

Final Report

Project: Peer Review Application by Learnification Technologies

University Of British Columbia

Okanagan COSC 499 - Summer 2023

Date: August 15th, 2023

Group Members:

Charlotte Zhang (charlottezhang6@gmail.com)

Shila Rahman (shilarahman16@gmail.com)

Prabhmeet Singh Deol (pdeol01@student.ubc.ca)

Lance Xu (lance924852785@gmail.com)

Sehajvir Singh Pannu (pannusehajvir@gmail.com)

TABLE OF CONTENTS

Project Charter, Scope and Requirements	7
1. Project Purpose, Problem Identification and Solution	7
Problem Identification:	7
Proposed Solution:	8
2. Project Objective and Related Success Criteria	8
3. Success Metrics	9
4. High-level Requirements	9
5. Assumptions And Constraints	9
Assumptions:	9
Constraints:	10
6. High-Level Project Description and Boundaries	10
7. High Level Risks	11
8. Summary Milestone Schedule	11
9. Stakeholders List	12
10. Requirements	12
Functional Requirements:	12
a. Functional Requirements for Students:	12
1. Account Management:	12
2. Course Management	13
3. Assignment and Assignment Assessment Management:	13
4. Group Evaluation:	14
b. Functional Requirements for Instructors:	15
1. Instructor Account:	15
2. Course Management:	15
3. Evaluation Team/Groups:	16
4. Assignment Management	16
5. Evaluation Survey Management:	17
c. Functional Requirements for Super Admin	18
1. Account Management:	18
access for instructors within the system.	19
d. System Functional Requirements	19
1. Group Evaluations and Assignment Assessments	19
Non-Functional Requirements:	19
a. Security	19
b. Compatibility	19
c. Localization	19
d. Usability	19
e. Accessibility	20

Technical Requirements:	20
User Requirements:	20
a. User Requirements for Students:	20
1. Account Management	20
2. Course Management	21
3. Assignment and Assignment Assessment Management	21
b. User Requirements for Instructors:	23
1. Instructor Account	23
2. Course Management	24
3. Evaluation Team/Groups	24
4. Assignment Management	24
5. Evaluation Survey Management	25
c. User Requirements for Super Admin:	26
1. Account Management	26
11. Requirements that were listed but not implemented	27
Design and Testing	28
12. Software Overview	28
13. List of User Groups and Usage Scenarios:	28
1. Students:	28
2. Instructors:	29
3. Super Admins:	29
Student Usage Scenario (Assignment Submission and Assignment Assessment):	29
Instructor Scenario:	29
Super Admin Scenario:	30
14. Use Case Models	30
a. Instructor Use Cases:	30
b. Student Use Cases:	44
c. Super-Admin Use Cases:	50
15. System Architecture	53
16. Data Flow Diagrams and Sequence Diagrams	54
a. Level 0 Data Flow Diagram	54
b. Level 1 Data Flow Diagram	56
c. Sequence Diagrams	59
17. Technical Specifications	60
18. UI Mockups	61
a. Instructor Frontend mockups	61
b. Student Frontend mockups	68
19. Project Management Methodology, Workflow and WBS/FBS	70
1. Methodology: Kanban	70
2. Workflow	70

1. Product Backlog Refinement:	70
2. Task Management	71
3. Continuous Development:	71
4. Continuous Delivery:	71
5. Ongoing Collaboration:	71
6. Continuous Integration:	71
3. Work Breakdown Structure	72
1. Summary of all the Tasks and the hours clocked	72
2. Work Breakdown structure weekly	83
3. Burnup Charts	84
4. BurnDown Charts	84
20. Features Completed/Core Requirements Delivered	85
21. Testing Plan/QA Plan	90
a. Goal	90
b. Scope	91
c. Out of Scope	91
d. Example test Cases that need to be tested for:	91
e. Testing Techniques	91
f. Security Principles Adhered	92
22. Test Report	93
23. Usability Testing Report	103
a. Goals	103
b. Research Questions	103
c. Participants	103
d. Methodology	103
e. Analysis	104
f. Results	104
g. Conclusion	106
24. Known Bugs	106
25. Key Lessons Learned	107
26. Project Delivery	108
a. Testing and deployment environment	108
1. Frontend Applications Overview	108
2. Microservices Overview (Express)	108
3. Reverse Proxy (Nginx):	109
4. Database Management:	109
5. Database Configuration:	109
6. Database User Access:	110
b. Project repository	110
c. User manual	110

d. User deploy guide for a development environment	111
e. Frontend Library Requirements and their Description	112
f. Backend Dependencies/Libraries and their Description	114
g. Features remaining	115
27. Approvals	115

Project Charter, Scope and Requirements

1. Project Purpose, Problem Identification and Solution

The Peer Review Application is designed with a dual purpose: revolutionizing collaborative learning through instructor-designed group evaluation forms and redefining assignment assessments. By facilitating student evaluations of group members' contributions and enabling instructors to assign and assess peer evaluations for assignments, the application fosters teamwork, accountability, and peer learning. This dynamic approach enhances the educational experience by promoting active engagement, comprehensive feedback, and mutual growth among students.

In many education settings, especially during group projects, there exists a significant challenge in adequately assessing individual contributions. This challenge often arises from the lack of granular insights available to the instructors, forcing them to depend heavily on subjective judgments to determine individual grades. Traditionally, the evaluation of student assignments has been characterized by a predominantly one-sided approach. Here, the instructors and teaching assistants (TAs) have been the primary assessors, and this narrow assessment model frequently results in a feedback loop that is limited in its scope and depth.

Problem Identification:

1. **Lack of individual Contribution Assessment:** Traditional methods do not provide adequate mechanisms to distinctly evaluate each member's contributions in group projects. This can lead to discrepancies in grading, with some students possibly receiving more or less credit than they deserve.
2. **Limitation of Singular Perspective:** Current evaluation methodologies are overly dependent on the viewpoint of the instructor or TA. This reliance on a single perspective often limits the depth of insight into a student's capabilities, progress, and areas needing improvement. There is a potential loss of recognizing the diverse nuances and strengths that peers might recognize in one another.
3. **Inefficiencies in Feedback Loops:** The present systems lack comprehensive feedback mechanisms. Without diverse insights, feedback often becomes one-dimensional, failing to cater to individual growth

trajectories and missing potential areas of improvement.

Proposed Solution:

1. Provide a structured platform for peers to collaboratively evaluate and provide feedback on individual contributions, ensuring fairness and comprehensiveness in the assessment process.
2. Encourage a collaborative assessment approach, enhancing the feedback loop and ensuring a more inclusive evaluation. This will capture diverse perspectives, reducing over-reliance on the instructor's viewpoint.
3. Contribute to a deeper understanding of each student's strengths, areas of improvement, and overall progress, optimizing the learning experience.

2. Project Objective and Related Success Criteria

The objective of this project is to develop a user-friendly platform that facilitates the entire assignment and evaluation process for students and instructors. The goal is to create a seamless system where students can easily submit their assignments and receive timely feedback. Additionally, students will have the opportunity to evaluate a set number of assignments assigned by the instructor, providing valuable input on their peers' work. The platform will also enable students to evaluate the contributions of their fellow group members for group assignments. Moreover, students themselves will be evaluated by their peers in a fair and anonymous manner. This peer evaluation process aims to promote collaboration, accountability, and the development of valuable skills. For instructors, the platform will provide a convenient interface to create assignments, manage evaluations, and view feedback. Instructors will have a comprehensive overview of students' progress, enabling them to assess individual and group performance effectively. Overall, the project's objective is to create an inclusive and interactive learning environment where students can submit assignments, receive feedback on their assignments, evaluate peers' contributions towards group projects/assignments, and receive feedback for their own contributions, while instructors have the necessary tools to manage assignments, evaluations, and track student progress. The platform will empower students to engage actively in the learning process and foster a collaborative and enriching educational experience.

3. Success Metrics

- a. Minimum number of clicks to accomplish a task.
- b. Students can perform the processes of peer evaluation for Group Contribution and individual assignments on the platform.
- c. Instructors can gather all the evaluations provided by the students for individual assignments and group assignments.
- d. Both students and instructors had a good experience after using the platform features.
- e. Stakeholders recognize that the final project product meets their expectations.

4. High-level Requirements

- a. The project aims to develop a comprehensive and user-friendly platform that facilitates efficient management of student assignments, evaluations, and course information.
- b. The system should provide students with easy access to course details, clear assignment tracking, streamlined evaluation processes, and seamless communication.
- c. Additionally, instructors should have robust capabilities for managing assignments, evaluations, and student information, allowing for flexibility in grading, team/group management, and account administration.
- d. The project seeks to enhance the overall learning experience by promoting effective assignment management, timely feedback, and collaborative evaluation among students and instructors.

5. Assumptions And Constraints

Assumptions:

- a. Proficiency in React.js and Node.js for coding can be ensured among the developers.
- b. The capstone team can successfully fulfill all the deliverables within the specified 9-week timeframe.
- c. Availability of stakeholders and team members will not pose significant challenges.

- d. Technical challenges will not hinder the developers progress.
- e. The project manager has the capacity to create a reasonable schedule and
- f. effectively monitor product quality.
- g. Minimal changes to project requirements are expected to facilitate on-time completion.

Constraints:

- a. Limited time remains a significant constraint,
- b. Incorporating new concepts and languages may require a learning curve for developers, potentially affecting efficient implementation.
- c. Unforeseen technical challenges could arise, potentially impeding the development process.

6. High-Level Project Description and Boundaries

- a. The application focuses on students anonymously evaluating other students' assignments and giving feedback. Additionally the application provides students with the ability to anonymously perform group evaluations. It will not include features unrelated to evaluations, such as course content delivery or student enrollment management.
- b. The system will support a predefined set of questions determined by the instructor for evaluations, including Multiple-Choice, Short Answer, Multiple Answer and Matrix Type questions (E.g. Likert Scale, 0-4 Scale). It will not support additional question types that are not within the defined scope.
- c. The system will allow instructors to manage evaluations and view student performance. It will not provide administrative features and course scheduling.
- d. The system will be developed for use on web browsers and tablets and will not be optimized for mobile platforms or devices.
- e. The project will focus on the development of the evaluation system and its core

functionalities. It will not include extensive customization options or integration with external systems beyond and defined scope.

- f. The project will consider integrations with external systems or APIs as necessary for its functioning. However, the project's boundaries will be set to include only the essential integrations, and any additional integrations may be subject to further assessment and approval from the Sponsor.

7. High Level Risks

- a. The limited number of hours available for project completion may impact the ability to achieve all project objectives effectively.
- b. Limited number of weekly client meetings.
- c. No in-person client meeting as the client is not in Kelowna
- d. The potential challenge of developers becoming proficient in new technologies like React.js, Node.js, and express.js could impact the pace of coding functionality.
- e. There is a risk of team members (developers) dropping the class, potentially disrupting the project's continuity.
- f. Developer(s) might get sick and not be able to code in a timely fashion.
- g. Weak team communication

8. Summary Milestone Schedule

Milestone No.	Deliverables	Due Date
1	Draft - Charter, Scope and Requirement Document	May 29, 2023
2	Final Submission - Charter, Scope, Requirements and Planning Milestone Document	June 4, 2023
3	Draft - Design and Testing Document	June 4, 2023
4	Final Submission - Design and Testing Document	June 7, 2023
5	Design Presentation	June 9, 2023
6	Mid-project Status Update and Assessment and MVP Design Review Presentation	July 5, 2023
7	Testing Event	August 4, 2023

8	Practice Presentations and Feedback	August 11, 2023
9	Final Report and Code Submission	August 15, 2023
10	Final Presentation and Project Delivery	August 17, 2023

9. Stakeholders List

- a. Parsa Rajabi (Project Sponsor)
- b. Sehajvir Singh Pannu (Team Lead, Client Liaison and Full-stack Developer)
- c. Lance Xu (Backend Developer)
- d. Prabhmeet Deol (Integration Lead and Backend Developer)
- e. Shila Rahman (Frontend Developer)
- f. Charlotte Zhang (Frontend Developer)

10. Requirements

Functional Requirements:

a. Functional Requirements for Students:

1. Account Management:

- a. The system must provide a user registration feature that allows students to create an account and accurately capture and store the following information: first name, middle name, last name, student Id, student email, and password.
- b. The system must provide a secure login functionality for students to access the platform using their registered email and password, ensuring authentication and verification of credentials.
- c. The system must allow students to manage their account details on the platform, including their name, email, and password, enabling them to update and modify their personal information securely, ensuring control over their account and maintaining account security.
- d. The system must provide students with the ability to reset their passwords in the event of forgetting or needing to change them, offering a secure and accessible password recovery process that ensures the privacy and security of user

accounts.

- e. The system must provide students with the ability to log out of the system, implementing appropriate security measures to ensure a secure disconnection and protect against unauthorized access.

2. Course Management

- a. The system must provide the students with a list of all the courses they are enrolled in by their instructors, clearly displaying the course details which includes, the course name, course code and section, course semester, course term and course delivery year.
- b. The system must provide the students with a list of all the student peers enrolled in the course they are enrolled in.
- c. The system must provide the students with a list of all the group members of their group for a course if the instructor has added them to a group.

3. Assignment and Assignment Assessment Management:

- a. The system must provide students with the ability to view a comprehensive list of assignments due on the platform, displaying relevant details such as assignment name, assignment description, assignment rubric, rating points, assignment available from date, assignment available until date, assignment deadline, accepted submission links, ensuring easy access to assignment information for monitoring and tracking progress.
- b. The system must allow students to submit assignments only as links (e.g., Google Drive link, Dropbox link, Google Spreadsheet link, Google Document Link, Google Slides link) on the platform, as specified by the instructor when creating student assignments.
- c. The system must perform link validation to ensure the submitted links for assignments are valid, specifically checking if the URL contains the appropriate format specified by the instructor (e.g., for a Google Docs submission, the URL should contain 'https://docs.google.com/'). This validation process ensures the reliability and integrity of the submitted work.
- d. The system must provide students with the ability to edit their assignment submission on the platform before the deadline, enabling them to make necessary revisions or updates to their work.

- e. The system must provide students with a list of 'x' number of other students' assignments for assignment assessments, determined by the instructor when creating the assignment, for the purpose of anonymous peer assignment assessment. The system will randomly assign 'x' number of assignments to each student to assess.
- f. The system must provide students with peer-assessed assignment feedback and provide the calculated final grade for an assignment based on the average assessment scores for their submitted assignments.

4. Group Evaluation:

- a. The system must provide students with a list of Group Evaluation Forms and relevant information like Group Evaluation Form name, Deadline and their Feedback/Grade status for a Group Evaluation form once the instructor releases the Form to the students.
- b. The system must provide the students with a list of their group members for which they have to provide evaluations for and their corresponding evaluation submission status.
- c. The system must provide the students with the ability to submit an evaluation for their group members through a comprehensive evaluation form created by the instructor that includes multiple choice questions, multiple answer questions, matrix questions, short answer questions. This allows for the collection of both quantitative and qualitative feedback.
- d. The system must require each student within a group to provide feedback to all other group members, ensuring comprehensive and inclusive evaluation among team members.
- e. The system must prompt the student with a text input if and only if the student evaluating another student does not give a full mark on a particular question. This ensures the provision of constructive feedback and opportunities for growth. The system should not allow a student to proceed until they have provided a comment feedback.
- f. The system must provide the students with the ability to view the scores and comments associated with their evaluations, without revealing the identity of the reviewers. This ensures transparency and provides valuable feedback for self-assessment and improvement.

b. Functional Requirements for Instructors:

1. Instructor Account:

- a. The system must provide a user registration feature that allows students to create an account and accurately capture and store the following information: instructor first name, instructor middle name, instructor last name, employee Id, instructor email, and password.
- b. The system must allow instructors to log in securely to the system using their registered email and password, ensuring secure access to their account and associated functionalities.
- c. The system must provide instructors with the ability to reset their passwords in case of forgetting or needing to change them, ensuring a secure and accessible password recovery process.
- d. The system must provide instructors with the ability to log out of the system, ensuring a secure disconnection and safeguarding their account from unauthorized access.
- e. The system must provide instructors with access to a feature that allows them to manage their account details on the platform, ensuring control over their personal information and account settings.

2. Course Management:

- a. The system must provide the instructors with the ability to create a new course they are teaching. The system must accurately capture course details such as course name, course code and section, course semester, course delivery year and course term.
- b. The system must provide instructors with the ability to enroll students to a course by importing their student IDs using a CSV file. The CSV file format for group enrollment should adhere to a specific format defined by the client.
- c. The system must provide the instructors with a sample CSV file, to which student IDs can be added and subsequently uploaded to the system to enroll students in the course.
- d. The system must allow instructors to create evaluation teams/groups in a course by importing Group names and corresponding student ids of the group members using a CSV file. The CSV file format for group enrollment should adhere to a specific format defined by the client.

3. Evaluation Team/Groups:

- a. The system must provide instructors with the ability to add or remove students from teams/groups, both during and after the group creation process. This feature ensures the necessary flexibility to manage team/group membership as required.
- b. The system must provide the instructors with the ability to add empty evaluation groups to a course by asking for a unique group name.
- c. The system must provide the instructors with the ability to remove/delete student groups from a course at any point of time. Removing a group will remove student-group associations from the system.

4. Assignment Management

- a. The system must provide instructors with the ability to create a new assignment for a course. System must accurately capture the assignment name, assignment description, assignment rubric, assignment submission type, 'x' number of assessments per student (the system will randomly generate 'x' membered groups for assignment assessment), assignment available from date and time, assignment available until date and time (same as assignment assessment deadline), assignment deadline.
- b. The system must provide the instructors with the list of student assignments including their assignment name, assignment available from date and time, assignment available until date and time, assignment deadline date and time.
- c. The system must provide the instructors with the ability to release/hide an assignment to /from the students' course dashboard.
- d. The system must provide the instructors with the ability to release/hide assignment feedback to/from the students' course dashboard.
- e. The system must provide the instructors with the ability to delete an assignment from a course.
- f. The system must provide the instructors with a list of all the students' submissions which will include submission status and submission links if students have successfully submitted an assignment.
- g. The system must provide the instructors with an overview of the student assignment assessments table. The table will include an evaluatee, their corresponding evaluators and link to the evaluatee final grade/feedback page.
- h. The system must provide the instructors with the assignment assessment feedback performed by a student evaluator for a student evaluatee.
- i. The system must provide the instructors with the assignment assessment final

grade calculated using the assignment assessments performed by all student evaluators for a student evaluatee.

- j. The system must provide the instructors with the ability to set and change the assignment assessment final grade for an evaluatee once all the assignment assessments are completed for a student evaluatee, ensuring flexibility in grading and accommodating adjustments, including the provision of 'fudge points'.

5. Evaluation Survey Management:

- a. The system must provide the instructors with the ability to create evaluation forms and select the questions to include, allowing them to design customized assessment criteria and collect specific feedback based on their evaluation requirements. The system must accurately capture evaluation form name, deadline, section names, section weightages, corresponding section questions, question text, question options if applicable and option points if applicable.
- b. The system must prevent the instructor from submitting an evaluation form creation request until and unless all the necessary evaluation creation validations have been successfully met.
- c. The system must provide the instructors with a list of all the evaluation forms they have created along with the following details; evaluation form name, and evaluation form deadline.
- d. The system must provide the instructors with the ability to edit/update an evaluation form before a student performs an evaluation -- once the evaluation has been performed by a single student, the system should not allow the instructor to edit an evaluation form.
- e. The system must provide the instructors with the ability to delete an evaluation form before a student performs an evaluation -- once the evaluation has been performed by a single student, the system should not allow the instructor to delete an evaluation form.
- f. The system must provide the instructors with the ability to publish/unpublish an evaluation form, providing control over the visibility and accessibility of evaluation forms for students
- g. The system must provide the instructors with the ability to publish/unpublish evaluation feedback for an evaluation form, providing control over the feedback data and accessibility of evaluations for students.
- h. The system must provide the instructors with the ability to select an evaluation form and then select a student group and provide the instructor with an

evaluation overview for that group. The overview includes student evaluatees, corresponding student evaluators and a link to view the final grade for the student evaluatee.

- i. The system must provide the instructors with the ability to choose an evaluator for an evaluatee and view the evaluation feedback provided by the evaluator for the evaluatee.
- j. The system must provide the instructors with the ability to export individual evaluation feedback data for a student evaluatee performed by a student evaluator in a CSV file format, allowing for convenient analysis, storage, and further manipulation of the data as needed.
- k. The system must provide the instructors with the ability to view the final grade for an evaluation form for an evaluatee calculated using all evaluation data for the evaluatee.
- l. The system must provide the instructors with the ability to set and change the evaluation form final grade for an evaluatee once all the evaluations have been performed for an evaluatee by the evaluators of their corresponding student evaluation group, ensuring flexibility in grading and accommodating adjustments, including the provision of 'fudge points'.

c. Functional Requirements for Super Admin

1. Account Management:

- a. The system must have a super admin account that is created using a postman call.
- b. The system must provide a secure login functionality for the super admin to access the platform using the email and password created while setting up the super admin account using postman, ensuring authentication and verification of credentials.
- c. The system must provide the super admin with a list of all the instructors within the system.
- d. The system must provide the super admin the ability to grant/revoke instructor

access for instructors within the system.

d. System Functional Requirements

1. Group Evaluations and Assignment Assessments

- a. The system must calculate the final grade for a group evaluation form from the quantifiable data derived from the feedback, allowing for allocation of specific weightage to each question as set by the instructor, ensuring accurate assessment of scores based on predetermined weightings.
- b. The system must deliver notifications through email when an evaluation form or an assignment is released. The system must also deliver notifications through email when feedback for an evaluation form or an assignment assessment is released.

Non-Functional Requirements:

a. Security

- a. The system must grant access to user accounts when users enter the correct username and password, ensuring secure authentication and authorized access to the platform.
- b. The system must not create an account for a user until they create a strong password (At least 12 characters long but 14 or more is better. A combination of uppercase letters, lowercase letters, numbers, and symbols)

b. Compatibility

- a. The system must be designed to run on computers and tablets, considering the impact of display screen size on the user interface (UI) to ensure optimal user experience and usability, while not requiring specific adaptation for mobile devices such as cell phones.

c. Localization

- a. The time of assignments deadline and group evaluations deadline must be in PDT/PST.

d. Usability

- a. Users must be able to predict the meanings of icons, such as inferring that tapping a button with a picture of a magnifying glass will open a search bar, ensuring intuitive and user-friendly interface design.
- b. The user interface elements, layouts, and interactions must be consistent across different devices and screen dimensions, ensuring a unified and seamless user

experience regardless of the device used.

e. Accessibility

- a. The web application must pass the audit test, such as Google Lighthouse, ensuring adherence to best practices, performance optimization, accessibility, and other relevant criteria for an optimal user experience.

Technical Requirements:

1. The front end system will be written in ReactJs framework and Tailwind CSS styling framework will be used for styling.
2. Back end will be written using Node.js and Express.js following MVC design choice.
3. The database driver for this application will be MySQL.
4. The project must make use of Github as a version control system and Github Projects as a Kanban Board.
5. Drone CI will be used to run and build the CI pipeline.
6. Github actions will be used to build the CD pipeline.
7. Every piece of code that is written should be associated with a ticket number using Github issues and the ticket number should be noted in the branch name of Github.
8. All github branches must go through pull request (PR) review and be reviewed by other team members before they're merged into the staging test branch for further development.
9. Docker desktop will be used to run the development environment on the developers local systems.
10. Postman will be used to manually test backend apis.
11. Production environment setup through ssh on Digital Ocean droplet provided by the client.

