

**APPLICATION ENGINEERING DEVELOPMENT - INFO 5100**

**DR. PROF. KAL BUGRARA**

**UNIVERSITY RANKING MODEL**

<b>Team Member</b>	<b>NUID</b>
<b>Geethashree Jagadishwara Raju</b>	<b>2198567</b>
<b>Karan Raman Agrawal</b>	<b>1090008</b>
<b>Yash Bhodia</b>	<b>1090359</b>

## **PROBLEM STATEMENT**

The task is to study ways to create a performance measurement solution to enable universities to measure the quality of the education they deliver to their students. The approach will be to look into how an educational system in terms of faculty and courses contribute to the growth of their graduates over a 5-year period. We must figure out ways to track the jobs and promotions graduates get over time and assign rankings accordingly. In addition, track the connection of courses and their relevance to graduates' growth.

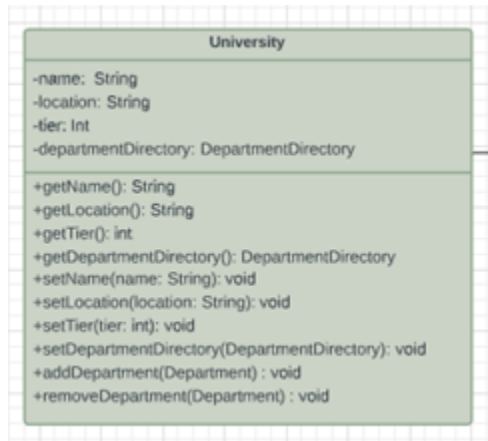
One of the deliverables will be to design a dashboard that enables college and university administrators to compare the performance of their academic units. One additional question is to consider ways to define your own ranking system for students to decide where they want to go for their studies. The current system is biased toward research.

## **JAVA IMPLEMENTATION**

The UI is created using the Java Swing library. The core functionality and the data models are modelled as Java classes, separated into packages as appropriate. The UI code makes use of the functionality provided by the model classes to execute the requests sent by the users. We provide a class "Ranking" which provides various metrics to evaluate a department based on the jobs, relevance of courses to the job, salary, promotions, gpa, etc. By using the metrics provided by the Ranking class, we can obtain powerful insights into how well a department is performing. This can help us identify areas of improvement and take the necessary actions required to improve. We can also rank the departments based on the metrics provided in this class.

## ENTITIES:

### University:



### Attributes:

name: This attribute stores the university name

location: This attribute stores location of the university

tier: This attribute stores the tier of the university

departmentDirectory: This object stores all the values from the department departmentDirectory class

### Methods:

getName(): method to retrieve university name

getLocation(): method to retrieve university location

getTier(): method to retrieve university tier

getDepartmentDirectory(): method to retrieve department details

setName(name: String): method to set university name

setLocation(location: String): method to retrieve university location

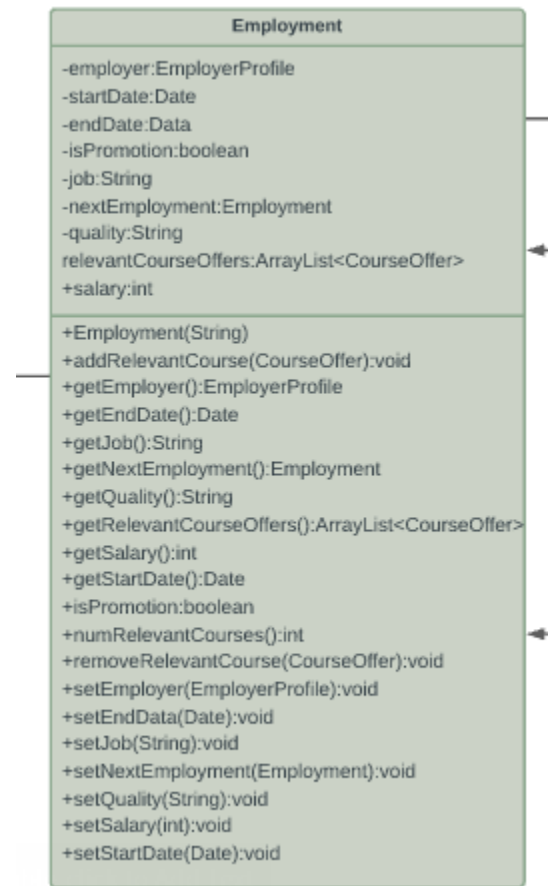
setTier(tier: int): method to retrieve university tier

setDepartmentDirectory(DepartmentDirectory): method to retrieve department details

addDepartment(Department) : void

removeDepartment(Department) : void

## Employment:



## Attributes:

employer: employer details will be added by the object  
startDate: start date of the course  
endDate: End date of the course  
isPromotion: Boolean value of promotion  
job: Job name  
nextEmployment: next employment details of the employer  
salary: Salary earned by the employee  
relevantCourseOffers: Course list employee

## Methods:

getSalary(): method to retrieve salary  
getEmployer(): method to retrieve employer data  
getJob(): method to retrieve job data  
getStartDate(): method to get start date  
getEndDate(): method to get end date  
setSalary(): method to set salary

