DATABASE MANAGEMENT SYTEMS FINAL PROJECT REPORT

PROJECT NAME: UNIVERSITY COURSE MANAGEMENT SYSTEM

Under the guidance of:

Dr. Jose Annunziato

Submitted By:

SUJITH MAHENDRAKAR SID: 001733750

MUTHANNA CHITTIAPPA SID: 001996895

ABHISEK AGRAWAL SID:001908953

1 PROJECT STATEMENT

In the project, University Course Management System web application, we have created a web app that maintains the activities of the professors and students. These activities vary from the enrollment of students in a particular semester to the reviews a professor may receive in a particular course that he teaches. To begin, a person who registers can either be a professor or a student. If the person who registers is a professor, the professor is given access to his/her particular page wherein the professor is allowed to add the courses that he/she teaches. A professor may also drop a course if they want to. Suppose the professor selects a particular course, this should take the professor to that particular course information. In this course page the professor is allowed to add assignments and moreover make announcements that they want. The professor may also view the ratings and reviews that they have received.

Now, moving on, if the person who registers is a student, the student is given access to their page. Next, in the student page, the student can select the semester that they he/she has enrolled in or the student can view the previous semesters that the student has taken. When a student selects the semester that they have taken, the student is shown the courses that they have taken for the particular semester. So, the student page also consists of a notifications tab that would provide the student with the notifications for the courses that the student has taken in the current semester. Going back, when a student selects a particular course, the student is provided with the information about that particular course and the homework that are to be done and submitted with their due dates. For a particular course, the student is also allowed to

add reviews and ratings. In todays, world people are more inclined to mobile phones and tablets. So we thought it would be greatly useful if we could make our webpage responsive.

To summarize it, we plan to create a link between the professors, students, the courses they teach and have taken respectively, the ratings and reviews that are received, the assignments that are to be done and submitted. So, this is our problem statement. In the next step, we shall see the proposed solution to our problem statement.

2 PROPOSED SOLUTION

So what we have proposed is to create a web application that would help us maintain and solve our problem statement. In this section is the detailed explanation about how we go about solving our problem. So what we have done is create class for person and a person can either be a professor or a student. So we have created classes for professors and students. So now the main point of access would depend on the id of the person. If the corresponding id is that of the professor, it would direct him to that particular page, otherwise it would direct the student to his particular page. Next, we created a class called semester. The semester class is to get the semester of that particular student. When we get the semester of a particular student, we need to get the courses that the student has registered in that particular semester. Moreover, we also need to specify the courses the professor teaches. So in order for us to that we got to create a new class called course. So the class course does what it needs to do with respect to the professor and the student. Next, while considering the courses, courses have events. So these events are to be displayed in the homepage of the student as notifications. And the

professor is the one who puts the data. Basically he is the one who adds the events. So, in order to do so, we created a class called event. The event class would give the professor access to add events for a particular course he/she teaches and these would be displayed in the homepage of the student as notifications. Next up is for the student to add reviews. To do so, we created a class called Review. In the Review class the student is allowed to give reviews for a particular course. So class Review would be connected to the class Course. The student would be allowed to post reviews and ratings for a particular course and the professor would be allowed to view it. We also need to manage the assignments that we had proposed. So now, the assignments are related to courses a particular student takes. So we create Assignment class. This assignment class would allow the professor to post the assignment. An assignment needs to be submitted by a particular student. So we need to create a submission class and this would take care of the submissions hopefully. So here again this is just the proposed solution. Next, let us have a look at the architecture of our project.