User Requirements:

a. User Requirements for Students:

1. Account Management

- a. The student will be able to create a new account by providing their first name, middle name (optional), last name (optional), student ID, email, and password, ensuring a seamless and user-friendly registration process.

- b. The student will be able to securely log in to the platform using their registered email and password, expecting a straightforward login process that protects the privacy and security of their account.
- c. The student will be able to manage their account details on the platform, including their name, email, and password, enabling them to update and modify their personal information securely, ensuring control over their account and maintaining account security.
- d. The student will have the option to reset their password in case of forgetting or needing to change it, expecting a user-friendly and easily accessible password reset process.
- e. The student will have the ability to easily access the logout functionality and receive confirmation of a successful disconnection, ensuring a secure disconnection from the system and safeguarding their account from unauthorized access.

2. Course Management

- a. The students will have a list of all the courses they are enrolled in by their instructors, clearly displaying the course details which includes, the course name, course code and section, course semester, course term and course delivery year.
- b. The students will have a list of all the student peers enrolled in the course they are enrolled in.
- c. The students will have a list of all the group members of their group for a course if the instructor has added them to a group.

3. Assignment and Assignment Assessment Management

- a. The students will be able to access a comprehensive list of assignments due on the platform, displaying relevant details such as assignment name, assignment description, assignment rubric, rating points, assignment available from date, assignment available until date, assignment deadline, accepted submission links, ensuring easy access to assignment information for monitoring and tracking progress.
- b. The students will be able to submit assignments only as links (e.g., Google Drive link, Dropbox link, Google Spreadsheet link, Google Document Link, Google Slides link) on the platform, as specified by the instructor when creating student assignments.
- c. The student will have to provide submission links (e.g. for a Google Docs

submission, the URL should contain 'https://docs.google.com/') for assignment submissions that will be validated by the system to ensure the reliability and integrity of the submitted work.

- d. The students will be able to edit their assignment submission on the platform before the deadline, enabling them to make necessary revisions or updates to their work.
- e. The students will have the ability to access and review a list of 'x' number of other students' assignments, as determined by the instructor, to perform anonymous peer evaluation. The student will be randomly given an 'x' number of assignments to assess. This feature will facilitate the fair assessment and feedback exchange among students.
- f. The students will be able to view peer assessed assignment feedback and view the calculated final grade for an assignment based on the average assessment scores for the submitted assignments.
- g. The student must receive notifications through email when their assignment assessment feedback becomes available. This notification mechanism ensures timely communication, allowing students to promptly access and review their assessment results. By providing timely feedback notifications, the system enhances the overall user experience for students.

4. Group Evaluation

- a. The student will be able to access a list of Group Evaluation Forms and relevant information like Group Evaluation Form name, Deadline and their Feedback/Grade status for a Group Evaluation form once the instructor releases the Form to the students.
- b. The students will be able to view a list of their group members for which they have to provide evaluations for and their corresponding evaluation submission status.
- c. The students will be able to submit an evaluation for their group members through a comprehensive evaluation form created by the instructor that includes multiple choice questions, multiple answer questions, matrix questions, short answer questions. This allows for the collection of both quantitative and qualitative feedback.
- d. The student will be able to provide feedback to all other group members, ensuring comprehensive and inclusive evaluation among team members.

- e. When evaluating another student, if a full mark is not given on a question, a student must provide a small comment specifying areas for improvement for that particular question. This requirement ensures that constructive feedback is provided and opportunities for growth are identified. The system should enforce the completion of the comment feedback before allowing the student to proceed, promoting thorough and valuable evaluations.
- f. The student will be able to view the scores and comments associated with their evaluations, without revealing the identity of the reviewers. This ensures transparency and provides valuable feedback for self-assessment and improvement.
- g. The student must receive notifications through email when their evaluation feedback becomes available. This notification mechanism ensures timely communication, allowing students to promptly access and review their assessment results. By providing timely feedback notifications, the system enhances the overall user experience for students.

b. User Requirements for Instructors:

1. Instructor Account

- a. The instructor will be able to create a new account by providing their first name, middle name (optional), last name (optional), employee Id, email, and password, ensuring a seamless and user-friendly registration process.
- b. The instructor will be able to log in securely to the system using their registered email and password, ensuring secure access to their account and associated functionalities.
- c. The instructor will be able to reset their passwords in case of forgetting or needing to change them, ensuring a secure and accessible password recovery process.
- d. The instructor will have the option to reset their password in case of forgetting or needing to change it, expecting a user-friendly and easily accessible password reset process.
- e. The instructor will have the ability to easily access the logout functionality and receive confirmation of a successful disconnection, ensuring a secure disconnection from the system and safeguarding their account from unauthorized access.

2. Course Management

- a. The instructor will have the ability to create a new course they are teaching. The system must accurately capture course details such as course name, course code and section, course semester, course delivery year and course term.
- b. The instructor will have the ability to enroll students to a course by importing their student IDs using a CSV file. The CSV file format for group enrollment should adhere to a specific format defined by the client.
- c. The instructor will have the ability to download a sample csv file provided by the system to which student IDs can be added and subsequently uploaded to the system to enroll students in the course.
- d. The instructor will have the ability to create evaluation teams/groups in a course by importing Group names and corresponding student ids of the group members using a CSV file. The CSV file format for group enrollment should adhere to a specific format defined by the client.

3. Evaluation Team/Groups

- a. The instructor will have the ability to add or remove students from teams/groups, both during and after the group creation process. This feature ensures the necessary flexibility to manage team/group membership as required.
- b. The instructor will have the ability to add empty evaluation groups to a course by asking for a unique group name.
- c. The instructor will have the ability to remove/delete student groups from a course at any point of time. Removing a group will remove student-group associations from the system.

4. Assignment Management

- a. The instructor will have the ability to create a new assignment for a course. Instructor must provide the assignment name, assignment description, assignment rubric, assignment submission type, 'x' number of assessments per student (the system will randomly generate 'x' membered groups for assignment assessment), assignment available from date and time, assignment available until date and time (same as assignment assessment deadline), assignment deadline.
- b. The instructor will have the ability to view the list of student assignments including their assignment name, assignment available from date and time, assignment available until date and time, assignment deadline date and time.
- c. The instructor will have the ability to release/hide an assignment to /from the

students' course dashboard.

- d. The instructor will have the ability to release/hide assignment feedback to/from the students' course dashboard.
- e. The instructor will have the ability to delete an assignment from a course.
- f. The instructor will have the ability to view a list of all the students' submissions which will include submission status and submission links if students have successfully submitted an assignment.
- g. The instructor will have the ability to view an overview of the student assignment assessments table. The table will include an evaluatee, their corresponding evaluators and link to the evaluatee final grade/feedback page.
- h. The instructor will have the ability to view the assignment assessment feedback performed by a student evaluator for a student evaluatee.
- i. The instructor will have the ability to view the assignment assessment final grade calculated using the assignment assessments performed by all student evaluators for a student evaluatee.
- j. The instructor will have the ability to set and change the assignment assessment final grade for a student evaluatee once all the assignment assessments are completed for an evaluatee, ensuring flexibility in grading and accommodating adjustments, including the provision of 'fudge points'.

5. Evaluation Survey Management

- a. The instructor will have the ability to create evaluation forms and select the questions to include, allowing them to design customized assessment criteria and collect specific feedback based on their evaluation requirements. The instructor must accurately provide evaluation form name, deadline, section names, section weightages, corresponding section questions, question text, question options if applicable and option points if applicable.
- b. The instructor will have to adhere to the evaluation form creation validation in order to successfully create an evaluation form.
- c. The instructor will have the ability to view a list of all the evaluation forms they have created along with the following details; evaluation form name, and evaluation form deadline.
- d. The instructor will have the ability to edit/update an evaluation form before a student performs an evaluation -- once the evaluation has been performed by a single student, the instructor will not be able to edit an evaluation form.
- e. The instructor will have the ability to delete an evaluation form before a student

performs an evaluation -- once the evaluation has been performed by a single student, the instructor will not be able to delete an evaluation form.

- f. The instructor will have the ability to publish/unpublish an evaluation form, providing control over the visibility and accessibility of evaluation forms for students.
- g. The instructor will have the ability to publish/unpublish any feedback data, providing control over the feedback data and accessibility of evaluations for students.
- h. The instructor will have the ability to select an evaluation form and then select a student group to get an evaluation overview for that group. The overview includes student evaluatees, corresponding student evaluators and a link to view the final grade for the student evaluatee.
- i. The instructor will have the ability to choose an evaluator for an evaluatee and view the evaluation feedback provided by the evaluator for the evaluatee.
- j. The instructor will have the ability to export individual evaluation feedback data for a student evaluatee performed by a student evaluator in a CSV file format, allowing for convenient analysis, storage, and further manipulation of the data as needed.
- k. The instructor will have the ability to view the final grade for an evaluation form for an evaluatee calculated using all evaluation data for the student evaluatee.
- l. The instructor will have the ability to set and change the evaluation form final grade for a student evaluatee once all the evaluations have been performed for a student evaluatee by the student evaluators of their corresponding student evaluation group, ensuring flexibility in grading and accommodating adjustments, including the provision of 'fudge points'.

c. User Requirements for Super Admin:

1. Account Management

- a. The super admin will have the ability to log in securely to the system using their registered email and password, ensuring secure access to their account and associated functionalities.
- b. The super admin will have the ability to view a list of all the instructors within the system.
- c. The super admin will have the ability to grant/revoke instructor access for instructors within the system

11. Requirements that were listed but not implemented

All the requirements that were listed down in the project charter and scope document have been successfully completed.

Design and Testing

12. Software 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.

13. List of User Groups and Usage Scenarios:

Our Platform, Peer Review Application has two main user groups: students and instructors

1. Students:

This user group consists of individuals who sign up to the Peer Review application and are enrolled in courses by their instructors. The students are required to submit assignments, assignment assessments and group evaluations on the application. They use

the software to submit their individual assignments, evaluate their peers' assignments, assess and provide feedback on group members' contributions towards group related, and receive feedback and grades for their own assignments assessed by other students and for their own group contributions.

2. Instructors:

Instructors are responsible for creating courses, creating assignments, forming groups and evaluations. They use the application to set up assignments, manage peer evaluation processes, provide feedback to students, and monitor overall course progress.

3. Super Admins:

Superadmins are high-level administrators who own the privilege to grant and revoke access to instructors for using the platform.

Here's an example usage scenario for each user group:

Student Usage Scenario (Assignment Submission and Assignment Assessment):

Alex is a college student majoring in computer science. Alex needs to submit his coding assignment for a software development course and provide peer evaluations for his classmates' submitted assignments as well. Alex logs into the Peer Review Application, his instructor has already enrolled him in the course so he navigates to the software development course from the student dashboard, navigates to the assignment table where he selects the assignment named "Software Development Assignment 1", submits a link to his code in the requested submission type set by the instructor before the assignment deadline. He now moves on to the Assignment Assessments, where he reviews three randomly assigned classmates' "Software Development Assignment 1" submissions using the rubric provided by the instructor for the same assignment. He provides ratings for the different criterias in the rubric and also provides detailed feedback wherever applicable. This process helps his peers improve their future projects and fosters constructive collaboration.

Instructor Scenario:

Professor Smith is an experienced computer science instructor. Professor Smith aims to create an engaging assignment that encourages students to build and develop functional mobile applications as a team for his software development course. Professor

Smith uses the Peer Review Application to set up the "Software Development Assignment 1" in his software development course where he has already enrolled students. He outlines evaluation criteria including app functionality, user interface design, and team collaboration. He assigns students into development teams and guides them throughout the project. As the assignment progresses, he monitors each team's submission of their app prototype and related documentation. He also tracks the peer evaluation process, ensuring fair assessment among student teams. By reviewing the aggregated feedback and evaluation scores, he measures the effectiveness of the assignment and identifies areas for enhancing the learning experience.

Super Admin Scenario:

John is a super admin. Professor Smith needs instructor access to perform course related activities. Professor Smith contacts admin John to ask him to grant him instructor access.

14. Use Case Models

a. Instructor Use Cases:

Legend:

UC-IA: Use Cases for Instructor for Account

UC-IC: Use Cases for Instructor for Course Management

UC-IE: Use Cases for Instructor for Evaluations Management

UC-IAS: Use Cases for Instructor for Assignments/Assignment Assessments

Use Case1	UC-IA-1
Name:	Create Account
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. Users enter their user first name, middle name, last name, student or instructor ID, email and password. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If credentials are valid, User is redirected to the login page.
Pre-Conditions	<ul style="list-style-type: none"> • User has a device that is connected to the internet. • User knows their instructor or student ID. • All information the user has entered is correct in format. • There is no duplicate instructor or student ID, email with the one

	user provided.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If validated , account information will be added to the database, and the user will be redirected to the login page.

Use Case 2	UC-IA-2
Name:	Login
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User enters their email and password. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If credentials are valid, User is redirected to the course list page.
Pre-Conditions	<ul style="list-style-type: none"> • User has a device that is connected to the internet. • User has an account.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If validated , the user is successfully logged in and redirected to the application dashboard.

Use Case 3	UC-IA-3
Name:	Logout
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User clicks the logout button on the top-right corner. 2. User has been redirected to the Login page. 3. JWT Token has expired.
Pre-Conditions	<ul style="list-style-type: none"> • User has a device that is connected to the internet. • User has an account. • User has already logged in to the system.
Post-Conditions	<ul style="list-style-type: none"> • After logout, users can login or sign up as instructor, student or super admin.

Use Case 4	UC-IC-1
Name:	Create New Course
Actors:	Instructor, Instructor application, database

Flow of Events	<ol style="list-style-type: none"> 1. User enters the course name, code, section, semester, year and term. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, courses created successfully will be displayed and the user will be redirected to the course list page.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User knows the information for the course they are creating. • All information the user has entered is correct in format. • There is no duplicate course that has the same info with the creating one in the database.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the Instructor can see a list of courses displayed as cards with basic information about that course.

Use Case 5	UC-IC-2
Name:	Publish and Unpublish Course
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User press publish and unpublish course toggle. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, the course will be visible or invisible to enrolled students.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, course will be visible or invisible to enrolled students.

Use Case 6	UC-IC-3
Name:	View Course Student List
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has arrived on the student list page. 2. User waits for the server to retrieve data from the database. 3. If no data returned, shows no student enrolled in this course. 4. If data is returned, display the student information.

Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If there is data returned, User can see a list of students displayed as rows with some information about that student. • Instructor will be able to add new students to the course.

Use Case 7	UC-IC-4
Name:	View Group Student List
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has arrived on the group student list page. 2. User waits for the server to retrieve data from the database. 3. If no data returned, shows no student enrolled in this group. 4. If data is returned, display the student information.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If there is data returned, User can see a list of students displayed as rows with some information about that student. • Instructor will be able to add new students to the group.

Use Case 8	UC-IC-5
Name:	View Course List
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the course list page. 2. User waits for the server to retrieve data from the database. 3. If no data is returned, the user will be presented with an error. 4. If data is returned, display a list of all the courses and their basic information.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in. • User has enrolled in one or more courses.
Post-Conditions	<ul style="list-style-type: none"> • If data is returned, display a list of all the courses and their basic information. • Instructor will be able to create a new course.

Use Case 9	UC-IC-6
Name:	View Course Group List
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the course group list page. 2. User waits for the server to retrieve data from the database. 3. If no data is returned, the user will be presented with an error. 4. If data is returned, display a list of course groups with basic information.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If data returns, a list of groups that belong to this course will be displayed.

Use Case 10	UC-IC-7
Name:	Add Students to Course by CSV
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User submits CSV files. 2. A File preview will be provided to let the user know what will be sent to the server. 3. After the user checks for the data ready to submit, press the submit button. 4. User waits for the server to verify credentials. 5. If credentials are invalid, User is presented with an error. 6. If successful, users will be redirected to the course dashboard.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User has the student info CSV file ready to upload. • All information in the CSV file is correct in format and there is no duplicate data in the database.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the Instructor will be redirected to the course dashboard.

Use Case 11	UC-IC-8
Name:	Add Groups to Course by CSV
Actors:	Instructor, Instructor application, database

Flow of Events	<ol style="list-style-type: none"> 1. User submits CSV files. 2. A File preview will be provided to let the user know what will be sent to the server. 3. After the user checks for the data ready to submit, press the submit button. 4. User waits for the server to verify credentials. 5. If credentials are invalid, User is presented with an error. 6. If successful, users will be redirected to the course dashboard and all groups will be added.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User has the group info CSV file ready to upload. • All information in the CSV file is correct in format and no duplicate with the database.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, Instructor will be redirected to the course dashboard and all groups will be added.

Use Case 12	UC-IC-9
Name:	Add Empty Groups to Course
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User enters the name of the new group. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, a new group will be added to the course.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User knows the group name to be created. • There is no duplicate group name in the database.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the Instructor will be redirected to the course dashboard.

Use Case 13	UC-IC-10
Name:	Remove Students from Group
Actors:	Instructor, Instructor application, database

Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the group members page 2. User select the student that they want to remove from the group 3. User waits for the server to verify credentials. 4. If credentials are invalid, User is presented with an error. 5. If successful, the selected student will be removed.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User knows which student to remove.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User is presented with an error. • If successful, selected student will be removed.

Use Case 14	UC-IC-11
Name:	Add Students to Group
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the group members page 2. User select the student that they want to add to the group from the course student list 3. User waits for the server to verify credentials. 4. If credentials are invalid, User is presented with an error. 5. If success, selected students will be removed.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User knows which student to add. • The student to be added is not in any other group that belongs to this course.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User is presented with an error. • If successful, selected students will be added.

Use Case 15	UC-IC-12
Name:	Remove Group from Course
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the course group list page 2. User select the group that they want to remove from the course 3. User waits for the server to verify credentials. 4. If credentials are invalid, User is presented with an error. 5. If successful, the selected group will be removed.

Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User knows which group to remove.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User is presented with an error. • If successful, the selected group will be removed.

Use Case 16	UC-IE-1
Name:	View Course Evaluation List
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated the evaluation list page. 2. User waits for the server to retrieve data from the database. 3. If no data return, shows no evaluation created for this course. 4. If data returns, display all the evaluation forms.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If data returns, a list of evaluations will be displayed as rows.

Use Case 17	UC-IE-2
Name:	Create Evaluation Form
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User enters all information required. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, the Instructor will be redirected to the course dashboard.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User knows the information for the evaluation form they are creating. • All information the user has entered is correct in format. • All section weightage must be added up as 100. • There is no duplicate form that has the same name for this course in the database.

Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the Instructor will be redirected to the course dashboard.
-----------------	--

Use Case 18	UC-IE-3
Name:	Release Evaluation Form
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User press release form toggle 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, the form will be visible and able to be filled to all course students.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User wants the form to be visible to all the course students.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, form will be visible and able to be filled to all course students.

Use Case 19	UC-IE-4
Name:	Release Evaluation Form Feedback
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User press release feedback toggle 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, form feedback will be visible to all the course students.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • Form must be already released. • User wants the form feedback to be visible to all the course students.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, form feedback will be visible to all the course students.

Use Case 20	UC-IE-5
-------------	---------

Name:	View Evaluation Feedback
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the evaluation feedback page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the evaluation to the user.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • Evaluator has evaluated the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the evaluation to the user.

Use Case 21	UC-IE-6
Name:	Edit Evaluation Form
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. Users edit all information required. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, users will be redirected to the course evaluation form list page.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • The form to be edited has already been created. • User knows the information for the evaluation form they are editing. • All information the user has entered is correct in format. • All section weightage must be added up as 100.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the Instructor will be redirected to the course dashboard.

Use Case 22	UC-IE-7
Name:	Remove Evaluation Form
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User presses the delete button at the end of the row. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, the evaluation form will be removed from the database.

Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • There must be no evaluation submitted on this form.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, Evaluation Form will be removed from the database.

Use Case 23	UC-IE-8
Name:	View Evaluatee Grade
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to evaluatee grade page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the grade summary to the user.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • All evaluators have evaluated the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the grade summary to the user.

Use Case 24	UC-IE-9
Name:	Edit Evaluatee Grade
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to evaluatee grade page. 2. User enter the new grade. 3. User waits for the server to insert data to the database. 4. If credentials are invalid, User is presented with an error. 5. If successful, student grades will be updated.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • All evaluators have evaluated the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If successful, student grades will be updated.

Use Case 25	UC-IAS-1
Name:	View Course Assignment List
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated the assignment list page. 2. User waits for the server to retrieve data from the database. 3. If no data return, shows no assignment created for this course. 4. If data returns, display all the course assignments.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If data returns, a list of assignments will be displayed as rows.

Use Case 26	UC-IAS-2
Name:	Create Assignment and Rubric
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. Users enter all information required. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, the Instructor will be redirected to the course dashboard.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User knows the information for the assignment and rubric info they are creating. • All information the user has entered is correct in format. • All section weightage must be added up as 100. • There is no duplicate assignment that has the same name for this course in the database.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the Instructor will be redirected to the course dashboard.

Use Case 27	UC-IAS-3
Name:	Release Assignment
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User press release form toggle 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error.

	4. If successful, the assignment will be visible and able to be answered to all course students.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • User wants the assignment to be visible to all the course students.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, the assignment will be visible and able to be answered to all course students.

Use Case 28	UC-IAS-4
Name:	Release Assignment Assessment Feedback
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User press release feedback toggle 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, assignment assessment feedback will be visible to all the course students.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • Assignment must be already released. • User wants the assignment assessment feedback to be visible to all the course students.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, assignment assessment feedback will be visible to all the course students.

Use Case 29	UC-IAS-5
Name:	View Assignment Assessment
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the Assignment assessment page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the assessment to the user.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • Evaluator has assessed the evaluatee.

Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the assessment to the user.
-----------------	--

Use Case 30	UC-IAS-6
Name:	Remove Assignment
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User press the trash can button at the end of the row. 2. User waits for server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If success, Assignment will be removed from the database.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface. • There must be no assessment submitted on this assignment.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If creation is successful, Assignment will be removed from the database.

Use Case 31	UC-IAS-7
Name:	View Assignment Grade
Actors:	Instructor, Instructor application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the assignment grade page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the grade summary to the user.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • All evaluators have assessed the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the grade summary to the user.

Use Case 32	UC-IAS-8
Name:	Edit Assignment Grade
Actors:	Instructor, Instructor application, database

Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to evaluatee grade page. 2. User enters the new grade. 3. User waits for the server to insert data to the database. 4. If credentials are invalid, User is presented with an error. 5. If successful, student grades will be updated.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • All evaluators have assessed the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If successful, student grades will be updated.

b. Student Use Cases:

Legend:

UC-SA: Use cases for Students for Student Account

UC-SC: Use cases for Students for Course management

UC-SE: Use cases for Students for Evaluations

UC-SAS: Use cases for Students for Assignment and Assignment assessments

Use Case 33	UC-SA-1
Name:	Create Account
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User enters their user first name, middle name, last name, student or instructor ID, email and password. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If credentials are valid, User is redirected to the login page.
Pre-Conditions	<ul style="list-style-type: none"> • User has a device that is connected to the internet. • User knows their instructor or student ID. • All information the user has entered is correct in format. • There is no duplicate instructor or student ID, email with the one user provided.

Post-Conditions	<ul style="list-style-type: none"> ● If credentials are invalid, Users can restart flow or exit. ● If validated , account information will be added to the database, and the user will be redirected to the login page.
-----------------	---

Use Case 34	UC-SA-2
Name:	Login
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User enter their email and password. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If credentials are valid, User is redirected to the course list page.
Pre-Conditions	<ul style="list-style-type: none"> ● User has a device that is connected to the internet. ● User has an account.
Post-Conditions	<ul style="list-style-type: none"> ● If credentials are invalid, Users can restart flow or exit. ● If validated, the user is successfully logged in and redirected to the course list page.

