Software Requirements and Design Document

For

Group 3

Version 1.0

Authors:
Marija Travoric
Jeyma Rodriguez
Jada Doby
Laura Obermaier
Margaret Rivas

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

A redesigned version of 'RateMyProfessor' , ProFessUp is a web application that allows users to rate professors. Users can filter, add, delete, and view professors; but are required to create an account or login in order to write reviews. Professors can be viewed individually, where reviews can be filtered by courses, which can be added and deleted. Only the user who wrote a review can delete it and users can edit their account data and delete the account on their account page. Users can opt for their review to appear anonymous and may select various mechanisms of rating a professor such as professor availability, extra credit opportunities, and attendance requirements. An overall score is displayed for each professor. Pages featured include the main mage "/", the sign-in page "/signIn.html", the sign-up page "/signUp.html", the professors page "/searchResults.html", and the account page "/account.html".

2. Functional Requirements (10 points)

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

- *The user being able to sign into ProFessUp - High*
- *The user being able to sign up for ProFessUp with their name and email - High*
- *Being able to use search for my professor and add professor to the database - High*
- *Restrict delete functionality for administration only – For Professors and Courses - High*
- *The system shall do allow user to add or search for courses once course implementation is completed – High*
- *The system shall allow the user to submit reviews and be able to view reviews by courses- High*
- *User should be able to add professors- High*
- *The system shall store all information inserted on the site to our mongo database- High*

3. Non-functional Requirements (10 points)

*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

1.User Experience: The user interface will be intuitive and user-friendly, ensuring that students can easily navigate through the application and provide reviews about professors by the courses they are teaching. We are also allowing the user the capability to add courses and professors.

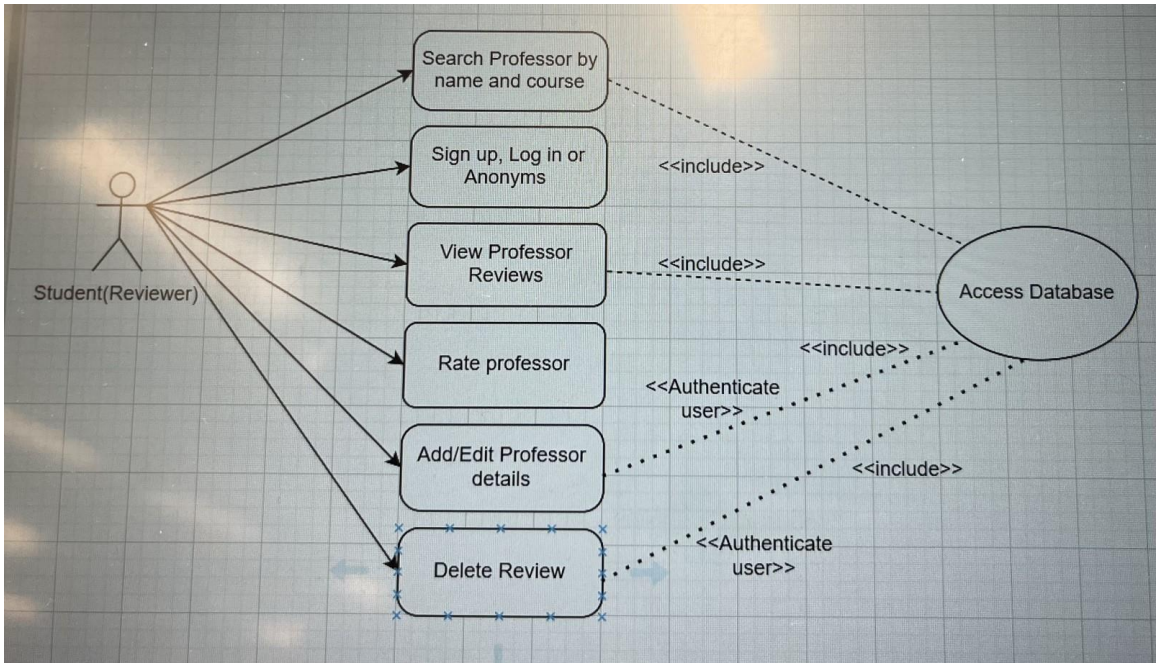
2.Anonymity and Privacy: The application will guarantee the anonymity of students providing reviews, ensuring that their identities are never revealed in the course evaluation process. Also, if they don't care to be anonymous, allow them to state their feedback while their identity is visible.

3.Data Accuracy: The application will use data from the Florida State University's main campus faculty records to ensure the information provided to students is accurate. Also incorporating the options of electives for the current year.

