Software Enginnering Fall 2023-2024
VALUNI Design Specifications

Hussein Heggi Ahmed Waseem Raslan Sarah Elsamanody Nour Abdalla Youssef Elmahdy
Ahmed Elbarbary Ahmed Jaheen

November 26, 2023

Contents

1 Project Proposal

VALUNI is a platform where students have the opportunity to share their experiences and exchange knowledge and feedback regarding both professors and courses. It would play a vital role in helping students hold sufficient knowledge that could ease up the process of choosing classes and preparing for exams. Our objective is to create a safe, dependable and organized environment that allows students to express their opinions in an honest and respectful way. We aim to motivate students when rating or writing their reviews through anonymity, and at the same time having certain guidelines that must be followed when using VALUNI. This will not only assist students, but also the overall University community including diverse facilities and professors.

1.1 Usage

• Search for professors and courses feedback.

• Rate professors on a variety of criteria which are explanation, fairness, organization, leniency, and accessibility.

• Rate courses based on difficulty level, workload, and learning outcomes.

• Course Evaluations per semester for universities to measure the educational quality.

1.2 Risks

1.2.1 Community Risks

• **Defamation:** Students may post false or misleading reviews of professors, which could damage the professor's reputation and career.

• **Bias:** Student ratings of professors may be biased. For example, distinct students may go through different experiences such as receiving undesired grades , influencing the integrity of their ratings.

• **Increased Reliance on student feedback:** Professors may rely too heavily on student feedback for validation, which could negatively impact their teaching style. In addition, students may depend on the review itself without actually giving the professor a chance.

1.2.2 Development Risks

• The website is not liked by its main target audience as well as not being used by the student body which will defeat the whole purpose of the website.

• The website is not perfectly secured which can lead to privacy issues and data breaches.

• The cost of developing and maintaining the website could be high which could result in problems with financial sustainability.

• The website may not be well-received by professors which can make the website unsuccessful.

1.3 Motivation

Nowadays, the majority of students find it difficult to navigate the right path when it comes to making decisions during their registration periods. It is a well known hustle amongst students as continuously trying to find the suitable courses and professors that fits their best interest is a challenge. In fact, some students initiated solving this problem by creating RateAUCProfessors, a FaceBook group, to provide reviews about Professors and pass on evaluations to next colleagues. Eventually, some of the group members started posting content that is out of context which made it lose its academic focus. Not to mention, the data keeps increasing each semester which makes it difficult for students to search in an efficient way. Therefore, we came up with VALUNI where our main purpose is to constantly and effectively serve the educational community. This is because it would incorporate not only accessible reviews on courses and professors but also help collect evaluations at the end of each semester; aiding the university to adopt rightful and recent evaluations regarding each aspect of the overall educational level.

1.4 MVP Requirements (Requirements to be Implemented)

User authentication (login etc)

User data encryption

Rating Courses

Rating Professors

Writing out a review with Rating

Parameterized Rating

Filtered Search on parameters

Editing Reviews

2 Requirements Specifications

2.1 Product Backlog

2.1.1 Functional Requirements

User authentication

User login and verification that user is a university student

Rating Courses

Rating Professors

Writing out a review with Rating

When Reviewing students have the option to write a review with the parameterized rating

Parameterized Rating

Students can rate professors and courses on different parameters, such as workload, leniency

Filtered Search on parameters

Students can filter out result or search for courses and professors based on parameters

Reporting reviews

Editing Reviews

Admin (university) access to review data

University can access critical semster review analytics as well as overall analytics

Search for courses

Search for professors

2.1.2 Non Functional Requirements

Fluid, Simple UI

Secure

Auto removing profanity on reviews

Prevent Review Bombing

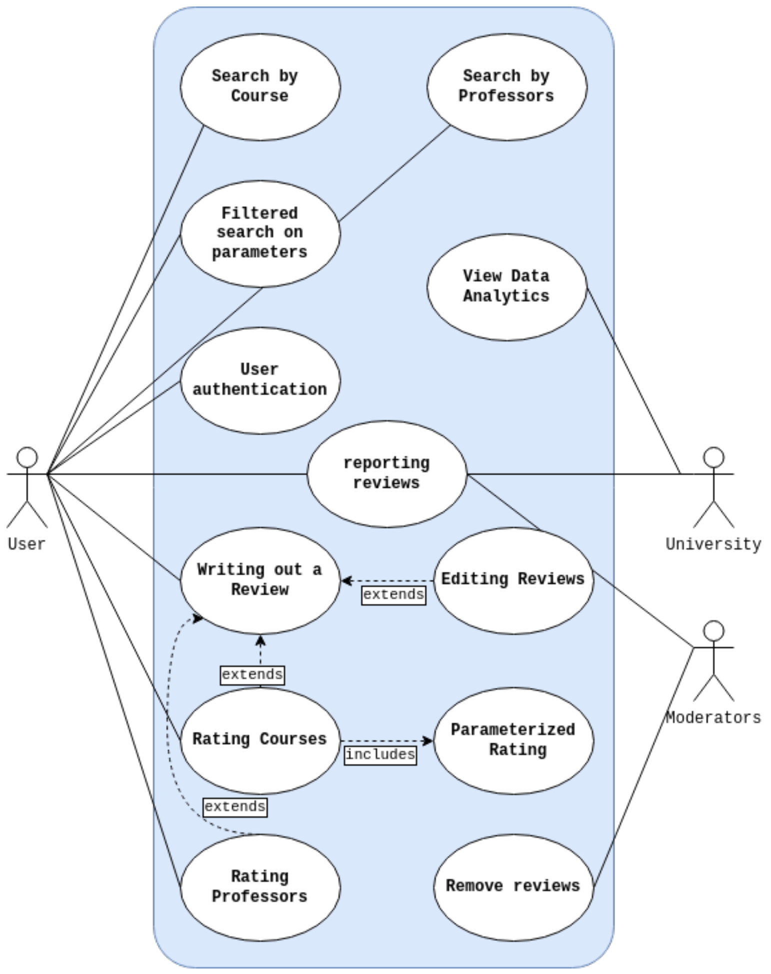
Anonymous

Customer Service (for universities)

Performance

Scalability

2.2 Use Case Diagram



2.3 Use Case Narratives

Use Case #1:

Title: User Authentication

Use Case Description: Authentication during login to ensure the existence of the user account.

