PROPOSED SOLUTION

For University Model

Group: Bazinga

OUR PROPOSAL

The proposed solution is a university course and faculty ranking system for which data will be fed by the current students and alumni and monitored by the admin.The University model will comprise of the Admin, Current Student and Alumni of the University.

- The admin will have privileges to view the ranking of the faculty and courses as well as manage the course catalog.
- The current student will have the option to add/drop a course from the Course Catalogue, which will include the different courses along with their faculty as well as job opportunities which will provide a clear picture to the student.
- The alumni's role in this model includes updating the portal with their job details including the courses and faculty that played key role in getting the job and also will have privileges to view their job history.

The university can monitor the performance with the help of the performance monitor screens which will fetch the course, faulty and the cumulated average of the ratings provided by the alumni for those courses. The ratings are for a given job role and promotion; thus, the university can track how the courses are ranked across multiple positions over a few years.

The performance monitor fetches data by course ranking, faculty ranking and also provides a custom filter to view the ranking trends over a period of years. The performance monitor also fetches the top 10 most well-rated courses and the top 10 most well-rated faulty as well. To arrive at a rating, we propose to add up all the rankings per the table below and take the cumulated average for the given number of students. This system ensures that a ranking is well-weighted by a student who has maintained an overall good CGPA and is having a senior job position.

Student CGPA range/ Current Position	1-2	2-3	3+
Entry-level	0.25*x	0.5*x	0.75*x
level 2	0.45*x	0.75*x	0.9*x
Mid Sr Level	0.5*x	0.9*x	x

x = ranking input obtained from an alumnus

DESIGN SPECIFICATION

The design model will comprise 3 actors: ADMIN, STUDENT and ALUMNI and the interaction of all the 3 actors with the University.

ALUMNI

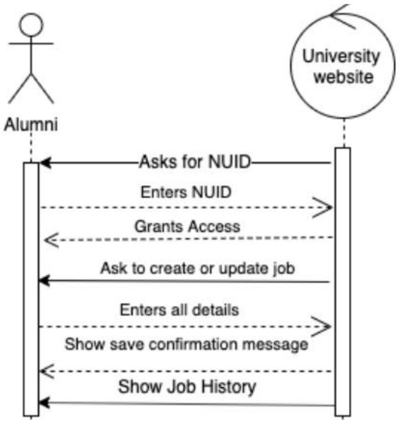

Figure 1: Sequence for alumni

- The alumni logs into the university portal and is asked to enter their NUID.
- On entering NUID, he will be granted access to the portal and will have access to either create a new job entry or update the job entry.
- Create or update job entry will ask for details like company name, new salary, designation and courses which helped for acquiring the job. On click of save, job history of that student is shown in a table.

.