3 ARCHITECTURE

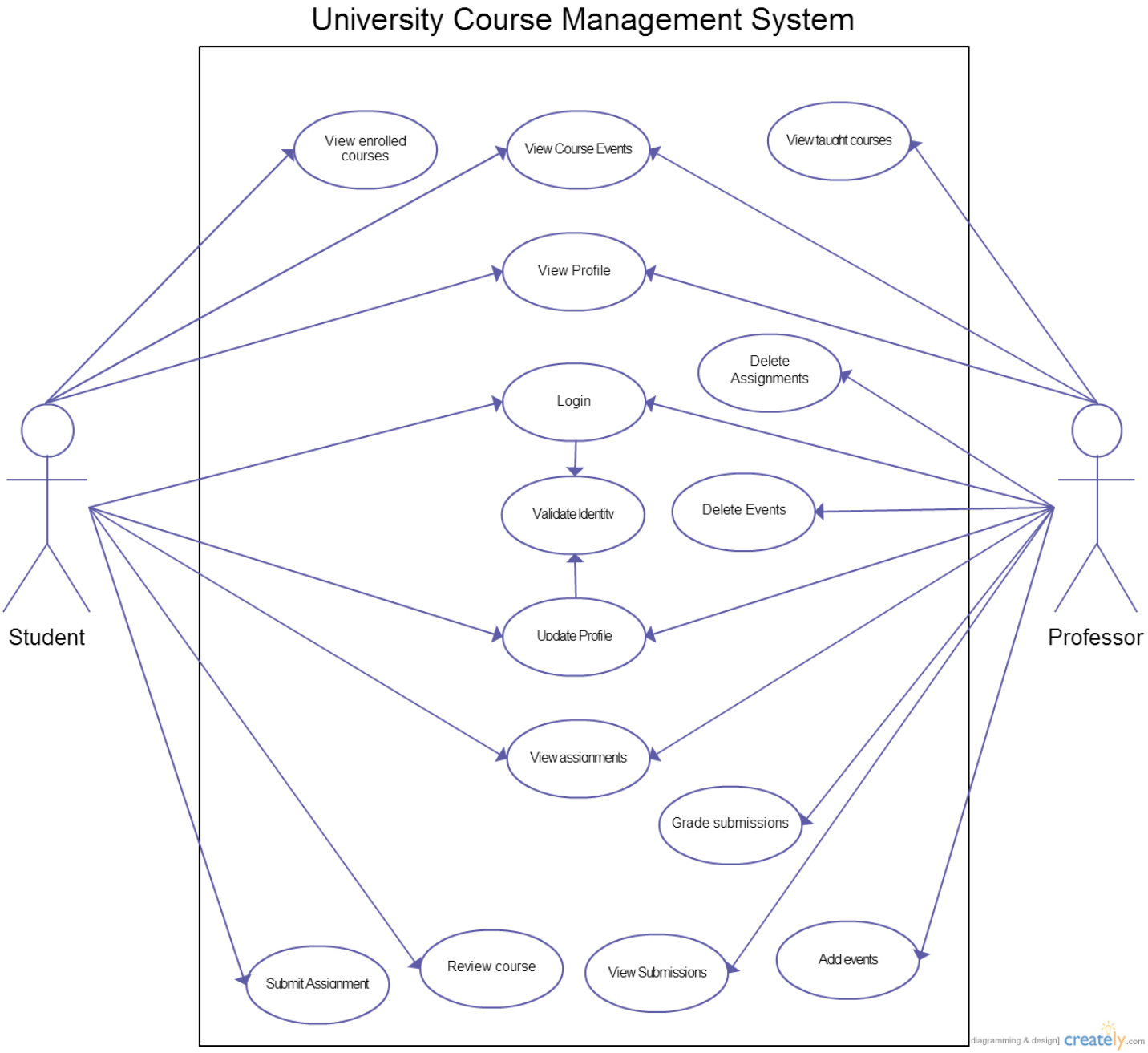
For the architecture of our project we have mainly used the Java Persistence API (JPA) as our object relational model. We have also used the JSON Web Signature (JWS). The implementation of the JPA in our project is just configuring the project facets to include JPA and then applying the changes made to the project. As for the JWS we got to add the Jackson and jersey jar file libraries and this would help our project run as a web service. These are mainly the two parts that we have used in our project. Regarding the UI we have deployed AJAX. This is mainly because we wanted to have a sophisticated level of system visibility. The users should

be able to take note on what is happening behind the scenes and the users would also need to verify if their actions have led to the expected results. We mainly used AJAX to help update parts and portions of our webpage without having to refresh the whole page. AJAX also brings in a whole other level of responsiveness and interactivity.