Primary Actor: User

Secondary Actor: Firebase

Pre-Condition: User must have an account.

Textual Description:

step	actor	direction
1	User	Opens website
2	Client	Provides Username & Password Field for Users.
3	User	Enters Valid Username & Password
4	Firebase	Verify Username & Password
5	User	Logs in Successfully & Views Homepage

Alternative Flows:
3. User username & password are not valid as user does not exist ; redirect or recommend signing up instead.
4. User username or password are invalid ; ask user to re-enter.

Post-Conditions:
User is authenticated and can access the app, homepage and its features.

Use Case #2:

Title: Search by Course

Use Case Description: Users can navigate through the website by searching over courses. Users use the search bar for desired courses to check its reviews or review it.

Primary Actor: User

Secondary Actor: Database access layer

Pre-Condition: User already logged in and has authentication done. Courses must exist in the database system to be found in search.

Textual Description:

step	actor	direction
1	User	Clicking on Search Menu
2	User	Writes the name of course
3	Data access layer	Searches for course
4	Data access layer	Returns Course
5	Client	Course is viewed

Alternative Flows:
1-User can pick directly from the homepage the course he desires.
2-If course is not found, nothing will be displayed.
Post-Conditions:
Users now have the desired course, they can choose to either read or write a review.

Use Case #3:

Title: Filtered Search on Parameters

Use Case Description: User filters the parameters they require when viewing reviews

Primary Actor: Data access layer

Secondary Actor: Client

Pre-Condition: User exists in order to access the settings and adjust his parameters Textual Description:

step	actor	direction
1	User	Go to settings
2	User	Choose adjust parameters
3	Data access layer	Removes the unwanted parameters
4	User	Confirms and can go back

Alternative Flows: User did not confirm his changes, leaving his parametrized ratings unchanged.

Post-Conditions: User has confirmed his preferred parameters to view.

Use Case #4:

Title: Writing Out a Review

Use Case Description: Users can choose/instructor a course to write a review about

Primary Actor: User

Secondary Actor: Data access layer

Pre-Condition: User successfully logs in as in this use case he is only choosing what to write a review about.

Textual Description:

Step	Actor	Description
1	User	Clicking on Search Menu
2	User	Writes the name of course
3	Data access layer	Searches for course/instructor
4	Data access layer	Returns course/instructor
5	User	Chooses course/instructor

Alternative Flows:
Choose course/instructor from homepage, and add the review.
Post-Conditions:
Users found the course/instructor they were looking to write a review about.

Use Case #5: Title: Rating Courses

Use Case Description:. He also has two options for writing reviews, rating only or adding a detailed review as well.

Primary Actor:User

Secondary Actor: Data access layer

oPre-Condition: User must have taken the course beforehand. Textual Description:

Step	Actor	Description
1	Client	Gives options for writing a review
2	User	Chooses option
3	Client	Field for writing only or writing and rating
4	User	Writes in field given and confirms

Alternative Flows:
Choose the course from the homepage, and add the review.
Post-Conditions:
Users successfully added their review on the course.

Use Case #6:

Title: Rating Professors

Use Case Description: Users can choose an instructor to write a review about. He also has two options for writing reviews, rating only or adding a detailed review as well. Primary Actor:User

Secondary Actor: Data access layer

Pre-Condition: User must have taken the course with the instructor beforehand.

Textual Description:

Step	Actor	Description
1	Client	Gives options for writing a review
2	User	Choses option
3	Client	Field for writing only or writing and rating
4	User	Writes in field given and confirms

Alternative Flows:
Choose the instructor from the homepage, and add the review.
Post-Conditions:
Users successfully added their review on the course.

Use Case #7:

Title: Search by Professors

Use Case Description: Users can navigate through the website by searching over courses which include professors. Users use the search bar for desired professors to check its reviews or review it.

Primary Actor: User

Secondary Actor: Database access layer

Pre-Condition: User already logged in and has authentication done. Instructors must exist in the database system to be found in search.

Textual Description:

Step	Actor	Description
1	User	Clicking on Search Menu
2	User	Writes the name of course
3	Data access layer	Searches for instructor
4	Data access layer	Returns instructor
5	Client	Instructor is viewed

Use Case 8:

Title: View Data Analytics

Use Case Description: University has access to view the data in the database.

Primary Actor: University

Secondary Actor: Database

Pre-Condition: There exists data in the database for the University to have access to.

Textual Description:

Step	Actor	Description
1	University	Requesting data
2	Data access layer	Fetch required data
3	Database	Search required data
4	Data access layer	Return formatted data

Alternative Flows:
No alternative flow
Post-Conditions:
University successfully has access to the database.

Use Case #9:

Title: Editing Reviews

Use Case Description: User can open his posted reviews and edit the content.

Primary Actor: User

Secondary Actor: Data access layer

Pre-Condition: User has already written the review to be edited.

Textual Description:

Step	Actor	Description
1	User	Check for post history
2	User	Choses review
3	User	Edits review
4	User	Confirms
5	Data access layer	Updates changes made

Post-Conditions:
Users edited their desired reviews.
Alternative Flows:
User forgets to confirm, so changes are not saved.
Alternative Flows Post-Conditions:
User review is not edited.

Use Case #10:

Title: Parameterized Rating

Use Case Description: A Rating based on multiple parameters, each parameter contributes to overall rating of professors/courses. User gets to Rate professors/courses based on these parameters.

Primary Actor: User

Secondary Actor: Data Access Layer

Pre-Condition: The User must have
A. taken a course with the professor (professor review) OR
B. taken the course in the current semester AND
C. User has not reviewed professor/course prior

Textual Description:

Step	Actor	Description
1	User	Fills Rating Form
2	Data Access Layer	Sends Rating to DB
3	Database	Stores rating

Use Case #11:

Title: Remove Reviews

Use Case Description: Ability to Remove reviews (Moderator Privilege)

Primary Actor: Moderator

Secondary Actor: Database

Pre-Condition: Review is present

Textual Description:

Step	Actor	Description
1	Moderator	Delete Review
2	Data Access Layer	Signal deleted Review
3	Database	Remove review

Alternative Flows: No alternative flow
Post-Conditions: The Review is deleted

Use Case #12:

Title: Reporting Reviews

Use Case Description: User can optionally report reviews they deem inappropriate.