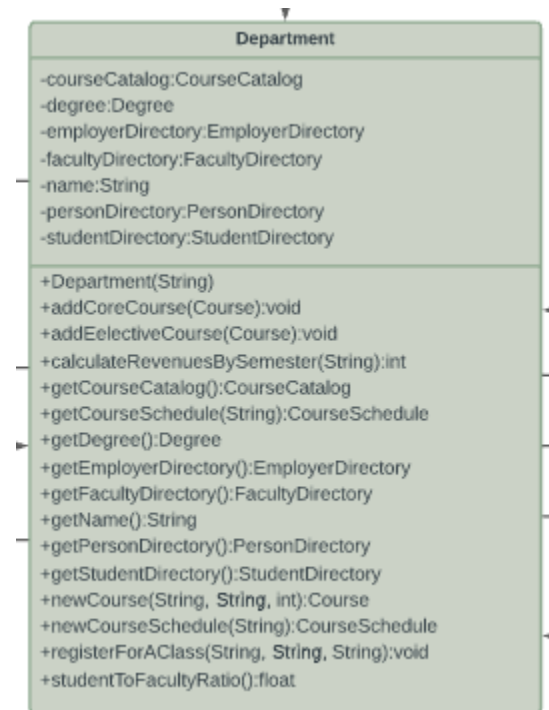
setEmployer(): method to set employer data

setJob(): method to set job data

setStartDate(): method to set start date

setEndDate(): method to set end date

## Department:



## Attributes:

courseCatalog: Object will hold values of the course class.

degree: Object will hold values of the degree class.

employerDirectory: object will hold values of employee class

facultyDirectory: object will hold values of faculty class

name: Department name

personDirectory: object will hold values of the person class

studentDirectory: object will hold values of student class

## Methods:

Department(String): this method for department details

addCoreCourse(Course): method to add core courses

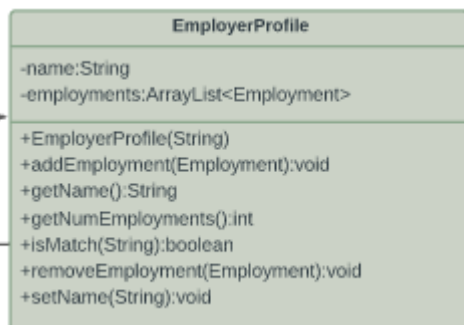
addElectiveCourse(Course): method to add elective courses

calculateRevenuesBySemester(String): method to calculate revenues by semesters

getCourseCatalog(): method to get course catalog

CourseCatalog getCourseSchedule(String): method to get course schedule  
 getDegree(): method to get details about degree  
 getEmployerDirectory(): method to get employer directory  
 getFacultyDirectory(): method to get faculty directory  
 getName(): method to get name  
 getPersonDirectory(): method to get person directory  
 getStudentDirectory(): method to get student directory  
 newCourse(String, String, int): method for new course  
 newCourseSchedule(String): method for new course schedule  
 registerForAClass(String, String, String): method to register for classes  
 studentToFacultyRatio(): method to check student to faculty ratio

### EmployerProfile:



### Attributes:

name: identified for this EmploymentProfile  
 employments:ArrayList<Employment> - list of employments in this EmploymentProfile

### Methods:

EmployerProfile(String) - constructor. Takes the name of the EmployerProfile as the argument and sets the name.  
 addEmployment(Employment):void - adds the Employment object to the "employments" ArrayList.  
 getName():String - Returns the name of the EmployerProfile  
 getNumEmployments():int - returns the number of employments in the "employments" ArrayList  
 isMatch(String):boolean - returns true if the passed in String is equal to the "name" field.  
 removeEmployment(Employment):void - removes the specified Employment object, if present, from the list of employments.  
 setName(String):void - sets the "name" field.

## EmploymentHistory



### Attributes -

`employments: ArrayList<Employment>` - list of Employment objects in the EmploymentHistory

### Methods -

`EmploymentHistory()` - constructor. Initializes the employments ArrayList to an empty ArrayList.

`getEmployments(): ArrayList<Employment>` - returns the list of Employments

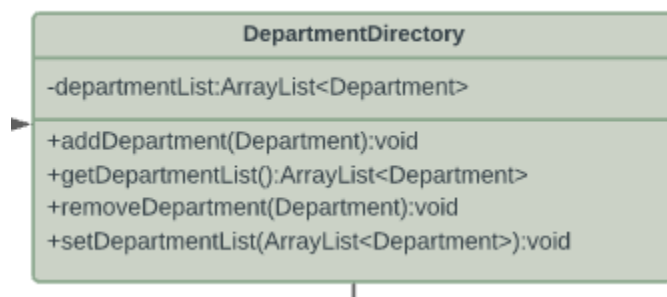
`getNumPromotions(): int` - returns the number of promotions in the list of employments in this EmploymentHistory

`getTotalEmployments(): int` - returns the total number of employments in this list of employments.

`hadJobWithRelevantCourse(): boolean` - returns true if there is at least one job in the list of employments that has relevant courses.

`newEmployment(String): Employment` - creates a new Employment object with the specified string, adds it to the list of employments, and returns the newly created Employment object.

## DepartmentDirectory



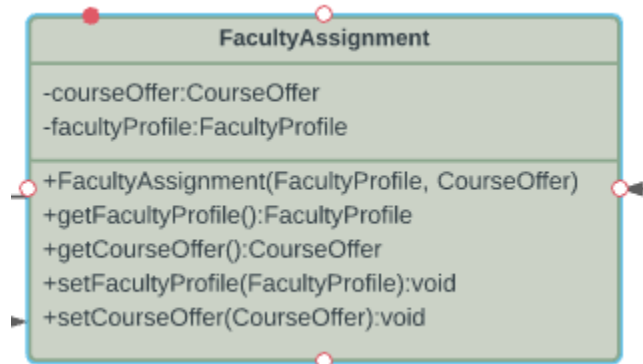
### Attributes -

`departmentList: ArrayList<Department>` - list of Department objects

### Methods -

addDepartment(Department):void - adds a Department to the "departmentList" ArrayList  
 getDepartmentList():ArrayList<Department> - returns the "departmentList" ArrayList  
 removeDepartment(Department):void - removes the first occurrence of the specified Department object, if it exists, in the "departmentList" ArrayList.  
 setDepartmentList(ArrayList<Department>):void - sets the "departmentList" ArrayList.

## FacultyAssignment



### Attributes -

courseOffer:CourseOffer - the course that is offered  
 facultyProfile:FacultyProfile - the faculty offering the course

### Methods -

FacultyAssignment(FacultyProfile, CourseOffer) - the constructor. Sets the passed in courseOffer and facultyProfile as it's fields.