Use Case 35	UC-SA-2
Name:	Logout
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User click logout button on the top-right corner. 2. User has been redirected to the Login page. 3. Token expired.
Pre-Conditions	<ul style="list-style-type: none"> ● User has a device that is connected to the internet. ● User has an account. ● User has already logged in to the system.
Post-Conditions	<ul style="list-style-type: none"> ● After logout, users can login or sign up as instructor, student or super admin.

Use Case 36	UC-SC-1
Name:	View Course Student List
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User have arrived on the student list page. 2. User waits for the server to retrieve data from the database. 3. If no data returned, shows no student enrolled in this course. 4. If data is returned, display the student information.

Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If there is data returned, User can see a list of students displayed as rows with some information about that student.

Use Case 37	UC-SC-2
Name:	View Course List
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the course list page. 2. User waits for the server to retrieve data from the database. 3. User receives a response that is sent back from the server, and the system will list all the courses and their basic information.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in. • User has enrolled in one or more courses.
Post-Conditions	<ul style="list-style-type: none"> • Users receive a response that is sent back from the server, and the system will list all the courses and their basic information.

Use Case 38	UC-SC-3
Name:	View Group Student List
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User have arrived on the group student list page. 2. User waits for the server to retrieve data from the database. 3. If no data returned, shows no student enrolled in this group. 4. If data is returned, display the student information.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If there is data returned, User can see a list of students displayed as rows with some information about that student.

Use Case 37	UC-SE-1
Name:	View Course Evaluation List
Actors:	Student, Student application, database

Flow of Events	<ol style="list-style-type: none"> 1. User has navigated the evaluation list page. 2. User waits for the server to retrieve data from the database. 3. If no data return, shows no evaluation created for this course. 4. If data returns, display all the evaluation forms.
Pre-Conditions	<ul style="list-style-type: none"> ● User has an ongoing connection with the server. ● User has logged in as an instructor. ● User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> ● If data returns, a list of evaluations will be displayed as rows.

Use Case 38	UC-SE-2
Name:	Evaluate others
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User evaluate others based on the evaluation form. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, evaluations will be stored into the database.
Pre-Conditions	<ul style="list-style-type: none"> ● User has an ongoing connection with the server. ● User has logged in as an instructor. ● User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> ● If data is returned, evaluations will be stored into the database.

Use Case 39	UC-SE-3
Name:	View Evaluation Feedback
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the evaluation feedback page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the evaluation to the user.
Pre-Conditions	<ul style="list-style-type: none"> ● User has an ongoing connection with the server. ● User has logged in as an instructor. ● Evaluator has evaluated the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> ● If credentials are invalid, User exits the flow. ● If data is returned, display the evaluation to the user.

Use Case 40	UC-SE-4
-------------	---------

Name:	View Evaluatee Grade
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to evaluatee grade page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the grade summary to the user.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • All evaluators have evaluated the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the grade summary to the user.

Use Case 41	UC-SAS-1
Name:	View Course Assignment List
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated the assignment list page. 2. User waits for the server to retrieve data from the database. 3. If no data return, shows no assignment created for this course. 4. If data returns, display all the course assignments.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If data returns, a list of assignments will be displayed as rows.

Use Case 42	UC-SAS-2
Name:	Submit Assignment
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User submits assignment links to the system. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If data returned, assignment submission will be stored into the database.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If data is returned, assignment submission will be stored into the

	database.
--	-----------

Use Case 43	UC-SAS-3
Name:	Evaluate others
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User assess others based on the assignment rubric. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If data returned, assessment will be stored into the database.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • User has navigated to the course interface.
Post-Conditions	<ul style="list-style-type: none"> • If data returned, assessment will be stored into the database.

Use Case 44	UC-SAS-4
Name:	View Assignment Assessment
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the Assignment assessment page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the assessment to the user.
Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • Evaluator has assessed the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the assessment to the user.

Use Case 45	UC-SAS-5
Name:	View Assignment Grade
Actors:	Student, Student application, database
Flow of Events	<ol style="list-style-type: none"> 1. User has navigated to the assignment grade page. 2. User waits for the server to retrieve data from the database. 3. If credentials are invalid, User is presented with an error. 4. If data is returned, display the grade summary to the user.

Pre-Conditions	<ul style="list-style-type: none"> • User has an ongoing connection with the server. • User has logged in as an instructor. • All evaluators have assessed the evaluatee.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, User exits the flow. • If data is returned, display the grade summary to the user.

c. Super-Admin Use Cases:

Legend:

UC-IA: Use Cases for Instructor for Account

UC-IC: Use Cases for Instructor for Course Management

UC-IE: Use Cases for Instructor for Evaluations Management

UC-IAS: Use Cases for Instructor for Assignments/Assignment Assessments

Use Case 46	UC-AA-1
Name:	Login
Actors:	Super Admin, Super Admin application, database
Flow of Events	<ol style="list-style-type: none"> 1. User enter their email and password. 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If credentials are valid, User is redirected to the course list page.
Pre-Conditions	<ul style="list-style-type: none"> • User has a device that is connected to the internet. • User has an account.
Post-Conditions	<ul style="list-style-type: none"> • If credentials are invalid, Users can restart flow or exit. • If validated , the user is successfully logged in and redirected to the course list page.

Use Case 47	UC-AA-2
Name:	Logout
Actors:	Super Admin, Super Admin application, database
Flow of Events	<ol style="list-style-type: none"> 1. User clicks the logout button on the top-right corner. 2. User has been redirected to the Login page. 3. Token expired.

Pre-Conditions	<ul style="list-style-type: none"> ● User has a device that is connected to the internet. ● User has an account. ● User has already logged in to the system.
Post-Conditions	<ul style="list-style-type: none"> ● After logout, users can login or sign up as instructor, student or super admin.

Use Case 48	UC-AA-3
Name:	Update Instructor Access Permission
Actors:	Super Admin, Super Admin application, database
Flow of Events	<ol style="list-style-type: none"> 1. User set Instructor Access Permission to be true or false 2. User waits for the server to verify credentials. 3. If credentials are invalid, User is presented with an error. 4. If successful, Instructor Access Permission will be added or removed.
Pre-Conditions	<ul style="list-style-type: none"> ● User has an ongoing connection with the server. ● User has logged in as super admin. ● Users know which instructor to give or take Access Permission.
Post-Conditions	<ul style="list-style-type: none"> ● If credentials are invalid, Users can restart flow or exit. ● If creation is successful, Instructor Access Permission will be added or removed.

Figure 1.1 Use case diagram

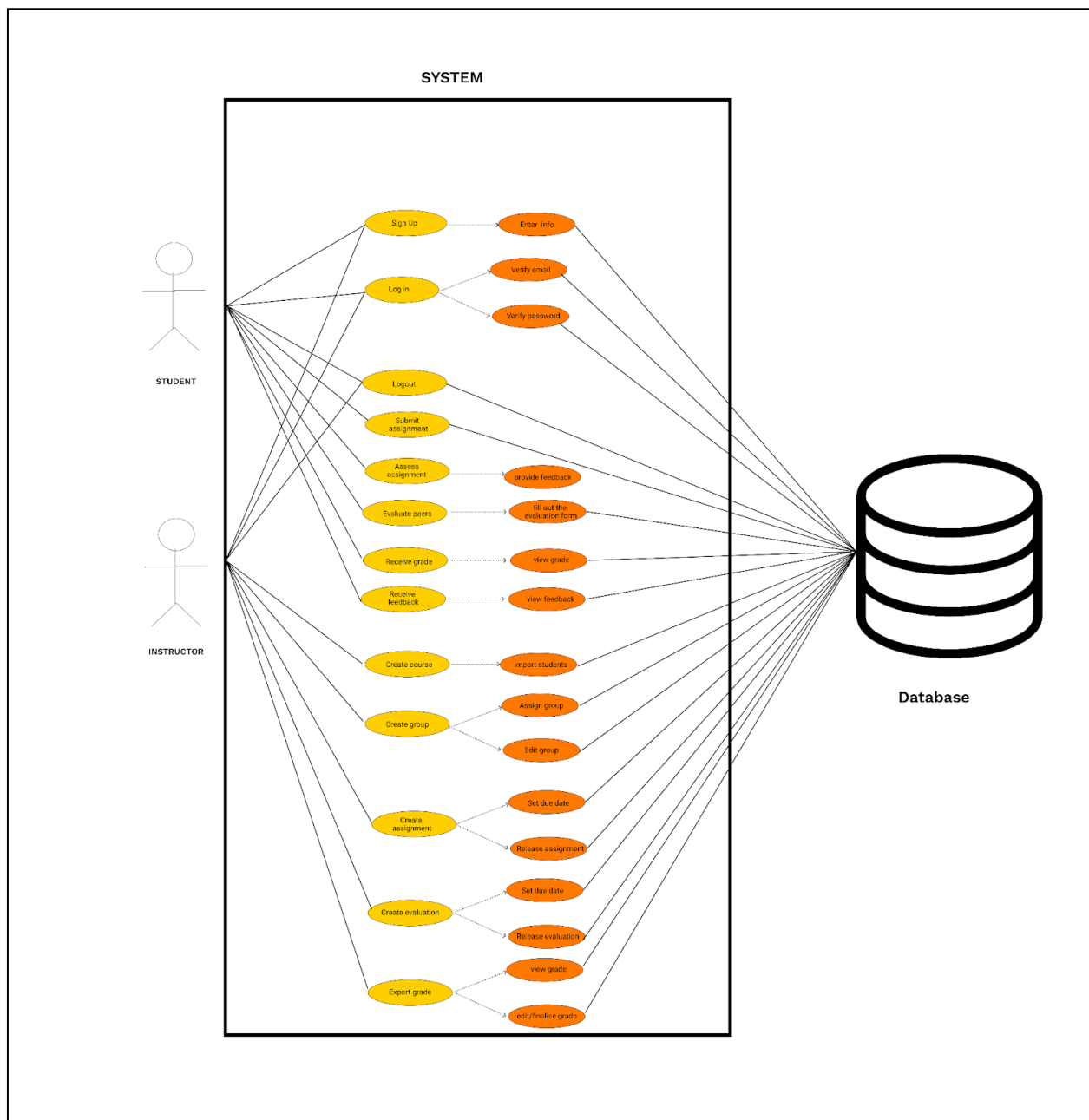


Figure 1.1 Shows the interactions between the main actors in the system, and some of the main actions in the application being performed.

15. System Architecture

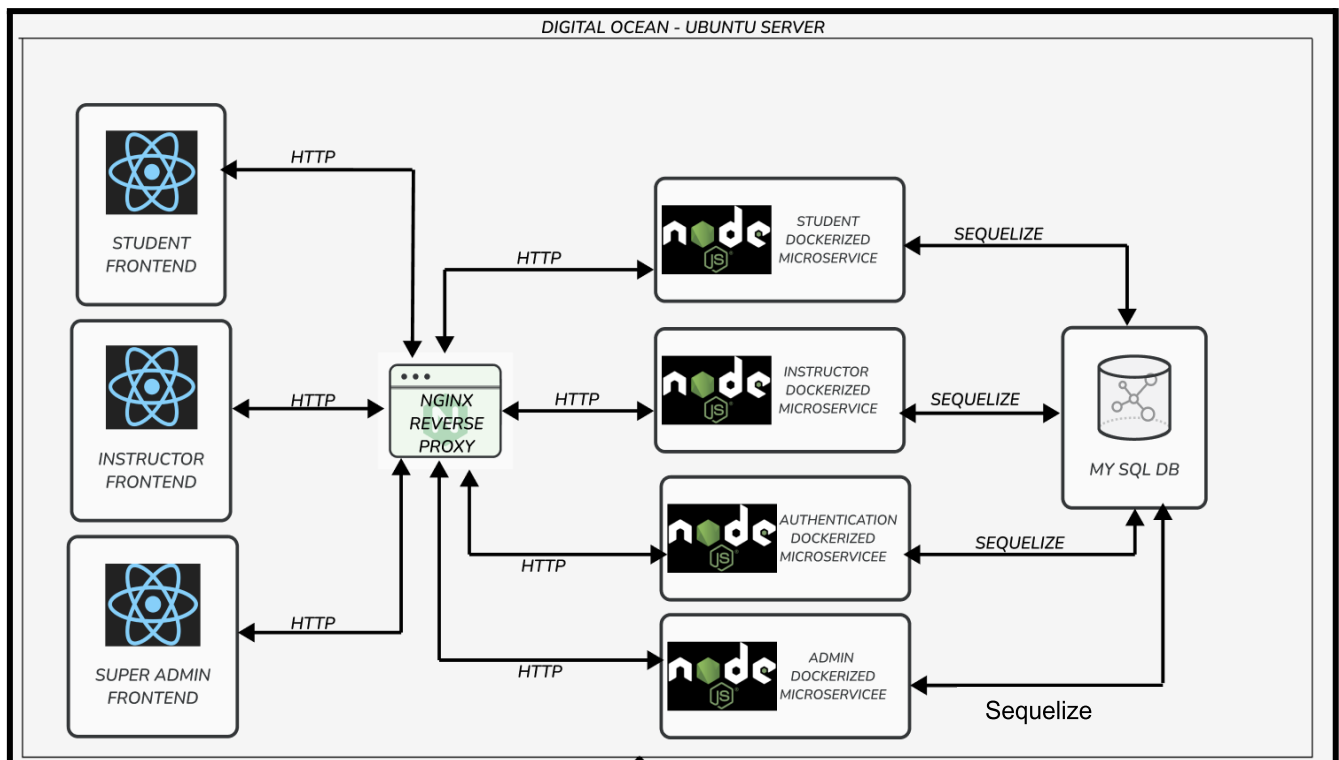


Figure 2.1 System architecture diagram of the Peer Review Application. Which highlights how each individual component interacts with each other.

- Frontend Applications** - Each Front End application is Built using React, and sends HTTP requests to the Nginx Reverse proxy , which acts as a gateway to the remaining Microservices. This can be another point of security which can check for DDOS attacks, and can blacklist / whitelist IP addresses (All this can be controlled by the nginx config files).
- Operating System** - All the applications (frontend and Backend Microservices) and every other Architecture component are hosted using Docker containers. These applications are built and compiled using Node js environments. Then all the containers are able to communicate with each other using a docker network.
- Why Multiple Microservices were used** - Multiple Microservices were used for the purposes of more code organization, Simpler Testing, More reliability (if code breaks down in one Microservice, rest of the services can still serve functionality), also can be easily scaled accordingly using an orchestration engine separately in the future. The Authentication microservice was built separately, so that if the client in the future wants to change the authentication they can easily do that as well.
- Controller Service Infrastructure for Microservices** - All the microservices have Controllers and Microservices for each individual route.

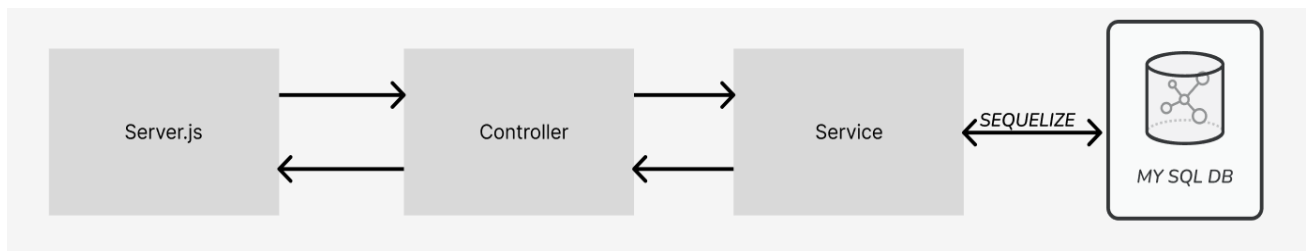


Figure : Sample Backend Microservice Architecture

The controllers are functions that extract the variables from the http request body. In these functions the variables are validated , to see if they have valid data or not , else an error is returned. Later those variables are extracted , and sent to the service functions. The service functions implement the Business logic through the sequelize library , then if data transformations need to be performed they are performed here.

16. Data Flow Diagrams and Sequence Diagrams

a. Level 0 Data Flow Diagram

Figure 2.2 Level 0 Data Flow Diagram illustrating the high-level overview of how data is passed within the system

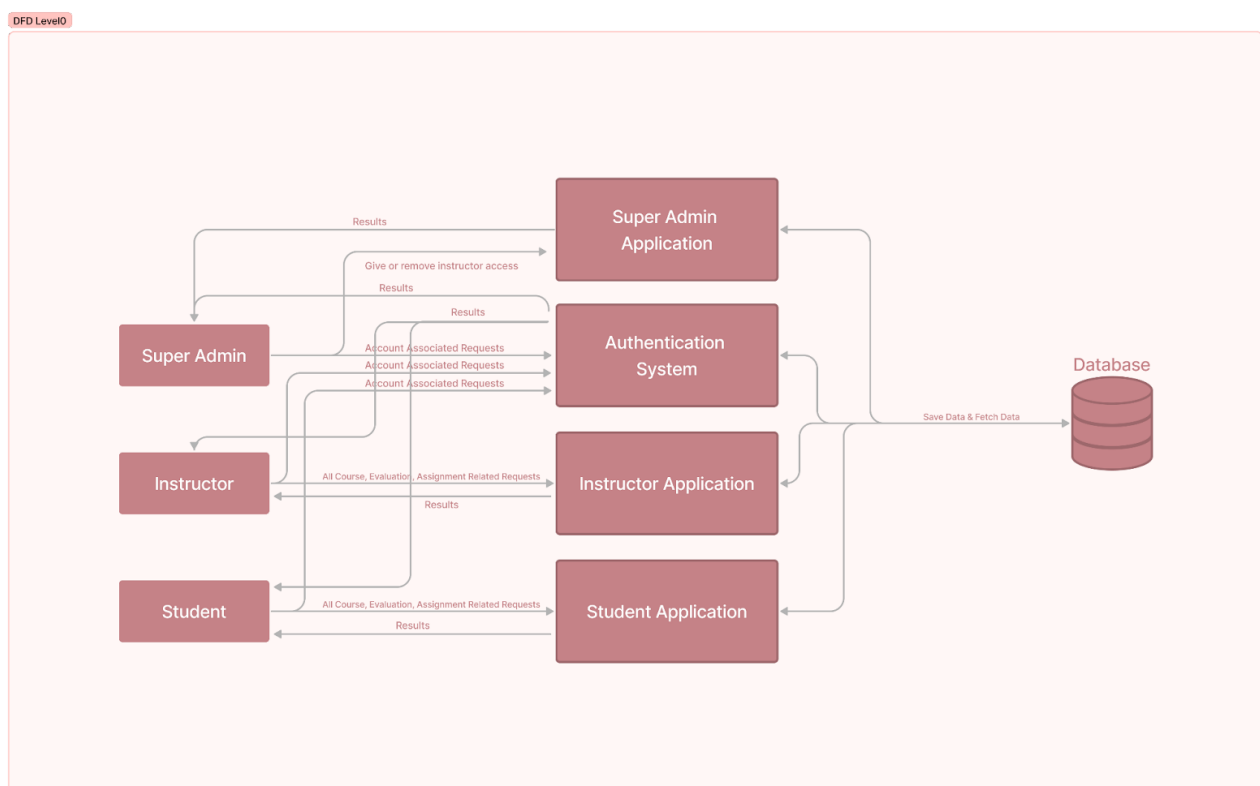


Figure 2.2 highlights the high level flow of data which is originating from the user's computer and is sent to the different user applications which further sends and receives data from the database and sends it back to the user via the application. Please note that since the project has a huge set of requirements and use cases the DFD does not show all the processes in the diagram but comprises an important subset of those requirements.

Entities:

1. **Super Admin:** This external entity represents the administrative interface that manages user roles and permissions. It interacts with the "Give or Remove Instructor Access" process.
2. **Instructor:** An external entity representing instructors who engage with course-related activities. They interact with the "Instructor Application" and contribute to "All Course, Evaluation, Assignment Related Requests."
3. **Student:** An external entity, representing students who participate in course-related activities. They interact with the "Student Application" and contribute to "All Course, Evaluation, Assignment Related Requests."
4. **Super Admin Application:** An external entity that serves as the administrative interface for managing instructor access within the system.
5. **Authentication System:** An internal component that interacts with other processes and entities within the system to ensure secure access and data protection.
6. **Instructor Application:** An external entity that serves as the interface designed to cater the actions of instructors within the system.
7. **Student Application:** An external entity that serves as the interface designed to cater the actions of students within the system.
8. **Database:** stores and manages the structured data used by the system. It serves as a foundational component that stores user information, course details, authentication data, and other relevant information necessary for the system's operation.

Processes:

1. **Give or Remove Instructor Access Request:** This process allows the Super Admin to grant or revoke instructor access privileges. It receives requests from the Super Admin App and interacts with the "Account Associated Requests" process.
2. **Account Associated Requests:** This process manages requests related to user accounts. It is responsible for handling account management actions such as login and forgot password.
3. **All Course, Evaluation, Assignment Related Requests:** This process manages requests related to course related actions, which further include evaluation and assignment related actions.

4. **Save Data and Fetch Data:** This process is responsible for sending requests to the database to collect and fetch corresponding data from the database.

b. Level 1 Data Flow Diagram

Figure 2.3 Level 1 Data Flow Diagram from the user's computer to the different microservices in the system and then to the database and then back to the user's computer.

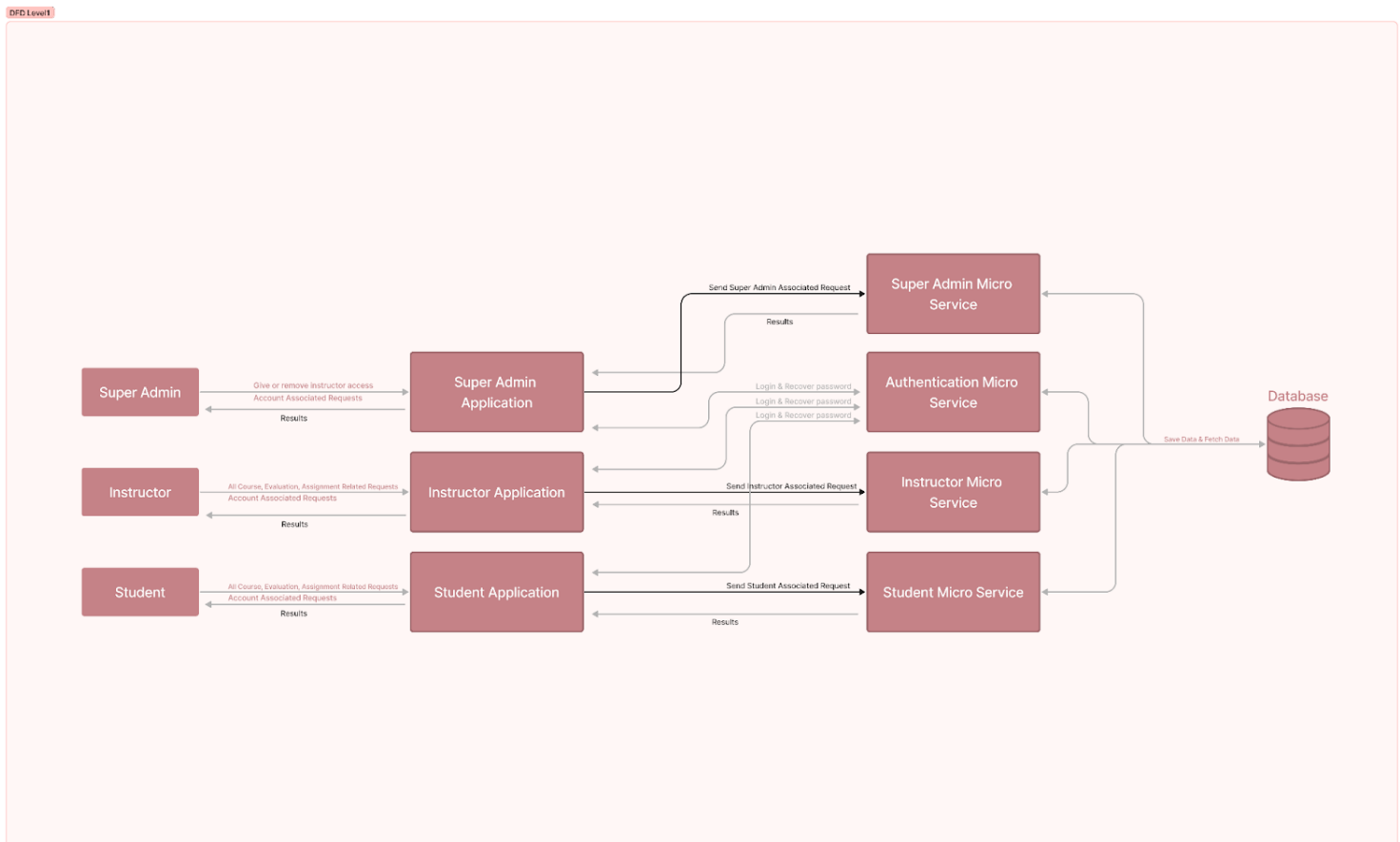


Figure 2.3 outlines the flow of data which is originating from the user's computer and is sent to the different microservices which handle the user actions. Depending on the request, the data is collected or stored from or to the database respectively. Once the data is being collected from the database it is then passed to the user application to view for the respective users.