Primary Actor: Client

Secondary Actor: Moderators

Pre-Condition: There is a review to report.

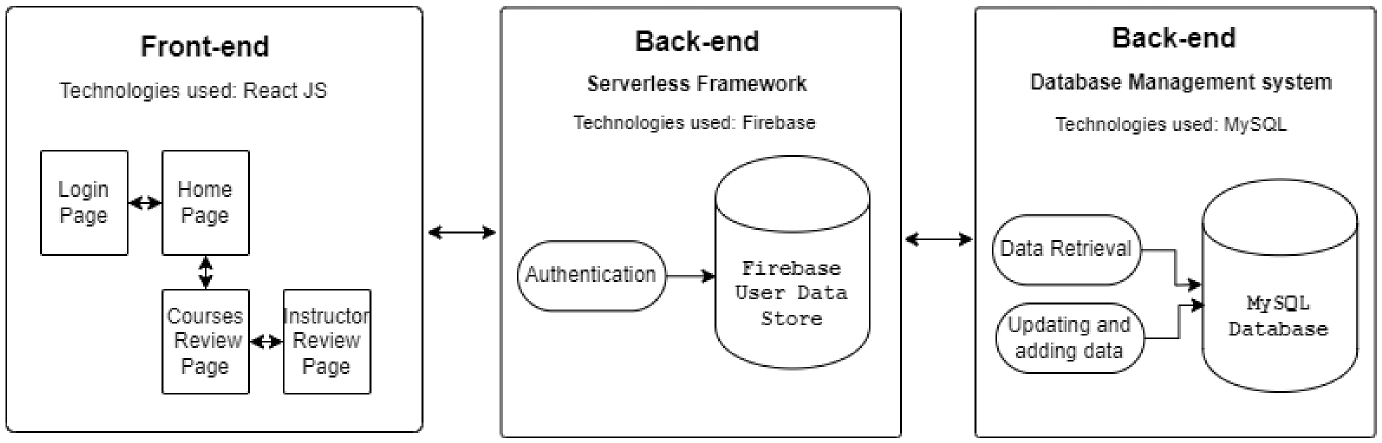
Textual Description:

Step	Actor	Description
1	User	Initiate report
2	Data Access Layer	File the report
3	Moderator	Review the report
4	Moderator	Delete Review

Post-Conditions:
Review is deleted from the database and is updated on the client side.
Alternative Flows:
4- Moderator, Dismiss report
Alternative Flow Post-Conditions:
Review is not deleted

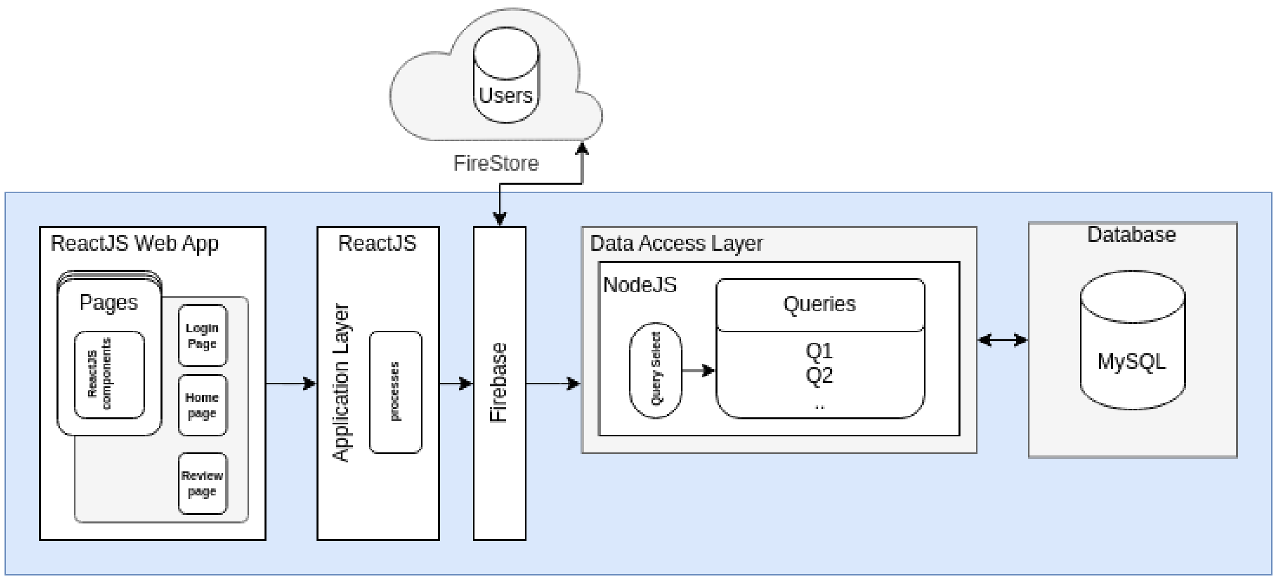
3 Design Specifications

3.1 High Level Architecture



Front-end is implemented using React JS where it is used to implement the login, home, results, and review pages. The authentication of users is done through firebase, a server-less framework, where it contains a database including the email addresses and passwords of the university students. The user's data which is previewed on the web app is accessed through a database on a local host which is managed and manipulated by MySQL workbench. The SQL database includes all necessary information that is needed for Valuni and it is updated whenever a user adds a new review. Lastly, the frontend and backend are connected together using Node js.

