Blackboard Tutor®

Software Engineering Project

August 2023

Project Manager,
BC Smit, 2020053969

Lead Developer,
K Kolanchu, 2020447641

Developer, SQA Manager
C Venter, 2019659602

UX Designer, Developer
J Smith, 2020046385

Table of Contents

Table of Figures

Blackboard Tutor®

Chapter 1: Project Proposal

Project Manager,
BC Smit, 2020053969

Lead Developer,
K Kolanchu, 2020447641

Developer, SQA Manager
C Venter, 2019659602

UX Designer, Developer
J Smith, 2020046385

Introduction

This chapter will provide more insight to the functionality and operation of the Peer-to-Peer Tutoring system. We will discuss details of the implementation and integration of the database. Furthermore, we will look at some of the user interfaces. The remaining user interfaces will be discussed in chapter 2.

Project Description

Purpose

The University of the Free State wants to improve the retention rate of first year students in the department of Computer Science and Informatics. For this reason, the University of the Free state requires a tutoring system. This system will be called the “Student Peer-to-Peer Tutoring System”.

Objectives

Based on surveys conducted by the University of the Free state the primary reason the retention rate of first-year students in the Department of Computer Science and Informatics is so low is because students struggle to develop their programming skills.

Therefore, the main objective of this system should be to provide students with an **online code compiling service**. Students must be able to write code, compile it and receive feedback instantaneously.

Furthermore, after discussions with the client, the requirements team identified the following additional objectives of the “Student Peer-to-Peer Tutoring System”:

- **Virtual classroom:** The system should include all the features of a virtual classroom. Tutors and lecturers should be able to teach students by means of collaborative sessions. The system should allow tutors and lecturers to create quizzes that students can complete. Lastly, there should be a way for students and tutors/lecturers to communicate. This should be either via live chat during collaborative sessions or a chat service.
- **Gamification:** The system should keep students motivated by rewarding a student’s hard work. The system must award badges and points to students based on good attendance, excellent grades, and completing coding problems. Furthermore, the points should be displayed on a leaderboard where users can view their own rank as well as the rank of other students.

Scope

The project scope as identified by the analysis team includes the following:

- **User Authentication:** The system must provide secure login using the University of the Free state Peoplesoft credentials.
- **User-Friendly System:** The system should have a user-friendly interface. The system should be accessible via different devices including smartphones, laptops, etc. For this reason, the requirements team have decided on a web-based system.
- **Dashboard**: The system must provide different intuitive dashboards for students, tutors, lecturers, and administrators.
- **Module-Specific Sections:** The system must provide dedicated sections for each academic module, including a vault that contains the study materials, notes, assignments, test papers, and exam papers for that module.
- **Schedule and Planning**: The system should allow lecturers and tutors to plan collaborative sessions. This must be achieved by a calendar functionality. Lecturers and tutors should be able to add events to a calendar that will be seen by all other users who form part of that module. Furthermore, the system should allow users to add private events (that will not be displayed to other users).
- **User Feedback:** The system should allow students to rate their tutor after a collaborative session.
- **Reporting Misconduct:** The system must provide students with the ability to report a tutor for misconduct.
- **Data Analysis:** The system must provide statistics on attendance, participation and performance which must be accessible to students, lecturers, and tutors to allow a lecturer or tutor to determine if the tutoring is having a positive effect on the student.

Components and Features

Database for Peer-to-Peer Tutoring System:

An Amazon Web Services Cloud database will be used to store the information required for the tutoring system. Amazon will be responsible for database backups and snapshots.

The database will be updated with the data in real-time. For example, after a student has completed a quiz, the result will be sent to the database and the average quiz mark will be calculated and updated immediately. The average tutor rating will be sent to the database and calculated half an hour after the

session was completed. The detail of this feature is discussed in the 'Rate Session' use case.

For the leaderboard the system will use a SQL job / scheduled task that will run at midnight daily that will calculate the new ranking of all students and update the leaderboard. This process can take multiple hours and therefore was decided to run in the early hours of the morning.

The following information will be obtained from the PeopleSoft system:

User Information:

- Student Number / Staff Number
- Name
- Surname
- Email Address
- Role

The AWS database will store the following information:

Study Item:

- Study Item ID
- Name
- Uploaded item (e.g. .pdf, .docx, .ppt, etc.)
- Size

Question:

- Question ID
- Description
- Difficulty,
- Points
- Option 1
- Option 2
- Option 3
- Option 4
- Answer

We acknowledge that the number of options should not have a fixed length of 4 but should rather allow the lecturer or tutor to set the length, however for the purposes of this assignment we will delimit the scope of this feature.

Quiz:

- Quiz ID
- List<Question>
- Difficulty
- Timer

Abstract Session:

- Session ID
- Name

Upcoming Session:

- Upcoming date

Archived Session:

- Size
- Date Uploaded
- Uploaded Recording

Module:

- Module Code
- Name
- List<StudyItem>
- List<Quiz>
- List<Sessions>

Abstract User:

- User code
- Name

- Surname
- Email Address
- List<Module>
- Calander
- Diary
- List<Chat>

Calander:

- Calander ID
- List<Event>

Event:

- Event ID
- Title
- Module Code
- Start Date Time
- End Date Time

Diary:

- Diary ID
- List<Entry>

Entry:

- Entry ID

  Recording:
  - Uploaded Recording (i.e., MP4)

  Text:
  - Title,
  - Body

To-Do:

- Title,
- Dictionary<String, Bool>

Chat:

- User Code (Sender)
- User Code (Receiver)
- List<Message>

Message:

- Message ID
- Body
- [Optional] Attachment
- Send Date Time

Student:

- Student Number
- Average Grade
- List<Tutor>
- List<Badges>
- List<Module>
- Average Attendance

Lecturer:

- Staff Number
- List<Tutor>
- List<Report>
- List<Application>

We acknowledge that a lecturer should also be able to view the statistics for all his students and tutors, however for the purposes of this assignment we will delimit the scope of this feature.

Tutor:

- Student Number
- Average Rating

Application:

- Application ID
- Student Number
- Module Code
- Module Mark
- Description
- Supporting Document

Badge:

- Badge ID
- Name
- Requirement
- Image

Report:

- Report ID
- Student Number (Reported)
- Student Number (Reporter *if not anonymous)
- Description
- Supporting Document
- Bool indicating if the student wants to Remain Anonymous

Student Grade:

- Student Number
- Quiz ID
- Difficulty Level
- Attempt Number
- Grade

Leaderboard:

- Student ID
- Number of Points
- Ranking
- Current View

Code problems:

- Problem ID
- Status
- Category
- Description
- Constraints
- Output

Code Student Mark:

- Student Number
- Problem ID
- Status
- Mark

Application Programming Interfaces for Peer-to-Peer Tutoring System:

The analysis team identified that the following application programming interfaces will be required to implement the collaborative session feature of the Peer-to-Peer Tutoring system:

- Google Cloud Speech-to-Text API (speech-to-text),
- Microsoft Teams API (video and audio).

Lastly, the following ASP .NET library will be used:

- SignalR (live chat).

Client Feedback

The analysis team conducted a comprehensive review of the requirements. It was concluded that all features requested by the UFS Department of Computer Science and Informatics can be implemented within a reasonable timeframe and budget.

However, the analysis team have identified potential redundancies within the requirements. It has been decided that some of the requested features can be combined as follows:

- **Vault / Study Material / Resources:**

  All features related to the storing and uploading of “study material” including assignments, tests, exams sections can be combined into a single “vault” section.

- **Diary / to-do List:**

  The “to-do list” can be combined with “diary”. This would provide a centralised area for all the users ideas, notes, and reminders. The detail of this feature is discussed under the “vault” use case.

  The user will be able to add different types of entries in a diary. These entries include:

  - Audio recording,
  - Text entry,
  - To-do list.

  Each To-do list item will have a checkbox next to it. The user will be able to toggle the checkbox to show that a to-do list item is completed. The detail of this feature is discussed under the “diary” use case.

- **Offline Access / Download:**

  The system that will be developed will be web-based. When a user wants to view a file offline, the user must download the file while they have access to the internet. The file will be downloaded in a dedicated folder created by the system. The system will create subfolders according to the type of item that was downloaded, e.g. study item, session recording etc. In each subfolder the material that has been downloaded by the user can be accessed, even when the user does not have an internet connection.

- **Notifications / Remind me / Announcements:**

  All communications made by lecturers and tutors will appear under an "Announcement" board that will be available on all users' dashboards. Kindly see Figure 1-3 User Dashboards.

- **Awards / Badges / Goal Setting:**

  Awards / Badges and Goal setting can be combined as a single "Badge" function. Badges are awarded to students when they achieve a goal e.g., have 80% attendance or have an 75% average grade for their quizzes.

Additional Comments

- **Lecturer/Tutor Code Feedback:**

  The client expressed that they would like a tutor/lecturer to view, mark and provide feedback to a student for a specific problem they completed on the online compilation service. However, the analysis deemed this feature to put unnecessary strain on the tutor/lecturer. Rather, the system will automatically be able to provide feedback to the user.

  We acknowledge that a user/lecturer should still be able to view the student's submission and provide additional comments, however for the purposes of this project we will not include this feature.

- **The impact that tutoring has on a student's performance:**

  Furthermore, the client mentioned that they want a way for a tutor/lecturer to determine if the tutoring is having a positive effect on a student.

  We take note of the above requirement however, however for the purposes of this project we will not include this feature. We believe that for now the "statistics" feature is sufficient.

User Interface Description

The following dashboards will appear for the users:

Kindly note that the details and user interface of all the below functions are discussed in detail in their respective case-diagrams and class-responsibility-collaboration cards.

Lecturer

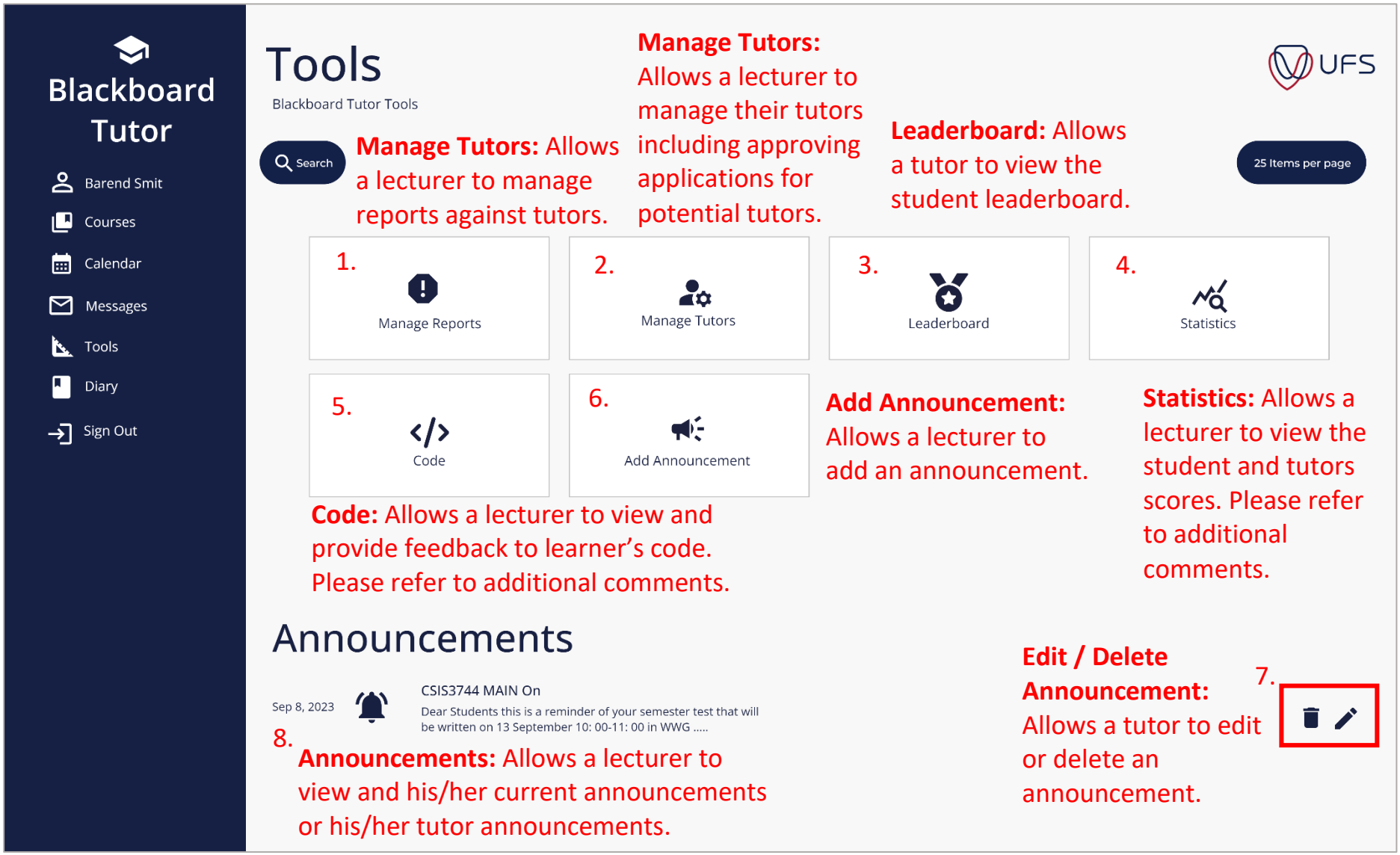


Figure 1: Lecturer Dashboard

Tutor

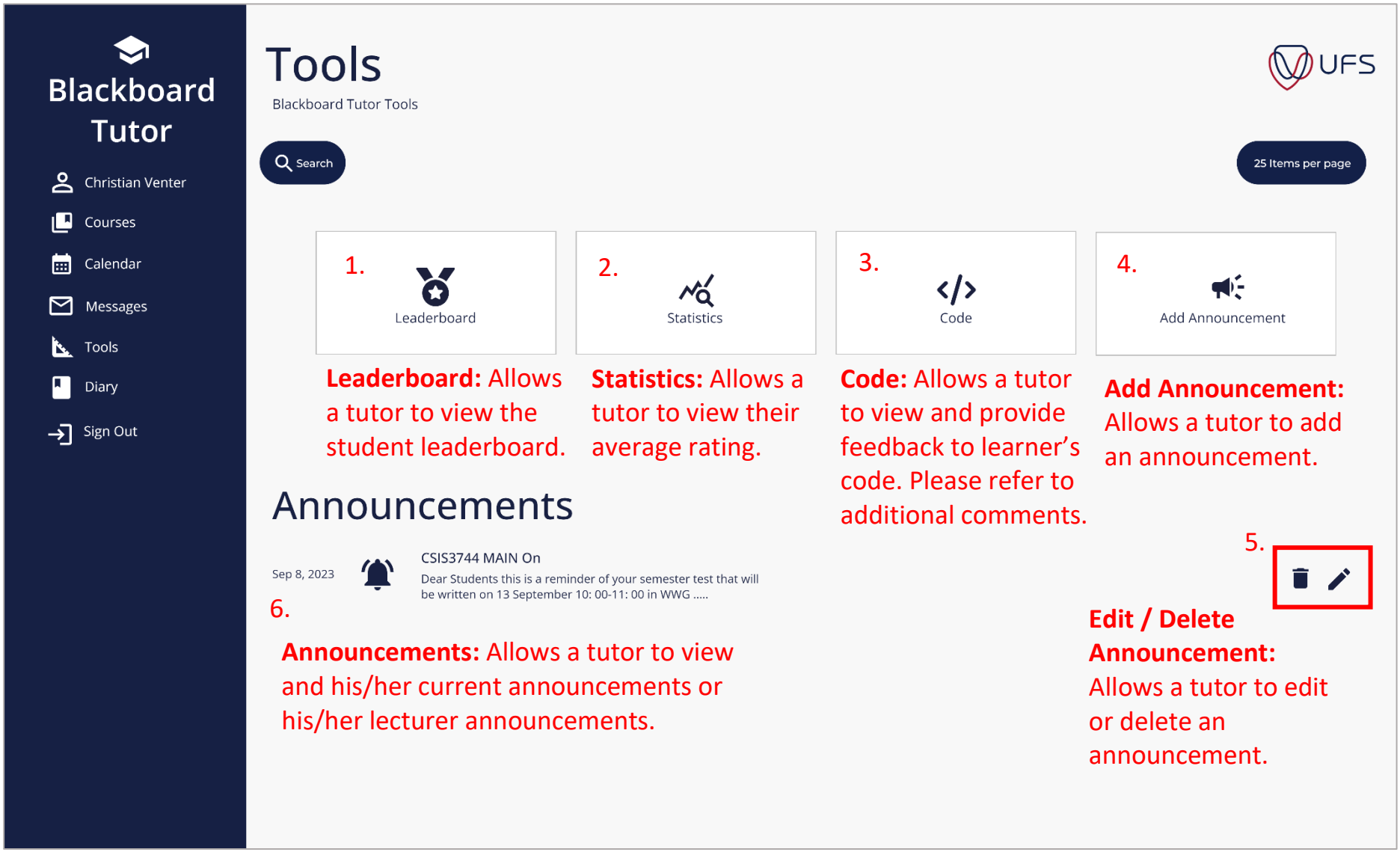


Figure 2: Tutor Dashboard

Student

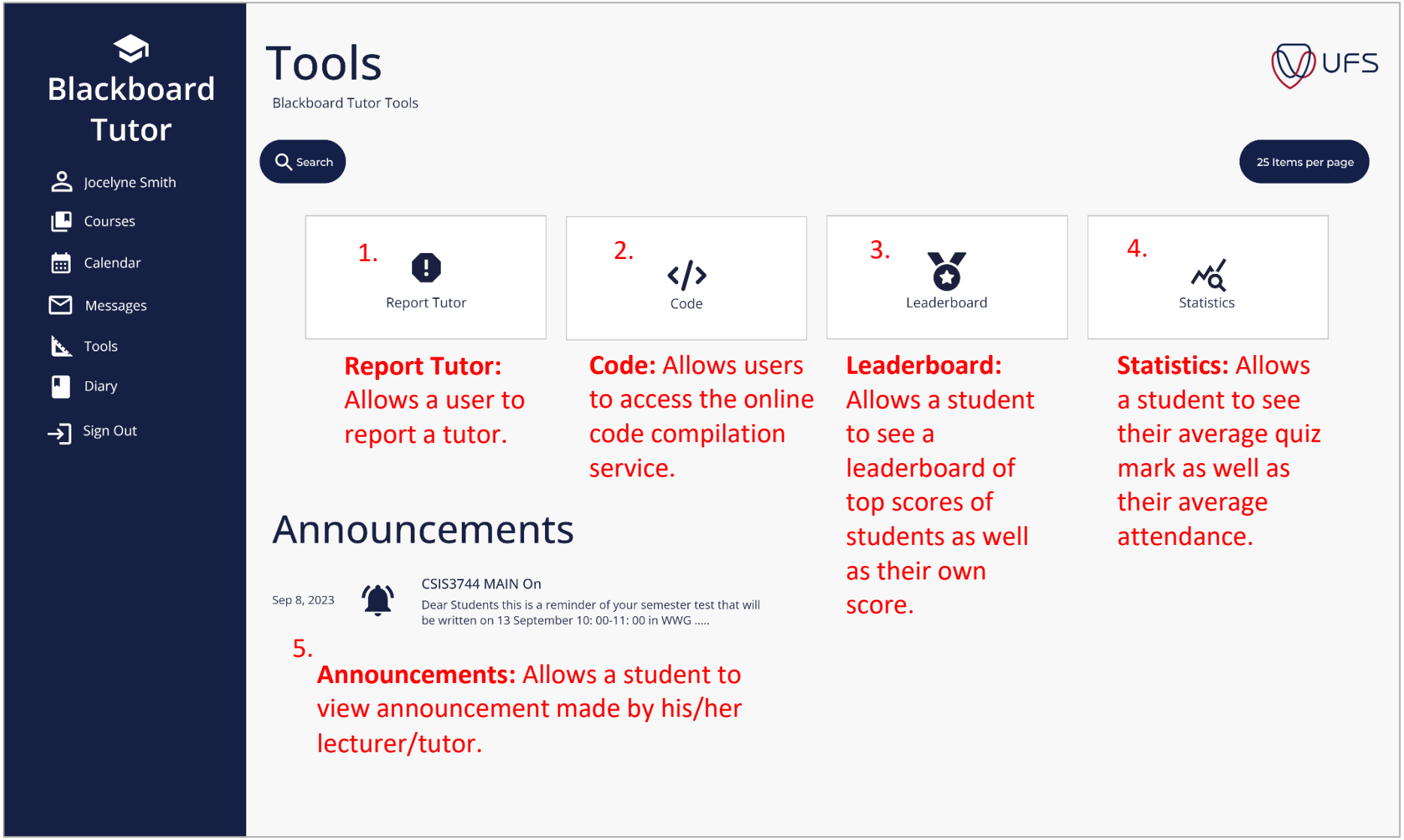


Figure 3: Student Dashboard

The following menus will be used by users to navigate the system:

Side Menus

The side menu appears when a user accesses the website.

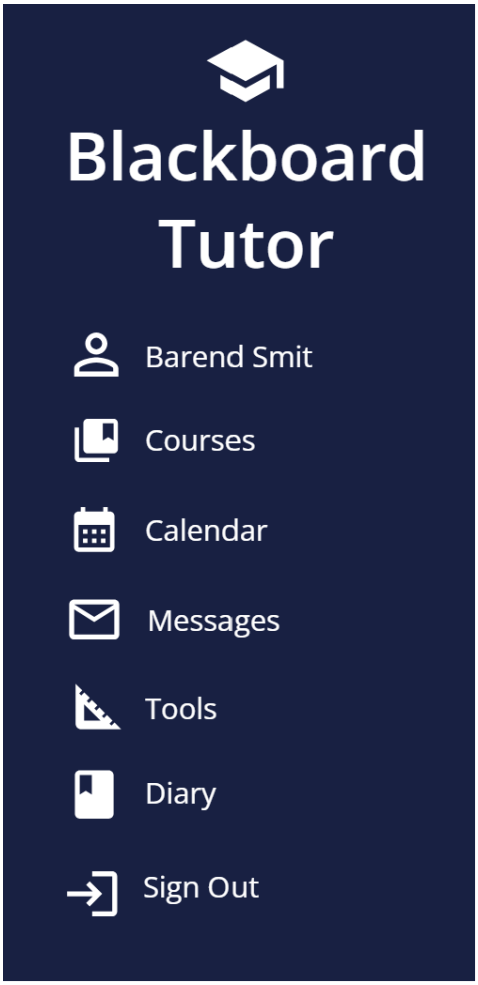


Figure 4: Home Menu

The user can navigate back to this menu when they click on the “Home” button within a module (kindly see the “Home” button on Figure 5 Module Menu).

This menu allows access to the following features.

- **Profile:** Allows a user to view his/her information including:
  - Staff number / Student number,
  - Full Name,
  - Email Address.
- **Courses:** Displays all the modules that the student is currently enrolled for or that the lecturer/tutor is currently teaching.

- **Calendar:** Calendar allows a user to view their daily schedule or monthly calendar. Users can add their own events. Events scheduled by tutors/lecturers are displayed to the students that are enrolled for their module.
- **Messages:** Allows messages to be sent between users. For lecturers this feature can be used to communicate with students that have made reports against tutors, even if the student chose to remain anonymous.
- **Tools:** Allows access to the additional features that are specific to each user (kindly see the above dashboards for the details of these features).
- **Diary:** A virtual diary which will allow users to add notes and voice recordings. There will also be an option to create a to-do list within the diary. Each To-do list item will have a checkbox next to it. The user will be able to toggle the checkbox to show that a to-do list item is completed.
- **Sign-out:** Once the user has clicked on the log-out button the system will ask them whether they wish to leave the website. Should the user wish to leave the system will return the user to the login page.

The detail of each of the above-mentioned features is discussed under their relevant use cases.

Module Side menu:

This menu appears when a user accesses a module:

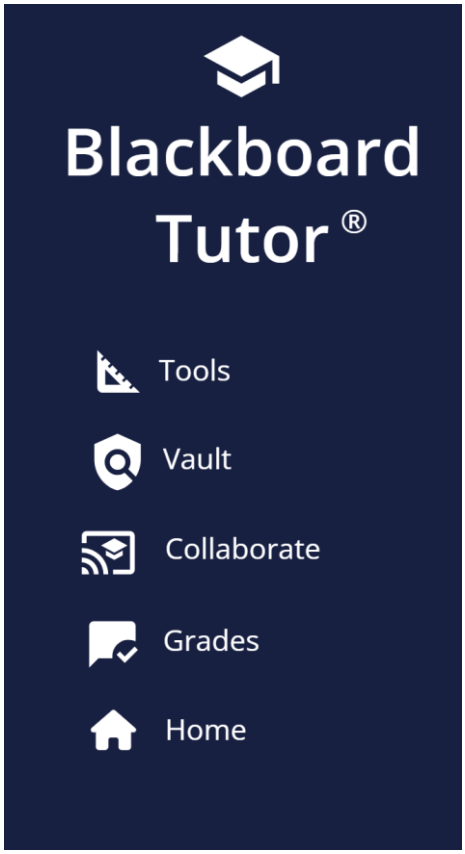


Figure 5: Module Menu

- **Tools:** Allows a student to access module-specific features such as quizzes. Similarly, it allows lecturers/tutors to access “Manage Quizzes” or “Manage Study Material”.
- **Collaborate:** Contains a list of upcoming and archived sessions. A user can join a session and experience a virtual classroom. The detail of this feature is discussed under the “session” use case.
- **Vault:** Displays all the material provided by lecturers or tutors. The name and size (in megabyte) for each item e.g. pdf, video, text, etc. will be visible.
- **Grades:** Allows a student to view their results for quizzes.
- **Home:** Allows a user to navigate back to the “Home” side-menu (Kindly see Figure 4 Home menu).

The following additional user interfaces can be used by users:

Log-out:

The “log-out” user interface provides the user with safety tips related to logging out. It also prompts the user to confirms if he/she wants to log-out or stay logged in.

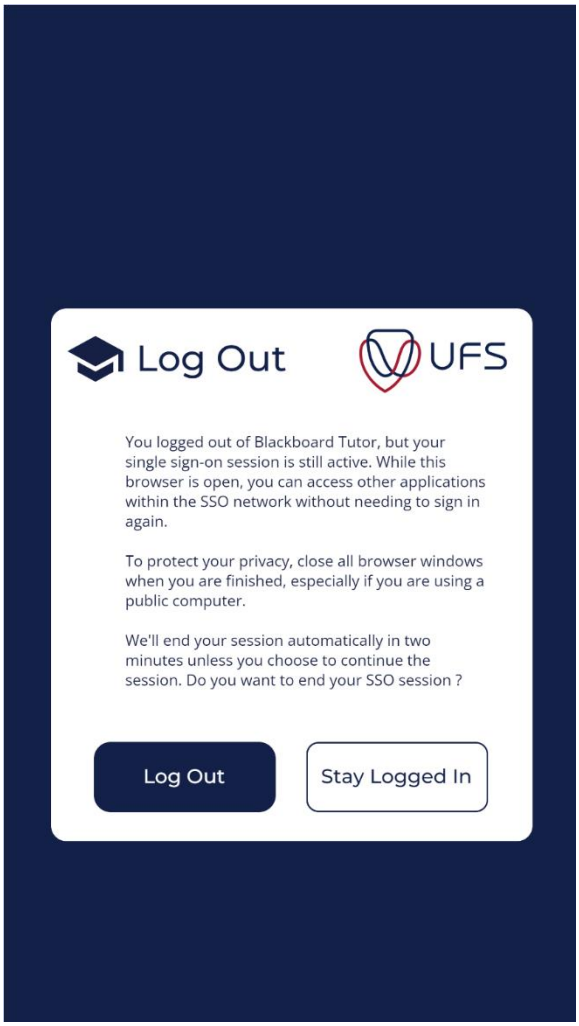


Figure 6: Log Out User Interface

Searching and Filtering:

Searching and Filtering is available for most features, including but not limited to vault, Calander, sessions etc.

1. **Searching:** Allows the user to enter a search query. The system will respond by only showing data match the user's search query. It will be used to locate specific items or content quickly.

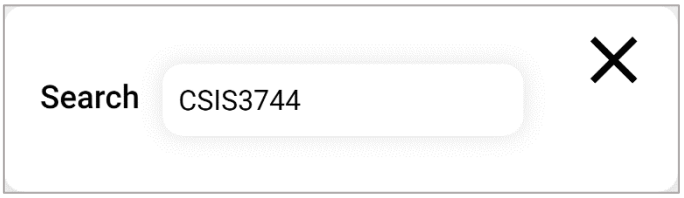


Figure 7: Searching User Interface Popup

2. **Filtering:** Allows the user to enter a filter query. The system will respond by only displaying a subset of the data that meets the given criteria. Filtering allows users to focus on specific subsets of data that are relevant to their needs.

Figure 8: Filtering User Interface Popup

We acknowledge that searching and filtering should have been included as use cases, but for the purposes of this assignment we will only discuss their function.

Conclusion

This chapter provided background on the Peer-to-Peer Tutoring system. It provided details about the project description including the purpose, objectives, and scope. Furthermore, it also provided more information about the components and features of the system, including details of the operation and integration of the database as well as the Application Programming Interfaces. Lastly this chapter discussed the feedback that was provided to the client and looked at some of the basic user interfaces including the dashboards, side menus, log-out, searching and filtering.

Blackboard Tutor®

Chapter 2: Specifications Document

Project Manager, BC Smit, 2020053969

Lead Developer, K Kolanchu, 2020447641

Developer, SQA Manager C Venter, 2019659602

UX Designer, Developer J Smith, 2020046385

Introduction

This chapter will provide a detailed look at the use-cases necessary for the Peer-to-Peer Tutoring system, along with their associated user interfaces.

Specifications Document

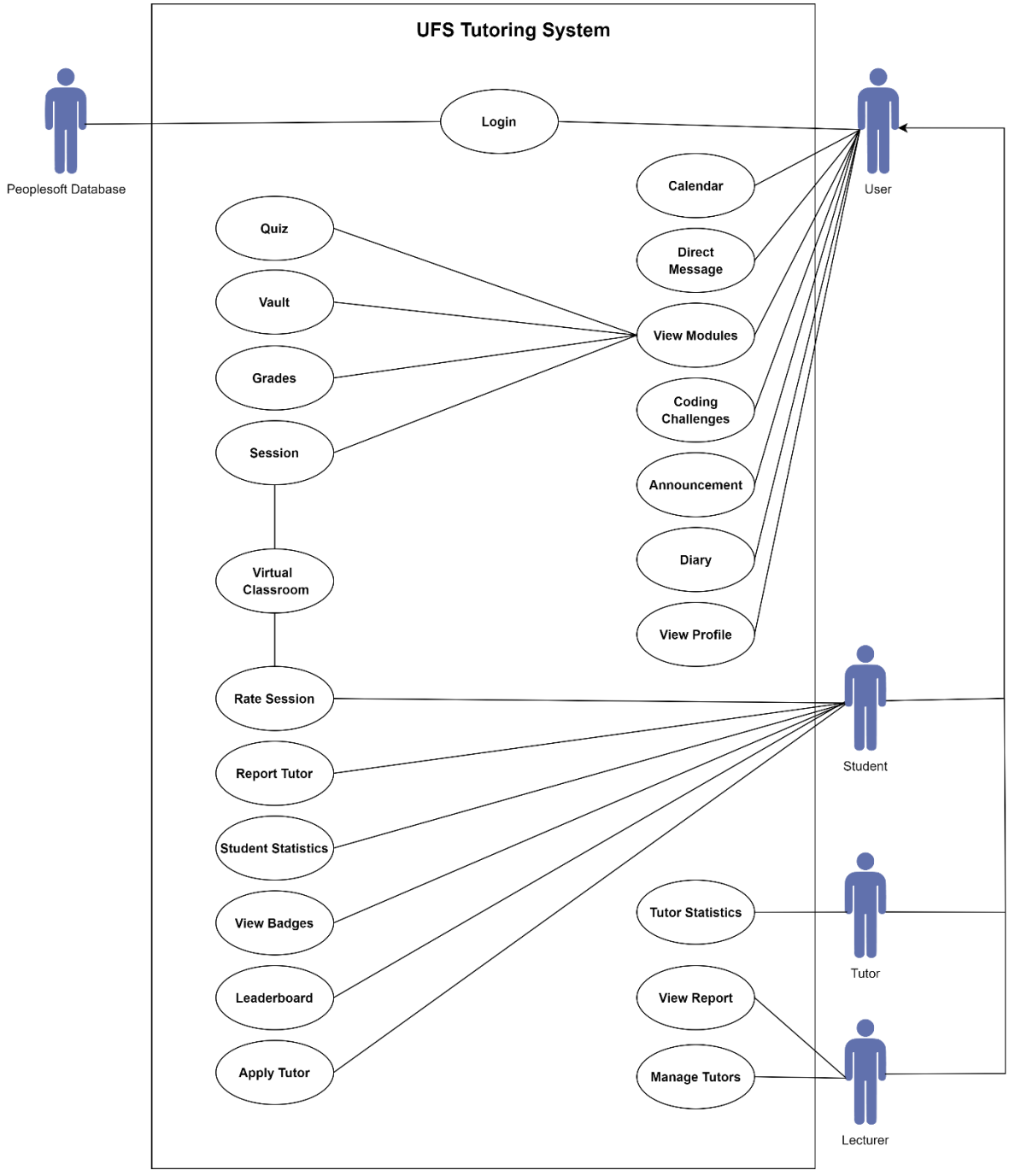


Figure 9: Consolidated Use Case Diagram

Although not depicted on this diagram accurately, all classes can only be accessed after the user has logged in.

Login

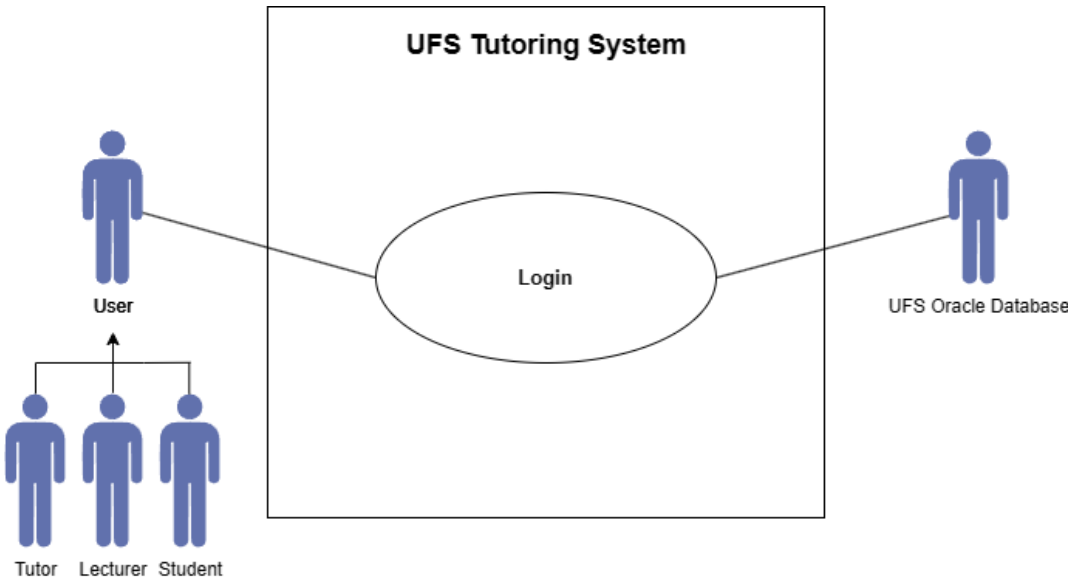


Figure 10: Login Use Case Diagram

Brief Description

The Login function enables registered first-year computer science and informatics students as well as their tutors/lecturers to login and access the system.

Step-by-Step Description

1. User accesses the website.

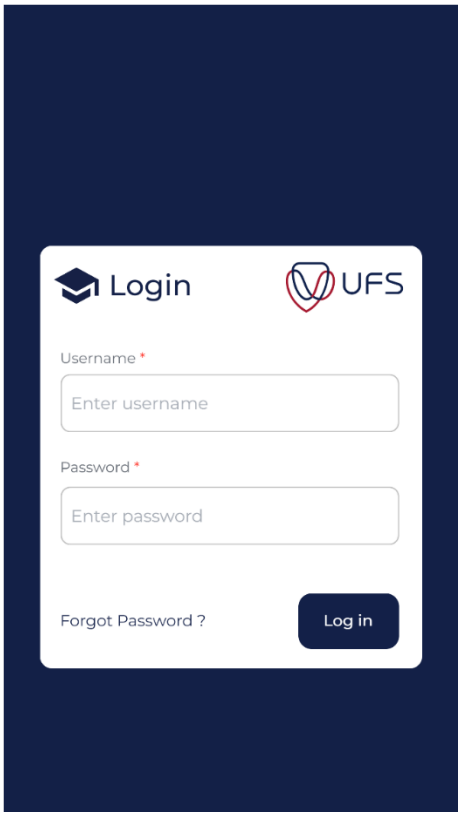


Figure 11: Login User Interface

2. System displays a login screen to the user.
3. The login screen prompts the user to enter their username and password.
4. User enters their credentials and clicks the login button.
5. System responds by determining if only valid input has been entered by the user:
    - numeric input for the username and,
    - alphanumeric and special characters for the password.
6. System checks that a profile for the username or password exists.
7. If the account exists, the system checks that credentials math those of the UFS Peoplesoft system.
8. If the credentials are correct, the user will be allowed to access to the system and their personalized dashboard will be displayed.

Best-case Scenario

1. Kamo accesses the website.
2. System displays a login screen to Kamo.
3. Kamo enters his correct PeopleSoft credentials and clicks the login button.
4. System checks Kamo's credentials.
5. System deems Kamo's credentials to be correct.
6. System allows Kamo access to his personalized dashboard.

Worst-case Scenario

1. Kamo accesses the website.
2. System displays a login screen to Kamo.
3. Kamo attempts to use SQL injection by entering SQL in the username field.
4. System does not allow the input and displays an error message stating: "Input only valid alphanumeric or special characters only for the password."

Alternative Scenario A

1. Kamo accesses the application.
2. System presents Kamo with a login page.
3. Kamo does not fill in his password and presses login.
4. System displays an error message stating: "Please enter your password."

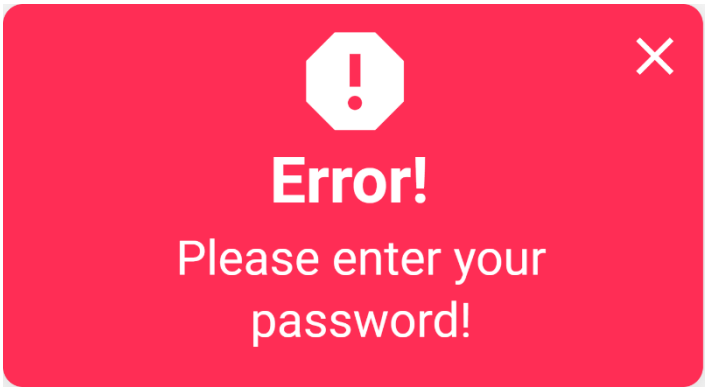


Figure 12: Valid password error

Alternative Scenario B

1. Kamo accesses the application.
2. System presents Kamo with a login page.
3. Kamo does not fill in his username and presses login.
4. System displays a popup with the following error message: “Please enter your username.”

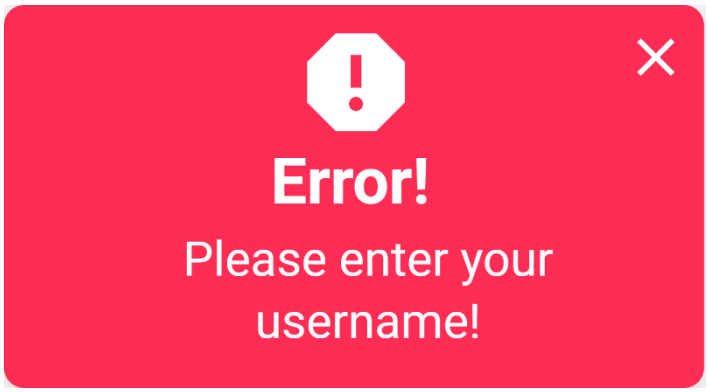


Figure 13: Valid username error

Alternative Scenario C

1. Kamo accesses the application.
2. System presents Kamo with a login page.
3. Kamo fills in his correct username, however an incorrect password.
4. System responds with a popup stating: “The username or password does not match.”

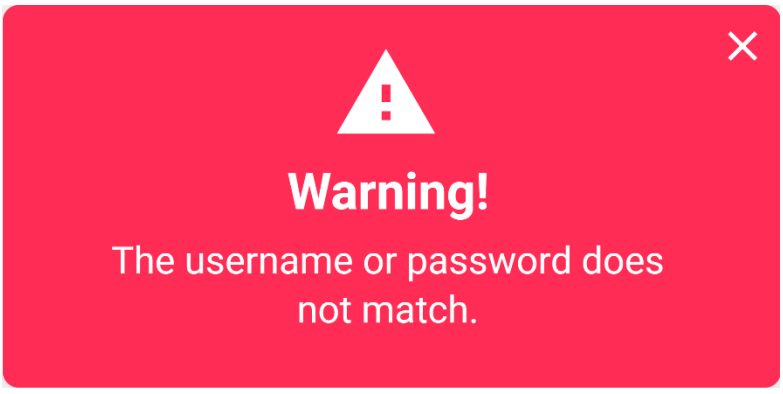


Figure 14: Username and Password does not match error

Alternative Scenario D

1. Kamo accesses the application.
2. System presents Kamo with a login page.
3. Kamo is a student in the Health Sciences Faculty. He tries to login using his correct PeopleSoft credentials.
4. Since he is not a valid user, the system displays an error message stating: “Your account does not exist. You are not computer science student or a lecturer/tutor. You cannot access the system. Contact ICT if this is a mistake.”

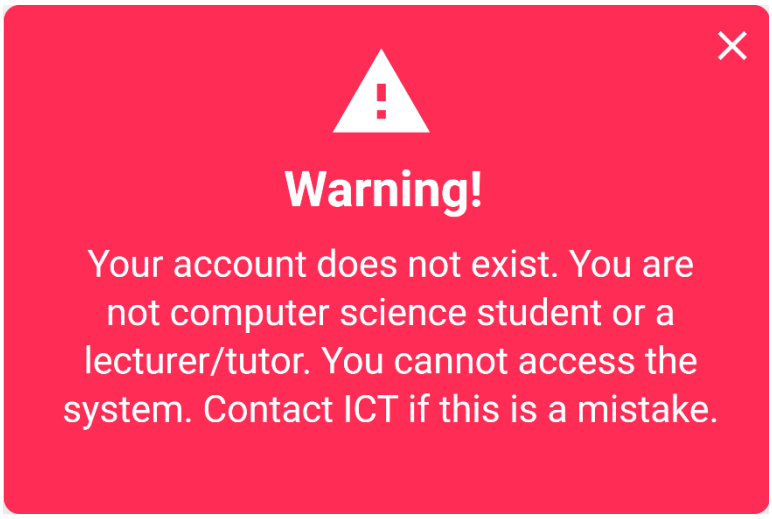


Figure 15: Account does not exist error

Alternative Scenario E

1. Kamo accesses the application.
2. System presents Kamo with a login page.
3. Kamo forgot his password, he wishes to change his password and clicks on the “Forgot password” button.
4. System responds with a popup stating: “Please reset your password using the PeopleSoft system.”.

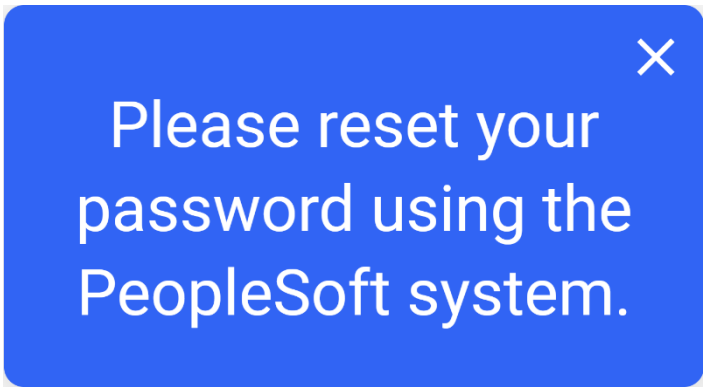


Figure 16: Reset Password Popup

View Modules

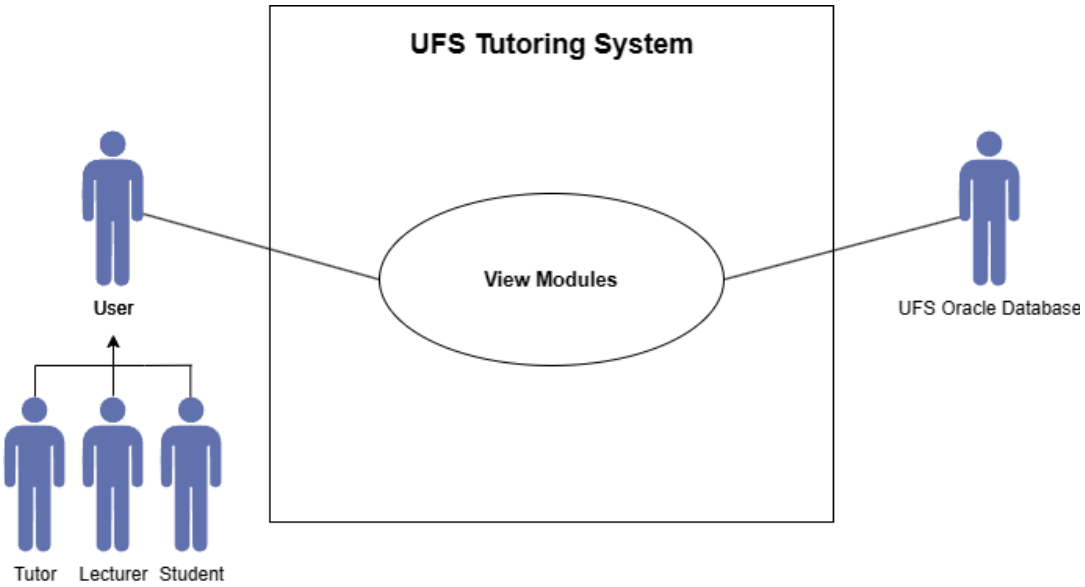


Figure 17: View Modules Use Case Diagram

Brief Description

The View Modules function allows students to view modules that they have registered for. Additionally, it allows lecturers to view modules that they teach, and it allows tutors to view modules that they will tutor for the semester.

Step-by-Step Description

For step 1 kindly refer to Figure 1-3 Dashboard.

1. User clicks on the courses button on the left pane of the dashboard.

For step 2 kindly refer to Figure 18 View Modules User Interface.

2. System takes them to the courses page. This page shows them all the modules that are relevant to them.
3. If the user is a tutor, they will see modules that they have been accepted to tutor for.
4. If the user is a lecturer, they will see all the modules that they are lecturing.
5. If the user is a student, they will see all the computer science modules they have been registered for.
6. User is able click in the module and the module specific information and features are displayed.

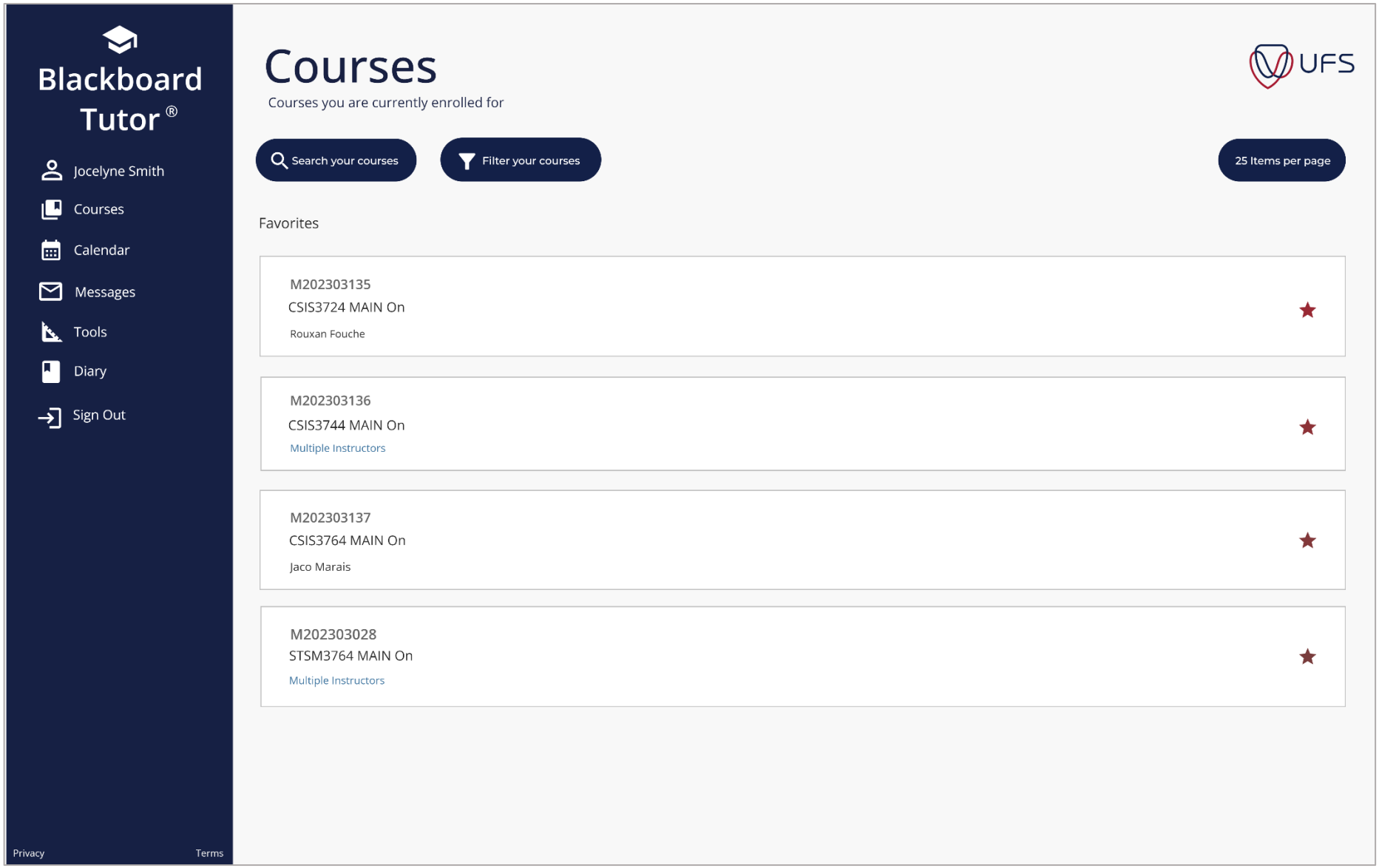


Figure 18: View Modules User Interface

Best-case Scenario

1. Chris (a student) is currently on the home page. Chris clicks on the courses button on the side menu.
2. System takes Chris to the courses page. This page shows Chris to view all the computer science modules that he is currently enrolled for.

Worst-case Scenario

1. Chris (a student) opens the systems dashboard.
2. Chris clicks the courses button.
3. System shows him the courses page. Chris is currently not enrolled for any modules. The system displays no modules to Chris.

Alternative Scenario A

1. Chris (a student) is currently on the home page. Chris clicks on the courses button on the side menu.
2. System takes him to the courses page.
3. Chris is a second-year student that has a failed a first-year module. The system only shows Chris the first-year module that he is registered for and not any of his second-year modules.

Alternative Scenario B

1. Tom (a lecturer) opens the dashboard.
2. Tom clicks the courses button.
3. System shows him the courses page, he can see all the modules that he must lecture for this semester.

Alternative Scenario C

1. Jocelyne (a tutor) opens the dashboard.
2. Jocelyne clicks the courses button.
3. System shows her the courses page, she sees the modules for which she has been accepted to tutor for the current semester.

Report Tutor

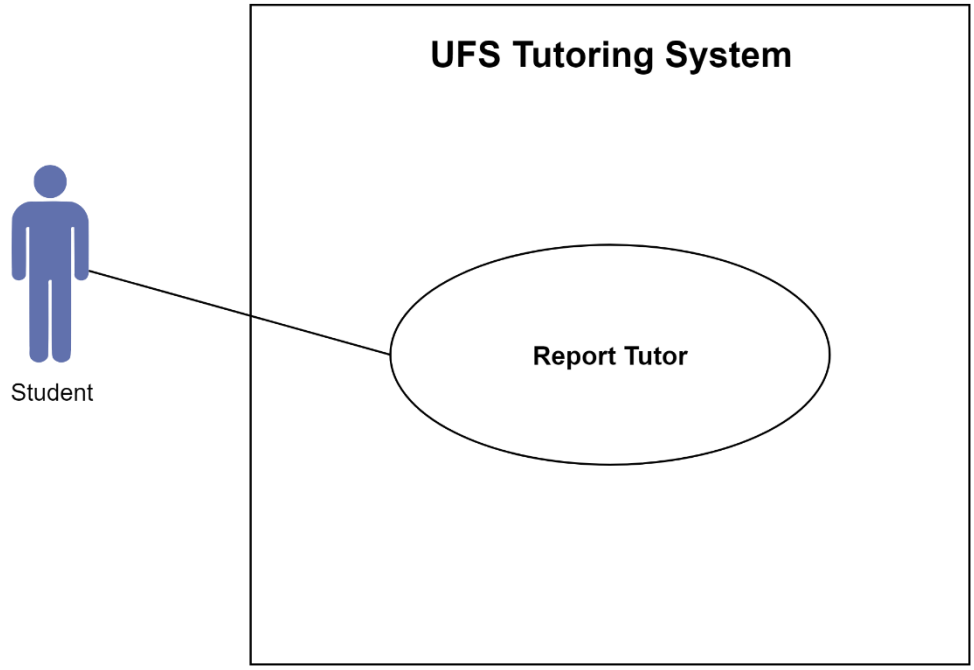


Figure 19: Report Tutor Use Case

Brief Description

The Report Tutor function allows a student to report a tutor for misconduct.

Step-By-Step Description

For steps 1-2 kindly refer to Figure 3 Student Dashboard.

1. Student clicks on the “Tools” button on the lefthand navigation pane.
2. Student clicks on the “Report Tutor” button on the “Tools” page.

For step 3 kindly refer to Figure 20 Report Tutor Popup.

3. The “Report Tutor” popup appears. It displays a dropdown menu containing: a list of the tutors that currently teach to the student, a description textbox where the student must enter a description of the offence that has occurred, a button to attach supporting documentation (e.g. image, or video), and an option to remain anonymous.