Entities:

1. **Super Admin:** This external entity represents the administrative interface that manages user roles and permissions. It interacts with the "Give or Remove Instructor Access" process.
2. **Instructor:** An external entity representing instructors who engage with course-related activities. They interact with the "Instructor Application" and contribute to "All Course, Evaluation, Assignment Related Requests."

3. **Student:** An external entity, representing students who participate in course-related activities. They interact with the "Student Application" and contribute to "All Course, Evaluation, Assignment Related Requests."
4. **Super Admin Application:** An external entity that serves as the administrative interface for managing instructor access within the system.
5. **Authentication System:** An internal component that interacts with other processes and entities within the system to ensure secure access and data protection.
6. **Instructor Application:** An external entity that serves as the interface designed to cater the actions of instructors within the system.
7. **Student Application:** An external entity that serves as the interface designed to cater the actions of students within the system.
8. **Super Admin Microservice:** The microservice handles requests from the super admin via the super admin application and depending on the type of request, it receives or stores data from or to the database and respectively returns it back to the super admin user.
9. **Instructor Microservice:** manages interactions between instructors and the system. It handles requests from the 'Instructor Application,' retrieves or stores data in the database based on request types, and provides accurate responses to instructors, enabling effective course management and communication.
10. **Student Microservice:** facilitates student-system interactions. It processes requests originating from the 'Student Application', manages data exchange with the database according to request specifics, and delivers precise responses to students, enhancing their ability to engage with courses and access relevant information.
11. **Authentication Microservice:** is dedicated to user authentication and access control. It handles login, logout, reset password, and forgot password functionalities for instructors, students, and super admins. This microservice ensures secure access to the system, maintaining data confidentiality and providing a seamless authentication experience.
12. **Database:** stores and manages the structured data used by the system. It serves as a foundational component that stores user information, course details, authentication data, and other relevant information necessary for the system's operation.

Processes:

1. **Give or remove instructor access:** This process allows the Super Admin to grant or revoke instructor access privileges. It receives requests from the Super Admin App and interacts with the "Account Associated Requests" process.

2. **Account associated requests:** This process manages requests related to user accounts. It is responsible for handling account management actions such as login and forgot password.
3. **Super admin associated requests:** This process manages all services related to the super admin, including viewing the information of all instructors in the system and granting or revoking instructor access privileges. Users send requests through the front-end super admin application, and then the application forwards the request to the backend super admin microserver for data retrieval.
4. **All Course, Evaluation, Assignment related request:** This process manages requests related to course related actions, which further include evaluation and assignment related actions.
5. **Instructor Associated requests:** This process manages all services related to the instructor user group. User actions are converted into JSON format requests, which are sent from the front-end instructor application to the back-end instructor microserver.
6. **Student Associated requests:** This process manages all services related to the student user group. User actions are converted into JSON format requests, which are sent from the front-end student application to the back-end student microserver.
7. **Save Data & Fetch Data:** This process is responsible for sending requests to the database to collect and fetch corresponding data from the database.
8. **Login:** The login process provides users with the ability to login into the system. User actions are converted into JSON format requests, which are sent from the front-end application to the back-end authentication microserver. All users must obtain the JWT token returned by this server before they have permission to use other services.
9. **Password Recovery:** The password recovery process allows users to reset their forgotten passwords. When a user forgets their password, they initiate a request through the front-end application. This action is then converted into a JSON format request and is sent to the back-end authentication microserver. The server responds by sending a verification email and a JWT token. After the user enters the verification code sent by the email with the JWT token, the user will be able to reset their password.

c. Sequence Diagrams

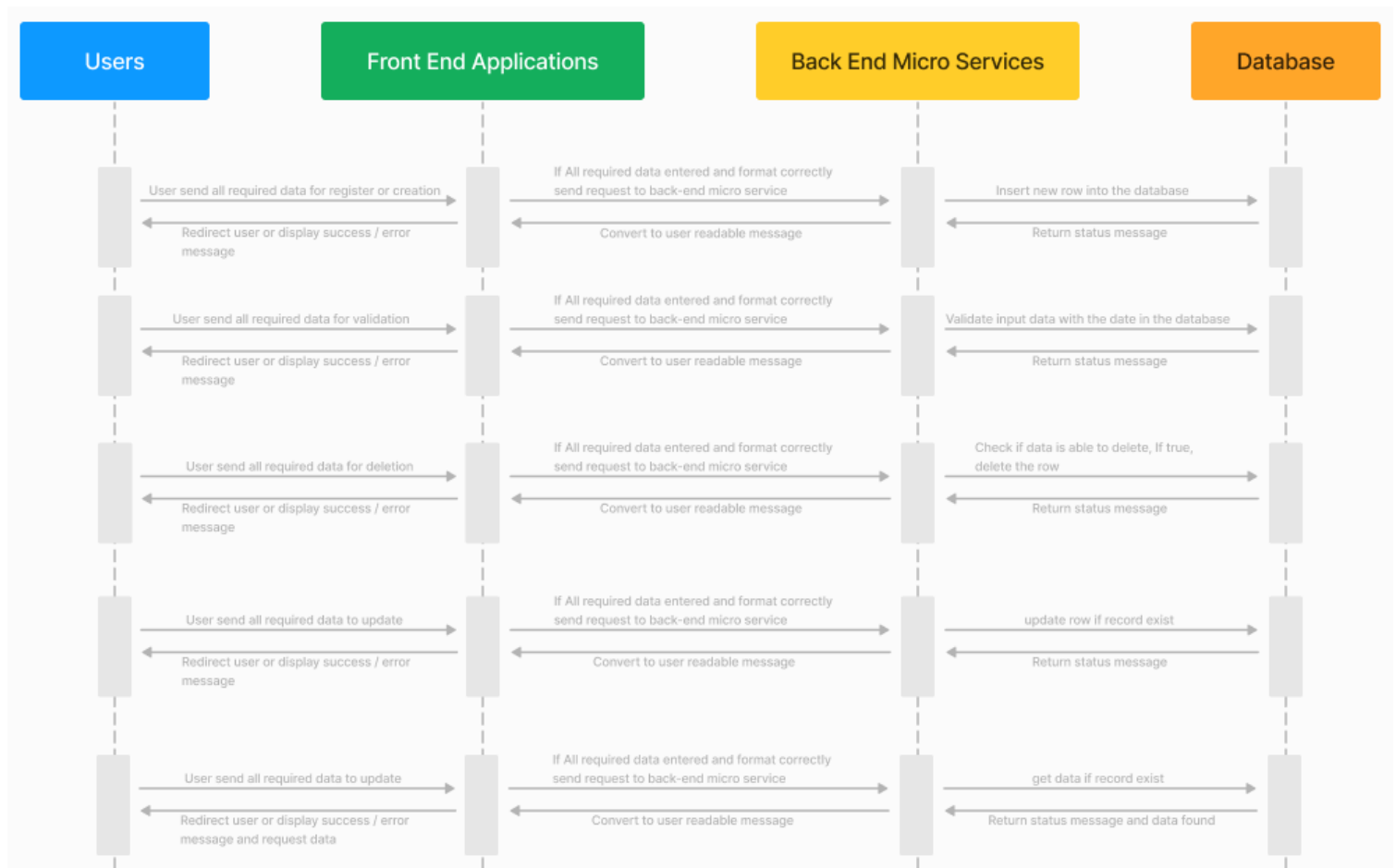


Figure 2.4 illustrates the complex interplay among our users, the front-end application, Authentication microservices, and the database. This delineation elucidates the nuanced flow of the user login procedure.

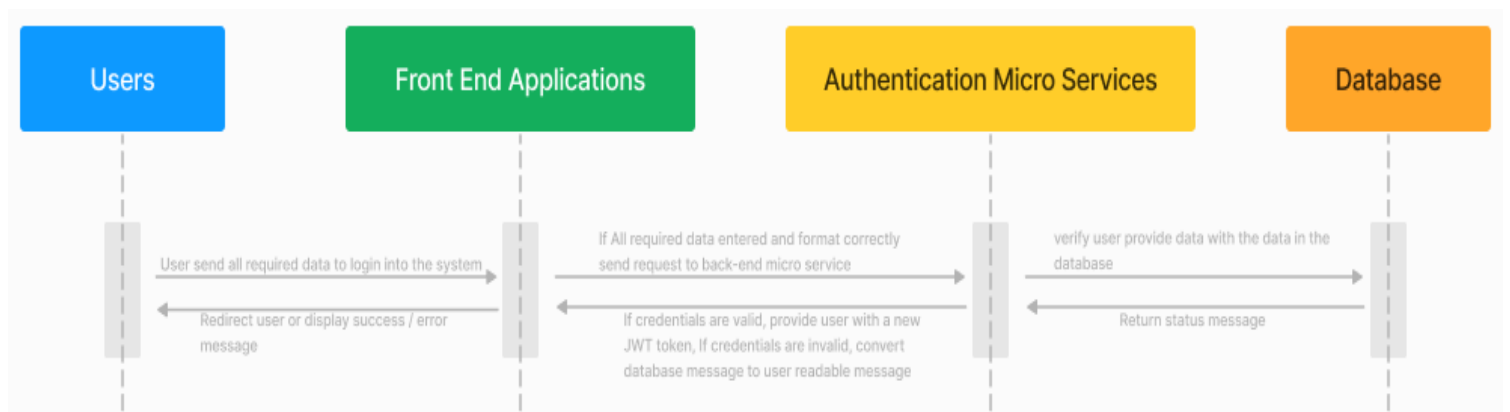


Figure 2.5 delineates the intricate interactions between our users, the front-end application, backend microservices, and the database. This illustration highlights four distinct operational workflows: insertion, validation, deletion, retrieval, and updating of database entries.

17. Technical Specifications

- a. For the client side we utilized the React JS framework to establish the user interface and interactive components of the software. It allowed us to break down complex user interface elements into smaller, reusable components. To fix the issue of separate User interfaces , we choose to have 3 separate react apps (Student , Instructor , Super Admin) , as it was requested that the users cannot navigate between these 3 apps by themselves.
- b. For the client side styling we will be using Tailwind CSS, a utility-first framework that accelerates UI design efficient utilization of predefined classes enabling rapid and cohesive styling across the application.
- c. 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, later if required transformations on those objects will be performed if required . Then be returned as JSON objects by the microservices , with additional status messages.

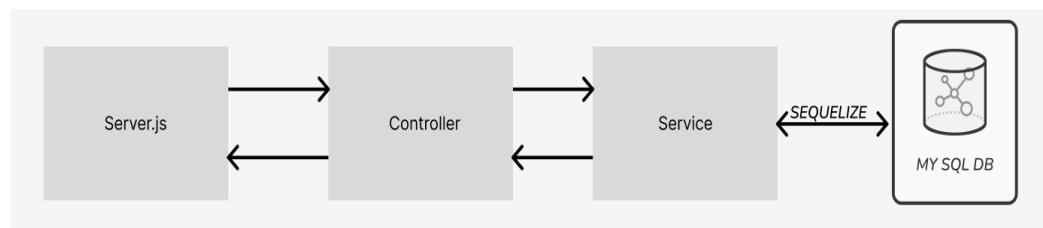


Figure : Sample Backend Microservice Architecture

- 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. To secure our microservices we made use of JWT token authentication using the JWT library.
- For backend unit testing, we utilized Vitest and Super test framework. With Vitest we can systematically evaluate individual units of code in isolation, ensuring all our backend apis operate as intended. Super test allowed us to write automated api tests that can be run alongside our unit tests inside the docker containers. All the microservices have been configured to run the tests , by passing in the command ``npm run tests`` through the docker terminal. And all the tests can be found inside the api_tests and tests repo, inside individual backend microservices folder.

- d. Each individual docker container can be used as a unique testing environment for each of the microservices. To run the tests , in the docker terminal for the microservice (accessible through docker desktop) execute the command “npm run tests”. Which will make use of the vitest framework to automatically run all the tests, and give you a summary of which tests are failing. We can also conclude that each microservice thus has its own testing environment.
- e. The database driver for this application will be MySQL.
- f. Github will be used as a Version Control. We will be using Drone CI for Continuous Integration. We have 3 branches (master, staging_test and prod branch) that will have Continuous Integration set up, any time a new code is merged to these branches, automated tests will run.
- g. Github actions will be used to build the CD pipeline.
- h. Github Projects will be used as a Kanban Board to manage the workflow of building the software.
- i. Docker desktop will be used to run the development environment on the developers local systems.
- j. Postman will be used to manually test backend apis.
- k. Production environment setup through ssh on Digital Ocean droplet provided by the client.

18. UI Mockups

a. Instructor Frontend mockups

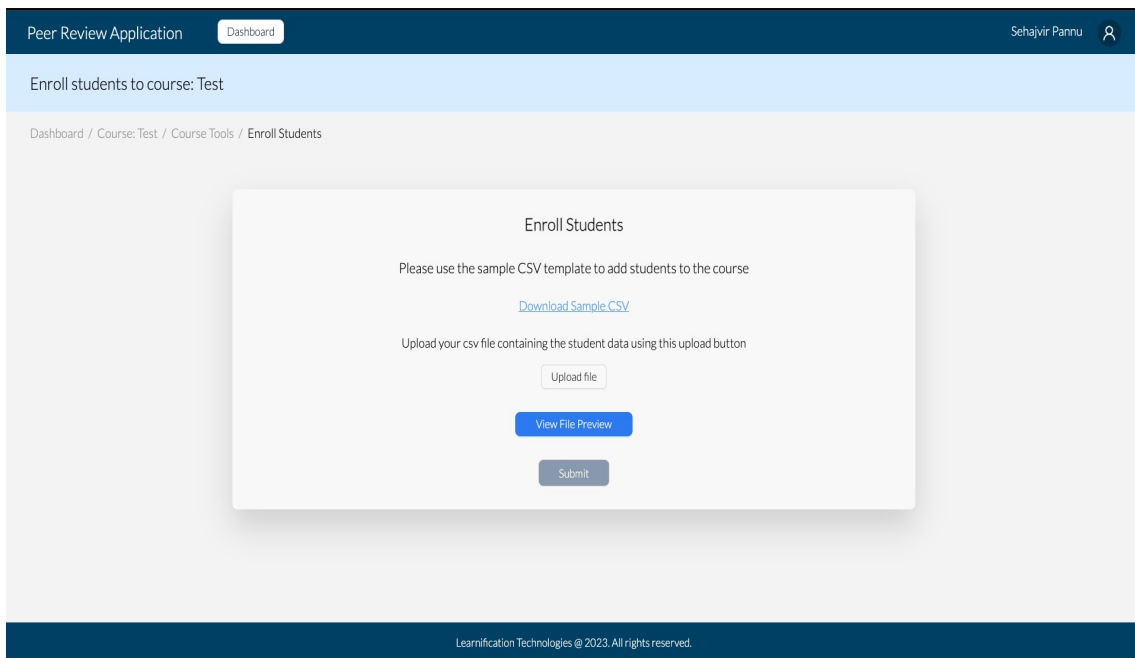
The mockup shows a web application interface for creating a new course. At the top, there is a dark blue header with the text "Peer Review Application" on the left, a "Dashboard" button in the center, and the user name "Sehajvir Pannu" with a profile icon on the right. Below the header is a light blue banner with the text "Create a new Course!". Underneath the banner, the breadcrumb "Dashboard / Create Course" is displayed. The main content area is titled "Create Course" and contains five form fields, each with a red asterisk and a clear icon (⊗):

- * Course Name ⊗: Input field with placeholder text "Enter the course name".
- * Course Code and Section ⊗: Input field with placeholder text "Enter the course code, example COSC499 001".
- * Course Semester ⊗: Input field with placeholder text "Enter the course semester, Example Fall".
- * Course Year ⊗: Input field with placeholder text "Enter the course year, Example 2023".
- * Course Term ⊗: Input field with placeholder text "Enter the course term, Example Jan-April(Term - 1)".

Below the form fields is a blue button labeled "Create Course". At the bottom of the page, there is a dark blue footer with the text "Learnification Technologies @ 2023. All rights reserved."

Figure 3.1 This mockup depicts the create course functionality of the instructor application. The instructor has the ability to create a new course by providing course name, course code

and section, course semester, course year and course term



Peer Review Application Dashboard

Sehajvir Pannu

Enroll students to course: Test

Dashboard / Course: Test / Course Tools / Enroll Students

Enroll Students

Please use the sample CSV template to add students to the course

[Download Sample CSV](#)

Upload your csv file containing the student data using this upload button

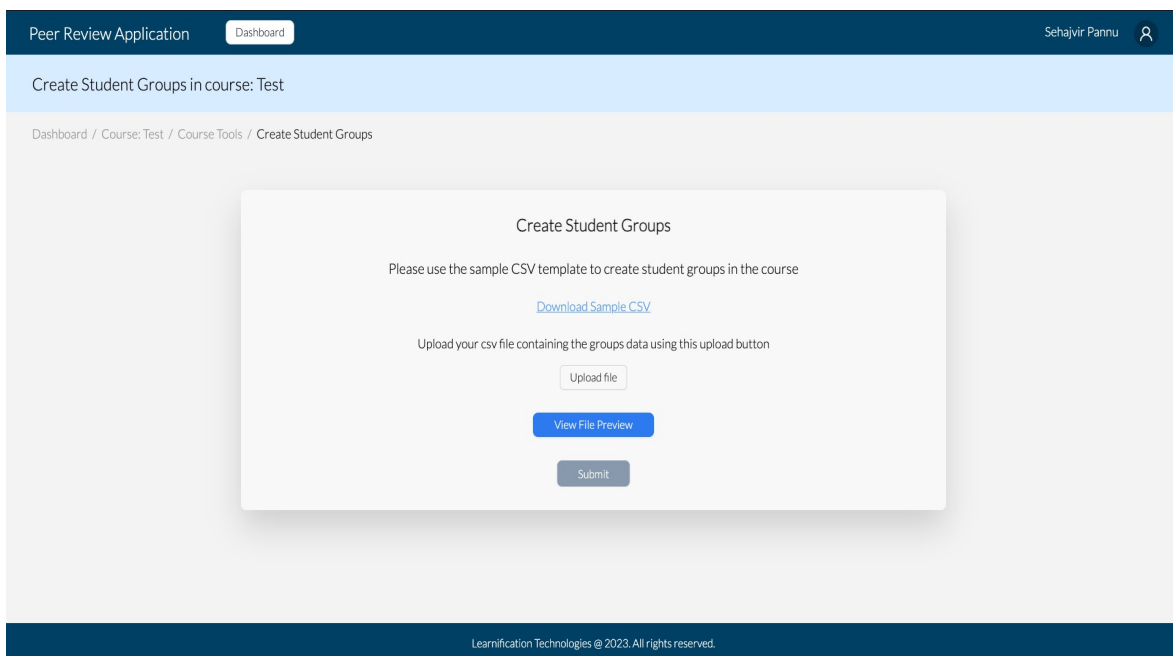
Upload file

View File Preview

Submit

Learnification Technologies @ 2023. All rights reserved.

Figure 3.2 This mockup depicts the enroll students functionality of the instructor application. The instructor is provided with the sample CSV file in which they can include student ids and upload to the application to enroll students into a course.



Peer Review Application Dashboard

Sehajvir Pannu

Create Student Groups in course: Test

Dashboard / Course: Test / Course Tools / Create Student Groups

Create Student Groups

Please use the sample CSV template to create student groups in the course

[Download Sample CSV](#)

Upload your csv file containing the groups data using this upload button

Upload file

View File Preview

Submit

Learnification Technologies @ 2023. All rights reserved.

Figure 3.3 This mockup depicts the add groups to the course functionality of the instructor application. The instructor is provided with the sample CSV file in which they can include group names and the student ids for corresponding students in the group and upload to the application to create student groups in a course.

Peer Review Application Dashboard

Sehajvir Pannu

Test

Dashboard / Course: Test / Student Groups

Students

Groups

Evaluation Forms

Assignments List

Groups Information

Search through the table using the Group name

+ Student Groups + Empty Group

Group Name	Expand All Groups
Apple Database and Image Analysis(1)	[Edit] [Delete]
Automating Database Question Generation and Marking (1) - A	[Edit] [Delete]
Automating Database Question Generation and Marking (1) - B	[Edit] [Delete]
Gamified Coding Practice Platform - Learnification Technologies (1)	[Edit] [Delete]
Love for Vernon Community Calendar (1)	[Edit] [Delete]
Manpower/scheduling Board - Horizon Electric (1)	[Edit] [Delete]
Peer Review Application - Learnification Technologies (1)	[Edit] [Delete]

Figure 3.4 This mockup depicts the student groups in the instructor application. The instructor has the ability to edit/delete student groups, search through the list of the student groups, and add empty student groups to the course.

Peer Review Application Dashboard

Sehajvir Pannu

Create Evaluation Form

Dashboard / Course: Test / Course Settings / Create Evaluation Form

Create Evaluation Form

Instructions for a successful Evaluation Form Creation:

- Evaluation Form Name should be unique.
- Each Evaluation Form should have a deadline.
- Each Evaluation Form should have at least 1 Section.
- Each section should have at least 1 question.
- The total weightage of all the sections should be 100%.
- Each Question should have a question text.
- Questions that have a question point(s) should have a numeric value (could be a decimal).
- Each Multiple Choice Question should have at least 2 options.

Form Name:

Deadline (PDT PST):

Section A

Section Name:

Section Weightage (P):

Learnification Technologies © 2023. All rights reserved.

Figure 3.5 This mockup depicts the create evaluation form functionality of the instructor application. The instructor has the ability to create an evaluation form by providing form name, form deadline, section names, and has the ability to add the questions to the sections.