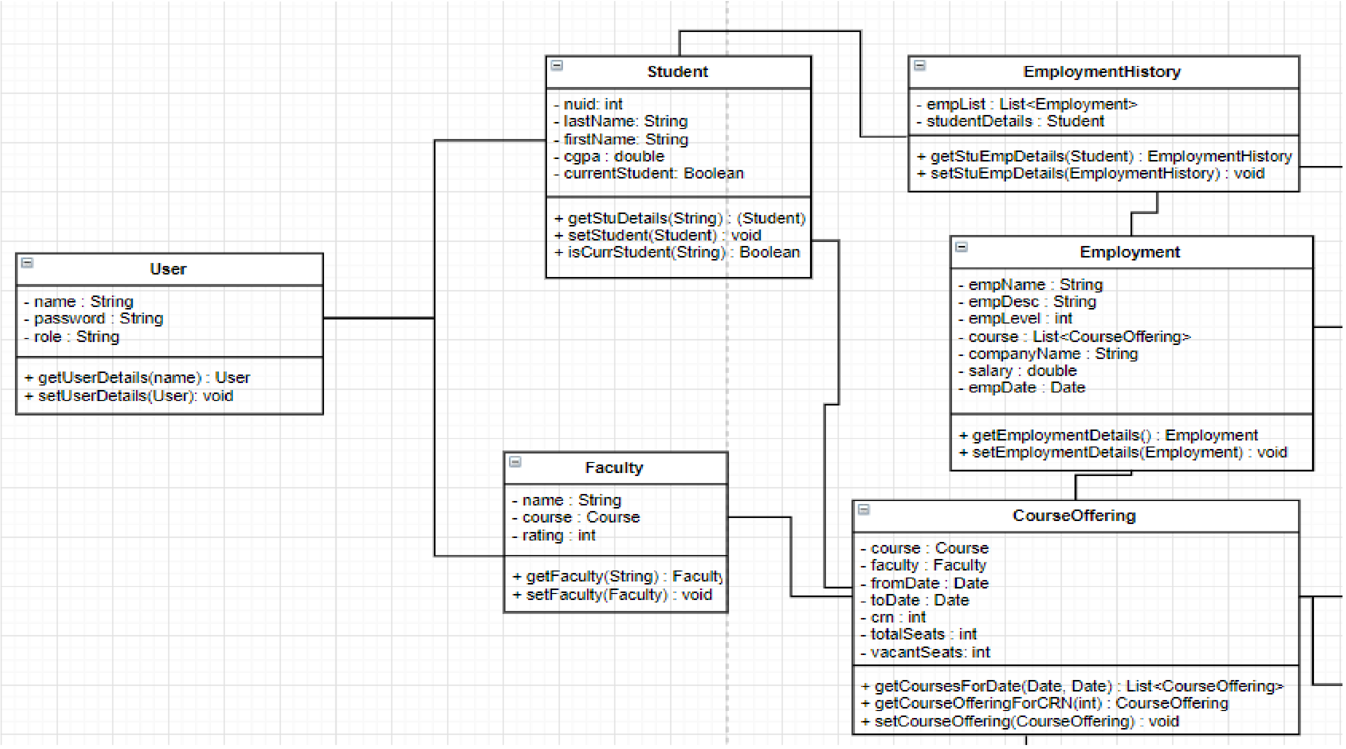
Student
- nuid: int
- lastName: String
- firstName: String
- cgpa : double
- currentStudent: Boolean
+ getStuDetails(String) : (Student)
+ setStudent(Student) : void
+ isCurrStudent(String) : Boolean
EmploymentHistory
- empList : List<Employment>
- studentDetails : Student
+ getStuEmpDetails(Student) : EmploymentHistory
+ setStuEmpDetails(EmploymentHistory) : void
User
- name : String
- password : String
- role : String
+ getUserDetails(name) : User
+ setUserDetails(User): void
Employment
- empName : String
- empDesc : String
- empLevel : int
- course : List<CourseOffering>
- companyName : String
- salary : double
- empDate : Date
+ getEmploymentDetails() : Employment
+ setEmploymentDetails(Employment) : void
Faculty
- name : String
- course : Course
- rating : int
+ getFaculty(String) : Faculty
+ setFaculty(Faculty) : void
CourseOffering
- course : Course
- faculty : Faculty
- fromDate : Date
- toDate : Date
- crn : int
- totalSeats : int
- vacantSeats: int
+ getCoursesForDate(Date, Date) : List<CourseOffering>
+ getCourseOfferingForCRN(int) : CourseOffering
+ setCourseOffering(CourseOffering) : void

Figure 2: Object model for student (both current and alumni)

