

# **Design and Testing Document**

## **Project: Peer Review Application by Learnification Technologies**

University Of British Columbia Okanagan

COSC 499 - Summer 2023

Date: June 9th, 2023

### **Group Members:**

Charlotte Zhang ([charlottezhang6@gmail.com](mailto:charlottezhang6@gmail.com))

Shila Rahman ([shilarahman16@gmail.com](mailto:shilarahman16@gmail.com))

Prabhmeet Singh Deol ([arsh.appleid@gmail.com](mailto:arsh.appleid@gmail.com))

Lance Xu ([lance924852785@gmail.com](mailto:lance924852785@gmail.com))

Sehajvir Singh Pannu ([pannusehajvir@gmail.com](mailto:pannusehajvir@gmail.com))

# TABLE OF CONTENTS

<b>1. Description</b>	<b>3</b>
1.1. Introduction	3
1.2. Purpose of the document	3
1.3. Mission statement	3
1.4. System Overview	3
<b>2. User Personas</b>	<b>4</b>
2.1 User Group Summary:	4
1. Student Persona	5
2. Instructor Persona	6
<b>3. Use Case Diagram</b>	<b>7</b>
	7
	7
3.1 Use Cases for Students:	7
<b>4. System Architecture</b>	<b>19</b>
4.1 MVC Framework	20
4.1.1 Controller	20
4.1.2 Service	20
4.2 React Next JS App	20
4.3 Database ER Diagram	20
4.4 Sequence diagram:	21
Sequence of the user interface, and the operations included in retrieving all the assigned evaluations for a student.	21
4.4.1 Login Sequence Diagram :	22
4.4.2 Start and Finish The Evaluation:	23
4.4.3 Remove a student From a group	24

4.4.4 Creation of a Group Evaluation by the Instructor	25
4.4.5 Add Students to a Course:	26
4.4.6 Create Assignment Questionnaire	27
4.4.7 View Group Evaluation Remarks	28
4.5 Data Flow Diagram	29
<b>5.0 User Interface</b>	<b>30</b>
<b>6.0 Technical Specifications</b>	<b>35</b>
6.1 Client Side	35
6.2 Server Side	35
6.3 MySQL Database	35
<b>7.0 Test Plan/QA Plan:</b>	<b>40</b>
7.1 Goal	40
7.2 Scope	40
7.3 Out of Scope	40
7.3 Example test Cases that need to be tested for :	40
7.4 Testing Techniques	41
1. Microservice testing	41
2. Automatic Api Testing	41
3. Database Testing	41
4. Front end Testing	41
5. Manual System Testing	42
6. Acceptance Testing	42
7.5 Branching Strategy:	42
7.6 Security Principles Adhered:	43

# 1. Description

## 1.1. Introduction

A web-based peer review application using React JS, Node Js, Express Js and for an anonymous assessment and feedback system. It promotes collaborative learning, fair evaluation, and simplifies group assignment management for faculty, enhancing the overall educational experience.

## 1.2. Purpose of the document

This document aims to describe the project design for client verification and developer reference. It includes use case models, sequence diagrams, ER diagrams, and other requirement information. The document also provides supporting requirements, constraints, and UI guidelines. It serves as a crucial reference throughout the project, ensuring adherence to specifications and effective communication between stakeholders.

## 1.3. Mission statement

The aim is to create a collaborative learning environment that empowers students to learn from each other and promotes constructive feedback, fair assessment, and collaborative work within the academic community.

## 1.4. System Overview

The Peer Review Application is a web-based application designed to streamline and enhance the peer evaluation process in educational institutions. The objective of the application is to provide instructors and students with a user-friendly platform that enables efficient evaluation and feedback exchange.

The application consists of several key components and functionalities. It features a secure login and registration system, allowing users to create accounts and access the application. There are two types of users in the system: students and instructors, each with their respective roles and permissions.

Upon logging in, students can view their assigned courses, assignments, and group evaluations. They can submit their assignments through the system and evaluate a set number of assignments assigned by their instructors. Additionally, students can assess other students' group contributions and receive evaluations from their peers, ensuring a fair and comprehensive evaluation process. Students have the ability to view feedback from instructors and fellow students anonymously, maintaining confidentiality.

Instructors have comprehensive control over the system. They can create assignments, define evaluation criteria, and set deadlines. They are responsible for assigning students to groups and monitoring the evaluation progress. Instructors can view and evaluate students' assignments, review group evaluations, and provide feedback. They have access to a dashboard that displays relevant details, such as assignment titles, due dates, and evaluation status, facilitating easy monitoring and tracking of student progress.

## 2. User Personas











### 2.1 User Group Summary:

The user group for this project consists of students and instructors involved in the assignment and evaluation process.

Students within this group will use the platform to submit assignments, receive timely feedback, and evaluate their peers' work. They will also have the opportunity to evaluate the contributions of their fellow group members for group assignments. The platform aims to foster collaboration, accountability, and skill development among students through a fair and anonymous peer evaluation process.

Instructors, on the other hand, will utilize the platform to create assignments, manage evaluations, and view feedback. They will have a comprehensive overview of students' progress, allowing them to assess individual and group performance effectively. The project's objective is to create an inclusive and interactive learning environment where students actively engage in the learning process, receive feedback, and collaborate with their peers.

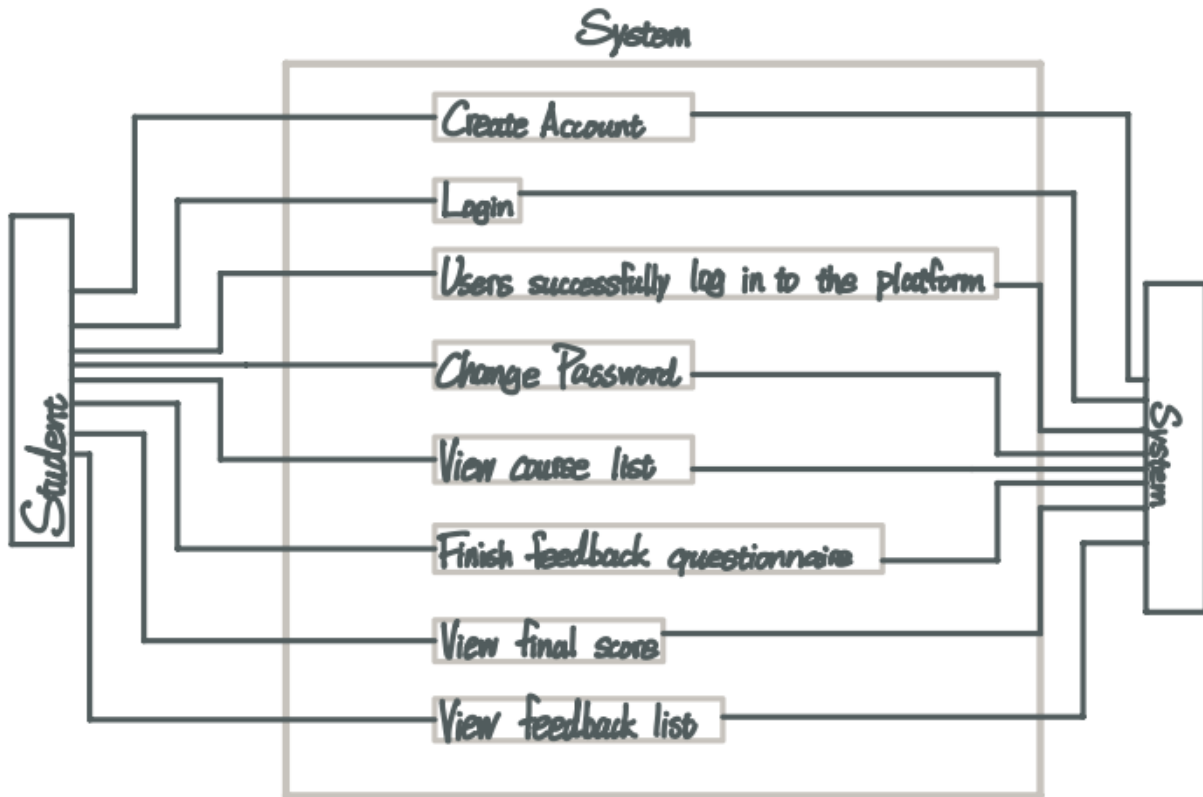
# 1. Student Persona

	<h3>Bio</h3> <p>Irene is a diligent and tech-savvy student pursuing a degree in Computer Science. She is highly motivated and takes her studies seriously. Irene values organization, efficiency, and convenience in her academic life. She is a final year student and has to do a lot of group work for her course. She is familiar with various online platforms and expects a user-friendly interface that simplifies her tasks and enhances her learning experience</p>	<h3>Pains points</h3> <ol style="list-style-type: none"><li>1. Manage to get course work done by deadline.</li><li>2. Worried if providing honest feedback would impact her work negatively</li><li>3. Receiving biased feedback/evaluation</li></ol>	<h3>Needs</h3> <ol style="list-style-type: none"><li>1. Maintaining anonymity while evaluating team members.</li><li>2. Receiving constructive feedback/evaluation on her work with proper reasoning</li></ol>
<h2>Irene Simpson</h2> <div><div><p>Age/Identifying Gender 23/Female</p></div><div><p>Location Kelowna, BC</p></div><div><p>Occupation Student</p></div><div><p>University UBCO</p></div></div>	<h3>Goals</h3> <p>(As a student)</p> <ol style="list-style-type: none"><li>1. Access assignments and relevant information easily.</li><li>2. Collaborate and evaluate team members effectively.</li><li>3. Receive timely feedbacks.</li><li>4. Monitor and track progress in courses.</li></ol>	<h3>Personality</h3> <div><div>Empathetic</div><div>Punctual</div><div>Independent</div><div>Organised</div><div>Practical</div></div>	<h3>Frequently used apps/websites</h3> <div></div>