Peer Review Application | Dashboard

Sehajvir Pannu

Dashboard / Course: Test / Course Settings / Create Assignment

Create Assignment

Assignment Name: *

Enter Assignment Name

Description: *

Enter Assignment Description

Submission Type: *

Select Submission Type

Number of Assessments per Student: *

Available From: *

August 16, 2023 23:59

Deadline: *

August 16, 2023 23:59

Available Until: *

August 16, 2023 23:59

Criteria	Rating	Points	Max Points
Enter Criteria	Enter Rating	0 + -	0

Add Criteria

Submit

Learnification Technologies © 2023. All rights reserved.

Figure 3.6 This mockup depicts the create assignment functionality of the instructor application. The instructor has the ability to create an assignment and assignment assessment by providing assignment name, assignment deadline, assignment available from date, assignment available until date, number of assessments per student for an assignment, and provide rubric for that assignment (criteria, criteria ratings, and rating points)

Peer Review Application | Dashboard

Sehajvir Pannu

Test: sqfcdac Evaluation Overview

Dashboard / Course: Test / Evaluation Forms / sqfcdac Evaluation Overview

Select a Group to view Group Evaluation Data: Peer Review Application - Learnification Technologies (1)

Group Evaluation Overview

Search the table using Evaluatee Name

Evaluatee	Evaluator(s)	View Grade
Charlotte Zhang	Lance X, Prabhmeet D, Sehajvir P, Shila R	View Final Grade
Lance Xu	Charlotte Z, Prabhmeet D, Sehajvir P, Shila R	View Final Grade
Prabhmeet Deol	Charlotte Z, Lance X, Sehajvir P, Shila R	View Final Grade
Sehajvir Pannu	Charlotte Z, Lance X, Prabhmeet D, Shila R	View Final Grade
Shila Rahman	Charlotte Z, Lance X, Prabhmeet D, Sehajvir P	View Final Grade

Learnification Technologies © 2023. All rights reserved.

Figure 3.7 This mockup depicts the group evaluation overview page of the instructor application. The instructor has the ability to select a group from the dropdown menu. On selecting a group, the instructor will get an evaluation overview where all the evaluatees and their respective evaluators will be listed. The instructor can select an evaluator and the feedback they provided for their respective evaluatee. The instructor also has the ability to view/edit the final grade of an evaluatee by selecting the “View Final Grade” link.

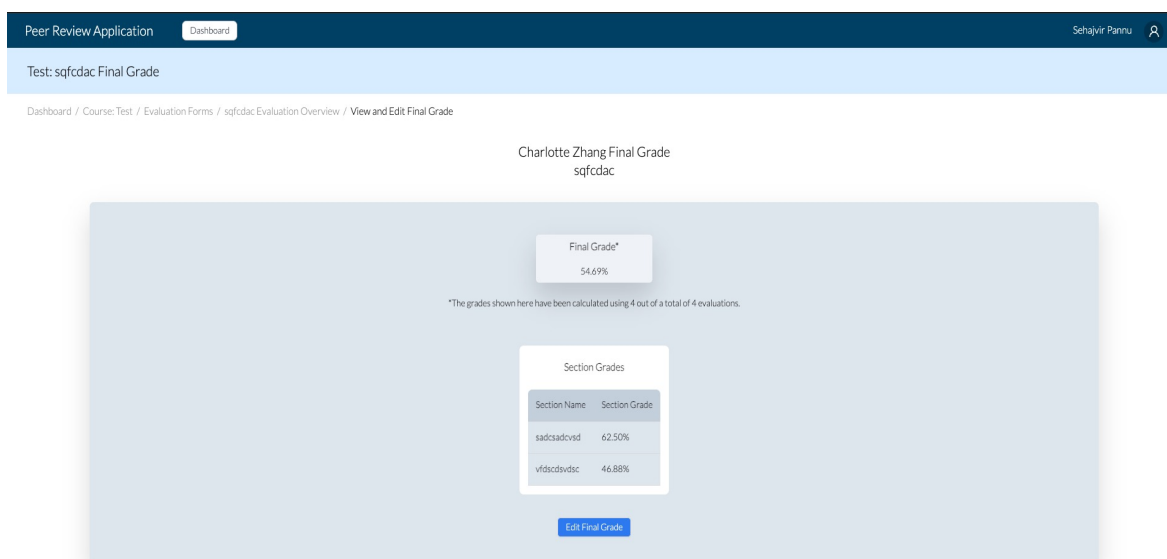


Figure 3.8 This mockup depicts the view and edit grade page of the instructor application for group evaluations. The instructor has the ability to see the final grade for an evaluatee for the group evaluation form and has the ability to edit the final grade once all the evaluations have been completed for an evaluatee

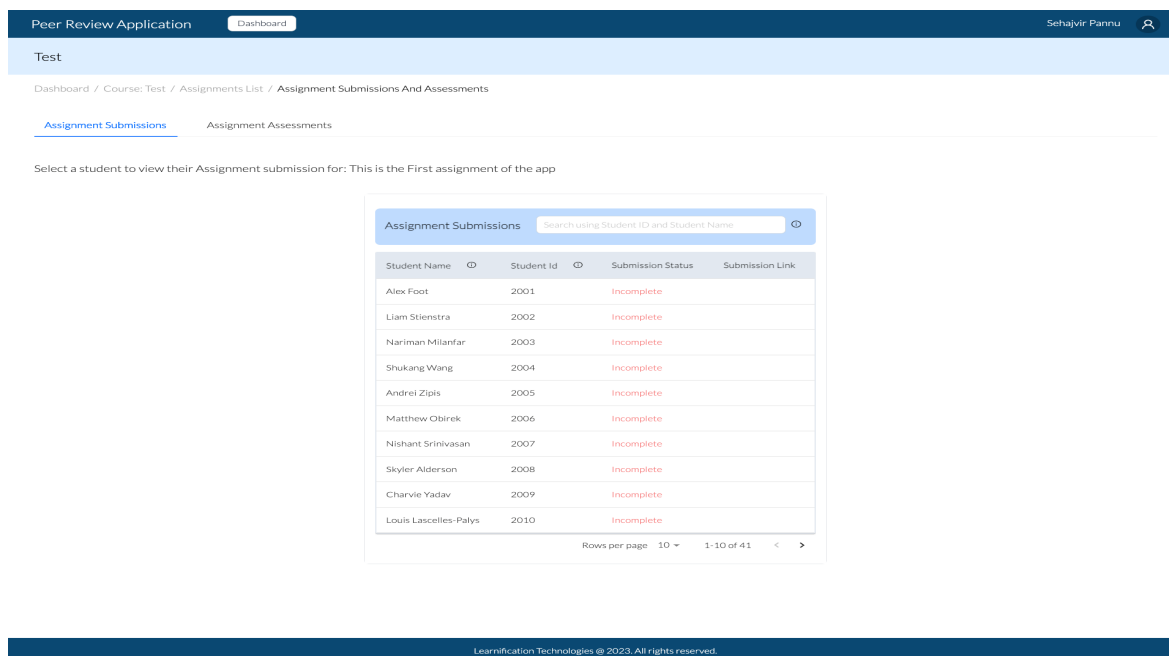


Figure 3.9 This mockup depicts the assignments submission page of all the students in the instructor application. The instructor has the ability to see the assignment submission for a student.

The mockup shows a web application interface for a Peer Review Application. At the top, there is a dark blue header bar with the text "Peer Review Application" on the left, a "Dashboard" button in the center, and the user name "Sehajvir Pannu" with a profile icon on the right. Below the header is a light blue banner with the word "Test". Underneath the banner is a breadcrumb trail: "Dashboard / Course: Test / Assignments List / Assignment Submissions And Assessments". Below the breadcrumb trail are two tabs: "Assignment Submissions" and "Assignment Assessments", with the latter being the active tab. The main content area starts with the text "Select a Student to view their Assignment Assessments:". To the right of this text is a search input field with a dropdown arrow. The input field contains the text "se", and a dropdown menu is open below it, showing the name "Sehajvir Pannu" as a selectable option.

Peer Review Application Dashboard Sehajvir Pannu

Test

Dashboard / Course: Test / Assignments List / Assignment Submissions And Assessments

Assignment Submissions Assignment Assessments

Select a Student to view their Assignment Assessments:

se

Sehajvir Pannu

Figure 3.10 This mockup depicts the assignment assessment functionality of the instructor application. The instructor has the ability to select a student and view the assignment assessments performed by the assesses for the student for their assignment.

Peer Review Application
Dashboard
Sehajvir Pannu

Test

Dashboard / Course: Test / Assignments / Assignment Assessments: This is the First assignment of the app

Assignment Feedback for Sehajvir Pannu by Chad Lantz

This is the First assignment of the app

Assignment Description

This is assignment 1 description. This is assignment 1 description. This is assignment 1 description. This is assignment 1 description

Criteria	Ratings	Maximum Points
This is assignment 1 criteria 1	This is assignment 1 criteria 1 rating 1	3
	This is assignment 1 criteria 1 rating 2	2
	This is assignment 1 criteria 1 rating 3	1
This is assignment 1 criteria 2	This is assignment 1 criteria 2 rating 1	5
	This is assignment 1 criteria 2 rating 2	3
	This is assignment 1 criteria 2 rating 3	0

Submission Link: [View Assignment](#)

Assessment:

1) This is assignment 1 criteria 1 (Max Points: 3)

☐ 3
☒ 2
☐ 1

This is the feedback student must provide if they are not giving full marks on a question

2) This is assignment 1 criteria 2 (Max Points: 5)

☒ 5
☐ 3
☐ 0

[Grade Overview](#)

Final Grade: 87.50%

Assessment Grades

This is assignment 1 criteria 1:	2 / 3 (66.67%)
This is assignment 1 criteria 2:	5 / 5 (100.00%)

Learnification Technologies © 2023. All rights reserved.

Figure 3.11 This mockup depicts the assignment assessment feedback functionality of the instructor application. The instructor has the ability to view the assignment assessments feedback performed by the assesses for the student for their assignment.

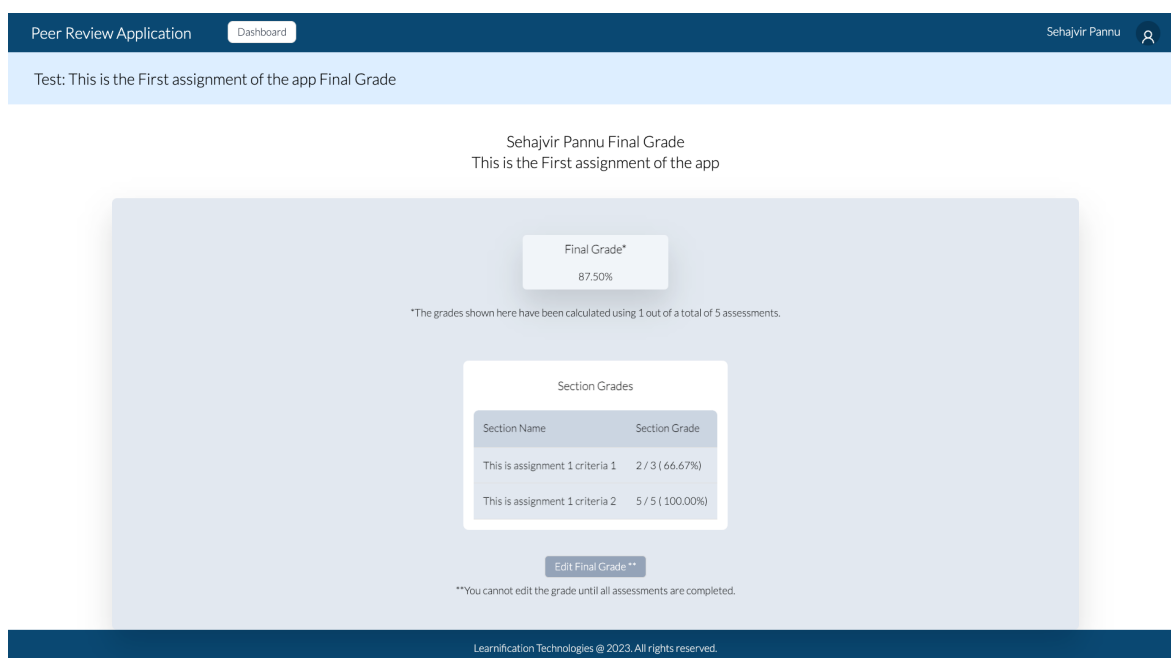


Figure 3.12 This mockup depicts the assignment view and edit grade functionality of the instructor application. The instructor has the ability to view the final grade for an assignment and edit the final grade once all the assignment assessments have been completed.

b. Student Frontend mockups

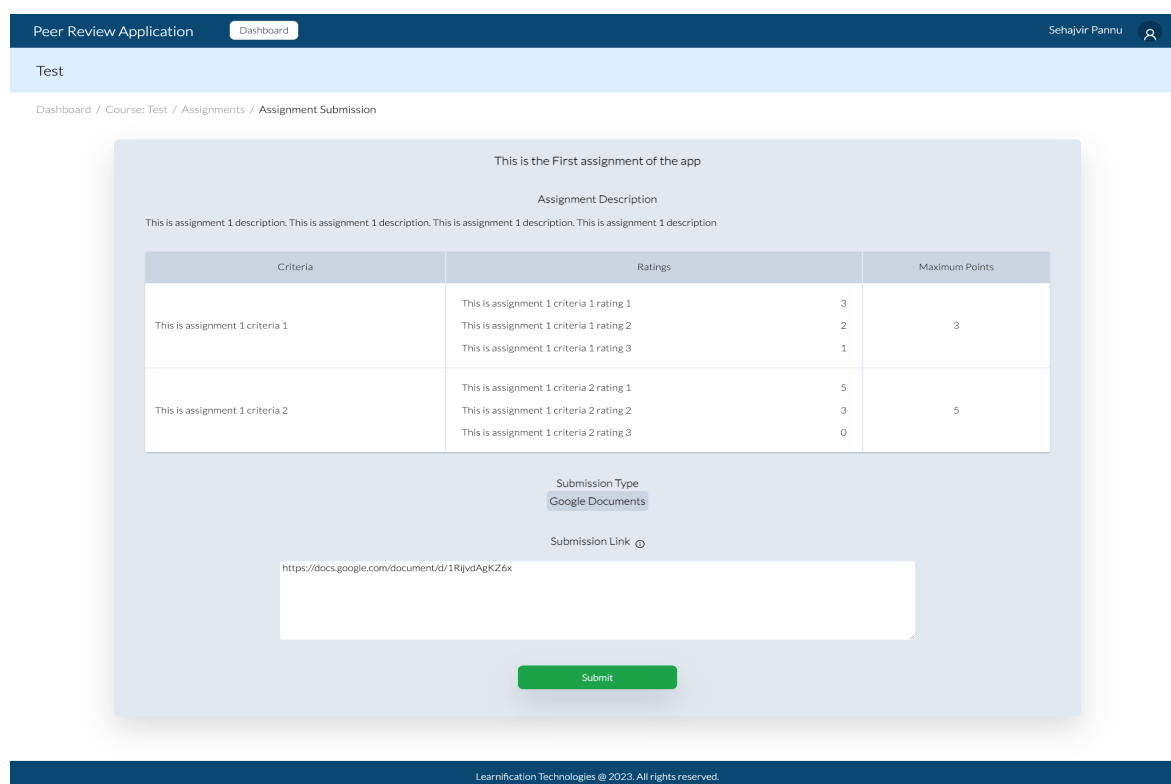


Figure 3.13 This mockup depicts the assignment submission functionality of the student application. the student has the ability to view the assignment details and submit the valid link for their assignment.

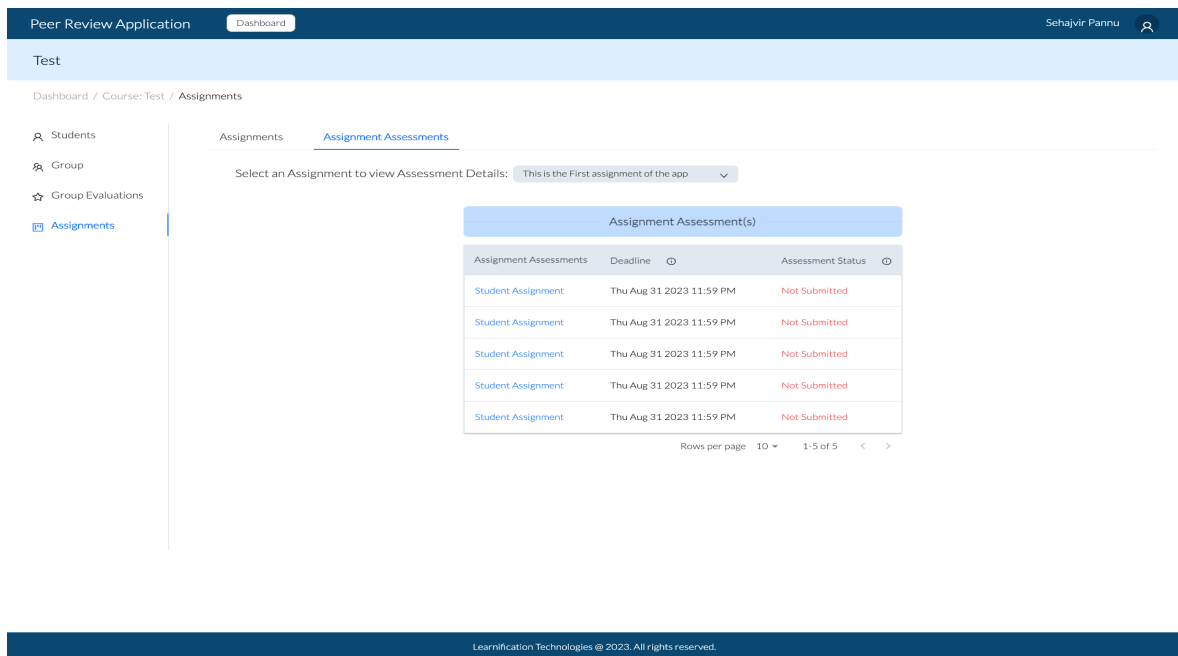


Figure 3.14 This mockup depicts the assignment assessment functionality of the student application. the student chooses an assignment from the dropdown and will be given a list of assignment assessments to perform.

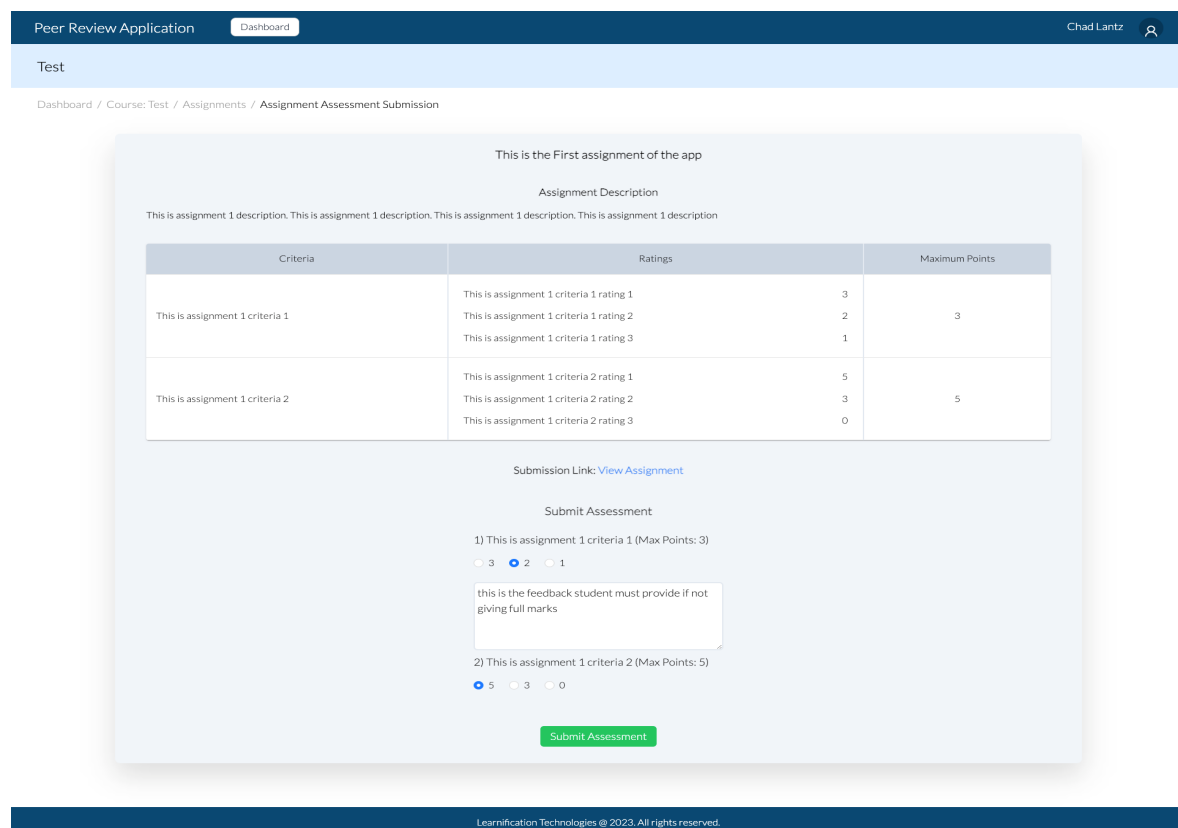


Figure 3.15 This mockup depicts the assignment assessment functionality of the student application. the student performs the assignment assessment for the selected anonymous student. they will be provided the submission link for the anonymous student along with the assignment details.

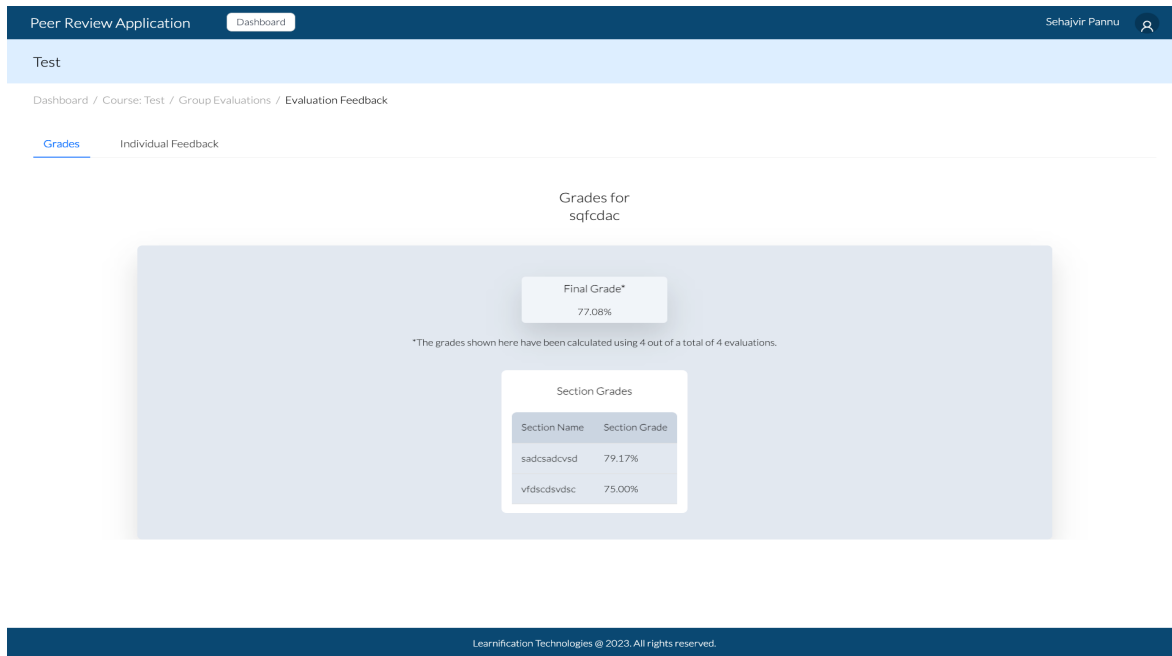


Figure 3.16 This mockup depicts the assignment assessment functionality of the student application. the student performs the assignment assessment for the selected anonymous student. they will be provided the submission link for the anonymous student along with the assignment details.