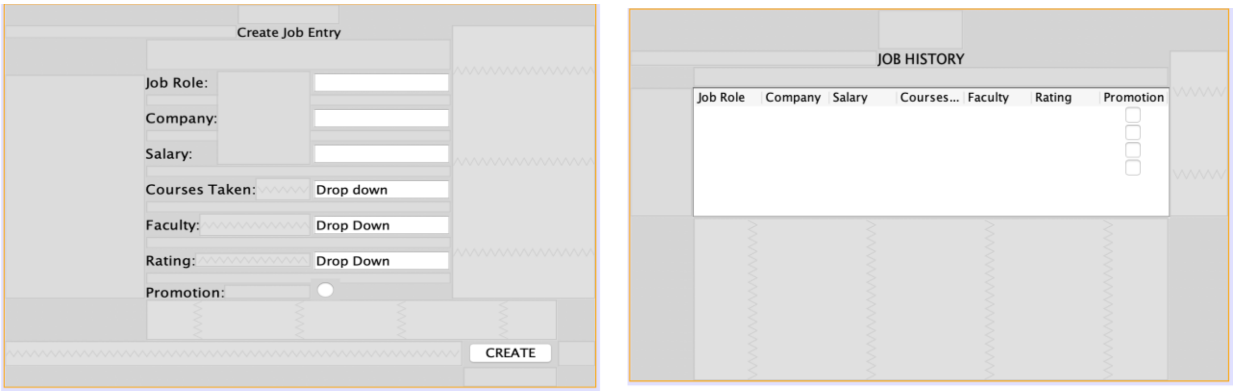

Figure 3: UI templates for Alumni

STUDENT

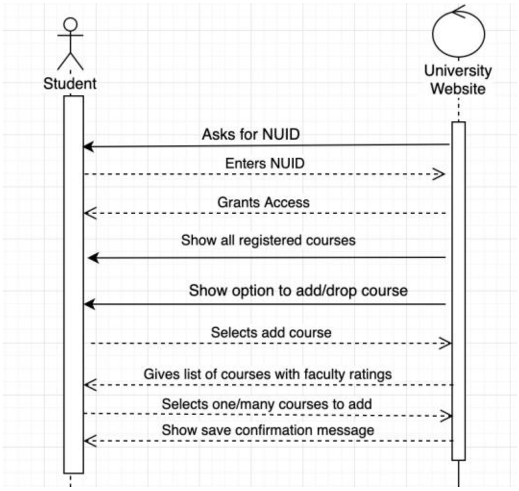
Student
University Website
Asks for NUID
Enters NUID
Grants Access
Show all registered courses
Show option to add/drop course
Selects add course
Gives list of courses with faculty ratings
Selects one/many courses to add
Show save confirmation message

Figure 4: Sequence diagram for current student

- The student logs into the university portal and is asked to enter their NUID. A student is determined to be current student using the Boolean variable of the student object. The object model for the student is the same as the alumni.
- Upon entering NUID, he will be granted access to the portal which shows him all his registered courses.
- He will also have privileges to add/drop a course.
- The portal provides information which includes all the available courses along with the faculty ratings and also possible job opportunities.