3.2 Detailed Level Architecture: Monolithic Layered Architecture

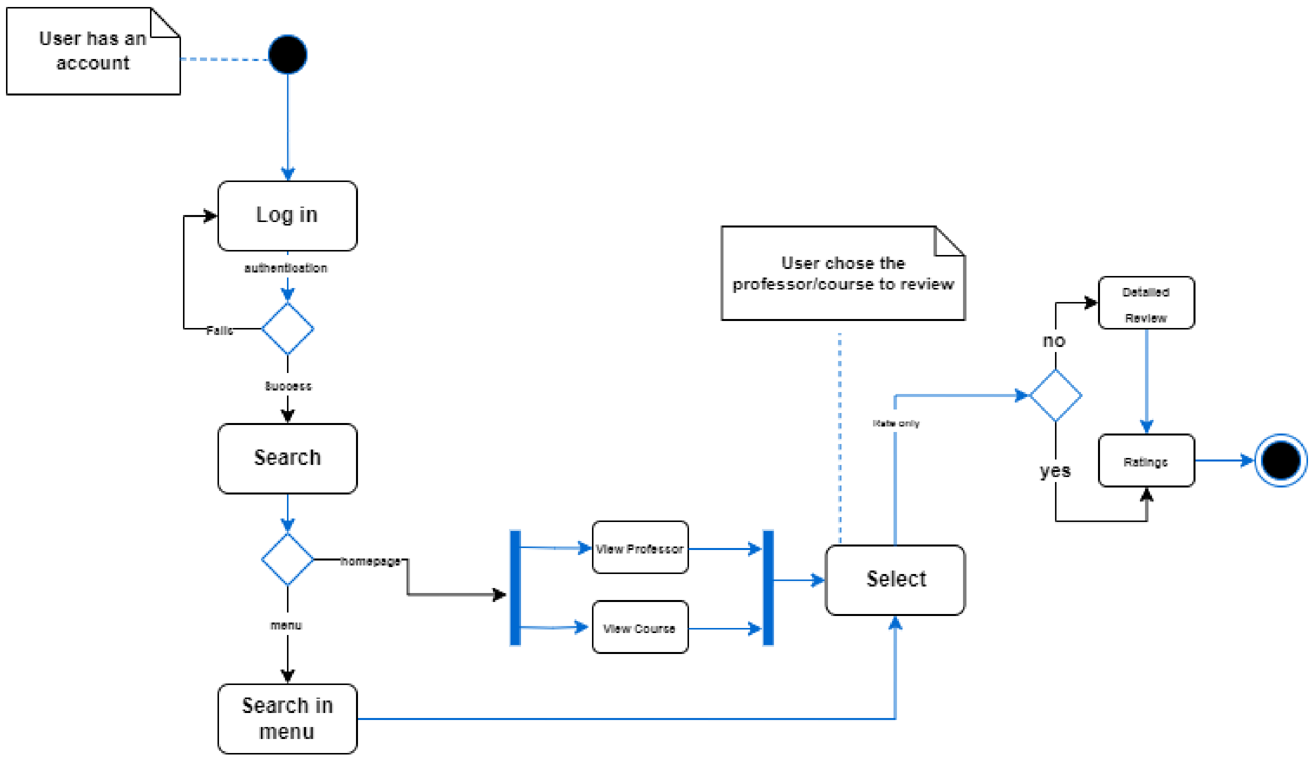


Monolithic architecture is suitable to represent the architecture of the Web app ,Valuni, as this project is not divided into subsystems, but rather the components are interconnected with one another. Therefore, the data being transferred continuously goes through diverse stages and technologies to reach the end-user. This diagram shows a clear path and flow of the Valuni architecture where the web app is built and linked with firebase through ReactJs. After the authentication process is done by firebase, the input is processed by the data access layers using queries which are manipulated and linked through the MySQL database that contains all the necessary data needed by the web application.

3.3 Activity Diagrams

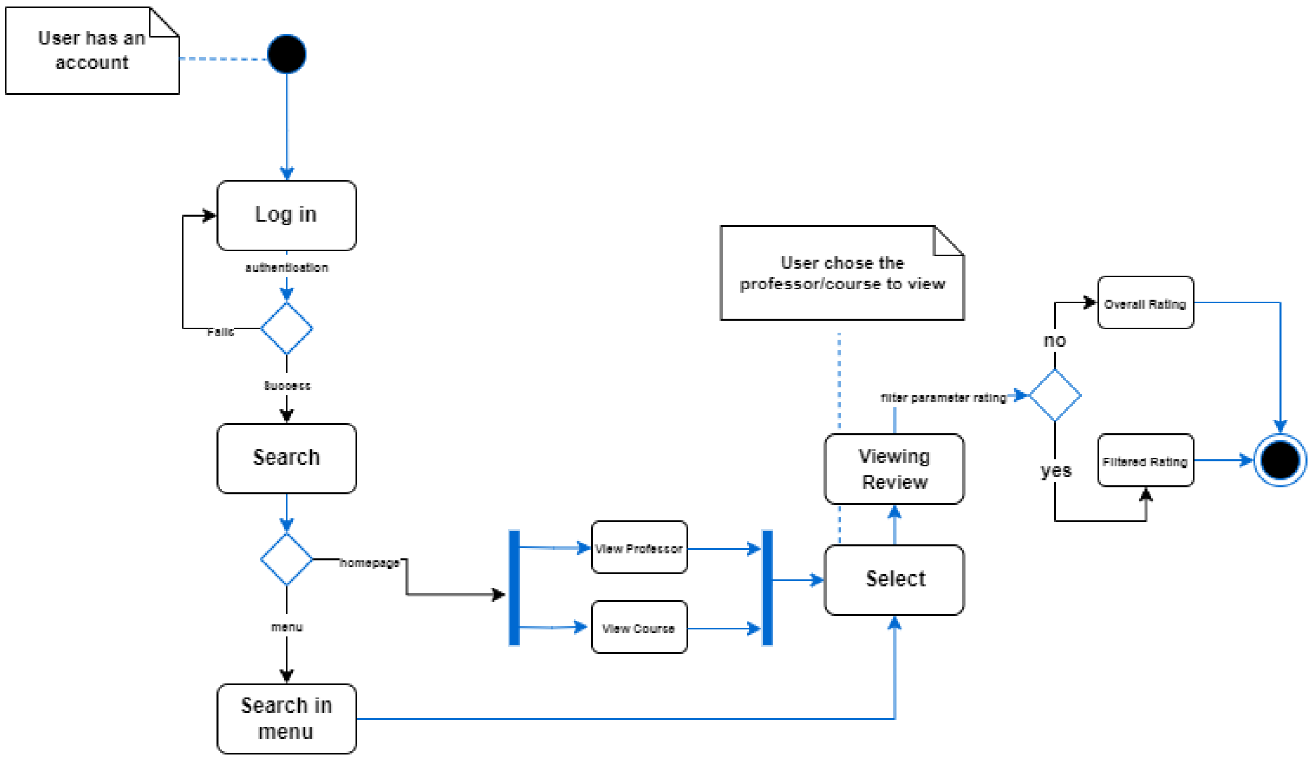
Valuni's main functionalities revolve around two main activities which are writing or reading reviews. Therefore we decided to represent each process through a diagram.