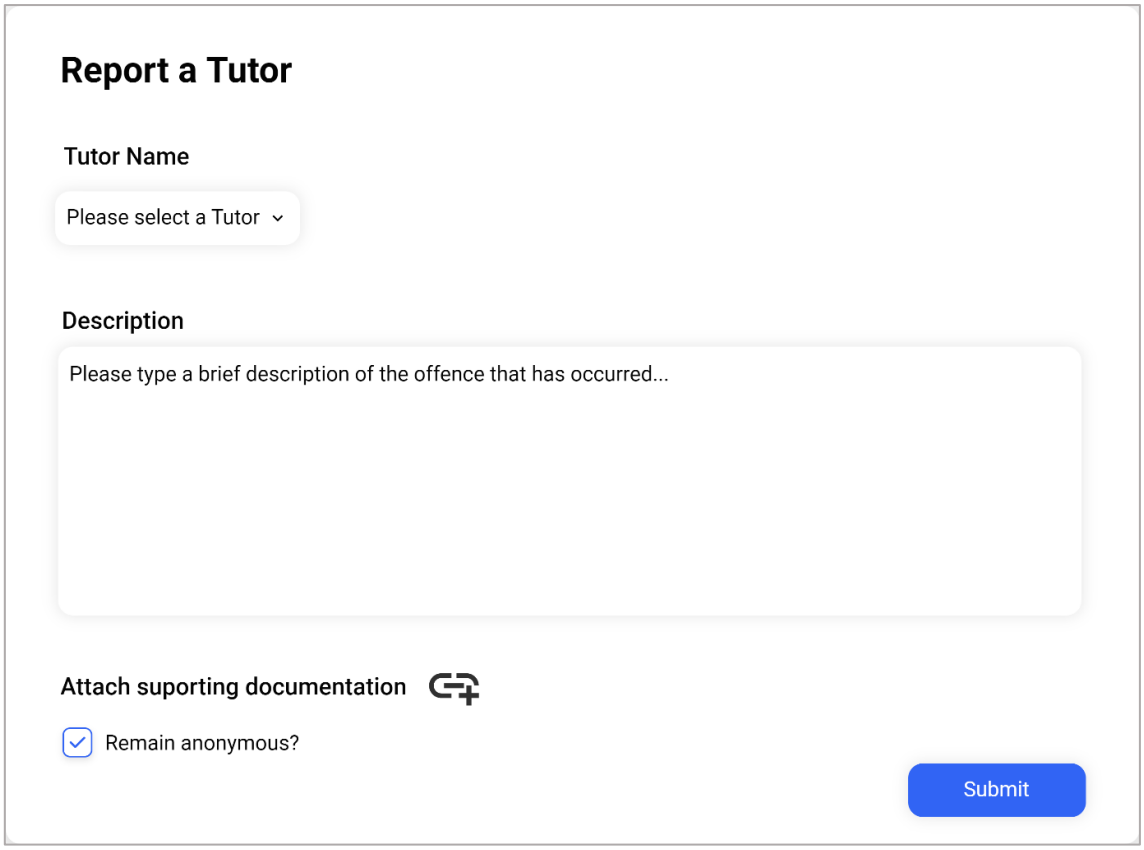


Figure 20: Report Tutor Popup

4. Student selects a tutor from the dropdown list.
5. Student enters a description of the offense that has occurred in the description textbox.
6. Student clicks the “Remain anonymous” checkbox.
7. Student clicks the submit button.
8. A confirmation popup is displayed stating: “Are you sure you want to submit this report?”.
9. Student clicks the “Yes” button.

For step 10-11 kindly refer to Figure 21 Confirmation Popup.

10. The confirmation popup and the “Report Tutor” popup both close.
11. A popup appears, stating: “Your report has been logged and a lecturer will be in contact with you soon”.

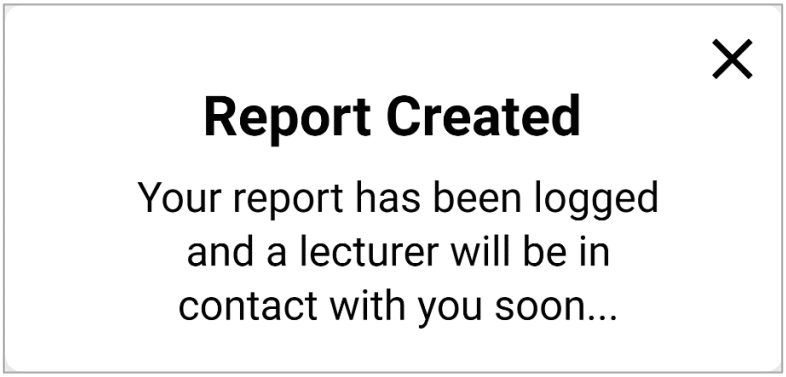


Figure 21: Confirmation Popup

Best-case Scenario

1. JP clicks on the “Tools” button on the lefthand navigation pane.
2. JP clicks on the “Report Tutor” button on the “Tools” page.
3. The “Report Tutor” popup appears.
4. JP selects a tutor from the dropdown list.
5. JP enters a description of the offense that has occurred in the description textbox.
6. JP checks “Remain anonymous” checkbox.
7. JP clicks the submit button.
8. A confirmation popup is displayed stating: “Are you sure you want to submit this report?”.
9. JP clicks the yes button.
10. The confirmation popup and the “Report Tutor” popup both close.
11. A popup appears, stating: “Your report has been logged and a lecturer will be in contact with you soon”.

We acknowledge that a student might back out of submitting a report at the last second. The system should therefore flag the report and save a copy for the lecturer to view, however for the purposes of this project we will not include this feature.

Worst-case Scenario

1. Riaan clicks on the “Tools” button on the lefthand navigation pane.
2. Riaan clicks on the “Report Tutor” button on the “Tools” page.
3. The “Report Tutor” popup appears.
4. Riaan selects a tutor from the dropdown list.
5. Riaan does not enter a description of the offense that has occurred in the description textbox.
6. Riaan checks the “Remain anonymous” checkbox.
7. Riaan clicks the submit button.
8. System does not allow Riaan to submit a report without a description. The system displays an error message above the description box, stating: “Please insert a description”.

Alternative Scenario

1. Darel clicks on the “Tools” button on the lefthand navigation pane.
2. Darel clicks on the “Report Tutor” button on the “Tools” page.
3. The “Report Tutor” popup appears.
4. Darel selects a tutor from the dropdown list.
5. Darel enters a description of the offense that has occurred in the description textbox.
6. Darel checks the “Remain anonymous” checkbox.
7. Darel accidentally clicks outside the “Report Tutor” popup.
8. A confirmation popup is displayed stating: “Are you sure you to leave?”.
9. Darel clicks the “Yes” button.
10. The “Report Tutor” popup closes.

View Report

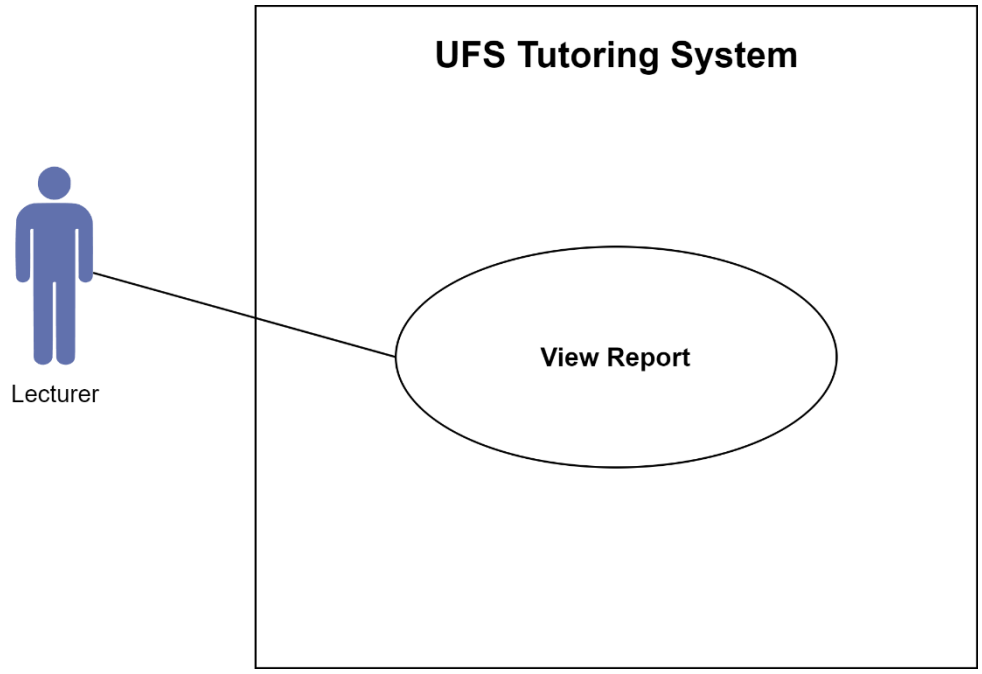


Figure 22: View Report Use Case Diagram

Brief Description

The View Report function allows lecturers to view a list of reports that students made against their tutors in the lecturer's modules. The lecturer can add comments and change the status of a specific report.

Step-By-Step Description

For steps 1-2 kindly refer to Figure 3 Student Dashboard.

1. Lecturer clicks on the "Tools" button on the lefthand navigation pane.
2. Lecturer clicks on the "Manage Reports" button on the "Tools" page.
3. The "Manage Reports" page opens with a list of reports. Each report has either the unique report ID (if the student submitted anonymously) or the student's full name (if they did not submit anonymously), the date the report was submitted and whether the report is pending or resolved.

For steps 3-5 kindly refer to Figure 23 Manage Reports User Interface.

4. Lecturer can click on the dropdown menu on the right-hand side of the page.

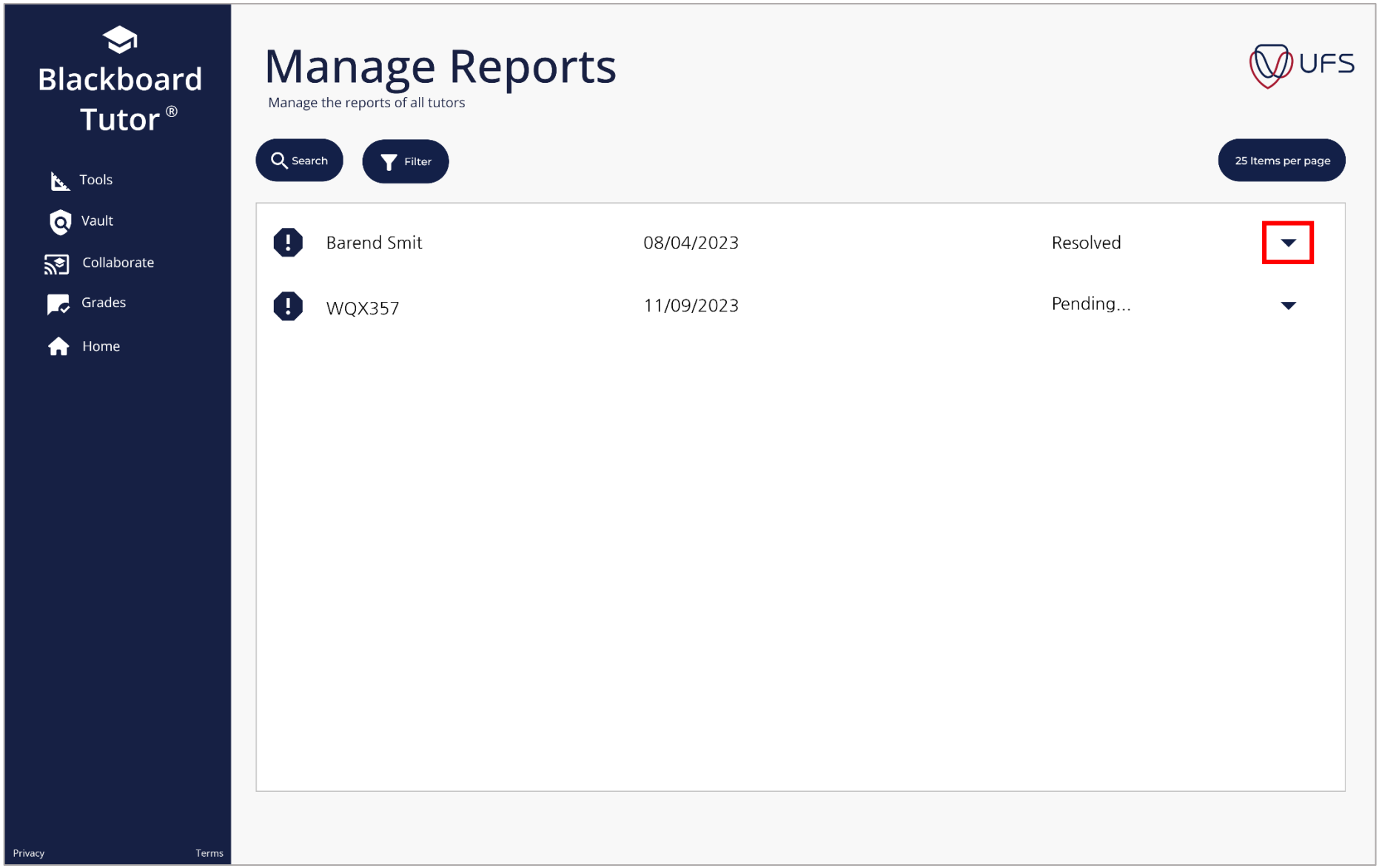


Figure 23: Manage Reports User Interface

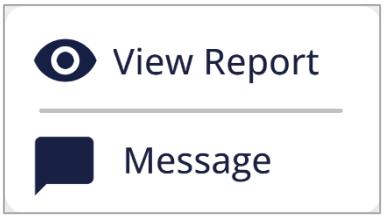


Figure 24: Manage Reports Popup

The above popup appears when the user clicks on the button in the red square on Figure 23 Manage Reports User Interface.

5. The context menu offers a “Message” feature (kindly see the “Direct Message” use case for more detail, specifically Figure 84 Lecturer communicating with Reporters) and a “View Report” feature.
6. Lecturer clicks on the “View Report” button.

For step 7 kindly refer to Figure 24 View Report Popup below.

7. A popup is displayed, including: the report ID (if the report was made anonymously) or the student’s name followed by the student number (if it was not made anonymously), the date the report was made, a list of all comments, a button to add a comment and a drop down box that allows a lecturer to change the status of the report (as either “Resolved” or “Pending”).

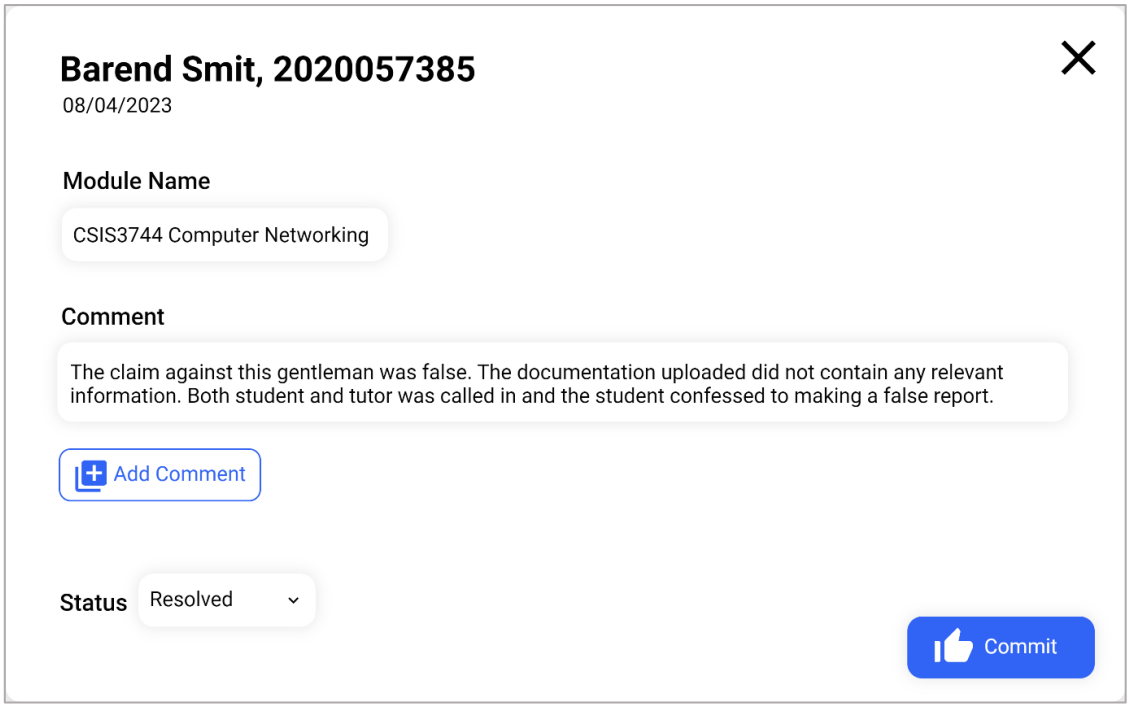


Figure 24: View Report Popup

8. Lecturer clicks on the status drop-down menu and selects resolved.
9. Lecturer clicks on the “Add Comment” button.

Figure 25: Add Comment to Report Popup

For step 10 kindly refer to Figure 25 Add Comment to Report Popup.

10. The above popup appears. It contains a textbox and an “Add” button.
    i. The lecturer types a comment.
    ii. The lecturer clicks on the “Add” button.
    iii. A confirmation popup appears, asking the lecturer if they are sure they want to add the comment.
    iv. The lecturer clicks the “Yes” button.
    v. The textbox popup closes, and the comment appears under the comment list.
11. Lecturer clicks the “Commit” button.
12. The popup closes.

Best-case Scenario

1. Professor Smith clicks on the “Tools” button on the lefthand navigation pane.
2. Professor Smith clicks on the “Manage Reports” button on the “Tools” page.
3. The “Manage Reports” page opens with a list of reports.
4. Professor Smith clicks on a specific report from the “View Reports” page.
5. A popup is displayed.
6. Professor Smith clicks on the drop-down menu and selects “Resolved”.
7. Professor Smith clicks the “Add comment” button.
8. Professor Smith types a comment in the comment textbox popup that appears.
9. Professor Smith clicks on the “Add” button on the popup, adding the comment to the comment list.
10. Professor Smith clicks the “Commit” button.
11. The popup closes.

Worst-case Scenario

1. Professor Smith clicks on the “Tools” button on the lefthand navigation pane.
2. Professor Smith clicks on the “Manage Reports” button on the “Tools” page.
3. The “Manage Reports” page opens with a list of reports.
4. Professor Smith clicks on a specific report from the “View Reports” page.
5. A popup is displayed.
6. Professor Smith clicks the “Add comment” button.
7. Professor Smith does not type in a comment in the comment textbox popup.
8. Professor Smith clicks on the “Add” button.
9. System responds with an error message: “Error! Please add a description”.

Alternative Scenario

1. Professor Smith clicks on the “Tools” button on the lefthand navigation pane.
2. Professor Smith clicks on the “Manage Reports” button on the “Tools” page.
3. The “Manage Reports” page opens with a list of reports.
4. Professor Smith clicks on a specific report from the “View Reports” page.
5. A popup is displayed.
6. Professor Smith clicks on x icon in the top right-hand corner of the popup.
7. System closes the popup.

Announcement

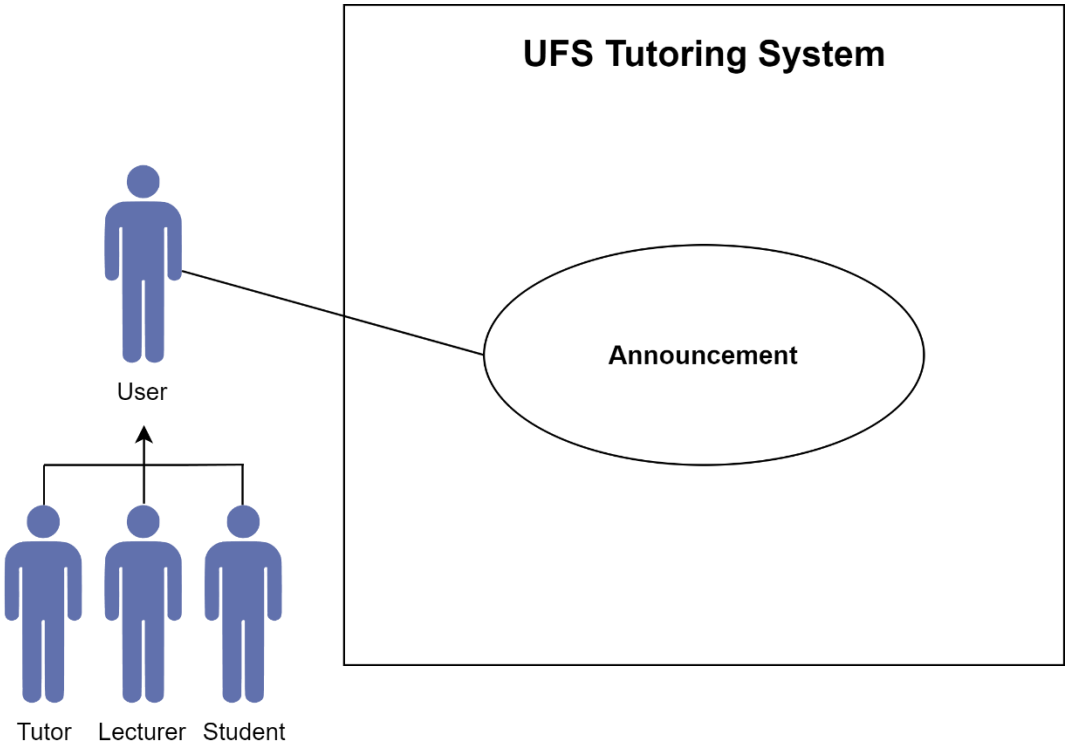


Figure 26: Announcement Use Case

Brief Description

The Announcement function allows a lecturer and a tutor to add, edit and delete announcements that are displayed on the dashboard for students in their modules to view. The currently active announcements are also displayed on the dashboards of all users for that specific module.

Step-By-Step Description

<u>If the user is a tutor/lecturer they can:</u>

1. Add Announcement:

For step i, kindly refer to Figure 1 Lecturer Dashboard.

i. User clicks the “Add Announcement” button on their dashboards.

For steps ii – v, kindly refer to Figure 27 Add Announcement Popup below.

ii. System displays a “Add Announcement”, including parameters for “Module Code” and “Description”.
iii. User populates the textboxes.
iv. User clicks on the “Post” button.
v. System posts the announcement making it visible on to all users that form part of that module dashboard under the “Announcements section”.

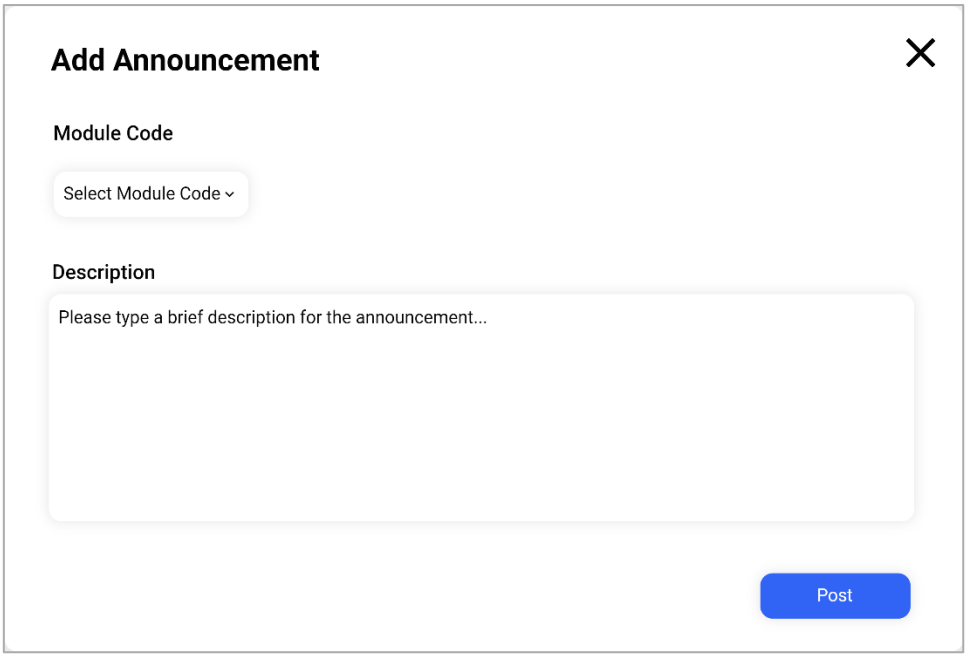


Figure 27: Add Announcement Popup

2. Edit Announcement:

For step i kindly refer to Figure 1 Lecturer Dashboard.

i. User clicks the “Edit” icon on the right-hand side of the announcement.

For step ii - iv kindly refer to Figure 28 Edit Announcement Popup below.

ii. System displays a “Edit Announcement”, including parameters for “Module Code” and “Description”.
iii. User changes the module code or description.
iv. User clicks on the “Post” button.
v. System displays the updated announcement under “Announcements section”.

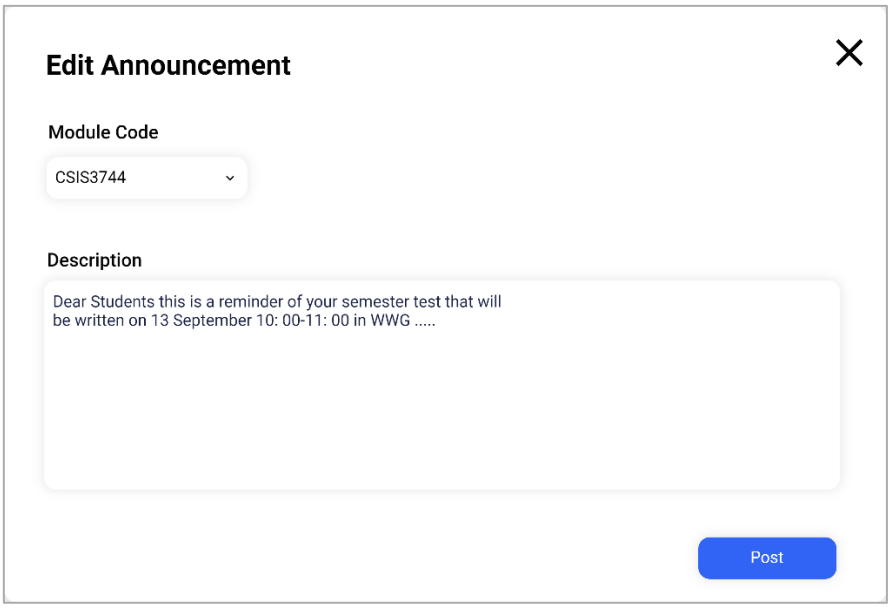


Figure 28: Edit Announcement Popup

3. Delete Announcement:

   For step i kindly refer to Figure 1 Lecturer Dashboard.

   i. User clicks the “Delete” icon on the right-hand side of the announcement.

   For step ii - iv kindly refer to Figure 29 Delete Announcement Popup below.

   ii. System displays a popup warning the user that they are about to delete an announcement.
   iii. User clicks on the “Delete” button.
   iv. System deletes the announcement and is no longer visible on any users’ dashboard.

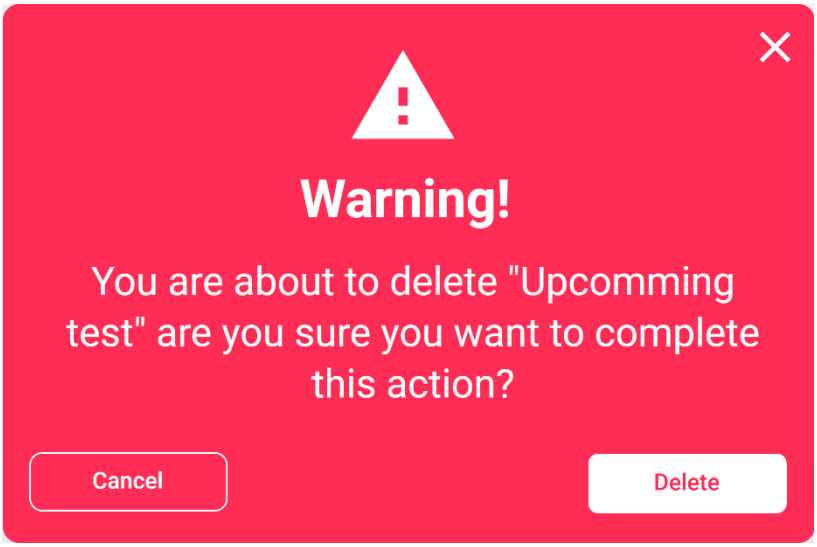


Figure 29: Delete Announcement Popup

<u>All users can:</u>

4. View announcement:

   i. The user can view currently active announcements on their dashboard.

Best-case Scenario

1. Pieter clicks the “Add Announcement” button on his dashboard.
2. System displays a “Add Announcement”, including parameters for “Module Code” and “Description”.
3. Pieter adds a module code and description.
4. Pieter clicks on the “Post” button.
5. The announcement is posted and is visible on every user’s dashboard under the “Announcements section”.

Worst-case Scenario

1. Pieter clicks the “Add Announcement” button on his dashboard.
2. System displays a “Add Announcement”, including parameters for “Module Code” and “Description”.
3. Pieter does not add a module code and description.
4. clicks the “Post” button.
5. An error is displayed stating: “Please add a module code and description”.

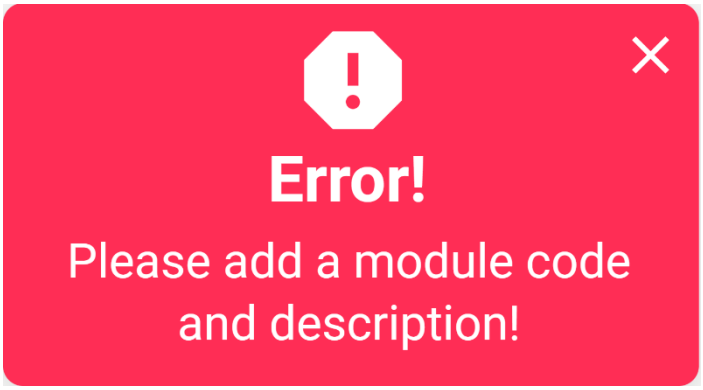


Figure 30: Add module code and description error

Alternative Scenario A

1. Pieter clicks the “Add Announcement” button on his dashboard.
2. System displays a “Add Announcement”, including parameters for “Module Code” and “Description”.
3. Pieter fills in the textboxes for the title and the description.
4. Pieter accidentally clicks on the “Close” button in the top right corner of the popup.
5. A warning popup appears, stating: “Are you sure you want to exit?”
6. Pieter clicks on the “No” button and continues.
7. Pieter makes the necessary additions.
8. Pieter clicks the “Post” button.
9. The announcement is posted and is visible on every user’s dashboard under the “Announcements section”.

Alternative Scenario B

1. Pieter clicks the “Add Announcement” button on his dashboard.
2. System displays a “Add Announcement”, including parameters for “Module Code” and “Description”.
3. Pieter does not fill in module code.
4. Pieter clicks on the “Post” button.
5. The system displays an error message: “Please add a module code!”.

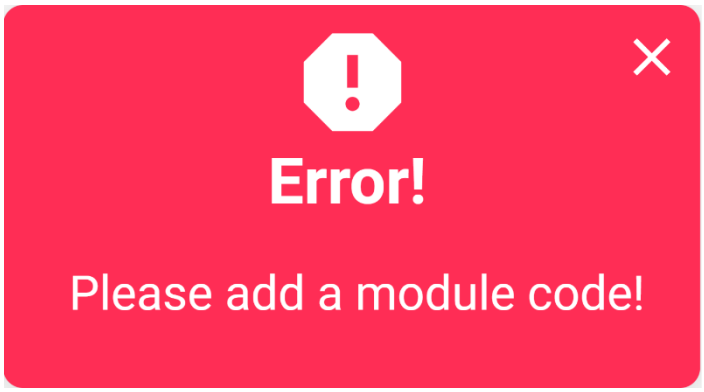


Figure 31: Add module code error

Alternative Scenario C

1. Pieter clicks the “Add Announcement” button on his dashboard.
2. System displays a “Add Announcement”, including parameters for “Module Code” and “Description”.
3. Pieter does not fill in a description.
4. Pieter clicks on the “Post” button.
5. The system displays an error message: “Please add a description!”.

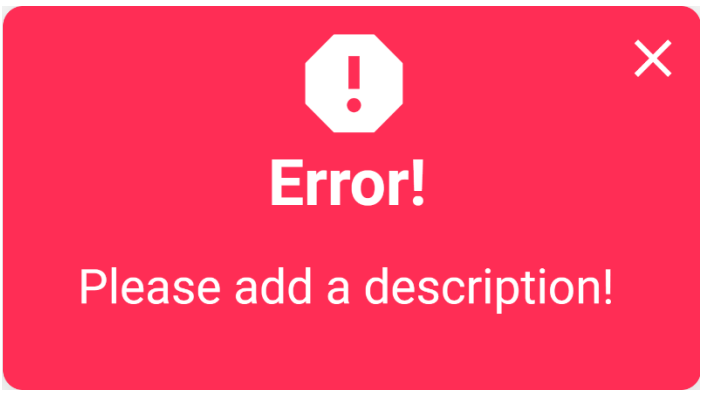


Figure 32: Add Description error

We acknowledge that there should be an “undo” button to undo a deleted announcement, however for the purposes of this assignment we will not include this feature.

Tutor Statistics

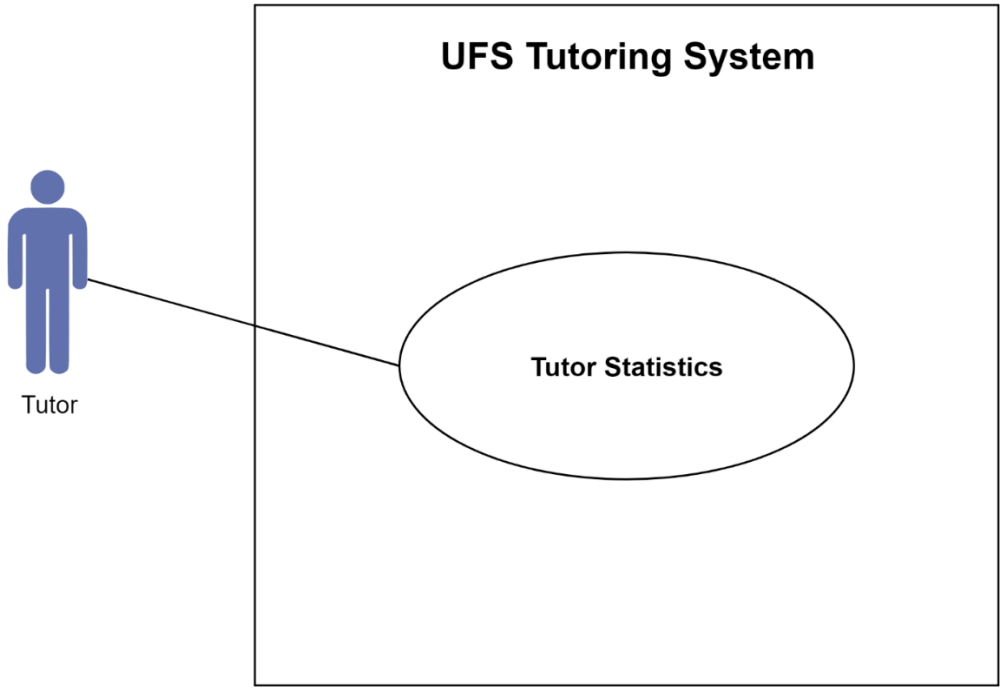


Figure 33: Tutor Statistics Use Case

Brief Description

The Tutor statistics function allows a tutor to view their average rating.

Step-By-Step Description

For step 1 kindly refer to Figure 2 Tutor Dashboard.

1. User clicks on the “Statistics” button on their dashboard.

For step 2 kindly refer to Figure 34 Tutor Statistics Popup below.

2. System displays a popup titled “Statistics”. The tutor will be able to see their average rating, converted to a percentage by the “Calculate Average” function as well as this percentage represented by a coloured bar which is determined by the “Determine Colour” function. Kindly see the Statistics CRC card and UML Card for the details of the “Calculate Average” and “Determine Colour” functions.

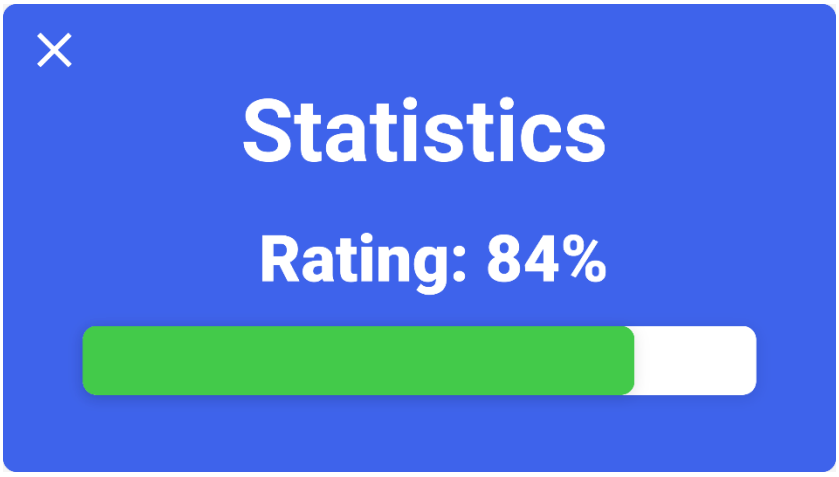


Figure 34: Tutor Statistics Popup

Best-case Scenario

1. Brad clicks on the statistics button on his dashboard.
2. A popup titled “Statistics” appears.
3. Brad views his average rating.
4. Brad clicks the close button in the top left to close the popup.
5. The popup closes and Brad’s dashboard is back in focus.

Worst-case Scenario

1. Brad clicks on the statistics button on his dashboard.
2. A popup titled “Statistics” appears.
3. Brad reads his average rating. While he’s reading, a student rates him for a tutoring session he just completed.
4. The Statistics popup does not display the new information, but the Calculate Average function and Determine Colour function have already been executed in the background.
5. Brad clicks the close button.
6. Brad clicks on the “Statistics” button again.
7. His rating has been updated to reflect the new ratings he received.
8. Brad clicks the close button.
9. The popup is closed.

Alternative Scenario

1. Brad clicks on the statistics button on his dashboard.
2. A popup titled “Statistics” appears.
3. Brad is a new tutor who has not received a rating yet. The popup displays a message stating: “No rating received yet.”
4. Brad clicks the close button.
5. The popup is closed.

Student Statistics

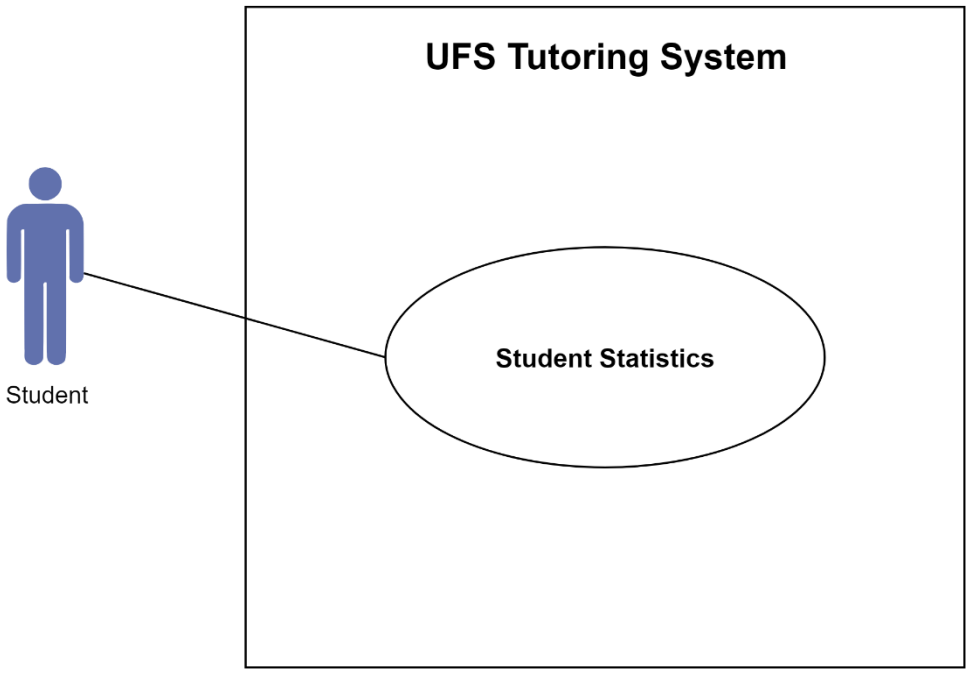


Figure 35: Student Statistics Use Case Diagram

Brief Description

The Student Statistics function allows a student to view their average quiz mark for and their average attendance rate.

We acknowledge that tutors and lecturers should also be able to view the attendance rate and quiz averages for their modules, however for the purposes of this project we will not include this feature.

Step-By-Step Description

For step 1 kindly refer to Figure 3 Student Dashboard.

1. User clicks on the “Statistics” button on their dashboard.

For step 2 kindly refer to Figure 36 Student Statistics Popup below.

2. System displays a popup titled “Statistics”. The user will be able to see their average quiz mark and their average attendance rate converted to a percentage by the “Calculate Average” function as well as this percentage represented by a coloured bar which is determined by the “Determine Colour” function. Kindly see the Statistics CRC card and UML Card for the details of the “Calculate Average” and “Determine Colour” functions.

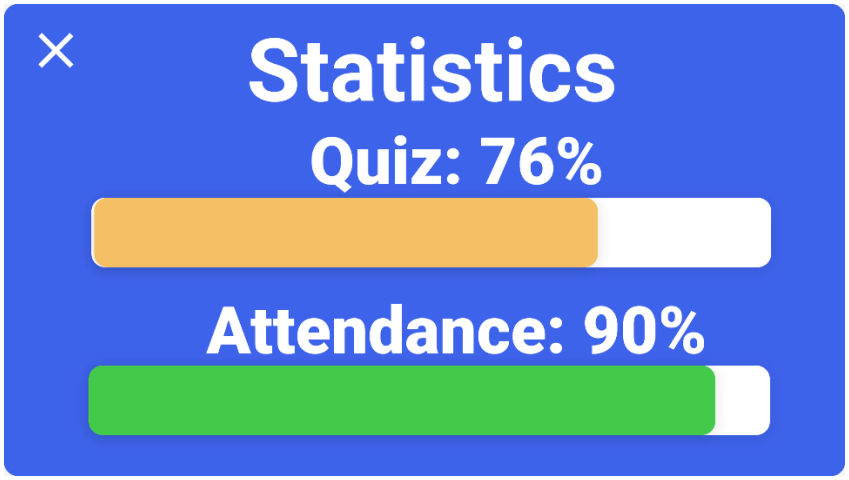


Figure 36: Student Statistics Popup

Best-case Scenario

1. Jenney clicks on the statistics button on her dashboard.
2. A popup titled “Statistics” appears.
3. Jenney views her average quiz mark and her average attendance rate.
4. Jenny clicks the close button.
5. The popup closes.

Worst-case Scenario

1. Jenney clicks on the statistics button on her dashboard.
2. A popup titled “Statistics” appears.
3. Jenney reads her average quiz performance and her attendance rate. While she’s reading it, she completes a quiz in another tab.
4. The Statistics popup does not display the new information, but the Calculate Average function and Determine Colour function have already been executed in the background.
5. Jenney clicks the close button.
6. Jenney clicks on the statistics button again.
7. Her average quiz performance has been updated to reflect the new ratings she received.
8. Jenny clicks the close button.
9. The popup closes.

Alternative Scenario

1. Jenney clicks on the “Statistics” button on her dashboard.
2. A popup titled “Statistics” appears.
3. Jenny is a new student who has not yet completed a quiz or attended a virtual session. The popup displays a message stating: “No statistics to display.”
4. Jenny clicks the close button.
5. The popup closes.

Grades

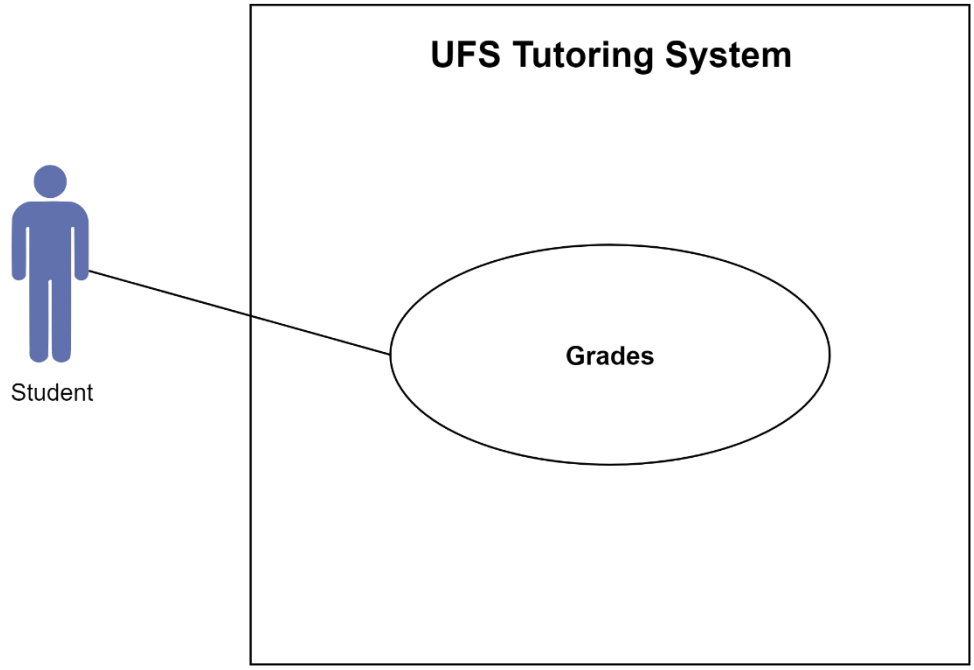


Figure 38: Grades Use Case

Brief Description

The Grades function allows a student to view the mark they achieved for quizzes they have completed for a specific module.

We acknowledge that tutors and lecturers should also be able to view student grades, however for the purposes of this project we will not include this feature.

Step-By-Step Description

For step 1 kindly refer to Figure 3 Student Dashboard.

1. User clicks the “Grades” button on the side-menu.

For step 2 kindly refer to Figure 37 Student Grades User Interface below.

2. System displays all the quizzes that they have taken for that module, including the quiz’s name, the difficulty of the quiz and the mark they achieved. The Determine Colour function is used to determine what colour should be displayed behind the mark of the quiz. Kindly see the Statistics CRC card and UML Card for the details of the “Determine Colour” function.

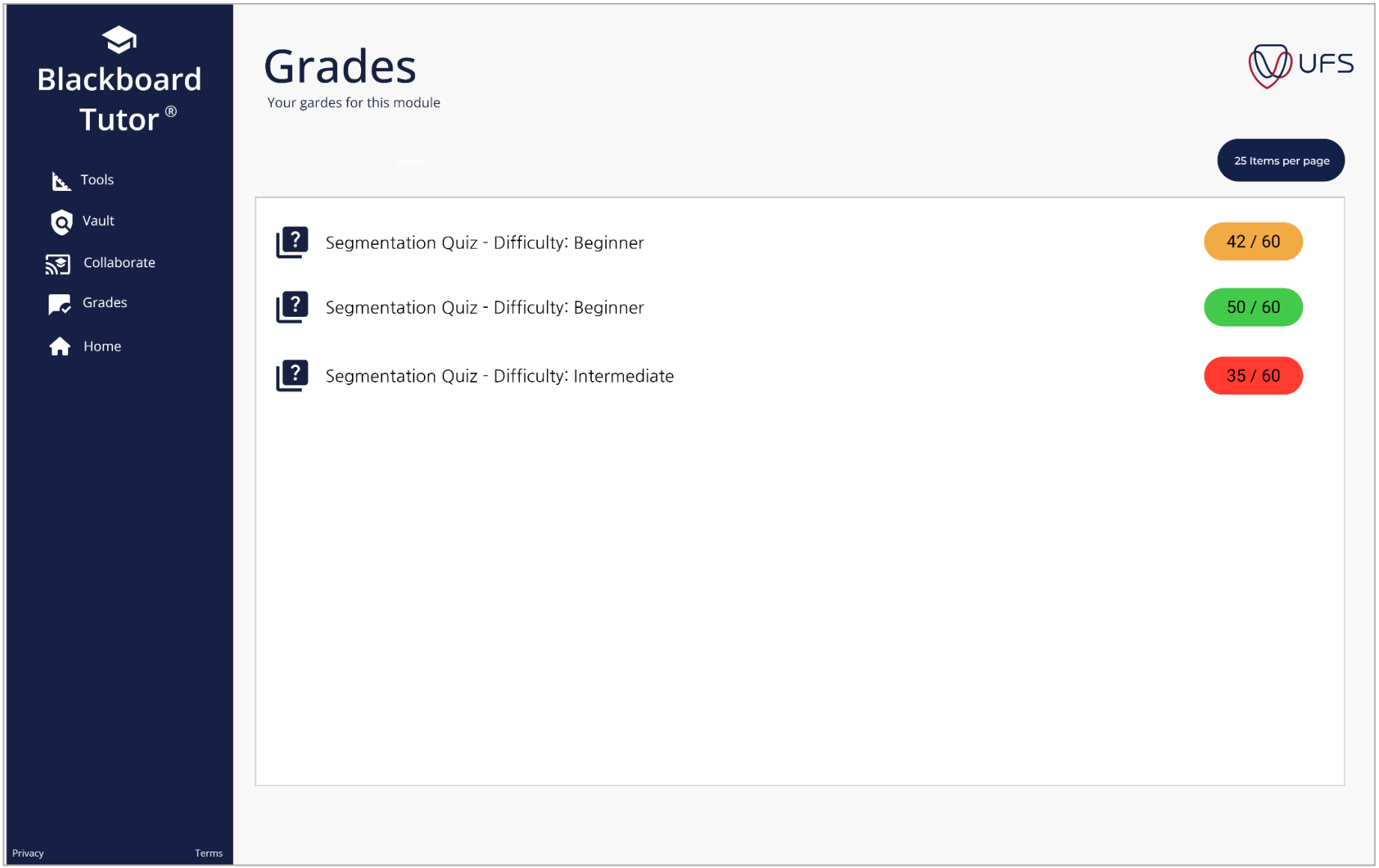


Figure 37: Student Grades User Interface

Best-case Scenario

1. Rob selects a module and clicks on the “Grades” button on the side-menu.
2. Rob can view his grades for all the quizzes he’s completed in that module.

Worst-case Scenario

1. Rob selects a module and clicks on the “Grades” button on the side-menu.
2. Rob has not written a quiz for that module yet.
3. The grade page displays no quizzes, and instead displays text stating: “You have not written any quizzes for this module. Come back when you have completed a quiz to view your marks.”.

Alternative Scenario

1. Rob is completing a quiz.
2. Rob did not answer all the questions by the time the quiz timer expires.
3. Rob selects the module and clicks on the “Grades” button on the side-menu.
4. The mark for the quiz is displayed. For all questions that were unanswered, the system awarded a 0 mark.

View Profile

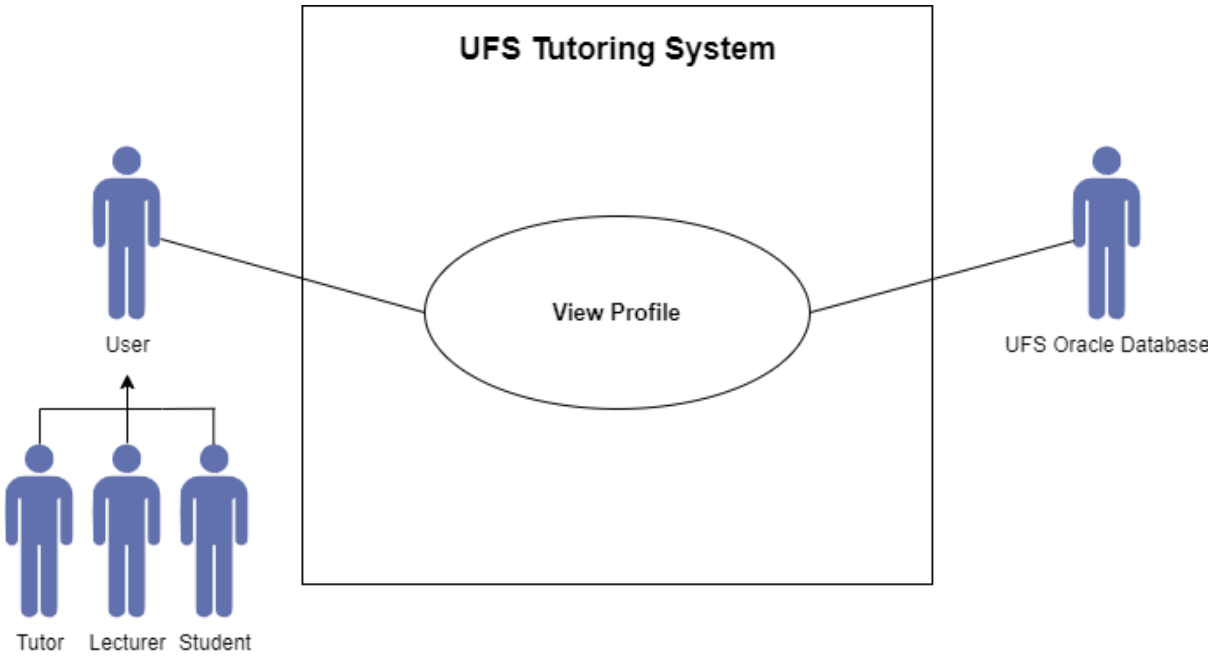


Figure 38: View Profile Use Case

Brief Description

The View Profile function allows a user to view their profile details, including their full name, student number/staff number, and email.

Step-By-Step Description

For step 1 kindly refer to Figure 1-3 Dashboards.

1. User clicks their full name on the side-menu.

For step 2 kindly refer to Figure 39 Student/Tutor Profile User Interface and Figure 40 Lecturer Profile User Interface below.

2. System displays the user's profile, which includes their full name, student number if they are a student or tutor and staff number if they are a lecturer.