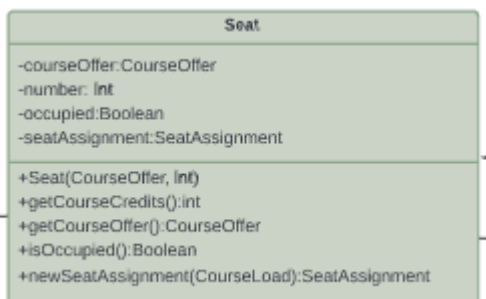
getFacultyProfile():FacultyProfile - returns the FacultyProfile

getCourseOffer():CourseOffer - returns the CourseOffer

setFacultyProfile(FacultyProfile):void - sets the FacultyProfile

setCourseOffer(CourseOffer):void - sets the CourseOffer

## Seat



### Attributes -

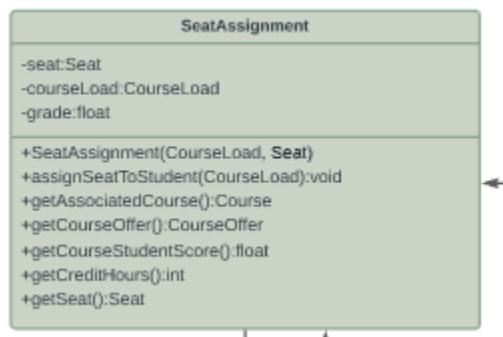


courseOffer: Object stores course offered information  
number: this field stores number of seats  
occupied: this field stores number of seats occupied  
seatAssignment: object stores seat assignment

### Methods -

Seat(CourseOffer, int): this method stores details about seat offered  
getCourseCredits(): this method is to get course credit details  
getCourseOffer(): this method is to get course offer details  
isOccupied(): this method is to check if seat is occupied  
newSeatAssignment(CourseLoad) : this method is for assigning new seat

### Seat Assignment



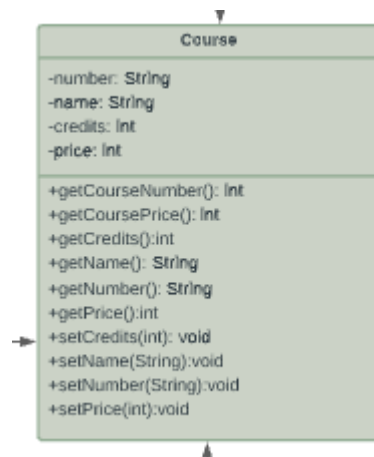
### Attributes:

Seat: this field stores values of seats  
courseLoad: this field stores values of course load  
Grade: this field stores values of grades

### Methods:

SeatAssignment(CourseLoad, Seat): this method is used for seat assignment  
assignSeatToStudent(CourseLoad): this method is used for assigning seat to the student  
getAssociatedCourse(): this method is used to get associated course details  
getCourseOffer(): this method is used to get course offer details  
getCourseStudentScore(): this method is used to get student course score  
getCreditHours(): this method is used to get credit hours of the course  
getSeat(): this method is used to get seat information

## Course



### Attributes:

number: this field stores values of number of courses

name: this field stores values of name of course

credits: this field stores values of credits

price: this field stores values of price of course

### Methods:

`getCourseNumber()`: this method is used to get course number

`getCoursePrice()`: this method is used to get the course price

`getCredits()`: this method is used to get credits

`getName()`: this method is used to get course name

`getNumber()`: this method is used to get course number

`getPrice()`: this method is used to get price

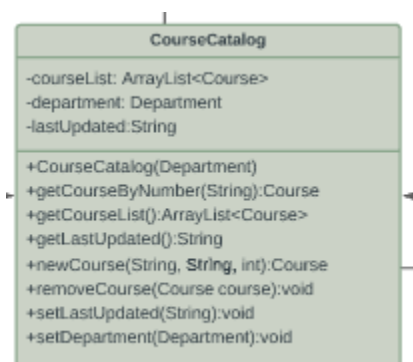
`setCredits(int)`: this method is used to set credits

`setName(String)`: this method is used to set name

`setNumber(String)`: this method is used to set number of course

`setPrice(int)`: this method is used to set price

### Course catalog:



**Attributes:**

courseList: ArrayList<Course>: This stores list of courses

department: this object stores all department details

lastUpdated: this field stores last updated catalog details

**Methods:**

CourseCatalog(Department): this method is used for course catalog

getCourseByNumber(String): this method is used to get number of course

getCourseList(): this method is used to get list of courses

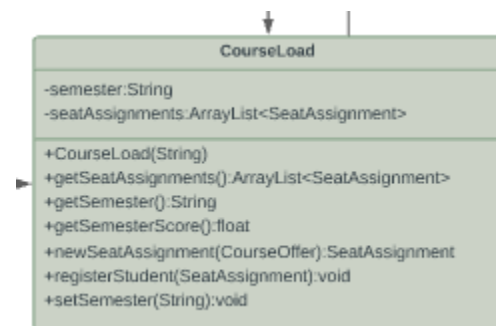
getLastUpdated(): this method is used to get details about last update made

newCourse(String, String, int): this method is used for a new course

removeCourse(Course course): this method is used to remove a course

setLastUpdated(String): this method is used to set details about the last update made

setDepartment(Department): this method is used to set department

**Course Load****Attributes:**

semester: this field is used to store values of semester

seatAssignments: ArrayList<SeatAssignment>: this list is used to store seat assignments