3.3.1 Writing Reviews



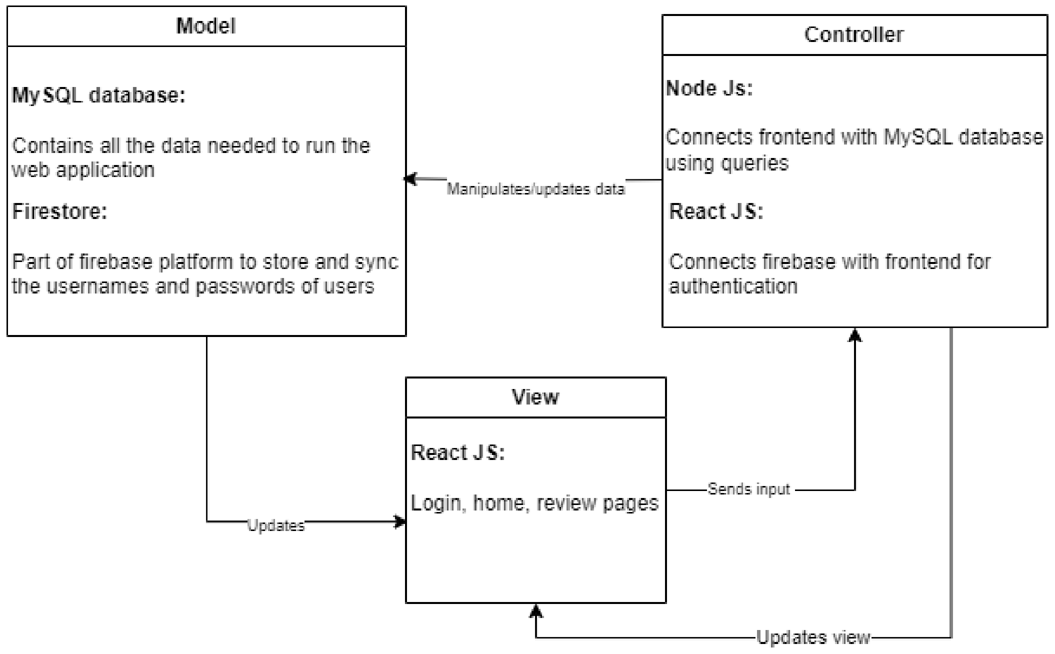
This visual representation illustrates the internal processes involved in achieving a key function, specifically the process of writing reviews. It begins with user authentication upon login, if it succeeds it goes on by offering two search options: directly through the homepage or the search menu. If the user does the search and selection directly from the homepage, they would have professors and courses as two simultaneous choices. After they choose, they can either rate and write a detailed review or rate only.

3.3.2 Reading Reviews



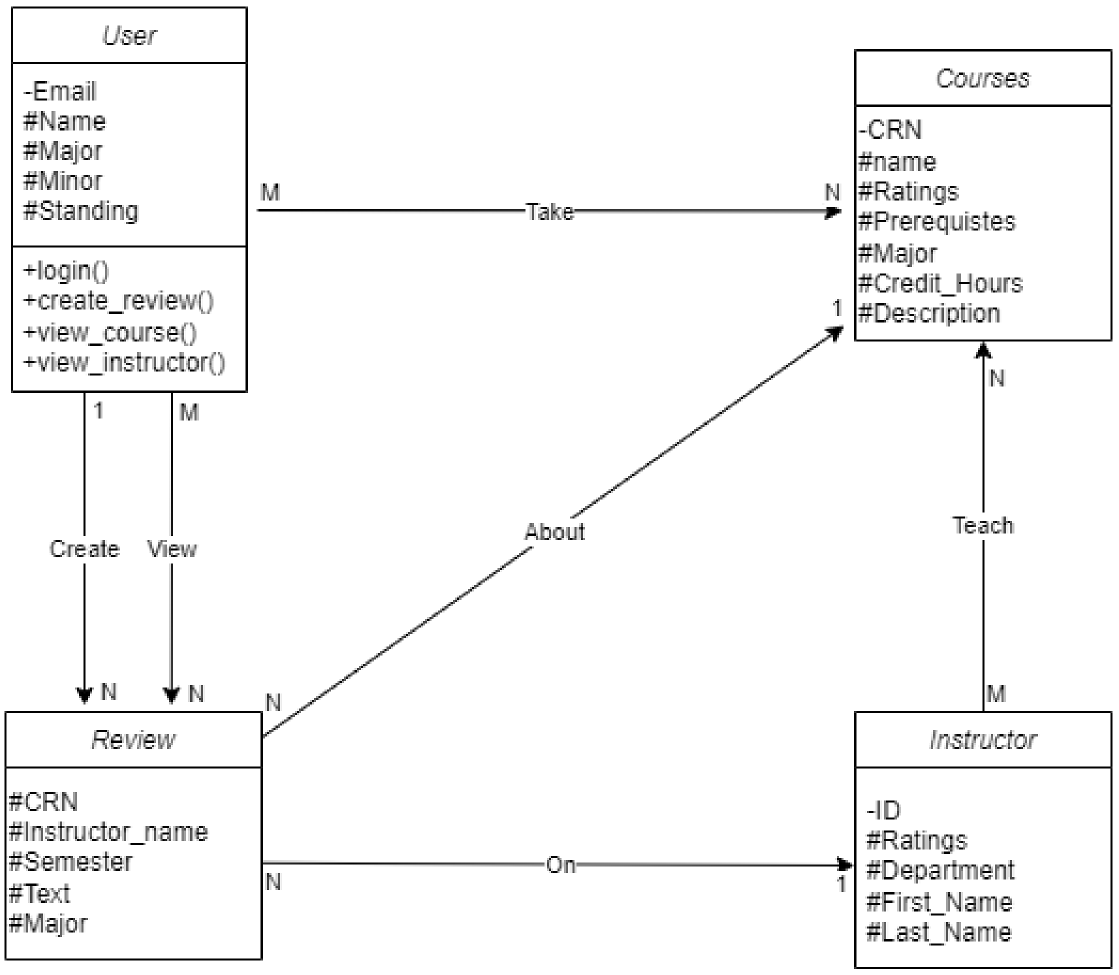
The second key option, reading reviews, is dynamically portrayed through this activity diagram where it begins with the same sequence as the previous diagram. However, after selecting the course or professor, the user views the detailed review, if it exists, followed by either the overall rating or the filtered rating, depending on their choice.