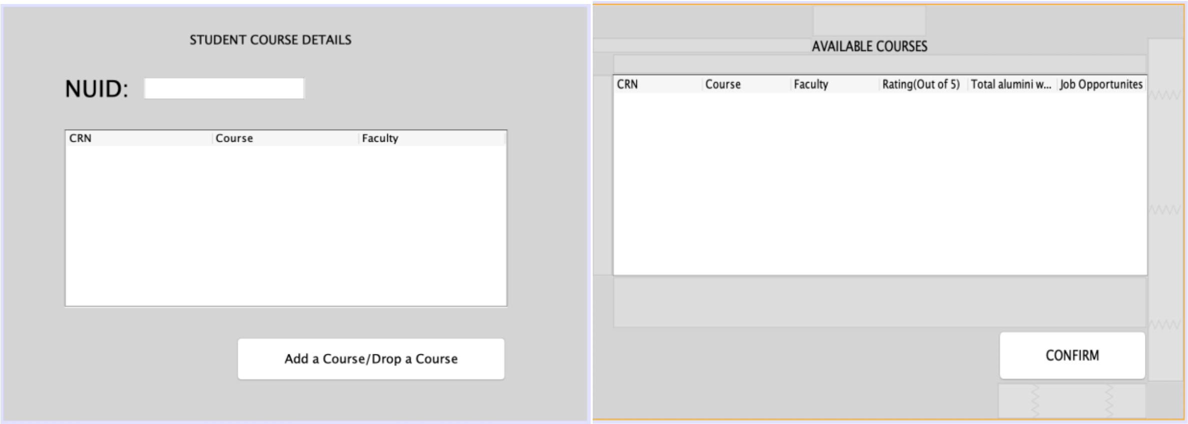

Figure 5: UI templates for current student

ADMIN

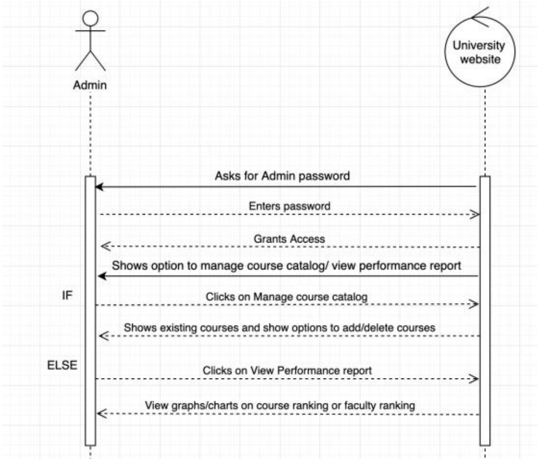
Admin
University website
Asks for Admin password
Enters password
Grants Access
Shows option to manage course catalog/ view performance report
IF
Clicks on Manage course catalog
Shows existing courses and show options to add/delete courses
ELSE
Clicks on View Performance report
View graphs/charts on course ranking or faculty ranking

Figure 6: Sequence Diagram for Admin/Faculty

- The admin logs into the university portal and is asked to enter their NUID.
- Upon entering NUID, he will be granted access to the portal.
- He will also have privileges to add/drop a course.
- The admin will also have access to view the rankings of the students over a period of 5 years and a report which includes graphs/charts indicating the most popular courses along with the faculty list.

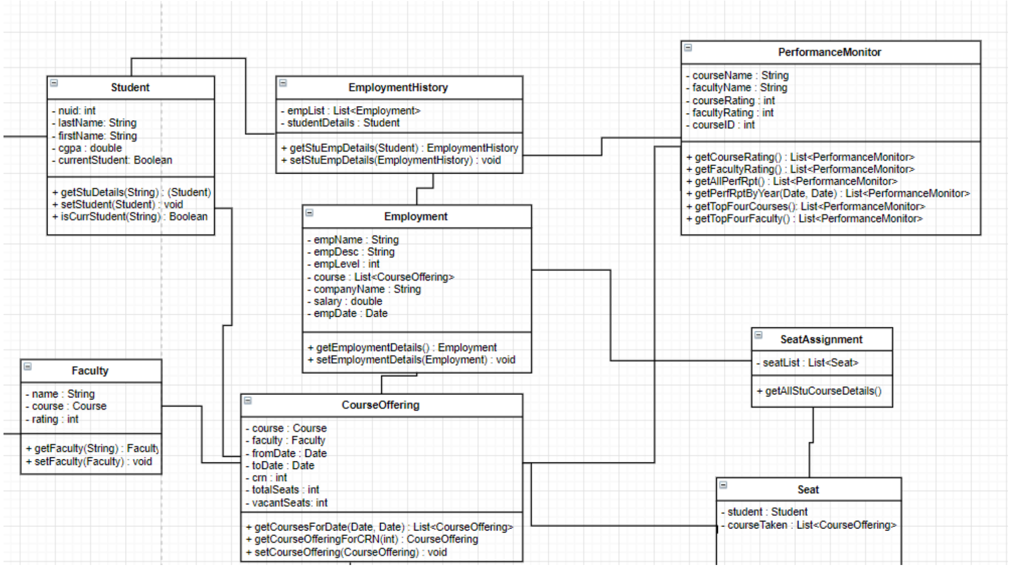
Student
- nuid: int
- lastName: String
- firstName: String
- cgpa : double
- currentStudent: Boolean
+ getStuDetails(String) : (Student)
+ setStudent(Student) : void
+ isCurrStudent(String) : Boolean
EmploymentHistory
- empList : List<Employment>
- studentDetails : Student
+ getStuEmpDetails(Student) : EmploymentHistory
+ setStuEmpDetails(EmploymentHistory) : void
PerformanceMonitor
- courseName : String
- facultyName : String
- courseRating : int
- facultyRating : int
- courseID : int
+ getCourseRating() : List<PerformanceMonitor>
+ getFacultyRating() : List<PerformanceMonitor>
+ getAllPerfRpt() : List<PerformanceMonitor>
+ getPerfRptByYear(Date, Date) : List<PerformanceMonitor>
+ getTopFourCourses(): List<PerformanceMonitor>
+ getTopFourFaculty() : List<PerformanceMonitor>
Employment
- empName : String
- empDesc : String
- empLevel : int
- course : List<CourseOffering>
- companyName : String
- salary : double
- empDate : Date
+ getEmploymentDetails() : Employment
+ setEmploymentDetails(Employment) : void
SeatAssignment
- seatList : List<Seat>
+ getAllStuCourseDetails()
Faculty
- name : String
- course : Course
- rating : int
+ getFaculty(String) : Faculty
+ setFaculty(Faculty) : void
CourseOffering
- course : Course
- faculty : Faculty
- fromDate : Date
- toDate : Date
- crn : int
- totalSeats : int
- vacantSeats: int
+ getCoursesForDate(Date, Date) : List<CourseOffering>
+ getCourseOfferingForCRN(int) : CourseOffering
+ setCourseOffering(CourseOffering) : void
Seat
- student : Student
- courseTaken : List<CourseOffering>