4.Security: The application must implement a secure sign-in process, requiring users to provide valid credentials (username and password) to access their accounts if they choose to be identifiable while leaving their reviews

4. Use Case Diagram (10 points)

*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*
***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



5. Class Diagram and/or Sequence Diagrams (15 points)

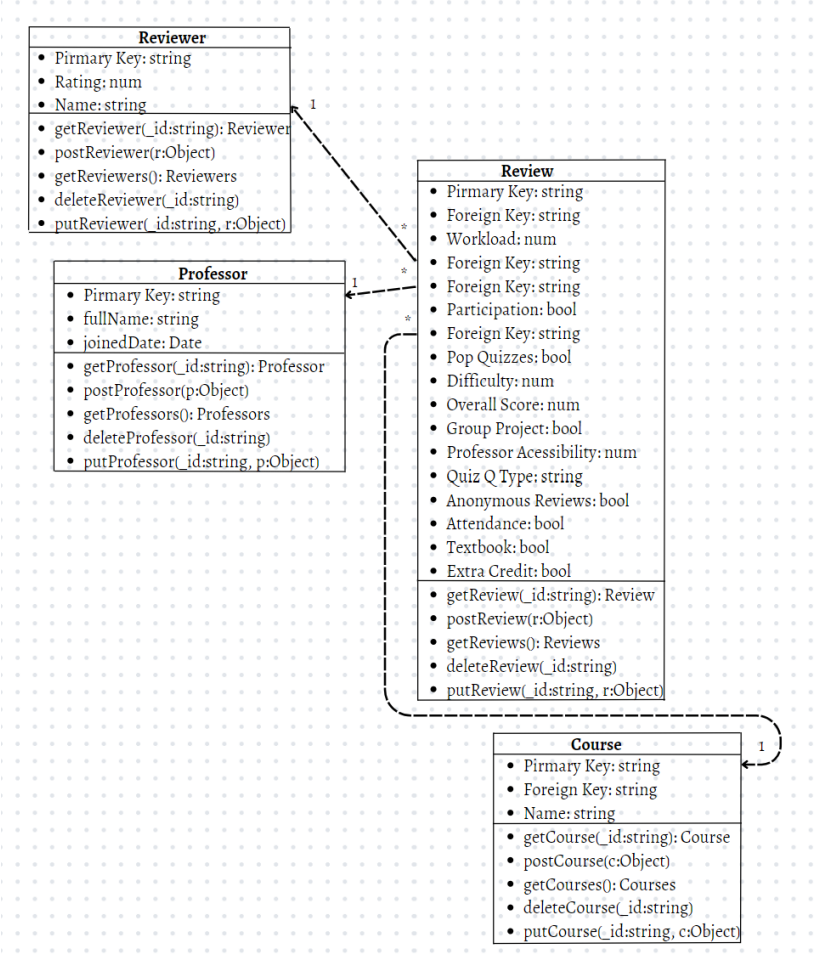
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

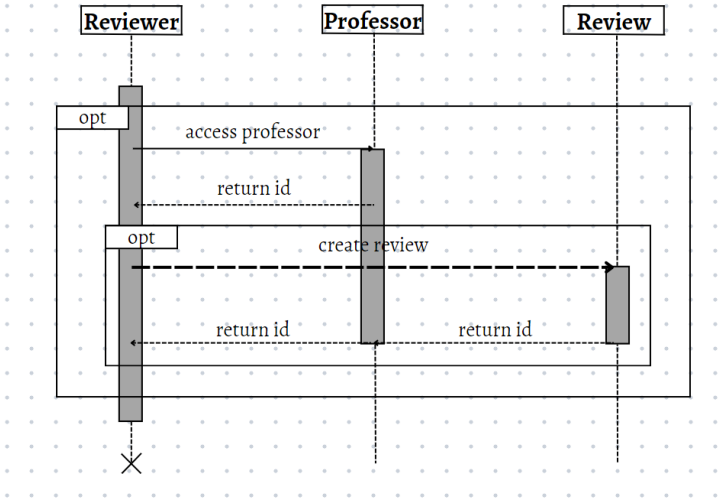
*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