3.4 MVC: Model View Controller Model


Model
MySQL database:
Contains all the data needed to run the web application
Firestore:
Part of firebase platform to store and sync the usernames and passwords of users
Controller
Node Js:
Connects frontend with MySQL database using queries
React JS:
Connects firebase with frontend for authentication
Manipulates/updates data
View
React JS:
Login, home, review pages
Updates
Sends input
Updates view

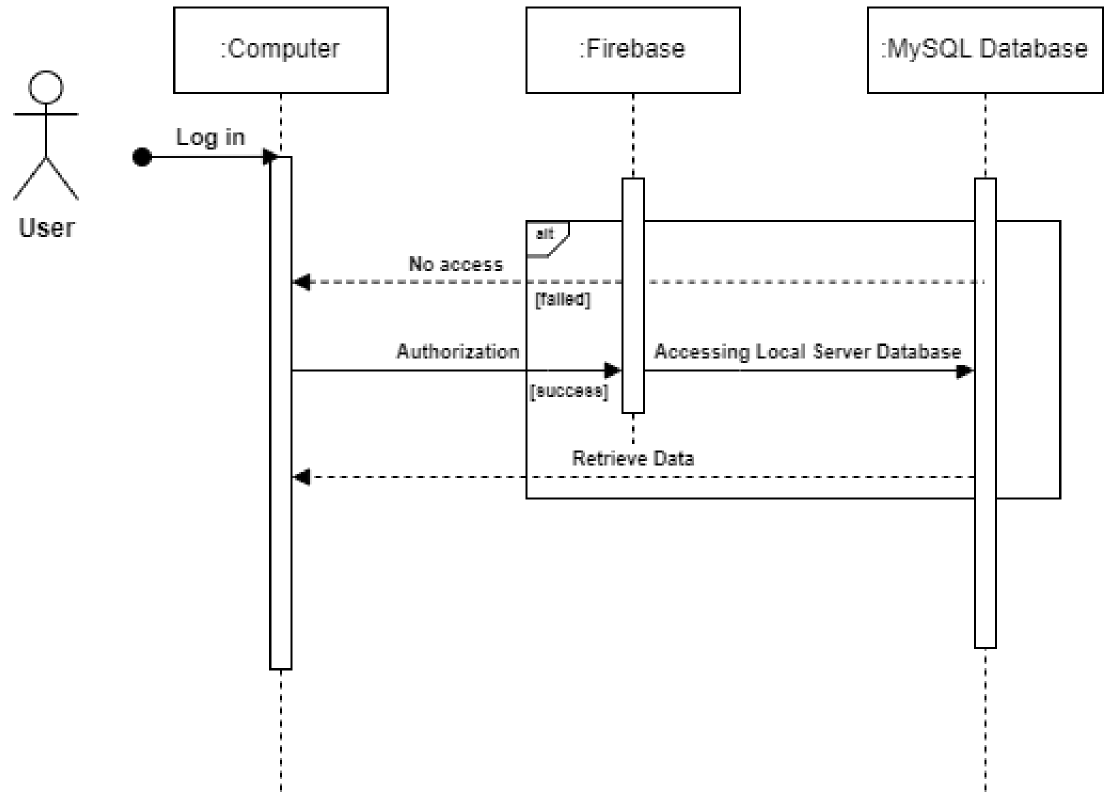
Since Valuni has a number of technologies used, it was crucial to add a design pattern to help in visualizing how our functionalities work and communicate behind the scenes. The model consists of the two main data structures used to store, update, and sync data which are MySQL databases along with Firestore. The content within these structures aid in the updating of the view of the application itself through the different GUIs presented by Valuni. The functionalities within the diverse web pages send input to the controller, processing and filtering it to reach the required output through the data access layer needed to update the view. On the other hand, the input sent may need some manipulation which is why it could be sent to the model to retrieve the needed data and finally it gets updated.