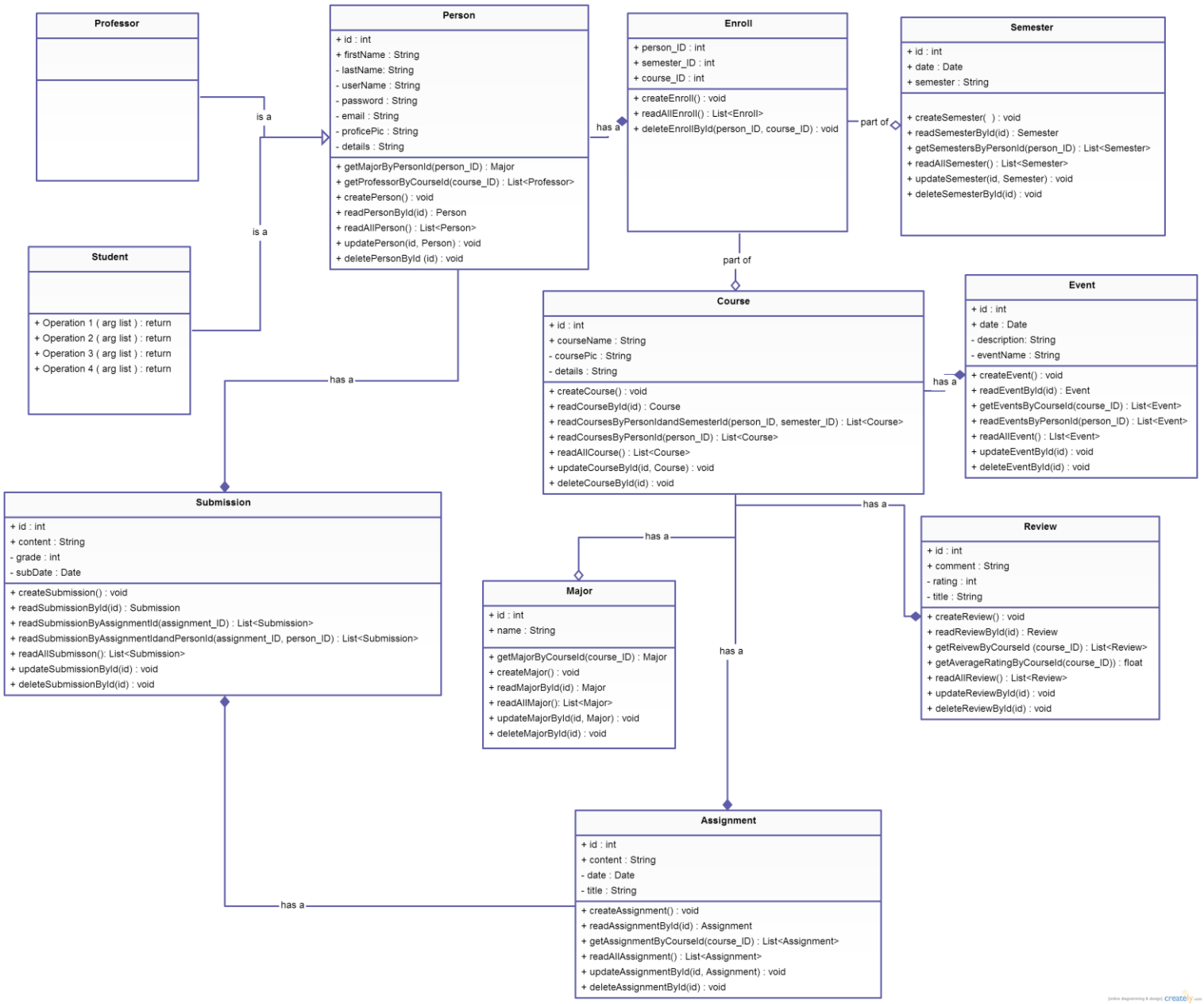
4 TECHNOLOGY STACK USED

1) Eclipse: Eclipse is the Integrated Development Environment (IDE) that we have used in our project.
2) JSON: Java Script Object Notation (JSON) is the data-interchange model that the project uses. Since our project requires us to exchange data between a server and a web application, we use JSON.
3) Jackson Jar files: Jackson is a high performing JSON processor java library.
4) Jersey Jar files
5) JSON Web Signature (JWS): JWS is used to secure content with digital signatures using JSON data structures.
6) Java Persistence API (JPA): We use the JPA because it is an ORM that forms a relation between the database and the web application.
7) AJAX: AJAX is used to develop the UI and we would
8) MySQL database: This technology is used to maintain our database.
9) JQuery: This library is used as a cross platform to simply the content on the client side.

