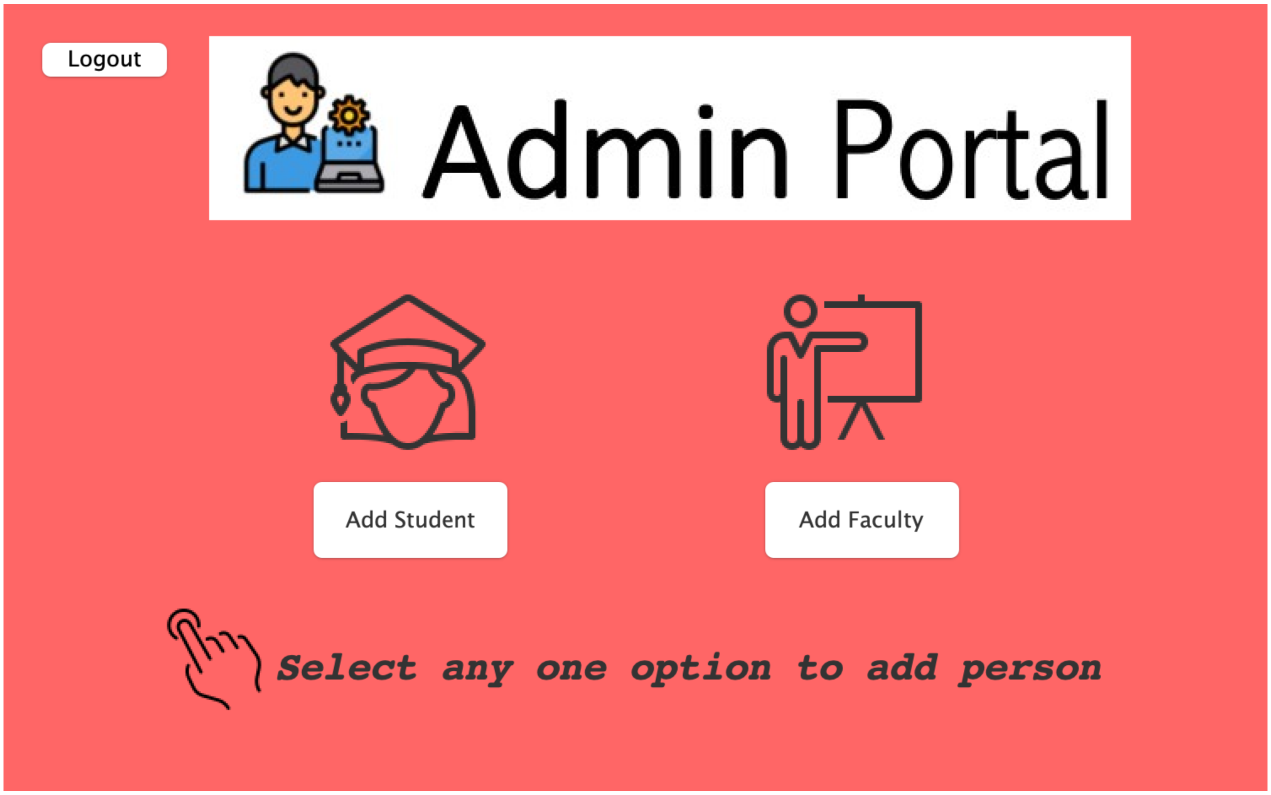
Faculty course details



Username

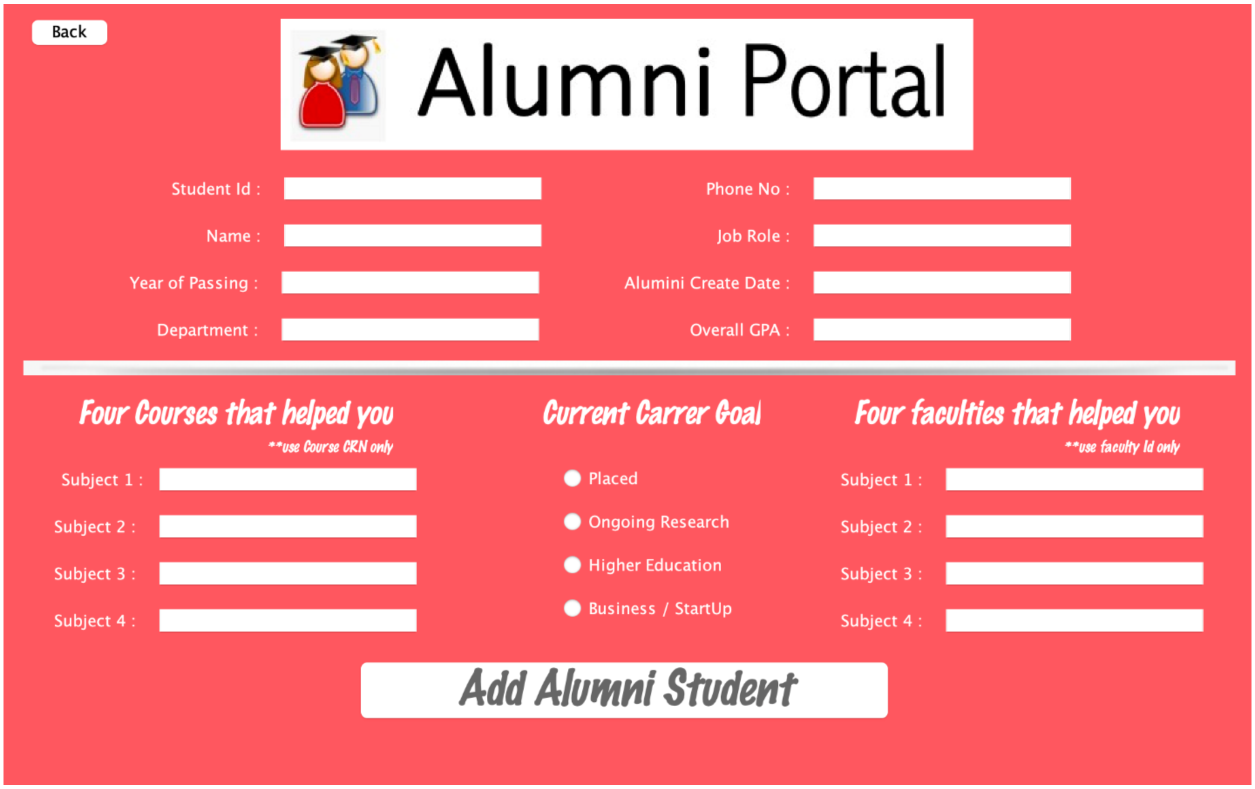
Password

Admin Login



Select any one option to add person

Admin Portal



Student Id :

Name :

Year of Passing :

Department :

Phone No :

Job Role :

Alumini Create Date :

Overall GPA :

Four Courses that helped you

**use Course CRN only

Subject 1 :

Subject 2 :

Subject 3 :

Subject 4 :