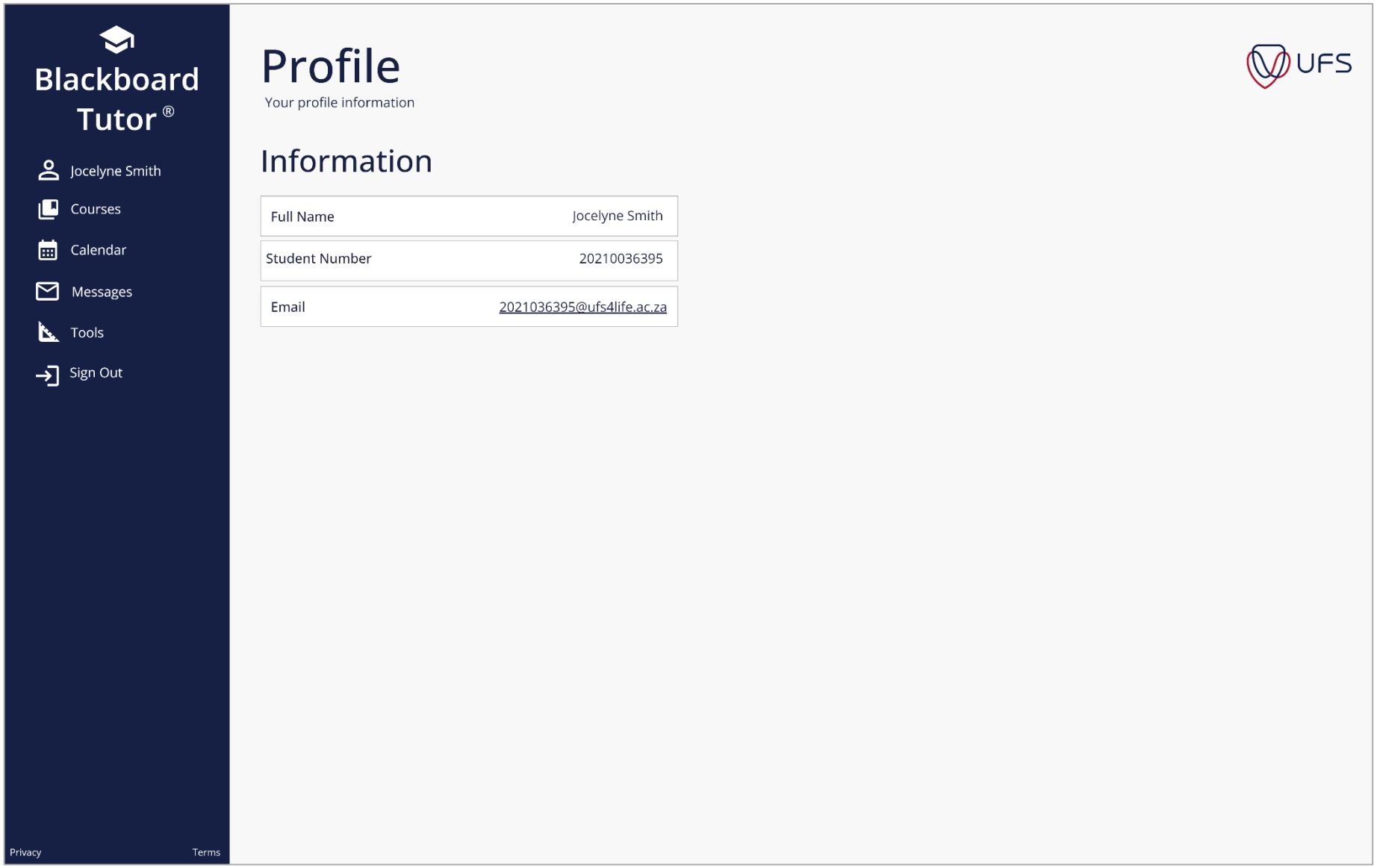


Figure 39: Student/Tutor Profile User Interface

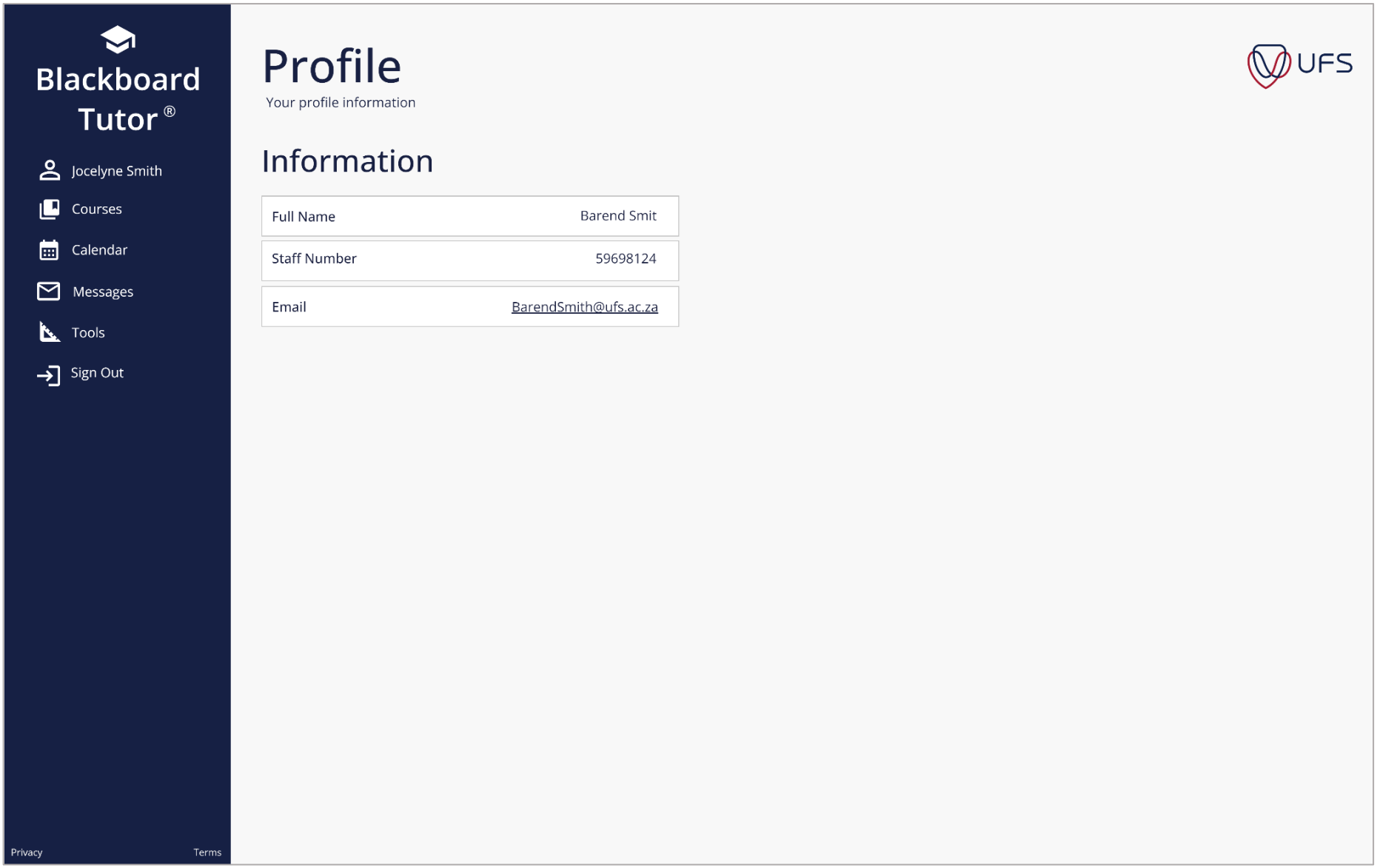


Figure 40: Lecturer Profile User Interface

Best-case Scenario

1. Sarah clicks on her full name on the side-menu.
2. Sarah is taken to the “View Profile” page.
3. Sarah views her information.

Worst-case Scenario

1. Sarah has changed her last name on the Oracle UFS database.
2. Sarah clicks on her full name on the left side-menu.
3. Sarah is taken to the “View Profile” page.
4. Sarah’s can view her information.
5. Her last name is updated to reflect the changes on the Oracle UFS database.

Vault

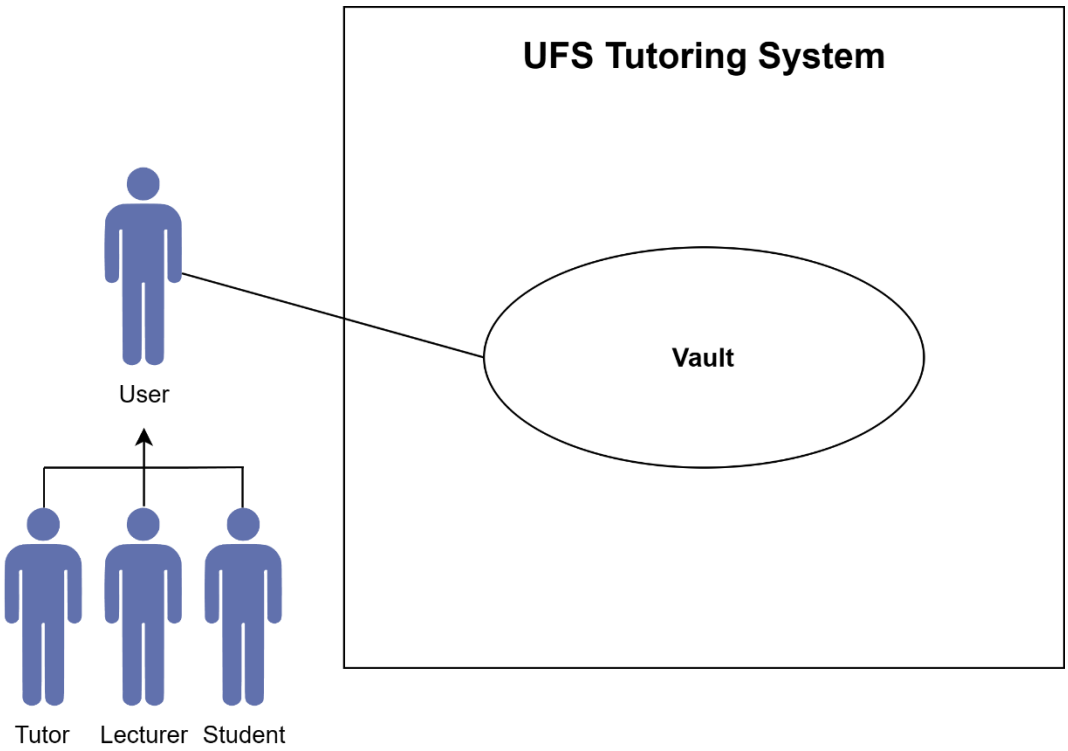


Figure 41: Vault Case Diagram

Brief Description

The Vault function allows students to view and download uploaded study material. It also allows tutors and lecturers to view, download, upload, edit and delete study material.

Step-By-Step Description

For steps 1-2 kindly refer to Figure 1-3 Dashboards.

1. User selects a module.
2. User clicks on the “Vault” button on the side-menu.
3. A list of uploaded content will appear on the page, including the content’s name, the size of uploaded material and an arrow at the right-hand side.
4. User clicks the arrow, and a context menu pops up. The options available will differ depending on if the user is a student or a tutor/lecturer.
5. View item.
   i. User clicks on the “View” button.
   ii. System displays the item in a new browser tab.
6. Download item.
   i. User clicks on the “Download” button.
   ii. System downloads the item to the vault folder on the user’s device. For more details, kindly see offline access/download under project proposal.

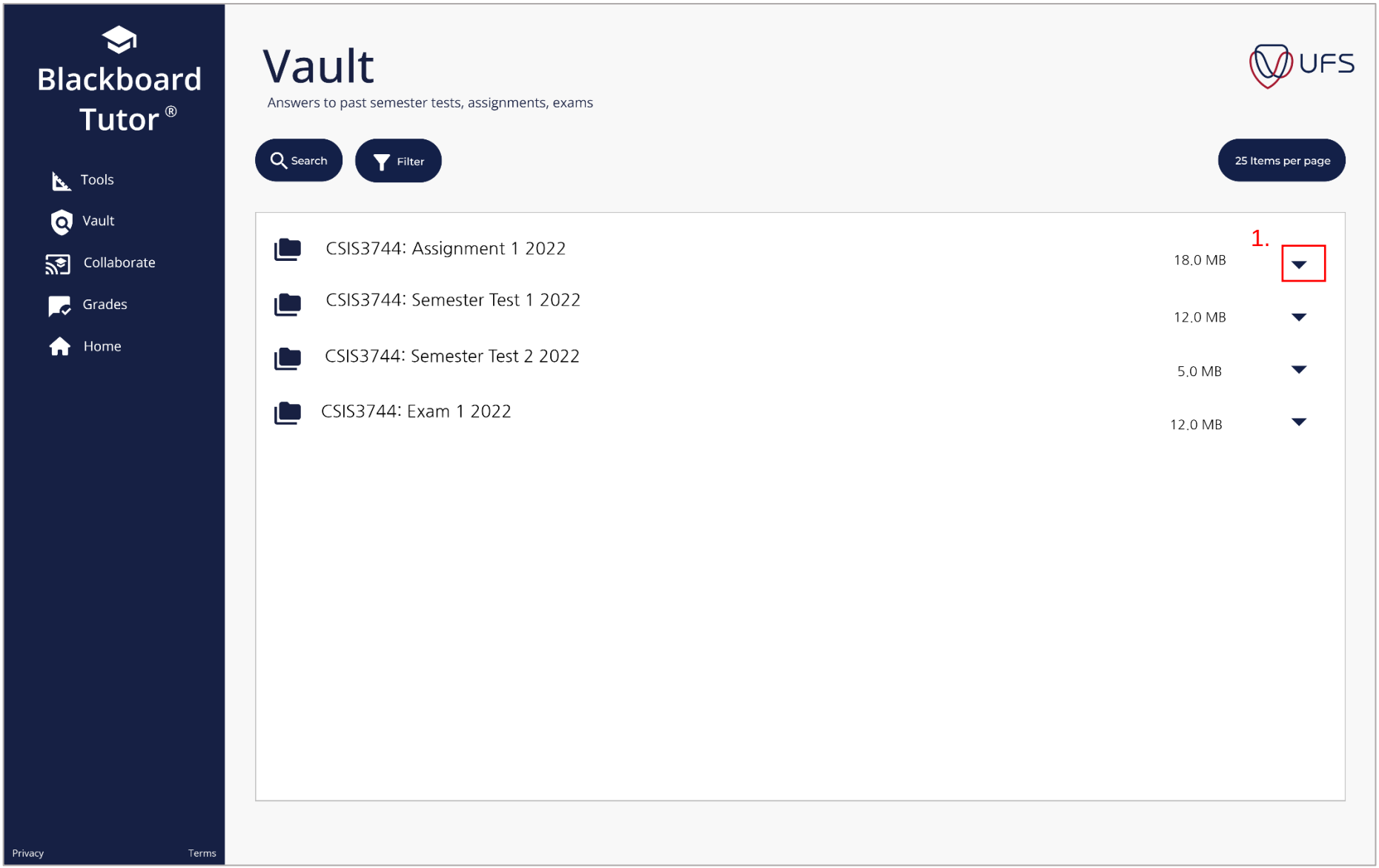


Figure 42: Student Vault User Interface

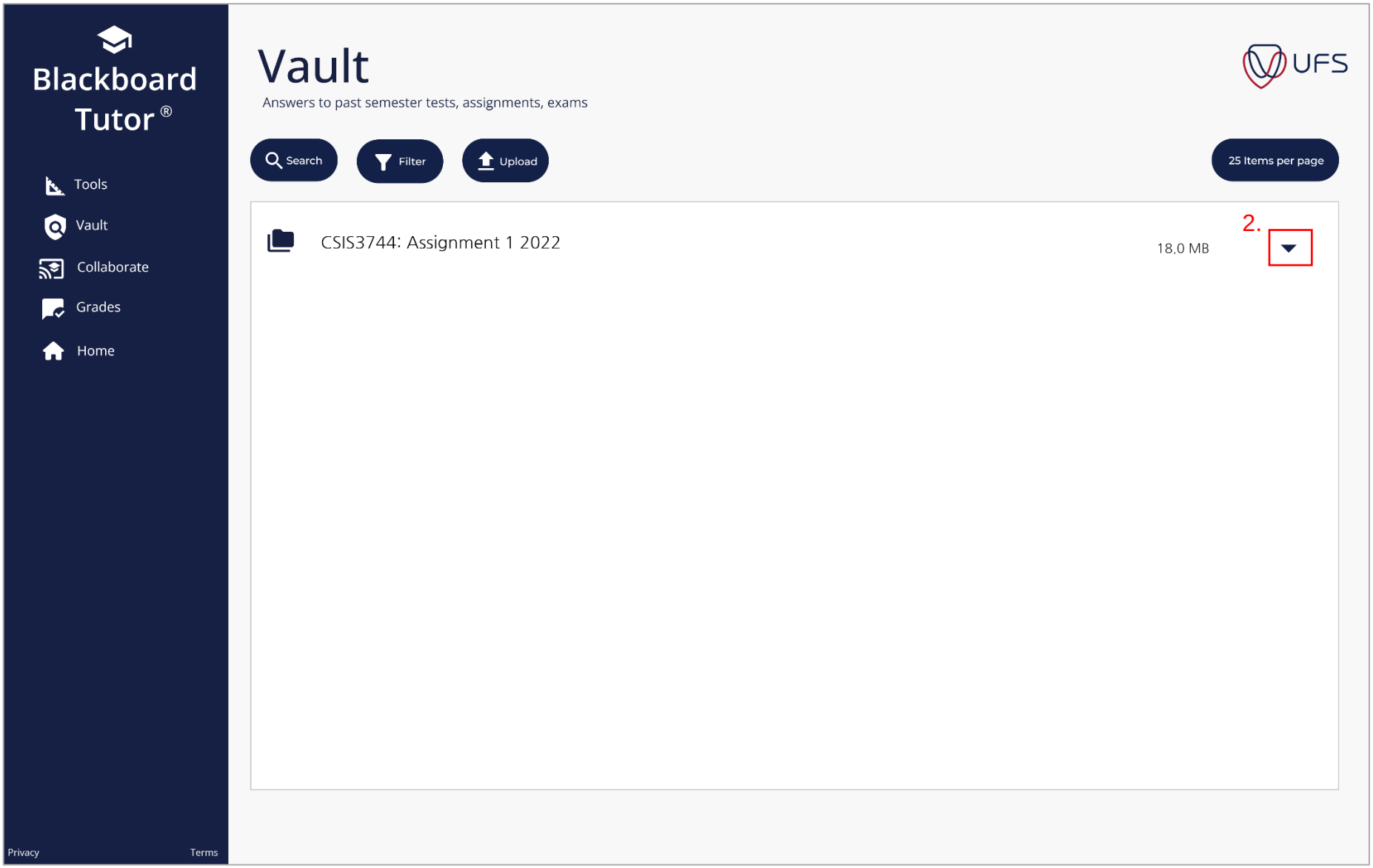


Figure 43: Lecturer/Tutor Vault User Interface

The below form popup when a tutor/lecturer clicks on the down arrow as indicated in 1.

1.

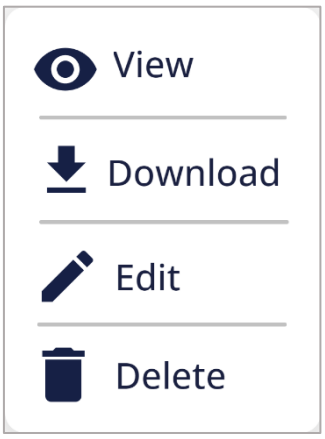


Figure 44: Lecturer/Tutor Popup

The below form popup when a student clicks on the down arrow as indicated in 2.

2.

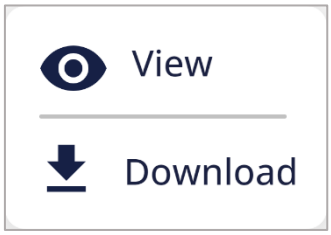


Figure 45: Student Popup

For the below kindly refer to Figure 43 Lecturer/Tutor Vault User Interface above.

<u>If the user is a tutor/lecturer, they can:</u>

7. Edit Resource.
    i. User clicks on the “Edit” button.
    ii. System displays a populated popup form with adjustable parameters titled “Change an item in vault”. The parameters are name and upload file.

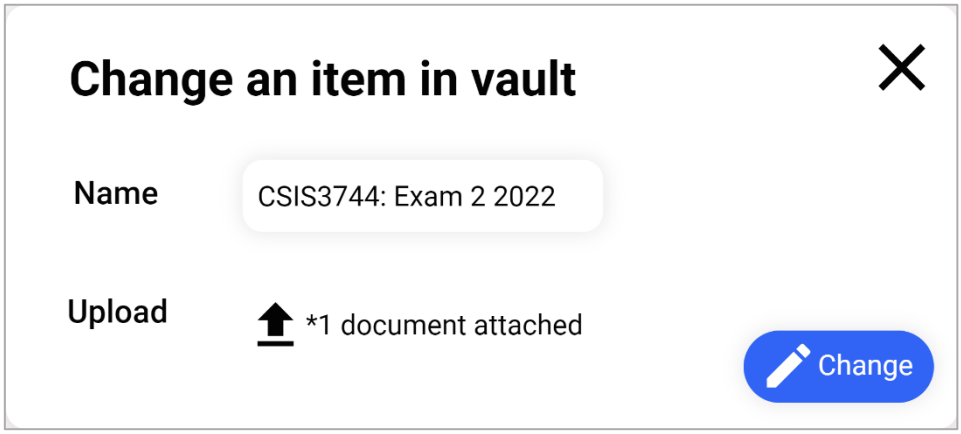


Figure 46: Edit Study Item Popup

8. Delete Item.
    i. User clicks on the “Delete” button.
    ii. System creates a delete popup message stating: You are about to delete an item are you sure you want to complete this action?”.
    iii. User clicks the “Delete” button.

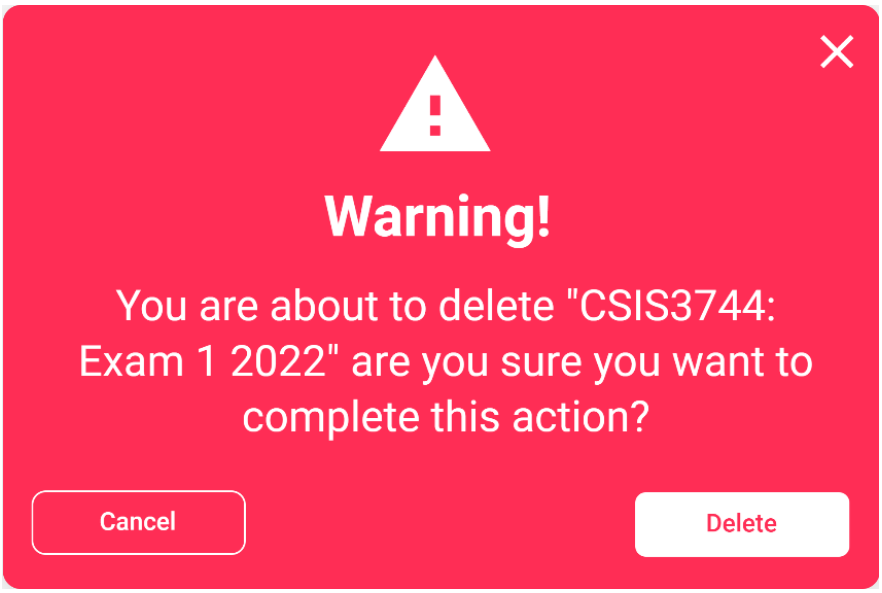


Figure 47: Delete Study Item Popup

9. Add Item.
    i. User clicks on the “Upload” button.
    ii. System displays a popup form with adjustable parameters titled “Add an item to vault”. The parameters are name and upload file.

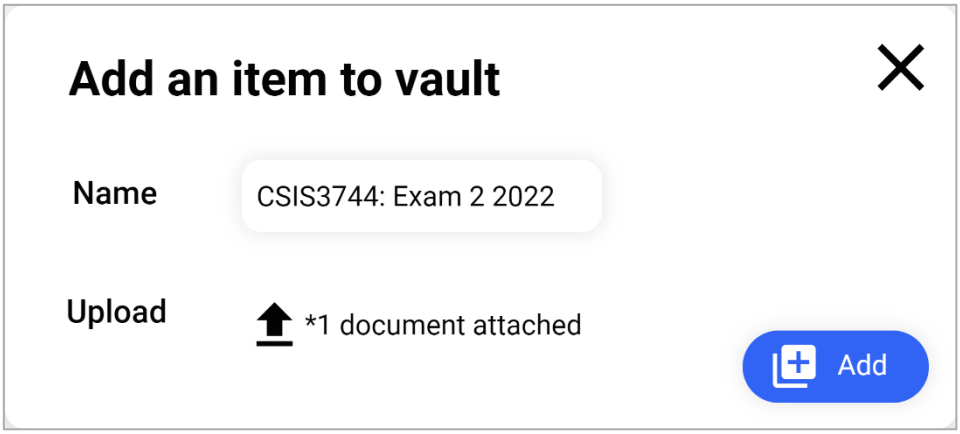


Figure 48: Add Study Item Popup

Best-case Scenario

1. Daniel (a student) selects a module and clicks on the “Vault” button on the side-menu.
2. Daniel sees a list of items in the vault.
3. Daniel clicks on the arrow on the right-hand side of the item.
4. Daniel clicks on the “View” option in the context menu.
5. The context menu closes.
6. The file he selected was a pdf. Since pdfs can be opened in the browser, it takes him to a new tab to view the pdf.
7. He closes the tab.
8. He’s back on the Vault page.

Worst-case Scenario

1. Dr Strydom selects a module and clicks on the “Vault” button on the side-menu.
2. Dr Strydom sees a list of items in the vault.
3. Dr Strydom clicks on the “Upload” button on the top of the page.
4. Dr Strydom selects more than 1 file to upload.
5. Dr Strydom clicks on the “Add” button.
6. The item is not added. Instead, an error message is displayed stating: “Error! You cannot upload more than 1 file!”

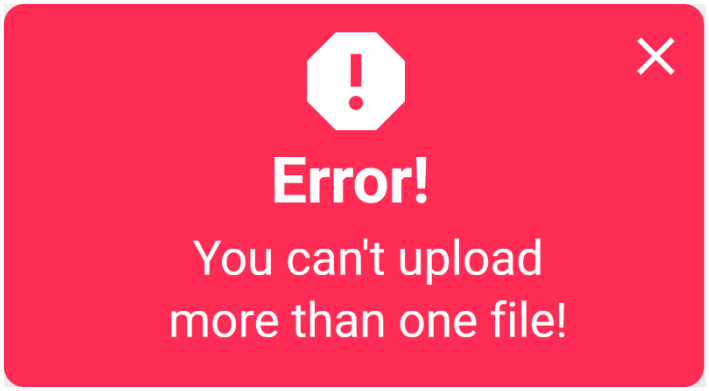


Figure 49: Can't upload more than 1 file error

Alternative Scenario A

1. Dr Strydom selects a module and clicks on the “Vault” button on the side-menu.
2. Dr Strydom sees a list of items in the vault.
3. Dr Strydom clicks on the “Upload” button on the top of the page.
4. Dr Strydom does not enter a name.
5. Dr Strydom clicks on the “Add” button.
6. The item is not added. Instead, an error message is displayed stating: “Error! Please add a name!”.

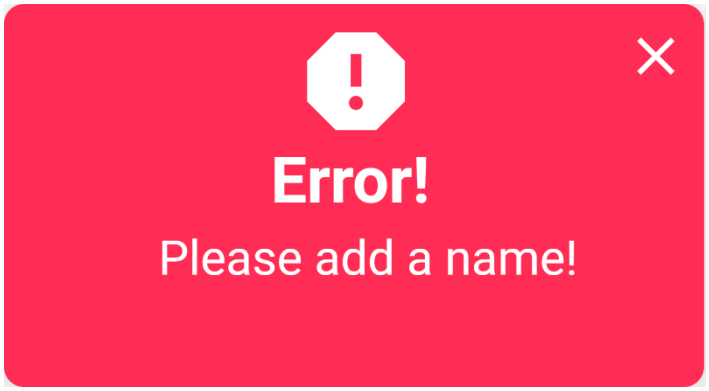


Figure 50: Add Name error

Alternative Scenario B

1. Dr Strydom selects a module and clicks on the “Vault” button on the side-menu.
2. Dr Strydom sees a list of items in the vault.
3. Dr Strydom clicks on the “Upload” button on the top of the page.
4. Dr Strydom does not enter a description.
5. Dr Strydom clicks on the “Add” button.
6. The item is not added. Instead, an error message is displayed stating: “Error! Pease add a description!”.

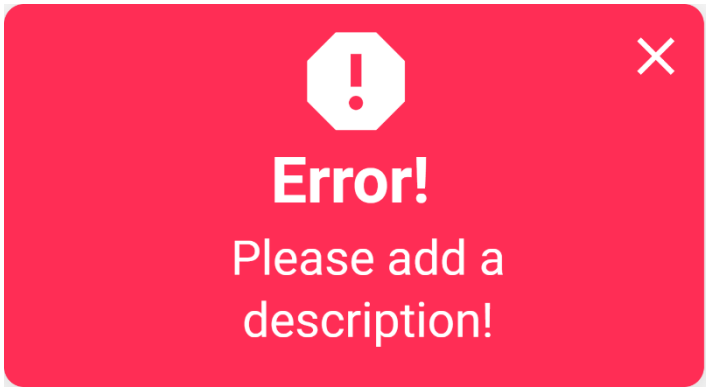


Figure 51: Add Description error

Alternative Scenario C

1. Daniel (a student) selects a module and clicks on the “Vault” button on the side-menu.
2. Daniel sees a list of items in the vault.
3. Daniel clicks on the arrow on the right-hand side of the item.
4. Daniel clicks on the “View” option in the context menu.
5. The context menu closes.
6. The file he selected was an excel file. Excel cannot be opened in the browser/ The system checks if it has already created a “View offline” folder. Since it has, the file is downloaded into the ‘Vault’ subfolder in the file.
7. The file is automatically opened after it was downloaded.

Alternative Scenario D

1. Daniel (a student) selects a module and clicks on the “Vault” button on the side-menu.
2. Daniel sees a list of items in the vault.
3. Daniel clicks on the arrow on the right-hand side of the item.
4. Daniel clicks on the “View” option in the context menu.
5. The context menu closes.
6. The file he selected was an excel file. Excel cannot be opened in the browser. The system checks if it has already created a “View offline” folder. Since it has not, the system creates a dedicated folder, and the file is downloaded into the ‘Vault’ subfolder.
7. The file is automatically opened after it was downloaded.

Apply to be a Tutor

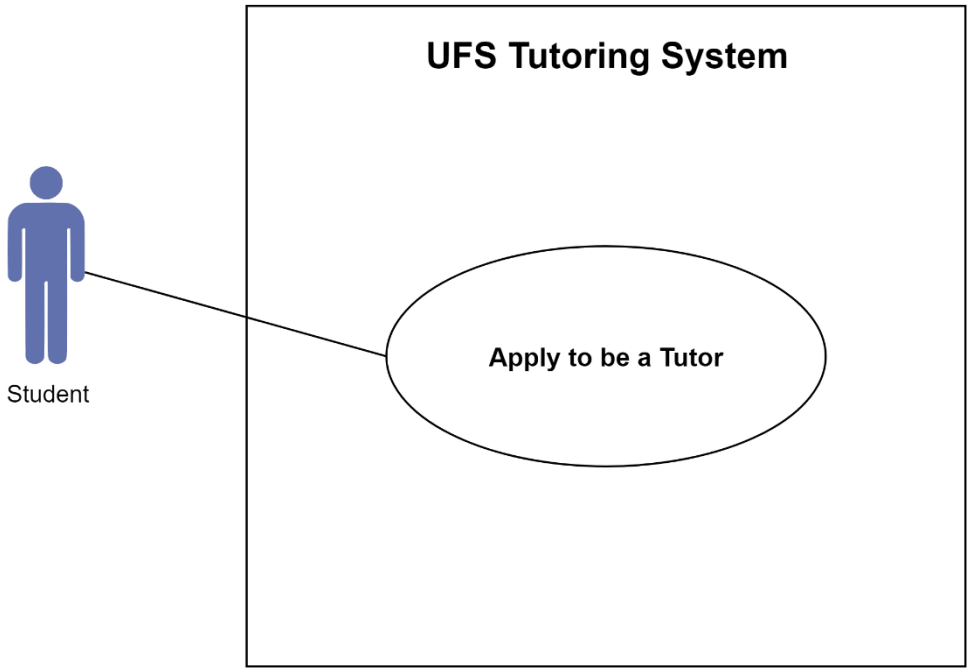


Figure 52: Apply to be a Tutor Use Case

Brief Description

The Apply to be a Tutor function allows a student who is a 2nd/3rd year computer science student (and who have passed all their first-year modules successfully) to apply to be a tutor for a specific first year module.

Step-By-Step Description

For step 1 kindly refer to Figure 53 Eligible Tutor Dashboard.

1. A 2nd/3rd year student clicks on the “Apply to be a tutor” button on their tools page.

The following page is presented when a 2nd/3rd student logs into the system since these students do not have 1st year modules.

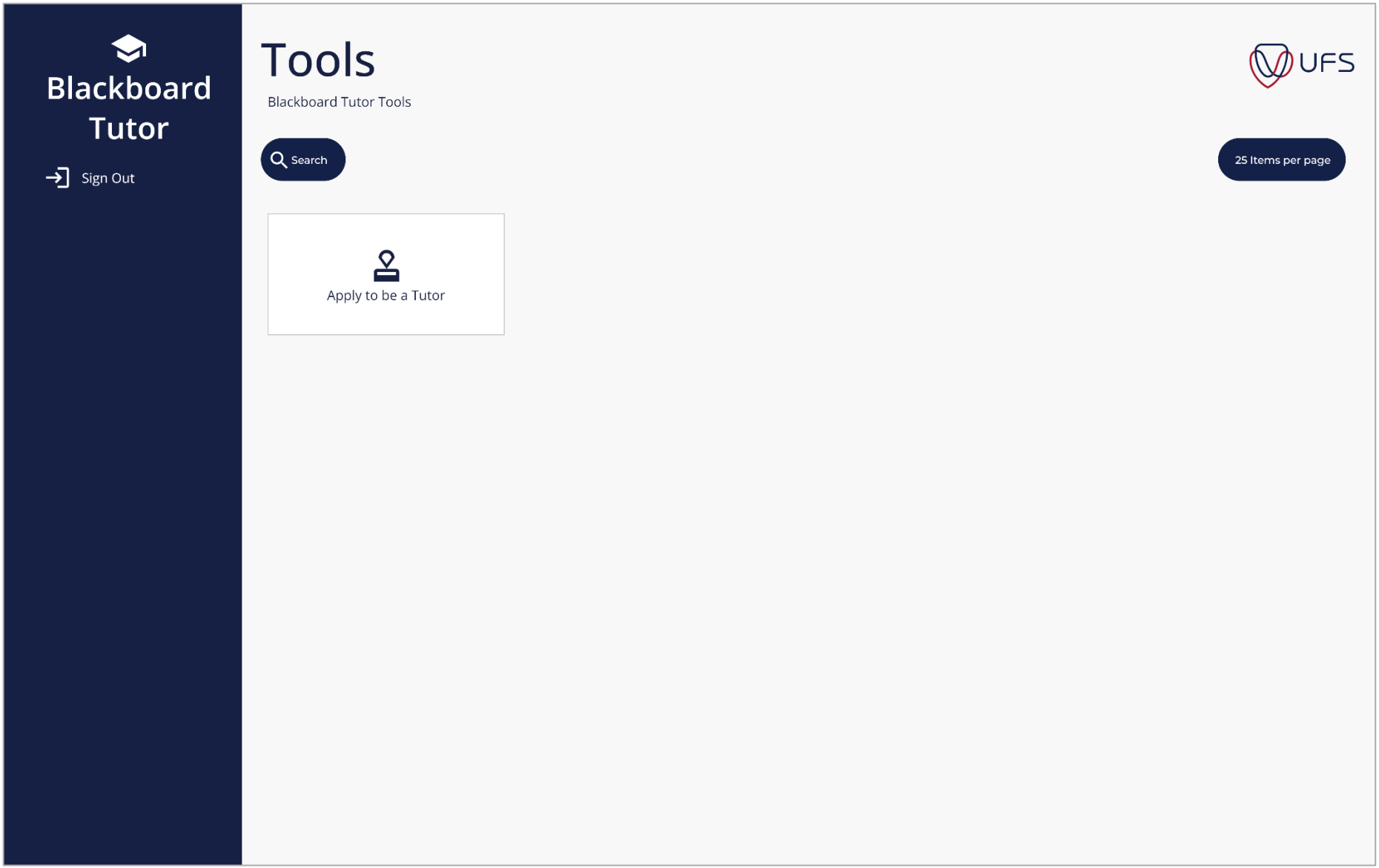


Figure 53: Eligible Tutor User Interface

For steps 2-5 kindly refer to Figure 54 Apply to be a Tutor popup below.

2. The student is presented with a dropdown box of modules that they are eligible to apply for, a description box and button to attach supporting documentation (only certain documents are acceptable e.g., pdf and docx).

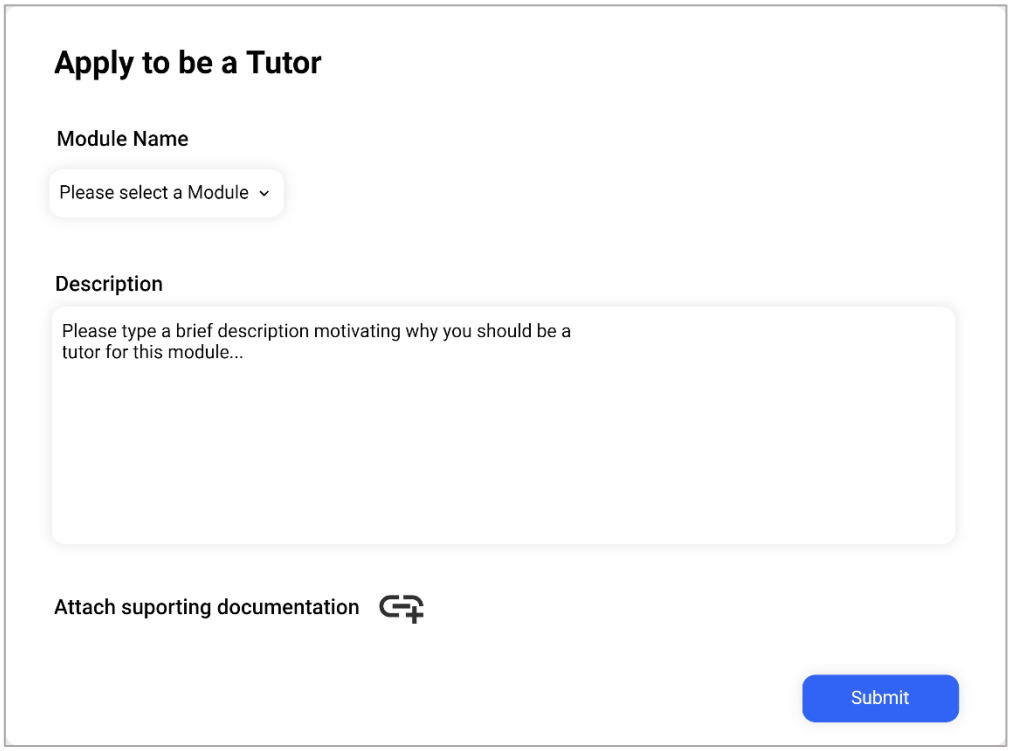


Figure 54: Apply to be a Tutor Popup

3. Student selects a module, writes a description, and attaches a file as supporting documentation.
4. Student clicks the “Submit” button.
5. The popup closes.
6. A notification pops up stating: “Thank you for applying! A lecturer will review your application shortly”.

Best-case Scenario

1. Logan clicks on the “Apply to be a tutor” button on their profile page (the only page they can access).
2. Logan is presented with a list of modules that he is eligible to apply for, a description box and button to attach supporting documentation.
3. Logan selects a module, writes a description, and attaches a file as supporting documentation.
4. Logan clicks the submit button.
5. The popup closes.
6. A notification thanking Logan for applying and stating that a lecturer will review their application shortly appears.

Worst-case Scenario

1. Logan clicks on the “Apply to be a tutor” button on his profile page.
2. Logan is presented with a list of modules that he is eligible to apply for, a description box and button to attach supporting documentation.
3. Logan does not select a module, does not enter a description, and does not attach supporting documentation.
4. Logan clicks the submit button.
5. The application does not go through. Error messages are displayed above each textbox stating: “Please select a module”, “Please enter a description” and “Please attach supporting documentation”.

Alternative Scenario A

1. Logan clicks on the “Apply to be a tutor” button on his profile page.
2. Logan has already applied to be a tutor in a specific module.
3. Logan is presented with a list of modules that he is eligible to apply for, a description box and button to attach supporting documentation.
4. Logan selects the module he has already applied for, writes a description, and attaches a file as supporting documentation.
5. Logan clicks the submit button.
6. The application does not go through. A notification pops up stating: “You have already submitted an application for this module”.

Alternative Scenario B

1. Logan clicks on the “Apply to be a tutor” button on his profile page.
2. Logan is presented with a list of modules that he is eligible to apply for, a description box and button to attach supporting documentation.
3. Logan does select a module, enters a description, and attaches supporting documentation that is not allowed.
4. Logan clicks the submit button.
5. The application does not go through. An error message is displayed stating: “Please attach a pdf or word document for your supporting documentation”.

Session

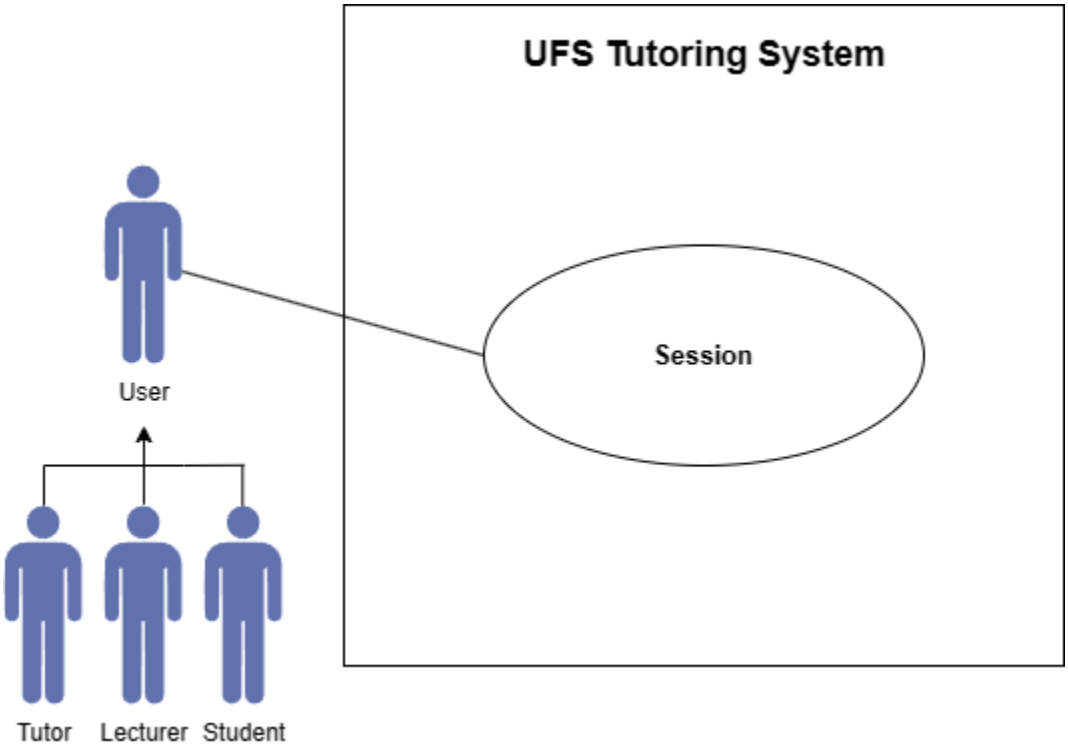


Figure 55: Session Use Case

Brief Description

The Session function allows users to view and download tutoring sessions. It also allows a tutor or lecturer to create, edit and delete sessions.

Step-by-Step Description

For step 1 kindly refer to Figure 18 View Modules User Interface.

1. User clicks the virtual classroom tab in any module view.

For step 2-3 kindly refer to Figure 56-57 Session User Interfaces.

2. System displays tutoring sessions.
3. User can interact with the sessions.

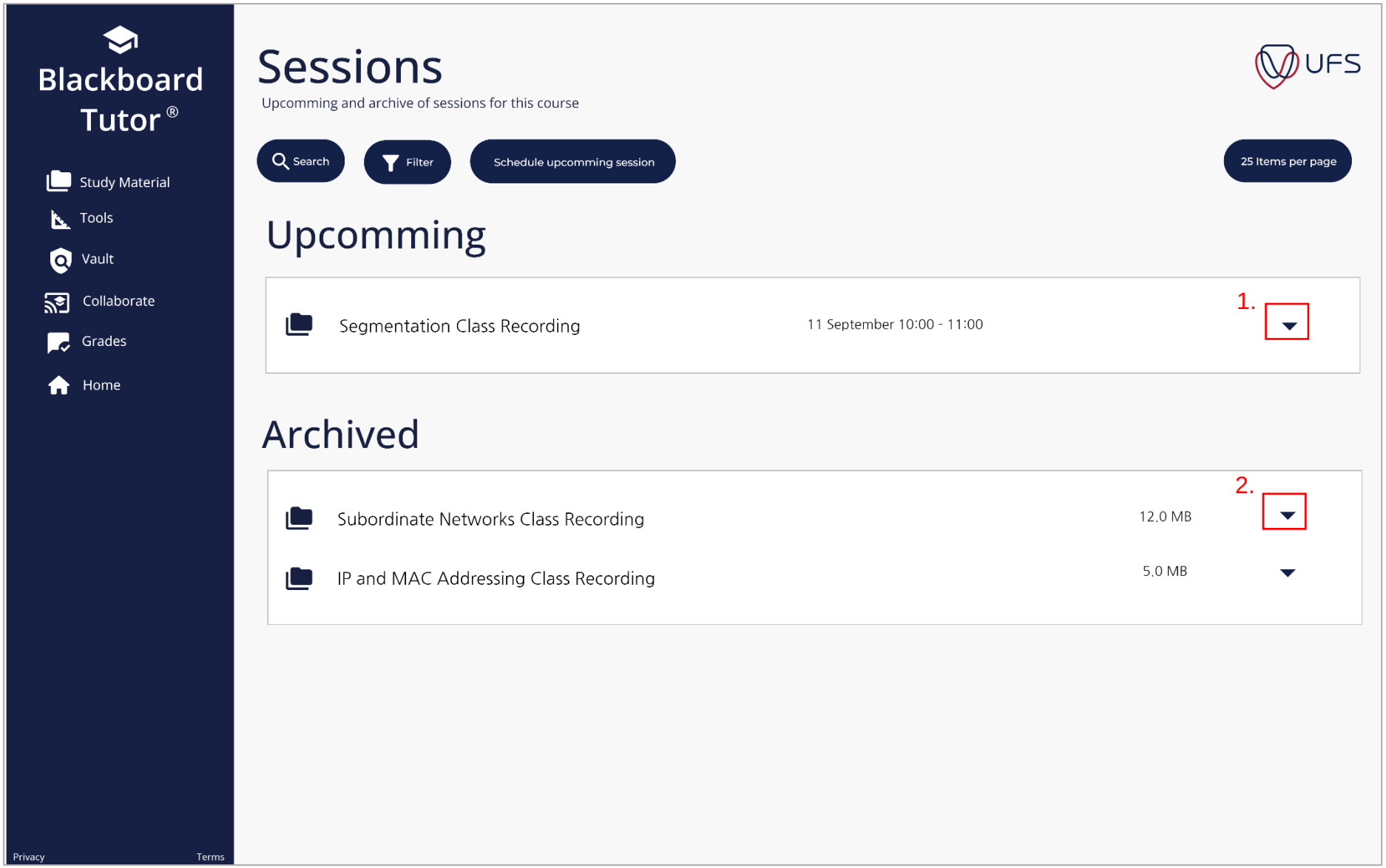


Figure 56: Tutor/Lecturer Session User Interface

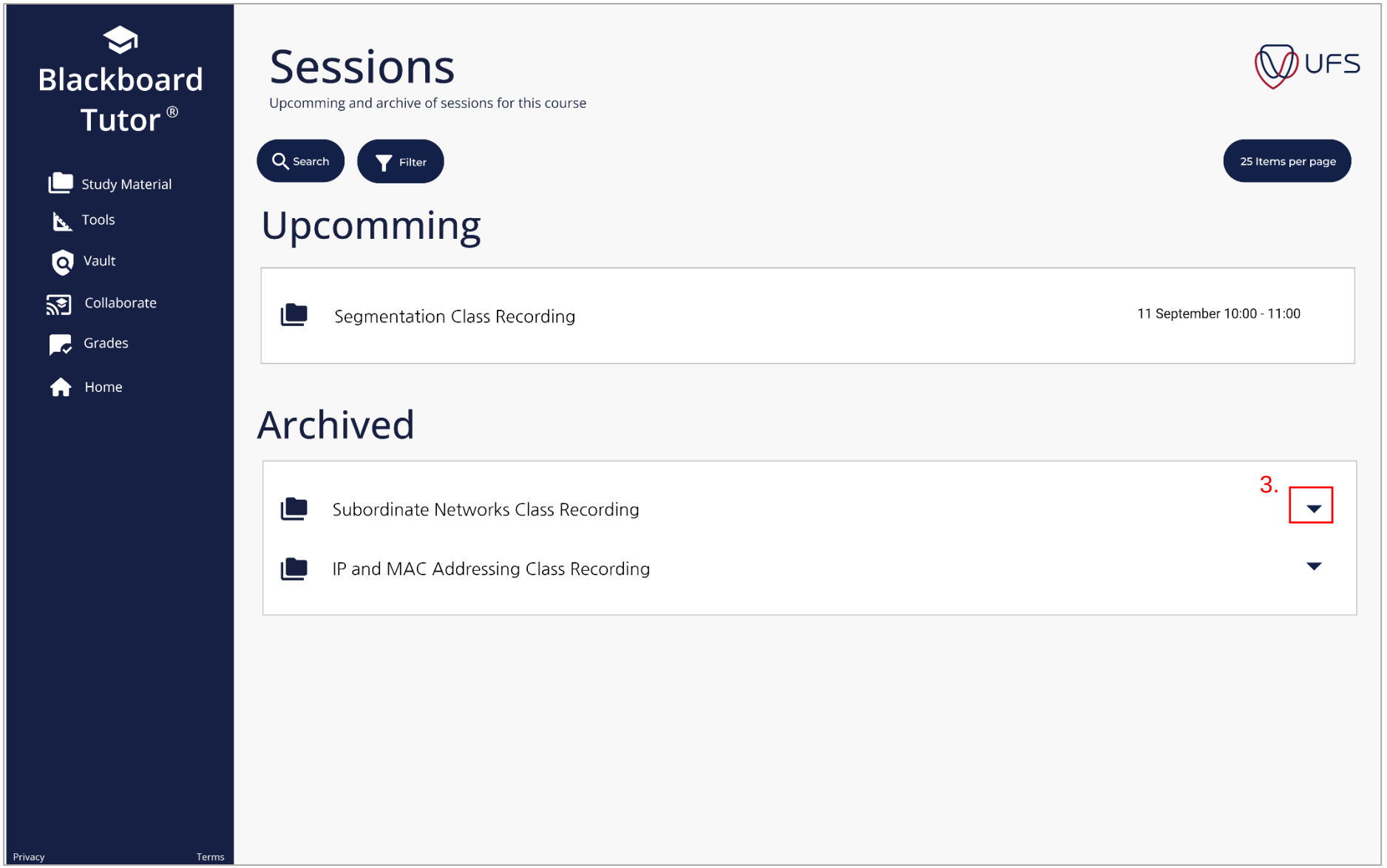


Figure 57: Student Session User Interface

If the user is a tutor/lecturer, they can:

The below forms popup when a tutor/lecturer click on the down arrow as indicated in 1 and 2 respectively.

1.

Figure 58: Tutor/Lecturer Upcoming Session Popup

2.

Figure 59: Tutor/Lecturer Archived Session Popup

The below form popup when a student clicks on the down arrow as indicated in 3.

3.

Figure 60: Download Popup

4. Add Tutoring Sessions.
   i. User clicks on the “Schedule upcoming session” button.
   ii. System displays a popup form with adjustable parameters titled “Add an Upcoming Session”. The parameters are name, start date time and end date time.

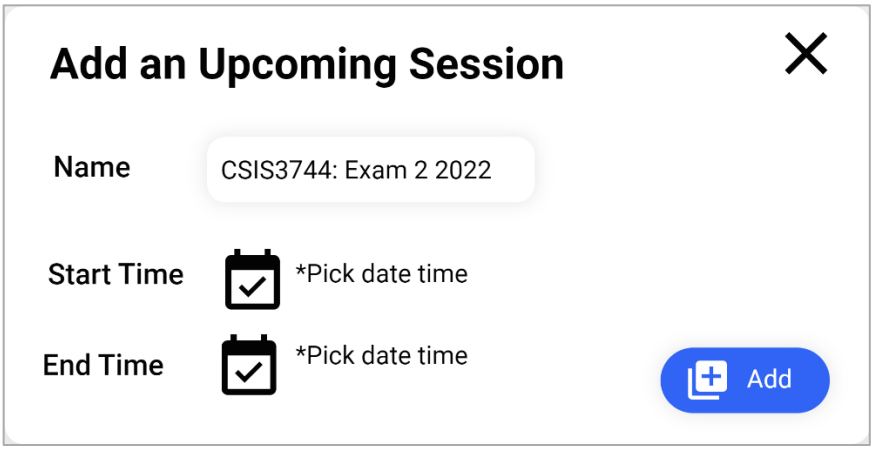


Figure 61: Add Upcoming Session Popup

5. Edit Upcoming Tutoring Session.
    i. User clicks on the “Edit” button.
    ii. System displays a populated popup form with adjustable parameters titled “Change an Upcoming Session”. The parameters are name, start date time and end date time.