3.5 Class Diagram



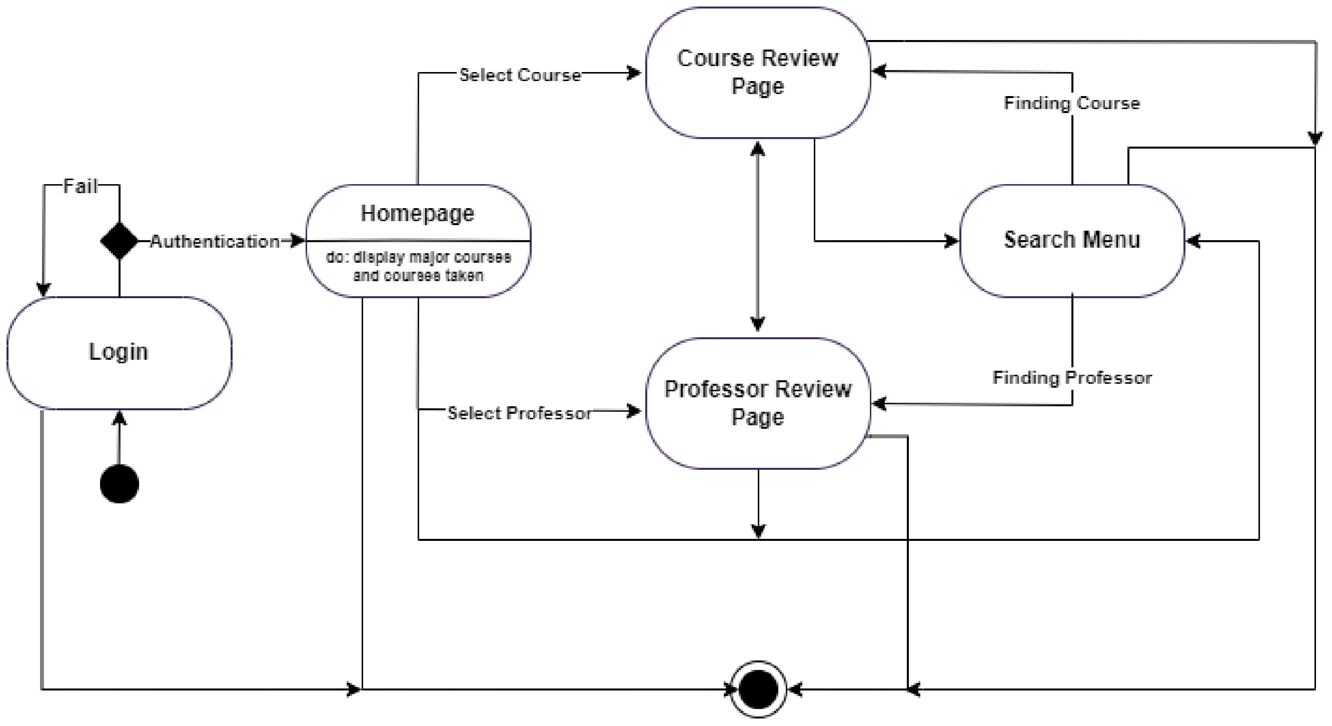
As a starting point, it was decided to implement four main classes: users, courses, review, and instructors. Each of these classes represent one of the tables from the database and are interconnected with each other. Since the user is the only one who can access the web app along with changing the database, its class is the only one containing methods. The rest of the classes contain the attributes which describe them as they do not affect the functionalities of the software.

3.6 Sequence Diagram



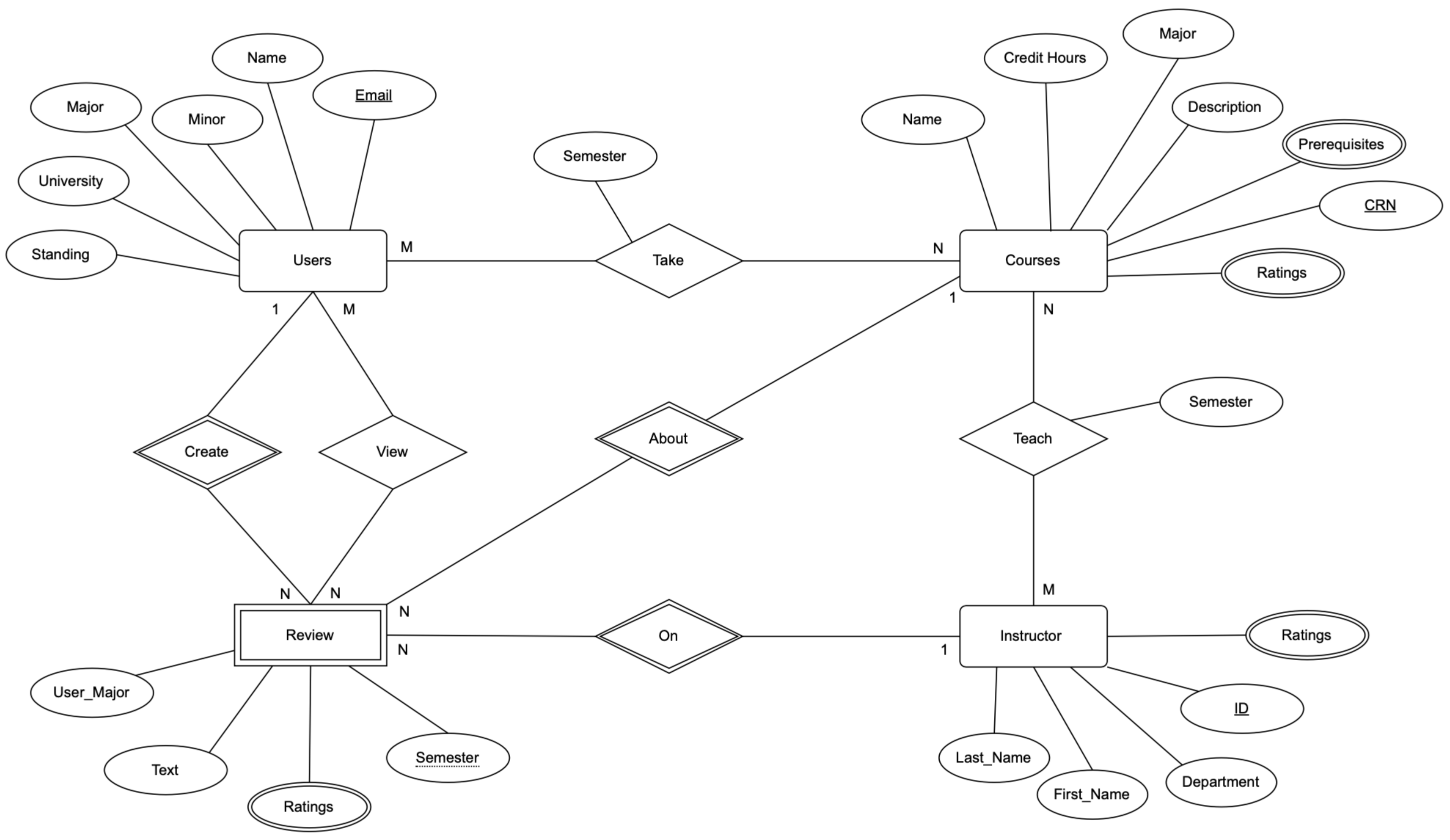
This sequence diagram portrays the connection between the system interfaces used by Valuni. The user starts by logging in and the authentication is done via firebase which is a serverless framework. If the user exits, the needed data will be retrieved from the database which is managed by MySQL workbench and reflected on the GUI of Valuni. On the other hand, if the authentication fails there will be no access to the database containing the user's information.