## 2. Instructor Persona

	<b>Bio</b> <p>Dr. Ken is an experienced and dedicated instructor in the field of Computer Science. He is passionate about providing a quality education to his students and ensuring their success and has over 12 years of teaching experience. Driven by a desire to streamline administrative tasks and enhance the learning experience, he seeks an efficient and user-friendly platform that supports his teaching methodologies and assessment strategies.</p>	<b>Needs</b> <ol style="list-style-type: none"><li>1. Maintain control over course content and assessment criteria</li><li>2. Streamlined feedback management</li></ol>	<b>Pain points</b> <ol style="list-style-type: none"><li>1. Time constraint: Having to grade too many assignments for one or many classes, in his busy schedule</li><li>2. Fairly analyse and evaluate individual contribution in a team</li><li>3. Feedback/evaluation management</li></ol>
<b>Dr. David Ken</b> <div><div><p>Age/Identifying Gender 45/Male</p></div><div><p>Location Kelowna, BC</p></div><div><p>Occupation Assistant Professor</p></div><div><p>University UBCO</p></div></div>	<b>Goals</b> <p>(As an instructor)</p> <ol style="list-style-type: none"><li>1. Design and manage assignments and evaluations effectively.</li><li>2. Ensure accuracy and fairness in grading assignments</li><li>3. Facilitate effective collaboration and feedback among students</li><li>4. Analyze and extract insights from evaluation data for improvement.</li></ol>	<b>Personality</b> <div><div>Empathetic</div><div>Punctual</div><div>Independent</div><div>Organised</div><div>Practical</div></div>	<b>Frequently used apps/websites</b> <div></div>

### 3. Use Case Diagram



#### 3.1 Use Cases for Students:

Use Case 1	Create Account
Primary actor	Students
Description:	Allows a student to create an account
Pre-condition:	Students must have an email.



Post-condition:	Students get an account.
Main scenario:	<ol style="list-style-type: none"> <li>1. User presses create account.</li> <li>2. User enters an email and password.</li> <li>3. System receives requests.</li> </ol>
Extensions:	<ol style="list-style-type: none"> <li>2.1 Email validation <ol style="list-style-type: none"> <li>2.1.1 If the email address that user inputs is invalid, the system should show error messages.</li> </ol> </li> <li>2.2 Password setting limitation. <ol style="list-style-type: none"> <li>2.2.1 Length of password must be between 8 and 14 characters.</li> <li>2.2.2 Password must be a strong password (password must consist a special character, uppercase character and a lower case)</li> </ol> </li> </ol>

Use Case 2	Login
Primary actor	Students
Description:	Allows students to log in their account.
Pre-condition:	Users must have an account.
Post-condition:	Users successfully log in to the platform.
Main scenario:	<ol style="list-style-type: none"> <li>1. User enters an email address and password.</li> <li>2. The system will give the users access to their account</li> </ol>
Extensions:	<ol style="list-style-type: none"> <li>1.1 Email and password are not matched. <ol style="list-style-type: none"> <li>1.1.1 If the password and email entered by the user do not match the data in the database or do not exist, an error is shown.</li> </ol> </li> </ol>

Use Case 3	Change password.
------------	------------------

Primary actor	Students
Description:	Allows users to change their password.
Pre-condition:	The user must have an account.
Post-condition:	User password changed.
Main scenario:	<ol style="list-style-type: none"> <li>1. Users navigate to account settings from the profile icon.</li> <li>2. Users enter their old password and a new password.</li> <li>3. Users submit the password change request with the “Update Password” button.</li> </ol>
Extensions:	<p>2.2 Password setting limitation.</p> <p>2.2.1 The old password must match their current password.</p> <p>2.2.1.1 If the old password does not match the current password show an error that tells the user.</p> <p>2.2.2 Length of password must be between 8 and 14 characters.</p> <p>2.2.3 Password must include at least one special character and upper &amp; lower case.</p>

Use Case 4	View course list
Primary actor	Students
Description:	Users can view all the courses that they registered.
Pre-condition:	Users must be logged in.
Post-condition:	Users can view all the courses that they registered.
Main scenario:	<ol style="list-style-type: none"> <li>1. User go to the course list page.</li> <li>2. User waits for server to retrieve data from database.</li> <li>3. If no data returned, shows are not enrolled in any course.</li> </ol>

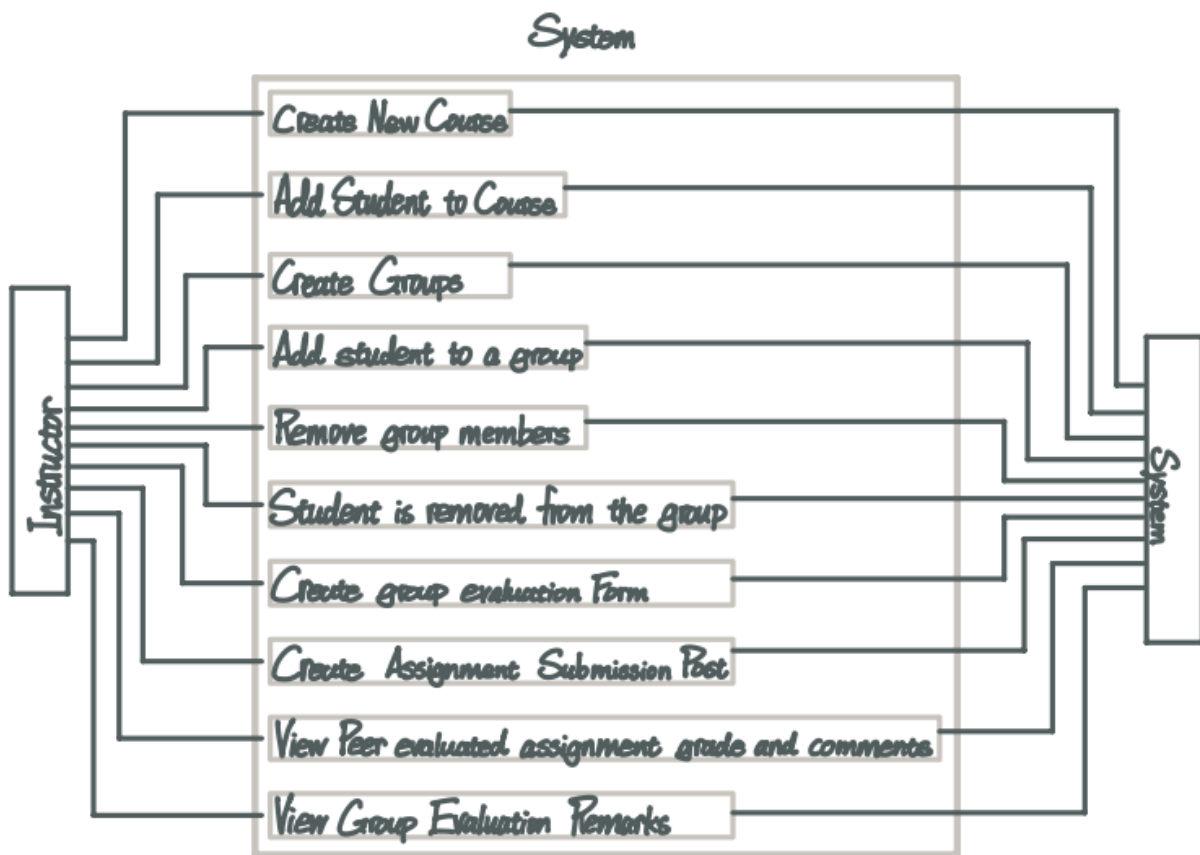
	4.If data is returned, display the course information.
--	--

Use Case 5	Finish feedback questionnaire
Primary actor	Students
Description:	Users can fill the questionnaires based on individual assignment or group assignment.
Pre-condition:	Users must be logged in.
Post-condition:	<ol style="list-style-type: none"> <li>1. Instructor can view the feedback that is given by the user.</li> <li>2. The group member(s) who got evaluated can see the feedback they received.</li> </ol>
Main scenario:	<ol style="list-style-type: none"> <li>1. Users select a group assignment or individual assignment for which they want to give feedback.</li> <li>2. The user selects a person who wants to give feedback.</li> <li>3. Users will provide feedback for the questions from the questionnaire anonymously.</li> </ol>

Use Case 6	View final score
Primary actor	Students
Description:	Students can view the feedback they received on group assignment or their individual assignment.
Pre-condition:	Students login to their accounts and the feedback/grade has been released by the instructor.
Post-condition:	Students get to know their grades for individual assignments and get

	feedback from other group members on their group performance.
Main scenario:	<ol style="list-style-type: none"> <li>1. Students click their grades.</li> <li>2. Students can view the rubric and comments of their grades.</li> </ol>

Use Case 7	View feedback list
Primary actor	Students
Description:	Allows users to view a list of feedback anonymously.
Pre-condition:	<ol style="list-style-type: none"> <li>1.The user must have an account.</li> <li>2.The user received at least one feedback from others.</li> </ol>
Post-condition:	The user sees all the feedback anonymously.
Main scenario:	<ol style="list-style-type: none"> <li>1.User go to the feedback list page.</li> <li>2.User waits for server to retrieve data from database.</li> </ol>
Extensions:	<p>2.2 Feedback Not Found</p> <p>2.2.1 If there is no feedback, the system shows "No Feedback."</p>



Use Case 1	Create new course
Primary actor	Instructors
Description:	Users can create new courses.
Pre-condition:	The user must be logged in.
Post-condition:	The new course will show up on the instructor dashboard.
Main scenario:	1. The user presses the Create Course button on the instructor

	<p>dashboard</p> <p>2. The user provides the system with the Course Name, code, semester, year, term (Start date - End date), number of course assignments, number of group evaluations and external course link (optional)</p>
--	---