**Methods:**

CourseLoad(String): This method is used for course load

getSeatAssignments(): This method is used to get seat assignment details

getSemester(): This method is used to get semester details

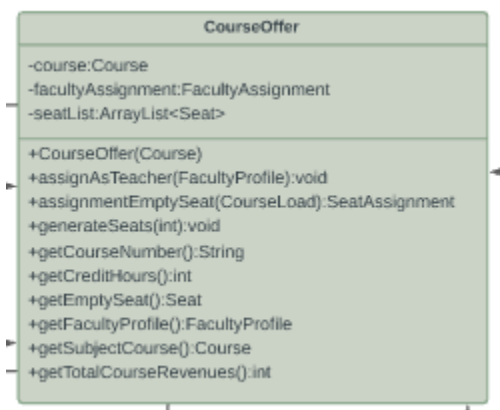
getSemesterScore(): This method is used to get semester score

newSeatAssignment(CourseOffer): This method is used for new seat assignment

registerStudent(SeatAssignment): This method is used for registering a student

setSemester(String): This method is used to set semester

### Course offer



### Attributes:

course: object stores course details

facultyAssignment: object stores faculty assignment details

seatList: ArrayList<Seat>: this list stores list of seat

### Methods:

courseOffer(Course): this method is for course offering to faculty

assignAsTeacher(FacultyProfile): this method is used for assigning a teacher

assignmentEmptySeat(CourseLoad): this method is used for assigning empty seat

generateSeats(int): this method is used for generating seats

getCourseNumber(): this method is used to get course number

getCreditHours(): this method is used to get credit hours

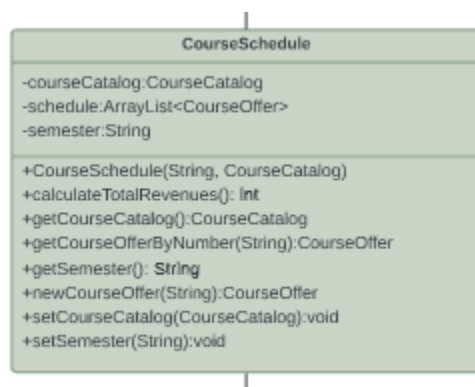
getEmptySeat(): this method is used to get empty seats

getFacultyProfile():this method is used to get faculty profile

getSubjectCourse():this method is used to get subject course

getTotalCourseRevenues(): this method is used to get total course revenue

### Course Schedule:



### Attributes:

courseCatalog:this object is for storing course details

schedule:ArrayList<CourseOffer> : this list is used to store course offer schedule

Semester:this field store values of semester

### Methods:

CourseSchedule(String, CourseCatalog): this method is used for course scheduling

calculateTotalRevenues(): this method is used to calculate total revenues

getCourseCatalog():this method is used to get course catalog

getCourseOfferByNumber(String):this method is used to get course offer by number

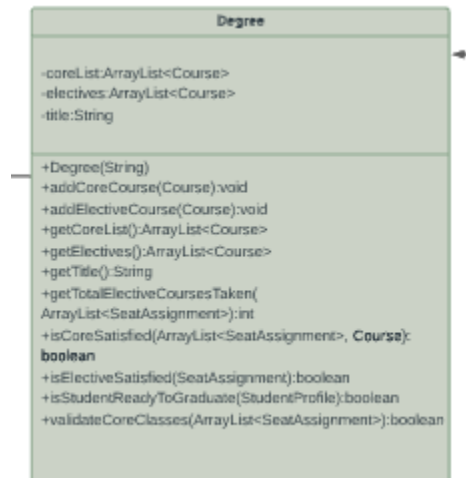
getSemester(): this method is used to get semester

newCourseOffer(String):this method is used for new course offer

setCourseCatalog(CourseCatalog):this method is used to set course catalog

setSemester(String): this method is used to set semester

## Degree



### Attributes:

`coreList: ArrayList<Course>`: this list stores values of list core courses

`electives: ArrayList<Course>`: this list stores values of electives

`Title`: this field stores values of titles for course

### Methods:

`Degree(String)`: this method is used for degree details

`addCoreCourse(Course)`: this method is used to add core courses

`addElectiveCourse(Course)`: this method is used to add elective courses

`getCoreList(): ArrayList<Course>`: this method is used to get core list

`getElectives(): ArrayList<Course>`: this method is used to get electives

`getTitle()`: this method is used to get title

`getTotalElectiveCoursesTaken( ArrayList<SeatAssignment>)`: this method is used to get total elective courses taken

`isCoreSatisfied(ArrayList<SeatAssignment>, Course)`: this method is used to check if core courses satisfy

`isElectiveSatisfied(SeatAssignment)`: this method is used to check if elective courses satisfy

isStudentReadyToGraduate(StudentProfile): this method is used to check if student is ready to graduate

validateCoreClasses(ArrayList<SeatAssignment>):this method is used to validate core classes

## StudentProfile

StudentProfile
-alumni:boolean -currentlyEmployed:boolean -employmentHistory:EmploymentHistory -graduationDate:Date -person:Person -transcript:Transcript
+StudentProfile(Person) +getCourseList():ArrayList<SeatAssignment> +getCourseLoadBySemester(String):CourseLoad +getCurrentCourseLoad():CourseLoad +getEmploymentHistory():EmploymentHistory +getGraduationDate():Date +getTranscript():Transcript +isAlumni():boolean +isCurrentlyEmployed():boolean +isMatch(String):boolean +newCourseLoad(String):CourseLoad +setAlumni(boolean):void +setCurrentlyEmployed(boolean):void +setEmploymentHistory(EmploymentHistory):void +setGraduationDate(Date):void

### Attributes -

alumni:boolean - if true, the student is an alumni. Current student otherwise.

currentlyEmployed:boolean - if true, the student is currently employed. Assumed to be unemployed currently otherwise.

employmentHistory:EmploymentHistory - employment history of the student

graduationDate:Date - if the student has graduated, the graduation date is specified here. null if the student hasn't graduated.

person:Person - the person object for the student

transcript:Transcript - the students transcripts

## Methods -

StudentProfile(Person) - constructor. Sets the person field to the passed in person object.

getCourseList():ArrayList<SeatAssignment> - returns the list of courses taken by the student

getCourseLoadBySemester(String):CourseLoad - returns the CourseLoad for the specified semester parameter

getCurrentCourseLoad():CourseLoad - Returns the CourseLoad for the current semester

getEmploymentHistory():EmploymentHistory - returns the EmploymentHistory of the student

getGraduationDate():Date - returns the "graduationDate" object. This may be null if the student has not graduated yet.

getTranscript():Transcript - returns the Transcript of the student

isAlumni():boolean - returns "alumni" field. Will be true if the student is an alumni. False otherwise.

isCurrentlyEmployed():boolean - Returns "currentlyEmployed" field.

isMatch(String):boolean - returns true if the passed in String is equal to the id on the person object in this StudentProfile class.

newCourseLoad(String):CourseLoad - adds a new course and returns the CourseLoad

setAlumni(boolean):void - sets the passed in value as the value of the "alumni" field.