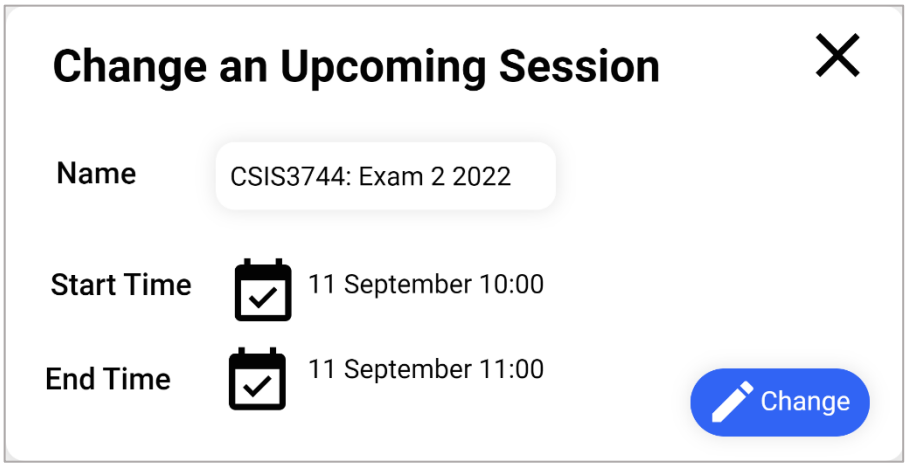


Figure 62: Edit Upcoming Session Popup

6. Edit Archived Tutoring Session.
    i. User clicks on the “Edit” button.
    ii. System displays a populated popup form with adjustable parameters titled “Change an Archived Session”. The parameters are name and video (the video must be a MP4).

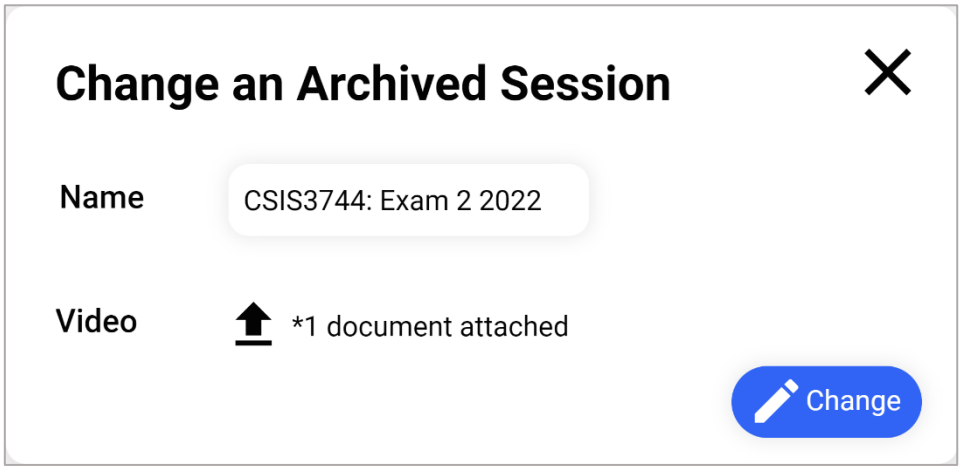


Figure 63: Edit Archived Session Popup

We acknowledge that the system should automatically be able to record and attach the recording, however for the purpose of this project the recording will be taken by the lecturer / tutor themselves and then uploaded.

7. Delete Tutoring Session.
    i. User clicks on the “Delete” button.
    ii. System creates a delete popup message stating: You are about to delete a recording are you sure you want to complete this action?”.
    iii. User clicks the “Delete” button.

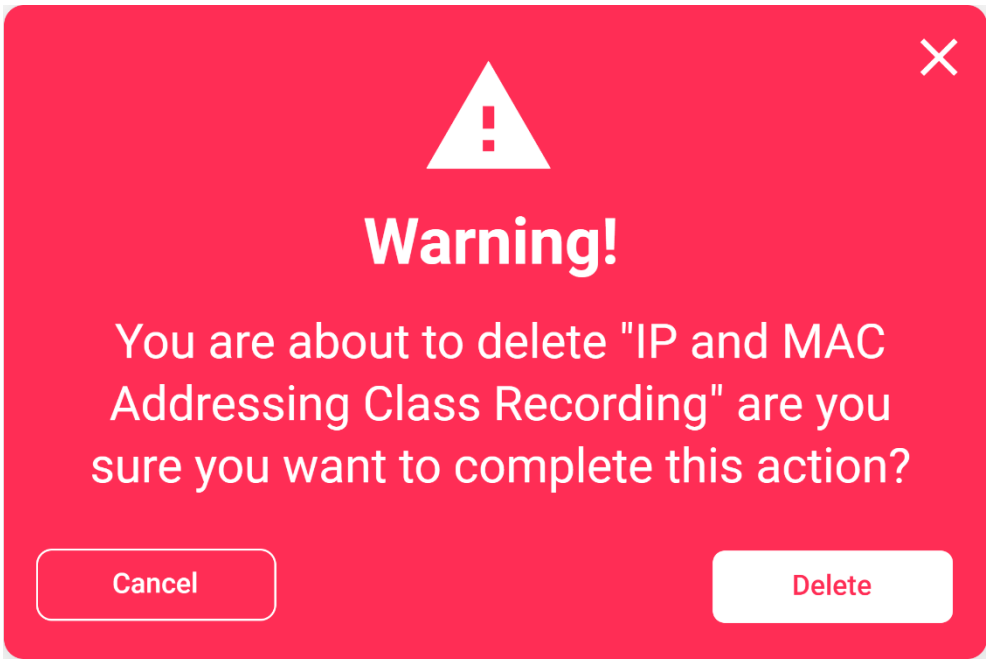


Figure 64: Delete Session Popup

<u>If the user is a student/lecturer or tutor, they can:</u>

8. Download Archived Tutoring Session.
    i. User clicks on the “Download” button.
    ii. System downloads archived tutoring session to archived session folder on the user’s device. For details, kindly see offline access/download under project proposal.

Best-case Scenario

1. Tom (a tutor) clicks on the “Virtual classroom” tab in his module view.
2. System displays tutoring sessions.
3. Tom clicks on the “Schedule upcoming session” button to create a tutoring session.
4. The System displays a popup form with adjustable session parameters for Tom to populate. The parameters of the session are name, start date time and end date time.
5. Tom populates the form and clicks on “Confirm”.
6. The system identifies all populated parameters as acceptable.
7. System closes the popup.

Worst-case Scenario

1. Tom (a tutor) clicks on the “Virtual classroom” tab in his module view.
2. System displays tutoring sessions.
3. Tom clicks on the “Schedule upcoming session” button to create a tutoring session.
4. The System displays a popup form with adjustable session parameters for Tom to populate. The parameters of the session are name, start date time and end date time.
5. Tom populates the form.
6. System checks if all populated parameters are acceptable.
7. System identifies that the datetime of the session clashes with another session for the same module.
8. System displays an error message stating: “Please select another date time, there is already a class scheduled for this time”.

Alternative Scenario A

1. Tom (a tutor) clicks on the “Virtual classroom” tab in his module view.
2. System displays tutoring sessions.
3. Tom clicks on the “Schedule upcoming session” button to create a tutoring session.
4. System displays a popup form with adjustable session parameters for Tom to populate. The parameters of the session are name, start date time and end date time.
5. Tom adds a name, start time and end time.
6. Tom accidentally clicks on the “Exit” icon.
7. System creates a confirmation popup stating: “Are you sure you want to leave? All you information will be lost”.
8. Tom clicks “Cancel”.
9. The popup closes and Tom can continue adding the session.

Alternative Scenario B

1. Tom (a tutor) clicks on the “Virtual classroom” tab in his module view.
2. System displays tutoring sessions.
3. Tom clicks on the “Schedule upcoming session” button to create a tutoring session.
4. System displays a popup form with adjustable session parameters for Tom to populate. The parameters of the session are name, start date time and end date time.
5. Tom adds a name and start date time, but not an end date time.
6. Tom clicks “Confirm”.
7. The session is not created, and an error message is displayed stating: “Please add a date!”.

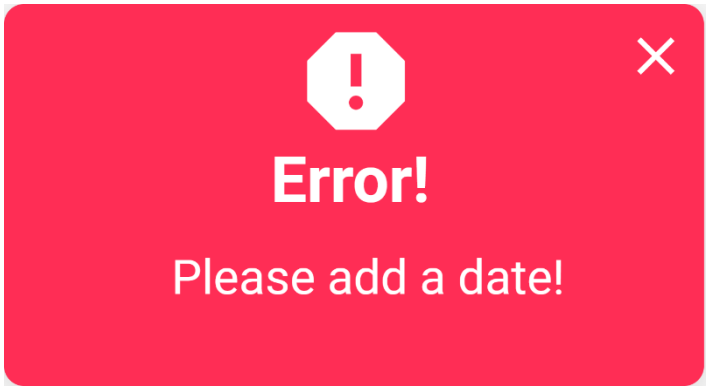


Figure 65: Add Date Error

Alternative Scenario C

1. Tom (a tutor) clicks on the “Virtual classroom” tab in his module view.
2. System displays tutoring sessions.
3. Tom clicks on the “Schedule upcoming session” button to create a tutoring session.
4. System displays a popup form with adjustable session parameters for Tom to populate. The parameters of the session are name, start date time and end date time.
5. Tom adds an end date time and start date time, but not a name.
6. Tom clicks “Confirm”.
7. The session is not created, and an error message is displayed stating: “Please add a date!”.

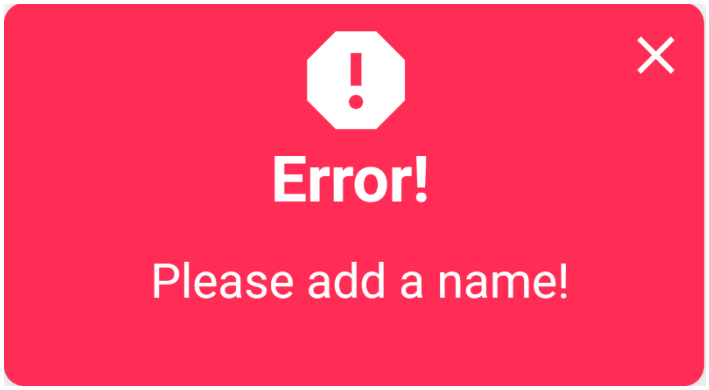


Figure 66: Add Name Error

Alternative Scenario D

1. Tom clicks on the drop-down arrow next to an archived session.
2. System displays a popup.
3. Tom clicks on “Download” session button.
4. System locates archived sessions folder on Tom’s device.
5. System cannot locate archived sessions folder.
6. System creates archived sessions folder in tutoring folder.
7. System starts downloading archived tutoring session to the archived session

folder. This will be indicated by a downloading symbol next to the session.
8. Tom sees that there is a downloading symbol next to the session.
9. System finished downloading archived tutoring session to archived session folder on Tom's device. This will be indicated by a finished downloading symbol next to the session.
10. Tom sees that there is a finished downloading symbol next to the session.

Alternative Scenario E

1. Tom (a tutor) clicks on the "Virtual classroom" tab in his module view.
2. System displays tutoring sessions.
3. Tom clicks on the "Schedule upcoming session" button to create a tutoring session.
4. System displays a popup form with adjustable session parameters for Tom to populate. The parameters of the session are name, start date time and end date time.
5. Tom adds a name. He tries to add an end date time and start date time for yesterday, but all previous date times are greyed out.

Alternative Scenario F

1. Tom clicks on the drop-down arrow next to an archived session.
2. System displays a popup.
3. Tom clicks on the "Edit" button.
4. System displays a populated form with adjustable session parameters for the user to change. The parameters are the name and video (the video must be a MP4).
5. Tom attaches a video that is not an MP4.
6. Tom clicks "Confirm".
7. The session is not changed, and an error message is displayed stating: "Incorrect video format, please attach a MP4!".

Virtual Classroom

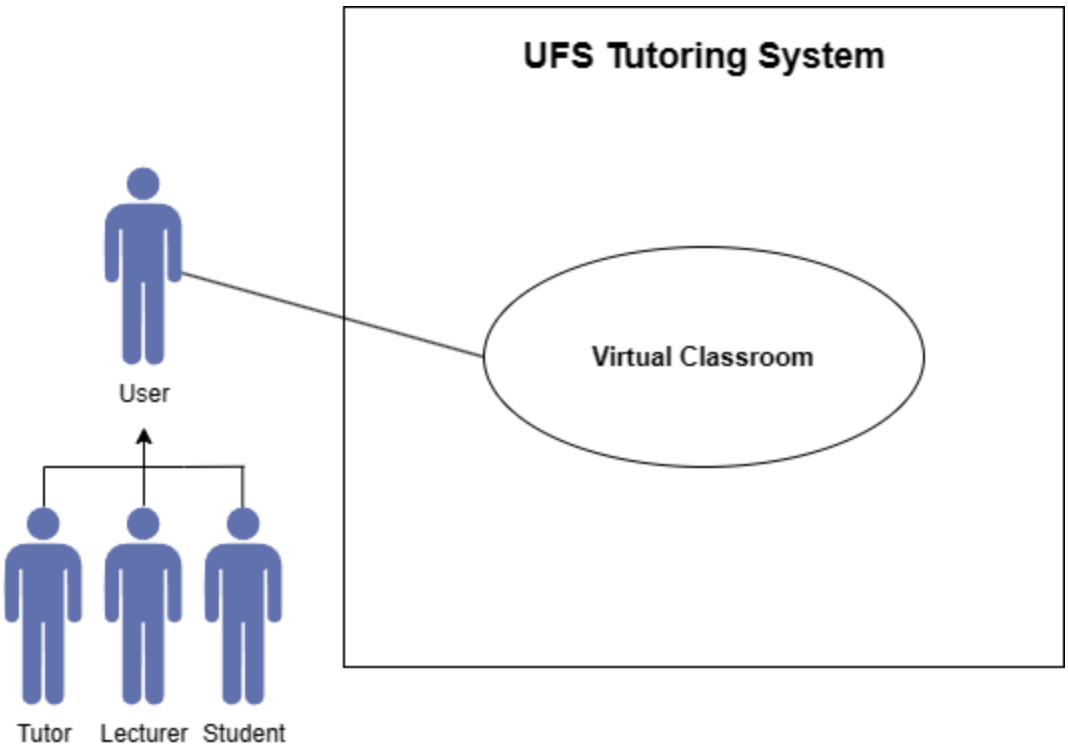


Figure 67: Virtual Classroom Use Case Diagram

Brief Description

The Virtual Classroom function allows a user to experience and interact in a classroom environment online.

Step-by-Step Description

System allows users to join session 10 minutes before the start of the session – only then will the “Join session” button for that session become visible.

1. User clicks on “Join session” button.

For step 2-3 kindly refer to Figure 68-70 Virtual Classrooms.

2. System displays the virtual classroom to the user.
3. User enters the virtual classroom with their sound and camera off.

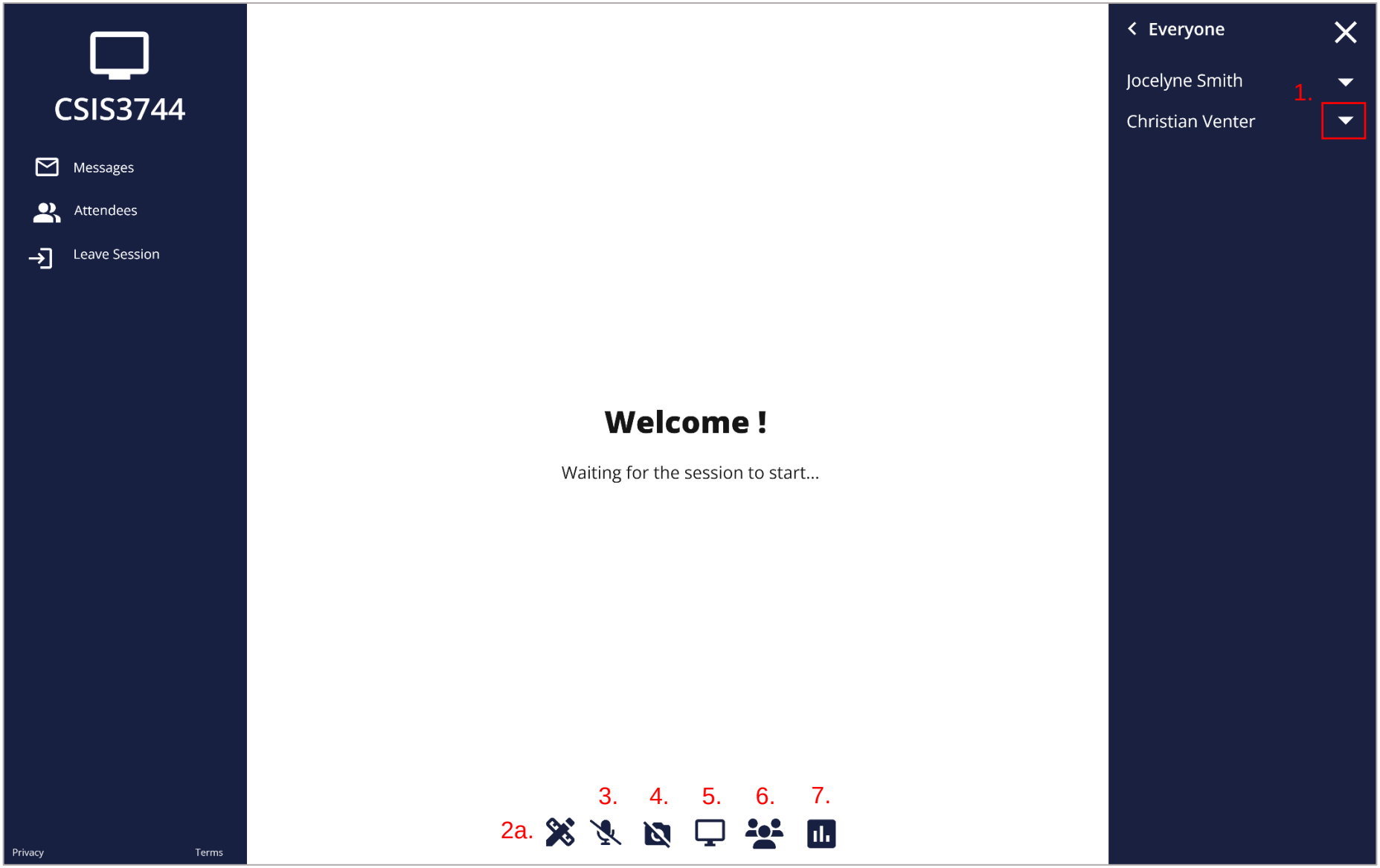


Figure 68: Lecturer/Tutor Virtual Classroom User Interface

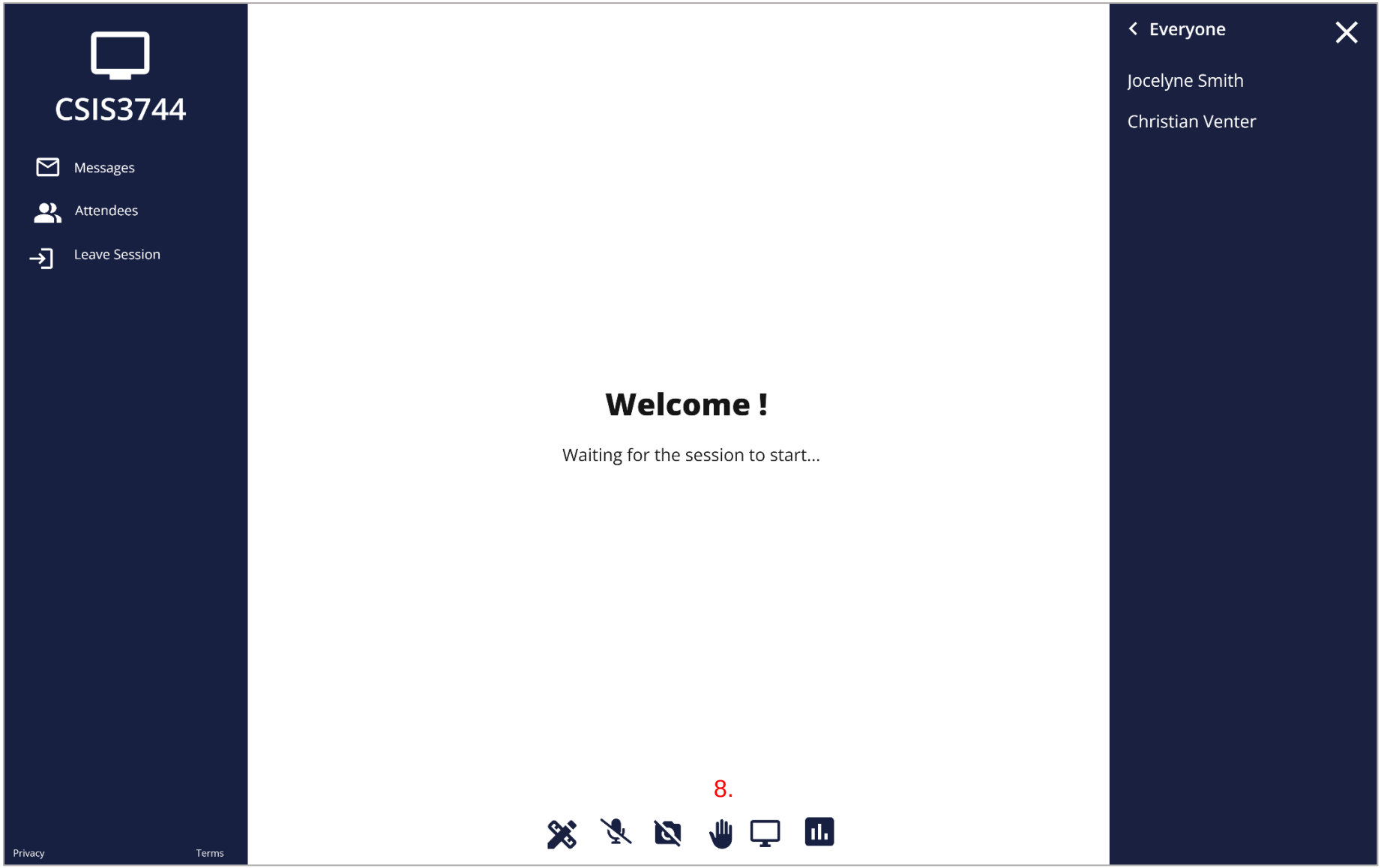


Figure 69: Student Virtual Classroom User Interface

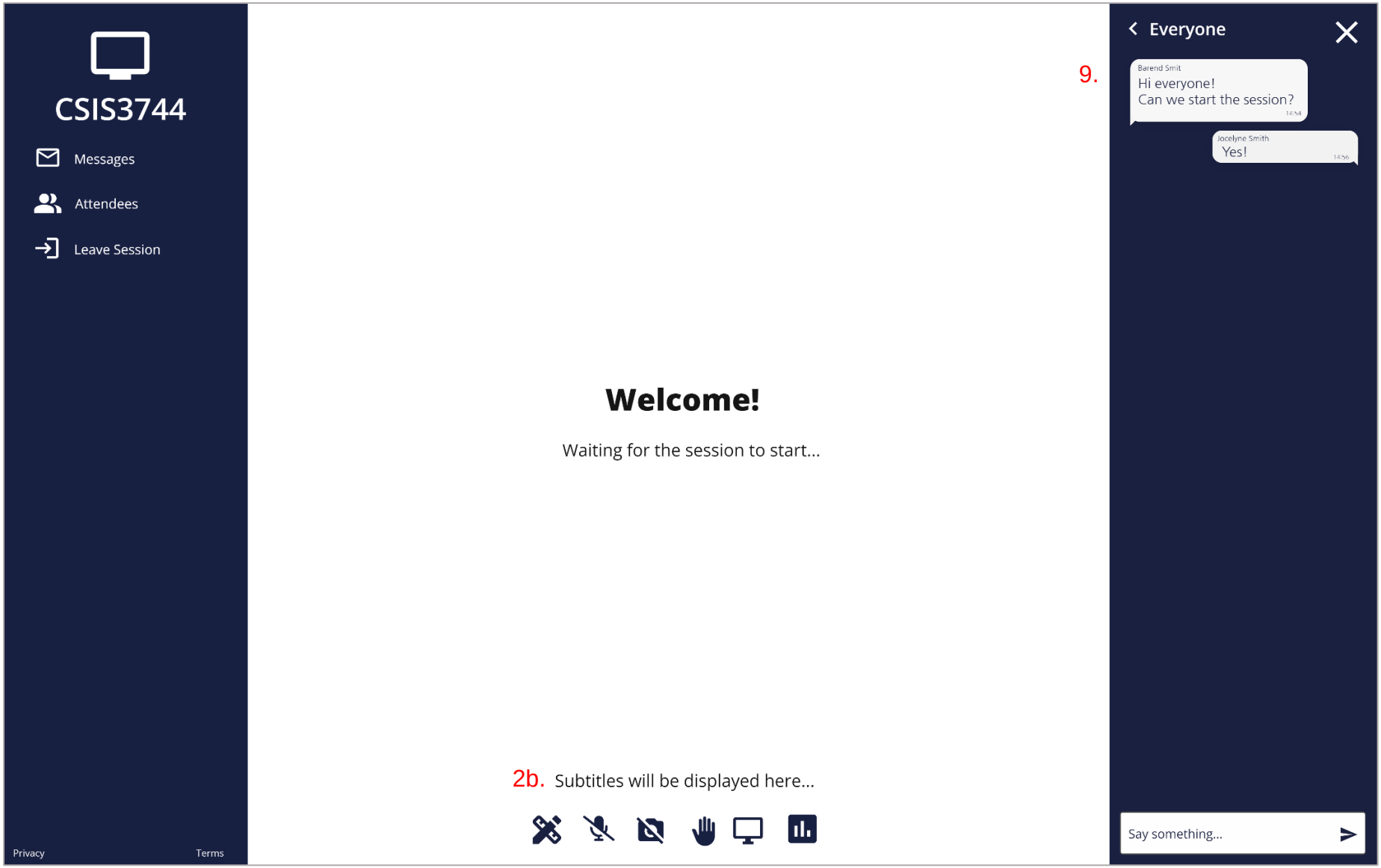


Figure 70: Chat feature in the Virtual Classroom

The below form popup when a tutor/lecturer clicks on the down arrow as indicated in 1.

1.

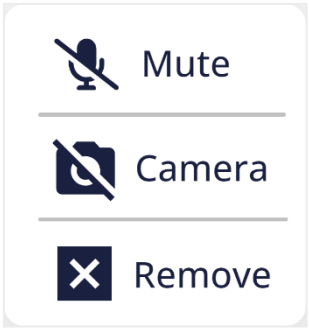


Figure 71: Tutor/Lecturer Virtual Classroom popup

This form allows a tutor/lecturer to mute, disable the camera of, or remove a student.

2a.

4. Speech to Text
   i. User clicks on the “Speech to Text” (STT) icon located at the bottom of their screen. The speech to text (STT) is currently off as indicated by a line through the button.
   ii. System turns the user’s speech to text (STT) on.
   iii. System indicates the sound being on by removing the line through the button.

Please view 2b to see how speech to text appears on the user’s screen.

3.

5. Sound Control
   i. User clicks on the “Sound” icon located at the bottom of their screen. The sound is currently off as indicated by a line through the button.
   ii. System turns the user’s sound on.
   iii. System indicates the sound being on by removing the line through the button.

4.

6. Camera Control
   i. User clicks on the “Camera” icon located at the bottom of their screen. The camera is currently off as indicated by a line through the button.
   ii. System turns the user's camera on.
   iii. System indicates the camera being on by removing the line through the button.

5.

7. Screen Sharing
   i. User clicks the “Share Screen” icon and chooses a screen to share.
   ii. System gains approval from the session leader and shares the screen with all participants in the session.

9.

8. Live Chat
    i. User types out a message in the live chat textbox and clicks the “Send” icon.
    ii. System displays message in list of live chat messages.

<u>If the user is a tutor/lecturer, they can:</u>

6. 9. Breakout Groups
    i. User clicks on the “Create Breakout Groups” icon and specifies how many participants there are in each group.
    ii. System divides the participants into various smaller virtual classrooms.

7. 10. Create Poll
    i. User clicks on the “Create Poll” icon.
    ii. System displays a popup form with adjustable poll parameters for the user to populate. The parameters of the popup are name, description, timer, questions, answers and allow multiple answers.

<u>If the user is a student, they can:</u>

8. 11. Hand Raising
    i. User clicks on the hand raising button.
    ii. System highlights the Hand Raising icon and raises the user’s name to the top of the participant list with a hand raising icon next to their name.

Best-case Scenario

1. Steve (a student) clicks on screen share screen icon.
2. System asks Steve’s device for available screens it can display and displays a popup with the selection.
3. Steve selects a screen with their question then clicks proceed.
4. System closes screen share popup.
5. System creates a request popup on Tom’s screen requesting to approve the screen.
6. Tom approves the request.
7. System closes screen approval popup.
8. System shares the screen with his classroom.
9. Steve clicks on the share screen icon again.
10. System stops displaying the shared screen.

Worst-case Scenario

1. System allows Steve (a student) to join the 15:10 session 10 minutes before the start of the session. At 15:00 the join session button for that session becomes visible.
2. Steve clicks on join session button at 15:05.
3. System displays the virtual classroom to Steve.
4. Tom (the tutor) is feeling ill, he cancels class and deletes the session while Steve is still in the classroom.
5. The system first kicks Steve from the session before deleting the session.

Alternative Scenario A

1. Steve clicks on the hand raising icon.
2. System unhighlights Steve's hand raising icon.
3. System lowers Steve's name to their alphabetical place in the participant list without a hand raising icon next to their name.

Alternative Scenario B

1. Steve clicks on the sound icon located at the bottom of his screen. The sound is currently on as indicated through the icon.
2. System turns Steve's sound off.
3. System indicates the sound being on by creating a line through the icon.

Alternative Scenario C

1. Steve clicks on the camera icon located at the bottom of their screen. The camera is currently on as indicated through the icon.
2. System turns Steve's camera off.
3. System indicates the camera being off by creating a line through the icon.

Alternative Scenario D

1. Steve clicks on the live chat textbox.
2. Steve types a message with inappropriate language like swear words.
3. Steve clicks on the send button next to the text box.
4. System flags message and displays a notification to Steve asking him to use more appropriate language.

Alternative Scenario E

1. Steve rapidly clicks on the hand raising button causing he name to jump rapidly on the participant list distracting other participants.
2. The system should not allow participants to rapidly select and deselect the hand raising icon.

Alternative Scenario F

1. Steve clicks on the live chat textbox.
2. Steve types a message containing sensitive information like his password.
3. Steve clicks on the send button next to the text box.
4. System identifies that Steve is sending his password and masks the password with asterixis.

Alternative Scenario G

1. Steve clicks on screen share button.
2. System asks Steve's device for available screens it can display.
3. Steve's computer system has no available screens to share.
4. System prompts Steve with the following message, "You have no screens which you can display".
5. System closes screen share popup.

Alternative Scenario H

1. Steve joins virtual classroom while breakout is in progress.
2. System searches for virtual classroom with least number of participants.
3. System transfers Steve to the virtual classroom with the least number of participants.

Rate Session

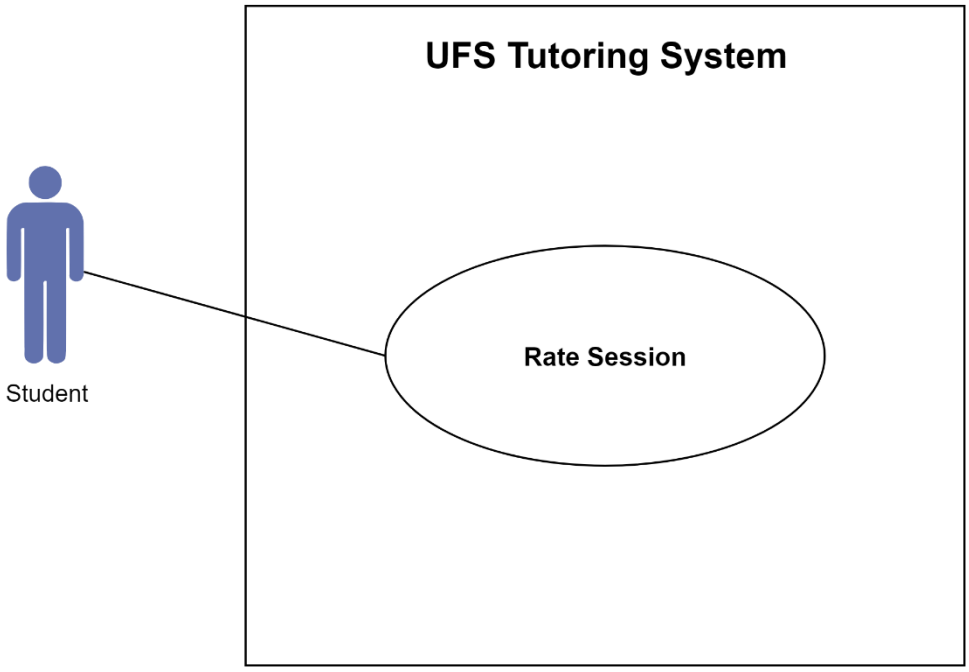


Figure 72: Rate Session Use Case

Brief Description

The Rate Session function allows a student to rate their tutor after every live session.

Step-By-Step Description

For step 1 kindly refer to Figure 73 Rate Tutor Popup below.

1. If the session was presented by a tutor, a student is presented with a popup to rate the tutoring session out of 5 stars after the session ends.

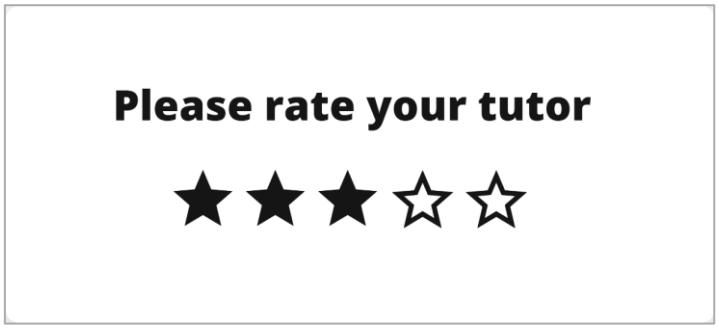


Figure 73: Rate Tutor Popup

2. Student selects how many stars he wants to give the tutor, including half-stars.
3. System saves the rating.
4. System closes the popup.
5. System displays a notification to thank the user for rating the session.

Best-case Scenario

1. Diana is presented with the rating popup.
2. Diana selects the amount of the stars she wants to award the session based on the tutor's helpfulness.
3. System closes the popup.
4. System displays a notification thanking Diana.

Worst-case Scenario

1. Diana is presented with the rating popup.
2. Sarah rates the tutor.
3. Sarah rejoins the tutoring session.
4. System does not display the rating popup again since she has already rated the session.

Alternative Scenario

1. Sarah is presented with the rating popup.
2. Sarah closes the tab.
3. System displays a confirmation popup appears stating: "Are you sure you want exit without rating your tutor? This will be your only chance to rate your tutor for this session".
4. Sarah clicks the "Exit anyway" button.
5. System is not affected by the fact that she did not rate her tutor.

Diary

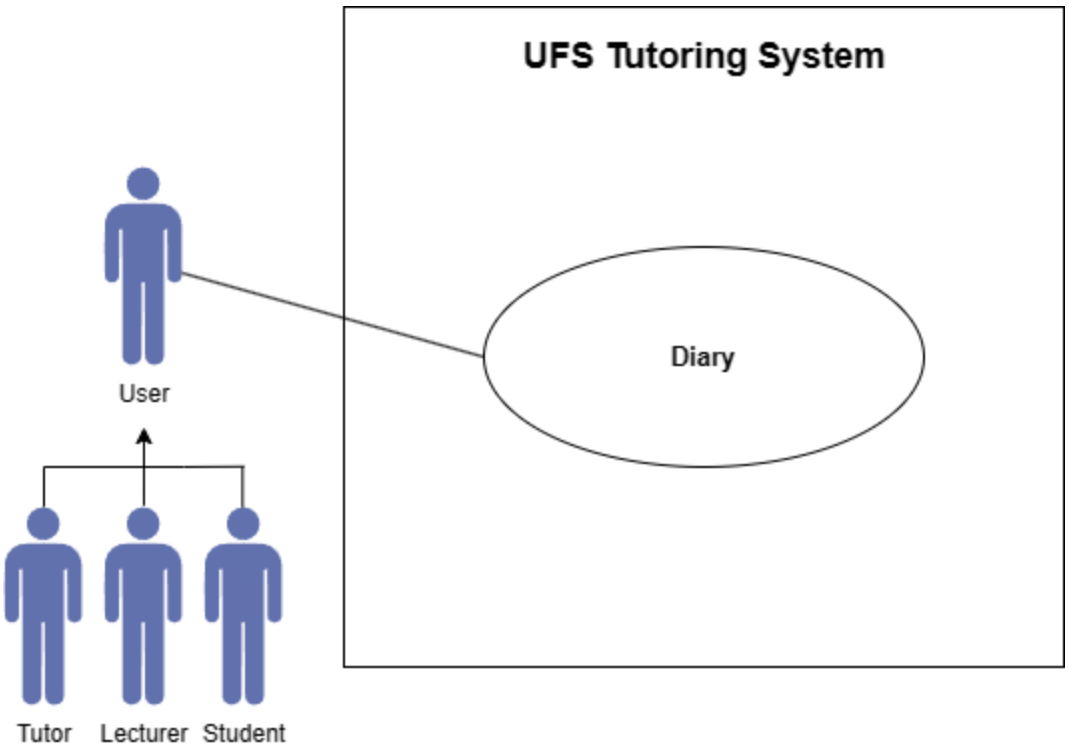


Figure 74: Diary Case Diagram

Brief Description

The Diary function allows a user to view, edit, delete, and create diary entries.

Step-by-Step Description

For step 1 kindly refer to Figure 1-3 Dashboards.

1. User clicks the diary tab.

For steps 2-6 kindly refer to Figure 78-81 Diaries below.

2. System displays the first two diary entries to user.

For step 3-6 kindly refer to the pager as indicated by 1 on Figure 77.

3. User clicks the right-side pager button.
4. System replaces the current two diary entries with the next two diary entries.
5. User clicks the left side pager button.
6. System replaces the current two diary entries with the previous two diary entries.

For step 7 kindly refer to the button as indicated by 3 on Figure 77.

7. User clicks and holds the voice note button.
8. System records the microphones input.
9. The user releases the button.

For step 10 kindly refer to the icon as indicated by 7 on Figure 80.

10. An icon appears where the user's cursor was in the diary, indicating a voice recording has been added.
11. The user clicks on the icon.
12. System displays the following popup:

Figure 75: Recording Popup

13. The user can interact with the popup. It shows when audio is being played, how long it has been playing, the remaining length of the recording. It also contains a pause button and play button.
14. When the user clicks outside the popup the popup is closed

15. Edit Diary Entries.

i. User clicks anywhere on an entry in the diary and starts typing.
ii. System allows user to edit the text.

For step iii kindly refer to the button as indicated by 6 on Figure 79.

iii. A user can toggle a checkbox next to a to do list item.

We acknowledge that we should include a redo / undo button however for the purposes of this assignment we will not include this feature.

16. Create Diary Entry.

For step i kindly refer to the button as indicated by 2 and 4 respectively on Figure 77.

i. User clicks on the to the "Create Diary Entry" icon or the "Create to-do List" icon.

For step ii kindly refer to the Figure 68 and 69 respectively.

ii. System creates a blank diary entry on the page where the user's cursor was.
iii. User edits the blank diary entry.

iv. Delete Diary Entry.

For step i kindly refer to the button as indicated by 6 respectively on Figure 79.

i. User clicks on the “Delete” button.
ii. System creates a delete popup message stating: “You are about to delete a diary entry. Are you sure you want to complete this action?”.
iii. User clicks the “Delete” button.

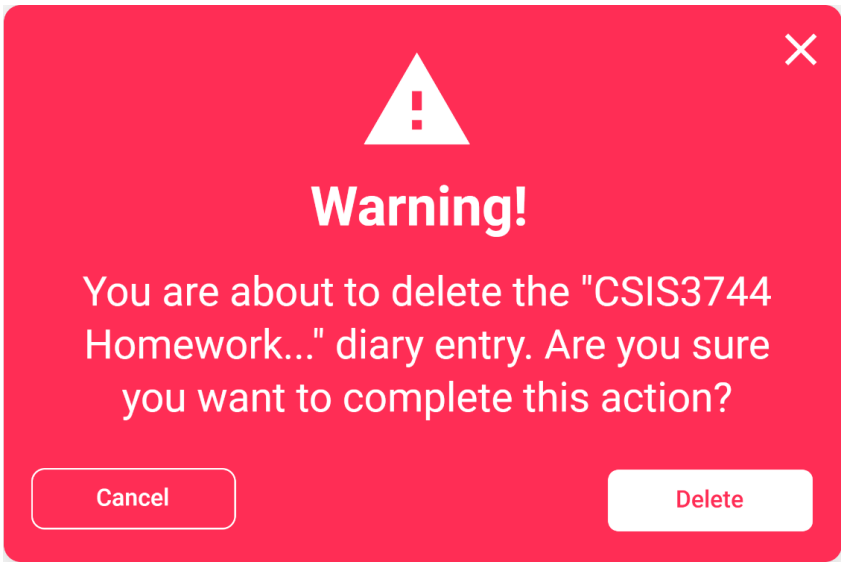


Figure 76: Delete diary entry popup

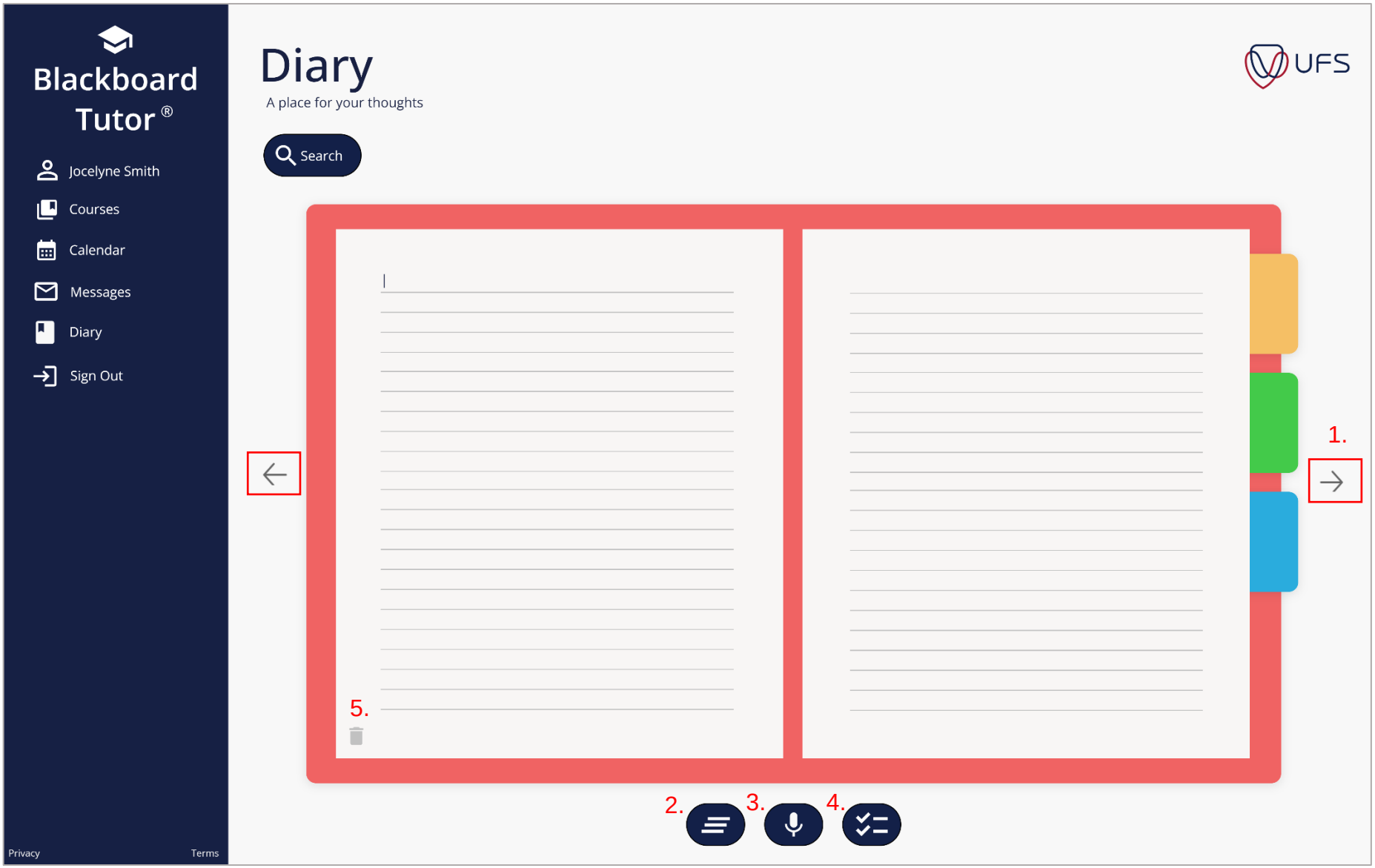


Figure 77: Blank Diary User Interface

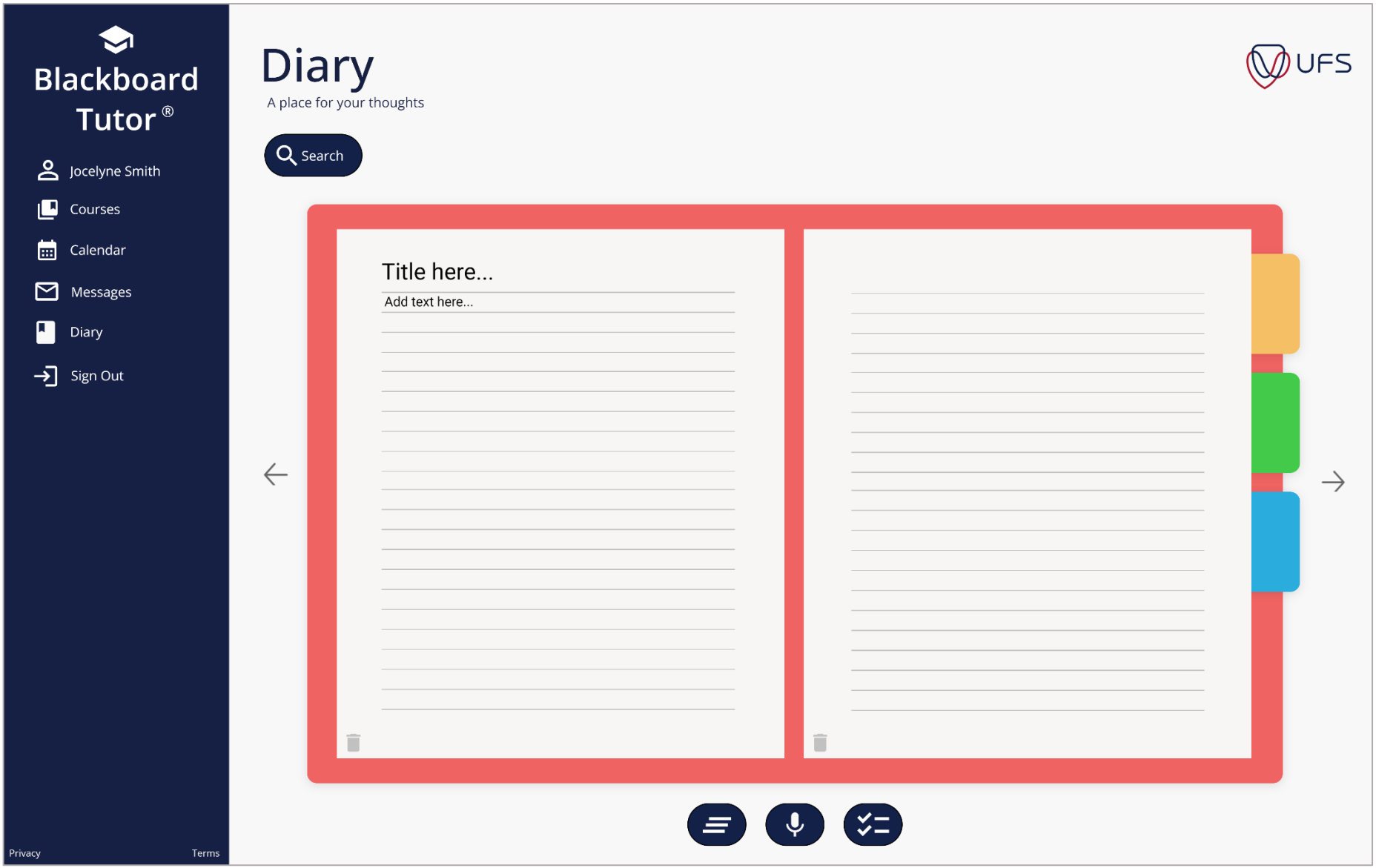


Figure 78: Diary with Text Entry

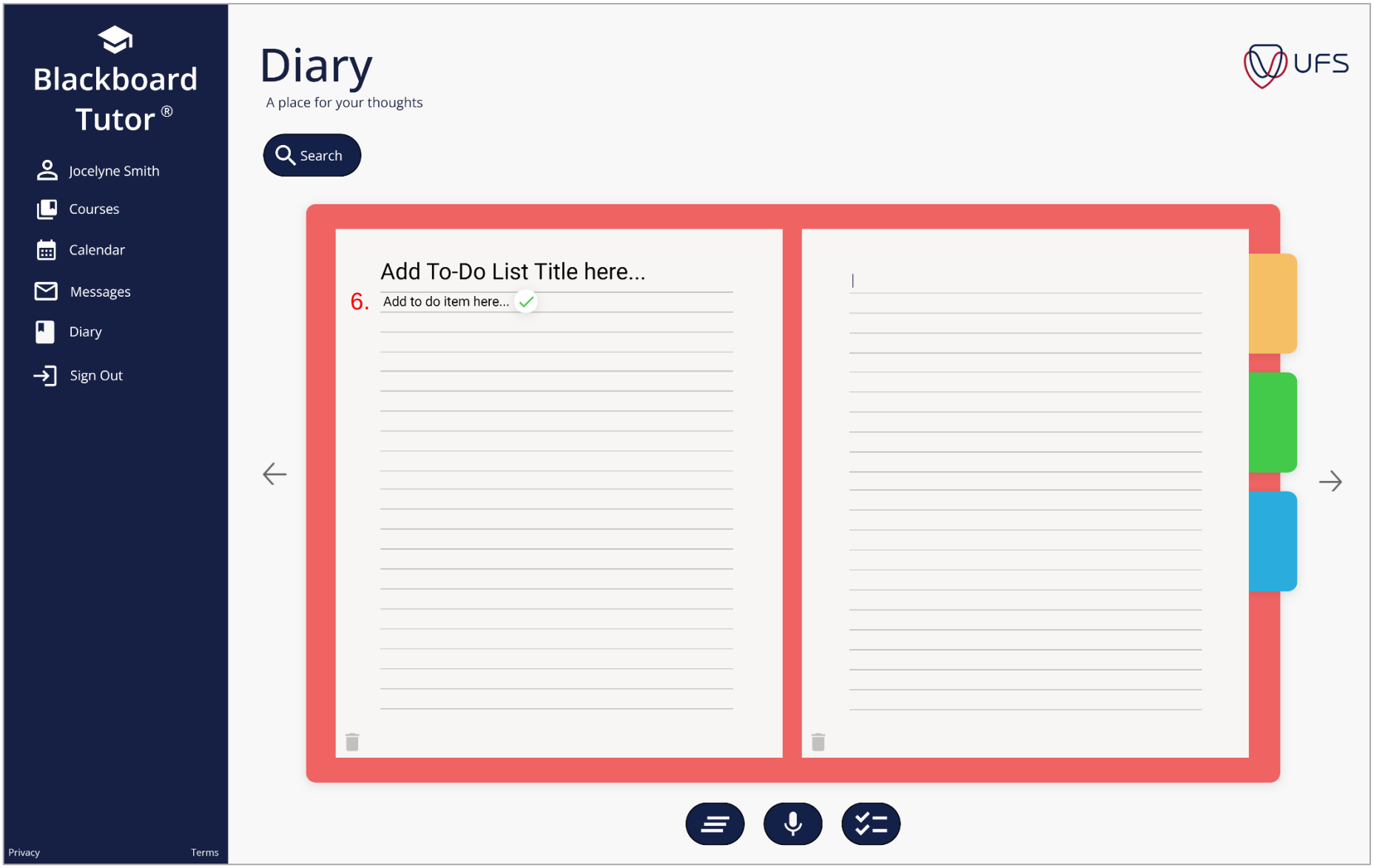


Figure 79: Diary with to-do list

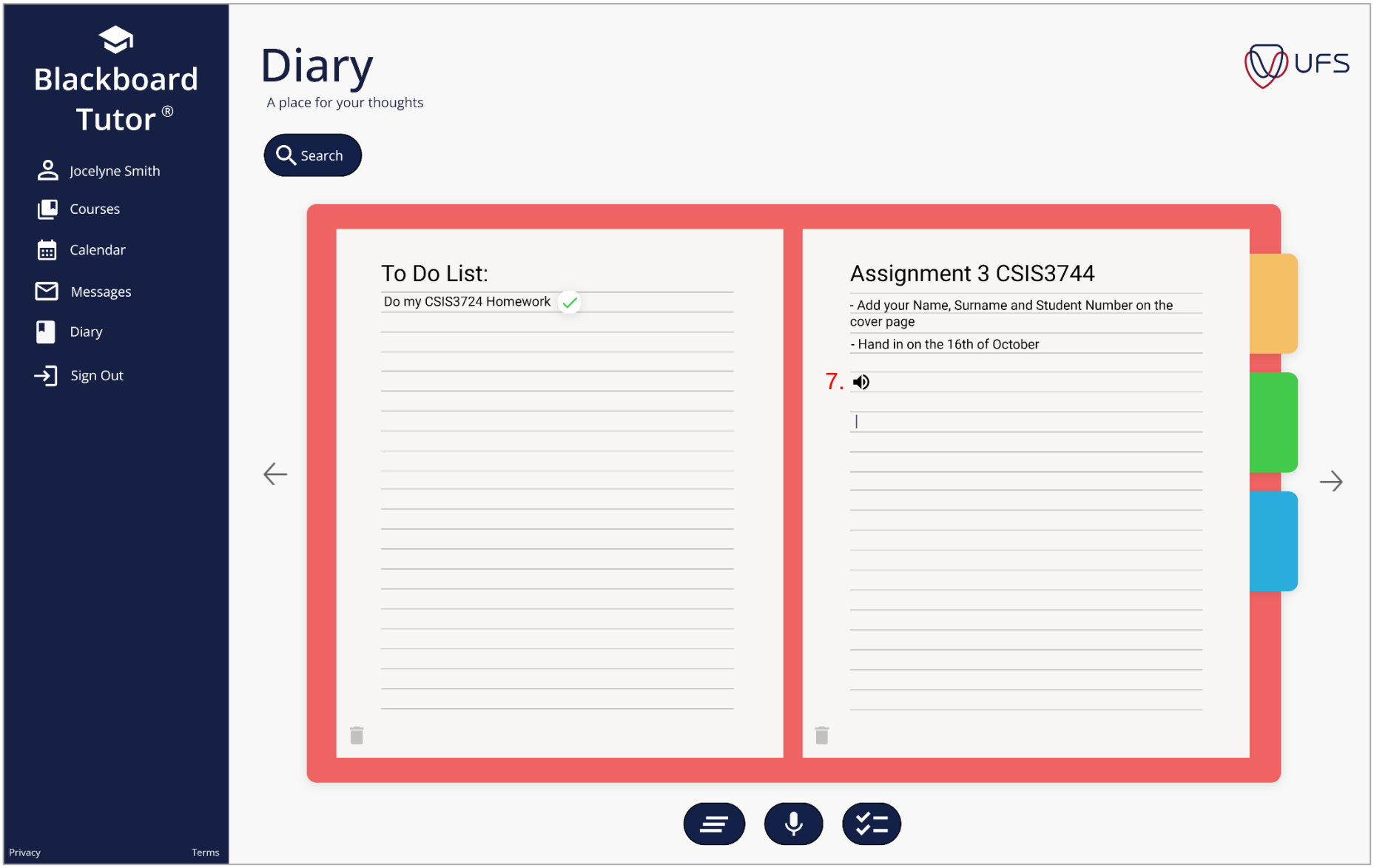


Figure 80: Completed Diary Entry

Best-case Scenario

1. Steve clicks the diary tab.
2. System displays the first two diary entries to Steve.
3. Steve clicks the right-side pager button.
4. System replaces the current two diary entries with the next two diary entries.
5. Steve clicks the left side pager button.
6. System replaces the current two diary entries with the previous two diary entries.
7. Steve clicks anywhere on a diary entry in the diary.
8. System recognises the in text placing and allows Steve’s keyboard to change information.
9. Steve enters a message through the keyboard.
10. Steve then clicks and holds the voice note icon.
11. System connects to Steve’s microphone.
12. System starts recording sound.
13. Steve releases the recording icon.
14. System stops recording.
15. System inserts the recording into the diary entry where Steve 's cursor was located. System inserts the recording into the diary entry where Steve 's cursor was located. The recording will be indicated by a speaker symbol within the text. The voice recording will be treated as a text symbol and can be deleted using the keyboard.