We have included the mock ups for our major functionalities, since we have so many functionalities we did not include the mockups for all the functionalities.

19. Project Management Methodology, Workflow and WBS/FBS

1. Methodology: Kanban

The peer evaluation application project will be implemented using a Kanban methodology. Kanban focuses on visualizing and optimizing the flow of work, promoting a smooth and efficient project execution. The project will utilize a Kanban board to manage tasks, collaborate effectively, and continuously improve the workflow.

2. Workflow

1. Product Backlog Refinement:

- Collaborate with stakeholders to identify and prioritize features and user stories.
- Break down user stories into smaller tasks or requirements and estimate

their effort.

- c. Continuously refine and update the product backlog.

2. Task Management

- a. Organized and Visualized user stories and tasks on a Kanban board with columns such as "To Do," "In Progress," "Ready for Testing," and "Done."
- b. Pull tasks into the appropriate columns as work progresses.

3. Continuous Development:

- a. Create a new branch for each task or feature.
- b. Implement the functionality based on the task requirements and test cases.
- c. Conduct code reviews and seek feedback from team members for quality assurance

4. Continuous Delivery:

- a. Perform thorough testing of the implemented functionality to verify its behavior and ensure it meets the requirements.
- b. Once a task is completed and thoroughly tested, address a pull request and merge the branch into the main development branch if code review reveals no problems.
- c. Deploy the updated code to the staging or production environment for evaluation once every 2 weeks.

5. Ongoing Collaboration:

- a. Hold regular meetings every Monday, Wednesday and Friday at 5:00 pm to review the Kanban board, discuss progress, and address any challenges.
- b. Foster effective communication and collaboration among team members.
- c. Continuously monitor and update the Kanban board to reflect the current status of tasks.

6. Continuous Integration:

- a. Reflect on the project's progress and performance during retrospectives.
- b. Identify areas for improvement and formulate strategies for enhancing the workflow.

3. Work Breakdown Structure

1. Summary of all the Tasks and the hours clocked

Task List	Actual Hours/Estimated Hours (Expected 18 hours per week from each team member: 216 hours total)					Total Hours
	Prabhmeet	Sehajvir	Shila	Lance	Charlotte	
	204.2/216	575.6/216	116.9/216	254/216	202.5/216	1,354.2/1080
Milestone						
PROJECT CONFIGURATION						
Docker Setup	6/4.5					4.5
Add CORS support	5/2	2.3/2				7.3
Nginx Configuration	0.5/1					0.5
Prettier Configuration	2.2/1					2.2
CICD Configuration	32.5/24					32.5
DroneCI Integration	1.1/2					1.1
Project Configuration	5.5/8					5.5
JWT token	24.5/6	12.3/4		4.5/6		41.3
Update Database				3.5/3		3.5
Redux		3.6/6				3.6
Env variable Configuration		5.6/4				5.6
DOCUMENTATION						
Client Meeting Documentation		12.4/16			3.5/2	15.9
Weekly Logs documentation		2.8/4		1.5/4		4.3
Update user requirements		2.6/4	1.5/2		11.5/6	15.6
Frontend ticket documentation		4.3/5		12.6/14		16.9

Backend ticket documentation				13.8/12		13.8
Test report about functional requirements					7.5/6	7.5
MVP upload requirements		3.5/4			11/12	14.5
Test report			7.8/9	6.4/4	6.2/2	20.4
Final Report	11/8	22.3/12		8.5/12	2.8/4	58.6
User experience questionnaire		2.3/1			8/6	10.3
Final presentation		6.5/8	3.6/5		10.5/8	20.6
Sequence Diagram					5.3/4	5.3
Workflow Diagram			2.2/2			2.2
FRONTEND IMPLEMENTATION						
Instructor Sign up page		3/1				3
Instructor Register page		2.5/1				2.5
Instructor Dashboard page		11.5/8				11.5
Instructor update Profile page					16.7/1	16.7
Instructor update Password page					28.4/1	28.4
Instructor course setting page		0.5/2	19.5/4			20
Instructor create course page		0.5/1	16.2/1			16.7
Instructor create form page		8.2/6				8.2
Student Register page		11.5/6				11.5
Instructor add student to course page		7.4/5				7.4

Instructor course preview page		3.6/3				3.6
Instructor add groups to course page		4.2/6				4.2
Instructor course group list page		9.6/10				9.6
Instructor update profile page			1.6/1		13.8/2	15.4
Instructor course list page		7.5/4				7.5
Student profile page		1/2			6/2 6	
Instructor user context		5/6				5
Instructor create evaluation form page		25.8/28				25.8
Instructor group evaluation overview page		1.8/2				1.8
Instructor view course form list page		2.5/2				2.5
Instructor create assignment page		2.5/3	36.8/8			39.3
Student login page					16.2/4	16.2
Instructor set course visibility		2/3				2
Instructor set form visibility		3.5/3				3.5
Instructor set evaluation feedback		2.8/3				2.8
Instructor get individual feedback page		4.7/6				4.7
Instructor edit form page		14.5/12				14.5
Instructor group evaluation overview page		7/6				7

Admin login page		0.9/1			3.8/1	4.7
Instructor edit grade page		16.5/18				16.5
Instructor get evaluation complete status list page		3.8/4				3.8
Student get course form list page		1.4/2				1.4
Student take evaluation page		17.5/12				17.5
Instructor delete form page		0.5/1				0.5
Student get individual grade page		2.5/3				2.5
Student get overall grade page		1.8/3				1.8
Student get individual feedback page		3.5/4				3.5
Admin dashboard page		2.2/1				2.2
Student submit assignment page		4/6			18/6	22
Student get course assignment list page		5/6			8.5/4	13.5
Forgot password page		7/2				7
Instructor Landing page			1.5/1			1.5
Instructor view assignment assessment page		2.5/3				2.5
Student assignment submissions table		2.5/3				2.5
Student assignment assessment list page		2.8/4				2.8

Student view assessment overall grade		4.5/6				4.5
Instructor view and edit assessment final grade		4.5/6				4.5
Instructor assignment assessment page		2.5/4				2.5
Student assignment assessment overview page		2.8/4				2.8
Instructor get student submission list page		4.6/5				4.6
Student view assessment final grade		5.3/5				5.3
Student submit assignment assessment page		4.9/3				4.9
Instructor get assignment list page		4/6	3.9/2			7.9
Instructor view assignment assessment overall grade page		4.7/5				4.7
Instructor export evaluation feedback to csv file		3.2/2				3.2
USER INTERFACE						
UI Mockup		3.5/6	8.3/5	0.4/2		12.2
Front End styling		19.7/12	6.7/4			26.4
BACKEND IMPLEMENTATION						
Defining Sequelize Models		1.2/1				1.2

Token Authentication API	10.5/6					10.5
Student Register API				1.7/2		1.7
Authenticate Instructor API	0.5/1	1.8/2		1.2/2		3.5
Authenticate Student API	3/4					3
Get Instructor Profile API	0.7/2			1.5/1		2.2
Get Student Profile API	0.7/1					0.7
Get Instructor ID API		2.3/2				2.3
Instructor add students to course API	0.5/1			3/4		3.5
Instructor add students to course preview API				3.6/2		3.6
Get course student list API	4/3	2.1/2		3.5/3		9.6
Get course assignment list API				1.2/4		1.2
Get course form list API				3.7/3		3.7
Get group student list API		2.5/4				2.5
Instructor create course API				1.7/2		1.7
Instructor create group API				2.8/3		2.8
Instructor get course list API				2.9/2		2.9
Instructor get course group list API				3.6/3		3.6
Instructor update profile API				2.3/4		2.3
Instructor create form API		3.8/2		11.5/16		15.3
Get form API				1.7/3		1.7

Instructor add groups to course API	12.5/2			4.9/2		17.4
Instructor add student to group API				1.9/3		1.9
Instructor set course visibility API				2/1		2
Instructor get assignment number API				1.7/2		1.7
Instructor remove student from group API				1.1/2		1.1
Instructor get form number API				2.2/2		2.2
Instructor delete empty group API				1/2		1
Instructor remove student and group API				1.3/2		1.3
Student update profile API	0.8/1					0.8
Student get profile API	0.1/1					0.1
Instructor get form API				2.5/3		2.5
Instructor edit form API				5.8/6		5.8
Instructor get assignment list API	1/0.5					1
Admin get instructor access API	10.5/6					10.5
Student get evaluation status API						
Student get course list API	2.2/1					2.2
Student get self group id	0.2/1					0.2
Student update password	2/0.5					2

Student get evaluation form record API				1.3/2		1.3
Instructor get overall evaluation grade API				1.8/1		1.8
Instructor get individual evaluation grade API				4.8/3		4.8
Instructor get group member evaluation status API				1.2/2		1.2
Instructor get evaluation API				2.4/2		2.4
Instructor set form feedback API				2.5/2		2.5
Admin get instructor list API	0.8/1					0.8
Admin signup API	2.5/1					2.5
Student submit group evaluation API				1.5/2		1.5
Student get evaluation overall Grade API				2.8/1		2.8
Student submit assignment API	3/6					3
Student get assignment list API				1/2		1
Instructor create assignment API				2.2/3		2.2
Instructor create assignment rubric API				1.8/2		1.8
Instructor edit form grade API				0.9/3		0.9
Instructor release assignment API	2/1					2
Forgot password API				7.4/6		7.4
Student get assignment list API				1.8/2		1.8

Instructor set assignment visibility API				1.8/2		1.8
Instructor get assignment overall grade API				2.9/5		2.9
Instructor get assignment and rubric				1.9/3		1.9
Instructor view assignment assessment API				3.1/3		3.1
Notification system				2/1		2
Instructor get assignment complete status API				1/2		1
Instructor set assignment visibility API				0.7/2		0.7
Student view assignment rubric				0.9/2		0.9
Instructor get assignment submission link API				2.3/2		2.3
Instructor get assignment rubric				0.7/1		0.7
Student get assignment assessment API				0.1/2		0.1
Instructor get individual assessment grade API				3.3/4		3.3
Instructor delete assignment API				2.6/3		2.6
Instructor edit rubric API	6/4					6
Student get submission link API	1.2/2					1.2
Student get assess group id API	2/1					2

Instructor edit assignment API	1.5/1					1.5
Student get assignment and rubric API				0.5/2		0.5
Student get Assessment overall grade API				1.5/3		1.5
Student get submission status list API				0.2/1		0.2
Student view assessment individual grade API				0.2/2		0.2
DEBUG						
Docker staging_test issues fixing	0.2/1					0.2
Docker compose issues fixing	1.7/2					1.7
Fixing Sequelize Models		1/2		1.3/1		2.3
Fixing Database entities	2.5/2					2.5
Frontend Debugging		54.5/60				54.5
Backend Debugging				26.3/32		26.3
Deployment Debugging		9.6/4				9.6
TESTING						
Database manual testing		6.4/2				6.4
Backend unit testing				34.7/48		34.7
Frontend unit testing		19.5/24				19.5
Integration Testing	1.8/2					1.8

Usability Testing	2.5/4		3.9/6		4/2	10.4
CLIENT						
Prepare Client meeting demo		8.9/12				8.9
Bridge meeting		12.3/12				12.3
DEPLOYMENT						
Initial Deployment	17.4/10	24.8/12				42.2
Github web hook	2/1					2
Final Deployment		28.3/12				28.3
OTHER						
Research Deployment options through Digital Ocean	2/1					2
Research ReactJS Frontend testing		5.4/8				5.4
Research Sequelize Mock for backend Unit testing				6.2/4		6.2
React learning					12.5/24	12.5
ExpressJS learning				1.9/2		1.9
Research on React JWS		2.8/1				2.8
Research on backend Unit test			1.6/2		3.8/4	5.4
Research on frontend Jest unit testing		3.5/2				3.5
Research on testing strategy	6.5/4					6.5
MVP presentation	8.8/6	5.9/6	1.8/6	3/6	4.5/6	24
Research on Password recovery		1.5/3				1.5
Research on Web Notification system		5.6/8				5.6

Preparing for Testing event	3.8/6	9.7/12				13.5

2. Work Breakdown structure weekly

	Sehajvir	Lance	Prabhmeet	Charlotte	Shila	Actual Total	Estimated Total (18 hr per person per week)
Week 3	20:02:35	4:11:27	Did not start Clockify time logs	5:42:00	Did not start Clockify time logs	29:56:02	90:00:00
Week 4	28:33:35	25:33:42	Did not start Clockify time logs	4:38:41	Did not start Clockify time logs	58:46:15	90:00:00
Week 5	34:51:18	29:23:49	22:03:39	18:31:35	08:46:09	113:36:40	90:00:00
Week 6	38:47:36	15:20:22	23:02:54	18:18:20	21:30:18	116:59:30	90:00:00
Week 7	42:00:17	29:13:01	07:30:42	17:23:32	01:27:54	97:35:26	90:00:00
Week 8	40:29:08	22:28:43	31:59:23	23:27:36	11:11:32	129:36:22	90:00:00
Week 9	53:37:54	25:42:18	18:16:04	17:01:55	19:21:51	134:00:02	90:00:00
Week 10	63:38:14	29:49:27	30:34:22	21:54:40	09:05:10	155:01:53	90:00:00
Week 11	63:17:25	23:07:54	20:20:03	23:49:18	21:51:54	152:26:34	90:00:00
Week 12	79:23:22	30:41:24	23:40:23	31:55:52	10:42:57	176:23:58	90:00:00
Week 13	77:02:33	17:29:29	25:42:24	29:55:29	12:03:38	162:13:33	90:00:00

We estimated 18 hours per person per week which sums up to a total of 90 hours for each week. Development started in week 3. During the initial 2 weeks, the team did not meet the expected 90 hour mark, as we were still in the requirements catching phase and two of our members did not have clockify setup due to which logs are missing.

3. Burnup Charts

[FRONTEND TICKETS]: This category contains all features of the frontend application. Each page is treated as a separate ticket, with each ticket having varying levels of complexity and encompassing one or more features.

[BACKEND TICKETS]: This category contains all features of the backend application. Each API is treated as a separate ticket, with each ticket having varying levels of complexity and encompassing one or more features.

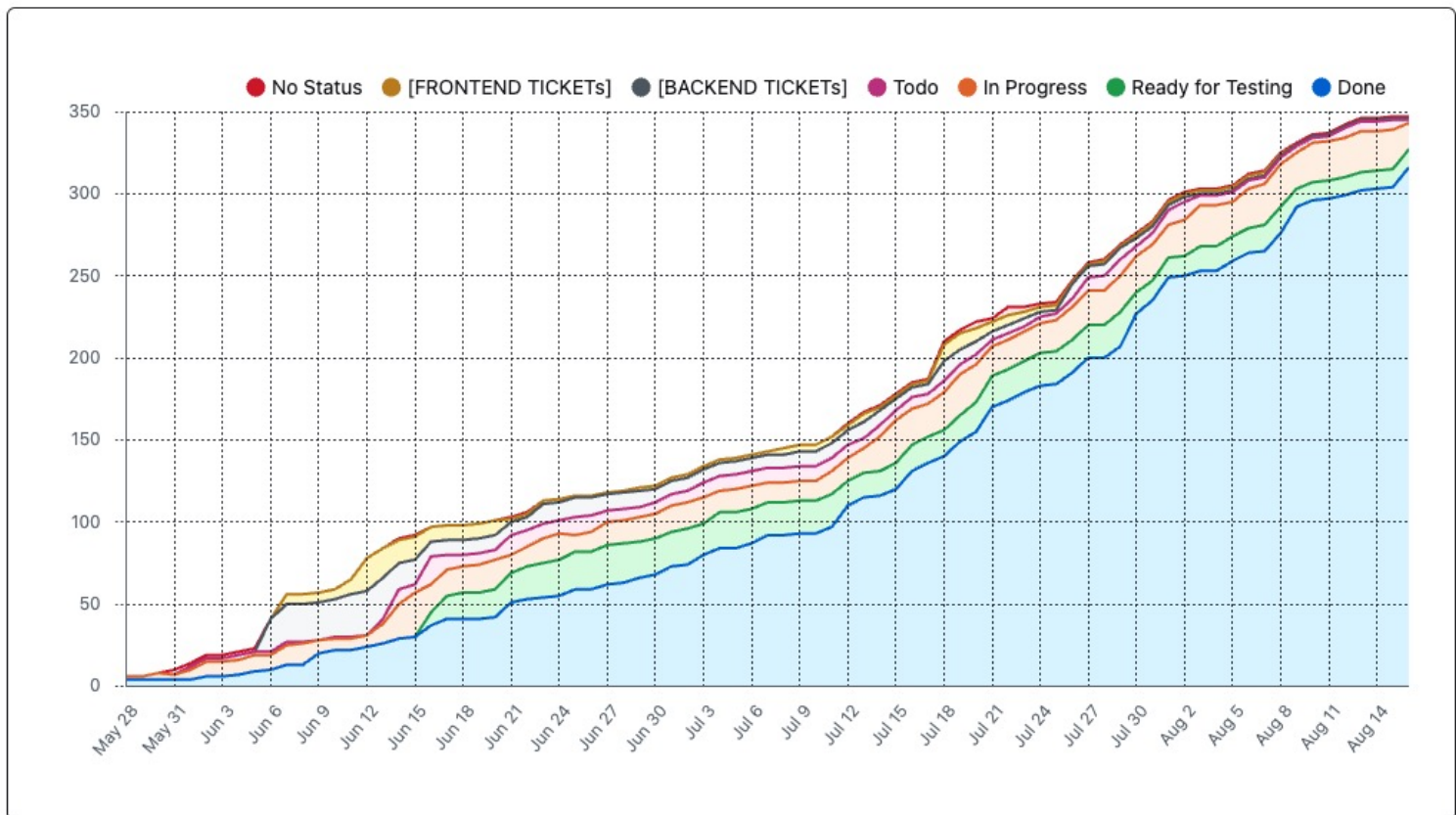


Figure 4.1 Displays progress report of all the Tickets Created, In Progress , and Finished.

4. BurnDown Charts

The reason the “hours remaining” has dropped below zero and reached negative values is because the original projected total work hours for this project was set at 900 hours. However, our actual working hours have far exceeded this projection, reaching around 1300 hours. Given the formula $\text{hours} = \text{estimated hours} - \text{hours completed}$, the curve on the graph dips below zero.

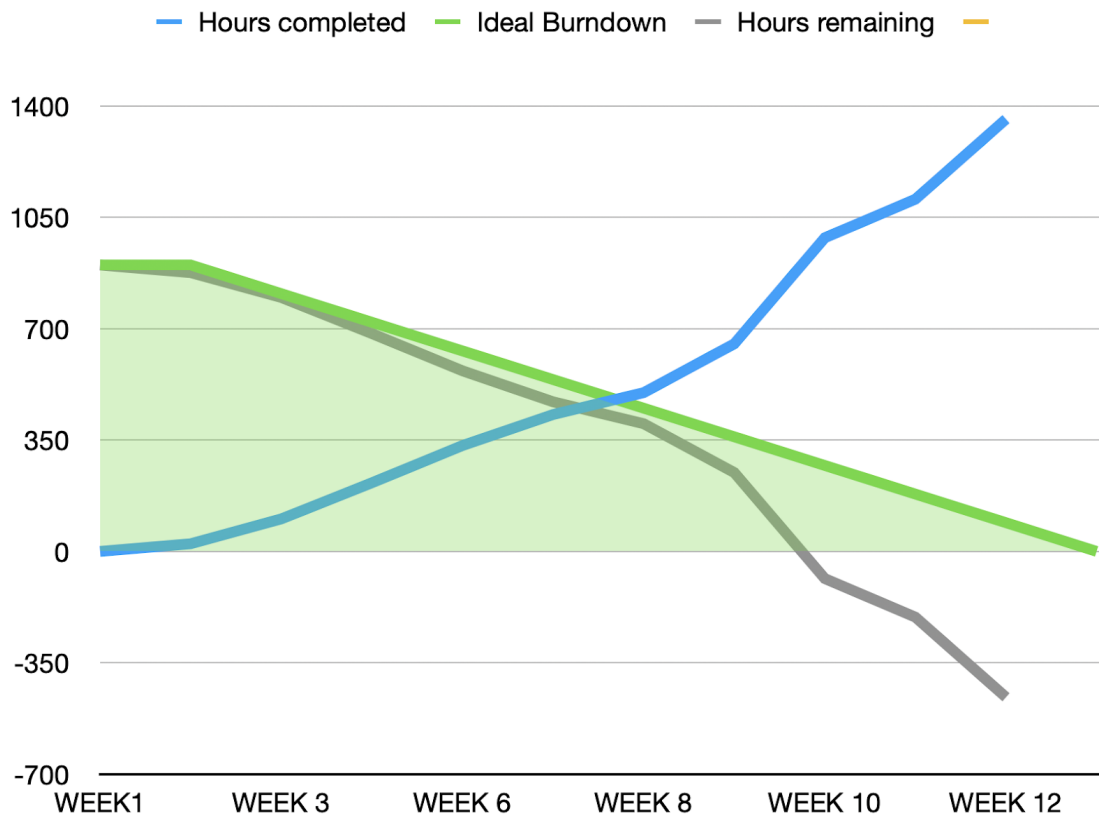


Figure 4.2 : Displays Total number of hours completed , against the number of weeks into the project.