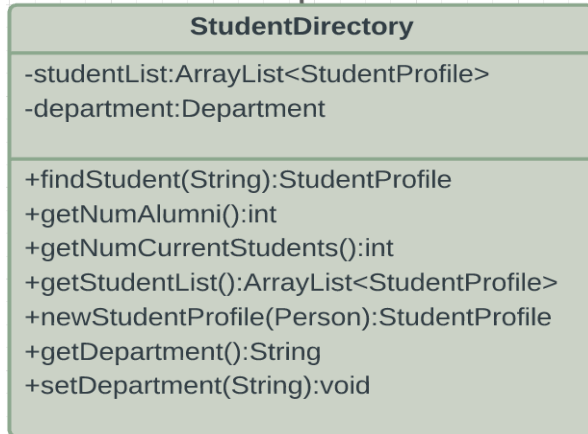
setCurrentlyEmployed(boolean):void - sets the passed in value as the value of the "currentlyEmployed" field.

setEmploymentHistory(EmploymentHistory):void - setter for the employmentHistory field.

setGraduationDate(Date):void - setter for the graduationDate field.



## StudentDirectory



### Attributes -

`studentList: ArrayList<StudentProfile>` - list of all students in this student directory.

`department: Department` - department to which this student directory belongs to.

### Methods -

`findStudent(String): StudentProfile` - finds and returns the `StudentProfile` whose `Person` id matches the passed in string. returns null if no match is found.

`getNumAlumni(): int` - returns the total number of alumni in this `StudentDirectory`.

`getNumCurrentStudents(): int` - returns the total number of current students (not alumni) in this `StudentDirectory`.

`getStudentList(): ArrayList<StudentProfile>` - returns the list of `StudentProfile`

`newStudentProfile(Person): StudentProfile` - creates a new `StudentProfile` with the specified `Person`, adds it to the list of `StudentProfiles` and returns the newly created `StudentProfile` object.

`getDepartment(): Department` - returns the department object

`setDepartment(Department): void` - sets the department object

## FacultyDirectory

FacultyDirectory
-department:Department -teacherList:ArrayList<FacultyProfile>
+FacultyDirectory(Department) +findTeachingFaculty(String):FacultyProfile +getTeacherList():ArrayList<FacultyProfile> +getTopProfessor():FacultyProfile +newFacultyProfile(Person):FacultyProfile +setTeacherList(ArrayList<FacultyProfile>):void

### Attributes -

department:Department - department to which this FacultyDirectory belongs to

teacherList:ArrayList<FacultyProfile> - list of faculty in this directory

### Methods -

FacultyDirectory(Department) - Constructor. Sets the department to the passed in department.

findTeachingFaculty(String):FacultyProfile - returns the FacultyProfile whose Person id matches the passed in String. returns null if there is no match.

getTeacherList():ArrayList<FacultyProfile> - returns a list of all FacultyProfile in this FacultyDirectory.

getTopProfessor():FacultyProfile - returns the FacultyProfile with the highest rating.

newFacultyProfile(Person):FacultyProfile - creates a new FacultyProfile, adds it to the list of FacultyProfile and returns the newly created object.

setTeacherList(ArrayList<FacultyProfile>):void - sets the list of FacultyProfile .

## Transcript

Transcript
<ul style="list-style-type: none"><li>-courseLoadList:HashMap&lt;String, CourseLoad&gt;</li><li>-currentCourseLoad:CourseLoad</li><li>-student:StudentProfile</li></ul>
<ul style="list-style-type: none"><li>+Transcript(StudentProfile)</li><li>+getCourseList():ArrayList&lt;SeatAssignment&gt;</li><li>+GetCourseLoadBySemester(String):CourseLoad</li><li>+getCurrentCourseLoad():CourseLoad</li><li>+getStudentTotalScore():float</li><li>+newCourseLoad(String):CourseLoad</li><li>+getStudentProfile:StudentProfile</li><li>+setStudentProfile(StudentProfile):void</li></ul>

### Attributes -

courseLoadList:HashMap<String, CourseLoad> - mapping between the semester (a String) and the CourseLoad for that semester.

currentCourseLoad:CourseLoad - CourseLoad for the current semester.

student:StudentProfile - the student to whom this transcript belongs to

### Methods -

Transcript(StudentProfile) - constructor. Creates a new Transcript and sets the passed in StudentProfile as the StudentProfile for the new object.

getCourseList():ArrayList<SeatAssignment> - returns a list of SeatAssignments specifying all the courses

GetCourseLoadBySemester(String):CourseLoad - returns the course load for the passed in semester string. Returns null if there is no match.

getCurrentCourseLoad():CourseLoad - returns the CourseLoad for the current semester.

getStudentTotalScore():float - returns the sum of scores across all semesters for this student.

newCourseLoad(String):CourseLoad - creates a new CourseLoad and adds it to the specified semester

getStudentProfile:StudentProfile - returns the StudentProfile

setStudentProfile(StudentProfile):void - sets the StudentProfile

## Person

Person
-id:String -name:String
+getPersonId():String +getName():String +setId(String):void +setName(String):void

### Attributes -

id:String - the id for this person

name:String - the name of the person

### Methods -

getPersonId():String - returns the id of this person

getName():String - returns the name

setId(String):void - sets the id as the passed in string

setName(String):void - sets the name as the passed in string

## Ranking

Ranking
+alumniEmploymentPercentage(Department):float +averageJobSalaryTier(Department):int +averagePromotions(Department):float +averageStudentJobSalary(Department):float +employedAlumInRelevantField(Department):float +getDepartmentRank(Department):float +getSalaryTier(float):int +getAverageGPA(Department):float +percentageAlumbWithRelevantCourses(Department):float +studentToFacultyRatio(Department):float

## Methods -

`alumniEmploymentPercentage(Department):float` - returns the percentage of alumni that are currently employed in the passed in Department object.