Use Case 2	Add student to course
Primary actor	Instructors
Description:	Users can add students to a course.
Pre-condition:	<ol style="list-style-type: none"> <li>1. User has an ongoing connection with the server.</li> <li>2. User has logged in as an instructor.</li> <li>3. User has navigated to the course settings page.</li> <li>4. User has the student list CSV file ready to upload.</li> <li>5. All information in the CSV file has the correct CSV format. (We have to assume that the instructor will provide the correct format according to the client)</li> </ol>
Post-condition:	<ol style="list-style-type: none"> <li>1. If credentials are invalid, Users can restart flow or exit.</li> <li>2. If creation success, Instructor can see a list of students displayed as rows with some information about that student</li> </ol>
Main scenario:	<ol style="list-style-type: none"> <li>1. User submit CSV file.</li> <li>2. User waits for server to verify credentials.</li> <li>3. If credentials are invalid, User is presented with an error.</li> <li>4. If success, back to student list page and the student just added will be display</li> </ol>

Use Case 3	Create groups
Primary actor	Instructors
Description:	Users can create groups.
Pre-condition:	The user must be logged in.
Post-condition:	Students are allocated to different groups.
Main scenario:	<ol style="list-style-type: none"> <li>1. The user selects a course from the course dashboard.</li> <li>2. After selecting the desired course the user can navigate to the course settings page.</li> <li>3. User navigates to the 'Add Groups' page from the course settings page.</li> <li>4. Users submit a CSV file that contains the group names and respective group members in a pre-defined format.</li> </ol>
Extensions:	<ol style="list-style-type: none"> <li>4.1 Credential invalidation <ol style="list-style-type: none"> <li>4.1.1 If credentials are invalid, the User is presented with an error.</li> </ol> </li> <li>4.2 File invalid <ol style="list-style-type: none"> <li>4.2.1 If the submitted file is not a csv file then an error is shown.</li> </ol> </li> </ol>

Use Case 4	Add student to a group
Primary actor	Instructors
Description:	Users can add group members in groups.
Pre-condition:	<ol style="list-style-type: none"> <li>1. The user must be logged in.</li> <li>2. The user must remove the student from their original group before</li> </ol>

	adding them to the new group.
Post-condition:	Students are assigned to the group the instructor wants them to be in.
Main scenario:	<ol style="list-style-type: none"> <li>1. The user selects the course they want to modify groups in.</li> <li>2. The user gets a list of all the groups in the course.</li> <li>3. The user selects the group where they want to add the student.</li> <li>4. The user clicks on the “Add a student to this Group” button.</li> <li>5. The user is prompted to input the student Id of the group they want to add to the group.</li> <li>6. The user clicks submit and then the user gets a confirmation that the student has been added to the new group.</li> </ol>

Use Case 5	Remove group members
Primary actor	Instructors
Description:	Users can delete group members in groups.
Pre-condition:	The user must be logged in.
Post-condition:	Student is removed from the group.
Main scenario:	<ol style="list-style-type: none"> <li>1. The user selects the course where the group is present that needs modification.</li> <li>2. The user gets the list of all the students in the selected group</li> <li>3. The user can click on the delete button at the end of the student’s name in the list to remove them from the group.</li> <li>4. The user will get a popup to confirm their destructive action</li> <li>5. If user clicks on “Remove”, the student will be removed from the group.</li> </ol>



Use Case 6	Create Group Evaluation Form
Primary actor	Instructor
Description:	Users can create an evaluation form for group assignments
Pre-condition:	<ol style="list-style-type: none"> <li>1. The user must be logged in as instructors</li> <li>2. Users must know what questions and options will be included in the evaluation form.</li> </ol>
Post-condition:	Students can view or fill the group evaluation form created by the instructor
Main scenario:	<ol style="list-style-type: none"> <li>1. The user will navigate to the group evaluation list</li> <li>2. The user will select the group assignment/project to create the evaluation form for.</li> <li>3. The user will have four question types to add to the form (e.g., Matrix questions, Multiple Choice Questions, Multiple Answer Questions, and Short Answer Questions)</li> <li>4. The instructor selects the deadline for group assignment/group project evaluations.</li> <li>5. The group assignment evaluation form gets uploaded to the database and on the application.</li> </ol>

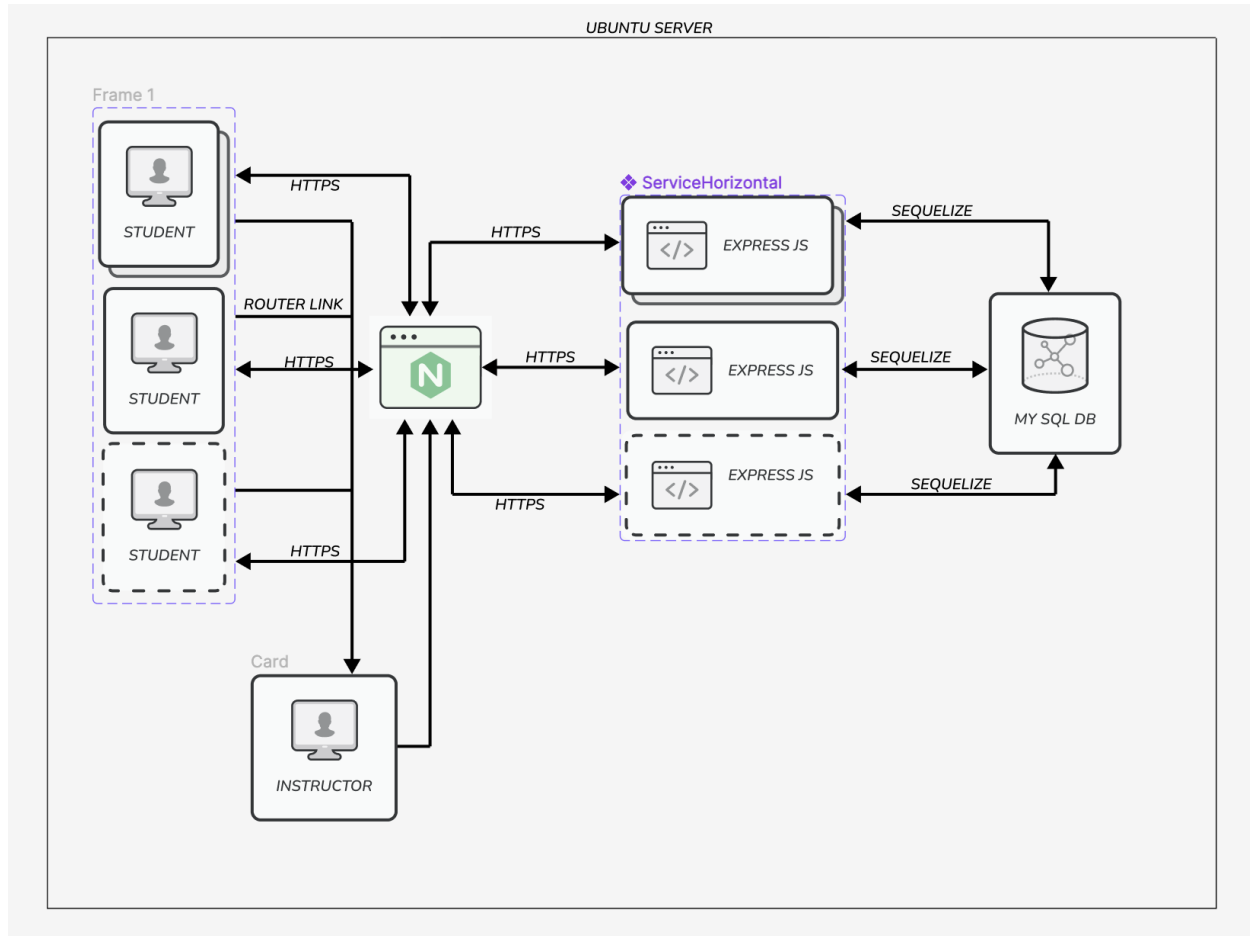
Use Case 7	Create Assignment Submission Post
Primary actor	Instructor
Description:	Users can create a submission page for students to submit their assignments.
Pre-condition:	<ol style="list-style-type: none"> <li>1. The user must be logged in as an instructor.</li> </ol>

	2. User knows the basic information for the assignment they are creating. 3. All information the user has entered is correct in format.
Post-condition:	Students of this course can submit their assignment links to this page.
Main scenario:	1. User have navigated to the course interface. 2. The user will provide the assignment name, the assignment rubric, assignment due date, and the accepted assignment submission format (google document link, dropbox link, etc.) 3. Assignment will be visible to all students taking this course.

Use Case 8	View Peer evaluated assignment grade and comments
Primary actor	Instructors
Description:	Instructors can view the score and comments that are provided by students..
Pre-condition:	Instructors login to their accounts.
Post-condition:	Users can see non-anonymous feedback from students.
Main scenario:	1. Instructors click their grades. 2. Instructor can view how many points each question received and the comments with the student's name.

Use Case 9	View Group Evaluation Remarks
Primary actor	Instructors
Description:	Instructor can see the submitted evaluations by students in a group
Pre-condition:	<ol style="list-style-type: none"> <li>1. User has an ongoing connection with the server.</li> <li>2. User has logged in as an instructor.</li> </ol>
Post-condition:	<ol style="list-style-type: none"> <li>1. If credentials are invalid, Users can restart flow or exit.</li> <li>2. Users can see the evaluation provided by an evaluator student for the corresponding evaluatee from the list.</li> </ol>
Main scenario:	<ol style="list-style-type: none"> <li>1. User has navigated to the course interface.</li> <li>2. User navigates to the group they want all the evaluations from.</li> <li>3. User clicks on the “Group Evaluations” button in the desired group.</li> <li>4. User gets the list of all the group assignments along with the list of each student and their evaluators.</li> <li>5. User selects an evaluation to view from a corresponding group assignment/project for an evaluatee and their evaluator.</li> </ol>

## 4. System Architecture



The student apps are docker containers that will be scaled using kubernetes which will be pulled from docker cloud. The Nginx Load balancer will redirect the incoming routes to the appropriate individual Microservices. This way all the microservices will sit on one end point, and all the microservices together will act as one API. The FrontEnd and the Api will be connecting using HTTPS connection only. Any microservice that needs to query the database , will rely on the Sequelize API to query the Database. This allows us to be able to change the type of database , without having to change any code , but just the database end points in the microservices. All of these apps will be hosted in their docker containers inside a Kubernetes Cluster on an ubuntu Server.