Current Carrer Goal

○ Placed

○ Ongoing Research

○ Higher Education

○ Business / StartUp

Four faculties that helped you

**use faculty Id only

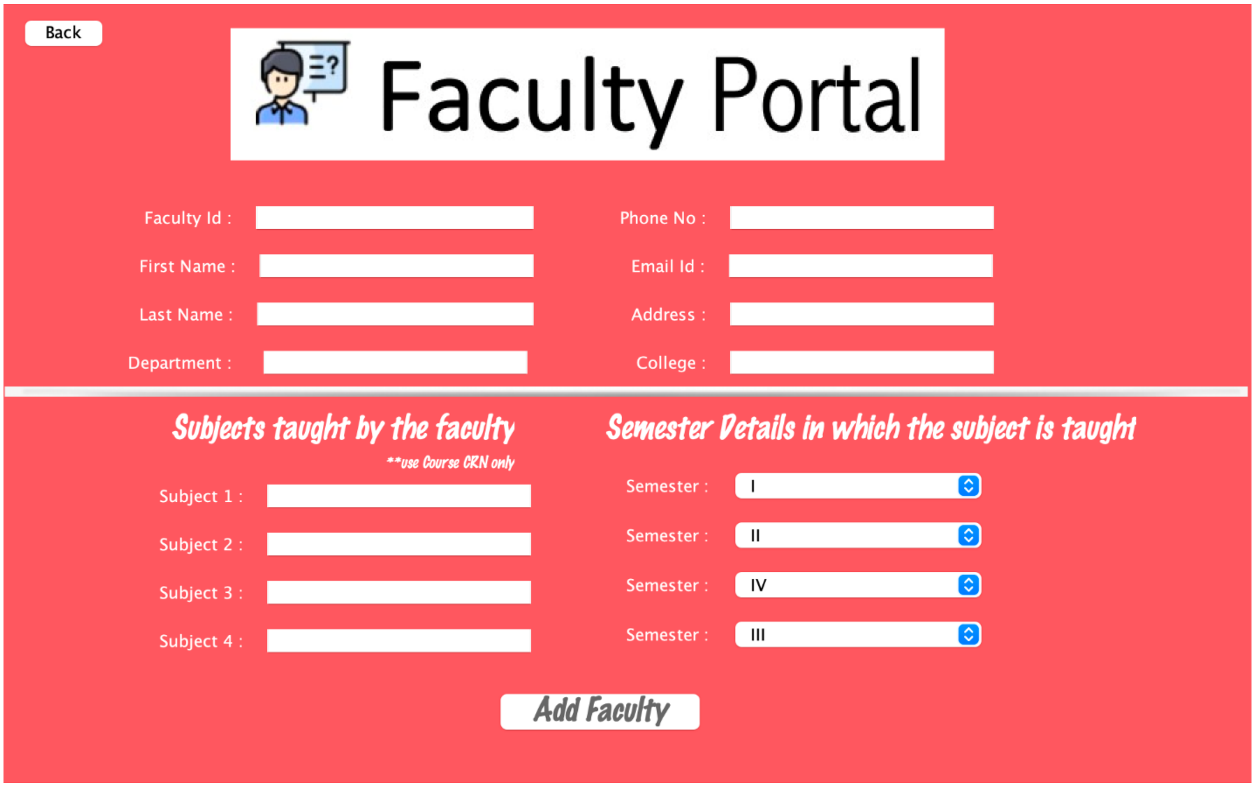
Subject 1 :

Subject 2 :

Subject 3 :

Subject 4 :

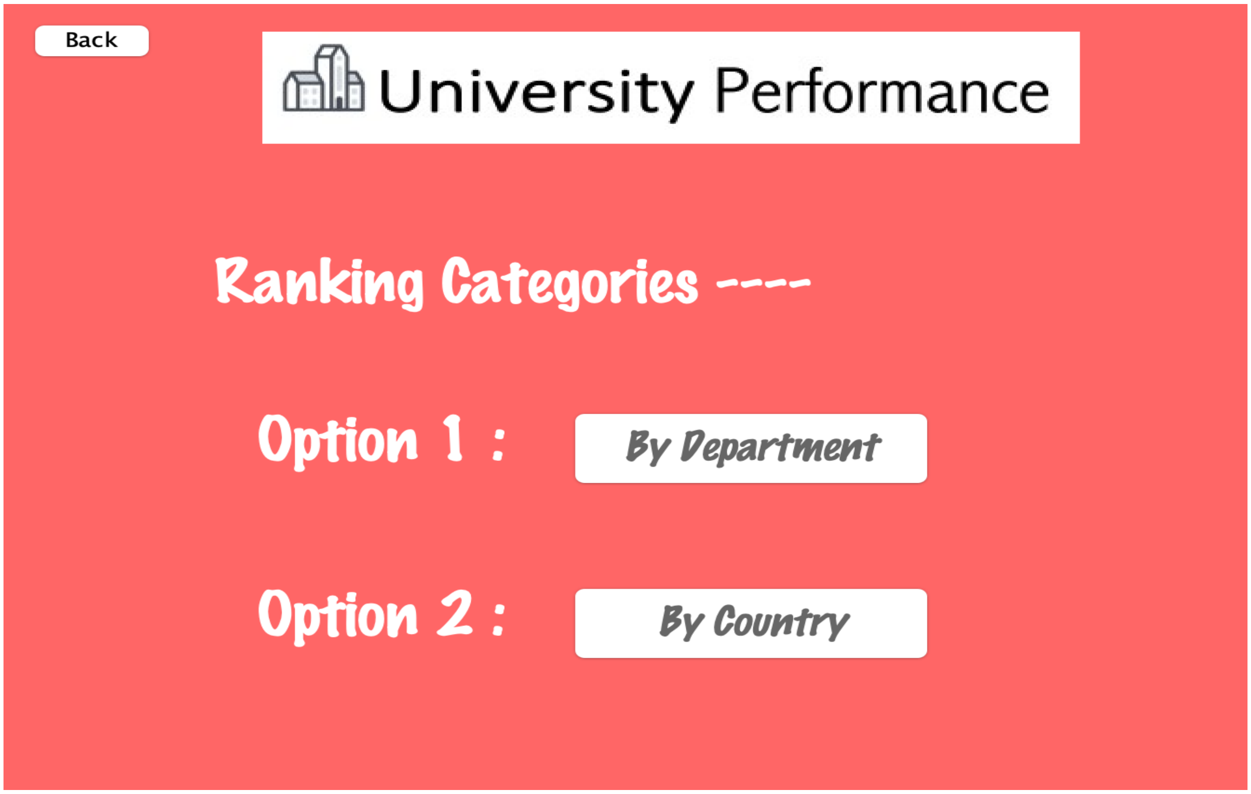
Admin can add/update alumni student details