Worst-case Scenario

1. Steve clicks the diary tab.
2. Steve deletes his only diary entry.
3. System deletes the diary entry.
4. System still displays a Blank Diary User Interface as indicated in Figure 78.

Alternative Scenario A

1. Steve clicks on the to-the create to do list diary entry icon.
2. System creates a blank to do list diary entry on the page with intuitive editing indicators.
3. Steve edits the blank to do list diary entry.

Alternative Scenario B

1. Steve clicks the diary tab, and the system displays the first two diary entries.
2. The system does not display a right-side or left-side pager button as there are no other entries.

Direct Message

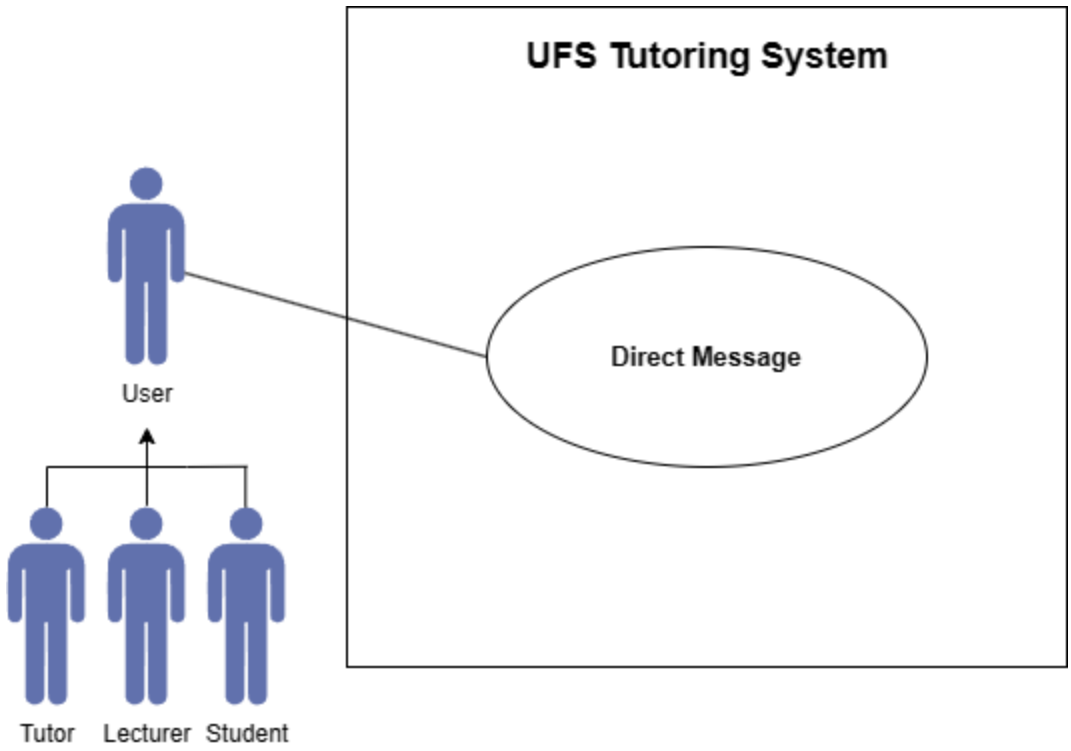


Figure 81: Direct Message Use Case Diagram

Brief Description

The Direct Message function allows users to view, send and delete messages. Additionally, lecturers can send messages to students that submitted reports (the report can appear as anonymous or not anonymous).

Step-by-Step Description

1. User clicks the “Messages” button on the side-menu.
2. System displays a list of other users whom they have messaged.
3. User clicks on a specific chat.
4. System displays all messages between the user and the other user with a text box at the bottom of the screen.

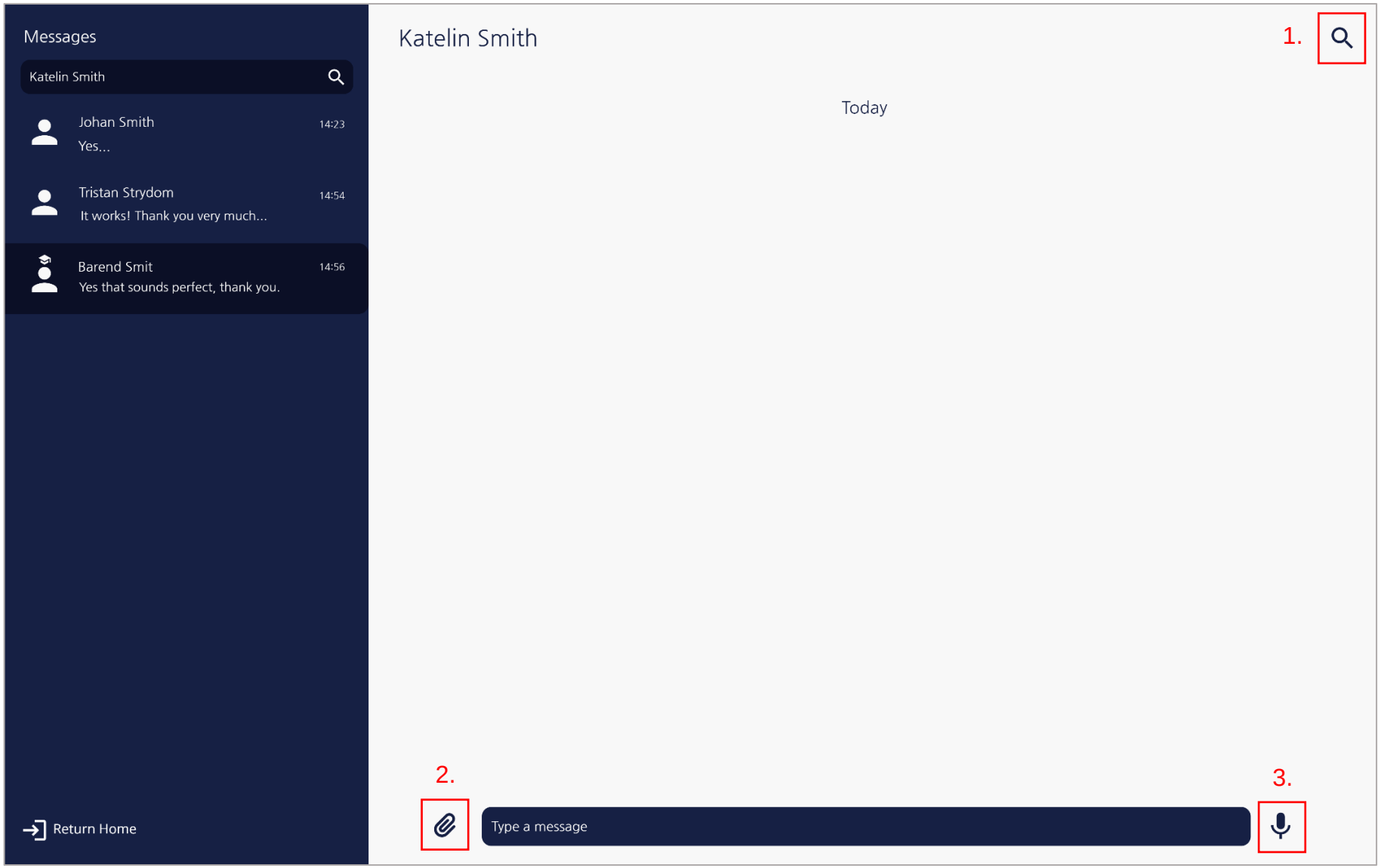


Figure 82: Message New User

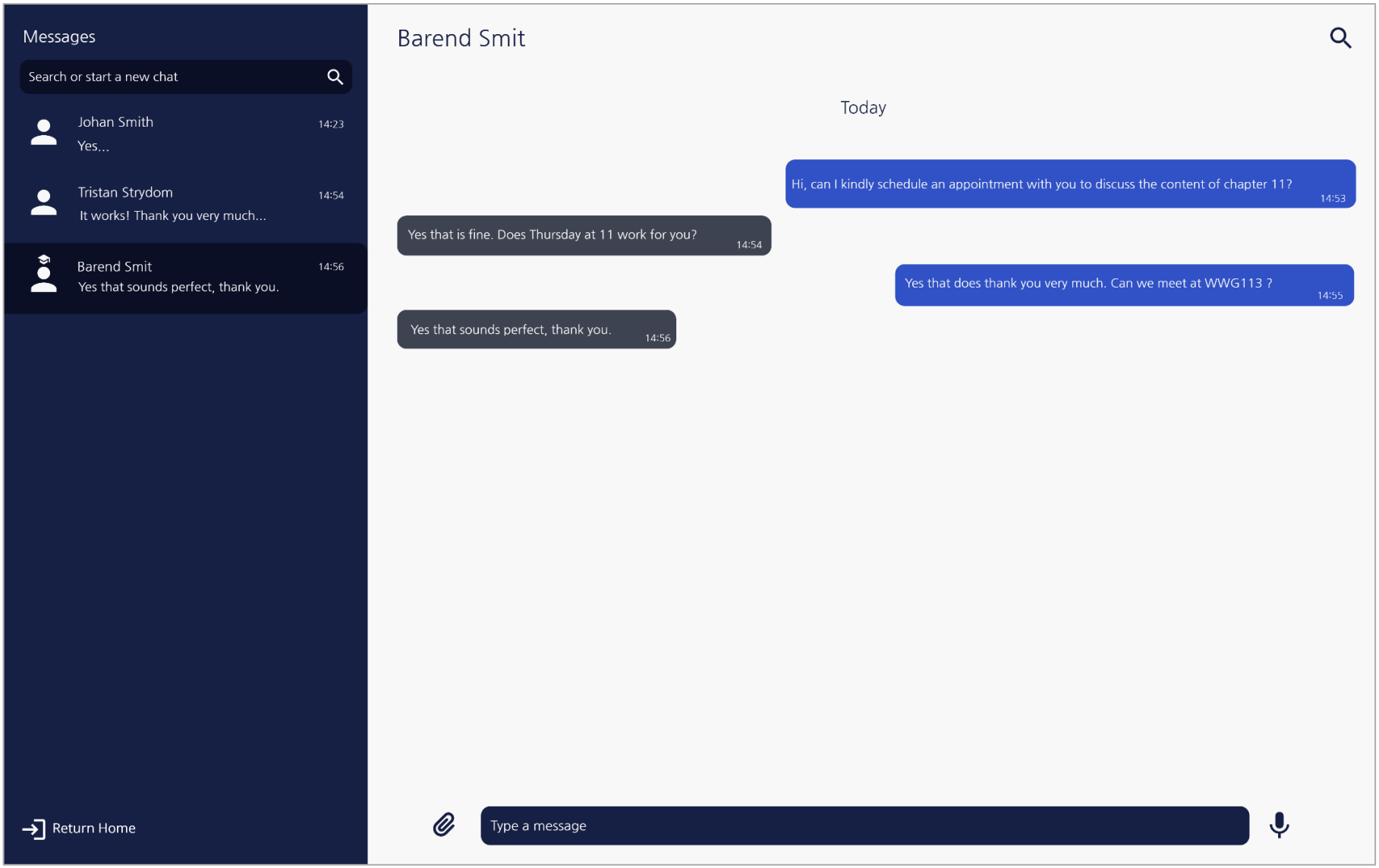


Figure 83: Messaging User

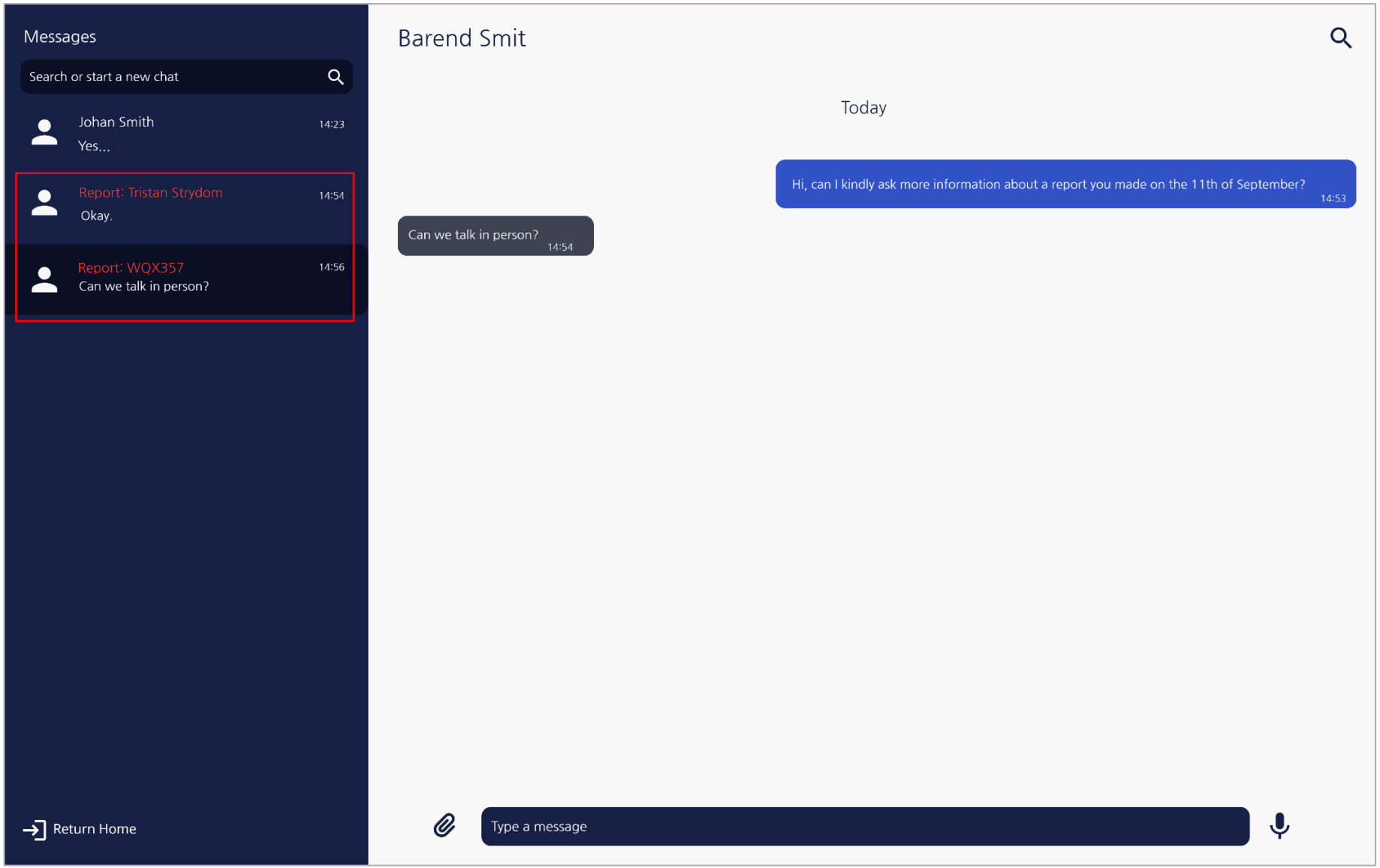


Figure 84: Lecturer communicating with Reporters

5. Send message:

   i. User clicks on the chat of the user they wish to send a message to.
   ii. User enters the message into the textbox in the chat and clicks send.
   iii. System sends message to the other user and displays the message in the user's chat.

We acknowledge that users should be able to send an attachment with a message or a voice recording, as indicated by 2 and 3 respectively, but for the purposes of this project we will not include this feature.

The other user will know they have a new message by a new message icon next to the user on their contact archive and the user will have their contact archive raised to the top of the list of contact archives.

6. Create Chat:

For step i-iv kindly refer to Figure 83 Messaging User.

   i. User enters a name in the search bar.
   ii. System displays a list of the closest matches to the entered information.
   iii. User clicks on an option from the list.
   iv. System creates a chat between the users.

We acknowledge that a user should also be able to search using a student number or staff number, but we are choosing not to include this to delimit the scope of the project.

7. Delete Message:

   i. User clicks on the “Delete” icon.
   ii. System creates a delete popup message stating: “You are about to delete this message are you sure you want to complete this action?”.
   iii. User clicks the “Delete” button.

We acknowledge that we should include the functionality to reply, share and forward messages as well as the ability to block users. However, for the purposes of this project we will not include these features.

Best-case Scenario

1. Steve clicks on Linda’s chat.
2. Steve clicks on the textbox in the chat.
3. System recognises the in text placing and allows Steve’s keyboard to enter information.
4. Steve enters a message through the keyboard.
5. Steve clicks the send message button.

6. System removes entered information from the textbox.
7. System sends the message to Linda and displays message to Steve.
8. Steve sees in his chat the message he sent.
9. Linda sees Steve's chat has risen to the top of the list of chats and there is a new message icon next to his contact archive.

Worst-case Scenario

1. Steve clicks on Linda's chat.
2. Steve clicks on the textbox in the chat.
3. System recognises the in text placing and allows Steve's keyboard to enter information.
4. Steve does not enter a message.
5. Steve clicks the send message button.
6. Since Steve entered nothing, the system refuses to send anything to Linda.
7. The cursor stays in the message box and waits for Steve to enter text.

Alternative Scenario A

1. Steve clicks the messages button.
2. System displays in his side bar the chats of Tom and Linda.
3. Steve clicks on Tom's chat but accidentally clicks on Linda's chat.
4. System displays all messages between Steve and Linda.

Alternative Scenario B

1. Steve clicks on the search bar.
2. System displays cursor in search bar.
3. Steve enters the name Donald Trump.
4. The system searches for Donald Trump but is not able to find a user with such a name.
5. System displays a list of the closest matches, while also displaying a message stating that it could not find a user named Donald Trump.

Alternative Scenario C

1. Steve clicks on the delete message button.
2. System creates a confirmation message stating: "Are you sure you want to delete this message".
3. User clicks "Confirm".
4. System permanently erases the message.
5. System closes the confirm or cancel popup message.
6. System replaces deleted messages with a deleted message icon.

Quiz

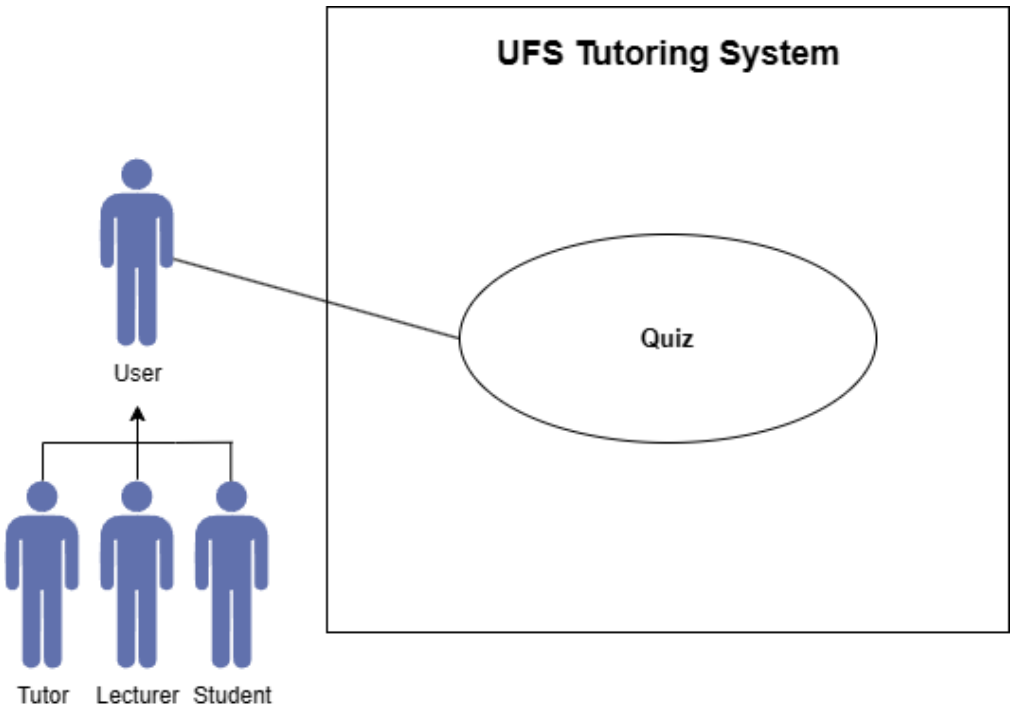


Figure 85: Quiz Use Case Diagram

Brief Description

The Quiz function allow a lecturer/tutor to create, edit, delete quizzes. A student can also complete a quiz multiple times each time unlocking a higher difficulty which can be completed.

Step-by-Step Description

<u>If the user is a student, they can:</u>

1. Complete Quiz.

For step 1 kindly refer to Figure 3 Student Dashboard.

i. User clicks the quiz button.

For steps 2-3 kindly refer to Figure 86 Student Quiz User Interface.

ii. System displays available quizzes.
iii. User clicks on a specific quiz.

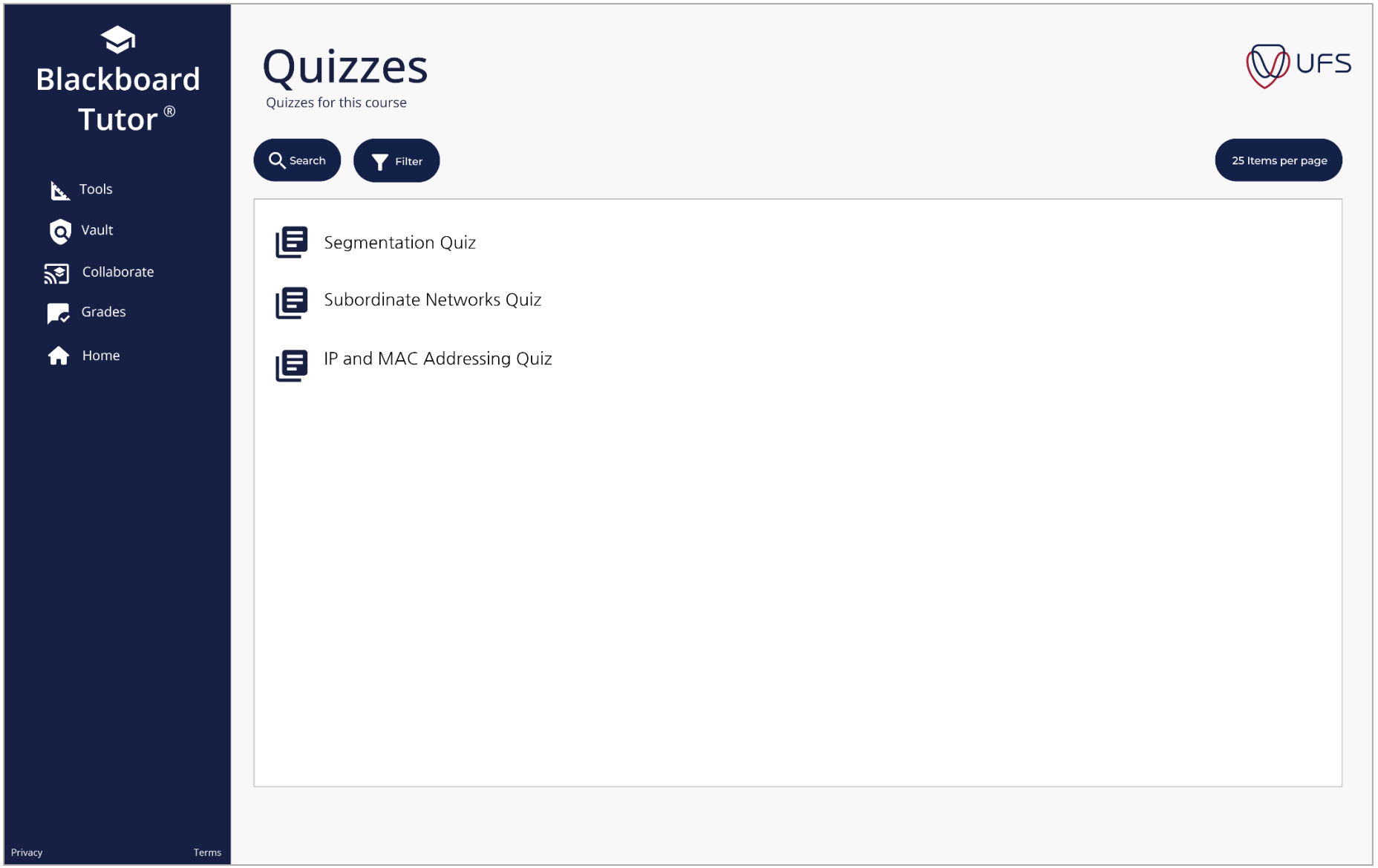


Figure 86: Student Quiz User Interface

iv. System displays a popup with possible quiz difficulties. Depending on if the user has completed the quiz and obtained the necessary mark, different difficulties are displayed.

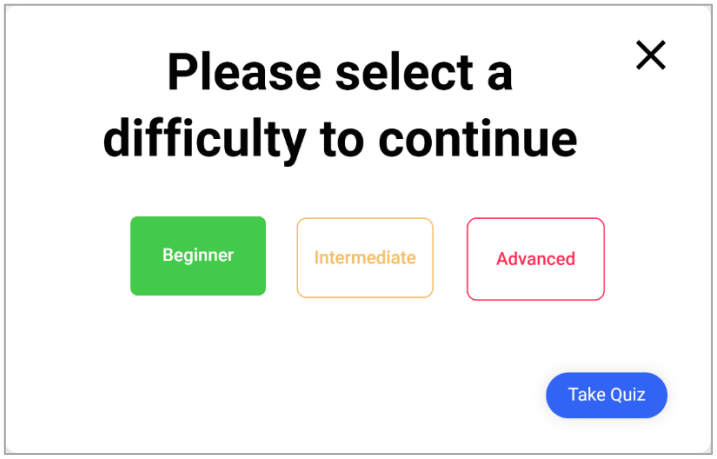


Figure 87: Quiz Difficulty - Beginner Selected

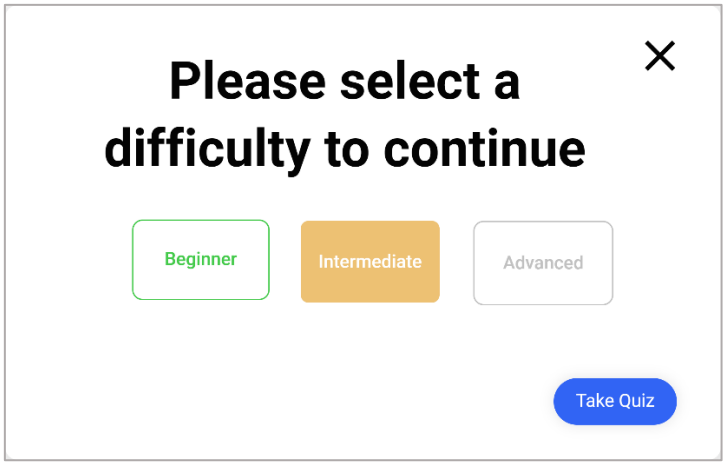


Figure 88: Quiz Difficulty - Intermediate selected & only few options available

v. User clicks on a difficulty level.

For steps 6-8 kindly refer to Figure 89 Student taking quiz.

vi. System opens the quiz of corresponding difficulty allowing the user to enter or select answers.
vii. User answers the quiz.

For steps 8 kindly refer to Figure 90 Quiz Information Popup.

viii. During the quiz, the user can click on the information button, which will display a popup containing quiz information.

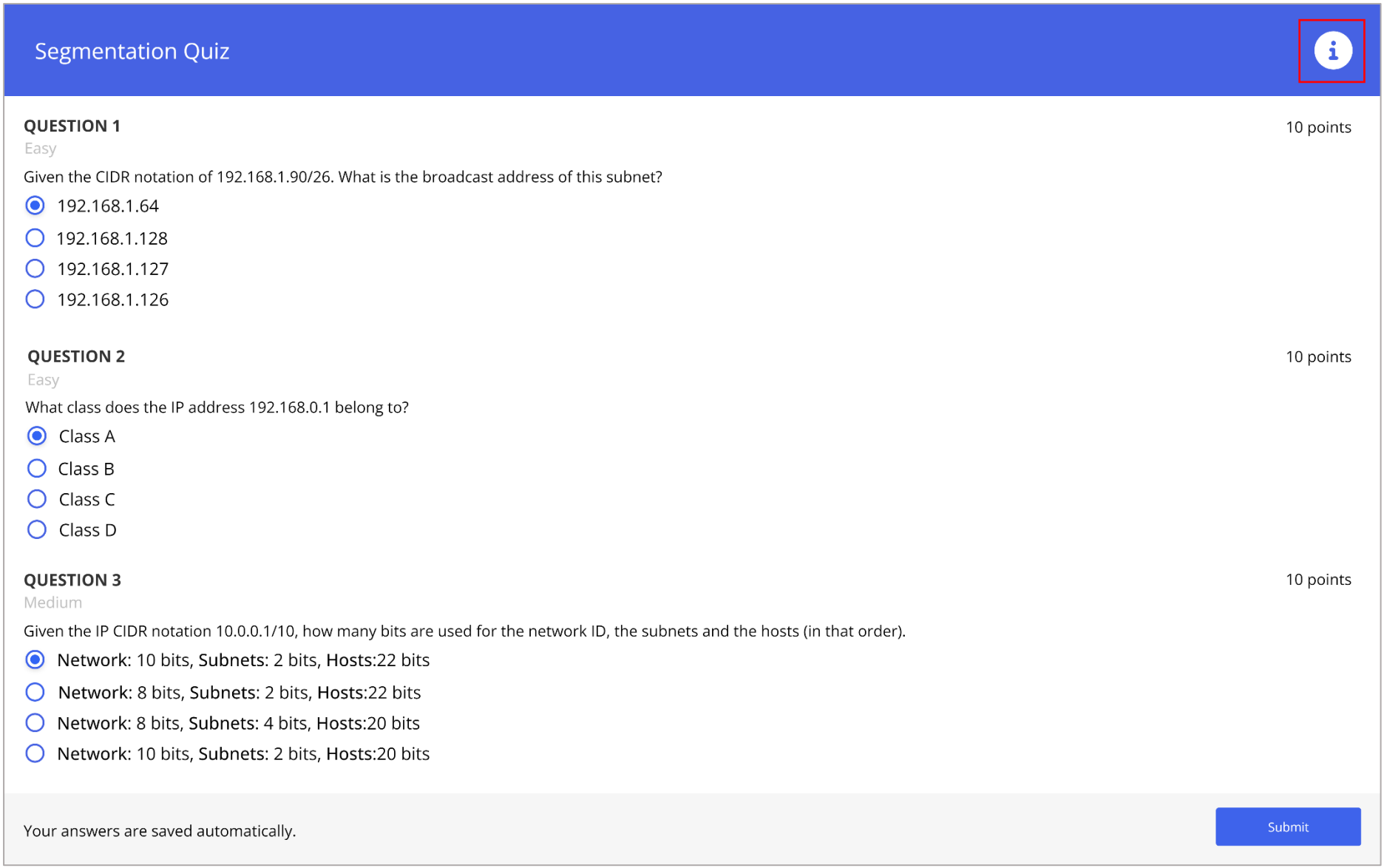


Figure 89: Student Taking Quiz User Interface

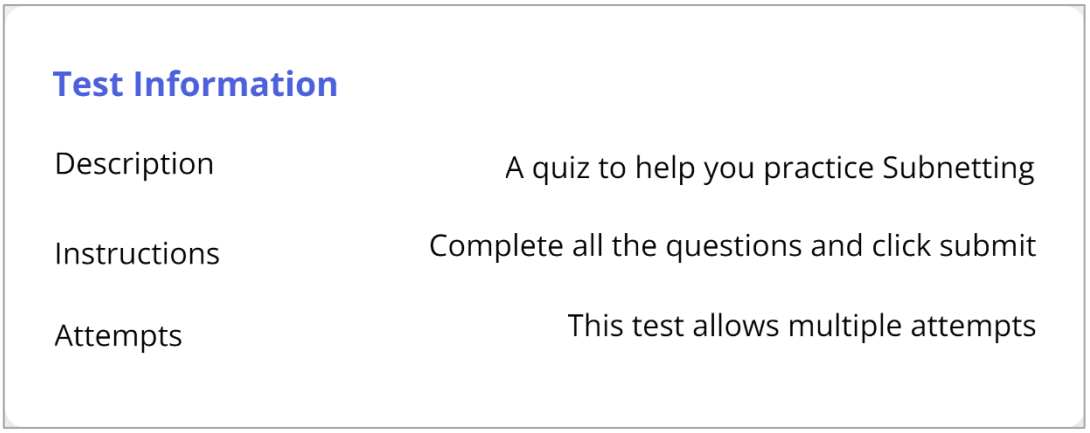


Figure 90: Quiz Information Popup

1. User clicks “Save and submit” button.
2. System creates a confirmation popup, stating: “Do you wish to save and submit the quiz answers?”.
3. User clicks “Confirm”.
4. System closes the confirmation popup.
5. System grades user quiz and stores the mark on the database.
6. System displays a popup of the obtained mark to the user.

Only a Tutor/Lecturer can make use of these features:

For step 1 kindly refer to Figure 1 Lecturer Dashboard.

1. Tutor/Lecturer clicks the “Manage Quizzes” button.

For steps 2-3 kindly refer to Figure 92 Lecturer/Tutor User Interface.

2. System displays available quizzes.
3. Tutor/Lecturer clicks on the drop-down button next to a specific quiz.
4. The following popup appears:

Kindly refer to Figure 91 below when a tutor/lecturer click on the dropdown arrow as indicated in 1 on Figure 92 Lecturer/Tutor Quiz User Interface.

Figure 91: Tutor/Lecturer Popup

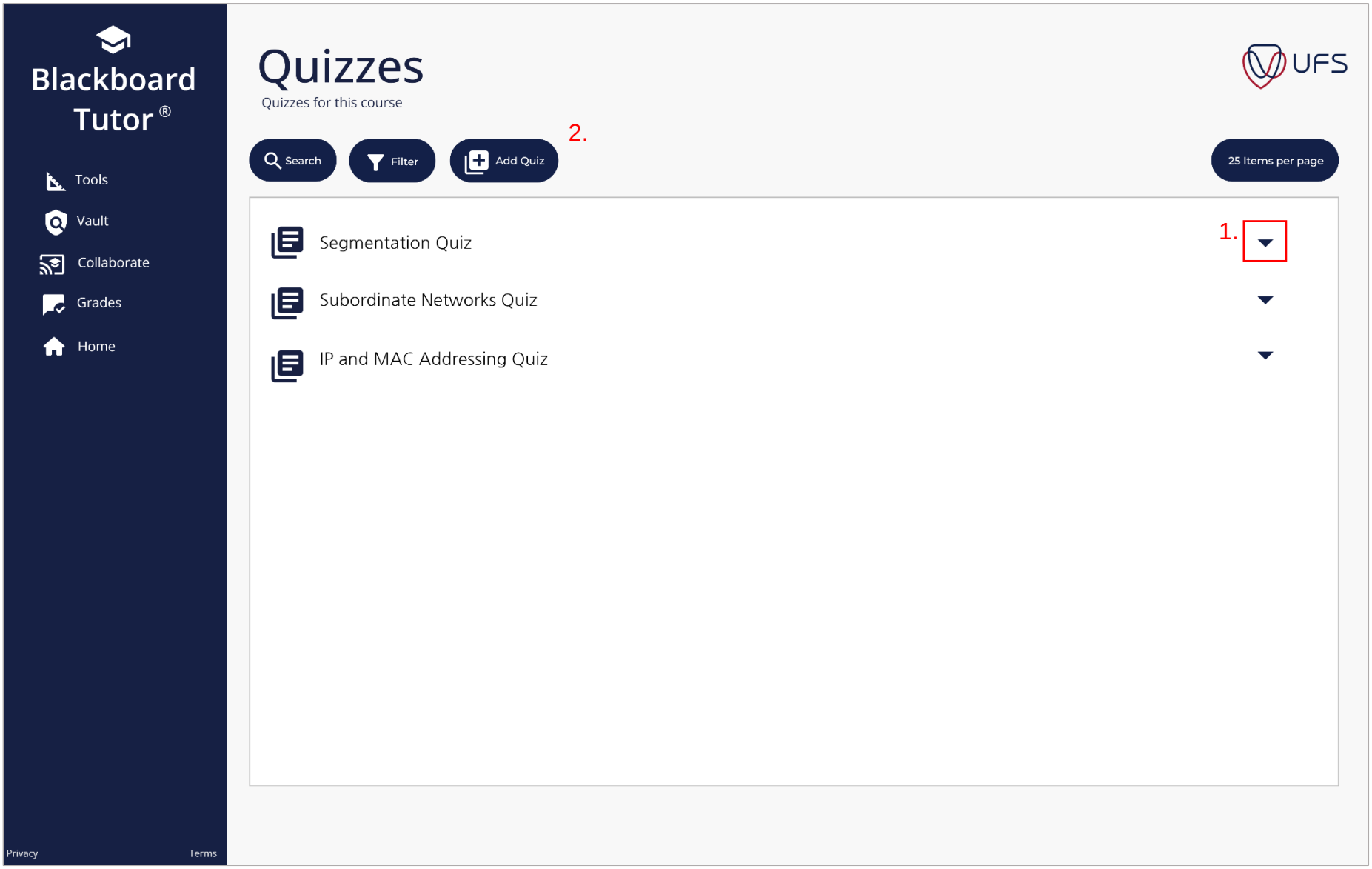


Figure 92: Lecturer/Tutor Quiz User Interface

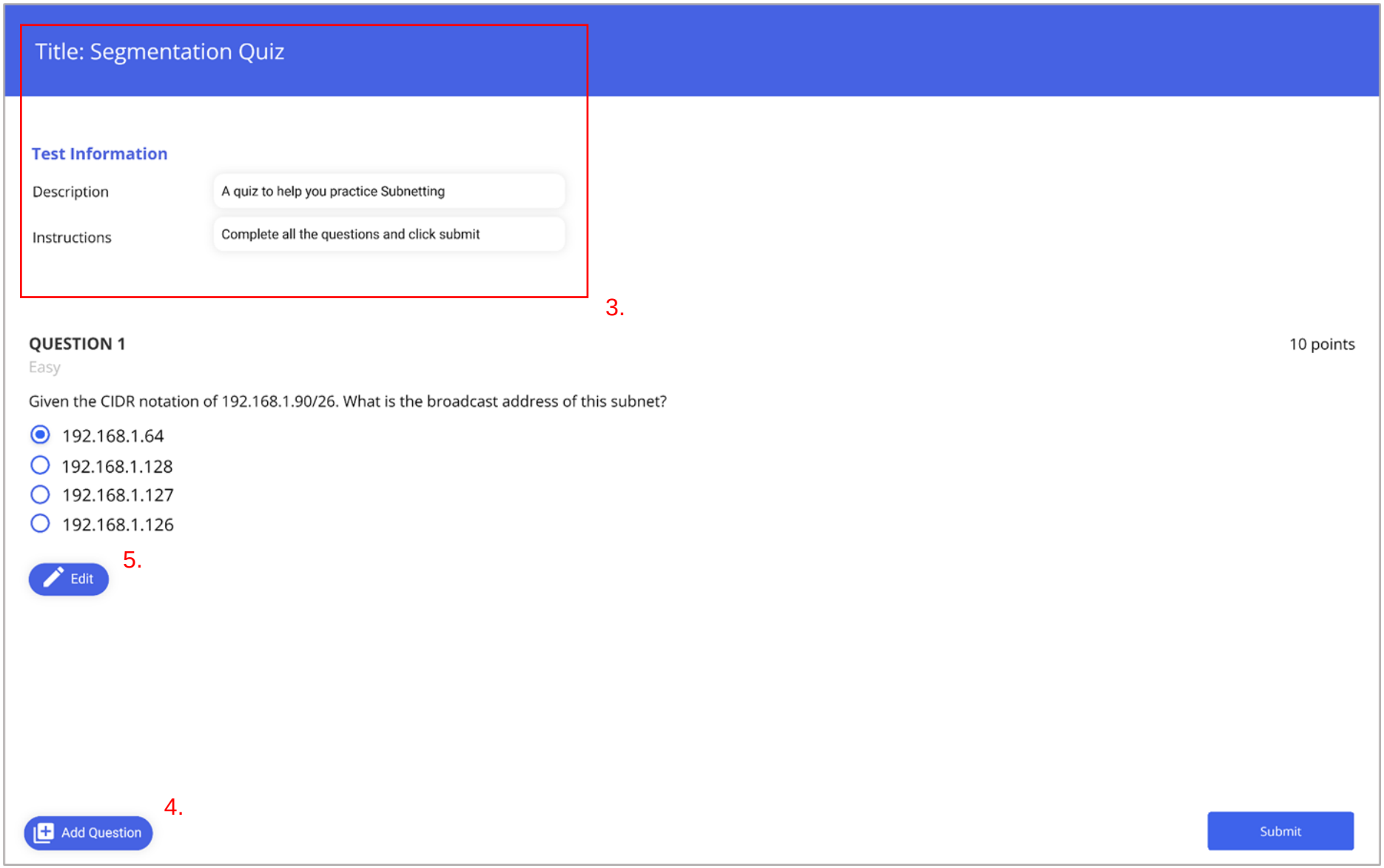


Figure 93: Tutor/Lecturer New Quiz

5. Create Quiz:

For step i kindly see Figure 91 Tutor/Lecturer Popup.

i. User clicks on the “Add Quiz” button.
ii. System displays “Create Quiz” popup. The page has adjustable quiz parameters for the user to populate. The parameters of the popup are title, description, instructions, date, time limit.
iii. User populates the quiz parameters.
iv. System checks if all altered parameters are acceptable.
v. System identifies that all altered parameters are acceptable.
vi. User clicks on the “Confirm” button.
vii. System displays the quiz (refer to the View Quiz step).
viii. User clicks the “Submit” button to create the quiz.

6. Edit Quiz:

For step i kindly see Figure 91 Tutor/Lecturer Popup.

i. User clicks on the dropdown arrow next to a quiz.
ii. User clicks on the “Edit” button.
iii. System displays the quiz page has adjustable quiz parameters for the user to populate. description, instructions, date, time limit of the quiz.
iv. User alters parameters.
v. System checks if all altered parameters are acceptable.
vi. System identifies that all altered parameters are acceptable.
vii. System saves the quiz changes.

7. View Quiz:

For step i kindly see Figure 91 Tutor/Lecturer Popup.

i. User clicks on the dropdown arrow next to a quiz.
ii. User clicks on the “View” button.
iii. System displays the quiz, with options to add questions and edit specific questions.
iv. User clicks the “Submit” button to save any changes they made to the quiz.
v. System checks if all parameter values are valid.
vi. System saves all edited parameters.

8. Delete Quiz:

For step i kindly see Figure 91 Tutor/Lecturer Popup.

i. User clicks on the dropdown arrow next to a quiz.
ii. User clicks on the “Delete” button.

For step iii kindly see Figure 94 Delete Quiz Popup.

iii. System creates a popup, stating: “You are about to delete a quiz. Are you sure you want to complete this action?”.

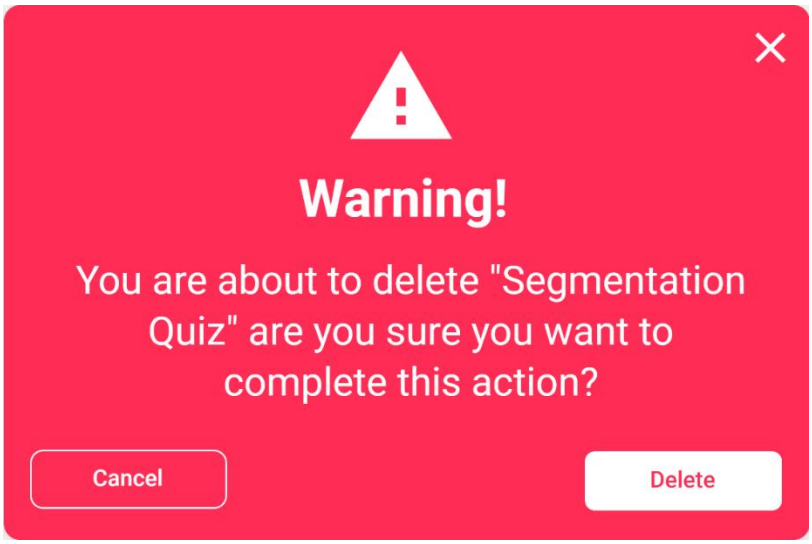


Figure 94: Delete Quiz Popup

iv. User clicks “Delete”.
v. System permanently erases the quiz.
vi. System closes the confirmation popup message.

9. Add Question:

For step i kindly see 4 in Figure 93 Tutor/Lecturer New Quiz.

i. User clicks the “Add Question” button.
ii. System displays a add question popup form with adjustable question parameters for the user to populate. The parameters of the popup are question, question difficulty level, points, options, and answer.

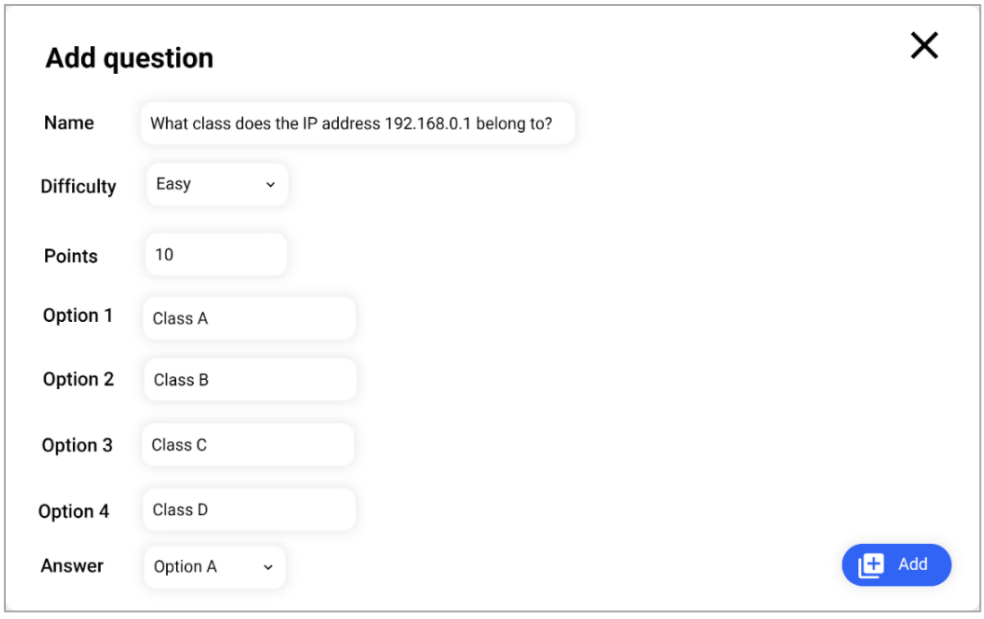


Figure 95: Add Question Popup

iii. User populates form.
iv. User clicks on the “Add” button.
v. System adds the question to the quiz popup form.
vi. User clicks “Confirm”.
vii. System checks if all populated parameters are acceptable.
viii. System identifies that all populated parameters are acceptable.
ix. System creates the question.
x. System closes the popup form.

10. Edit Question:

For step i kindly see 5 in Figure 94 Tutor/Lecturer New Quiz.

i. When the user is in the “Create Quiz/Edit Quiz” page they can click on the “Edit Question” button that will be displayed beneath each question.

For steps ii-vi kindly see Figure 96 Edit Question Popup.

ii. An “Edit Question” popup will display with parameters for the question’s name, difficulty, points, four answer options and a correct answer choice.
iii. The user alters the parameters.
iv. The user clicks the change button.
v. System checks if all parameter values are valid.
vi. System saves all edited parameters
vii. The popup closes.
viii. The altered question will now be displayed on the quiz page.

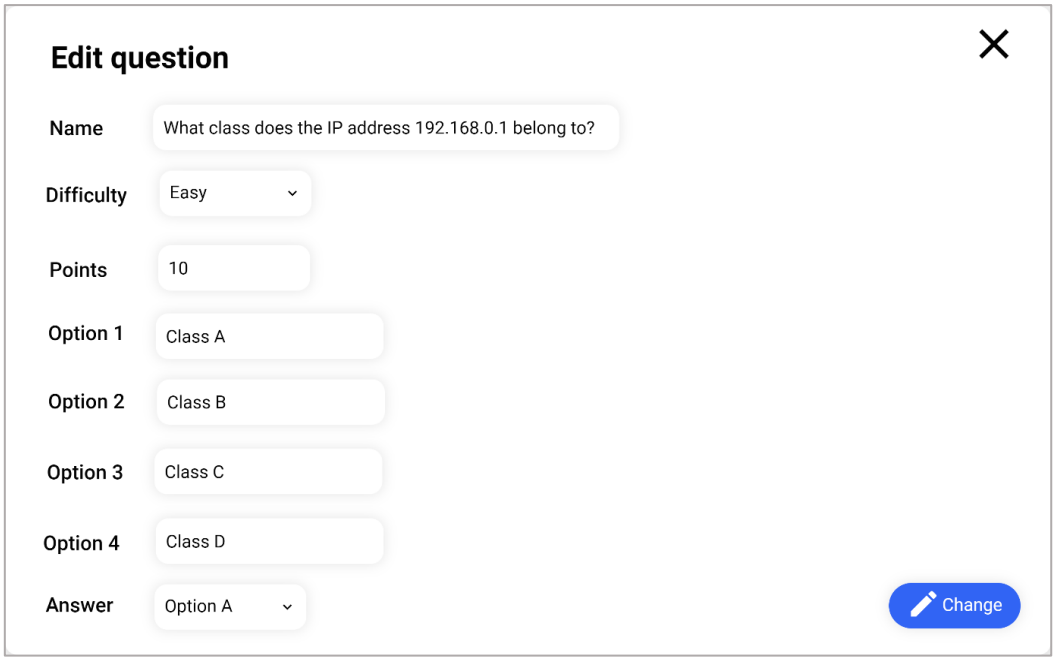


Figure 96: Edit Question Popup

Best-case Scenario

1. Tom (a tutor) clicks on create quiz button.
2. System displays a popup form with adjustable quiz parameters for Tom to populate. The parameters of the quiz are name, description, instructions, date, time limit.
3. Tom populates the form.
4. Tom clicks confirm.
5. System checks if all populated parameters are acceptable.
6. System identifies that all populated parameters are acceptable.
7. System creates the quiz.
8. System closes the popup form.

Worst-case Scenario

1. Tom (a lecturer) clicks on the “Add Quiz” button.
2. System displays a popup form with adjustable quiz parameters for Tom to populate. The parameters of the quiz are name, description, instructions, date, time limit, questions, answers, and question difficulty level.
3. Tom clicks the “Confirm” button.
4. System displays the quiz.
5. Tom clicks on the “Submit” button without adding any questions to the quiz.

For step 6 kindly refer to Figure 97 Can’t upload quiz error.

6. System displays an error message stating: “Error! You can’t upload a quiz without questions!”.
7. System won’t allow Tom to submit without adding at least one question. If Tom closes the page, then the quiz will be lost and he will have to create a new one.

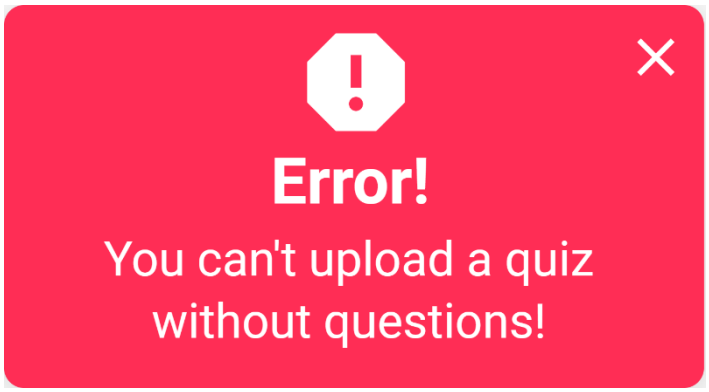


Figure 97: Can't upload quiz error

Alternative Scenario A

1. Tom (a tutor) clicks on “Create” button.
2. System displays a popup form with adjustable quiz parameters for Tom to populate. The parameters of the quiz are name, description, instructions, date, time limit, questions, answers, and question difficulty level.
3. Tom populates the form but does not add instructions.
4. Tom clicks confirm.
5. System checks if all populated parameters are acceptable.
6. System identifies that not all populated parameters are acceptable.
7. The system does not add the quiz.
8. System displays an error message stating: “Error! Please add instructions!”.