## 4.1 MVC Framework

The Express JS microservices follow a MVC design pattern. Since our App has a frontend application, our microservices will only be returning serialized data instead of encoded HTML pages. Therefore our microservices still rely on the controller and service aspect of the MVC design model.

### 4.1.1 Controller

The controller will be js files that will receive the https requests , and parse the request to variables. Then those variables will be passed as parameters to service functions, which will perform ETL from the MY SQL on the database, and then return that data.

### 4.1.2 Service

These will be JS files , that will perform ETL on the database and also map data to different new objects , that will be returned to js controllers.

## 4.2 React Next JS App

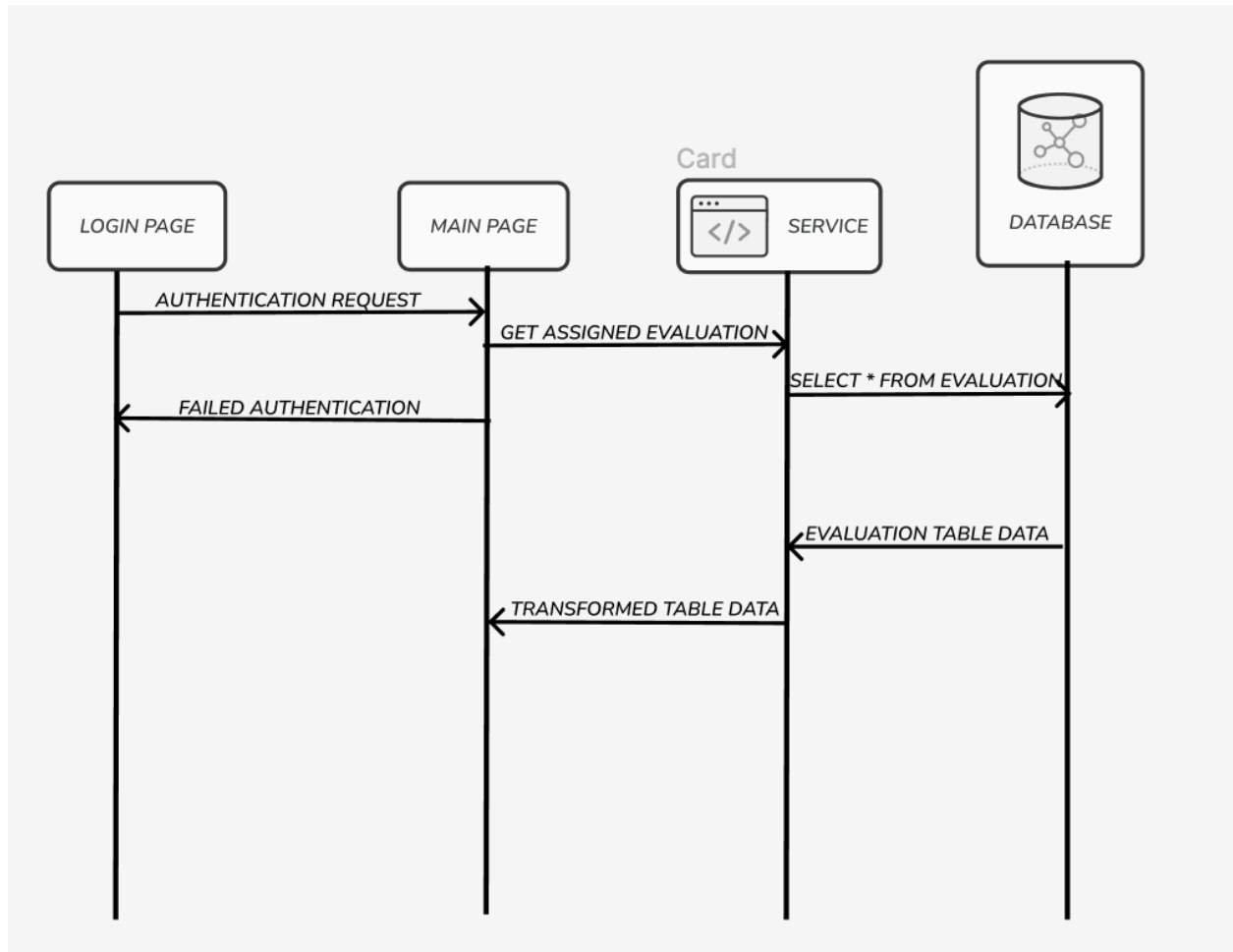
React Running with Next JS gives us faster running react applications , and also some simpler functions and UI components to build the UI.

## 4.3 Database ER Diagram

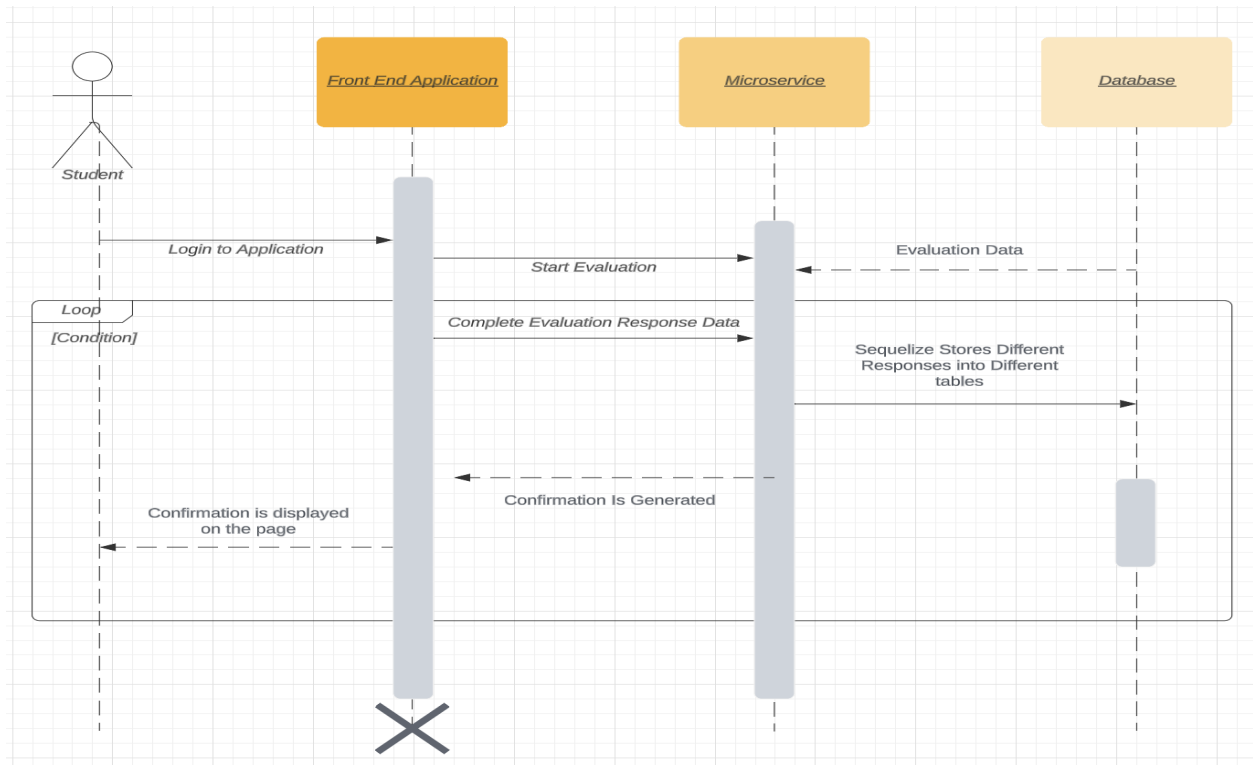
This diagram shows how information will be represented as objects in a relational schema (on the next page):

Sequence of the user interface, and the operations included in retrieving all the assigned evaluations for a student.