20. Features Completed/Core Requirements Delivered

Features/Requirements	Status
Student Peer Review Application	Status
The system must provide a user registration feature that allows students to create an account and accurately capture and store the following information: first name, middle name, last name, student Id, student email, and password.	Complete
The system must provide a secure login functionality for students to access the platform using their registered email and password, ensuring authentication and verification of credentials.	Complete
The system must allow students to manage their account details on the platform, including their name, email, and password, enabling them to update and modify their personal information securely, ensuring control over their account and maintaining account security.	Complete
The system must provide students with the ability to reset their passwords in the event of forgetting or needing to change them, offering a secure and accessible password recovery process that ensures the privacy and security of user accounts.	Complete
The system must provide students with the ability to log out of the system,	Complete

implementing appropriate security measures to ensure a secure disconnection and protect against unauthorized access.	
The system must provide the students with a list of all the courses they are enrolled in by their instructors, clearly displaying the course details which includes, the course name, course code and section, course semester, course term and course delivery year.	Complete
The system must provide the students with a list of all the student peers enrolled in the course they are enrolled in.	Complete
The system must provide the students with a list of all the group members of their group for a course if the instructor has added them to a group.	Complete
The system must provide students with the ability to view a comprehensive list of assignments due on the platform, displaying relevant details such as assignment name, assignment description, assignment rubric, rating points, assignment available from date, assignment available until date, assignment deadline, accepted submission links, ensuring easy access to assignment information for monitoring and tracking progress.	Complete
The system must allow students to submit assignments only as links (e.g., Google Drive link, Dropbox link, Google Spreadsheet link, Google Document Link, Google Slides link) on the platform, as specified by the instructor when creating student assignments.	Complete
The system must perform link validation to ensure the submitted links for assignments are valid, specifically checking if the URL contains the appropriate format specified by the instructor (e.g., for a Google Docs submission, the URL should contain 'https://docs.google.com/'). This validation process ensures the reliability and integrity of the submitted work.	Complete
The system must provide students with the ability to edit their assignment submission on the platform before the deadline, enabling them to make necessary revisions or updates to their work.	Complete
The system must provide students with a list of 'x' number of other students' assignments for assignment assessments, determined by the instructor when creating the assignment, for the purpose of anonymous peer assignment assessment. The system will randomly assign 'x' number of assignments to each student to assess.	Complete
The system must provide students with peer-assessed assignment feedback and provide the calculated final grade for an assignment based on the average assessment scores for their submitted assignments.	Complete
The system must provide students with a list of Group Evaluation Forms and relevant information like Group Evaluation Form name, Deadline and their Feedback/Grade status for a Group Evaluation form once the instructor releases the Form to the students.	Complete
The system must provide the students with a list of their group members for which they have to provide evaluations for and their corresponding evaluation submission status.	Complete
The system must provide the students with the ability to submit an evaluation for	Complete

their group members through a comprehensive evaluation form created by the instructor that includes multiple choice questions, multiple answer questions, matrix questions, short answer questions. This allows for the collection of both quantitative and qualitative feedback.	
The system must require each student within a group to provide feedback to all other group members, ensuring comprehensive and inclusive evaluation among team members.	Complete
The system must prompt the student with a text input if and only if the student evaluating another student does not give a full mark on a particular question. This ensures the provision of constructive feedback and opportunities for growth. The system should not allow a student to proceed until they have provided a comment feedback.	Complete
The system must provide the students with the ability to view the scores and comments associated with their evaluations, without revealing the identity of the reviewers. This ensures transparency and provides valuable feedback for self-assessment and improvement.	Complete
Instructor Peer Review Application	Status
The system must provide a user registration feature that allows students to create an account and accurately capture and store the following information: instructor first name, instructor middle name, instructor last name, employee Id, instructor email, and password.	Complete
The system must allow instructors to log in securely to the system using their registered email and password, ensuring secure access to their account and associated functionalities.	Complete
The system must provide instructors with the ability to reset their passwords in case of forgetting or needing to change them, ensuring a secure and accessible password recovery process.	Complete
The system must provide instructors with the ability to log out of the system, ensuring a secure disconnection and safeguarding their account from unauthorized access.	Complete
The system must provide instructors with access to a feature that allows them to manage their account details on the platform, ensuring control over their personal information and account settings.	Complete
The system must provide the instructors with the ability to create a new course they are teaching. The system must accurately capture course details such as course name, course code and section, course semester, course delivery year and course term.	Complete
The system must provide instructors with the ability to enroll students to a course by importing their student IDs using a CSV file. The CSV file format for group enrollment should adhere to a specific format defined by the client.	Complete
The system must provide the instructors with a sample CSV file, to which student IDs can be added and subsequently uploaded to the system to enroll students in the	Complete

course.	
The system must allow instructors to create evaluation teams/groups in a course by importing Group names and corresponding student ids of the group members using a CSV file. The CSV file format for group enrollment should adhere to a specific format defined by the client.	Complete
The system must provide instructors with the ability to add or remove students from teams/groups, both during and after the group creation process. This feature ensures the necessary flexibility to manage team/group membership as required.	Complete
The system must provide the instructors with the ability to add empty evaluation groups to a course by asking for a unique group name.	Complete
The system must provide the instructors with the ability to remove/delete student groups from a course at any point of time. Removing a group will remove student-group associations from the system.	Complete
The system must provide instructors with the ability to create a new assignment for a course. System must accurately capture the assignment name, assignment description, assignment rubric, assignment submission type, 'x' number of assessments per student (the system will randomly generate 'x' membered groups for assignment assessment), assignment available from date and time, assignment available until date and time (same as assignment assessment deadline), assignment deadline.	Complete
The system must provide the instructors with the list of student assignments including their assignment name, assignment available from date and time, assignment available until date and time, assignment deadline date and time.	Complete
The system must provide the instructors with the ability to release/hide an assignment to /from the students' course dashboard.	Complete
The system must provide the instructors with the ability to release/hide assignment feedback to/from the students' course dashboard.	Complete
The system must provide the instructors with the ability to delete an assignment from a course.	Complete
The system must provide the instructors with a list of all the students' submissions which will include submission status and submission links if students have successfully submitted an assignment.	Complete
The system must provide the instructors with an overview of the student assignment assessments table. The table will include an evaluatee, their corresponding evaluators and link to the evaluatee final grade/feedback page.	Complete
The system must provide the instructors with the assignment assessment feedback performed by a student evaluator for a student evaluatee.	Complete
The system must provide the instructors with the assignment assessment final grade calculated using the assignment assessments performed by all student evaluators for a student evaluatee.	Complete
The system must provide the instructors with the ability to set and change the assignment assessment final grade for an evaluatee once all the assignment	Complete

assessments are completed for a student evaluatee, ensuring flexibility in grading and accommodating adjustments, including the provision of 'fudge points'.	
The system must provide the instructors with the ability to create evaluation forms and select the questions to include, allowing them to design customized assessment criteria and collect specific feedback based on their evaluation requirements. The system must accurately capture evaluation form name, deadline, section names, section weightages, corresponding section questions, question text, question options if applicable and option points if applicable.	Complete
The system must prevent the instructor from submitting an evaluation form creation request until and unless all the necessary evaluation creation validations have been successfully met.	Complete
The system must provide the instructors with a list of all the evaluation forms they have created along with the following details; evaluation form name, and evaluation form deadline.	Complete
The system must provide the instructors with the ability to edit/update an evaluation form before a student performs an evaluation -- once the evaluation has been performed by a single student, the system should not allow the instructor to edit an evaluation form.	Complete
The system must provide the instructors with the ability to delete an evaluation form before a student performs an evaluation -- once the evaluation has been performed by a single student, the system should not allow the instructor to delete an evaluation form.	Complete
The system must provide the instructors with the ability to publish/unpublish an evaluation form, providing control over the visibility and accessibility of evaluation forms for students	Complete
The system must provide the instructors with the ability to publish/unpublish evaluation feedback for an evaluation form, providing control over the feedback data and accessibility of evaluations for students.	Complete
The system must provide the instructors with the ability to select an evaluation form and then select a student group and provide the instructor with an evaluation overview for that group. The overview includes student evaluatees, corresponding student evaluators and a link to view the final grade for the student evaluatee.	Complete
The system must provide the instructors with the ability to choose an evaluator for an evaluatee and view the evaluation feedback provided by the evaluator for the evaluatee.	Complete
The system must provide the instructors with the ability to export individual evaluation feedback data for a student evaluatee performed by a student evaluator in a CSV file format, allowing for convenient analysis, storage, and further manipulation of the data as needed.	Complete
The system must provide the instructors with the ability to view the final grade for an evaluation form for an evaluatee calculated using all evaluation data for the evaluatee.	Complete
The system must provide the instructors with the ability to set and change the	Complete

evaluation form final grade for an evaluatee once all the evaluations have been performed for an evaluatee by the evaluators of their corresponding student evaluation group, ensuring flexibility in grading and accommodating adjustments, including the provision of 'fudge points'.	
Super Admin Peer Review Application	Status
The system must have a super admin account that is created using a postman call.	Complete
The system must provide a secure login functionality for the super admin to access the platform using the email and password created while setting up the super admin account using postman, ensuring authentication and verification of credentials.	Complete
The system must provide the super admin with a list of all the instructors within the system.	Complete
The system must provide the super admin the ability to grant/revoke instructor access for instructors within the system.	Complete
System Functional Requirements	Status
The system must calculate the final grade for a group evaluation form from the quantifiable data derived from the feedback, allowing for allocation of specific weightage to each question as set by the instructor, ensuring accurate assessment of scores based on predetermined weightings.	Complete
The system must deliver notifications through email when an evaluation form or an assignment is released. The system must also deliver notifications through email when feedback for an evaluation form or an assignment assessment is released.	Complete

21. Testing Plan/QA Plan

a. Goal

The System has been tested at multiple points of development , to ensure that the components previously built are still working accordingly with integration of new components . This was done by Automated unit and api testing (about 397 automated test cases being run) on a standalone drone server. Developers also performed manual testing on the whole system throughout development , to ensure new features being developed on the frontend carried out the right business logic and met clients Expectations. A user usability testing survey (blackbox testing) has also been done, which gave us insights regarding if our system was easy to use and understand. This survey also tested the business logic (functional requirements) of our app, by normal users. The most important testing goal is to make sure that the product being delivered is in alignment with the Clients Expectations, data is being stored correctly and as it is expected to be stored , the users are easily able to navigate the system, and the

requirements of the system.

b. Scope

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

c. 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

d. Example test Cases that need to be tested for:

- 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.
- 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.
- Dashboard Summary view of an evaluation is displaying all the values correctly, based on the right functional requirements to display the summary.

e. Testing Techniques

- **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. [ViTest library](#) - allows us to pass parameters to functions, and then compare the returned results with expected results

- **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.
- **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. The database will be tested to ensure that all the data from the Use cases that need to be saved into the database.
- **Front end Testing:** Manual testing will be performed for the UI components, selenium katalon recorder will be used to manually test the frontend applications
- **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.
- **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.

f. Security Principles Adhered

Having separate front-end and back-end applications ensures that users cannot gain access to the business logic. 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. 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.

22. Test Report

Requirements	Type of test	Pass or Fail	Contributor	Associated Test file
	I: Integration Testing S: System Functionality Testing O: Operational Acceptance Testing UN: Unit Testing US: Usability Testing A: Acceptance Testing	P: Pass F: Fail	SP: Sehajvir Singh Pannu LX: Lance Xu SR: Shila Rahman PD: Prabhmeet Singh Deal CZ: Charlotte Zhang	
Functional Requirements: Instructor				
1 - Account				
The system must provide a user registration feature that allows instructors to create an account.	UN, US	P	SP, PD, LX	instructorRegister.test.js
The system must accurately capture and store the following information: first name, middle name, last name, instructor ID, email, and password.	I, UN	P	SP, PD, LX	instructorRegister.test.js
The system must provide a secure login functionality for instructors to access the platform using their registered email and password, ensuring authentication and verification of credentials.	UN,US	P	SP, PD, LX	authenticateInstructor.test.js

The system must provide instructors with the ability to reset their passwords in the event of needing to change them.	UN, US	P	SP, PD, LX	updateInstructorPassword.test.js
The system must offer a secure and accessible password recovery process that ensures the privacy and security of user accounts.	UN, US	P	SP, PD, LX	resetPassword.test.js
The system must provide instructors with the ability to log out of the system, implementing appropriate security measures to ensure a secure disconnection and protect against unauthorized access.			SP, PD, LX	
The system must allow instructors to edit instructors' own profile on the platform, including their first name, middle name, last name, email and password.	UN, US	P	SP, PD, LX	updateInstructorProfile.test.js
2 - Course			SP, PD, LX	
Instructors must be able to view all the courses that are created by themselves.	UN, US	P	SP, PD, LX	getCourseList.test.js
Instructors must be able to view all students who register for this course(first name, middle name, last name, email and student ID).	UN, US, A	P	SP, PD, LX	getCourseStudentList.test.js
Instructors must be able to view all evaluation forms that were created for this course.	UN, US	P	SP, PD, LX	getCourseFormList.test.js
Instructors must be able to view all groups that were created for this course.	UN, US	P	SP, PD, LX	getCourseGroups.test.js

Instructors must be able to view all assignments that were created for this course.	UN, US	P	SP, PD, LX	getAssignmnetList.test.js
Instructors must be able to find a navigation bar to search students by student name or ID on the page.	S, US	P	SP, PD, LX	
Instructors must be able to assign students to groups.	S, UN, US	P	SP, PD, LX	addStudentToGroup.test.js
Instructors must be able to delete and remove students from groups.	UN, US	P	SP, PD, LX	removeStudentFromGroup.test.js
Instructors must be able to add new students in courses by CSV files.	UN, US	P	SP, PD, LX	addStudentToCouse.test.js addStudentToCoursePreview.test.js
Instructor must be able to view all students that are sorted by groups.	UN	P	SP, PD, LX	
Instructors must be able to create new groups in courses by CSV file.	US	P	SP, PD, LX	
Instructors must be able to edit group members within groups	UN	P	SP, PD, LX	removeStudentFromGroup.test.js addStudentToCoursePreview.test.js
Instructors must be able to delete groups from the course.	UN, US	P	SP, PD, LX	removeStudentFromGroup.test.js removeEmptyGroup.test.js
Instructors must be able to set the course visibility to publish or unpublished the course.	UN, US	P	SP, PD, LX	instructorSetCourseVisibility.test.js
3 - Evaluation Form			SP, PD, LX	
Instructors must be able to create evaluation forms.	I, S, UN	P	SP, PD, LX	instructorCreateForm.test.js

Instructors must be able to view all the questions and their options in the evaluation form.	I, S, UN	P	SP, PD, LX	instructorGetForm.test.js
Instructors must be able to edit the evaluation form if there is no evaluation submitted.	UN, US	P	SP, PD, LX	instructorEditForm.test.js
Instructors must be able to find the evaluation feedback submitted by students belonging to this course.	UN, US	P	SP, PD, LX	instructorGetFormAnswer.test.js
Instructors must be able to view the grade on one evaluation from one evaluator to one evaluatee.	UN, US	P	SP, PD, LX	instructorGetIndividualGrade.test.js
Instructors must be able to view the grade on one evaluation from multiple evaluators to one evaluatee.	UN, US	P	SP, PD, LX	instructorGetOverallGrade.test.js
Instructors must be able to set the form visibility to release or unreleased the evaluation form.	UN, US	P	SP, PD, LX	instructorSetFormVisibility.test.js
Instructors must be able to set the feedback visibility to release or unreleased evaluation feedback.	UN, US	P	SP, PD, LX	instructorSetFormShareFeedback.test.js
Instructors must be able to edit students' grades on the evaluation.	UN, US	P	SP, PD, LX	instructorEditGrade.test.js
4 - Assignment			SP, PD, LX	
Instructors must be able to create assignment and assignment rubric.	UN, US	P	SP, PD, LX	instructorCreateAssignment.test.js instructorCreateRubric.test.js
Instructors must be able to view the assignment and assignment rubric created for this course.	UN, US	P	SP, PD, LX	getAssignment.test.js

Instructors must be able to view a list that provides the information about which students have completed the assignment and which students have not.	UN, US	P	SP, PD, LX	getAssignmentCompletionStatus.test.js
Instructors must be able to view the assignment submissions from the students in this course.	UN, US	P	SP, PD, LX	instructorGetSubmissionLink.test.js
Instructors must be able to find the assignment assessment submitted by a student belonging to this course.	UN, US	P	SP, PD, LX	viewAssessment.test.js
Instructors must be able to remove the assignment if there is no assessment or submission record for this assignment.	UN, US	P	SP, PD, LX	removeAssignment.test.js
Instructors must be able to view the grade on one assessment from one evaluator to one evaluatee.	UN, US	P	SP, PD, LX	viewAssessmentIndividualGrade.test.js
Instructors must be able to view the grade on one assessment from multiple evaluators to one evaluatee.	UN, US	P	SP, PD, LX	viewAssessmentOverallGrade.test.js
Instructors must be able to set the form visibility to release or unreleased the assignment.	UN, US	P	SP, PD, LX	instructorSetAssignmentVisibility.test.js
Instructors must be able to set the feedback visibility to release or unreleased the assignment assessment.	UN, US	P	SP, PD, LX	instructorSetAssignmentSharefeedback.test.js
Functional Requirements: Student			SP, PD, LX	
1 - Account			SP, PD, LX	

The system must provide a user registration feature that allows instructors to create an account.	UN, US	P	SP, PD, LX	studentRegister.test.js
The system must accurately capture and store the following information: first name, middle name, last name, instructor ID, email, and password.	I, UN	P	SP, PD, LX	studentRegister.test.js
The system must provide a secure login functionality for instructors to access the platform using their registered email and password, ensuring authentication and verification of credentials.	I, US	P	SP, PD, LX	authenticateStudentService.test.js
The system must provide students with the ability to reset their passwords in the event of needing to change them.	I, US	P	SP, PD, LX	studentUpdatePassword.test.js
The system must offer a secure and accessible password recovery process that ensures the privacy and security of user accounts.	UN, US	P	SP, PD, LX	resetPassword.test.js
The system must provide students with the ability to log out of the system, implementing appropriate security measures to ensure a secure disconnection and protect against unauthorized access.			SP, PD, LX	

The system must allow students to edit students' own profile on the platform, including their first name, middle name, last name, email and password.	I, US	P	SP, PD, LX	
2 - Course			SP, PD, LX	
Students must be able to view all the courses in which they are enrolled.	I, US		SP, PD, LX	getCourseList.test.js
Students must be able to view all students who register for this course(first name, middle name, last name, email and student ID).	I, UN, US, A		SP, PD, LX	getCourseStudentList.test.js
Students must be able to view all evaluation forms that were created for this course.	UN, US	P	SP, PD, LX	studentGetFormList.test.js
Students must be able to view all groups that were created for this course.			SP, PD, LX	
Students must be able to view all assignments that were created for this course.	UN, US	P	SP, PD, LX	getAssignmnetList.test.js
Students must be able to find a navigation bar to search students by student name or ID on the page.	S, US	P	SP, PD, LX	
Students must be able to view all other group members that are enrolled in their own groups.	UN	P	SP, PD, LX	getGroupStudentList.test.js
3 - Evaluation Form			SP, PD, LX	
Students must be able to view all the questions and their options in the evaluation form.	I, S, UN	P	SP, PD, LX	studentGetForm.test.js

Students must be able to find the evaluation feedback submitted by other group members.	UN, US	P	SP, PD, LX	viewEvaluationFeedback.test.js
Students must be able to view the grade from one evaluator.	UN, US	P	SP, PD, LX	studentGetIndividualGrade.test.js
Students must be able to view the grade from multiple evaluators.	UN, US	P	SP, PD, LX	studentGetOverallGrade.test.js
Students must be able to submit evaluations.	UN, US	P	SP, PD, LX	submitGroupEvaluation.test.js
4 - Assignment			SP, PD, LX	
Students must be able to view the assignment and assignment rubric created for this course.	UN, US	P	SP, PD, LX	getAssignment.test.js
Students must be able to view a list that provides the information about which group members have assessed and which group members have not.	UN, US	P	SP, PD, LX	getAssessmentStatus.test.js
Students must be able to submit their assignment as a link.	I, US		SP, PD, LX	submitAssignment.test.js
Students must be able to view their own assignment submissions.	I, US		SP, PD, LX	getSubmissionLink.test.js
Students must be able to assess assignments for other random students in this course.	UN, US	P	SP, PD, LX	submitAssessment.test.js
Students must be able to view the grade on their assignment from multiple evaluators.	US		SP, PD, LX	
Functional Requirements: Super Admin			SP, PD, LX	

The system must provide a list of all instructors that have registered into the system.	UN, US		SP, PD, LX	getAllInstructors.test.js
The system must provide a user registration feature that allows super admins to create an account.	UN, US		SP, PD, LX	registerSuperAdmin.test.js
Super Admins must be able to give and remove instructors' access permission.	UN, US		SP, PD, LX	updateInstructor.test.js
Non-Functional Requirements:			SP, PD, LX	
1 - Security			SP, PD, LX	
The system needs to authenticate user accounts by verifying the correct username and password, ensuring secure access and authorization to the platform.	UN, US	P	SP, PD, LX	authenticateStudentService.test.js
The application should require users to create a strong password (with a minimum length of 12 characters, but preferably 14 or more), consisting of a combination of uppercase letters, lowercase letters, numbers, and symbols, before their account is created.	US	P	SP, PD, LX	
The system should send a verification code that will expire in 3 minutes, when users forget their password and want to reset them.	UN, US	P	SP, PD, LX	resetPassword.test.js
2 - Compatibility			SP, PD, LX	

The system should be designed to run smoothly on computers and tablets, taking into consideration the varying display screen sizes. This ensures that the user interface (UI) provides an optimal experience and usability regardless of the device being used.	O, US	P	SP, PD, LX	
While the system should be responsive and adaptable to different screen sizes, it is not necessary to specifically optimize it for mobile devices such as cell phones. The focus should be on delivering a seamless experience on computers and tablets.	US	P	SP, PD, LX	
3 - Localization			SP, PD, LX	
The deadline for assignments should be determined based on the time zone set by the instructor.	S, A	P	SP, PD, LX	
4 - Usability			SP, PD, LX	
Users must easily understand the meaning of icons, enabling them to intuitively interpret actions associated with them, such as, inferring that tapping a button with a picture of a magnifying	O	P	SP, PD, LX	

23. Usability Testing Report

a. Goals

The goal of the usability testing includes to identify the pain points of the current user experience while accomplishing the tasks while using our prototype.

We have two main goals for conducting the test case:

Usability : To access if ‘Peer Review by Learnification Technology’ prototype is effective in enabling users to complete all tasks efficiently

Accessibility : To understand if ‘Peer Review by Learnification Technology’ prototype is easy to understand for all our users.

b. Research Questions

Usability:

1. How easy is it to learn, navigate and use the platform? (i.e deviating from the happy path?)
2. How efficient is the platform in helping our users complete their respective tasks? (i.e clicks/time taken?)

c. Participants

For our usability test, we are testing sample users from our COSC 499 class’s testing event at University of British Columbia’s Okanagan campus who are familiar with using similar platforms and hold sufficient knowledge to provide important feedback and insights for our usability testing.

Ideally, we will have each user to test out both the instructor and student app to gather their feedback and insights.

d. Methodology

Observations will be conducted to identify pain points and potential areas of opportunity in the early stage of development that our users may encounter to navigate through the platform. Observation will follow ‘think out loud’ method and would be conducted in person and there will be one moderator acting as the note taker.

We design the tasks beforehand (example, login/sign up/ create course etc.) and ask the user to complete the tasks. Once they are done performing the task, they will be prompted to do the post survey questionnaire which includes the System Usability Scale and a few additional matrix questions tailored to our applications main tasks such as creating evaluation form, and performing evaluations.

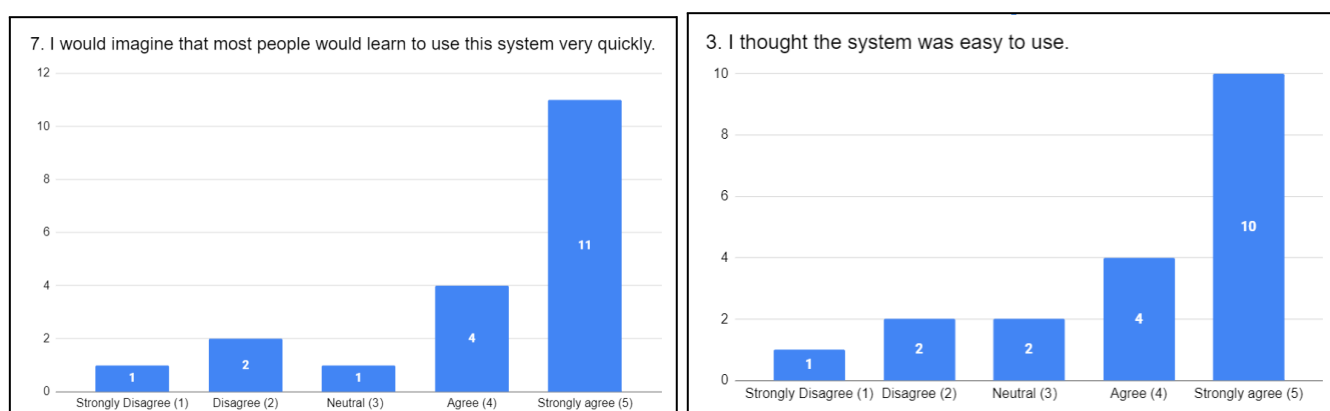
e. Analysis

The primary focus was on task completion. Successful task completion indicated a passed test, while incomplete tasks indicated a failure. Additionally, deviations from the expected user journey were considered, and user feedback was taken into account for improvements.