`averagePromotions(Department):float` - finds the number of promotions for each student in the department and returns the average number of promotions for a student in the passed in department.

`averageStudentJobSalary(Department):float` - finds the salary for each student in the department and returns the average salary for a student in the passed in department.

`getSalaryTier(float):int` - returns a "tier" for the passed in Salary. In our example, tier 1 = < \$30,000. tier 2 = \$30,001 - \$60,000. Tier 3 = \$60,001 - \$100,000. Tier 4 = \$100,000-\$250,000, Tier 5 = \$250,000-\$500,000. Tier 6 = \$500,000-\$1,000,000. Tier 7 (max tier) = anything greater than \$1,000,000

`averageJobSalaryTier(Department):int` - finds the average salary for a student in the passed in department and returns the tier for that salary.

`percentageAlumnWithRelevantCourses(Department):float` - returns the percentage of alumni that have at least one employment that has a course relevant to the employment in the passed in department.

`employedAlumInRelevantField(Department):float` - returns the percentage of alumni that are currently employed who are also employed in a field for which there is a relevant course.

`getDepartmentRank(Department):float` - assigns a rank to the passed in department. The rank is calculated as  $\text{percentageAlumnWithRelevantCourses(Department)} * \text{averageJobSalaryTier(Department)} * \text{averagePromotions(Department)}$ . Thus, departments with higher ranking have higher average promotions, higher average salary, higher percentage of employment, and higher percentage of students employed in a field related to the courses they took at the University.

`getAverageGPA(Department):float` - returns the average GPA for students in a department

`studentToFacultyRatio(Department):float` - returns the ratio of student to faculty in the passed in department.

## **UNIVERSITY RANKING SOLUTION**

### **Average GPA:**

The average GPA is the major factor that helps in ranking the universities. This factor helps in determining how students are performing in their respective departments and is the major factor for scholarships.

### **Student to faculty ratio:**

This attribute helps to rank the university based on the number of students which directly to the number of teachers.

### **Average promotions:**

This attribute lists out the number of promoted graduates over a 5 year time within different companies which help to determine how good the students are which directly related to university's ranking.

### **Average StudentJobSalary :**

This attribute lists out the Salary ranges of the students. Helps to determine how students are doing in their professional career.

### **Number of Placement(Employment percentage)**

This attribute will list out the number of students who have been placed and determine the rating of the college based on that.

### **Number of employedAluminiaRevelantField:**

This attribute determines how many employees have been working in the relevant field post completion of their graduation.