#### 4.4.1 Login Sequence Diagram :

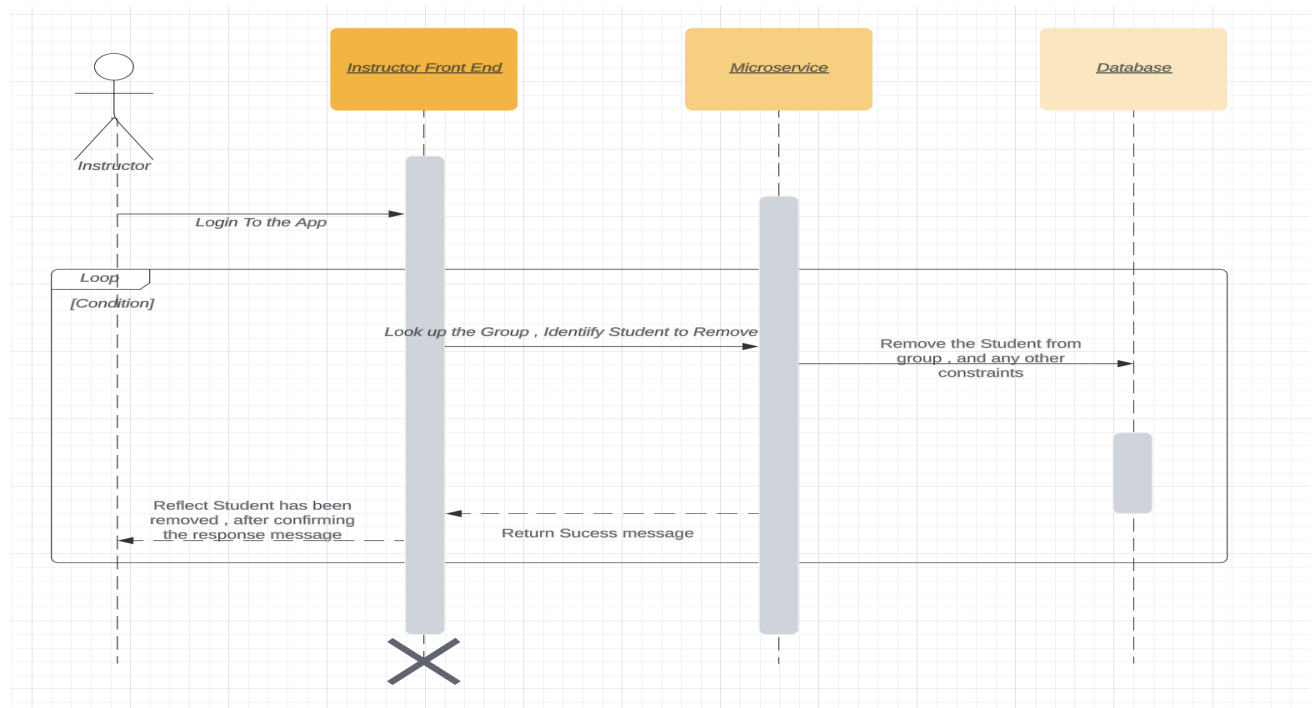


#### 4.4.2 Start and Finish The Evaluation:

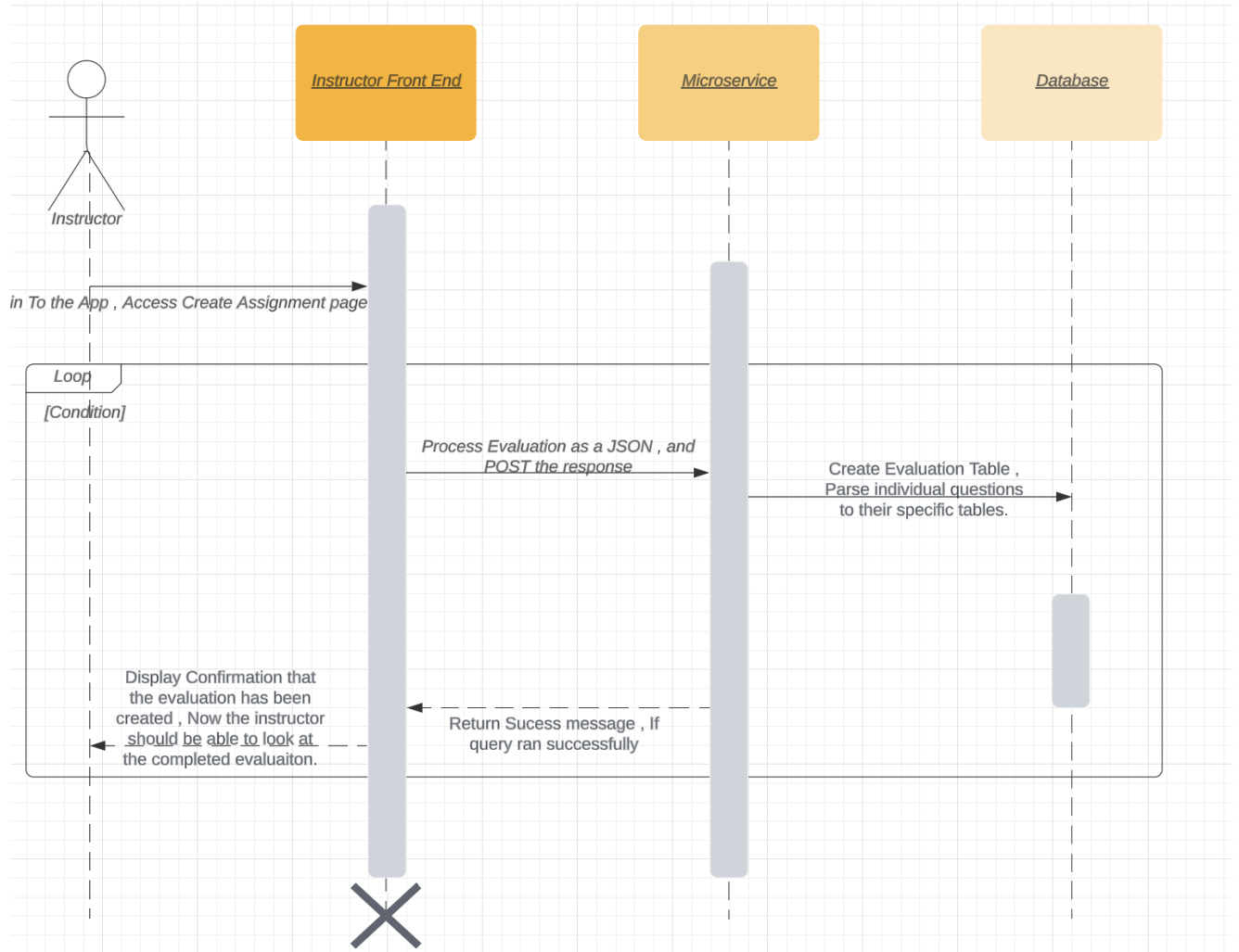




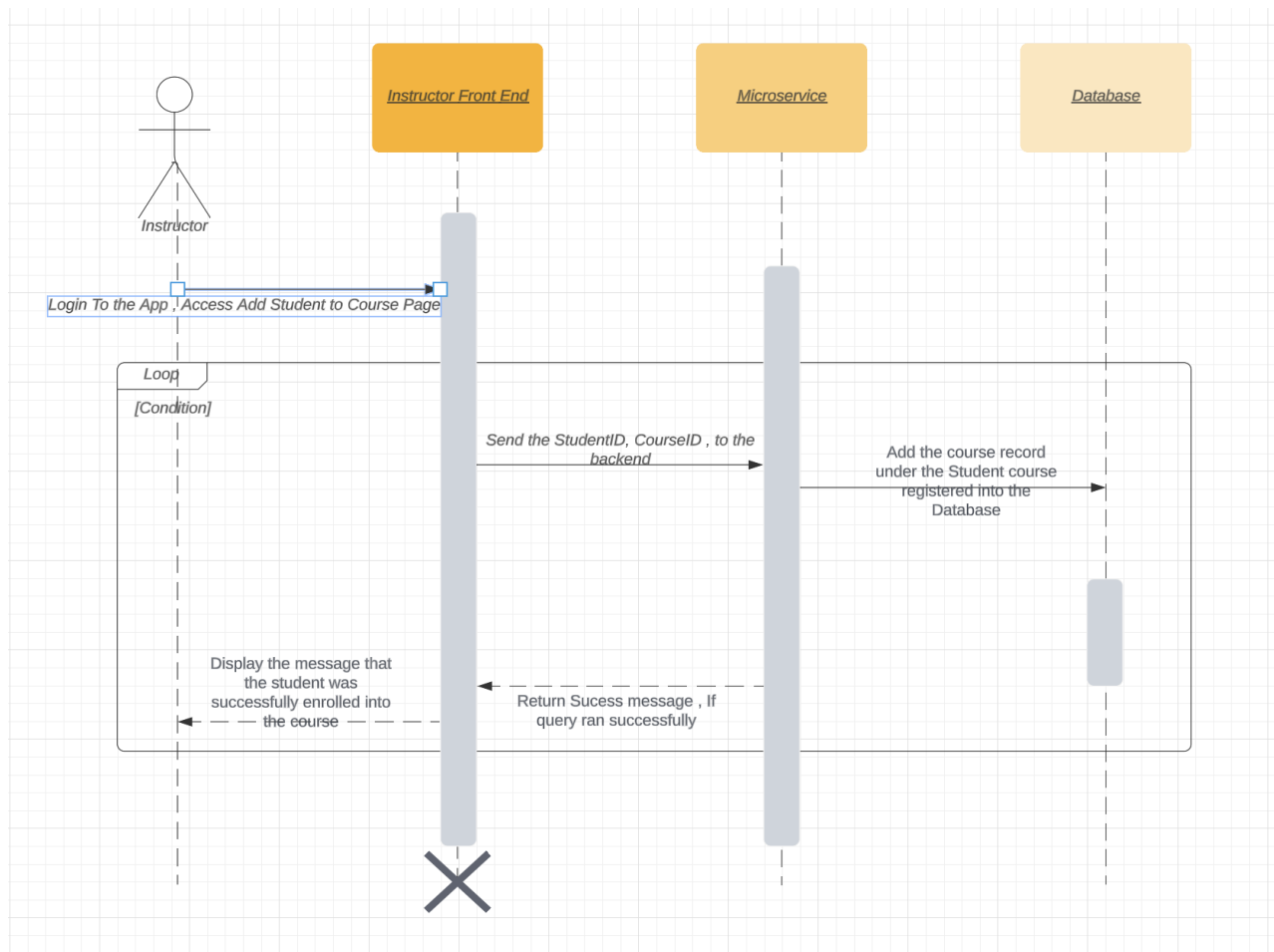
#### 4.4.3 Remove a student From a group