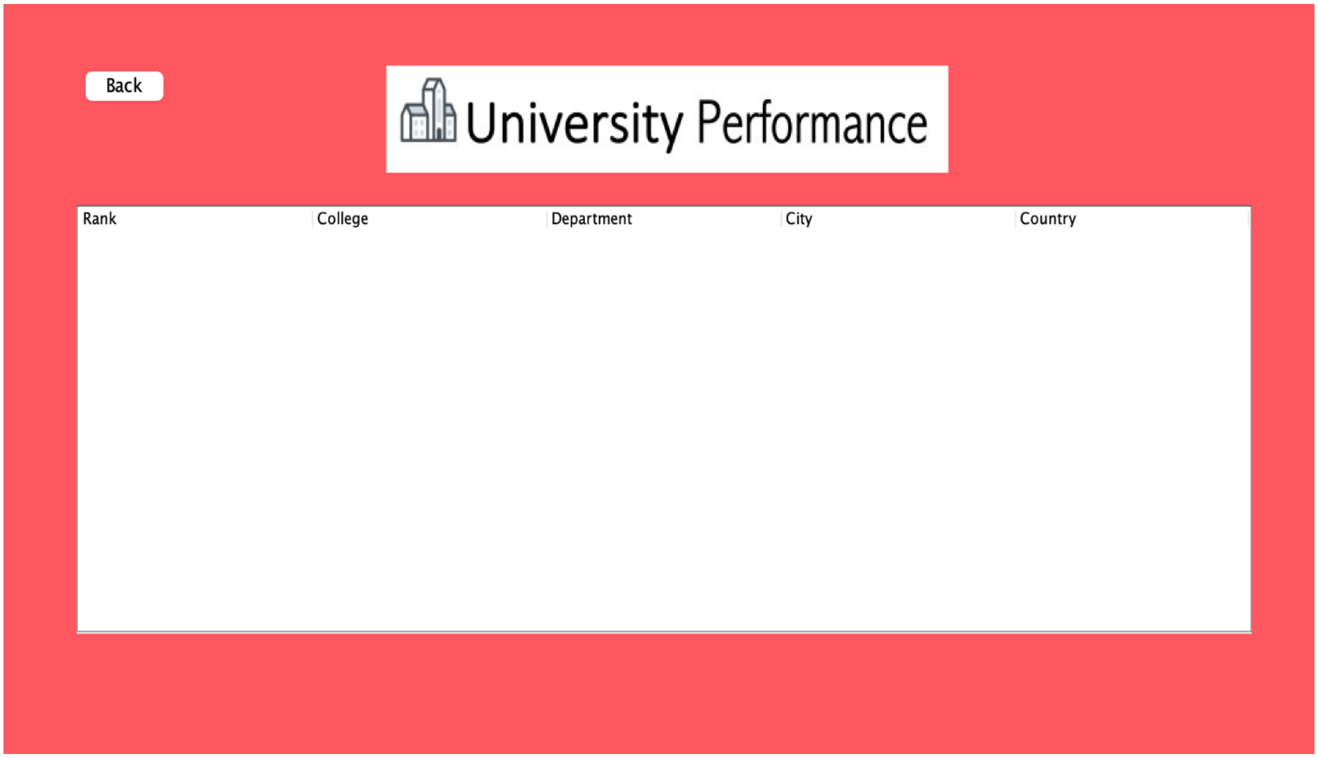
Faculty Id :		Phone No :	
First Name :		Email Id :	
Last Name :		Address :	
Department :		College :	