## Dashboard

Create Student

Update Student

Create Faculty

Update Faculty

Create Department

Update Department

Register Alumni

Update Alumni

View Student

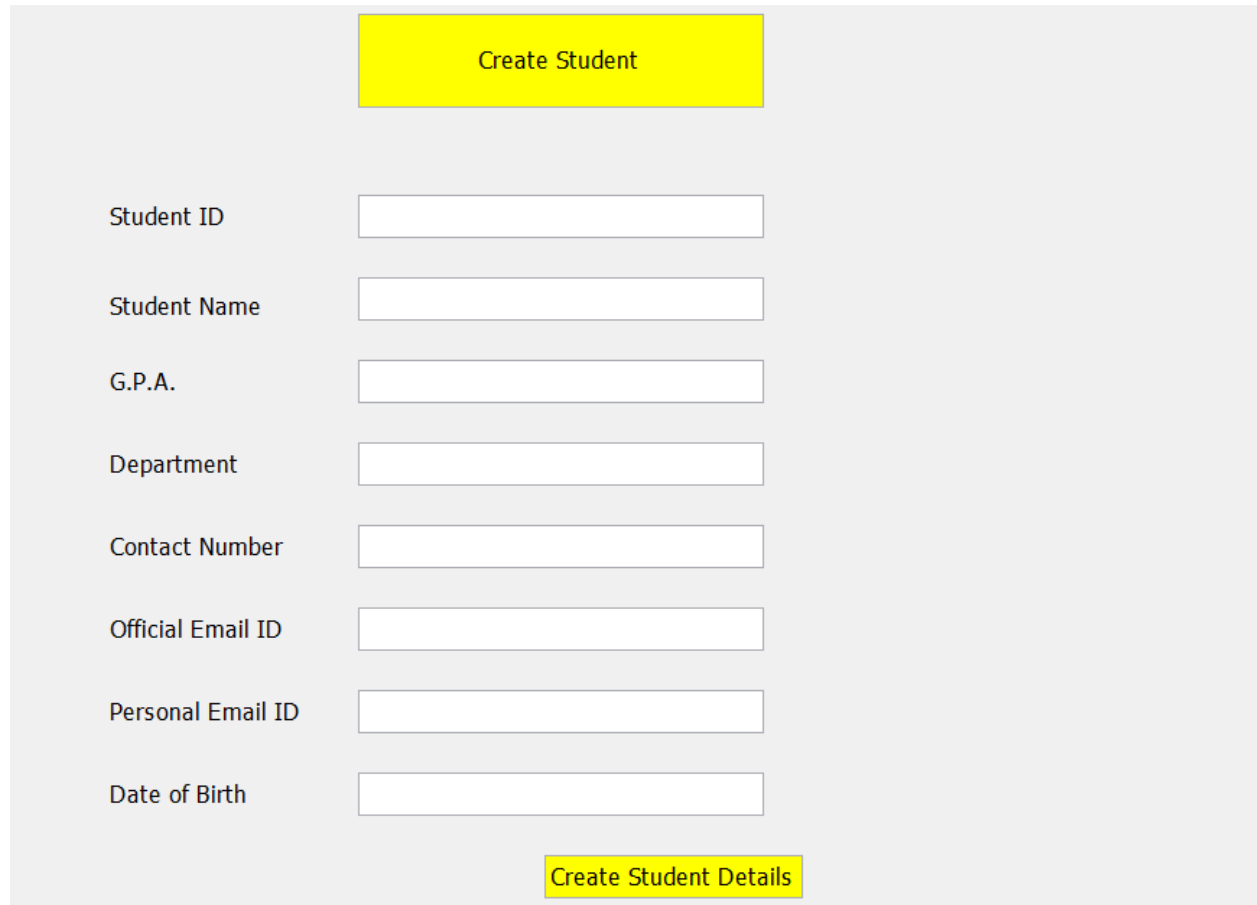
View Alumni

View Faculty

View Department

## Main Screen

This screen will help the user to navigate to different modules of the application.



The image shows a web application interface for creating a new student. It features a light gray background. At the top center, there is a yellow rectangular button with the text "Create Student". Below this button, on the left side, are seven labels: "Student ID", "Student Name", "G.P.A.", "Department", "Contact Number", "Official Email ID", and "Date of Birth". To the right of each label is a white rectangular input field. At the bottom center of the form area, there is another yellow rectangular button with the text "Create Student Details".

	Create Student	
Student ID	<input type="text"/>	
Student Name	<input type="text"/>	
G.P.A.	<input type="text"/>	
Department	<input type="text"/>	
Contact Number	<input type="text"/>	
Official Email ID	<input type="text"/>	
Personal Email ID	<input type="text"/>	
Date of Birth	<input type="text"/>	
	Create Student Details	

This module is used to add new students to the university, mainly during fresh intakes.



	Update Student	
Student ID	<input type="text"/>	
Student Name	<input type="text"/>	
G.P.A.	<input type="text"/>	
Department	<input type="text"/>	
Contact Number	<input type="text"/>	
Official Email ID	<input type="text"/>	
Personal Email ID	<input type="text"/>	
Date of Birth	<input type="text"/>	
	Update Student Details	

This interface will be used to update the information of students.

## View Student Information

Student ID

Student Name

G.P.A.

Department

Contact Number

Official Email ID

Personal Email ID

Date Of Birth

This Interface will be used to view the latest information of the student.

	Create Faculty	
Faculty ID	<input type="text"/>	
Faculty Name	<input type="text"/>	
Designation	<input type="text"/>	
Department	<input type="text"/>	
Contact Number	<input type="text"/>	
Official Email ID	<input type="text"/>	
Personal Email ID	<input type="text"/>	
Qualification	<input type="text"/>	
	Create Faculty Details	

This interface is used to create a faculty.

	Update Faculty	
Faculty ID	<input type="text"/>	
Faculty Name	<input type="text"/>	
Designation	<input type="text"/>	
Department	<input type="text"/>	
Contact Number	<input type="text"/>	
Official Email ID	<input type="text"/>	
Personal Email ID	<input type="text"/>	
Qualification	<input type="text"/>	
	Update Faculty Details	

This interface is used to update the information of the faculty

## View Faculty Information

Faculty ID

Faculty Name

Designation

Department

Contact Number

Official Email ID

Personal Email ID

Qualification

This Interface is used to view the information of the faculty

## Faculty Directory

Faculty ID	Faculty Name	Subject Taught	Ranking

Search by Faculty ID

Search

This interface is used to view the information about faculty, like subjects taught, ranking.

Alumni Registration	
Student ID	<input type="text"/>
Student Name	<input type="text"/>
Department	<input type="text"/>
Employment Status	<input type="text"/>
Employer	<input type="text"/>
Designation	<input type="text"/>
Personal Email ID	<input type="text"/>
Year Of Graduation	<input type="text"/>
<input type="button" value="Register Alumni"/>	

This interface is used to register a current student as an Alumni.

Alumni Update	
Student ID	<input type="text"/>
Student Name	<input type="text"/>
Department	<input type="text"/>
Employment Status	<input type="text"/>
Employer	<input type="text"/>
Designation	<input type="text"/>
Personal Email ID	<input type="text"/>
Year Of Graduation	<input type="text"/>
<input type="button" value="Update Alumni Details"/>	

This interface is used to update the information of Alumni, such as change in employer, salary, etc.



## View Alumni Information

Student ID

Student Name

Department

Employment Status

Employer

Designation

Personal Email ID

Year Of Graduation

This interface is used to view the latest information of Alumni

## Course Registration

Course Faculty

Course Name

CRN Number

Remaining Seats

Student Name

Student ID

Department

Register for the Course

This interface will be used by students to register for courses.

## Create Department

Department Name

Department ID

Create Department

This interface is used to create a new department within a university.

## Update Department

Department Name

Department ID

Update Department

This interface is used to update the current department within the university.