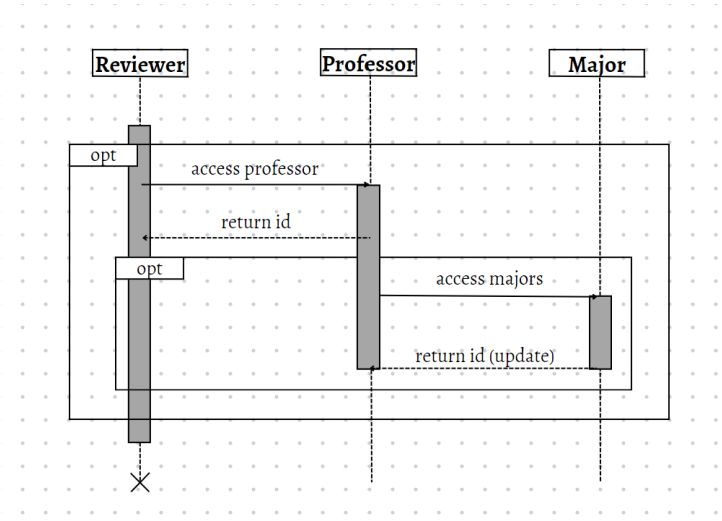
*If the main **paradigm** in your system is **not** **Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

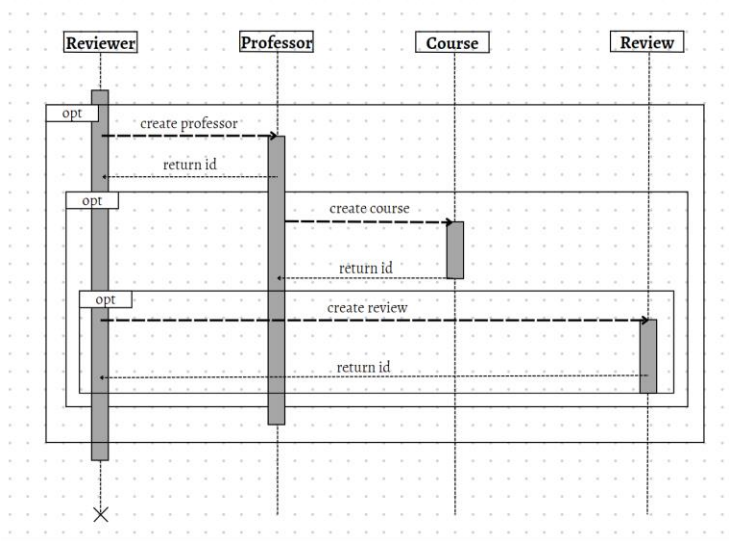
***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*


Reviewer
Pirmary Key: string
Rating: num
Name: string
getReviewer(_id:string): Reviewer
postReviewer(r:Object)
getReviewers(): Reviewers
deleteReviewer(_id:string)
putReviewer(_id:string, r:Object)
Professor
Pirmary Key: string
fullName: string
joinedDate: Date
getProfessor(_id:string): Professor
postProfessor(p:Object)
getProfessors(): Professors
deleteProfessor(_id:string)
putProfessor(_id:string, p:Object)
Review
Pirmary Key: string
Foreign Key: string
Workload: num
Foreign Key: string
Foreign Key: string
Participation: bool
Foreign Key: string
Pop Quizzes: bool
Difficulty: num
Overall Score: num
Group Project: bool
Professor Acessibility: num
Quiz Q Type: string
Anonymous Reviews: bool
Attendance: bool
Textbook: bool
Extra Credit: bool
getReview(_id:string): Review
postReview(r:Object)
getReviews(): Reviews
deleteReview(_id:string)
putReview(_id:string, r:Object)
Course
Pirmary Key: string
Foreign Key: string
Name: string
getCourse(_id:string): Course
postCourse(c:Object)
getCourses(): Courses
deleteCourse(_id:string)
putCourse(_id:string, c:Object)
1
*
1
*
1
*


Reviewer
Professor
Review
opt
access professor
return id
opt
create review
return id
return id





6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

Hardware Platform: The software will operate on standard computer hardware, including desktops and laptops.

-Operating Systems:

- *Windows*
- *macOS*

-Web Browsers: The application will be accessible via any popular web browsers.

-Database Management System: MongoDB

-Development Tools:

- *Programming Language: The application is developed using React with JavaScript.*
- *Integrated Development Environment (IDE): Visual Studio Code Package*
- *Managers: NPM (Node Package Manager) will be used for managing software dependencies and packages within the project.*
- *Node: node will be used for running the serverside code*

7. Assumptions and Dependencies (5 points)

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

External API's and Databases: We are using built-in API elements from React framework. We are using MongoDB for our database management. Also, the customized the application is exclusively for Florida State University computer science students. We are utilizing React and Node.js for our website as well.