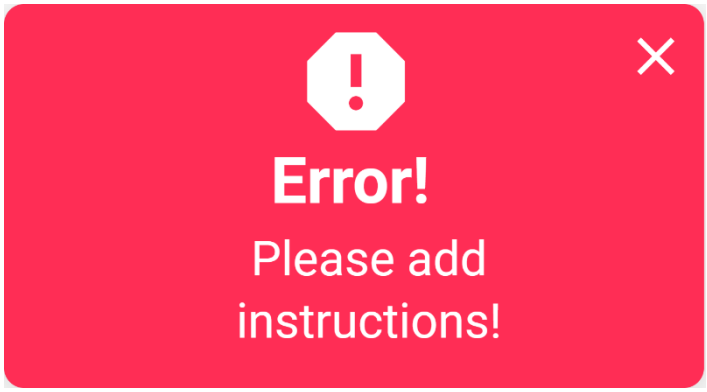


Figure 98: Add instructions error

Alternative Scenario B

1. Tom (a tutor) clicks on “Create” button.
2. System displays a popup form with adjustable quiz parameters for Tom to populate. The parameters of the quiz are name, description, instructions, date, time limit, questions, answers, and question difficulty level.
3. Tom populates the form but does not add a description.
4. Tom clicks confirm.
5. System checks if all populated parameters are acceptable.
6. System identifies that not all populated parameters are acceptable.
7. The system does not add the quiz.
8. System displays an error message stating: “Error! Please add a description!”.

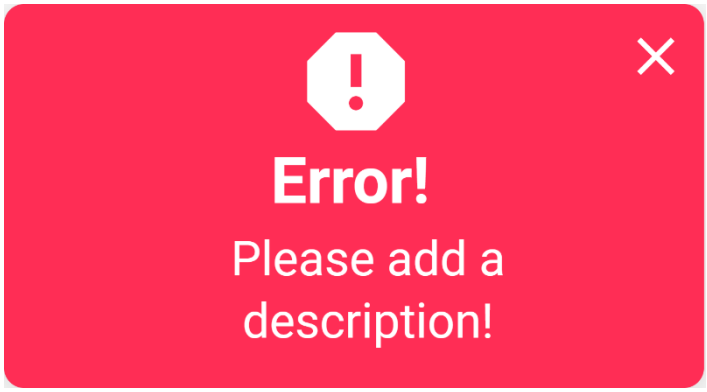


Figure 99: Add description error

Alternative Scenario C

Complete Quiz

1. Steve clicks the quiz button.
2. System displays available quizzes.
3. Steve clicks on a specific quiz.
4. System displays a popup with possible quiz difficulties.
5. Steve clicks on a difficulty level.
6. System opens the quiz of corresponding difficulty allowing Steve to enter or select answers.
7. Steve answers the quiz.
8. Steve runs out of time and the quiz automatically saves and submits.
9. System grades Steve answer(s) storing the acquired mark in grades.
10. System displays a popup of the obtained mark to the Steve.
11. System stores the completed quiz in the assessment vault.

Alternative Scenario D

Complete Quiz

1. Rob is completing a quiz on his personal computer. The power fails in the middle of a quiz, and he is disconnected from the quiz.
2. Rob logs into the system on his phone. When he logs in, he is automatically taken to the quiz and can complete the quiz.

Leaderboard

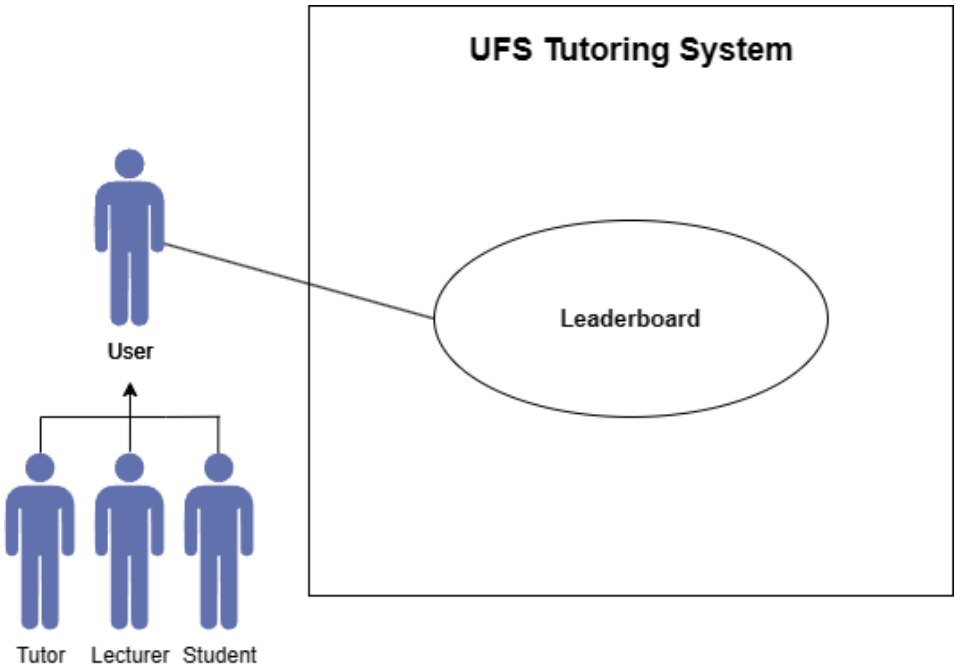


Figure 100: Leaderboard Use Case Diagram

Brief Description

The Leaderboard function allows users to view a leaderboard of the top students with the most points on the system. The functionality also allows a student to view their position on the leaderboard.

Step-by-Step Description

1. Student clicks on the “Leaderboard” button on the dashboard.
2. The first time a student accesses the leaderboard the system displays a pop-up asking for permission to show the student’s name on the leaderboard.

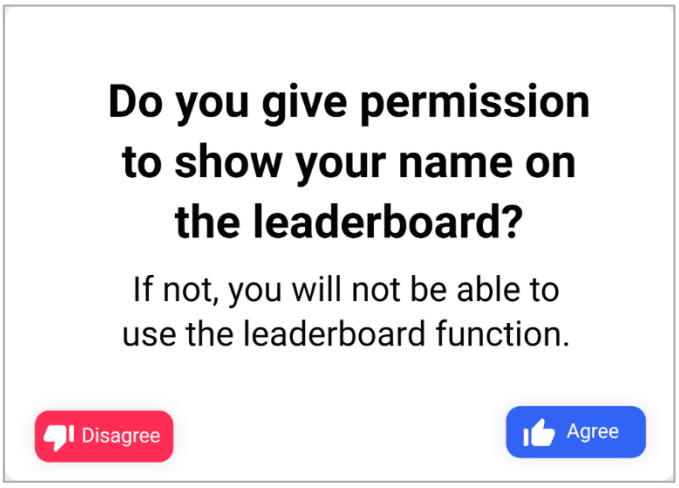


Figure 101: Leaderboard Permission Popup

3. Student clicks on agree and they will be taken to the leaderboard.

For step 4, kindly see Figure 104 Student Leaderboard User Interface.

4. System displays the students name on the leaderboard if they are one of the top 5 students with the most points.
5. The system has three options to sort the leaderboard:
    - All Time - students who have the most points for that year.
    - Week - students that have won the most points for that current week.
    - Month - students who have won the most points for the current month.
6. System displays the leaderboard with the students' names and their points.
7. System displays a button that says my ranking.
8. Student clicks on "My Ranking".
9. System displays the students ranking and the number of points that they have received in the form of a popup.

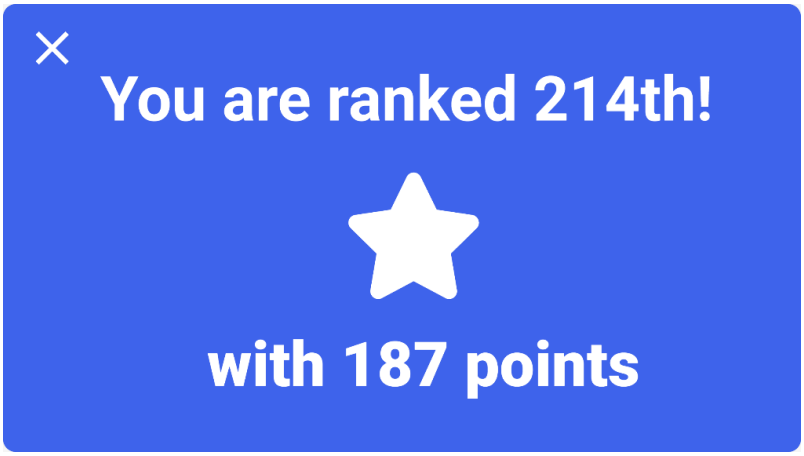


Figure 102: Ranking Popup

Best-case Scenario

1. Kamo clicks on leaderboard for the first time.
2. System displays a pop-up asking Kamo for permission to show his name and points in the leaderboard.
3. Kamo clicks on agree.
4. System displays the leaderboard with the names and points of the top 5 students who have agreed to show their details on the leaderboard.
5. Kamo clicks on my ranking button.
6. System displays Kamo's ranking in the form of a pop-up.

Worst-case Scenario

1. Jocelyne clicks on the leaderboard for the first time.
2. System displays a pop-up asking Jocelyne for permission to show her name and points on the leaderboard.
3. Jocelyne clicks on disagree.
4. System displays Jocelyne's dashboard again.

Alternative Scenario A

1. Chris clicks on the leaderboard button, but it is not his first time, last time he clicked on agree.
2. System displays the leaderboard.

Alternative Scenario B

1. Kyle clicks on the leaderboard button, it is not his first time, last time he clicked on disagree.
2. System displays a warning pop-up asking Kyle for permission alerting him that he had previously disagreed to show his information in the leaderboard.
3. Kyle clicks on agree.
4. System displays the leaderboard.

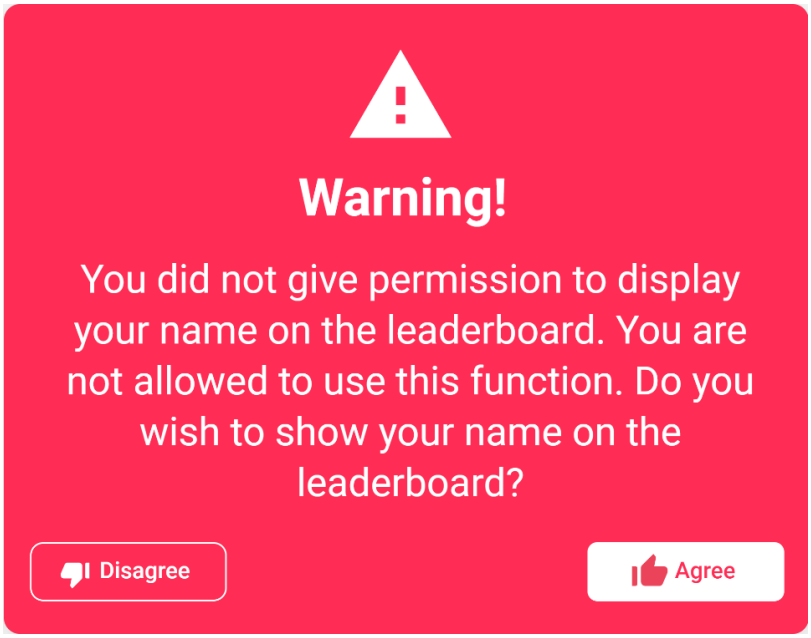


Figure 103: Leaderboard Permission Warning Popup

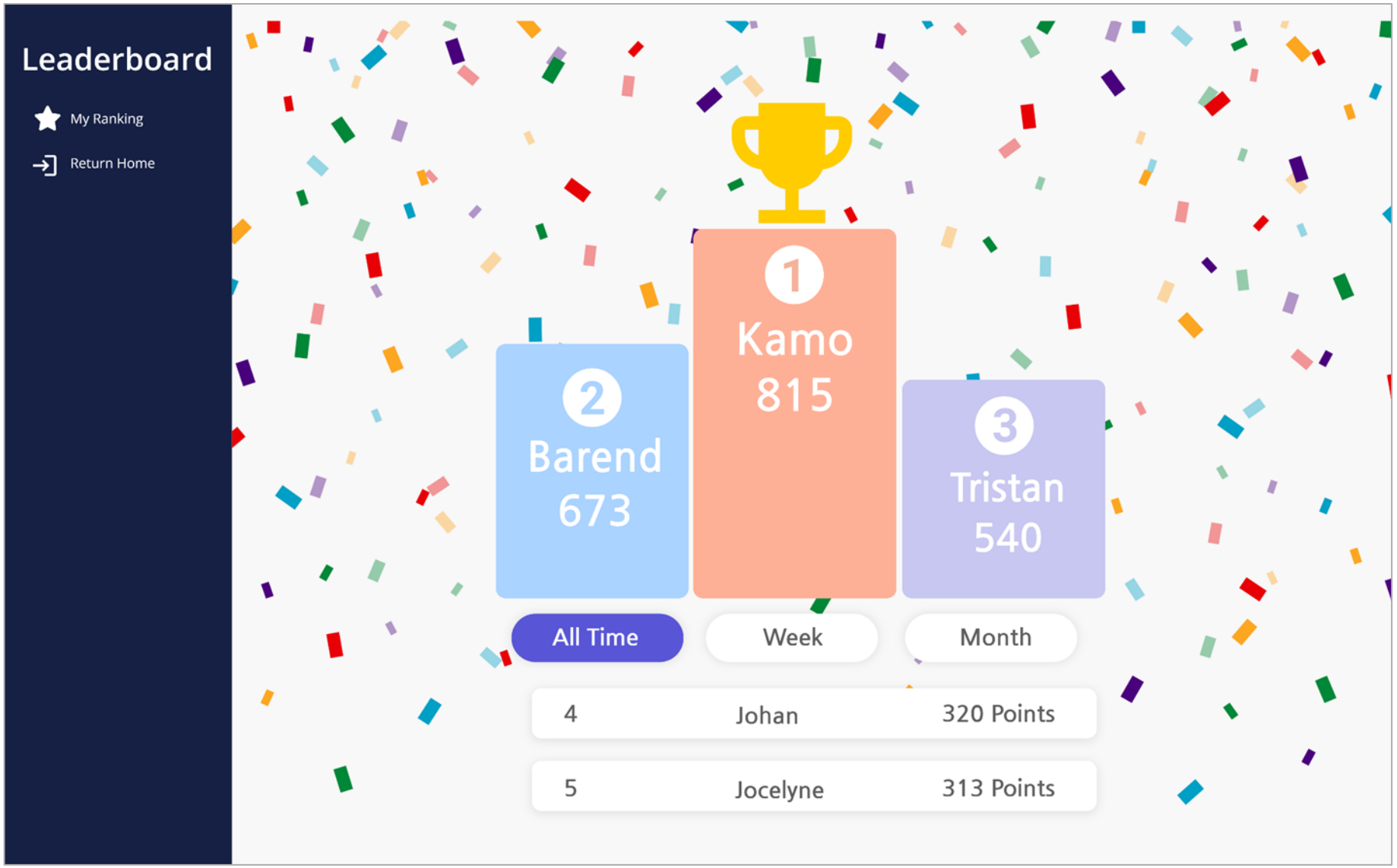


Figure 104: Student Leaderboard User Interface

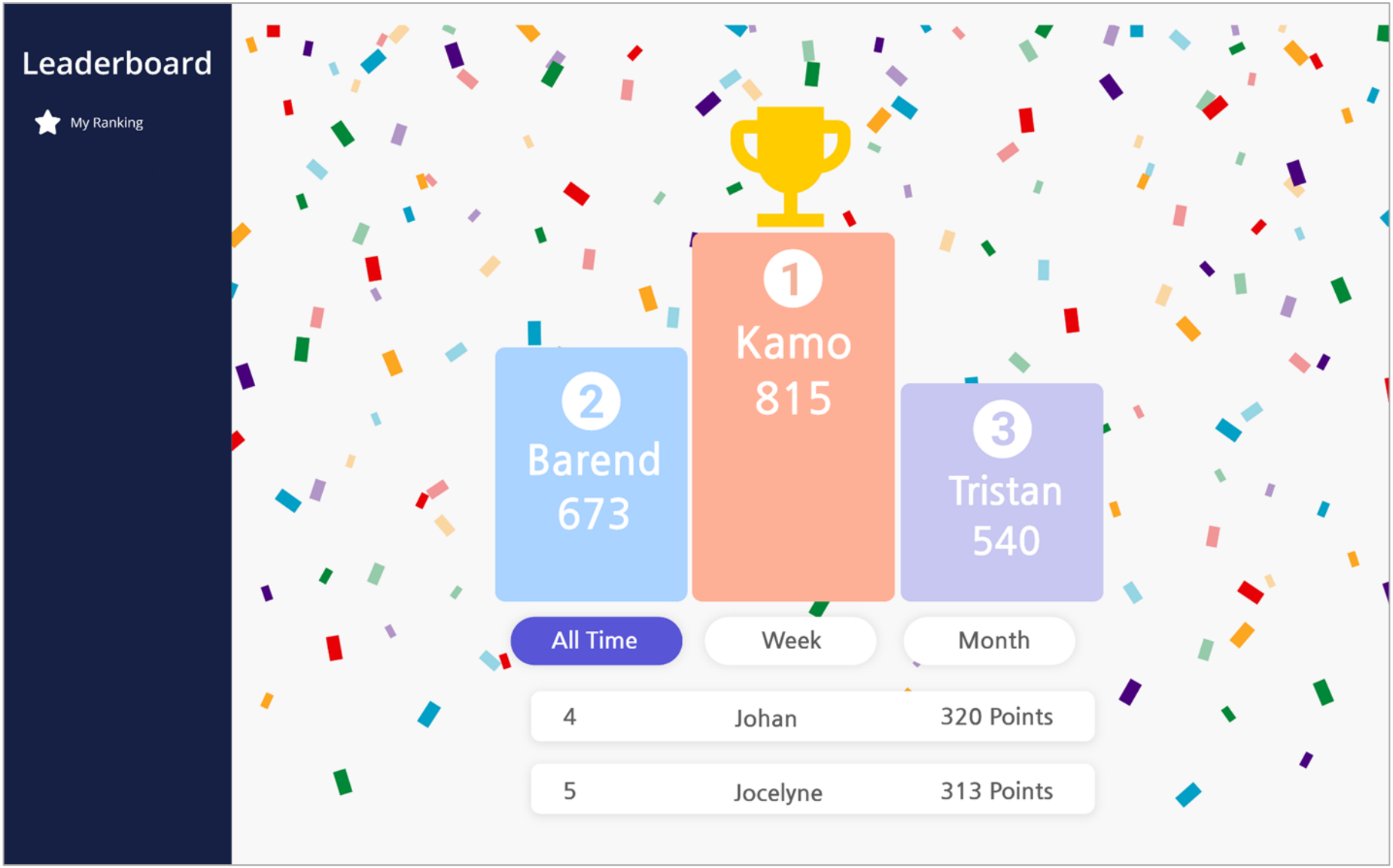


Figure 105: Lecturer/Tutor Leaderboard User Interface

View Badges

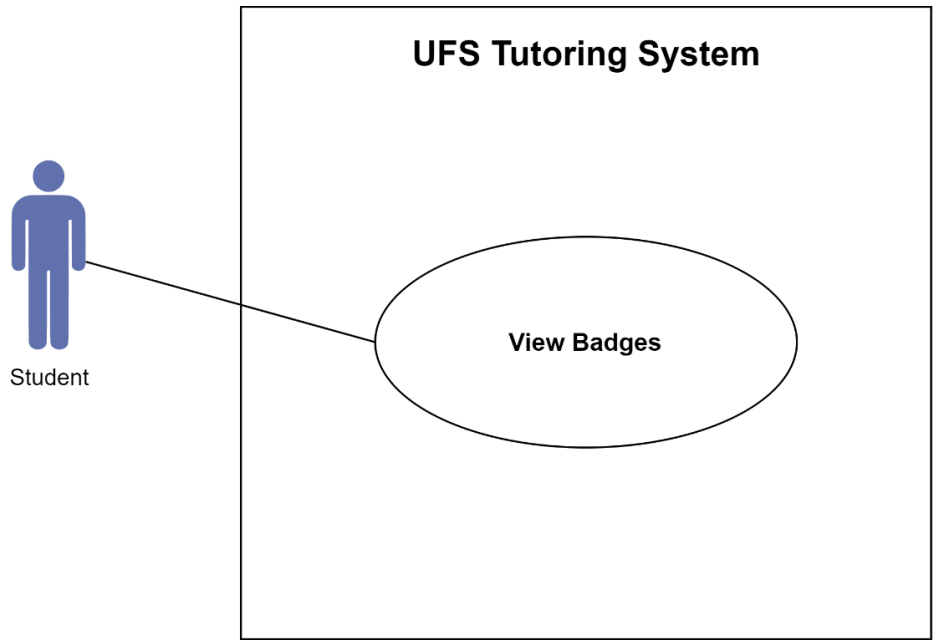


Figure 106: View badges Use Case Diagram

Brief Description

The View Badges function allows a student to view badges that they have won for completing tasks throughout the semester.

We acknowledge that lecturers should be able to add, edit and delete badges that can be awarded to students, but for the purposes of this project we will not implement this feature.

Step-by-Step Description

1. Student clicks on the “Badges” button on the dashboard.
2. System displays all their badges that the student has won throughout the semester. The current available badges that the student can win are awarded for:
   - When they have attended 80% of the tutor sessions.
   - Achieving a 75% average for their quizzes.
   - Points that are awarded for every coding challenge completed.

Best-case Scenario

1. Kamo clicks on the “Badges” button on the dashboard.
2. The system displays all the badges Kamo has won throughout the semester.

Worst-case Scenario

1. Jocelyne clicks on the “Badges” button on the dashboard.
2. The system displays no badges since Jocelyne has not won any badges this semester yet.

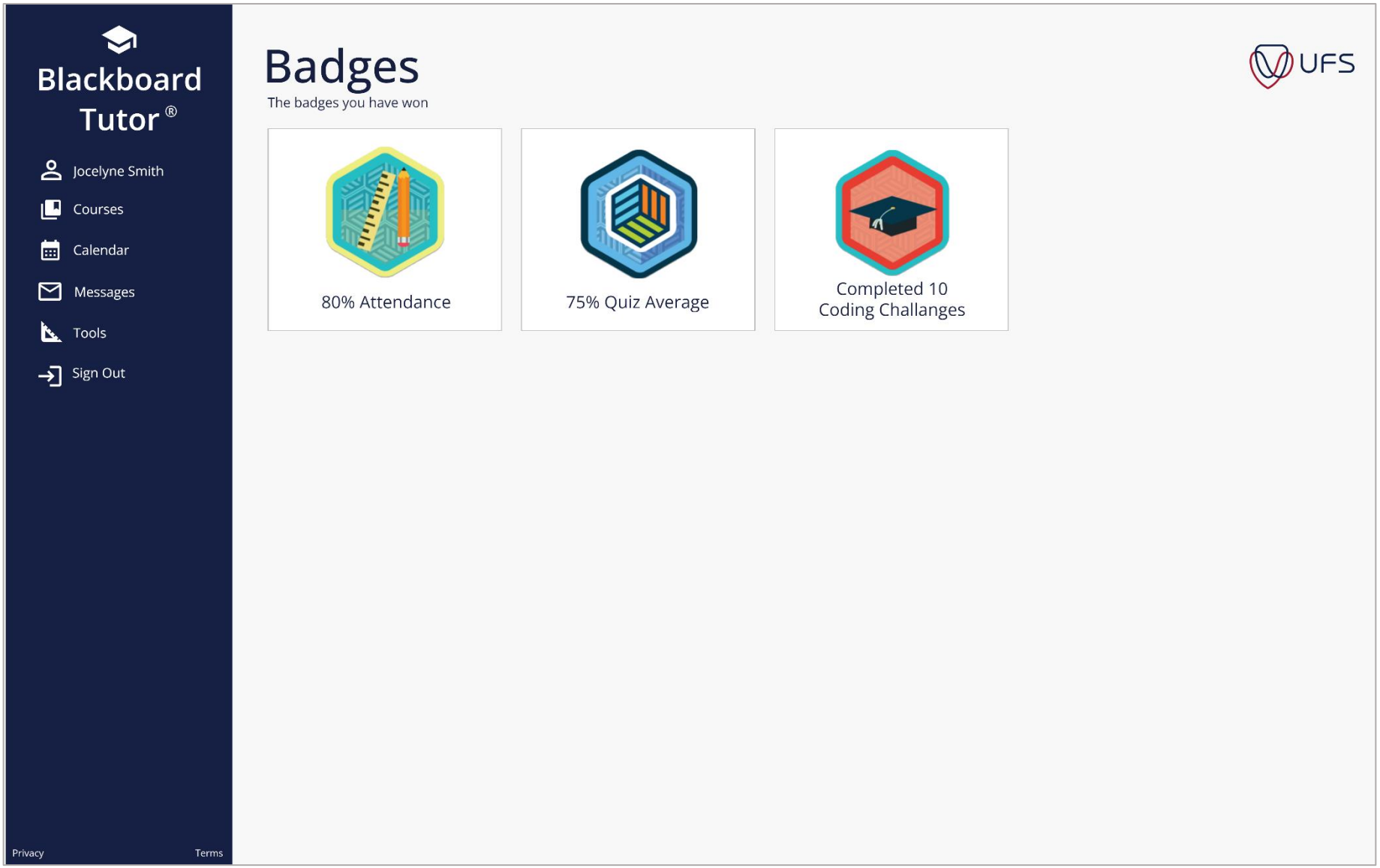


Figure 107: Student Badges User Interface

Coding Problems

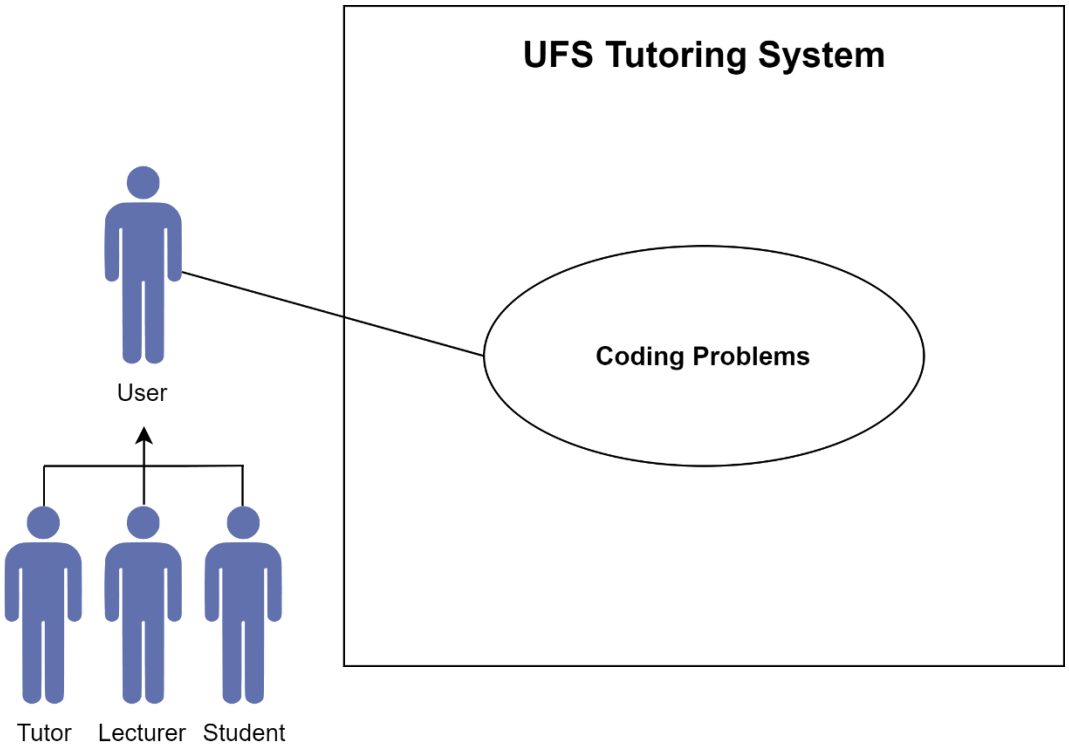


Figure 108: Coding Problems Use Case Diagram

Brief Description

The Coding Challenges function allows a user to solve a coding problem and be graded by the system.

Step-by-Step Description

1. User clicks on the “Coding Challenge” button in the dashboard.
2. System displays a screen with Beginner, Intermediate and Advanced coding problems that are available for them to solve.
3. User clicks on a problem.
4. System displays a coding environment with a text editor and a terminal where the user will see the output of their code. There is also a sidebar with the coding question that the user must solve.
5. User attempts to solve the coding question.
6. User clicks on the “Run” button.
7. System compiles the code. System displays output in terminal.
8. User continues to code until they are satisfied with the code output.
9. User clicks on the “Submit” button.
10. System grades the code.
11. System displays a congratulations message to the user.

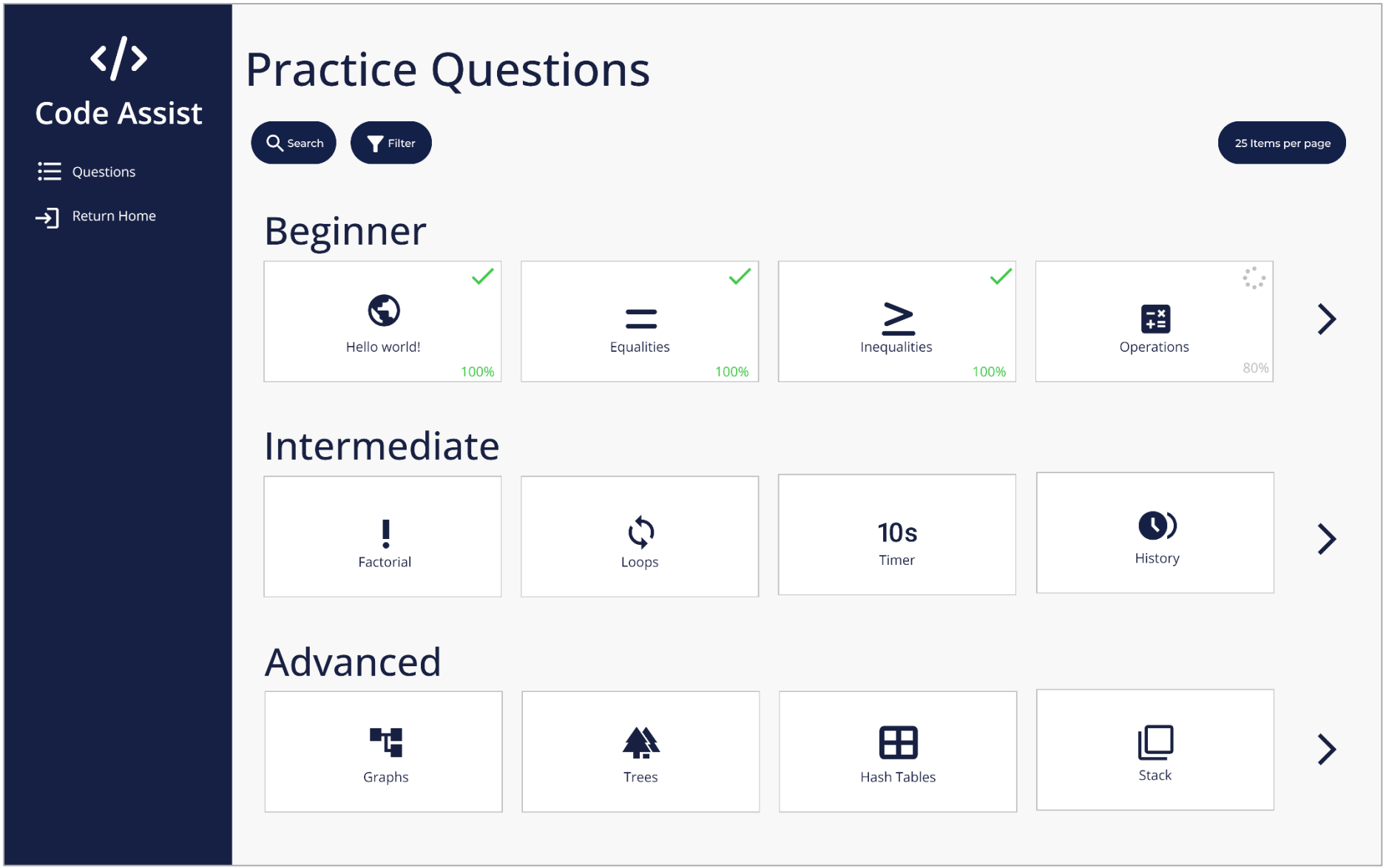


Figure 109: Coding Problems Home page User Interface

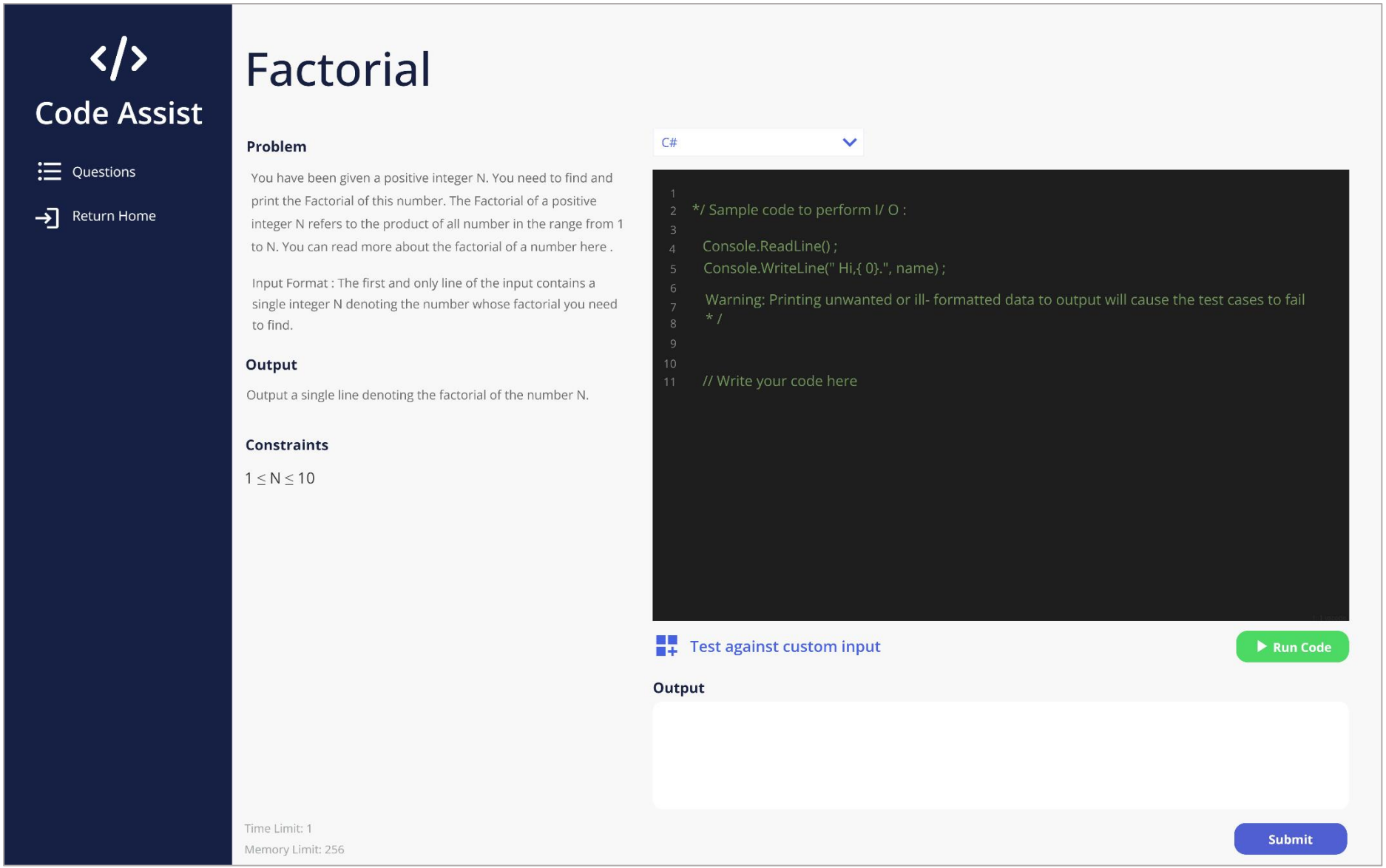


Figure 110: Attempt Coding Problem User Interface

Best-case Scenario

1. Kamo clicks on the “Coding Challenge” button.
2. System displays a screen with Beginner, Intermediate and Advanced coding problems that are available for Kamo to solve.
3. Kamo clicks on the Factorial problem.
4. System displays a coding environment with a text editor and a terminal where Kamo will see the output of his code. There is also a sidebar with the coding question that Kamo must solve.
5. Kamo attempts to solve the coding question.
6. Kamo clicks on the “Run” button.
7. System compiles the code. System displays output in terminal.
8. Kamo continues to code until he is satisfied with the code output.
9. Kamo clicks on the “Submit” button.
10. System grades the code.
11. System displays a congratulations message to the user.

Worst-case Scenario

1. Jocelyne clicks on coding challenges.
2. System shows Jocelyne the coding screen with sections for beginner, intermediate and advanced coding problems.
3. Jocelyne clicks on a problem.
4. System shows Jocelyne the coding environment and coding question.
5. Jocelyne does not attempt the question.
6. Jocelyne clicks on the exit button.
7. System takes her back to the coding problems screen.

Alternative Scenario

1. Kyle clicks on coding challenges.
2. System shows Kyle the coding screen with sections for beginner, intermediate and advanced coding problems.
3. Kyle clicks on a problem.
4. System shows Kyle the coding environment and coding question.
5. Kyle attempts the coding problem.
6. Kyle then presses the exit button.
7. System takes him back to the coding problems screen.

Figure 111: Chat feature in the Virtual Classroom

Manage Tutors

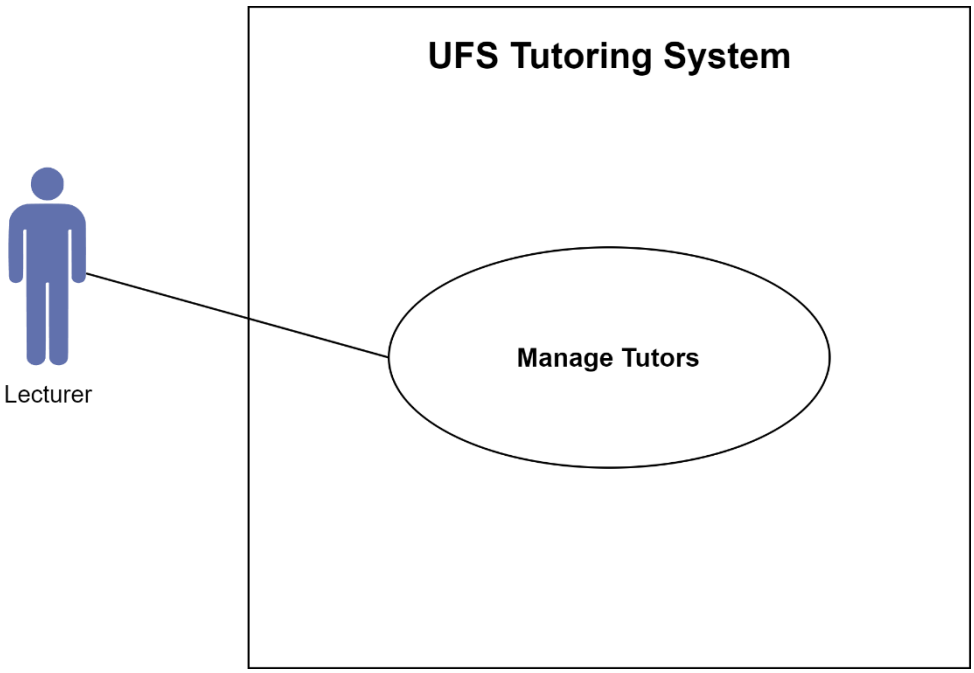


Figure 112: Manage Tutors Use Case Diagram

Brief Description

The manage tutor use case allows a lecturer to approve prospective tutors and remove problematic tutors.

Step-by-Step Description

Figure 113: Lecturer Popup

For steps 1-3 kindly refer to Figure 115 Lecturer Manage Tutors User Interface.

1. Lecturer clicks the “Manage Tutors” button on the dashboard.
2. System displays a list of tutors, showing current tutors and students who have applied to be tutors.
3. Lecturer clicks the drop-down arrow next to a specific tutor’s name on the right-hand side.

If the student is already a tutor, they can:

4. Remove Tutor.
   i. Lecturer clicks on the “Remove Tutor” button.
   ii. A dialog warns the lecturer that they will removing this tutor.
   iii. Lecturer clicks on the OK button to remove the tutor.
   iv. The popup closes.
   v. The tutor is removed from the list.

If the student applied to be a tutor, they can:

5. Approve Application:

   i. Lecturer clicks on the “Approve Application” button.
   ii. A popup appears, showing the student’s name, their student number, the module they applied for, the motivating text they added and a button to view the supporting documentation. The lecturer can reject or approve the application.
   iii. Lecturer clicks the button to view the supporting documentation.
   iv. The pdf opens in a new tab.
   v. Lecturer closes the new tab and returns to the page with the popup.
   vi. Lecture clicks on “Approve” button.
   vii. A popup appears, asking the lecturer if they are sure they want to approve this student as a tutor.
   viii. Lecturer clicks OK and both popups close.

For step 5 kindly refer to Figure 114 View Application Popup.

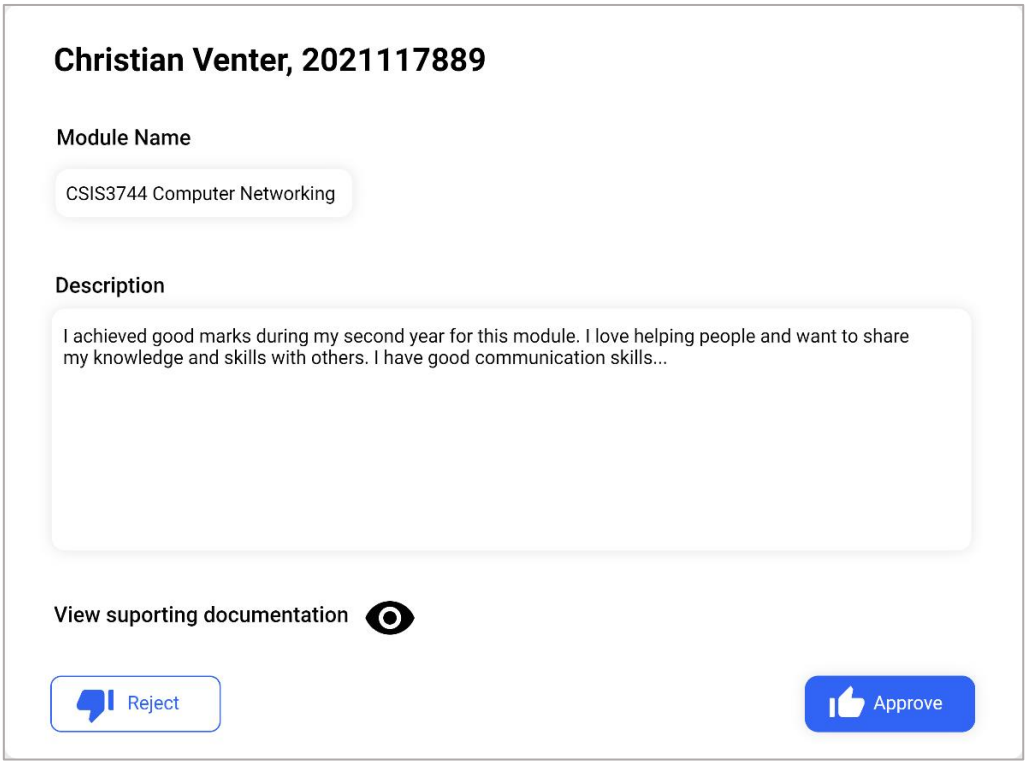


Figure 114: View Application Popup

Best-case Scenario

1. Lizel clicks the manage tutor button on the dashboard.
2. A list of tutors is displayed, showing current tutors and students who have applied to be tutors.
3. Lizel clicks the drop-down arrow next to Danny's name (a tutor who must be removed because of misconduct) on the right-hand side.
4. A context menu with a "Remove Tutor" option will be displayed.
5. Lizel clicks on the "Remove Tutor" button.
6. A dialog warns the lecturer that they will be removing Danny as a tutor.
7. Lizel clicks on the OK button to remove Danny.
8. The popup closes.
9. Danny is removed from the list.

Worst-case Scenario

1. Lizel clicks the "Manage Tutors" button on the dashboard.
2. Lizel does not have any tutors yet.
3. The system will display that there are no tutors instead of trying to fetch the non-existent tutors (which would result in a crash).

Alternative scenario A

1. Lizel clicks the manage tutor button on the dashboard.
2. A list of tutors is displayed, showing current tutors and students who have applied to be tutors.
3. Lizel clicks the drop-down arrow next to Danny's name (a 2nd year student who applied to be a tutor) on the right-hand side.
4. A context menu with a "Approve Application" option will be displayed.
5. Lizel clicks on the "Approve Application" button.
6. A popup appears, showing the Danny's information.
7. Lizel clicks on "Reject" button.
8. A popup appears, stating: "Are you sure you want to reject Danny as a tutor?"
9. Lizel clicks OK and both popups close.
10. Danny is removed from the list on the "Manage Tutors" page.

Alternative scenario B

1. Lizel clicks the manage tutor button on the dashboard.
2. A list of tutors is displayed, showing current tutors and students who have applied to be tutors.
3. Lizel clicks the drop-down arrow next to Danny's name (a tutor who must be removed because of misconduct) on the right-hand side.
4. A context menu with a "Remove Tutor" option will be displayed.
5. Lizel clicks on the "Remove Tutor" button.
6. A dialog warns Lizel that they will removing Danny as a tutor.
7. Lizel clicks cancel.
8. The system does nothing. The screen is the same as when they entered the manage tutors screen.

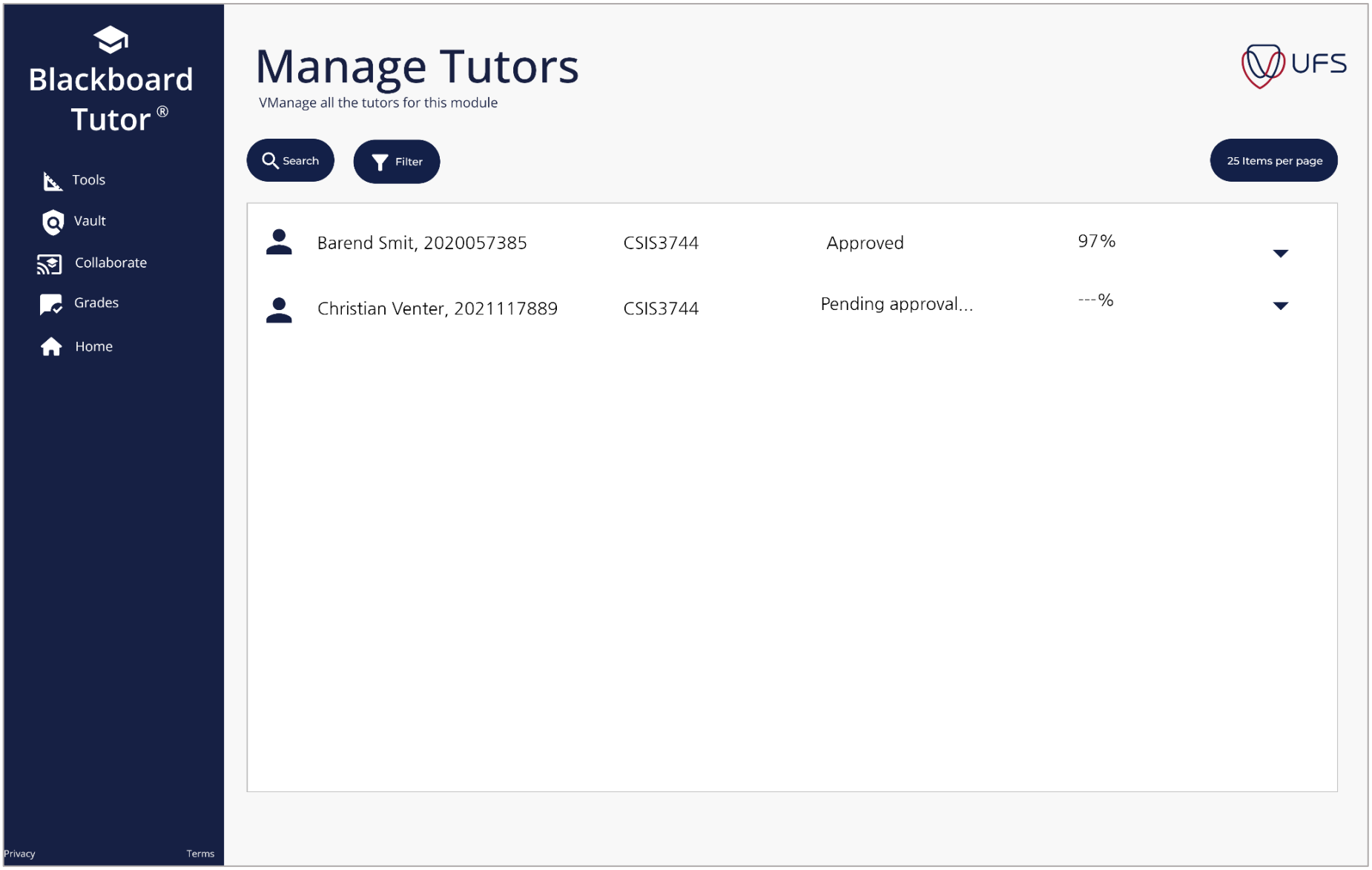


Figure 115: Lecturer Manage Tutors User Interface

Calendar

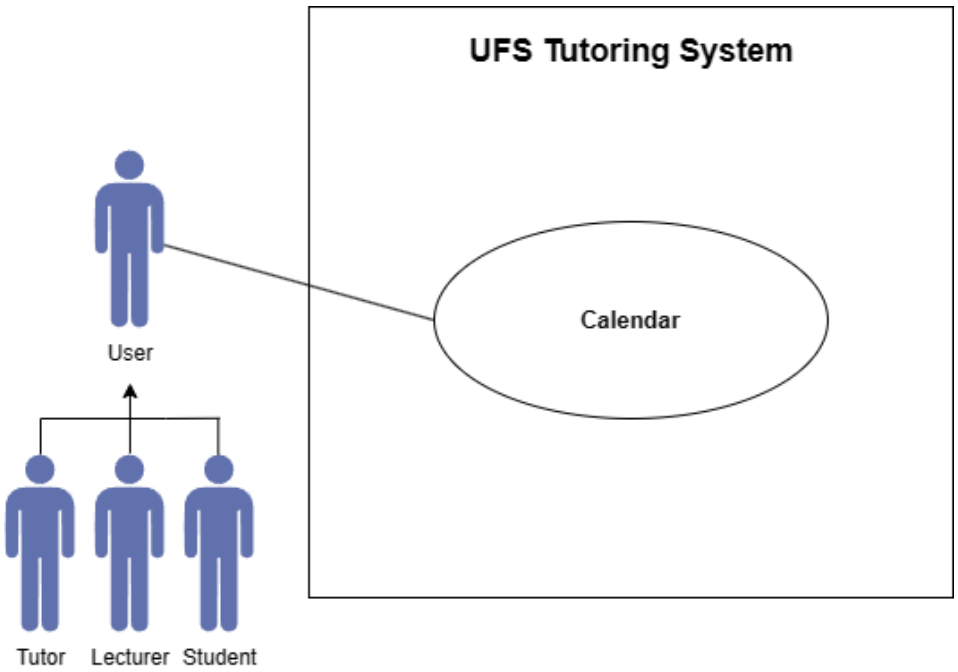


Figure 116: Calendar Use Case Diagram

Brief Description

The calendar feature allows a user to manage their schedule. It allows a user to view upcoming events either using a daily or monthly view. It also allows users to add, edit, and delete events. Users can drag and drop events to schedule the event for a different time. Additionally, lectures and tutors can schedule events that will be visible on students' calendars.

Step-by-Step Description

For step 1 kindly refer to Figure 1-3 Dashboard.

1. User clicks on the "Calendar" button on the side-menu.

For step 3 kindly refer to Figure 121 Monthly Calander View.

2. System displays a monthly calendar.

For step 3 kindly refer to Figure 117 Add Event Popup.

3. Add Event:
   i. User clicks on a specific date on the calendar.
   ii. System displays a popup form with adjustable parameters titled "Add Event". The parameters are title, module code, start date time and end date time. Additionally, if the user is a tutor/lecturer they will have the extra parameter to choose whether to make the event visible to students.

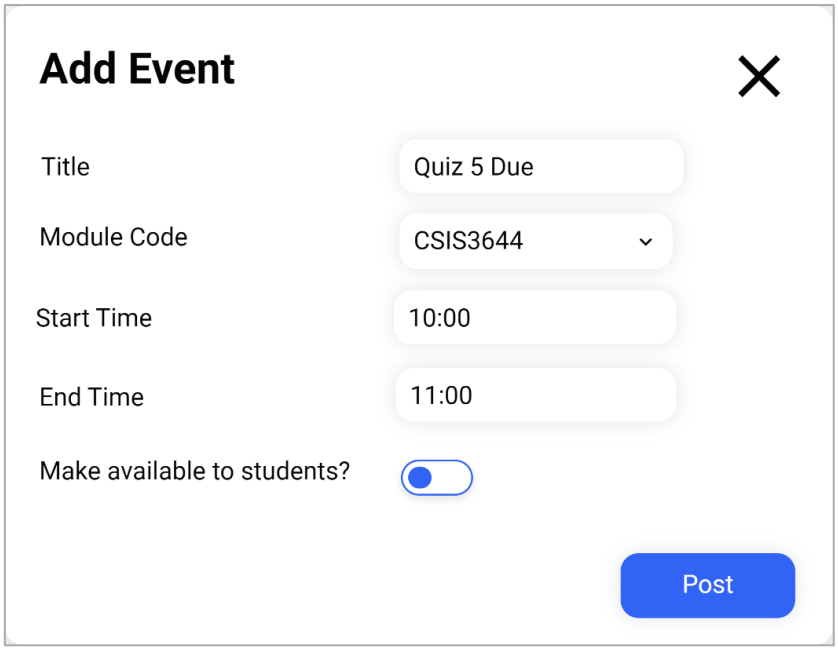


Figure 117: Add Event Popup

For step 4 kindly refer to Figure118 Edit Event Popup.