5 USE CASES


University Course Management System
View enrolled courses
View Course Events
View taught courses
View Profile
Delete Assignments
Login
Validate Identity
Delete Events
Update Profile
View assignments
Grade submissions
Submit Assignment
Review course
View Submissions
Add events
Student
Professor
diagramming & design] creately.com

6 DATA MODEL


Professor
Person
+ id : int
+ firstName : String
- lastName: String
- userName : String
- password : String
- email : String
- proficePic : String
- details : String
+ getMajorByPersonId(person_ID) : Major
+ getProfessorByCourseId(course_ID) : List<Professor>
+ createPerson() : void
+ readPersonById(id) : Person
+ readAllPerson() : List<Person>
+ updatePerson(id, Person) : void
+ deletePersonById (id) : void
Enroll
+ person_ID : int
+ semester_ID : int
+ course_ID : int
+ createEnroll() : void
+ readAllEnroll() : List<Enroll>
+ deleteEnrollById(person_ID, course_ID) : void
Semester
+ id : int
+ date : Date
+ semester : String
+ createSemester( ) : void
+ readSemesterById(id) : Semester
+ getSemestersByPersonId(person_ID) : List<Semester>
+ readAllSemester() : List<Semester>
+ updateSemester(id, Semester) : void
+ deleteSemesterById(id) : void
Student
+ Operation 1 (arg list) : return
+ Operation 2 (arg list) : return
+ Operation 3 (arg list) : return
+ Operation 4 (arg list) : return
Course
+ id : int
+ courseName : String
- coursePic : String
- details : String
+ createCourse() : void
+ readCourseById(id) : Course
+ readCoursesByPersonIdandSemesterId(person_ID, semester_ID) : List<Course>
+ readCoursesByPersonId(person_ID) : List<Course>
+ readAllCourse() : List<Course>
+ updateCourseById(id, Course) : void
+ deleteCourseById(id) : void
Event
+ id : int
+ date : Date
- description: String
- eventName : String
+ createEvent() : void
+ readEventById(id) : Event
+ getEventsByCourseId(course_ID) : List<Event>
+ readEventsByPersonId(person_ID) : List<Event>
+ readAllEvent() : List<Event>
+ updateEventById(id) : void
+ deleteEventById(id) : void
Submission
+ id : int
+ content : String
- grade : int
- subDate : Date
+ createSubmission() : void
+ readSubmissionById(id) : Submission
+ readSubmissionByAssignmentId(assignment_ID) : List<Submission>
+ readSubmissionByAssignmentIdandPersonId(assignment_ID, person_ID) : List<Submission>
+ readAllSubmisson(): List<Submission>
+ updateSubmissionById(id) : void
+ deleteSubmissionById(id) : void
Major
+ id : int
+ name : String
+ getMajorByCourseId(course_ID) : Major
+ createMajor() : void
+ readMajorById(id) : Major
+ readAllMajor(): List<Major>
+ updateMajorById(id, Major) : void
+ deleteMajorById(id) : void
Review
+ id : int
+ comment : String
- rating : int
- title : String
+ createReview() : void
+ readReviewById(id) : Review
+ getReivewByCourseId (course_ID) : List<Review>
+ getAverageRatingByCourseId(course_ID)) : float
+ readAllReview() : List<Review>
+ updateReviewById(id) : void
+ deleteReviewById(id) : void
Assignment
+ id : int
+ content : String
- date : Date
- title : String
+ createAssignment() : void
+ readAssignmentById(id) : Assignment
+ getAssignmentByCourseId(course_ID) : List<Assignment>
+ readAllAssignment() : List<Assignment>
+ updateAssignmentById(id, Assignment) : void
+ deleteAssignmentById(id) : void
is a
is a
has a
part of
part of
has a
has a
has a
has a
has a
has a