Figure 7: Object Model for admin/faculty

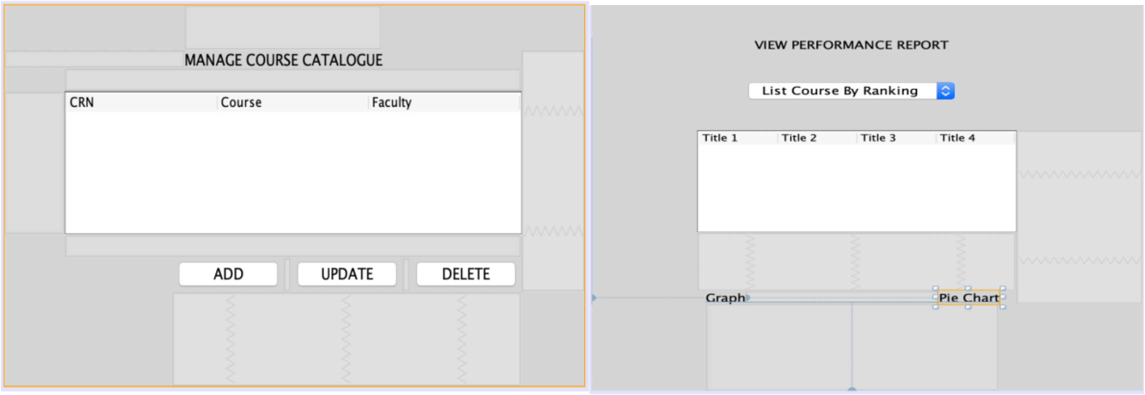

Figure 8: UI Template for Admin

SEQUENCE DIAGRAM

The below sequence diagram helps us understand the interactions between the actors in this model i.e. Admin, Student and Alumni; and the University in a sequent