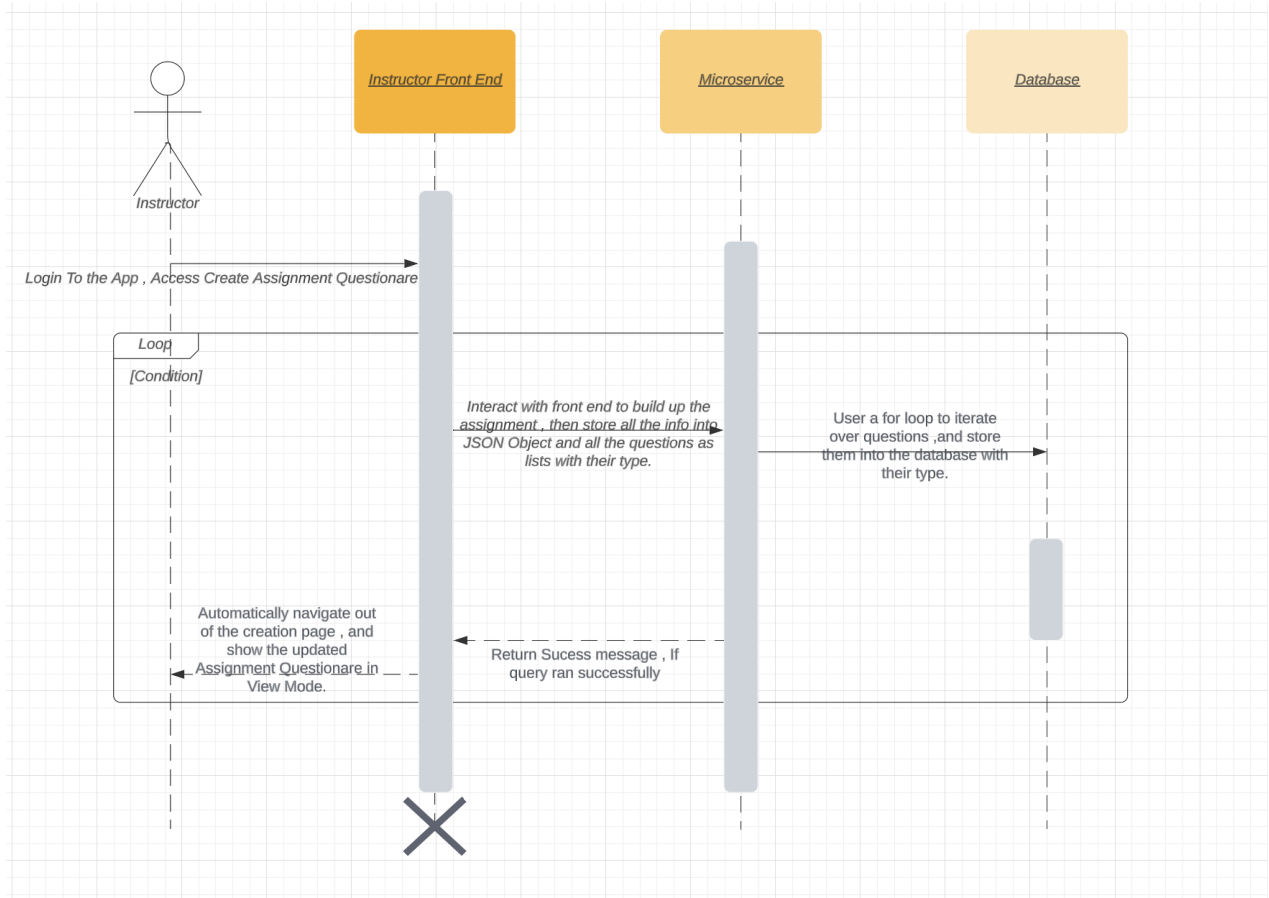
#### 4.4.4 Creation of a Group Evaluation by the Instructor



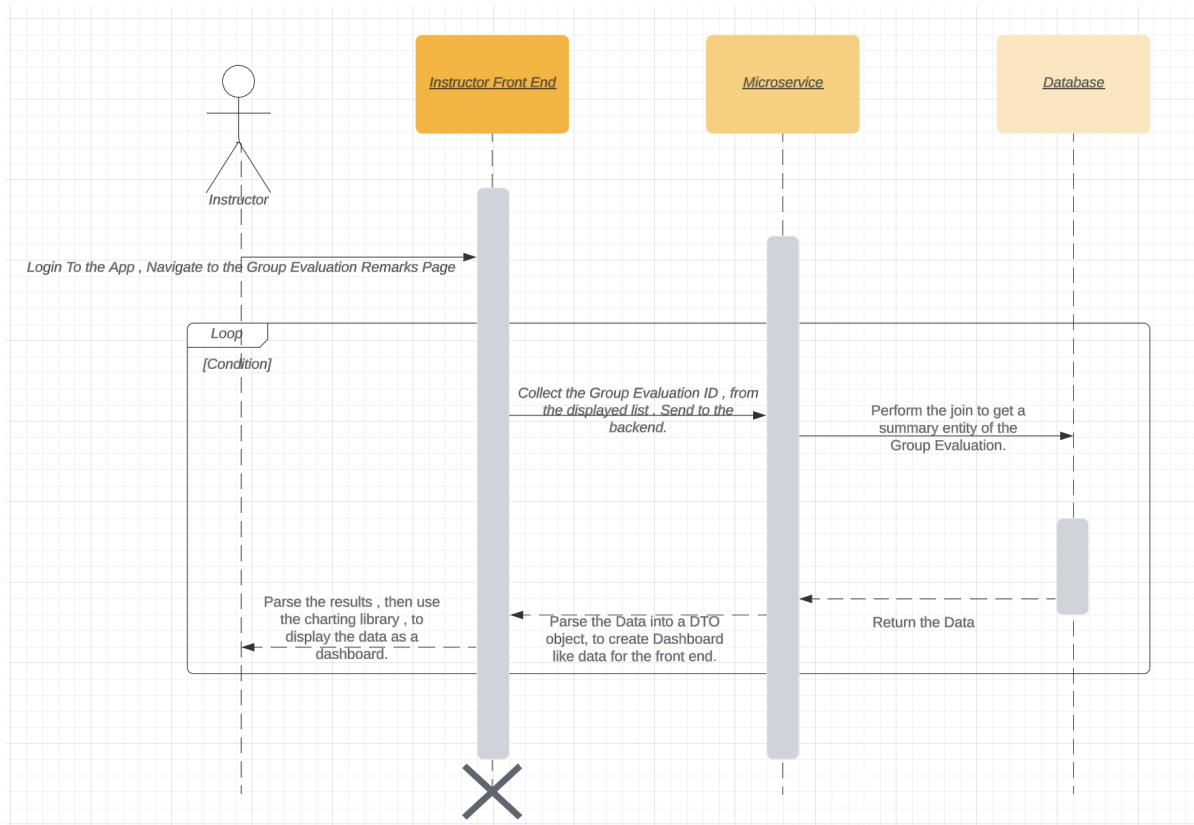
#### 4.4.5 Add Students to a Course:



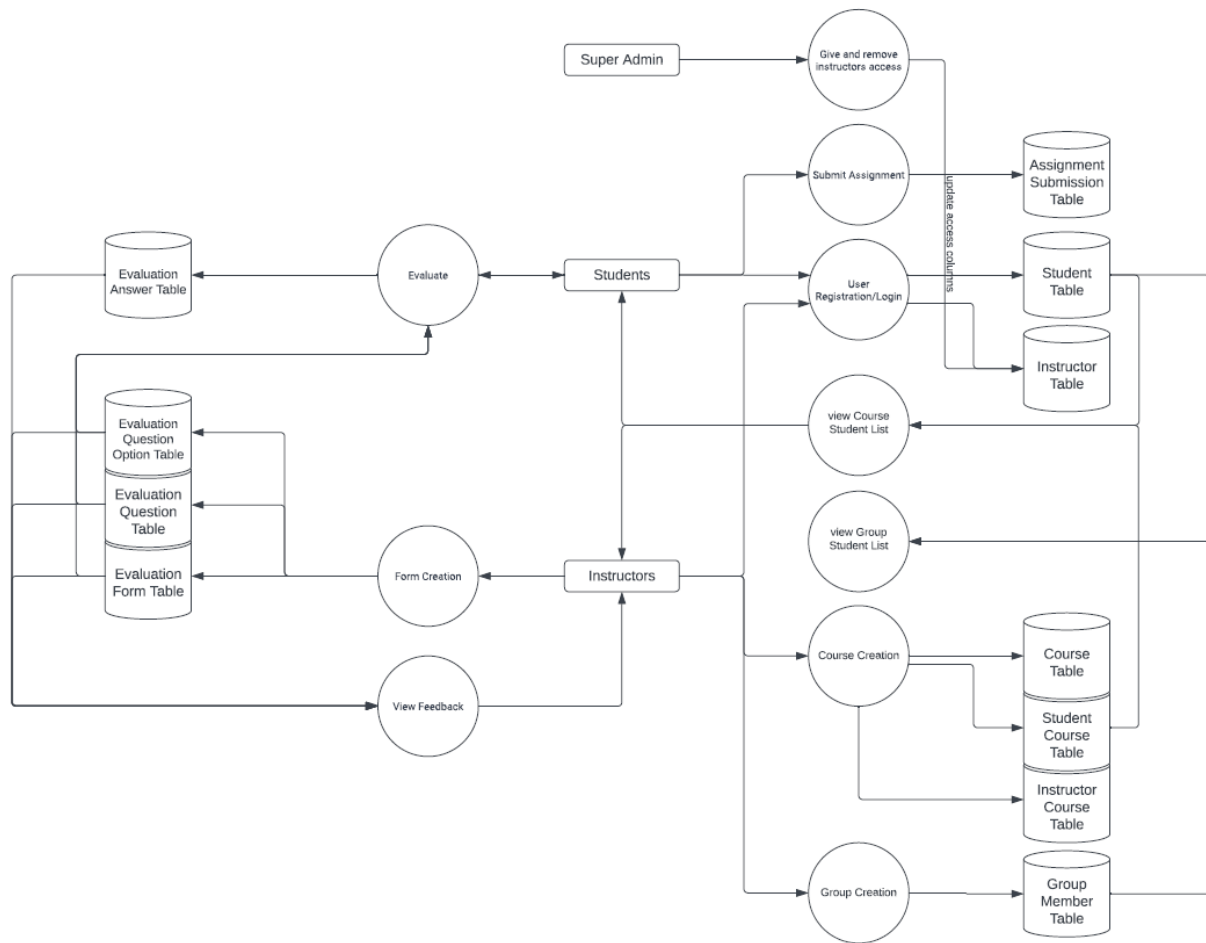
#### 4.4.6 Create Assignment Questionnaire




#### 4.4.7 View Group Evaluation Remarks



## 4.5 Data Flow Diagram



## 5.0 User Interface



### Sign Up

First Name \*

Middle Name

Last Name

Email \*

Student ID \*

Password \*

Confirm Password \*

Already have an account? [Log In](#)

Student Sign Up Page

Peer Review

by Learnification

Dashboard

Logout

Student's Name

Courses

COSC499 201

Summer 2021 May - August

COSC499 201

Summer 2021 May - August

COSC499 201

Summer 2021 May - August

COSC499 201

Summer 2021 May - August

COSC499 201

Summer 2021 May - August

COSC499 201

Summer 2021 May - August

Learnification Technologies @ 202X. All rights reserved.