Subjects taught by the faculty **use Course CRN only		Semester Details in which the subject is taught	
Subject 1 :		Semester :	I
Subject 2 :		Semester :	II
Subject 3 :		Semester :	IV
Subject 4 :		Semester :	III

Admin can add/update faculty details

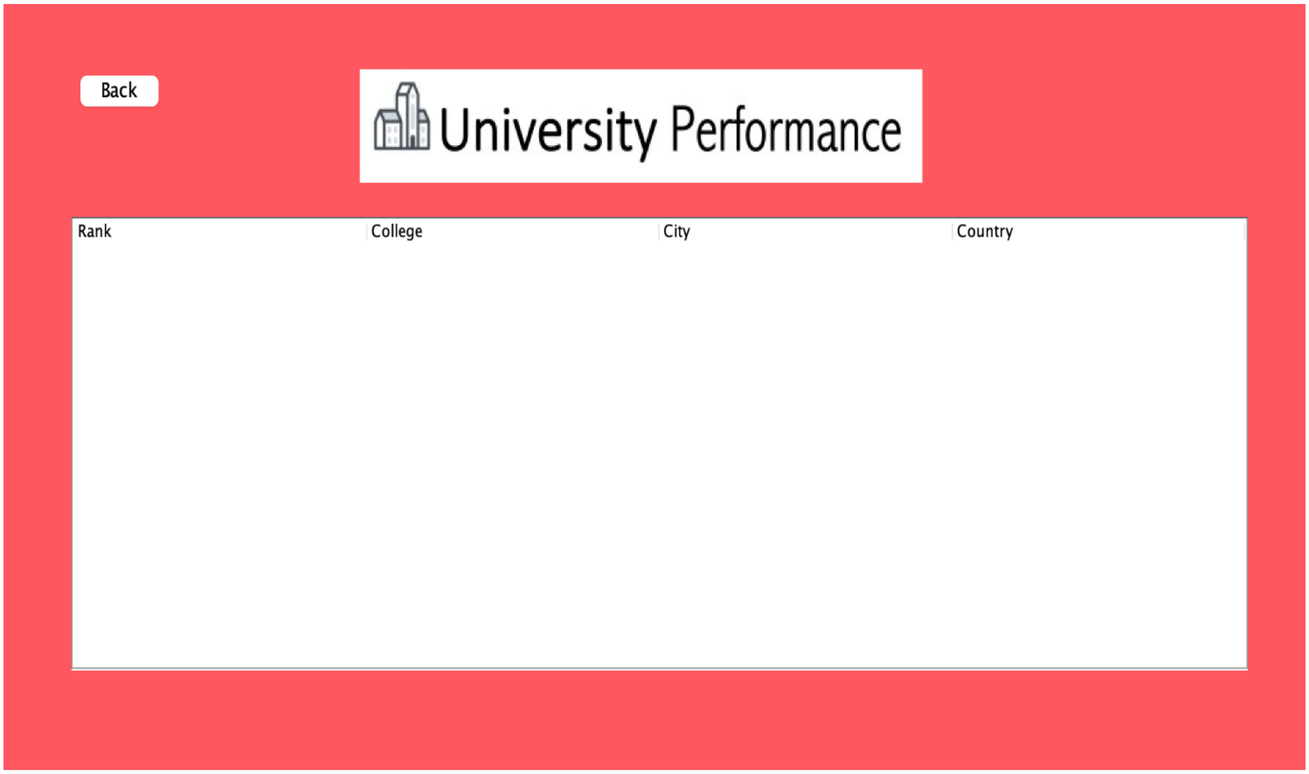


University Performance Screen



Rank	College	Department	City	Country

University Ranking (Department wise)



Rank	College	City	Country

University Ranking (Country wise)

THANK YOU