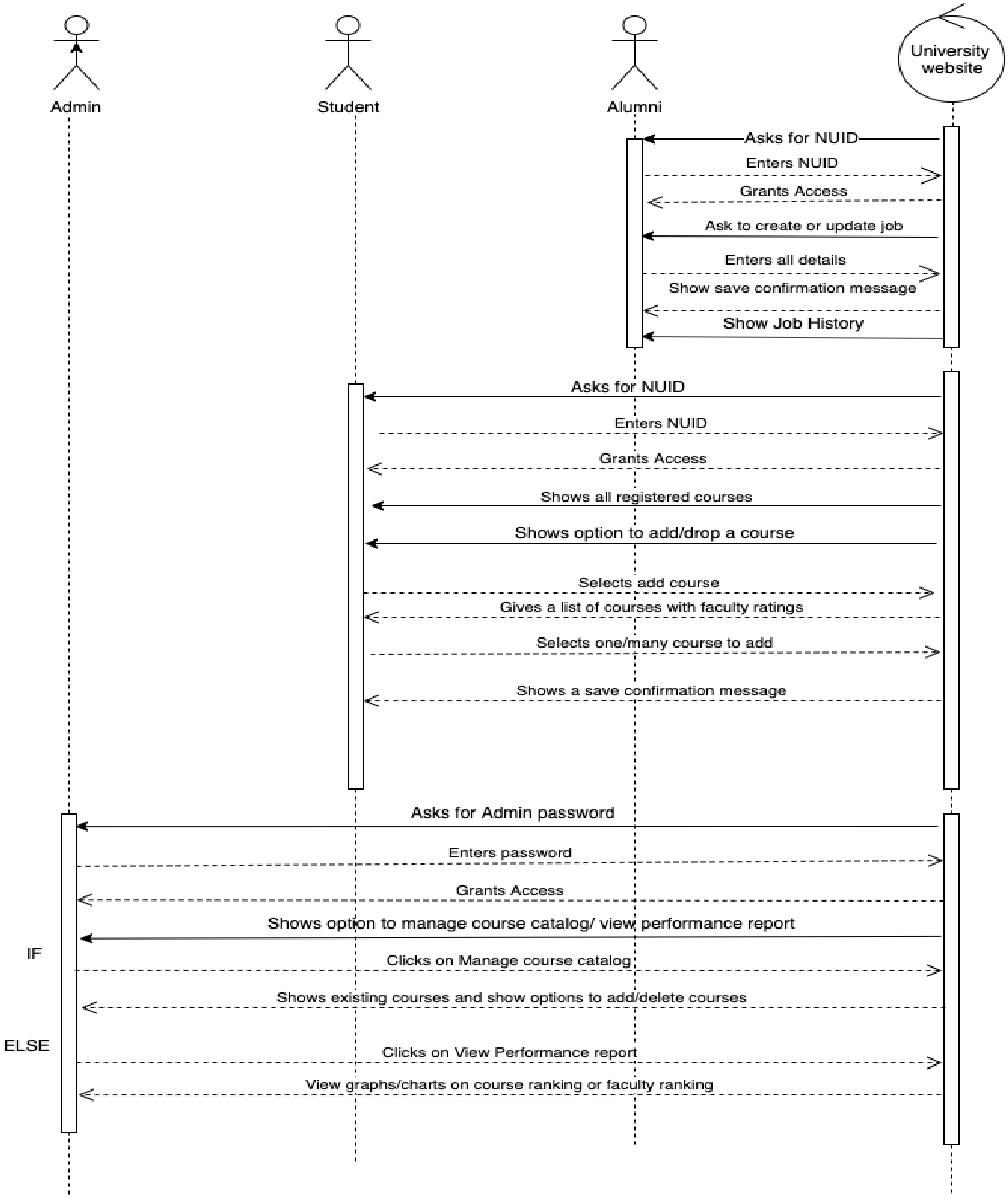
Admin
Student
Alumni
University website
Asks for NUID
Enters NUID
Grants Access
Ask to create or update job
Enters all details
Show save confirmation message
Show Job History
Asks for NUID
Enters NUID
Grants Access
Shows all registered courses
Shows option to add/drop a course
Selects add course
Gives a list of courses with faculty ratings
Selects one/many course to add
Shows a save confirmation message
Asks for Admin password
Enters password
Grants Access
Shows option to manage course catalog/ view performance report
IF
Clicks on Manage course catalog
Shows existing courses and show options to add/delete courses
ELSE
Clicks on View Performance report
View graphs/charts on course ranking or faculty ranking

Figure 9: Sequence diagram for the Proposed solution

CLASS DIAGRAM

The following class diagram shows the distinct set of objects involved to obtain a university's ranking model. Brief details are as follows –

The abstract user class is used by Student and Faculty/Admins to maintain their respective credentials. Determination of whether a student is currently enrolled or is an alumnus is stored as a Boolean for every student.

Every time an alumnus wants to input a job (new job or a promotion), alumni would need to add all the job-related details along with the top 3 courses that helped and rate the same.

Faculty/Admin module allows the University to add new course offerings for upcoming semesters and also view the performance monitor