4. Edit Event:
   i. User clicks on an event.
   ii. System displays a populated popup form with adjustable parameters titled "Edit Event". The parameters are title, module code, start date time and end date time. Additionally, if the user is a tutor/lecturer they will have the extra parameter to choose whether to make the event visible to students.

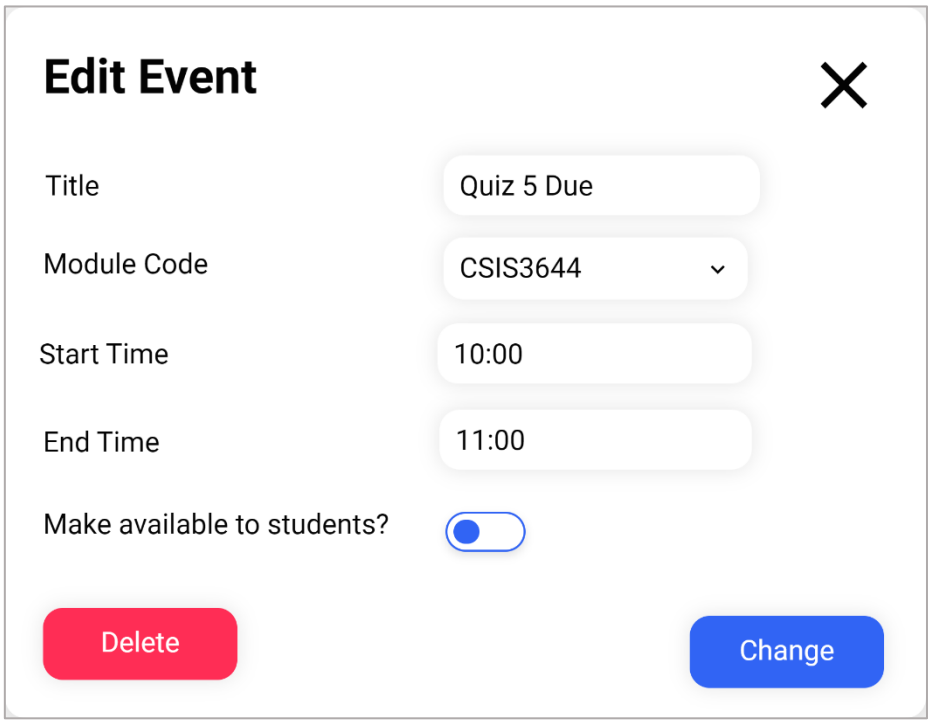


Figure 118: Edit Event Popup

For step 5 kindly refer to Figure119 Delete Event Popup.

5. Delete Event:
   i. User clicks on an event.
   ii. User clicks on the “Delete” button.
   iii. System creates a delete popup message stating: “You are about to delete an event are you sure you want to complete this action?”.
   iv. User clicks the “Delete” button.

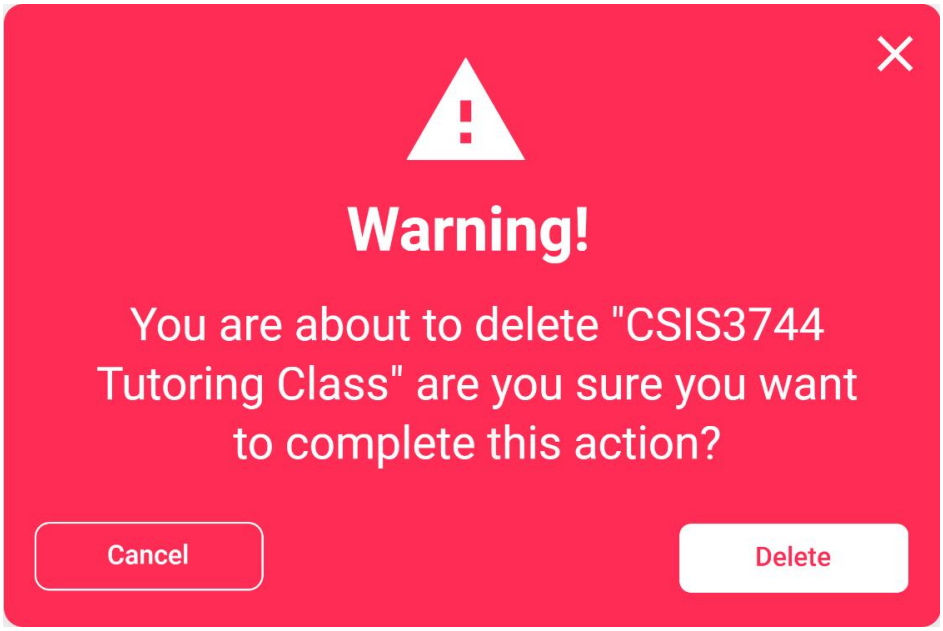


Figure 119: Delete Event Popup

6. Move Event:
   i. User clicks and holds on an event.
   ii. User moves their cursor holding the event to another location on the calendar.
   iii. User releases the event.
   iv. System changes the start time and end time to match the times in the location it was dropped.

Best-case Scenario

1. Kamo (a lecturer) clicks on the Calendar button on the dashboard.
2. System displays a by month view of the calendar.
3. Kamo clicks on a specific date, September 15, on the calendar.
4. Kamo clicks on the calendar again.
5. System shows Kamo the event form.
6. Kamo enters "Midterm Exam Review Session" as the title, 2:00 PM as the start time, 4:00 PM as the end time and selects “CS101” as the module code. Kamo checks the “Make available to students?” is checked.
7. Kamo clicks the post button.
8. System adds the event to Kamo's calendar, while also adding it to the calendar of all students registered for “CSIS3724” and the calendar of the tutor for “CSIS3724”.

Worst-case Scenario

1. Kamo clicks on the “Calendar” button on the dashboard.
2. System displays their calendar with a by month view.
3. Kamo clicks and holds on an event on the calendar. Kamo then drags the event outside of the calendar.
4. System displays an error message that the user cannot drag and drop an event to the outside of the calendar.

Alternative Scenario A

1. Kamo clicks on the “Calendar” button on the dashboard.
2. System displays a by month view of the calendar.
3. Kamo notices that he has a previously scheduled lecture from 10:00 AM to 11:00 AM on September 15.
4. Kamo schedules another event titled "Guest Lecture on AI" from 10:30 AM to 11:30 AM on the same date.
5. System allows Kamo to schedule the overlapping event.
6. System displays the calendar with the updated events.

Alternative Scenario B

1. Kamo clicks on the “Calendar” button on the dashboard.
2. System displays the by month view of the calendar. The month that is shown to Kamo is October 2023
3. Kamo wants to move the “Lecture on World Peace event” on the 14 October 2023 to 10 November 2023. Kamo clicks and holds the “Lecture on World Peace event” and moves it, he then hovers the event on the arrow of the navigation breadcrumbs.
4. System displays the November month of the calendar.
5. Kamo drops the “Lecture on World peace event” on 10 November 2023.
6. System removes the event from the 13 October 2023 and adds it to 10 November 2023.

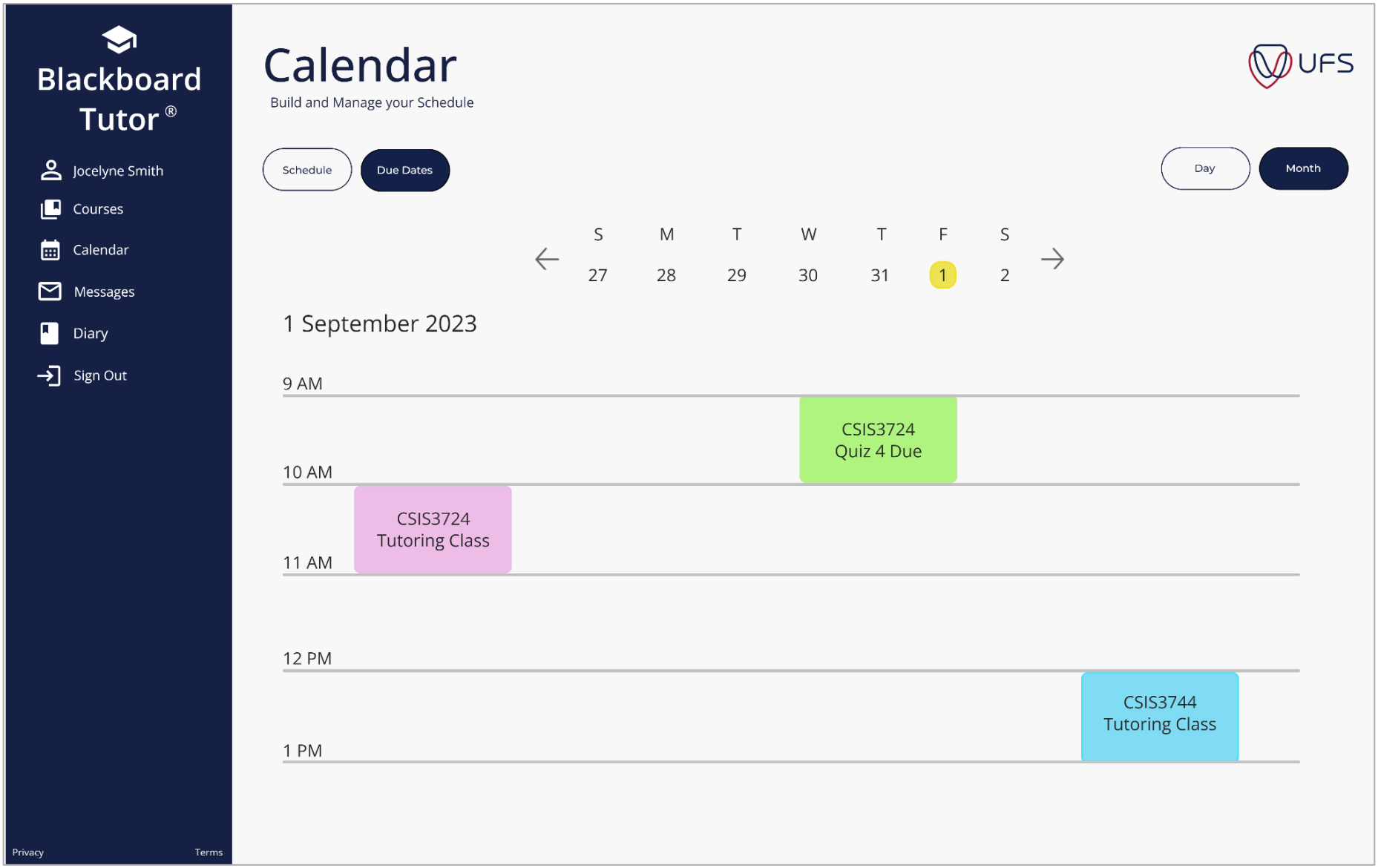


Figure 120: Weekly Calander View

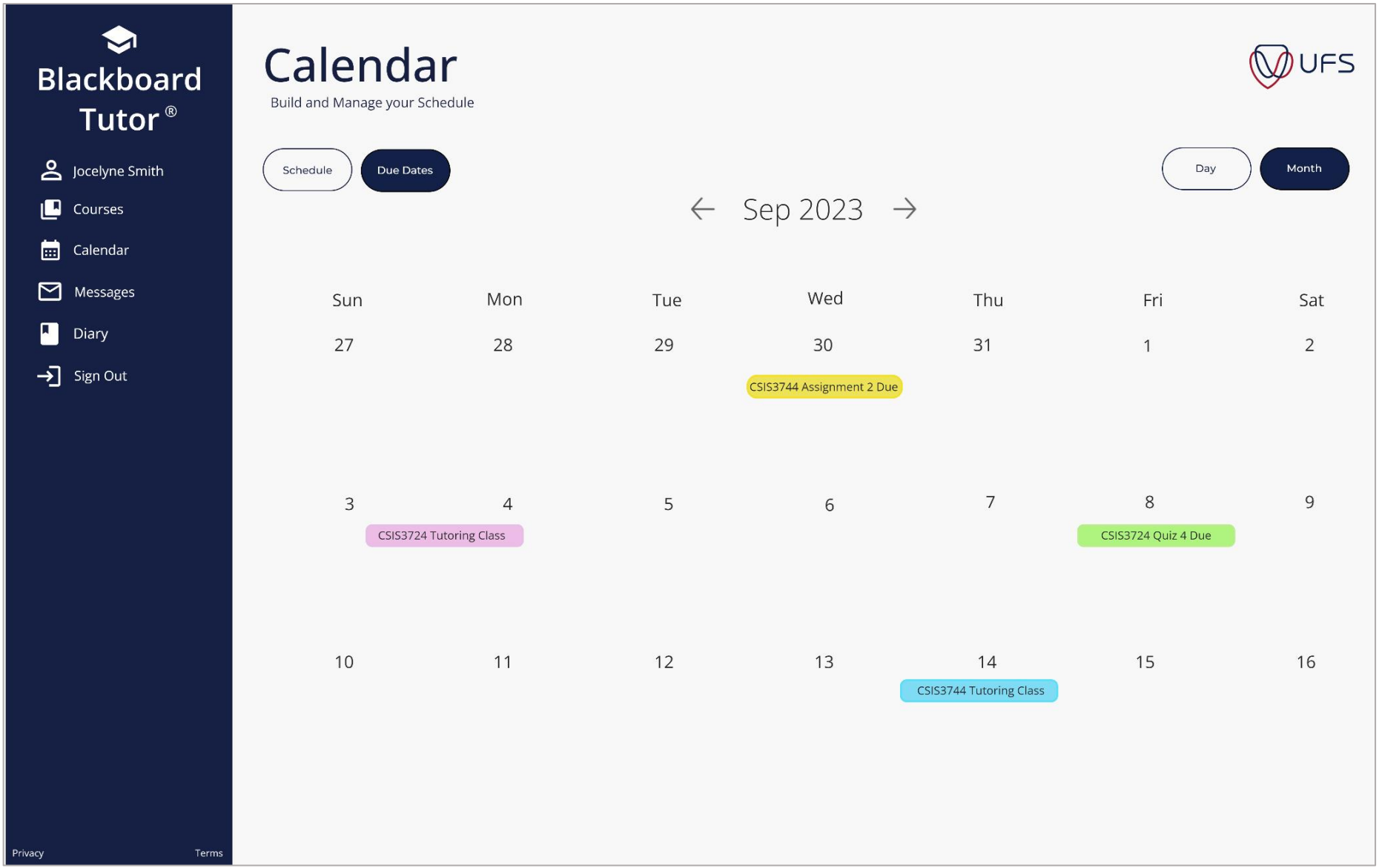


Figure 121: Monthly Calander View

Conclusion

This chapter provided great details for any use-cases including user interfaces, step-by-Step, best-case scenarios, worst case scenarios and many other scenarios. These will aid developers by functioning as test cases that will ensure the development of software is of good quality.

Blackboard Tutor®

Chapter 3: Software Management Plan

Project Manager,
BC Smit, 2020053969

Lead Developer,
K Kolanchu, 2020447641

Developer, SQA Manager
C Venter, 2019659602

UX Designer, Developer
J Smith, 2020046385

Introduction

The chapter will provide detail on the Software Management Plan. The Software Management Plan serves as a guide and will assist stakeholders throughout the software development lifecycle. It allows for efficient communication collaboration and risk management.

1. Overview

1.1. Project Summary

1.1.1. Purpose, Scope, and Objectives

Purpose

The purpose of the system is to facilitate peer-to-peer tutoring to improve the retention rate of first-year students in the Department of Computer Science and Informatics.

Objectives

The main objectives of the system are:

- Online code compiling service,
- Virtual classroom,
- Gamification.

Scope

The project scope as identified by the analysis team includes the following:

- User Authentication,
- User-Friendly System,
- Dashboard,
- Module-Specific Sections,
- Schedule and Planning,
- User Feedback,
- Misconduct Reporting,
- Data Analysis.

The details of the above objectives and scope can be found under the "Project Proposal" section of this document.

1.1.2. Assumptions and constraints

Assumptions

The assumptions identified by the analysis team includes the following:

- **Availability of Peer Tutors:** An assumption is made that 2nd and 3rd years students will be motivated by the monetary compensation to become tutors. We assume that from those interested a portion of students will be appointed to becoming tutors. We assume that before these tutors are appointed, they will receive adequate training from the University of the Free state Department of Teaching and Learning as per the University of the Free state guidelines for other tutors.
- **User Engagement:** An assumption is made that first-year students in the department of Computer science and Informatics will acknowledge that they are struggling with their studies and seek help from the “Peer-to-Peer tutoring system”.
- **Technology Competence:** An assumption is made that lecturers, tutors and students are technologically competence to use the system or otherwise will receive training.
- **System Maintenance:** An assumption is made that the University of the Free state’s Information communication technology team will provide the necessary hardware for the system. We assume that their personnel will have the necessary skills to deal with the configuration of the necessary hardware, software and will be able to overall maintain the system.
- **Acceptance Testing:** An assumption is made that the system will undergo user acceptance testing. We want to ensure that the product meets the needs and expectations of the stakeholders.
- **Feedback Loop:** An assumption is made that there will be a mechanism for collecting feedback from users that will aid in improving the system after delivery.
- **Scalability:** An assumption is made that the system will be scalable if demand for peer tutoring increases. This can include hardware and software scalability.
- **Data Privacy:** An assumption is made the system will treat user data with strict confidentiality and user data is not shared/sold to any thirds parties**.**

Constraints

The constraints identified by the analysis team include the following:

- **Budget:** The University of the Free state has set a limited budget for development, testing, and maintenance of the peer-to-peer tutoring system.
- **Deadline:** The University of the Free state has set a strict deadline for the completion of the system. We must establish a strict timeline with a fixed release date that coincides with the start of the new academic term.
- **Technology Stack:** A constraint identified is that the University of the Free state have limitations imposed to the specific technologies and the development team can use.
- **Regulatory Compliance:** A constraint identified is that the system must adhere to the Protection of Personal Information Act.
- **Developer Availability:** A constraint identified is that there is a limited availability of skilled developers which will impact the speed of development.
- **Tutor availability:** A constraint identified is that there might not be enough skilled tutors available. This could severely impact the quality of tutoring.
- **Integration:** The University of the Free state has not allowed us to integrate the tutoring system with existing educational platforms. The only integration that has been allowed is using PeopleSoft credentials to log into the tutoring system.
- **User Adoption:** A constraint identified to the adoption rate among students and tutors, which might be influenced by factors such as usability and training.
- **Security:** A constraint identified is that the University of the Free state's Information Communication Technology team must ensure the security of the user data. Furthermore, the system must be protected against potential breaches or cyberattacks.
- **Scalability:** A constraint identified could be scalability. The system could limit the number of users it can support simultaneously.
- **Accessibility**: A constraint identified could be that the system must adhere to University of the Free state's Centre for Universal Access and Disability Support's vision. The system should be accessible to users with disabilities. This means the system must adhere to accessibility standards like WCAG (Web Content Accessibility Guidelines).

1.1.3. Project Deliverables

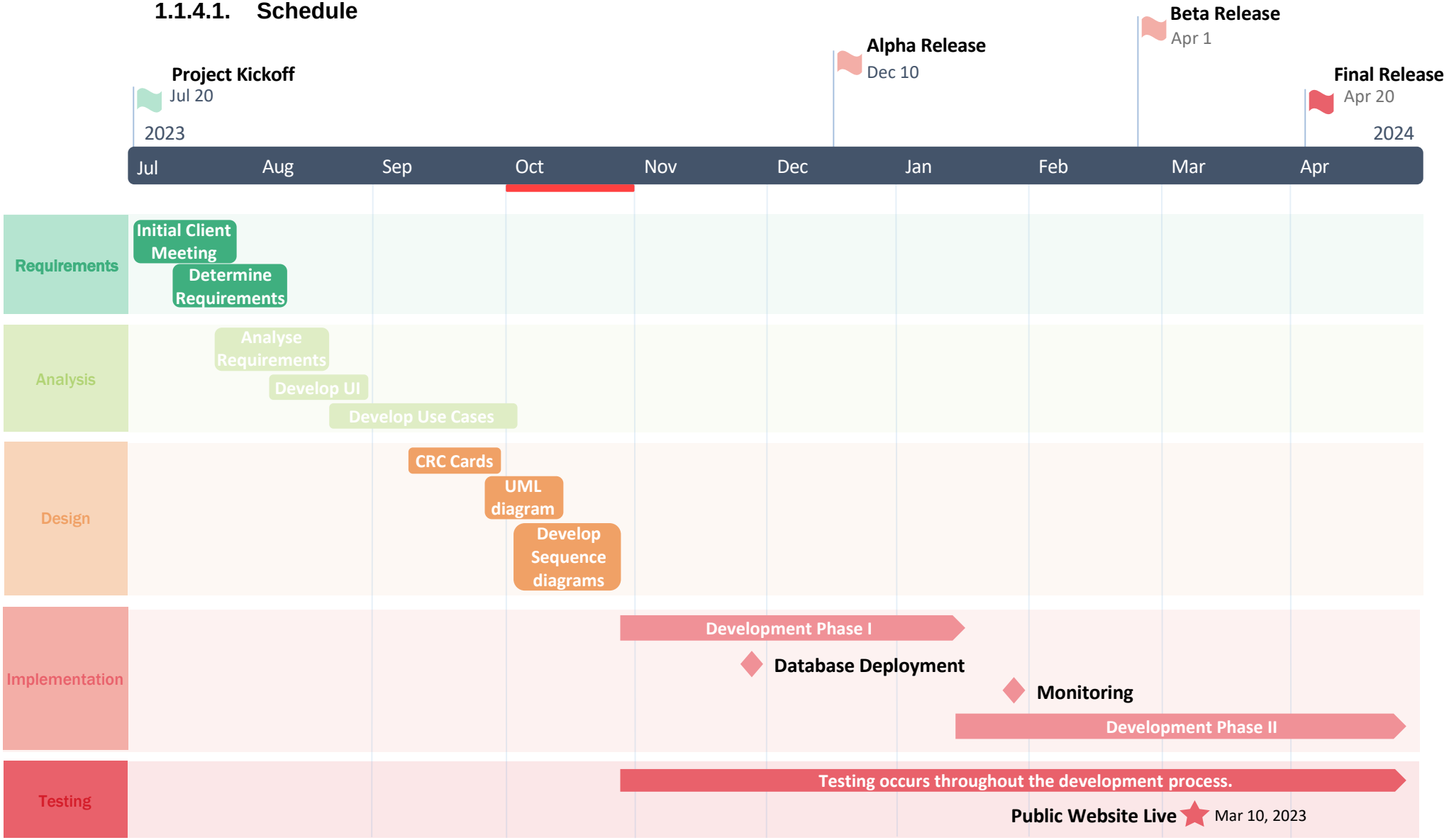
The project must have the following deliverables:

- **Project Plan:** A comprehensive project plan including: the project's scope, schedule, and budget. (100 hours after the project commences).
- **Software Product:** The “Peer-to-peer tutoring” system.
- **User Manual:** A document containing instructions for students, tutors, and lecturers on how to use the tutoring system effectively.
- **Maintenance and Support Plan:** A document stipulating how ongoing maintenance, bug fixes, and user support will be handled after the system is deployed.

The software project, user manual, maintenance and support plans will be delivered to the client 1,019 hours (±9 months) after the project commences.

1.1.4. Schedule and Budget Summary

1.1.4.1. Schedule



*Kindly note that this diagram follows our development process and thus most of the tasks are not in their traditional order.

1.1.4.2. Budget

The estimate for the duration of the project is: 6240 hours (±9 months). The hours are divided into the workflows as follows:

- Requirements: 80 hours
- Analysis: 160 hours
- Design: 400 hours
- Implementation: 2400 hours
- Testing: 3200 hours

The budget for the system involves various expenses, including:

- personnel salary,
- software and hardware costs,
- general expenses.

1. **Personnel Salary:**

Project Manager

BC Smit

Requirements	R240 per person/per hour × 20
Analysis	R240 per person/per hour × 45
Design	R240 per person/per hour × 95
Implementation	R240 per person/per hour × 600
Testing	R240 per person/per hour × 800
Total	R374,400

Lead Developer

K Kolanchu

Requirements	R240 per person/per hour × 18
Analysis	R240 per person/per hour × 47
Design	R240 per person/per hour × 95
Implementation	R240 per person/per hour × 600
Testing	R240 per person/per hour × 800
Total	R374,400

Developer, Software Quality Assurance Manager

C Venter

Requirements	R240 per person/per hour × 23
Analysis	R240 per person/per hour × 44
Design	R240 per person/per hour × 93
Implementation	R240 per person/per hour × 600
Testing	R240 per person/per hour × 800
Total	R374,400

UX Designer, Developer

J Smith

Requirements	R240 per person/per hour × 17
Analysis	R240 per person/per hour × 40
Design	R240 per person/per hour × 103
Implementation	R240 per person/per hour × 600
Testing	R240 per person/per hour × 800
Total	R374,400

Total Personnel Salary:

BC Smit	R374,400
K Kolanchu	R374,400
C Venter	R374,400
J Smith	R374,400
Total	**R1,497,600**

2. Software and Hardware Costs:

Visual Studio (Commercial Licenses)	R 2,000
SQL Server Management Studio (Commercial Licenses)	R 2,000
Cloud services (AWS) for deployment	R 4,000
Total	**R 8,000**

3. Miscellaneous Expenses:

Total	R1,505,600
General Expenses	5%
Total	**R75,280**

4. Total Cost:

Total	R1,505,600
General Expenses	R75,280
Total	**R1,580,880**

See “5.2.4 Budget Allocation” for the budget breakdown by workflow.

1.2. Evolution of the project management plan

Any change in the software project management plan must first be approved by the project manager, BC Smit before it is implemented.

Furthermore, the change must be documented by J Smith to keep the software project management plan correct, complete, and up to date.

2. References

Schach, S.R., 2011. Object-Oriented and Classical Software Engineering. 8th ed. New York: McGraw-Hill

3. Definitions and acronyms

Client – The University of the Free State

System – The Peer-to-Peer Tutoring System

4. Project Organization

4.1. External Interfaces

All work related to the system software project will be performed by:

- BC Smit (Project Manager, Developer),
- K Kolanchu (Lead Developer),
- C Venter (Developer, Software Quality Assurance Manager),
- J Smith (UX Designer, Developer),

The details of what each member will contribute are discussed below.

The Project manager, BC Smit will meet weekly with the client to report progress and discuss possible changes and modifications.

4.2. Internal Structure

The development team consists of:

- BC Smit (Project Manager, Developer),
- K Kolanchu (Lead Developer),
- C Venter (Developer, Software Quality Assurance Manager),
- J Smith (UX Designer, Developer),

Additionally, the development team will be working closely with a University of the Free state Department of Computer Science and Informatics representative, Rouxan Fouché.

4.3. Roles and Responsibilities

All members of the team will perform the design, implementation, and testing workflow.

Furthermore, the following applies:

- K Kolanchu will be responsible for all artefacts relating to the:
  - calendar and
  - leaderboard.

- BC Smit will be responsible for all artefacts relating to the
  - quiz,
  - session,
  - diary, and
  - chat.

  He will be responsible for the virtual classroom functionality, building the chat

platform (including the function that allows lecturers to communicate with students who have made reports against tutors).

- C Venter will be responsible for all artefacts relating to:
    - managing tutors,
    - managing reports,
    - allowing a user to view their profile,
    - grades,
    - Vault,
    - and statistics.

- J Smith will be responsible for the artifacts that relate to:
    - authenticating users,
    - login and
    - logout.

  She will also be responsible for integration with the AWS database.

Each team member is responsible for the quality of the software components he or she produces. However, C Venter Software will act as the Quality Assurance Manager and will be responsible for making sure that code is of good quality and fault free. Integration testing will be performed collectively by all team members. BC Smit will report progress to the client representative.

5. Managerial Process Plans

5.1. Start-up Plan

5.1.1. Estimation Plan

Using bottom-up Approach the analysis team divided the project. Each artifact was estimated, and all the estimates were combined for an overall figure. The analysis team estimated that the total development time will be approximately 9 months and will cost roughly R1,580,880.

5.1.2. Staffing Plan

All members of the team

- BC Smit (Project Manager, Developer),
- K Kolanchu (Lead Developer),
- C Venter (Developer, Software Quality Assurance Manager),
- J Smith (UX Designer, Developer),

will be needed for the entire 9-month duration of the project development.

BC Smit will act as the project manager for the entire 9 months. Additionally for the last 5 months of the project he will function as both manager and programmer.

All members will work as system analysts and J Smith will work as UX designer for the first 3 months of the project.

For the last 5 months all team members will work as programmers and testers.

5.1.3. Resource Acquisition Plan

The hardware, software and CASE tools that is required is already available.

The hardware includes:

- personal computers (Windows),
- mobile devices (android and IOS).

The client has provided us with Linux server that will host the system software. The server is located at the University of the Free state Information and Communication Technology Services building.

The software includes:

- Visual Studio 2022 and
- SQL server management studio 2019.

However, storage will be needed by the system and will the hosted-on AWS cloud storage.

Any additional hardware will be obtained from *Rectron (PTY) LTD*.

5.1.4. Project Staff Training Plan

No additional staff training is necessary.

5.2. Work Plan

5.2.1. Work Activities and Schedule Allocation

All team members will form part each workflow of the unified process.

Our approach to this project will encompass the following steps:

- **Requirements Workflow** (80 hours): The requirements team will collaborate closely with the client representative of UFS Department of Computer Science and Informatics to develop the requirements artefacts.

  Status: Completed

- **Analysis Workflow** (160 hours): The analysis team will review the requirements artefacts and create a comprehensive system design, including:
  - user interfaces,
  - specifications document,
  - and database structure.

  The above will be reviewed by the client and approved. Once approved, the software product management plan will be complied.

  Status: Complete

- **Design Workflow** (400 hours): Each member of the design team will produce his/her own design artifacts. The project manager BC Smit will then the team's design artifacts.

  Status: In progress

- **Implementation Workflow** (2400 hours): The development team will develop the system based on the approved design.

  Status: In progress

- **Testing Workplace:** (3200 hours): Testing will occur throughout the development of the system. C Venter Software will act as the Quality Assurance Manager and will ensure that quality and fault free code. Lastly, all members will perform integration testing.

  Status: In progress

5.2.2. Resource Allocation

Team members will work independently on their assigned artifacts. Daily meeting will occur where team members can consult one another and request assistance with their artefacts.

The project manager BC Smit will be responsible for overseeing project progress will make sure that each team members adheres to budget constraints, and deadlines. He will monitor the progress of the team members daily.

BC Smit will meet with the client representative at the end of each week and will communicate any changes that needs to be made to the team.

J Smith will be responsible for documenting any changes and ensuring that documentation is up to date.

C Venter will act as the Quality Assurance Manager and will be responsible for ensuring fault free and high-quality code. He is therefore in charge of risk management.

5.2.3. Budget Allocation

Requirements		R19,200
Analysis		R38,400
Design		R96,000
Implementation		R576,000
Testing		R768,000
	Sub Total	**R1,497,600**

5.3. Control plan

Any change that affects the deadlines or budget of the project must be approved by BC Smit and documented by J Smith. The changes will be communicated to the other members during the weekly meetings.

The project manager BC Smit will be responsible for ensuring that the project is completed on time, within budget and satisfies the client's needs.

Each team member will be responsible for ensuring that their assigned artefacts is of good quality. C venter will act as quality assurance manager and will review all the artifacts submitted by the group members.

During implementation, each member will develop their and test their own unit. Where necessary, members can also make use of pair programming. C Venter will perform unit testing. He may assign a different team member which will then test the unit using test cases.

BC Smit will determine whether the project is on track and weather the specification document and project management plans are adhered to. Any major problems faced by the team members must be reported in the weekly meeting to BC Smit.

5.4. Risk Management Plan

Opinto does not own a similar software product to the peer-to-peer tutoring system. Therefore, no other product can be used as a model for risk management. To mitigate potential risks, all development team members will this carry out extensive testing of their own artifact before it is published on the Git.

Similarly, all software components must undergo testing while being developed. The programmer will develop a component and thereafter another programmer will test the component. Thereafter the Quality Assurance Manager C Venter will perform unit testing to ensure that the component is of adequate quality.

The members of the development team will collectively perform integration and product testing.

The project manager, BC Smit will for monitor adherence to the specifications document and software product management plan.

5.5. Project Close-out Plan

Not applicable.

6. Technical Process Plans

6.1. Process Model

For the development of the “Student Peer-to-Peer Tutoring System” the development team will use the:

- Unified Process methodology and
- Iterative-and-Incremental Life-cycle model.

6.2. Methods, Tools, and Techniques

For the development of the “Student Peer-to-Peer Tutoring System” the development team will utilize the workflows of the Unified Process:

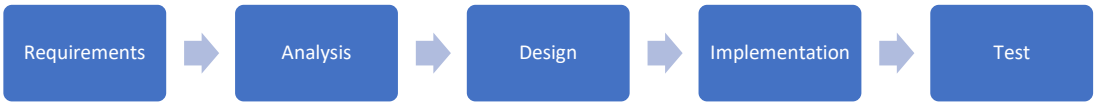


Figure 122: Unified Process

The coding languages that have been selected for development are:

- C# MVC Core,
- JavaScript.

The database and database management system that has been selected is Microsoft SQL Server.

The APIs that will be used are:

- Google Cloud Speech-to-Text API (for speech-to-text),
- Microsoft Teams API (for video conferencing).

Lastly, the ASP .NET library, SignalR, will be used to implement live chat.

6.3. Infrastructure Plan

Development will take place on personal computers. The computers will run on Windows. Developers will use Visual Studio 2022 and SQL server management studio 2019. The system software will be hosted on a Linux server located at the University of the Free state Information and Communication Technology Services building. The storage required by the system will the hosted-on AWS cloud storage.

6.4. Product Acceptance Plan

Client acceptance will be achieved by following the steps defined in the Unified Process.

7. Supporting Process Plans

7.1. Configuration Management Plan:

GitHub will be used for version control.

7.2. Testing Plan

All members will be responsible for testing their own code. C Venter will act as software quality assurance manager. The testing activities will be followed as defined in the testing workflow of the Unified Process.

7.3. Documentation Plan

J Smith will be responsible for producing the project documentation as defined in the Unified Process.

7.4. Quality Assurance Plan

J Smith, BC Smit, C Venter, and K Kolanchu will each test their assigned artifacts. C Venter will act as software quality assurance manager. He will use unit testing methods to ensure that the code produced is of good quality and fault free. Furthermore, C Venter will be responsible for Properly documenting the changes to ensure clarity and understanding by other team members.

Integration and product testing will be conducted by all development team members. The project manager, BC Smit will manage integration and product testing. BC Smit is responsible for ensuring that the requirements stated in the specifications document are met.

7.5. Reviews and Audits Plan

Not applicable.

7.6. Problem Resolution Plan

The team will meet daily to discuss any changes to artifacts. Problems will be brought up and dealt with during these meetings.

7.7. Subcontractor Management Plan

Not applicable.

7.8. Process Improvement plan

We acknowledge that there should be a way for users of the system to submit reports of bugs and suggestions for improvements to the system, however for the purposes of this assignment we will delimit the scope and this plan will not be included.

8. Additional Plans

8.1. Security

The following security measures will be implemented:

- **User authentication:** Students will login to the system using their PeopleSoft credentials.
- **Database authentication:** The database can only be accessed using system administrator credentials.
- **Transmission encryption:** Data transmitted between client devices and the server will be encrypted using protocols like HTTPS/SSL.
- **Database encryption:** Sensitive data stored in databases will be encrypted to prevent unauthorized access.
- **Input Validation:** The system will validate all input. This will prevent malicious input from being processed and causing security issues.
- **Session Management:** The system will make use of secure session management, including session timeout and secure storage of session tokens.
- **File Upload Security:** The system will validate uploaded files to prevent malicious uploads or uploads that are too big that can cause the system to crash.

8.2. Training

The UFS teaching and learning department representatives will receive training on using the peer-to-peer tutoring system. J Smith will be responsible for providing the training using the user manuals. Since the product is straightforward 2 working days will be sufficient for training. The UFS teaching and learning department representatives will then provide training to the lecturers and tutors.

8.3. Maintenance

Post-delivery maintenance will be provided by Opinto at no additional cost to the University of the Free state for a period of three months.

Conclusion

During this chapter, we discussed the elements of the Software Management Plan. We had a look at the Project Summary, Project Organization, Managerial Process Plans, Technical Process Plans, Supporting Process Plans and Additional Plans. These plans outline the strategies, processes, and guidelines needed for efficient project management. The Software Management Plan ultimately provides the structure and direction needed to achieve the project goals, mitigate risk, and ensure the project's success.

Blackboard Tutor®

Chapter 3: Quotation

Project Manager,
BC Smit, 2020053969

Lead Developer,
K Kolanchu, 2020447641

Developer, SQA Manager
C Venter, 2019659602

UX Designer, Developer
J Smith, 2020046385

Peer-to-Peer Tutoring System Quote

Opinto

Where ideas become solutions

Date :	5 October 2023
Quote No :	2465
Expiration Date :	16 October 2024

Address :

4 Reier Street,

Pellissier

Bloemfontein, SA 9301

Tel: +27 79 646 1642

Recipient :

University of the Free state,

205 Nelson Mandela Drive

Universitas, Bloemfontein, SA 9501

Tel: +27 83 450 3639

QUANTITY	DESCRIPTION	UNIT PRICE	AMOUNT
1	Requirements		R19,200
1	Analysis		R38,400
1	Design		R96,000
1	Implementation		R576,000
1	Testing		R768,000
		Sub Total	**R1,497,600**
1	Support Software		R8,000
1	General Expenses		R75,280
		Sub Total	**R1,580,880**
		Vat 15%	**R237 132**
		Total Amount	**R1,818,012**

This Quotation is prepared by Opinto.

Quotation accepted by University of the Free state.

If you have any enquiries about this, please contact Barend Smit on Tel : +27 65 455 1632

Thank You for Your Business

Blackboard Tutor®

Chapter 4: Design Document

Project Manager, BC Smit, 2020053969

Lead Developer, K Kolanchu, 2020447641

Developer, SQA Manager C Venter, 2019659602

UX Designer, Developer J Smith, 2020046385

Introduction

The Design Document outlines the essential components of the Peer-to-Peer Tutoring System. The Design Document includes Sequence Diagrams, State Charts, Noun Extraction, CRC Cards, UML Diagrams, and Pseudo Code. The Design Document provides the necessary design details to the developers ensuring the development of high quality and fault free code. It also provides a structured plan and clear vision for stakeholders and team members.

Sequence Diagrams

Sequence diagrams were drawn up for every use case. They are listed below:

For the readers sake, the sequence diagrams are provided in the same order as the use-cases.