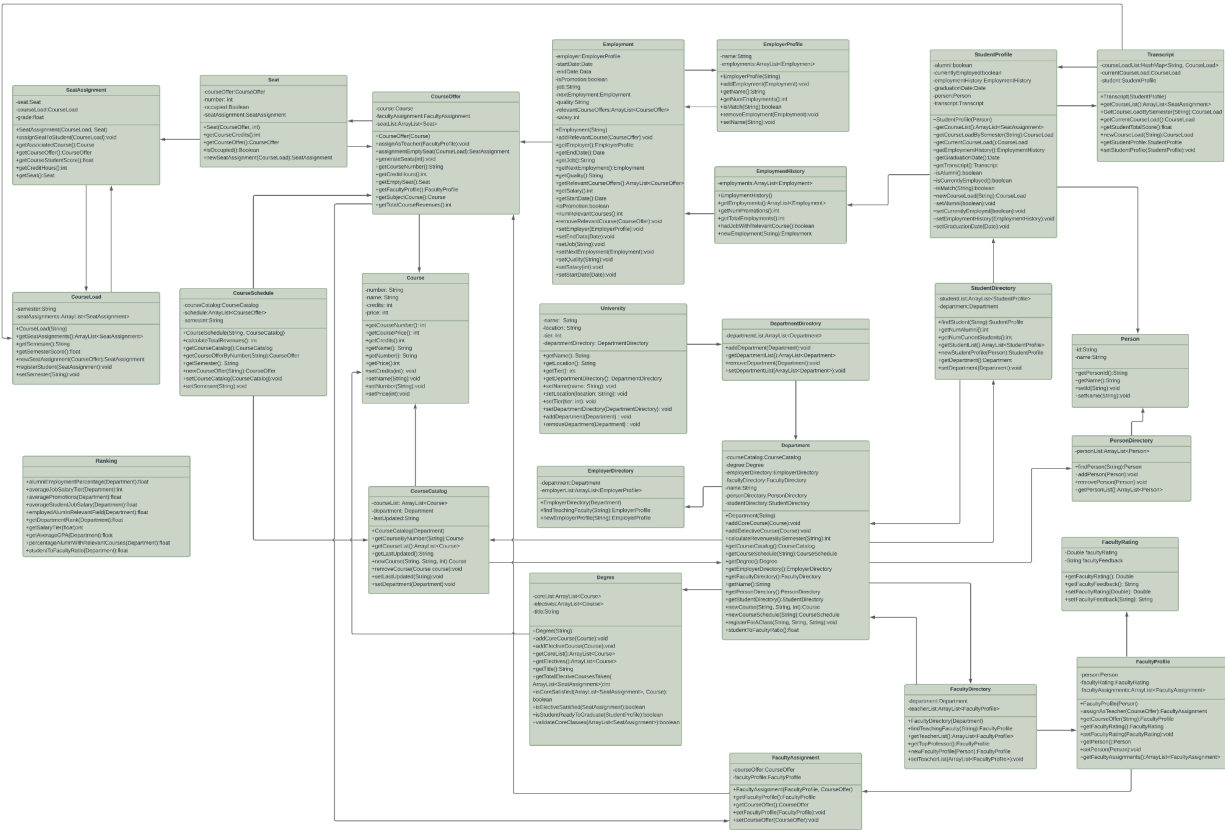
## Department Directory

Department ID	Department Name	Ranking

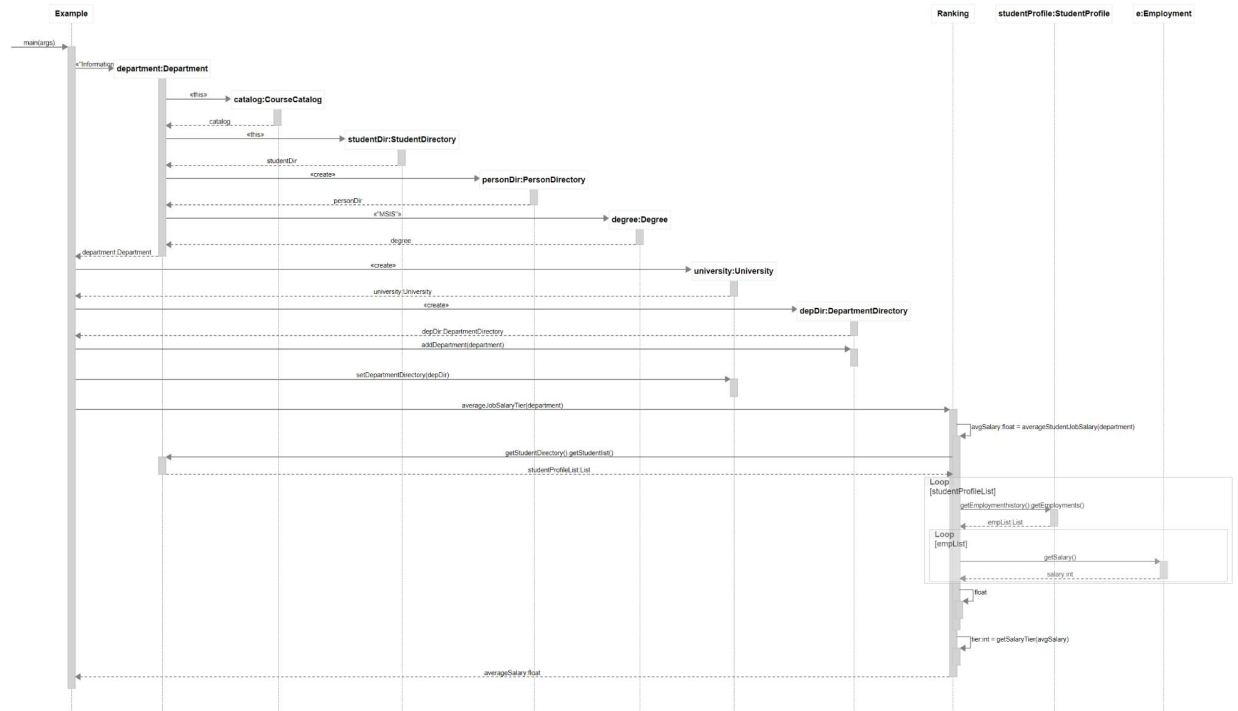
Search by department ID

Search

This interface is used to determine the best department within the university.



UML Diagram of the University Model



Sequence Diagram of the University Model