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) Jada

ProfFessUp is a more secure, data-verified rate for my professor application. These tools will give students the ability to have an anonymous identity while revealing honest truths about their course experience. Also, the way the data would be more accurate and accessible through our application is we filter straight from the University's public staff and faculty records. In addition to that, this tool allows students to select the best options for them, and it will produce a recommendation of teachers with those course qualities. Overall, ProfFessUp is the new upcoming way universities could help students choose better learning environments for themselves. (Jada Doby)

2. Functional Requirements (10 points) Jada

*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.*

1.The page for recommendation for students to select choices that important to them so they can be matched with professor that have those qualities in their courses – Medium Priority
2.The system shall search professor by name – High Priority
3.The system will filter through professor search and show the user – High Priority.
4.The system should accumulate all the individual reviews and output the overall scores that will be presented to the reviewers – High Priority

Non-functional Requirements (10 points) Marija

*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 feedback without confusion.

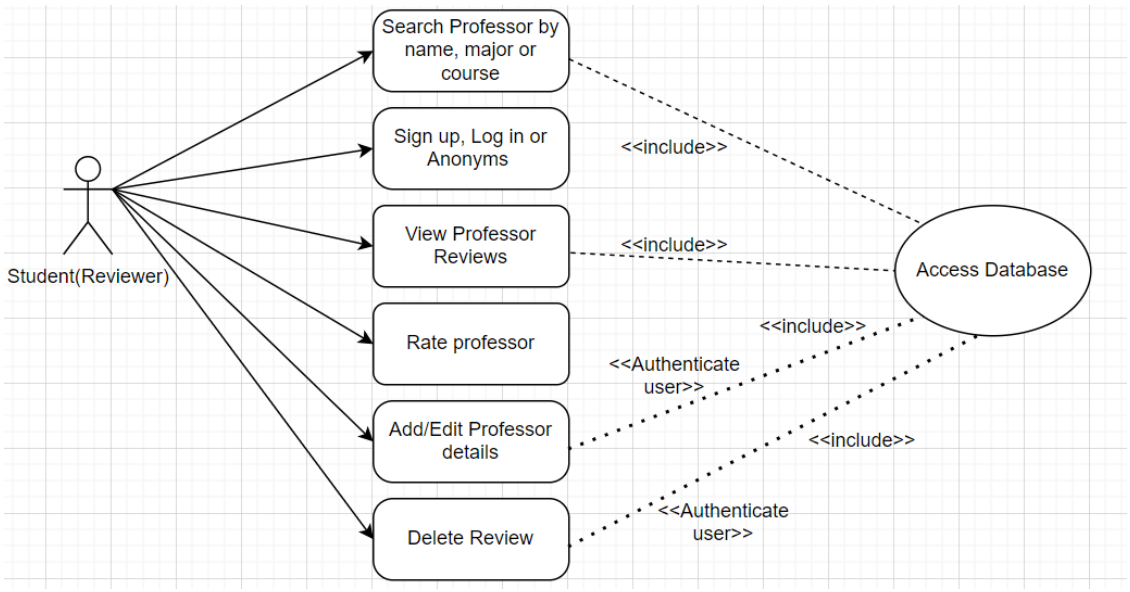
2.Anonymity and Privacy: The application will guarantee the anonymity of students providing feedback, ensuring that their identities are never revealed in the course evaluation process.

3.Data Accuracy: The application will use data from the University's public staff and faculty records to ensure the information provided to students is accurate.

4.Security: The application must implement a secure sign-in process, requiring users to provide valid credentials (username and password) to access their accounts.

3. Use Case Diagram (10 points) Jemya

*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.*



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

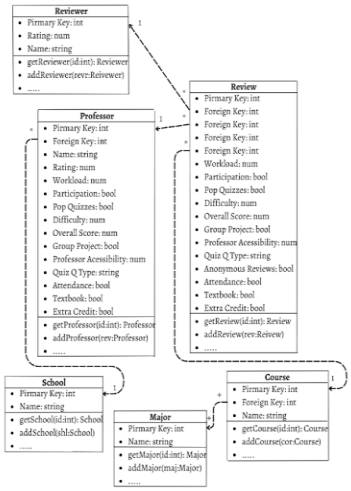
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

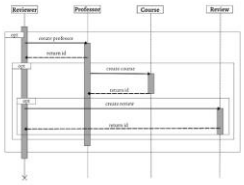
*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

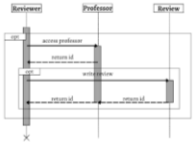
*If the main **paradigm** in your system is **not Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams,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

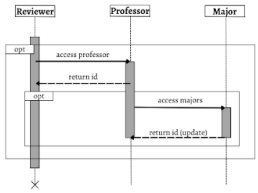
***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). 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: int
Rating: num
Name: string
getReviewer(id:int): Reviewer
addReviewer(revr:Reivewer)
.....
Review
Pirmary Key: int
Foreign Key: int
Foreign Key: int
Foreign Key: int
Foreign Key: int
Workload: num
Participation: bool
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:int): Review
addReview(revr:Reivew)
.....
Professor
Pirmary Key: int
Foreign Key: int
Name: string
Rating: num
Workload: num
Participation: bool
Pop Quizzes: bool
Difficulty: num
Overall Score: num
Group Project: bool
Professor Acessibility: num
Quiz Q Type: string
Attendance: bool
Textbook: bool
Extra Credit: bool
getProfessor(id:int): Professor
addProfessor(revr:Professor)
.....
School
Pirmary Key: int
Name: string
getSchool(id:int): School
addSchool(shl:School)
.....
Major
Pirmary Key: int
Name: string
getMajor(id:int): Major
addMajor(maj:Major)
.....
Course
Pirmary Key: int
Foreign Key: int
Name: string
getCourse(id:int): Course
addCourse(cor:Course)
.....


Reviewer
Professor
Course
Review
opt
create professor
return id
opt
create course
return id
opt
create review
return id


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


Reviewer
Professor
Major
opt
access professor
return id
opt
access majors
return id (update)

5. Operating Environment (5 points) Marija

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 TypeScript.*
- *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.*

6. Assumptions and Dependencies (5 points) Marget

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 plan to use external API's from RateMyProfessor and others to help us have a solid ratings page (for demos and more). We are using MongoDB for our database. Also, we plan to customize the app to each school, so we need data on the school's colors and more. Any changes or disruptions in these third-party services could affect the functionality and user experience of the platform.

We plan to utilize React and Node.js for our website as well and we are not very familiar with utilizing Typescript so this might be a potential issue in development speed.