Student Dashboard



Peer Review

by Learmification

Dashboard

Logout

Name

Introduction to Programming

Student

Group


Evaluation

Student Information

Name	Middle Name	Last Name	Email	Student ID
Person A	Person A	Person A	a@gmail.com	12/12/23
Person B	Person B	Person B	a@gmail.com	12/12/23
Person C	Person C	Person C	a@gmail.com	12/12/23
Person D	Person D	Person D	a@gmail.com	12/12/23
Person E	Person E	Person E	a@gmail.com	12/12/23
Person A	Person A	Person A	a@gmail.com	12/12/23
Person B	Person B	Person B	a@gmail.com	12/12/23
Person C	Person C	Person C	a@gmail.com	12/12/23
Person D	Person D	Person D	a@gmail.com	12/12/23


CompanyName @ 202X. All rights reserved.

**Instructor view: Student List**

 **Peer Review**  
by Learmification

[Dashboard](#)

[Logout](#)

Name 

Introduction to Programming

Course Number: COMP 111

[Course Setting](#)

Term: WINTER 2022

Students

→

Group

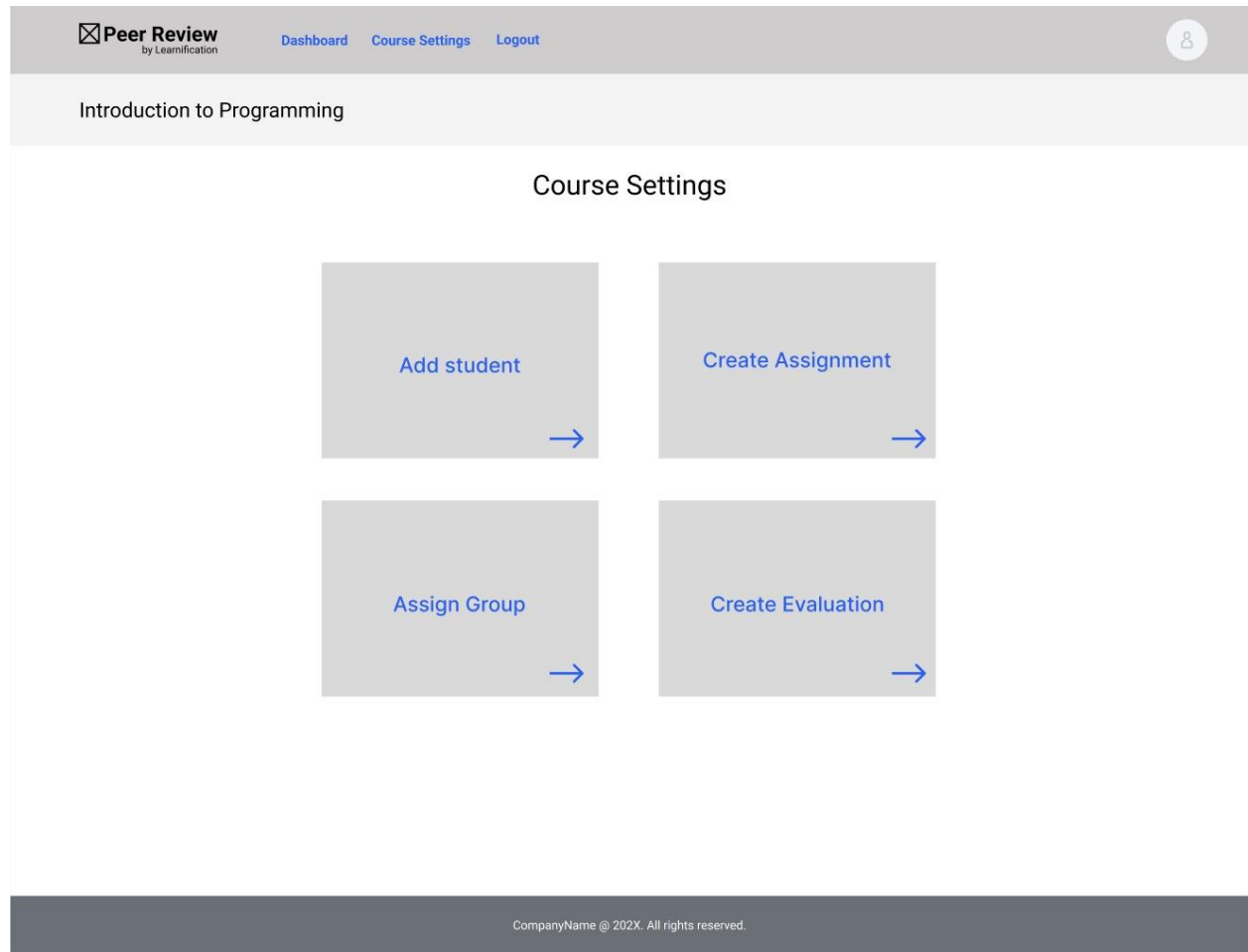
→

Evaluation

→

CompanyName @ 202X. All rights reserved.

**Instructor view: Course Preview**



**Instructor view: Course Settings Page**

## 6.0 Technical Specifications

### 6.1 Client Side

Client side uses React components , to render UI components and display the data being returned from the backend microservices. React components are a mix of

### 6.2 Server Side

The Server side Express JS app makes use of Sequelize api , to query the database. All the database tables will be stored as entities in the Microservices which will then be transformed to DTO classes , if different return type objects than the database objects are expected.

In the microservice app the controller receives the https request , then parses the request to parse extract the https request variables (body parameters, query string variables). Then sends those variables as parameters to the service methods which then make use of the Sequelize api to query the database, and return the data. The microservice app can also perform Data transformations if required .

### 6.3 MySQL Database

#### **STUDENT**

- STUDENT\_ID: INT [PK] - The unique ID of the student
- FIRST\_NAME: VARCHAR(50) - The first name of the student
- MIDDLE\_NAME: VARCHAR(50) - The middle name of the student
- LAST\_NAME: VARCHAR(50) - The last name of the student
- EMAIL: VARCHAR(50) - The email address of the student (also used as login)

- MD5\_HASHED\_PASSWORD: VARCHAR(255) - The hashed password of the student

## **SUPER\_ADMIN**

- SUPER\_ADMIN\_ID: INT [PK] - The unique ID of the super admin
- EMAIL: VARCHAR(100) - The email address of the super admin
- MD5\_HASHED\_PASSWORD: VARCHAR(255) - The hashed password of the super admin

## **INSTRUCTOR**

- INSTRUCTOR\_ID: INT [PK, Auto Increment] - The unique ID of the instructor
- FIRST\_NAME: VARCHAR(50) - The first name of the instructor
- MIDDLE\_NAME: VARCHAR(50) - The middle name of the instructor
- LAST\_NAME: VARCHAR(50) - The last name of the instructor
- EMAIL: VARCHAR(50) - The email address of the instructor
- MD5\_HASHED\_PASSWORD: VARCHAR(255) - The hashed password of the instructor

## **COURSE**

- COURSE\_ID: INT [PK, Auto Increment] - The unique ID of the course
- INSTRUCTOR\_ID: INT - The ID of the instructor associated with the course
- COURSE\_NAME: VARCHAR(50) - The name of the course
- COURSE\_CODE: VARCHAR(50) - The code/identifier of the course
- COURSE\_SEMESTER: VARCHAR(50) - The semester of the course
- COURSE\_YEAR: VARCHAR(50) - The year of the course
- COURSE\_TERM: VARCHAR(50) - The term/period of the course
- COURSE\_ASSIGNMENTS: INT - The number of assignments in the course
- COURSE\_GROUP\_EVALUATIONS: INT - The number of group evaluations in the course
- COURSE\_VISIBILITY: TINYINT(1) - The visibility status of the course (0 - not visible, 1 - visible)

- EXTERNAL\_COURSE\_LINK: VARCHAR(255) - The link to an external course (optional)

### **INSTRUCTOR\_COURSE**

- INSTRUCTOR\_ID: INT - The ID of the instructor
- COURSE\_ID: INT - The ID of the course
- PRIMARY KEY (INSTRUCTOR\_ID, COURSE\_ID)
- FOREIGN KEY (INSTRUCTOR\_ID) REFERENCES INSTRUCTOR(INSTRUCTOR\_ID)
- FOREIGN KEY (COURSE\_ID) REFERENCES COURSE(COURSE\_ID)

### **STUDENT\_COURSE**

- STUDENT\_ID: INT - The ID of the student
- COURSE\_ID: INT - The ID of the course
- FOREIGN KEY (STUDENT\_ID) REFERENCES STUDENT(STUDENT\_ID)
- FOREIGN KEY (COURSE\_ID) REFERENCES COURSE(COURSE\_ID)
- PRIMARY KEY (STUDENT\_ID, COURSE\_ID)

### **COURSE\_GROUP\_EVALUATION**

- GROUP\_ID: INT [PK, Auto Increment] - The unique ID of the group evaluation
- GROUP\_NAME: VARCHAR(255) - The name of the group
- COURSE\_ID: INT - The ID of the course
- FOREIGN KEY (COURSE\_ID) REFERENCES COURSE(COURSE\_ID)

### **GROUP\_MEMBERS\_TABLE**

- STUDENT\_ID: INT - The ID of the student
- GROUP\_ID: INT - The ID of the group evaluation
- FOREIGN KEY (GROUP\_ID) REFERENCES COURSE\_GROUP\_EVALUATION(GROUP\_ID)

- FOREIGN KEY (STUDENT\_ID) REFERENCES STUDENT(STUDENT\_ID)
- PRIMARY KEY (STUDENT\_ID, GROUP\_ID)

## **EVALUATION\_FORM**

- FORM\_ID: INT [PK, Auto Increment] - The unique ID of the evaluation form
- FORM\_NAME: VARCHAR(255) - The name of the evaluation form
- COURSE\_ID: INT - The ID of the course
- DEADLINE: DATETIME - The deadline for submitting the evaluation form
- FOREIGN KEY (COURSE\_ID) REFERENCES COURSE(COURSE\_ID)