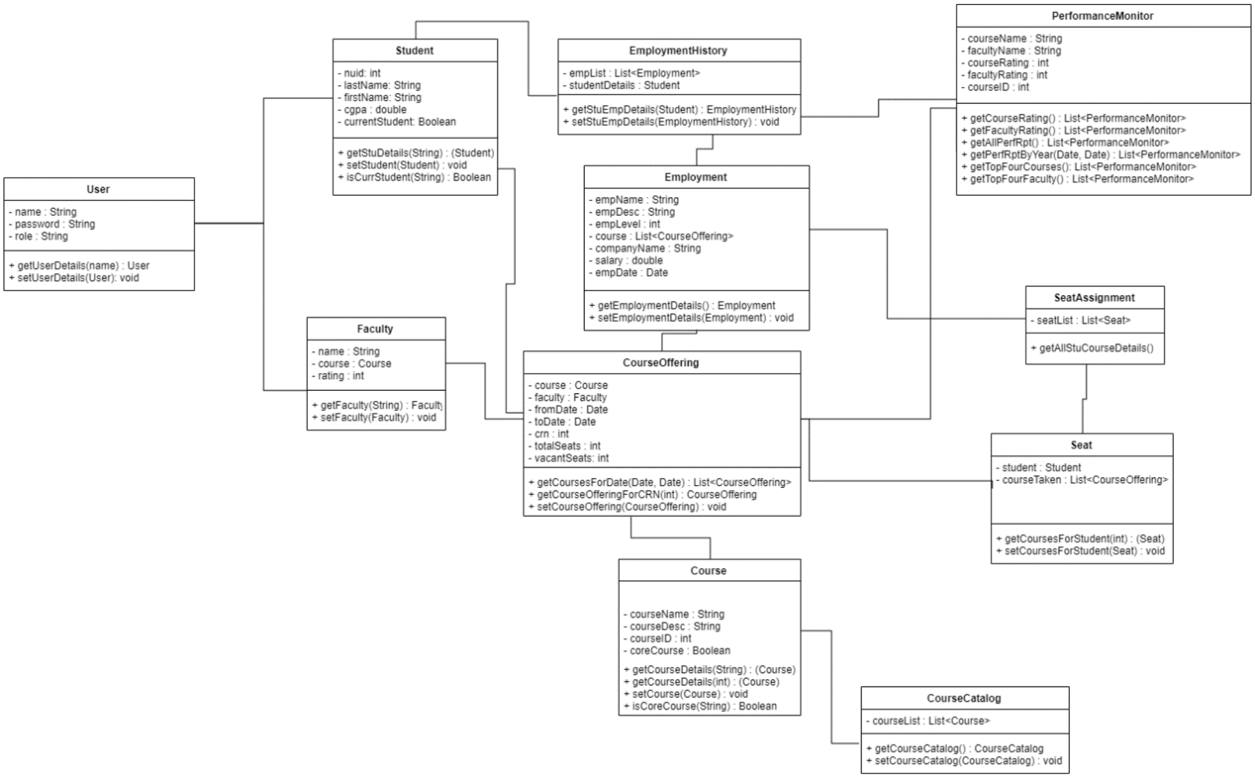
Student
- nuid: int
- lastName: String
- firstName: String
- cgpa : double
- currentStudent: Boolean
+ getStuDetails(String) : (Student)
+ setStudent(Student) : void
+ isCurrStudent(String) : Boolean
EmploymentHistory
- empList : List<Employment>
- studentDetails : Student
+ getStuEmpDetails(Student) : EmploymentHistory
+ setStuEmpDetails(EmploymentHistory) : void
PerformanceMonitor
- courseName : String
- facultyName : String
- courseRating : int
- facultyRating : int
- courseID : int
+ getCourseRating() : List<PerformanceMonitor>
+ getFacultyRating() : List<PerformanceMonitor>
+ getAllPerfRpt() : List<PerformanceMonitor>
+ getPerfRptByYear(Date, Date) : List<PerformanceMonitor>
+ getTopFourCourses(): List<PerformanceMonitor>
+ getTopFourFaculty() : List<PerformanceMonitor>
User
- name : String
- password : String
- role : String
+ getUserDetails(name) : User
+ setUserDetails(User): void
Employment
- empName : String
- empDesc : String
- empLevel : int
- course : List<CourseOffering>
- companyName : String
- salary : double
- empDate : Date
+ getEmploymentDetails() : Employment
+ setEmploymentDetails(Employment) : void
SeatAssignment
- seatList : List<Seat>
+ getAllStuCourseDetails()
Faculty
- name : String
- course : Course
- rating : int
+ getFaculty(String) : Faculty
+ setFaculty(Faculty) : void
CourseOffering
- course : Course
- faculty : Faculty
- fromDate : Date
- toDate : Date
- crn : int
- totalSeats : int
- vacantSeats: int
+ getCoursesForDate(Date, Date) : List<CourseOffering>
+ getCourseOfferingForCRN(int) : CourseOffering
+ setCourseOffering(CourseOffering) : void
Seat
- student : Student
- courseTaken : List<CourseOffering>
+ getCoursesForStudent(int) : (Seat)
+ setCoursesForStudent(Seat) : void
Course
- courseName : String
- courseDesc : String
- courseID : int
- coreCourse : Boolean
+ getCourseDetails(String) : (Course)
+ getCourseDetails(int) : (Course)
+ setCourse(Course) : void
+ isCoreCourse(String) : Boolean
CourseCatalog
- courseList : List<Course>
+ getCourseCatalog() : CourseCatalog
+ setCourseCatalog(CourseCatalog) : void