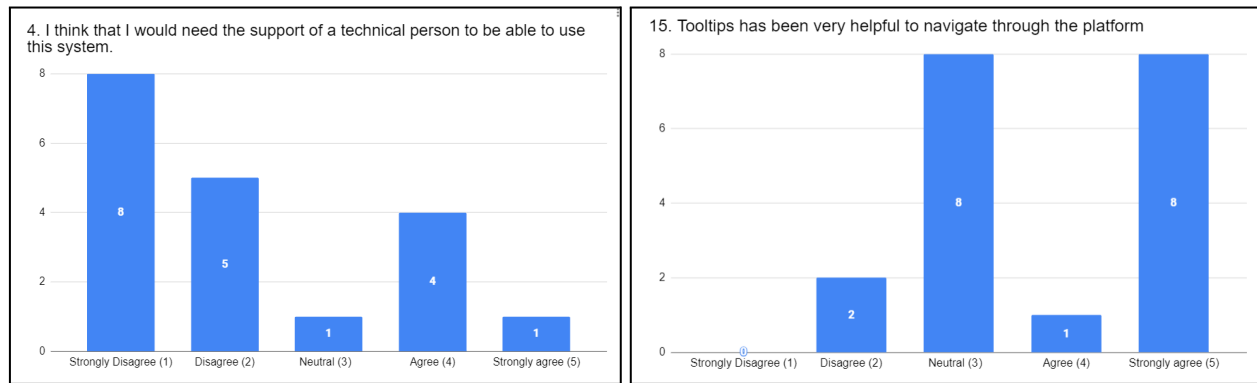
f. Results

A total of 19 participants tested the application. Based on the post-survey questionnaire responses, the System Usability Scale yielded a score of **76.38** out of 100, which is significantly above the average (greater than 68) as per usability standards. Most users expressed satisfaction with the platform. However, valuable feedback was collected to enhance the user experience.

Below are a few important barcharts we could extract from our post survey responses.



Most users agree that the system is easy to learn and easy to use



There was a more distributed/ mixed response when it came to the need of having support of a technical person to be able to use the system, we saw a similar trend when it came to the effectiveness of having tooltips.

17. What, if anything, was the most frustrating task to perform in the student app and any suggestions to improve that?

9 responses

- No frustrating tasks
- I thought that the selection to choose which student to evaluate was a bit unintuitive
- Nothing
- N/A
- It was unclear on feedback pop-ups whether they would disappear on their own or you needed to click the X
- Nope
- Navigating around the site was a little confusing, but once I got to the correct place the actual activities were easy
- Student view pretty straightforward. Instructor view needs design overhaul

16. What, if anything, was the most frustrating task to perform in the instructor app and any suggestions to improve that?

11 responses

- Add groups but not that frustrating was just a bit confused for a sec
- No frustrating tasks
- I feel like the tooltips could use some improvement and providing back buttons would be good
- Filling forms without feedback on errors (submission not going through)
- Overall experience can be improved upon.
- Nothing was great
- Nothing, was good!
- It was unclear on feedback pop-ups whether they would disappear on their own or you needed to click the X
- None

We also had open end questions to directly incorporate qualitative feedback from the users. We had some insightful feedback including “I thought that the selection to choose which student to evaluate was a bit unintuitive”, “Student view pretty straightforward. Instructor view needs design overhaul” providing us with key pointers on which part of the Application needs more improvement.

Key observations:

- Users found the system easy to learn and use.
- Responses varied regarding the need for technical support to use the system and the effectiveness of tooltips.
- Qualitative feedback indicated areas for improvement, such as making the student evaluation selection process more intuitive and redesigning the instructor view.

g. Conclusion

The usability testing results indicate a positive user perception of the 'Peer Review by Learnification Technology' prototype. The SUS score demonstrates above-average usability. Valuable insights were gathered for further refinement of the platform to enhance user experience. Incorporating user feedback will contribute to creating a more intuitive and efficient system.

24. Known Bugs

Frontend Known Bugs		
Short Description	Context	Possible Solution
DatePicker issue in Create Evaluation Form for instructor application	In the create evaluation form, if the application is opened on a Tablet, the date-time picker will have a different layout due to configuration of the datepicker library chosen. The datepicker automatically switches the layout for different devices	The only possible solution for this issue would be choosing a different date-time picker library
Add Option in Create Evaluation Form for multiple choice questions and matrix type question	In the Create Evaluation Form, when adding an extra option (other than the predefined options) for the multiple choice question and matrix type question, the option point will not be pre populated. For the prepopulated options, the option points will be pre populated.	We weren't able to find a possible solution for this as we did not have enough time left to explore this bug in depth.
Backend Known Bugs		
Section Weightage for Create Evaluation form	In Create Evaluation Form, if there is a single short answer type question in a section, the section weightage would still be divided but the question wouldn't have any points for it so while calculating the final grade, this section would be calculated towards the final grade which is not supposed to	Check the length of the section array from the incoming JSON and type of the questions for that section and handle this specific edge-case.

	happen.	
Architecture Known Bugs		
Routing problem through Front End to Nginx Reverse Proxy	Http post requests from the frontend app cannot be routed internally through the docker network. Currently the work-around has been to expose the nginx container locally, then route all the requests through that local port. In the production Environment , the IP address of the server is being used , to route requests through the local 3000 port. This has its own issues , that someone can send requests through the internet using the servers IP , and to port 3000 and interact with the APIs. Although this is protected through JWT routes, the authentication service can still be used without JWT auth.	Ideally the front end application should be able to communicate with the backend Apis , using the container name of the Nginx reverse proxy at port 80 (From the dockerfile , and nginx config). And then the microservices can be accessed through the first route address. This method did work in the default docker compose file , but on the production server it did not.

25. Key Lessons Learned

Through the journey of creating a peer evaluation platform, our team has gained valuable insights and lessons that have significantly contributed to our growth as both developers and collaborators. Adopting the Kanban methodology proved effective in maintaining a steady workflow, enhancing transparency, and ensuring each project phase was meticulously documented. The significance of comprehensive documentation became evident, allowing for seamless transitions between project stages and facilitating future improvements. However, we would also emphasize the importance of early and consistent communication within the team. Clearer communication channels would enhance collaboration and ensure that any challenges or roadblocks are promptly addressed. Moreover, incorporating periodic peer evaluations during the project could further promote a culture of constructive feedback and accountability among team members. These lessons learned have equipped us to approach future endeavors with enhanced clarity, cohesion, and a profound appreciation for the iterative development process.

26. Project Delivery

a. Testing and deployment environment

1. Frontend Applications Overview

- The main repository contains three distinct React frontend applications housed within the frontend repo.
- Every application has its own Dockerfile.
- There are two primary Dockerfiles for all frontend applications:

Development Dockerfile: Deploys the app using a node server. By attaching all the frontend app volumes to the Docker container, any code modifications are instantly captured. The hot-reload node server reflects these changes within the Docker environment, then rebuilds the apps to reflect the new changes.

Production Dockerfile: First, the React files are compiled into static HTML files within a Node environment. Subsequently, these files are hosted on an Nginx server.

2. Microservices Overview (Express)

- The Express microservices folder houses the three primary microservices: student, instructor, and authentication.
- Distinct Dockerfiles for both development and production exist:

Development Dockerfile: Utilizes a node server. The hot reload feature, supported by attaching added volumes to the container, is achieved using the `npm run dev` command.

Production Dockerfile: Relies on a Node server to host the Node.js application. These microservices are separated for streamlined code management, URL address management, and to facilitate potential future modifications. For instance, the authentication microservice is distinct to enable future security enhancements, like Azure active directory integration. One could solely modify the authentication logic without impacting other microservices.

- JWT authentication tokens are implemented both for maintaining React app state and bolstering API service security. Valid JWT tokens, produced solely by the authentication microservice, are prerequisites for accessing other microservices. If needed, the authentication can be disabled by modifying the code in the `server.js` file within the microservices.

3. Reverse Proxy (Nginx):

- All microservices utilize an Nginx reverse-proxy service container. This facilitates consolidation of microservices under a singular IP address, streamlining development and management.
- The Nginx configuration directs traffic to specific microservices based on routes.
- Further documentation on adding or managing microservices is available in **newMicroserviceDoc.md**.

4. Database Management:

- The database is an independent service and requires distinct management.
- All data resides in the **'home/peer_eval_database'** directory.
- The **'init.sql'** file contains initial database structures and some preliminary test data.
- For a database reset, one must manually bring down the database container and erase the database image. The **'reset_database.sh'** script in the **'peer_eval_database'** directory accomplishes this.

5. Database Configuration:

- If deploying to a different server, follow these steps:
 1. Relocate the database folder outside the project repository, ideally within the home directory.
 2. Update the **'docker-compose.prod_db.yaml'** file to ensure the correct context for the Dockerfile and `db_data`.
 3. Once the context is set, **'reset_database.sh'** can reset the database. Run it using **'bash reset_database.sh'**.

4. For first-time setups, execute **'docker-compose -f docker.compose.prod_db.yaml up -d'**.

6. Database User Access:

- The database operates on the Docker network and is accessible via the container name from the Docker compose file.
- Login details:
 - **Database Name: 'test_Database'**
 - **Username: 'root'**
 - **Password: 'password'**
- The credentials can be altered in the database's Dockerfile. If changed, remember to update the connection details in the **'config/config.json'** file of each microservice. During development, connect using MYSQL with **'localhost'** as the database location and **'3306'** as the port.

b. Project repository

Github Repository Link: [Link](#)

c. User manual

- Since the app has been deployed on the server , using a Continuous development pipeline which pulls the latest code from the github repository. After pulling the latest code, it will automatically rebuild using the latest code.
- **Accessing the Application:**
 1. Student Front End App:

Direct your browser to the IP address **'161.35.234.81:81'**. This will grant you access to the Student Front End Application.
 2. Instructor Front End App:

To access the Instructor platform, navigate to **'161.35.234.81:82'**.
 3. Super Admin App:

For Super Admin functionalities, use **'161.35.234.81:83'**.

- **Accessing the Application:**

For users logging in for the first time, please proceed with the following steps:

1. Navigate to the desired application interface (Student, Instructor, or Super Admin).
2. If you do not already have an account, choose the option to create one.
3. Follow the on-screen prompts to set up your account.
4. Login in using the newly created account.

- **Application Requirements:**

As this is a cloud-based application, there's no need for any installations on your end. Simply ensure you have a stable internet connection and a modern browser to access and utilize the platform.

- **Instructor Application User Manual**

- **Sign Up Page**

Users input their First name, Middle name (optional), Last name (optional), email address, employee id, password and confirm the password. Users can click on the submit button to validate all the fields they added in the sign up form and if everything is valid an account is registered.

- **Sign In Page**

Users are prompted to enter their registered email address and the password for their registered account.

Validations are performed on the inputs and once the validations are complete and successful the users can log in to the account.

Users also have the ability to use the 'Create Account' link on the sign in page to navigate to the Sign up Page to create an account if they don't have an account registered.

If the users forget their password, they can use the 'Forgot Password' link to navigate to the forgot password page where they can reset their password.

- **Instructor Dashboard**

Once successfully logged in the instructor has the instructor dashboard. Features on the dashboard include, create course, Profile settings, logout and all the courses the user is teaching.

- **Course-Page**

User gets to the course page by selecting a course from the instructor dashboard. On this page the user gets a course preview of the course with the options to navigate to the student list page, student groups page, course assignments list and course evaluations list.

- **Student List Page**

The user navigated to this page from the course page of a course. Here the user gets a list of all students enrolled in the course.

- **Group List Page**

The user navigates to this page from the course preview page where they will get the list of all the groups. From this page they have the option to edit a particular group, delete a particular group, add an empty group and add student groups to the course via a CSV file.

- **Evaluation Forms List Page**

The user navigates to this page from the course preview page where they will get the list of all the evaluation forms the user has created along with an evaluation deadline for each form. Here, they also have the toggles to release form and feedback for that particular form along with a button that will prompt the deletion of the form, another button that will navigate them to edit the form page and a button that will take them to create a new form.

- **Create Evaluation Form**

The user will land here by clicking on the “New Form” button on the evaluation forms list page. Here the user will have to accurately enter all the evaluation form details to create a new evaluation form. The system will not allow the creation of the new evaluation form until all the fields are validated.

- **Evaluation Overview Page**

When the user clicks on the evaluation form name from the evaluation list page, they will land on this page. On this page, the user has to select a group from the dropdown menu to view the evaluations data of that group. The list displayed after selecting a group, will have evaluatees and their corresponding group members as evaluators. The user can click on each evaluator to view the evaluation provided by that evaluator for the respective evaluatee and also the final grade calculated based on this feedback. The user can also view the final grade of the evaluatee based on all the evaluations and if all the evaluations have been performed for the evaluatee then the user can also edit the final grade.

- **Assignment List Page**

The user navigates to this page from the course preview page where they will get the list of all the assignments the user has created along with assignment deadline, assignment available from date, and assignment available until date for each assignment. Here, they also have the toggles to release assignment and assignment assessment for that particular assignment along with a button that will prompt the deletion of the assignment, and a button that will take them to create a new assignment.

- **Create Assignment Page**

The user will land here by clicking on the “Create Assignment” button on the assignment list page. Here the user will have to accurately enter all the assignment details to create a new assignment. The system will not allow to create the new assignment until all the fields are validated.

- **Assignment Details Page**

The user navigates to this page by clicking on the assignment name. This page shows the assignment submission status of each student

along with students' name, their student id, and their submission link (if student submitted their assignment).

- **Assignment Assessments Details Page**

The user will navigate to this page by selecting the "Assignment Assessments" from the tab on the top of the page they land when they click on an assignment from the assignment list page. Here, the user will have to select a student to view their assessment feedback and grades. The user will be provided with a search bar as well, in which they can filter out the student they are looking for. By selecting a student, the user will see a grid type list of students (evaluators) for the selected student (evaluatee). The evaluators are the students that have assessed the evaluatee's assignment based on the submission the evaluatee had made. By clicking on each evaluator, the user can view the feedback provided by the evaluator for the evaluatee and also the calculated grade on the basis of this feedback. The user can also view the final grade for the evaluatee calculated on the basis of all the evaluations performed for that evaluatee and if all the evaluations have been performed, the user can also edit the final grade for the selected evaluatee on this page.

- **Student Application User Manual**

- **Sign Up Page**

Users input their First name, Middle name (optional), Last name (optional), email address, student id, password and confirm the password. Users can click on the submit button to validate all the fields they added in the sign up form and if everything is valid an account is registered.

- **Sign In Page**

Users are prompted to enter their registered email address and the password for their registered account.

Validations are performed on the inputs and once the validations are complete and successful the users can log in to the account.

Users also have the ability to use the ‘Create Account’ link on the sign in page to navigate to the Sign up Page to create an account if they don’t have an account registered.

If the users forget their password, they can use the ‘Forgot Password’ link to navigate to the forgot password page where they can reset their password.

- **Student Dashboard**

Once successfully logged in the student has the student dashboard. Features on the dashboard include Profile settings, logout and all the courses the user is enrolled in by the instructor.

- **Course-Page**

User gets to the course page by selecting a course from the student dashboard. On this page the user gets a course preview of the course with the options to navigate to the student list page, groups page, course assignments list and course evaluations list.

- **Student List Page**

The user navigated to this page from the course page of a course. Here the user gets a list of all students enrolled in the course.

- **Group List Page**

The user navigates to this page from the course preview page where they will get the list of all the members of the group the user is in.

- **Evaluation Forms List Page**

The user navigates to this page from the course preview page where they will get the list of all the evaluation forms they have to perform along with an evaluation deadline for each form. Here, they will also get a button to view their feedback for each evaluation performed by their group members and the same page will also have the final grade calculated based on all the evaluations.

- **Evaluation Overview Page**

The user will be navigated to this page when they click on the evaluation form name on the evaluation list page. Here, they will see the evaluation status for all the evaluations they have to perform for this specific form. The user has to perform the evaluations of their group members. By clicking on each group member, the user will be navigated to the next page, where they will have to accurately perform the evaluations for the selected group member.

- **Assignment List Page**

On the assignment tabs, the user will have the list of all the assignments the instructor has created and released. Here, the list will have the assignment name, assignment deadline date, assignment available from date, assignment available until date, submission state, and “View Grade/Feedback” button if the instructor has released the feedback for this assignment. By clicking on the assignment name, the system will navigate the user to the next page where the user will have all the assignment details (including assignment name, assignment description, assignment rubric, and type of the link they have to submit for this assignment) and on this page, the user will be submitting their assignment by providing the valid link.

- **Assignment Assessment**

The “Assignment Assessment” tab will have a dropdown where the user will have to select an assignment they want to assess the other students’ submissions. When the user selects an assignment from the dropdown, a table will be displayed with all the assignment assessments the user has to perform.

- **Super Admin Application User Manual**

- **Sign In Page**

Users are prompted to enter their email address and the password for their registered account.

Validations are performed on the inputs and once the validations are complete and successful the users can log in to the account.

- **Dashboard Page**

Users will get a list of all the instructors registered in the system. The table will also have the checkboxes along with the instructor names, the user can select/unselect these boxes and when they select the box, the system will display 2 buttons on the top “Grant Access” and “Revoke Access”. If the user selects an instructor and clicks on the Grant Access button, the selected instructor will be provided with the instructor access and if the user clicks on the Revoke Access button, the selected instructor will be revoked from the instructor access.

d. User deploy guide for a development environment

1. Prerequisites:

Before proceeding, ensure you have the following applications installed on your local machine:

Docker: Essential for creating and managing your application containers.

Docker Desktop: Provides a visual interface for Docker functionalities.

Docker Compose: A tool for defining and running multi-container Docker applications.

Node: Will help us install dependencies during development.

2. Cloning the Project:

Clone the project repository to your local machine: ‘git clone [\[Link to the repository\]](#)’

3. Installing Dependencies:

Execute the provided script (inside a terminal in the main project repo) to install all necessary node modules and dependencies for each app, using the given command in the main repo. All the frontend and backend dependencies can be found in the table below in point e) and f).

‘bash install_dependencies.sh’

4. Launching Development Containers:

Run the following command (inside a terminal in the main project repo) to initiate and launch the development containers and images:

docker-compose up -d

5. Accessing Front End Applications:

Post deployment, utilize **Docker Desktop** to open the designated ports for accessing the frontend applications. The apps will be available via the respective ports on **'localhost'**.

e. Frontend Library Requirements and their Description

Dependency Name	Version	Link	Description
@ant-design/cssinjs	1.16.2	@ant-design/cssinjs - npm	CSS-in-JS solution for Ant Design components.
@ant-design/icons	5.2.5	@ant-design/icons - npm	Ant Design icons library.
@babel/core	7.22.10	@babel/core - npm	Babel compiler core.
@babel/preset-env	7.22.10	@babel/preset-env - npm	Babel preset for transforming JavaScript for specific environments.
@babel/preset-react	7.22.5	@babel/preset-react - npm	Babel preset for React applications.
@emotion/react	11.11.1	@emotion/react - npm	Emotion is a popular CSS-in-JS library.
@emotion/styled	11.11.0	@emotion/styled - npm	Emotion Styled is a styling library for Emotion.
@mui/icons-material	5.14.3	@mui/icons-material - npm	Material-UI icons library.
@mui/material	5.14.5	@mui/material - npm	MUI (formerly Material-UI) is a popular React UI framework.
@mui/x-date-pickers	6.11.1	@mui/x-date-pickers - npm	Material-UI X Date Pickers for MUI.
antd	5.8.3	antd - npm	Ant Design is a comprehensive UI design language and framework.
axios	1.4.0	axios - npm	Axios is a promise-based HTTP client for the browser and Node.js.

babel-plugin-import	1.13.8	babel-plugin-import - npm	Babel plugin for importing components on demand from Ant Design and Material-UI.
crypto-js	4.1.1	crypto-js - npm	Crypto-JS is a library for cryptographic functions in JavaScript.
dayjs	1.11.9	dayjs - npm	Day.js is a fast, immutable, and internationalizing date library.
formik	2.4.3	formik - npm	Formik is a popular library for managing forms in React applications.
history	5.3.0	history - npm	History is a JavaScript library that manages session history and provides utilities for managing navigation in a web application.
jwt-decode	3.1.2	jwt-decode - npm	jwt-decode is a library for decoding JWT tokens.
moment	2.29.4	moment - npm	Moment.js is a popular date and time manipulation library.
papaparse	5.4.1	papaparse - npm	PapaParse is a fast in-browser CSV (or delimited text) parser library.
react-dom	18.2.0	react-dom - npm	React DOM is the entry point to the DOM-specific methods in React.
react-redux	8.1.2	react-redux - npm	React Redux is the official state management library for React.
react-router-dom	6.15.0	react-router-dom - npm	React Router is a library for routing in React applications.
react-scripts	5.0.1	react-scripts - npm	React Scripts is a set of scripts for creating React applications.
react	18.2.0	react - npm	React is a JavaScript library for building user interfaces.
redux-persist	6.0.0	redux-persist - npm	Redux Persist is a library for persisting Redux store in React Native or web apps.
redux	4.2.1	redux - npm	Redux is a predictable state container for JavaScript apps.
yup	1.2.0	yup - npm	Yup is a library for schema validation.

f. Backend Dependencies/Libraries and their Description

Dependency Name	Version	NPM Repository URL	Description
cookie-parser	1.4.4	cookie-parser - npm	Parse HTTP request cookies with signatures.
cors	2.8.5	cors - npm	CORS is a middleware that can be used to enable CORS with various options.
debug	2.6.9	debug - npm	A small debugging utility that provides color-coded debug output.
express	4.16.1	express - npm	Fast, unopinionated, minimalist web framework for Node.js.
fs	0.0.1-security	fs - npm	Node.js File System module for interacting with the file system.
http-errors	1.6.3	http-errors - npm	Create HTTP errors for Express applications with more details.
jade	1.11.0	jade - npm	Jade is a template engine for Node.js and the browser.
jsonwebtoken	9.0.1	jsonwebtoken - npm	JSON Web Token implementation for Node.js.
jwt-decode	3.1.2	jwt-decode - npm	Library to decode JWT tokens in the browser.
morgan	1.9.1	morgan - npm	HTTP request logger middleware for Node.js.
mysql2	3.3.3	mysql2 - npm	MySQL client for Node.js with focus on performance.
path	0.12.7	path - npm	Node.js path module for working with file and directory paths.
sequelize	6.32.0	sequelize - npm	Multi-dialect ORM for Node.js and io.js.
express-jwt	8.4.1	express-jwt - npm	Middleware that validates a JSON Web Token (JWT) and sets req.user.
jest	29.5.0	jest - npm	Delightful JavaScript testing framework with a focus on simplicity.

supertest	6.3.3	supertest - npm	HTTP assertions for testing Express.js apps.
vitest	0.31.4	vitest - npm	Framework-agnostic test runner for the web.

g. Features remaining

All the features have been successfully completed.

27. Approvals

Client Signature

Date

Team Representative

Date