3.7 State Diagram



Valuni has four web pages, login, home, course review, and professor review page. This diagram is a representation of the dynamic behind the transition between different web pages, where each arrow describes how to transform from one page to another. Since the web application could be closed from any page, all pages are connected to the end point.

3.8 Entity Relationship Diagram



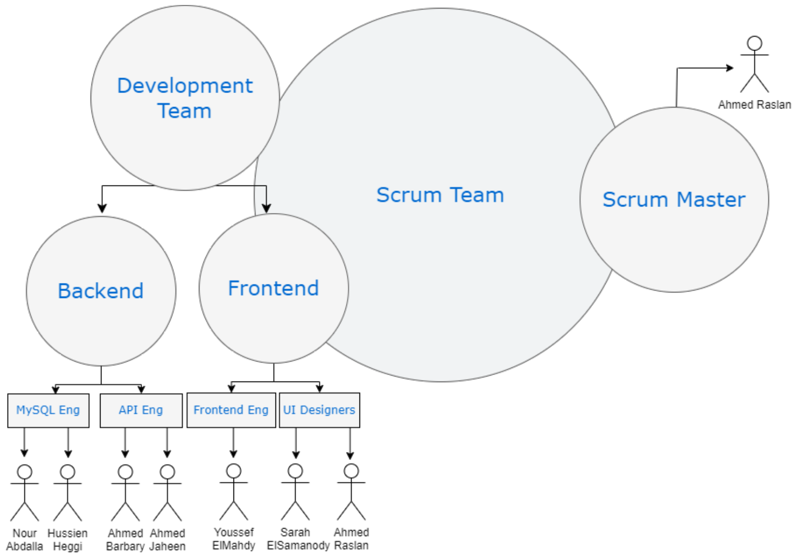
The ERD graphically represents and depicts the structural organization of entities, their attributes, and relationships within a database system. Strong entities in our database are users, courses, and instructors, which possess distinct attributes and play vital roles in the Valuni database schema. On the other hand, review is a weak entity as it lacks an independent key for unique identification, depending on the existence of other entities. This is because a review can not exist without users creating them about either courses or instructors.

4 Second Sprint Retrospective

4.1 Sprint log

Title	URL	Assignees	Status
Masking firebase API keys	https://github.com/Ahmed-Waseem77/valuni/issues/25	Ahmed-Waseem77, elbarbary, mego74	In Progress
Valuni Presentation	https://github.com/Ahmed-Waseem77/valuni/issues/7	nouryasser1, SamanodyJr	Done
Search drop down menu	https://github.com/Ahmed-Waseem77/valuni/issues/37	SamanodyJr, youssef-S-Elmahdy	In Progress
Review box component	https://github.com/Ahmed-Waseem77/valuni/issues/41	youssef-S-Elmahdy	Done
Review page button	https://github.com/Ahmed-Waseem77/valuni/issues/42	youssef-S-Elmahdy	Done
Finishing DB SQL schema	https://github.com/Ahmed-Waseem77/valuni/issues/38	Hussein-Heggi, nouryasser1	Done
Finishing up NodeJS API	https://github.com/Ahmed-Waseem77/valuni/issues/40	Ahmed-Waseem77, elbarbary, mego74	Done
generating some mock up data	https://github.com/Ahmed-Waseem77/valuni/issues/39	Hussein-Heggi, nouryasser1	Done
Review Page	https://github.com/Ahmed-Waseem77/valuni/issues/36	SamanodyJr, youssef-S-Elmahdy	Done
Documentation Diagrams for various methodologies	https://github.com/Ahmed-Waseem77/valuni/issues/27	nouryasser1	Done
Connect the application to the local server database	https://github.com/Ahmed-Waseem77/valuni/issues/4	Hussein-Heggi, nouryasser1	Done
Creating the SQL Database on the local server	https://github.com/Ahmed-Waseem77/valuni/issues/22	Hussein-Heggi, nouryasser1	Done
Database schema design	https://github.com/Ahmed-Waseem77/valuni/issues/8	elbarbary, Hussein-Heggi, mego74, nouryasser1	Done
Home Page/ Landing page	https://github.com/Ahmed-Waseem77/valuni/issues/35	Ahmed-Waseem77, SamanodyJr, youssef-S-Elmahdy	Done
Merging Backed functionalities and resolving resulting conflicts	https://github.com/Ahmed-Waseem77/valuni/pull/30	Ahmed-Waseem77	Done
Dev f update	https://github.com/Ahmed-Waseem77/valuni/pull/43	Ahmed-Waseem77	Done

The primary objective for this sprint is almost finishing the front-end where the functionalities built will be completely operational, yet not linked to the mock up data. In addition, alongside is a fully operational login system, all seamlessly connected to the database. Lastly, the backend would contain almost all the data and queries needed to connect and link the rest of the webpages to it in the next sprint.



4.2 SCRUM team organization

In the development of Valuni, the team was divided into two parts: frontend and backend. When each team started working on their tasks, it was decided that the teams should be further divided to implement the idea of pair programming, working in a more effective and efficient way. Each sub-team had their own tasks and responsibilities which were all merged together at the end of the sprint. The scrum master, Ahmed Raslan, continuously communicated with both ends of the team to ensure that everything was in place.

4.3 Pair Programming

Most of the Tasks were done in pairs to increase cooperation between teams. Even found ourselves having 3 team members working together in some rare cases. This made our team more aware of the code and the hurdles we are facing.

.

4.4 Sprint Review

Understimation of the complexity of linking multiple frameworks together and getting them to work was a key hurdle in our progress. Moving on, we should measure the level of competence of our developers witht the amount of time we have.

For the next sprint our main focus is to develop the MVP to a deployable state.