Figure 10: Class diagram for the Proposed solution

RESULTS

The below pie chart is a result of data collected from the alumni over the last 5 years. It's a cumulative data of the ratings for each subject.

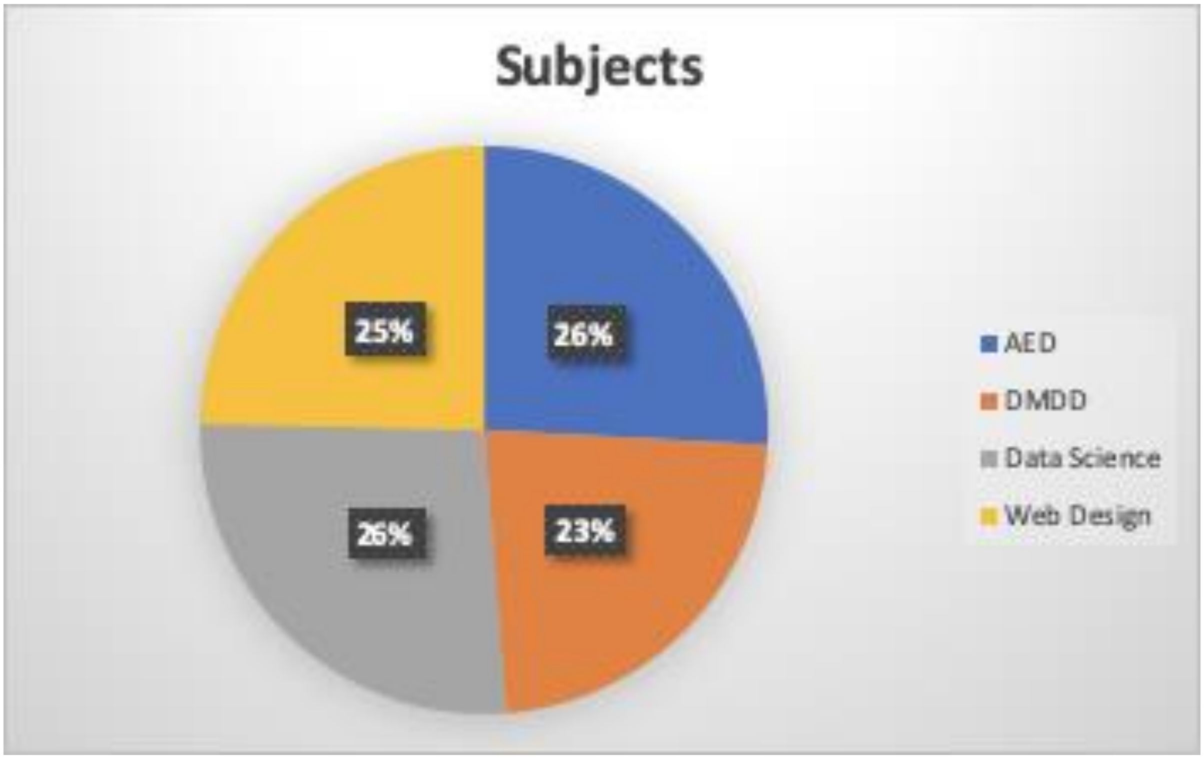

Figure 11: Sample example chart of the performance analysis

CONCLUSION

i	*With the help of the sequence and the class diagram, we can understand the proposed solution and make the system efficient to show how a course/faculty is contributing to job/promotion attainment of the students.*