## **EVALUATION\_QUESTION**

- QUESTION\_ID: INT [PK] - The unique ID of the evaluation question
- FORM\_ID: INT - The ID of the evaluation form
- QUESTION\_TEXT: VARCHAR(1000) - The text of the evaluation question
- QUESTION\_TYPE: VARCHAR(255) - The type of the evaluation question
- QUESTION\_WEIGHT: FLOAT - The weight/importance of the evaluation question
- QUESTION\_SECTION: VARCHAR(255) - The section/category of the evaluation question
- FOREIGN KEY (FORM\_ID) REFERENCES EVALUATION\_FORM(FORM\_ID)

## **EVALUATION\_QUESTION\_OPTION**

- OPTION\_ID: INT [PK] - The unique ID of the evaluation question option
- QUESTION\_ID: INT - The ID of the evaluation question
- OPTION\_TEXT: VARCHAR(255) - The text of the evaluation question option
- FOREIGN KEY (QUESTION\_ID) REFERENCES EVALUATION\_QUESTION(QUESTION\_ID)

## **EVALUATION\_ANSWER**

- EVALUATION\_ID: INT [PK] - The unique ID of the evaluation answer

- FORM\_ID: INT - The ID of the evaluation form
- EVALUATOR\_ID: INT - The ID of the student who is evaluating
- EVALUATEE\_ID: INT - The ID of the student being evaluated
- QUESTION\_ID: INT - The ID of the evaluation question
- ANSWER: VARCHAR(5000) - The answer to the evaluation question
- FOREIGN KEY (FORM\_ID) REFERENCES EVALUATION\_FORM(FORM\_ID)
- FOREIGN KEY (EVALUATOR\_ID) REFERENCES STUDENT(STUDENT\_ID)
- FOREIGN KEY (EVALUATEE\_ID) REFERENCES STUDENT(STUDENT\_ID)
- FOREIGN KEY (QUESTION\_ID) REFERENCES EVALUATION\_QUESTION(QUESTION\_ID)

## **COURSE\_ASSIGNMENT**

- ASSIGNMENT\_ID: INT [PK, Auto Increment] - The unique ID of the assignment
- COURSE\_ID: INT - The ID of the course
- ASSIGNMENT\_NAME: VARCHAR(255) - The name of the assignment
- DEADLINE: DATETIME - The deadline for submitting the assignment
- FORM\_ID: INT - The ID of the evaluation form associated with the assignment
- FOREIGN KEY (COURSE\_ID) REFERENCES COURSE(COURSE\_ID)
- FOREIGN KEY (FORM\_ID) REFERENCES EVALUATION\_FORM(FORM\_ID)

## **ASSIGNMENT\_SUBMISSION**

- SUBMISSION\_ID: INT [PK, Auto Increment] - The unique ID of the assignment submission
- ASSIGNMENT\_ID: INT - The ID of the assignment
- STUDENT\_ID: INT - The ID of the student submitting the assignment
- SUBMISSION\_LINK: VARCHAR(1000) - The link to the submitted assignment
- FOREIGN KEY (ASSIGNMENT\_ID) REFERENCES COURSE\_ASSIGNMENT(ASSIGNMENT\_ID)
- FOREIGN KEY (STUDENT\_ID) REFERENCES STUDENT(STUDENT\_ID)



## 7.0 Test Plan/QA Plan:

### 7.1 Goal

The System needs to be tested at multiple points of development , to ensure that the components previously built are still working accordingly with integration of new components . Moreover the functions being built are also returning expected values that we expect them to deliver. Moreover Manual testing on the whole system needs to be done, to ensure that all the components of the system are working expectedly in a synchronous manner. This will only be done once , before the delivery of our MVP product, to ensure the client has access to a working demo . The most important testing goal is to make sure the product being delivered is in alignment with the Clients Expectations, and the requirements of the system.

### 7.2 Scope

The scope of testing will include all the backend apis and frontend interfaces of the application.

### 7.3 Out of Scope

Currently there is nothing that has been identified as out of scope to test this application. Since this application relies heavily on CRUD operations of the stored information, and basic mathematical data transformation . Current testing techstack and techniques, should be able to test all the functional requirements of the system.

### 7.3 Example test Cases that need to be tested for :

1. Student is able to Perform a peer evaluation successfully , and all his answers are recorded correctly in the database. While also making sure when the student comes back to review his evaluation, the exact same results are shown.

2. Instructor is able to create an evaluation, and all the data from the evaluation is stored in the correct manner in the database. Moreover when the evaluation is presented to students , it has the exact same data that the instructor had input into the system.
3. Dashboard Summary view of an evaluation is displaying all the values correctly , based on the right functional requirements to display the summary.

## 7.4 Testing Techniques

### 1. Microservice testing

All the service functions and any parsing functions will have unit tests that will test the expected outputs. If a service function relies on multiple functions to retrieve the final results , integration tests for those functions will be written to test the methods. The library used to run these tests is the Vitest library.

- a. ViTest library - Allows us to pass parameters to functions, and then compare the returned results with expected results

### 2. Automatic Api Testing

Automatic api testing will be performed by writing out test cases in postman , and then storing them in the git repo. Once every Controller in an Api is done, the automatic tests for the controller will be written.

### 3. Database Testing

User stories and scenarios will be used , to ensure that all the required data that needs to be stored , can be stored in the tables.

- a. The database will be tested to ensure that all the data from the Use cases that need to be saved into the database.

### 4. Front end Testing

Manual testing will be performed for the UI components , while for reusable react components the react testing library will be used to store automated tests in the repo.

- a. React Testing library - Allows us to pass input variables into our react HTML components, then Verify if the rendered components values are similar to the expected value.

## 5. Manual System Testing

Once we have an MVP ready , manual testing will be performed on the working product to test to see. All the values , and functions are working in the manner they are supposed to be working. Meaning the CRUD operations performed on the front end , are being accurately carried out on the database.

- We will mainly test all the Use Cases that we promised for the MVP, as a demo user we are able to perform those.
- **Black Box Testing** - One of the developers will be asking our friends who may or may not have experience in software development .

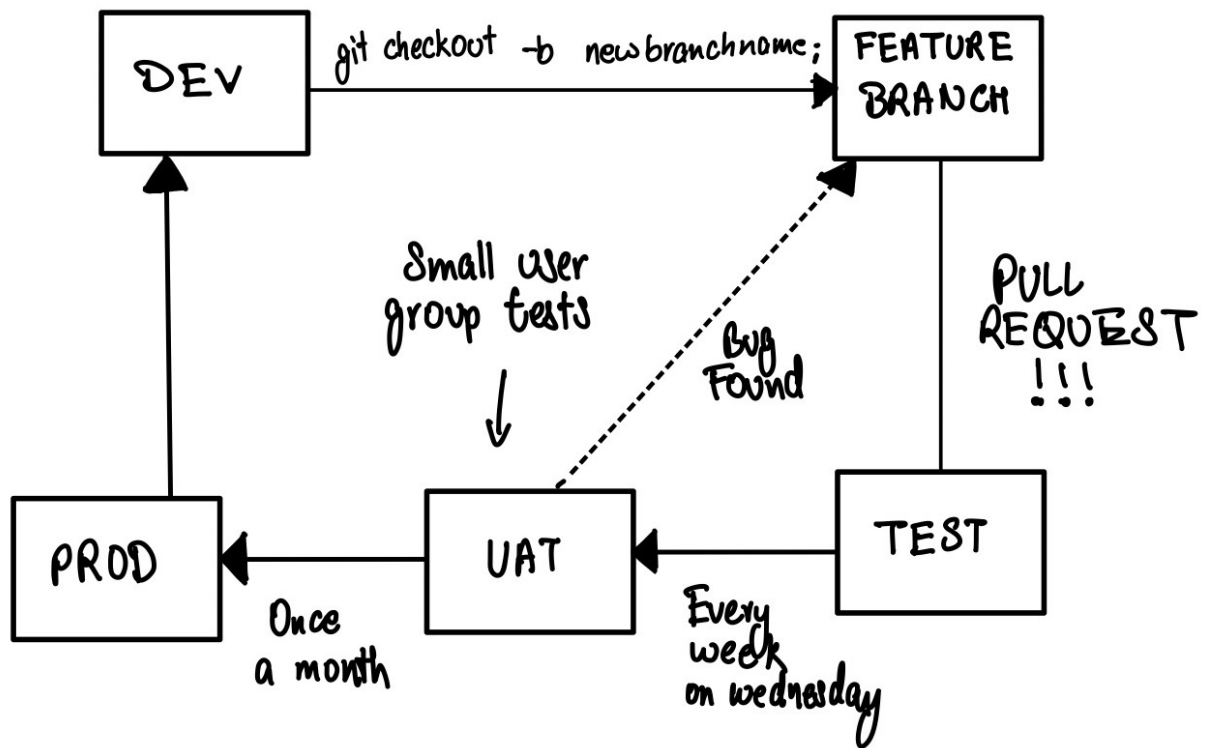
## 6. Acceptance Testing

After the delivery of the MVP , the Client will be asked to test the functionality of the MVP product and to see that the functionality delivered is what the client was expecting.

## 7.5 Branching Strategy:

The following branching strategy will be used for development . The dev branch is only expected to receive code from UAT , and the production branch. To ensure DEV branch always has working code, for anyone to branch out of.

All the new feature branches are only allowed to come out of the DEV or UAT (only if a bug is found) branch only. THE CI pipelines will handle the merging of the UAT branch with PROD , at set times.



## 7.6 Security Principles Adhered:

1. Having separate front-end and back-end applications ensures that users cannot gain access to the business logic.
2. All communication between the front-end and back-end applications will be done exclusively over HTTPS. This means that every microservice controller will need to have its own signed SSL certificate to allow this communication.
3. We will be using the Sequelize API, which employs coded SQL logic to query the database. This approach effectively eliminates the risk of SQL injection.