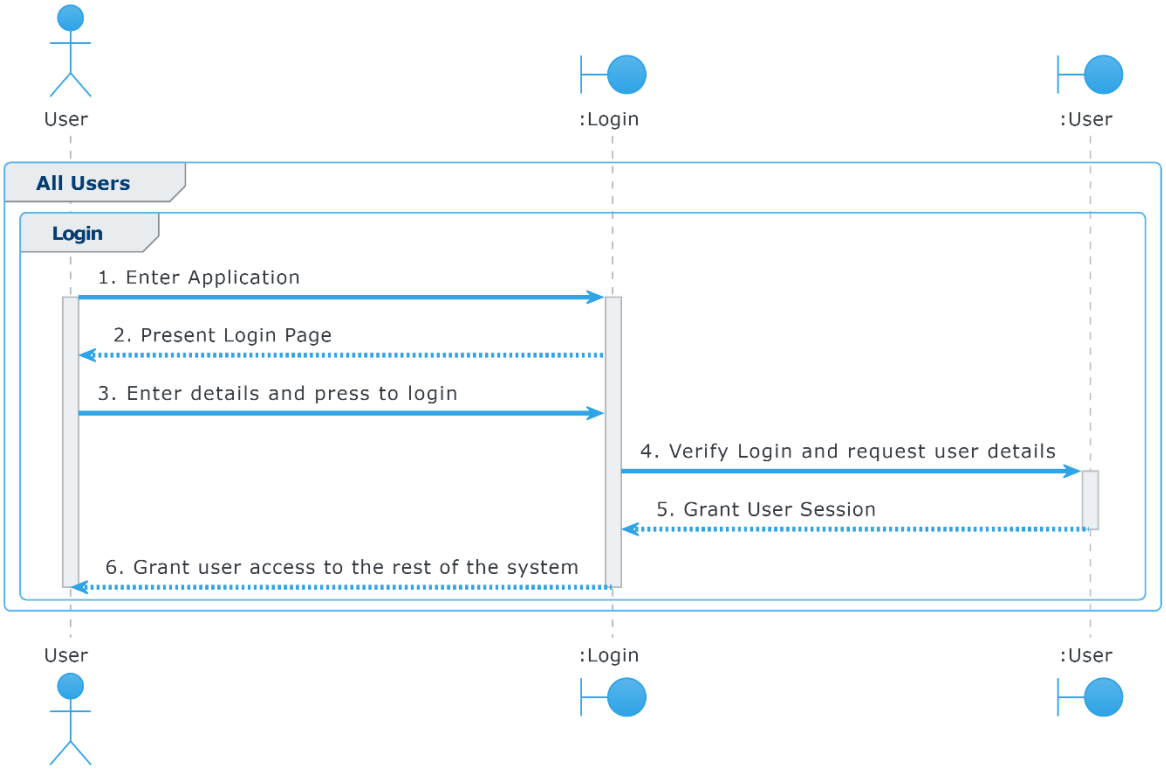


Figure 123: Login Sequence Diagram

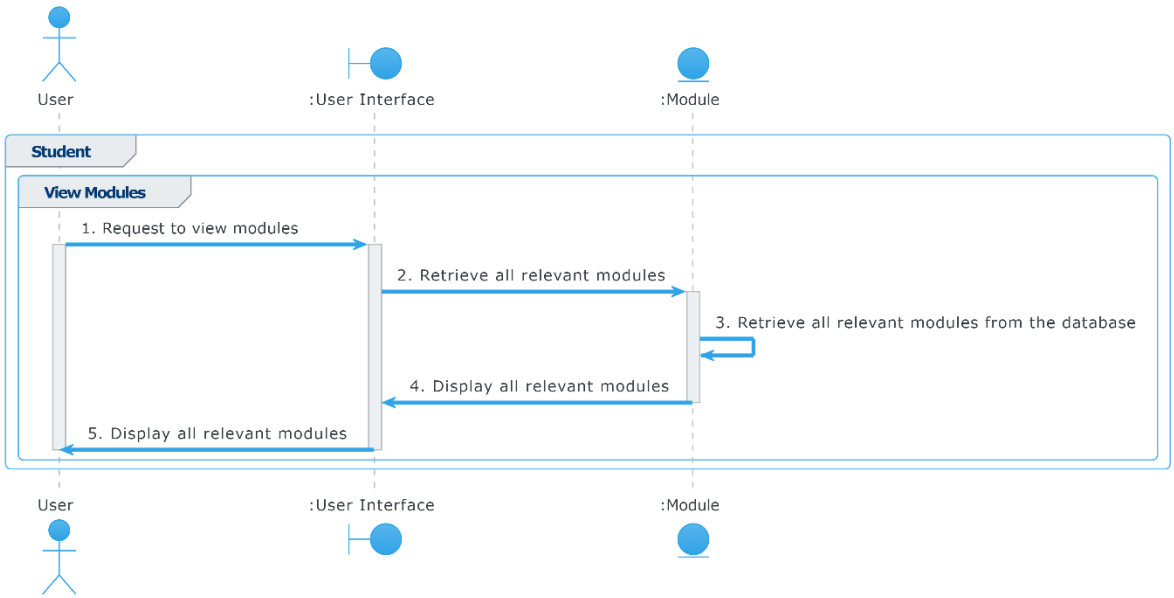


Figure 124: View Modules Sequence Diagram

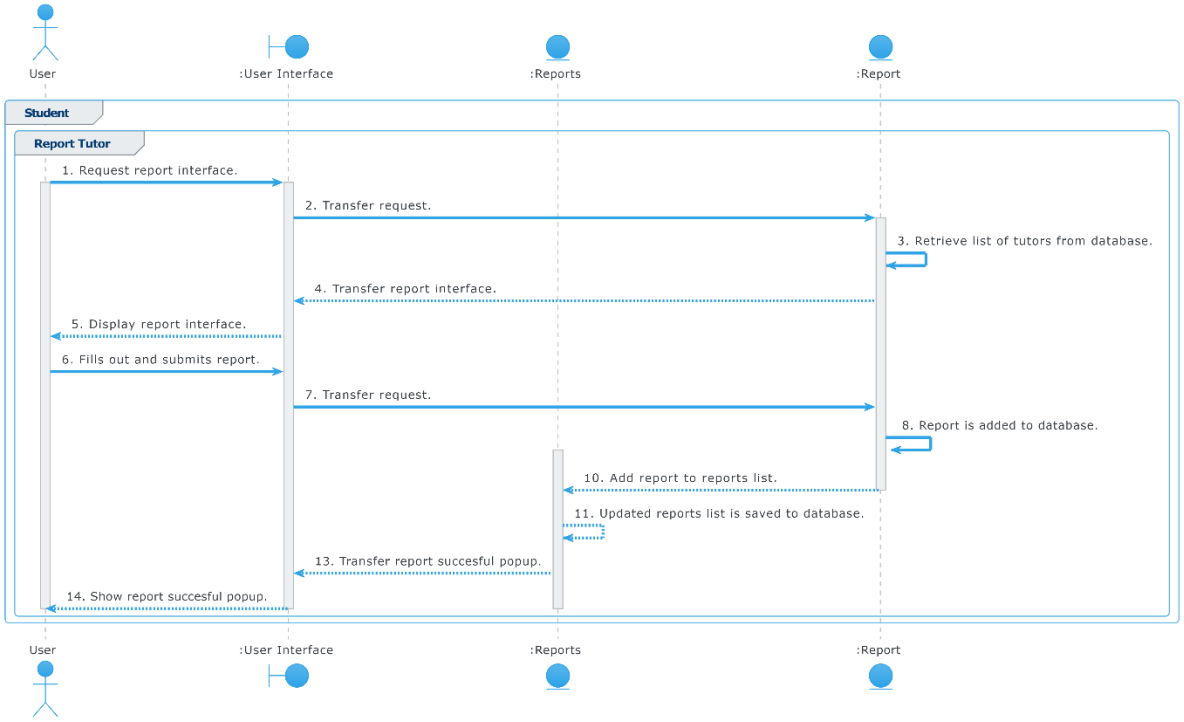


Figure 125: Report Tutor Sequence Diagram

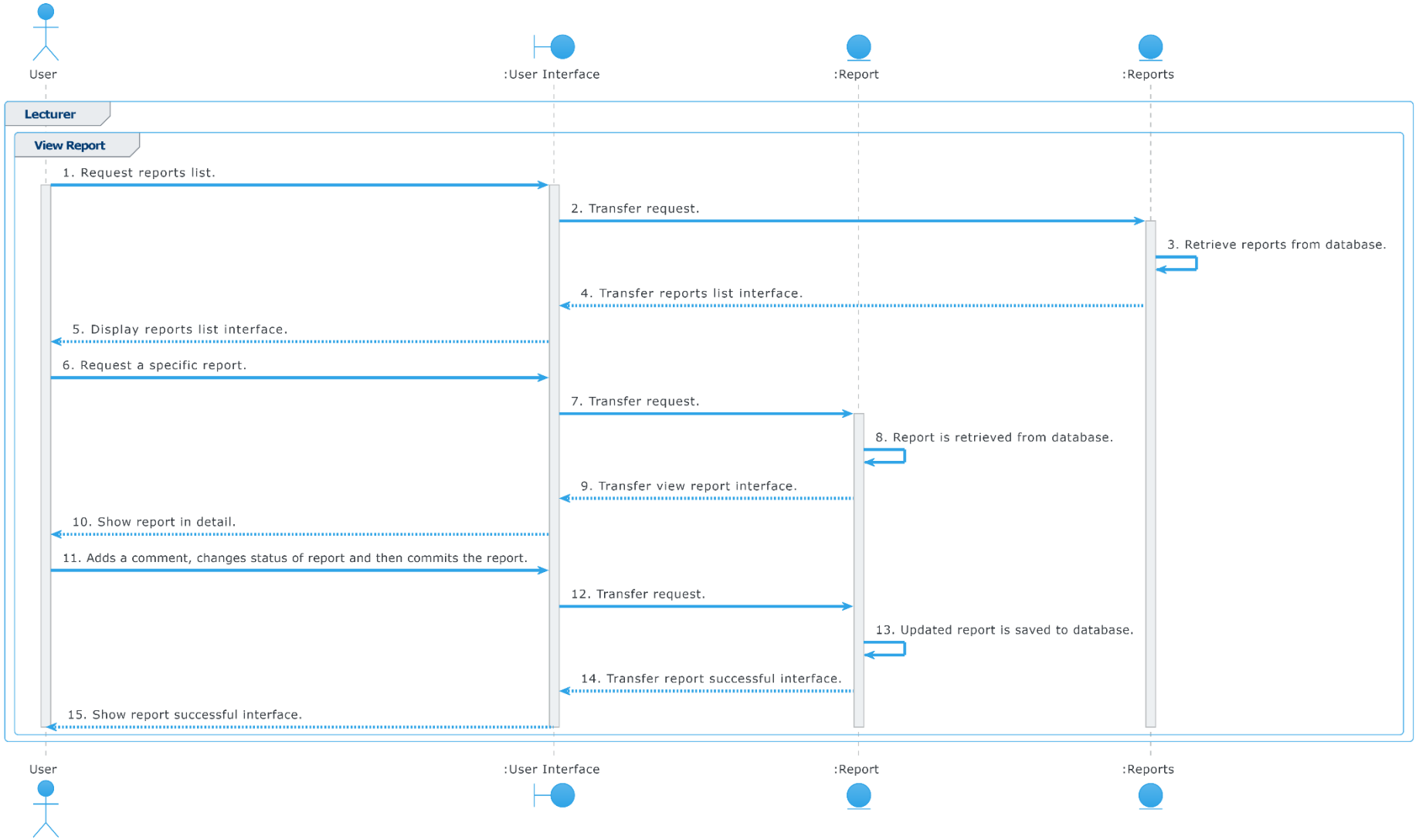


Figure 126: View Report Sequence Diagram

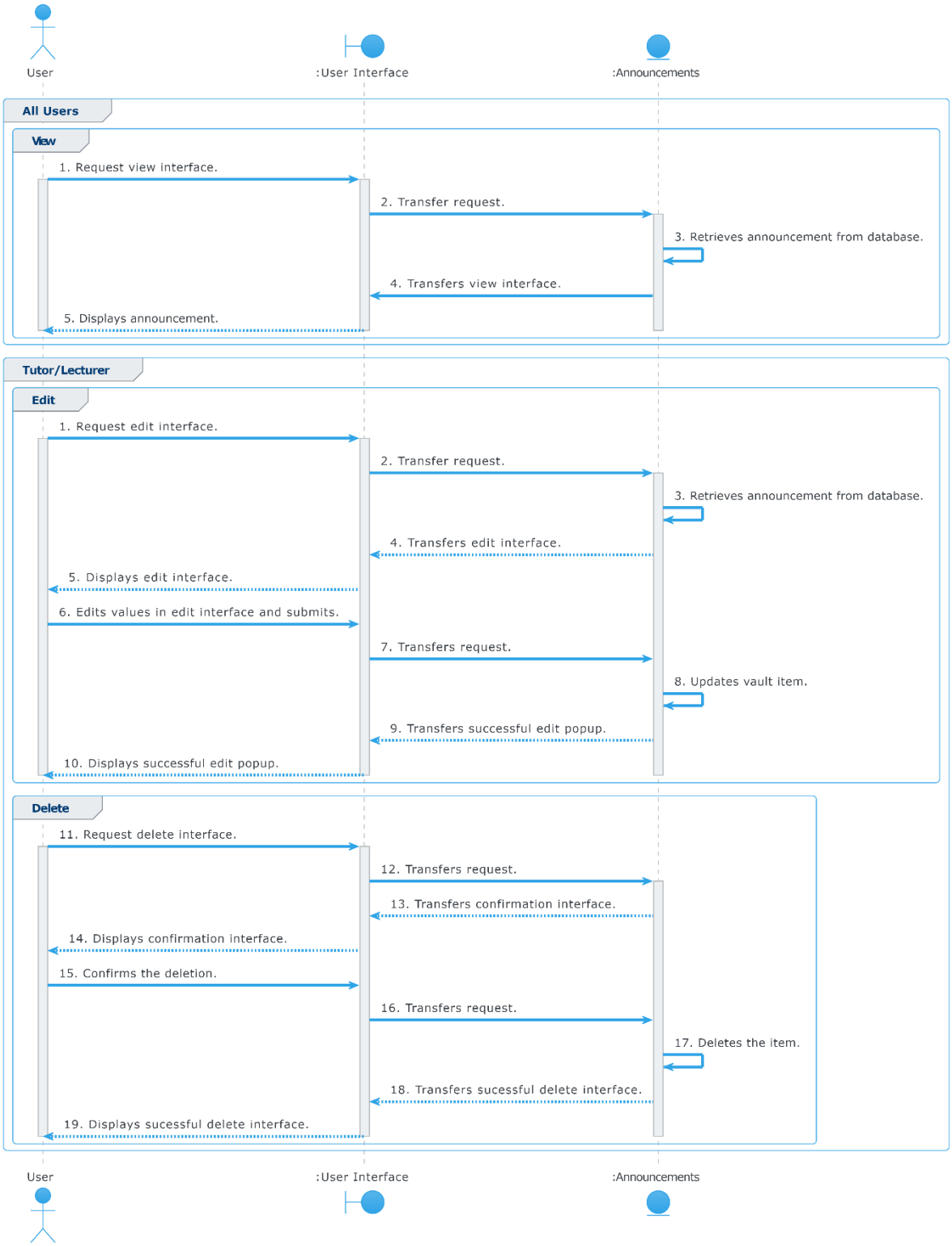


Figure 127: Announcement Sequence Diagram

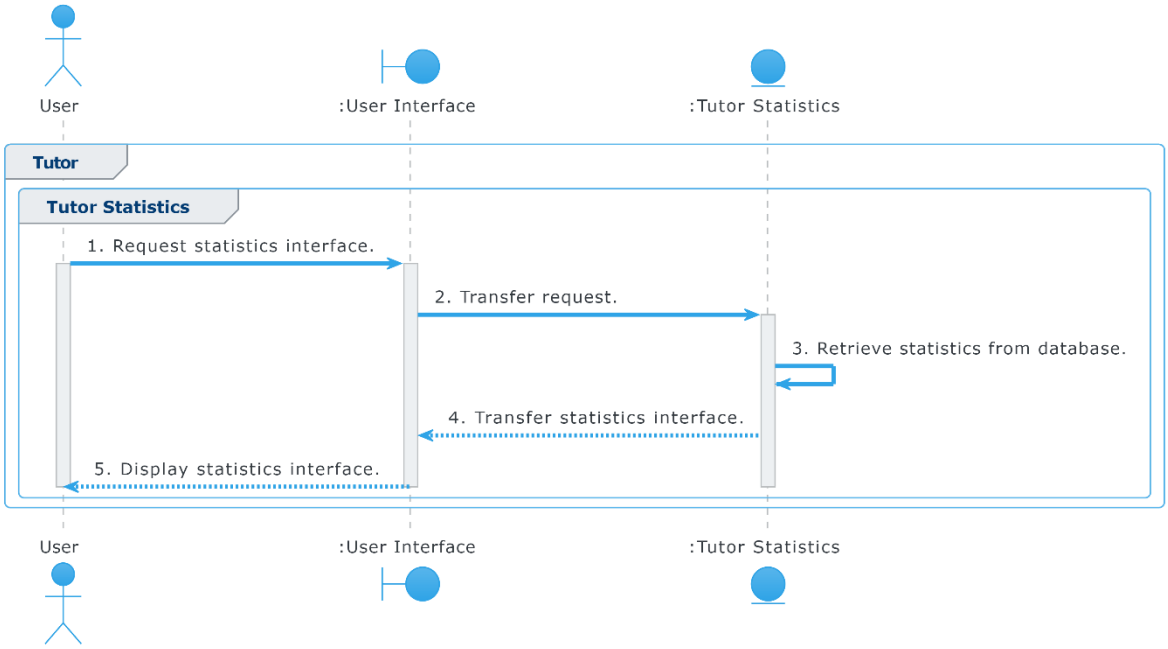


Figure 128: Tutor Statistics Sequence Diagram

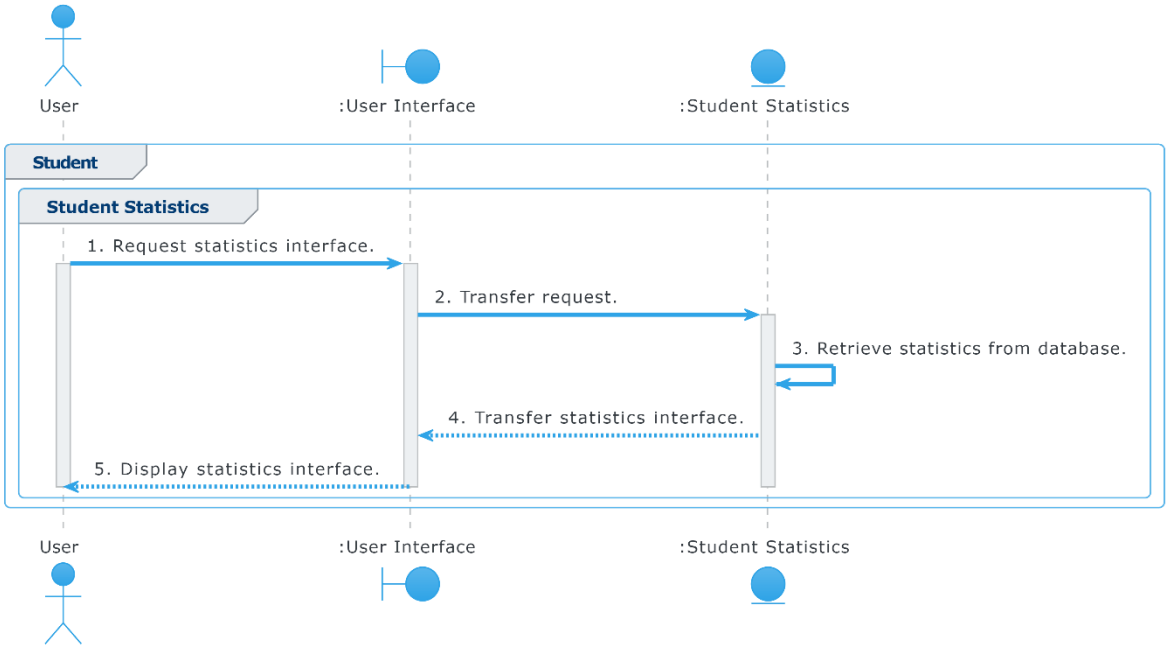


Figure 129: Student Statistics Sequence Diagram

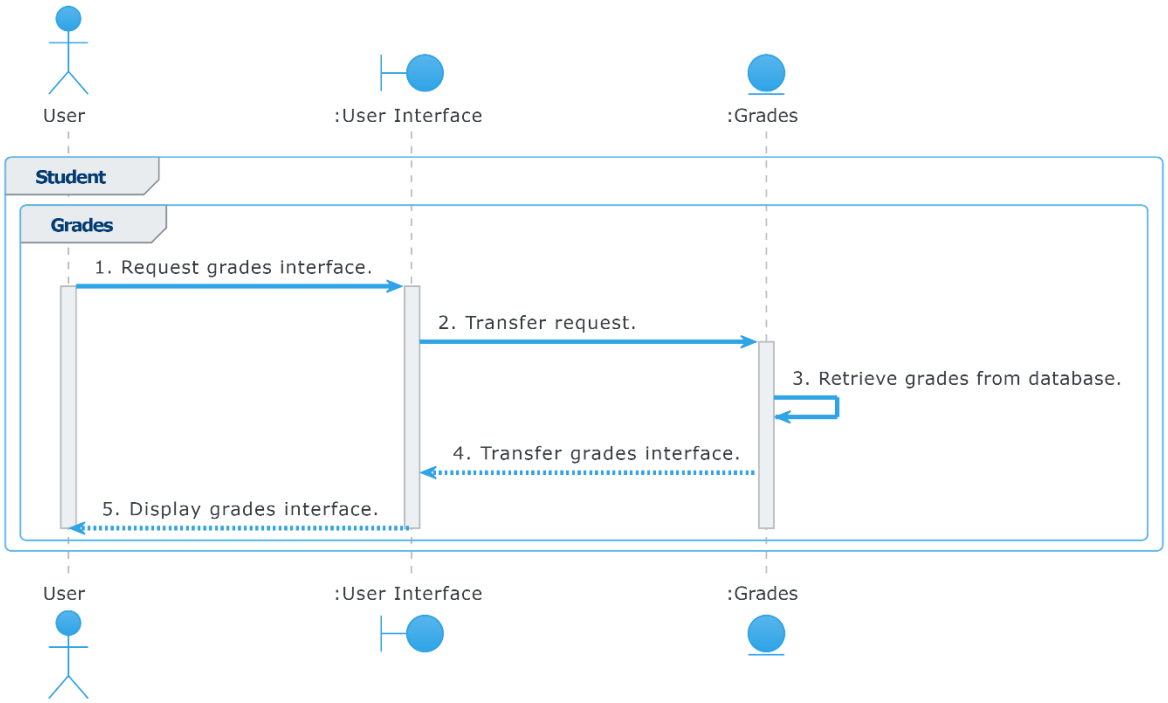


Figure 130: Grades Sequence Diagram

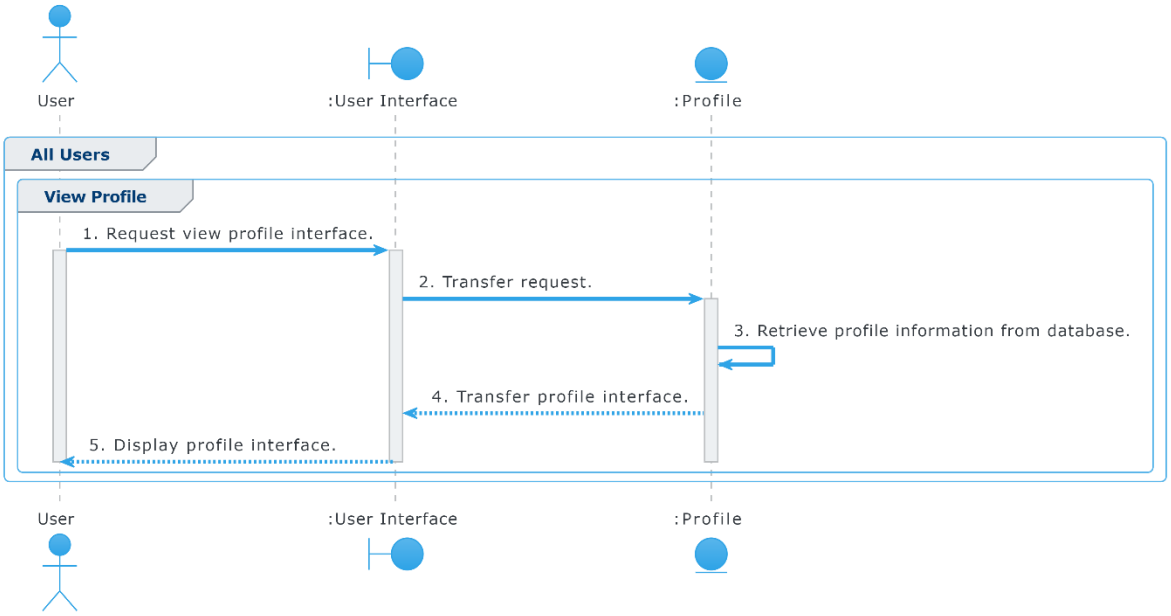


Figure 131: View Profile Sequence Diagram

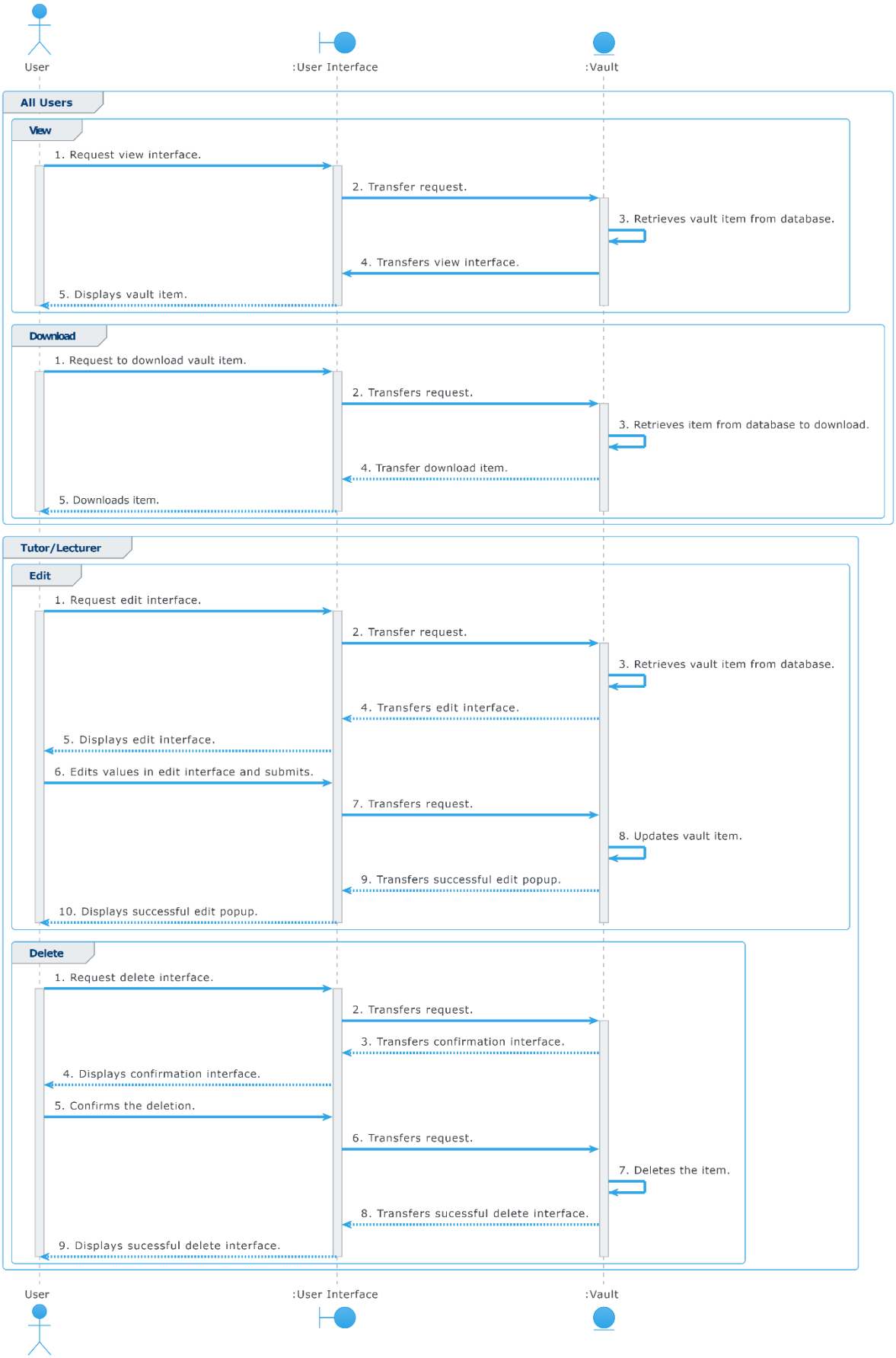


Figure 132: Vault Sequence Diagram

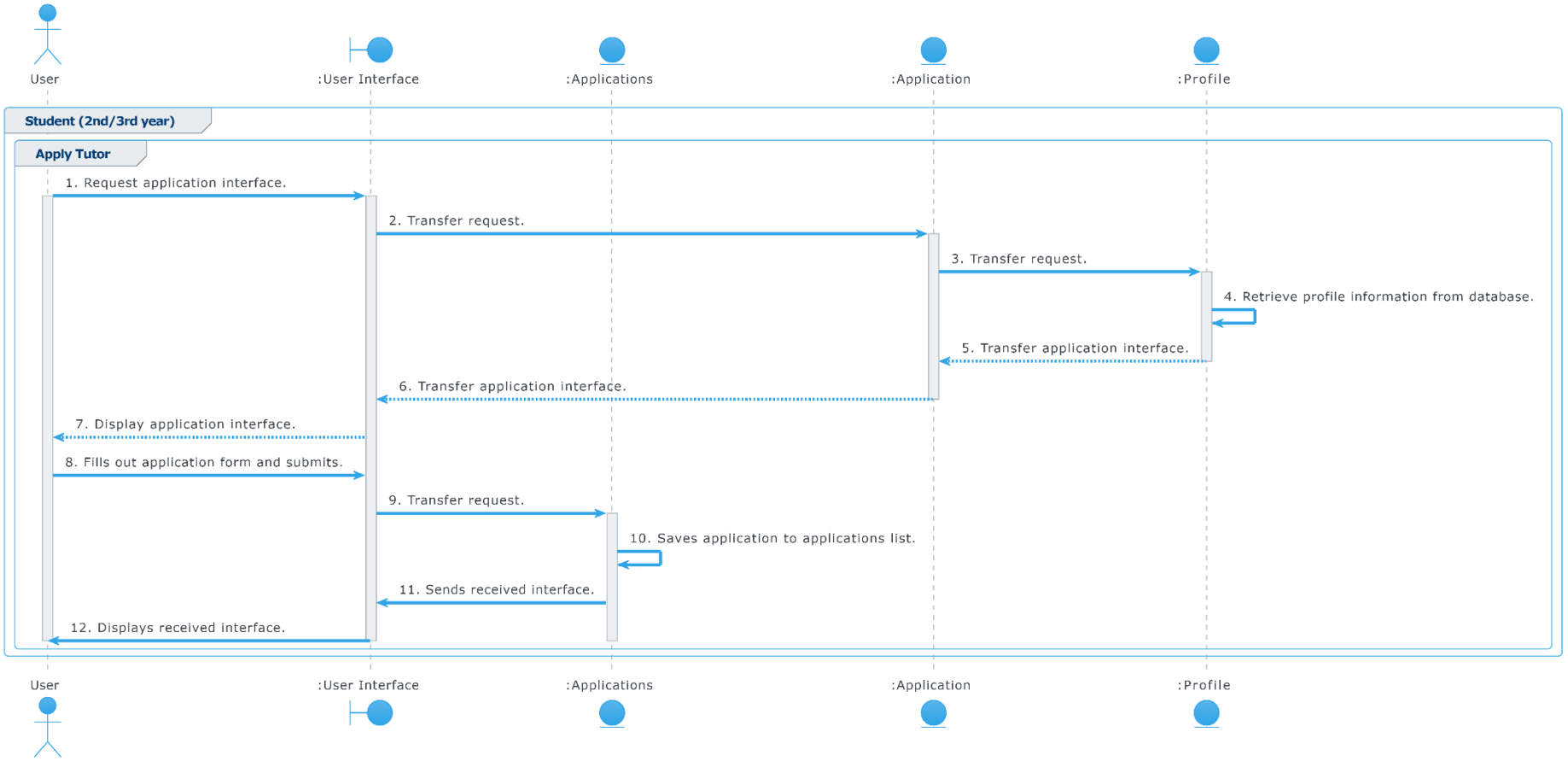


Figure 133: Apply to be a Tutor Sequence Diagram

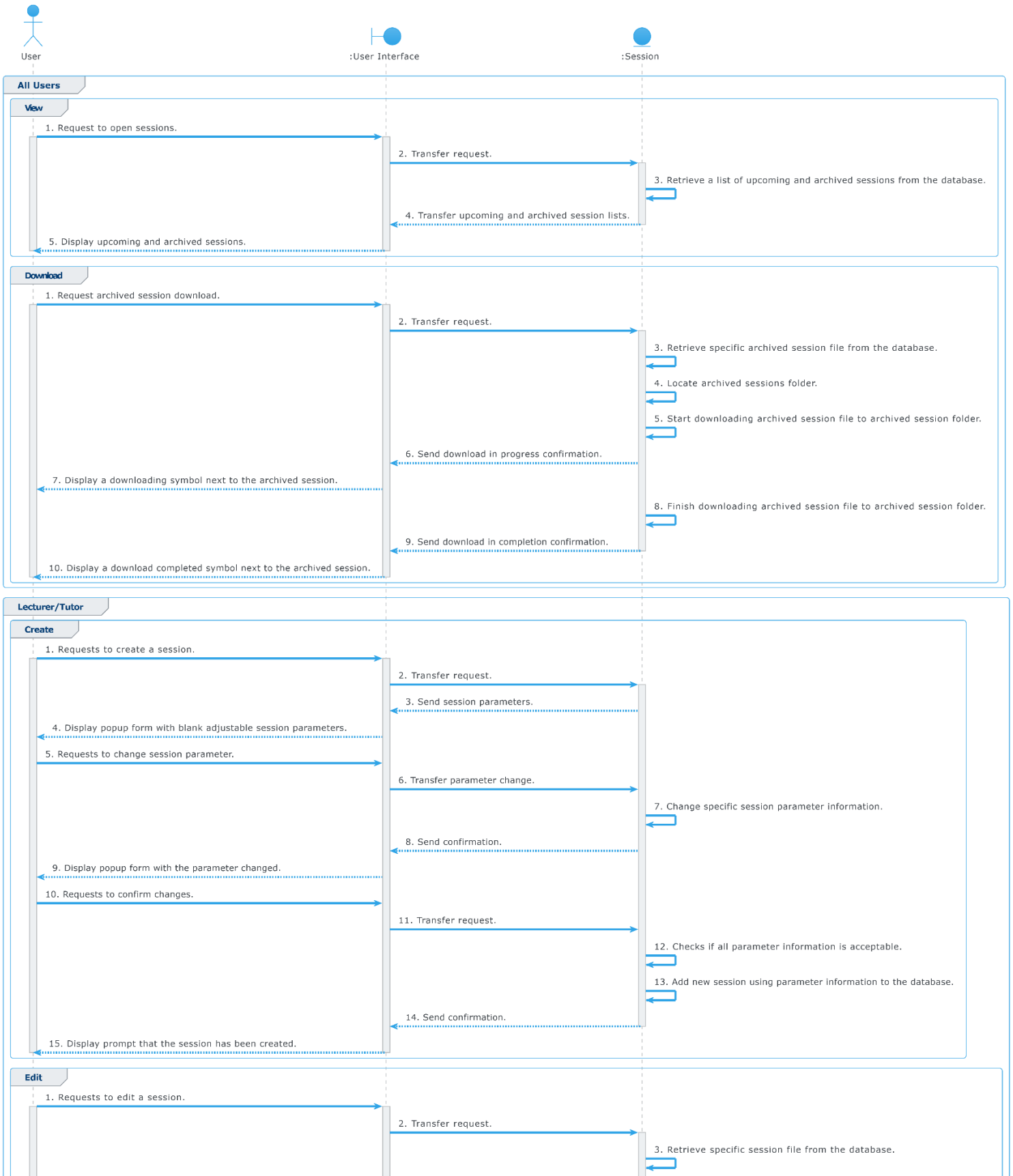


Figure 134: Session Sequence Diagram

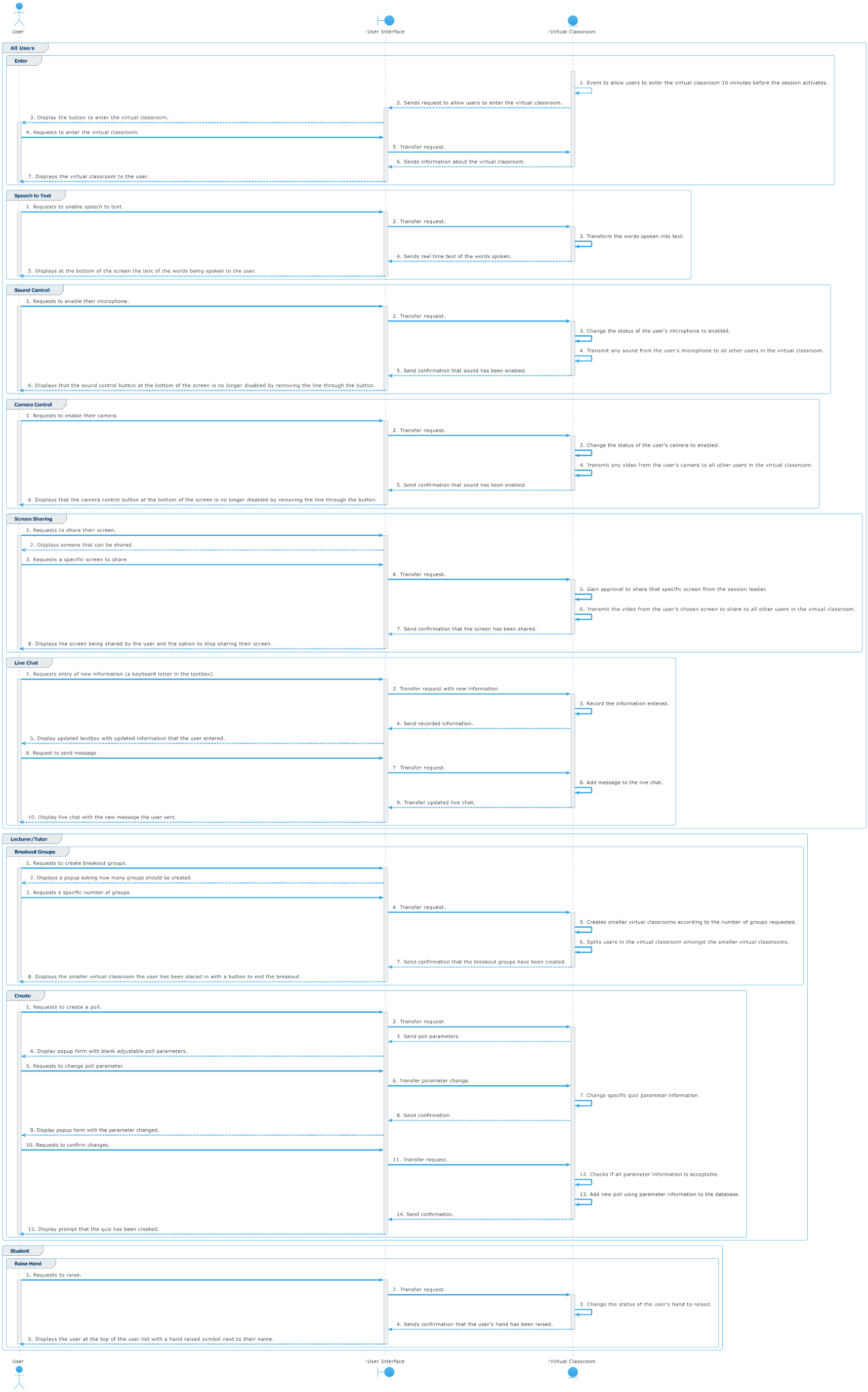


Figure 135: Virtual Classroom Sequence Diagram

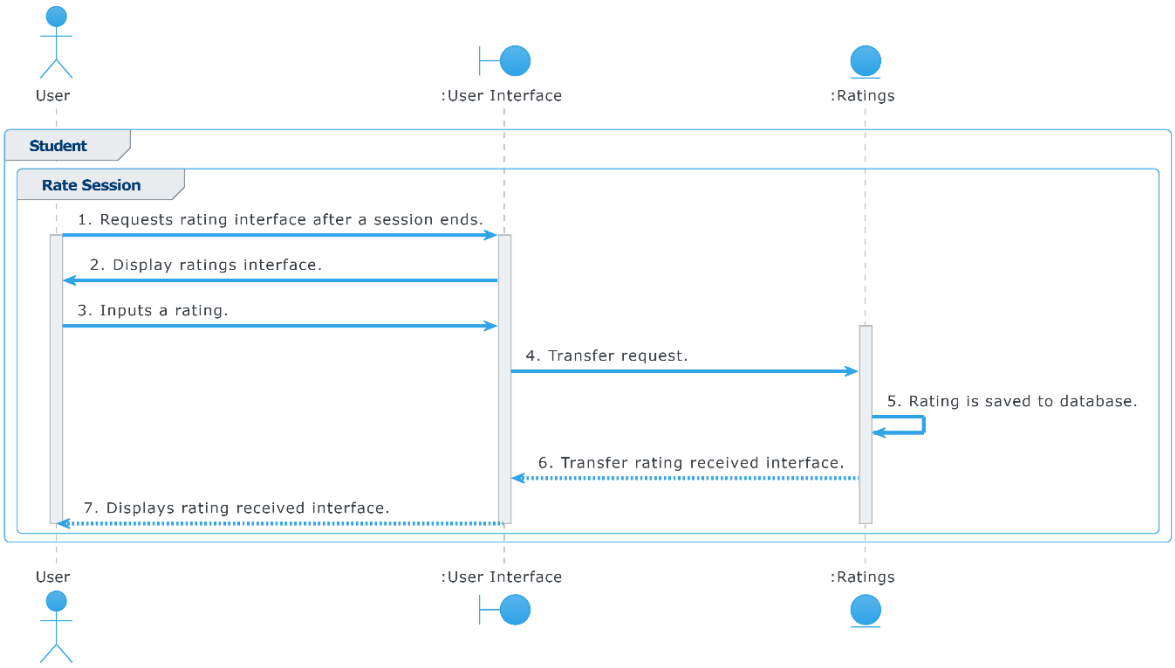


Figure 136: Rate Session Sequence Diagram

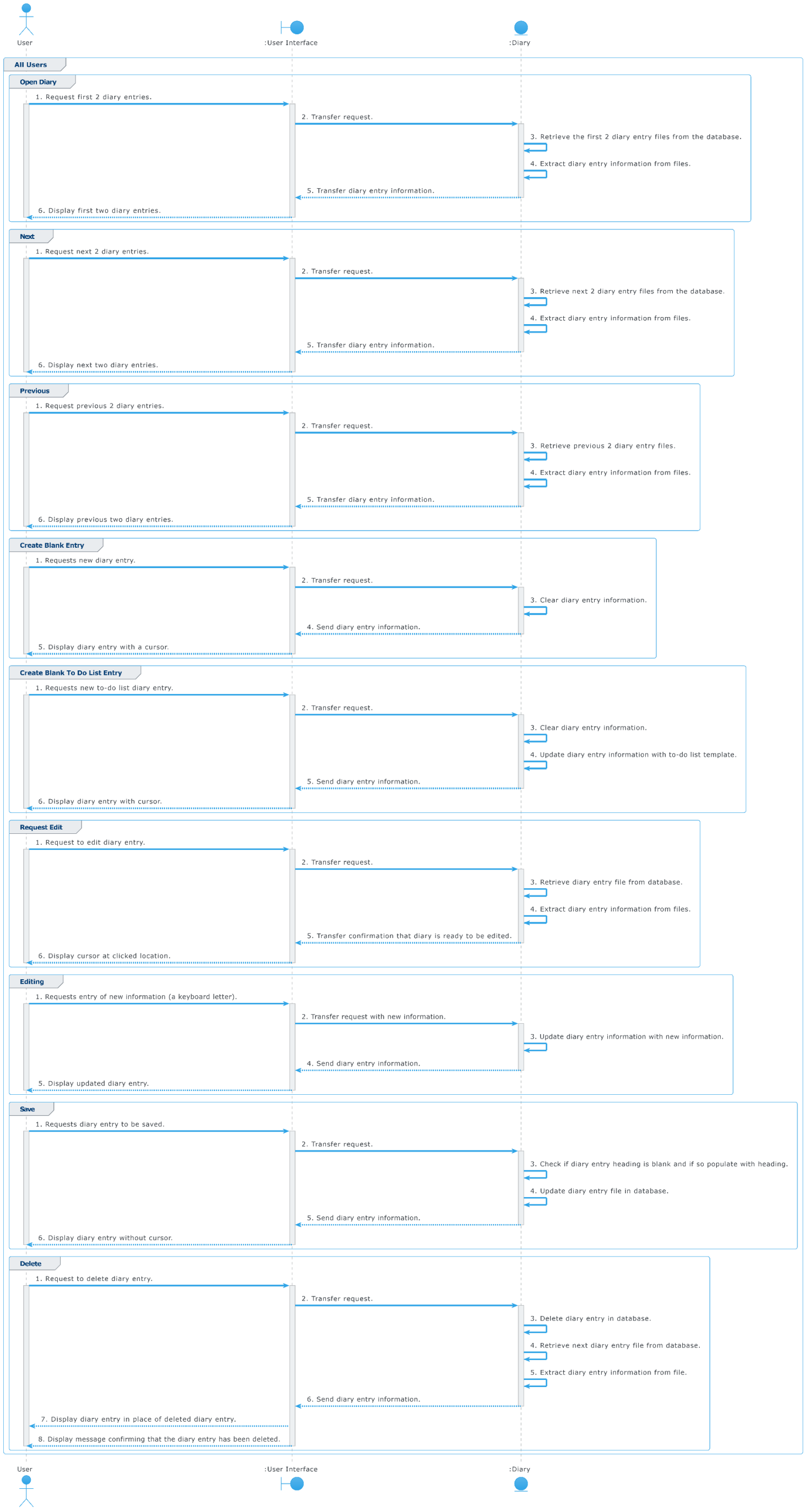


Figure 137: Diary Sequence Diagram

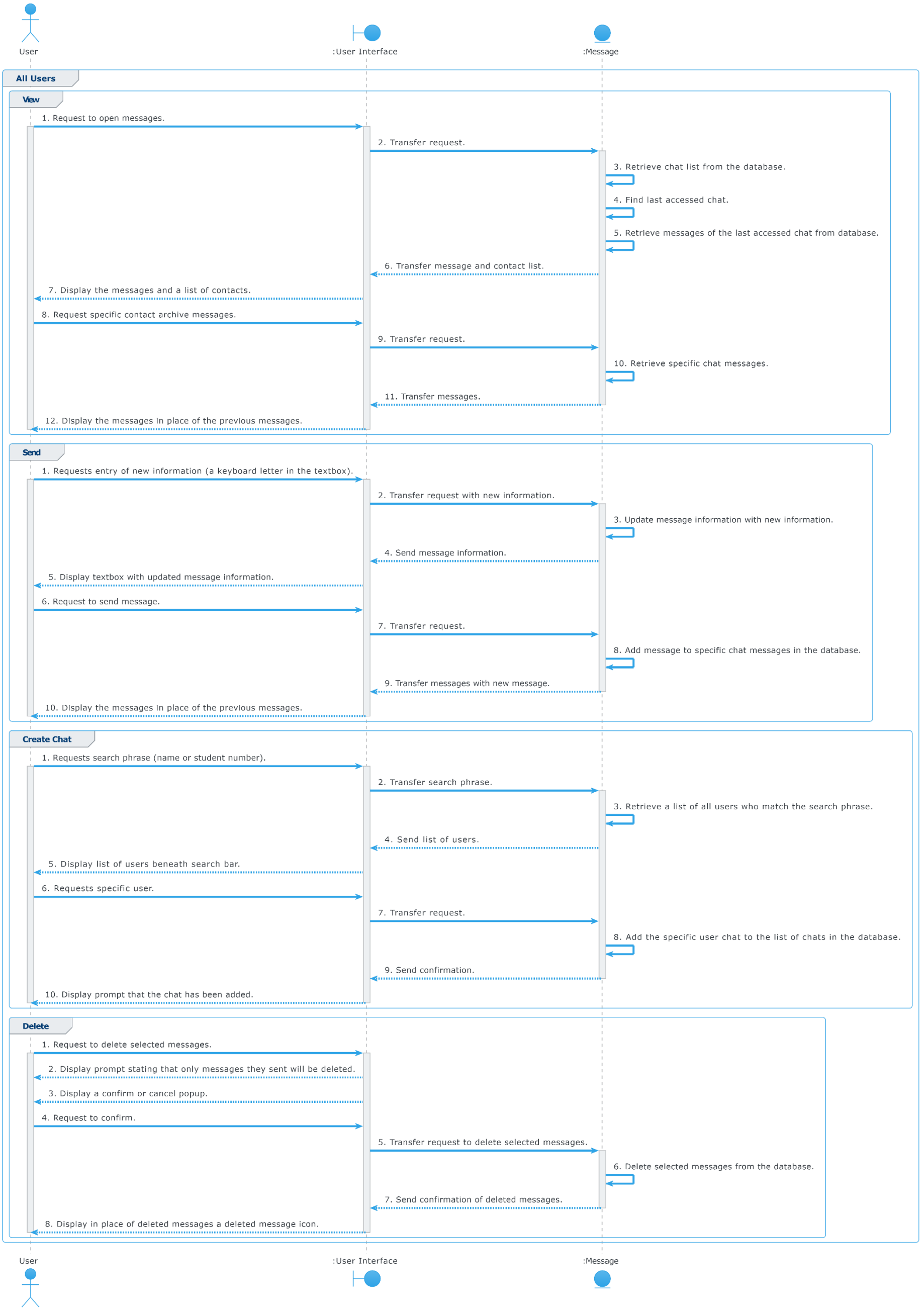


Figure 138: Direct Message Sequence Diagram

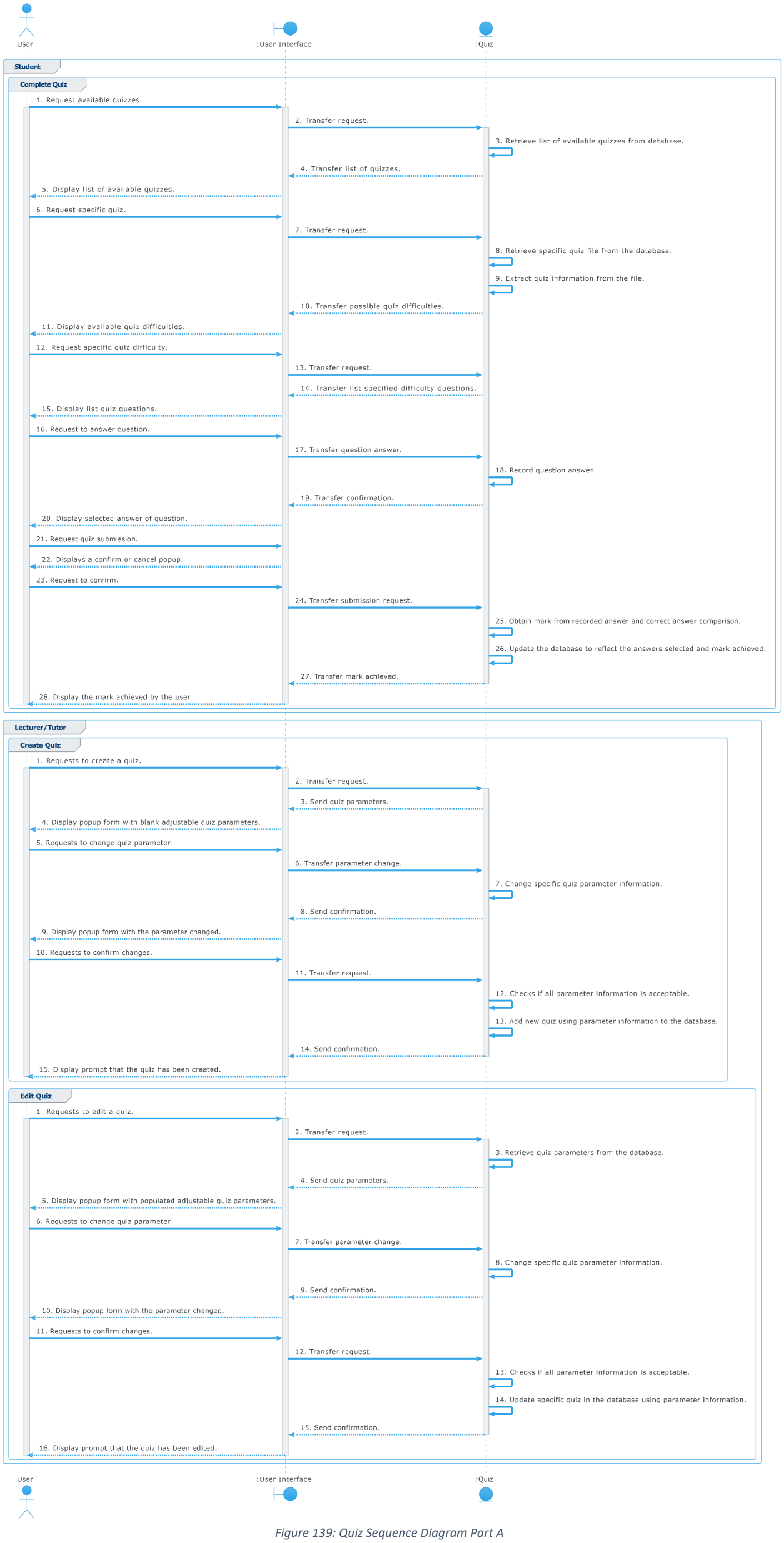


Figure 139: Quiz Sequence Diagram Part A

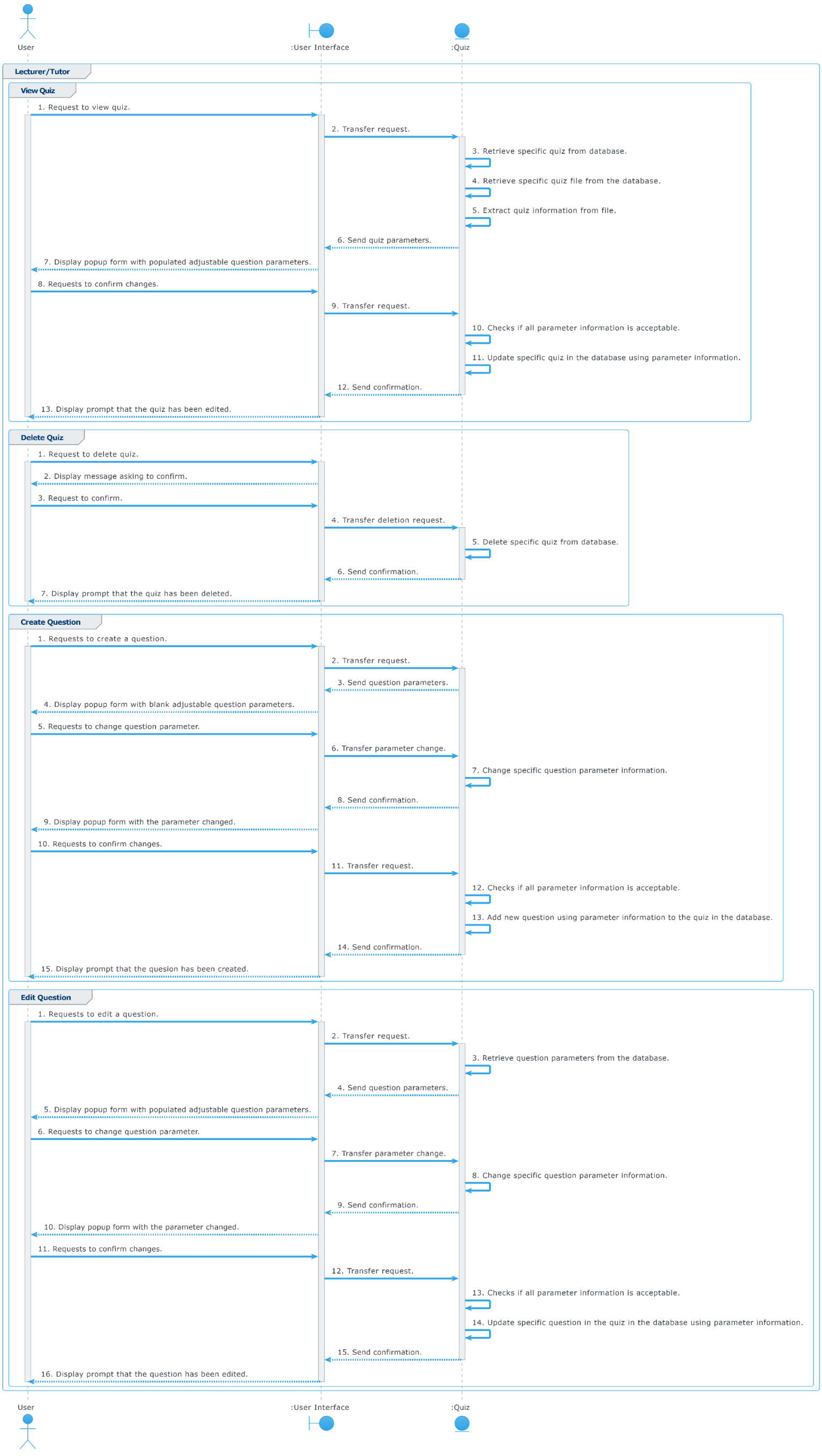


Figure 140: Quiz Sequence Diagram Part B

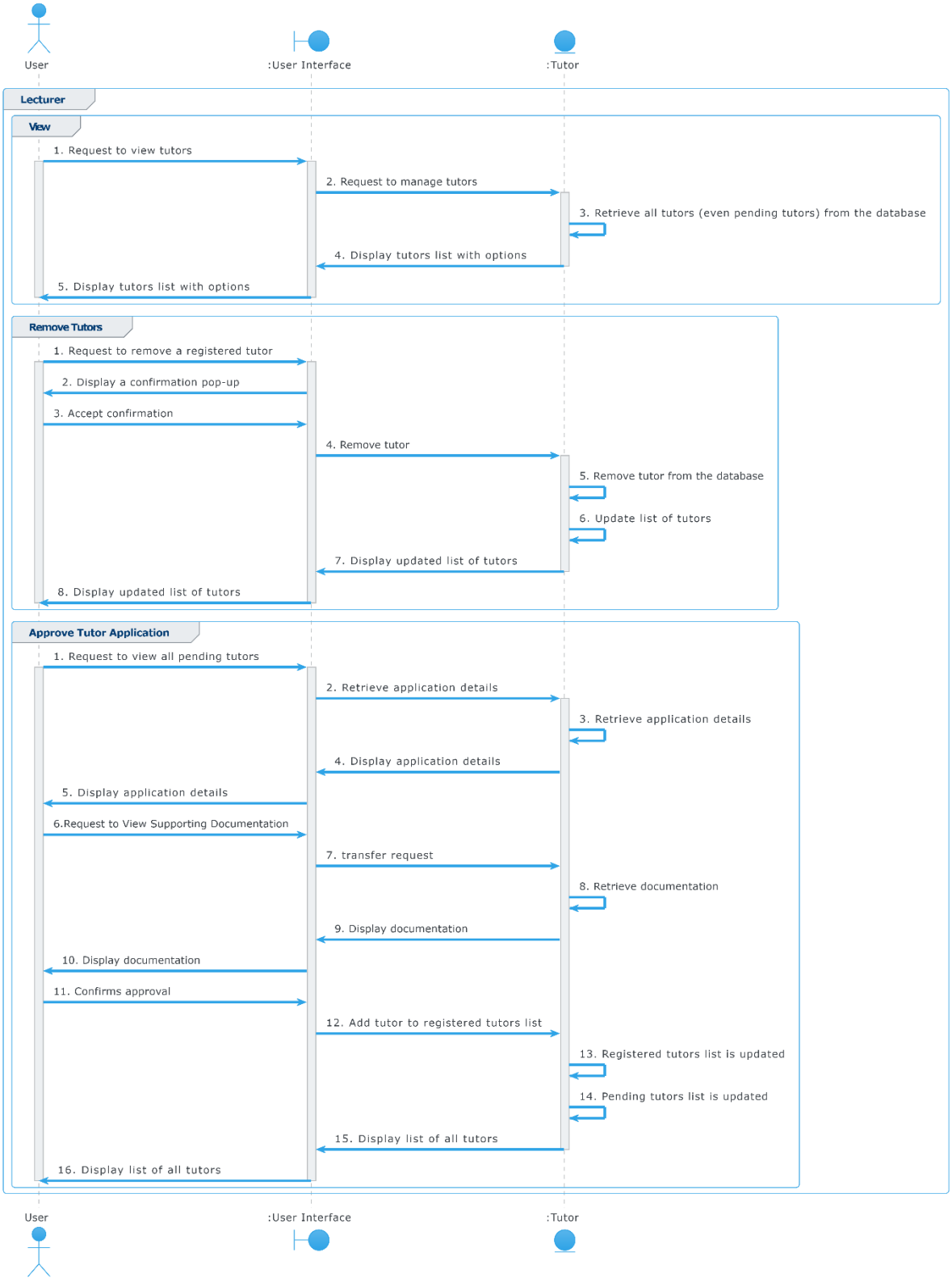


Figure 141: Coding Problems Sequence Diagram

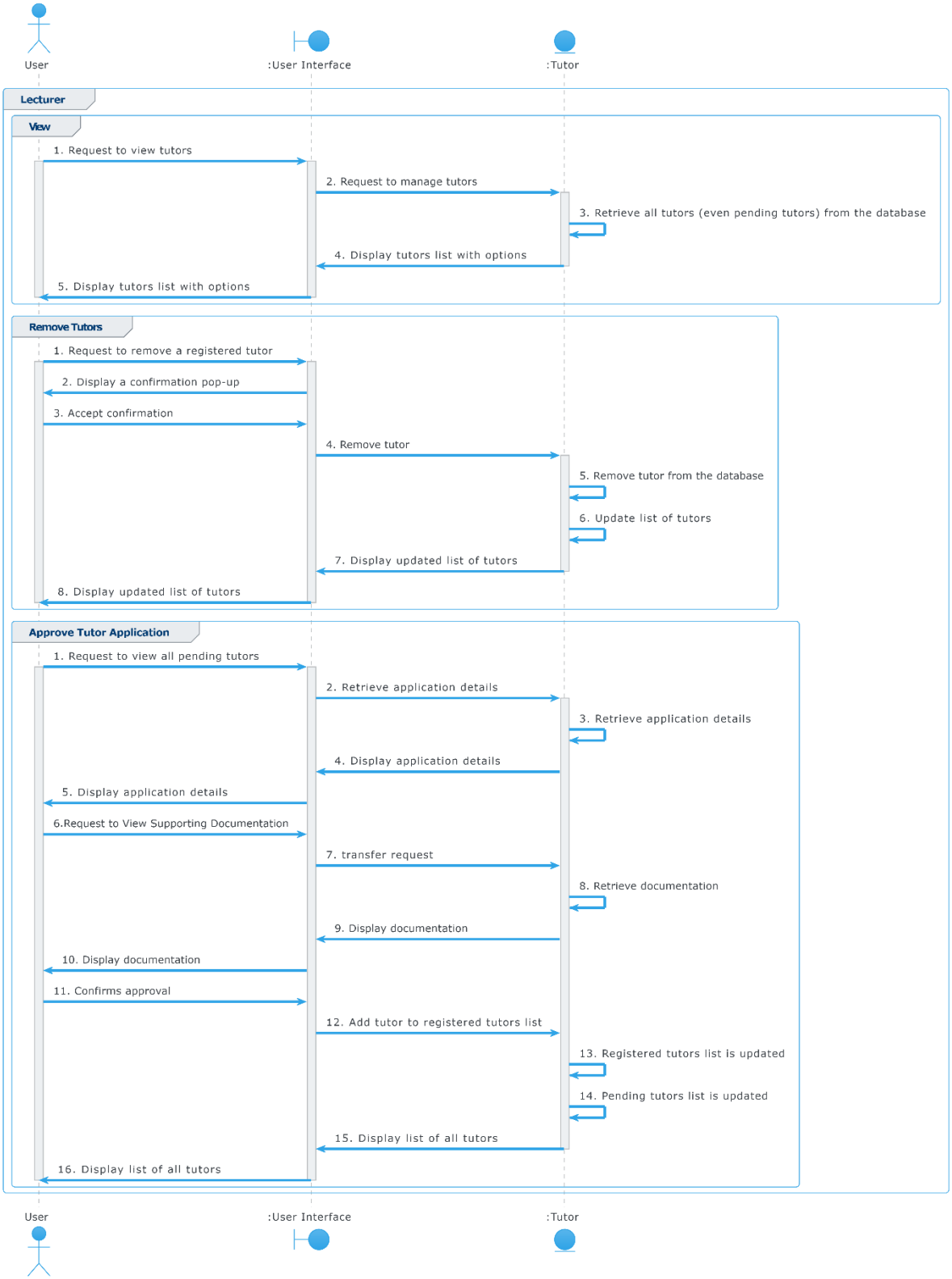


Figure 142: Manage Tutors Sequence Diagram

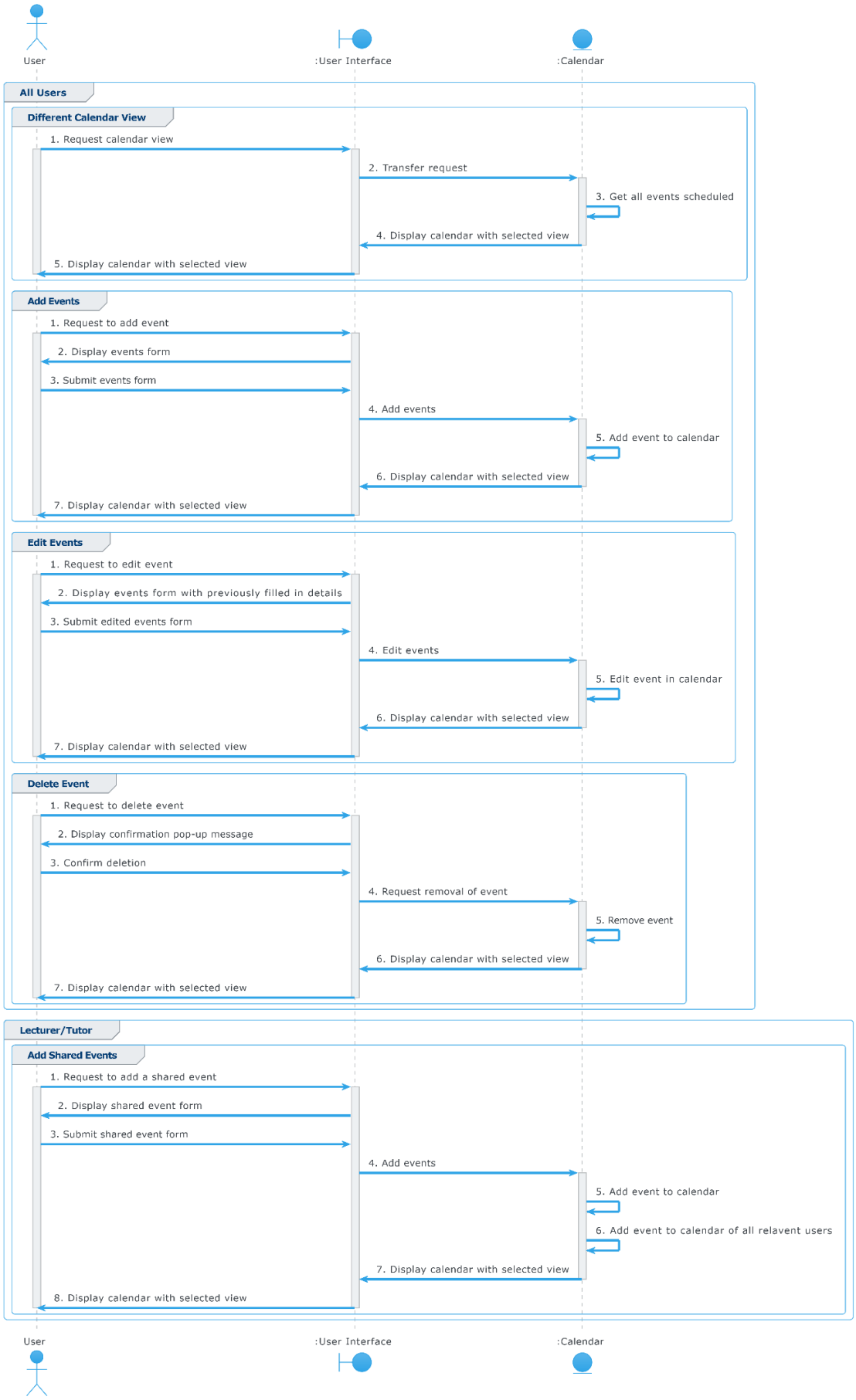


Figure 143: Calendar Sequence Diagram

State Charts

State charts were drawn for each of the following roles:

- Lecturer,
- Tutor,
- Student (who as currently enrolled for 1st year computer science modules),
- And additionally, a 2nd/3rd year student who can apply to be a tutor.

The state charts are as follows:

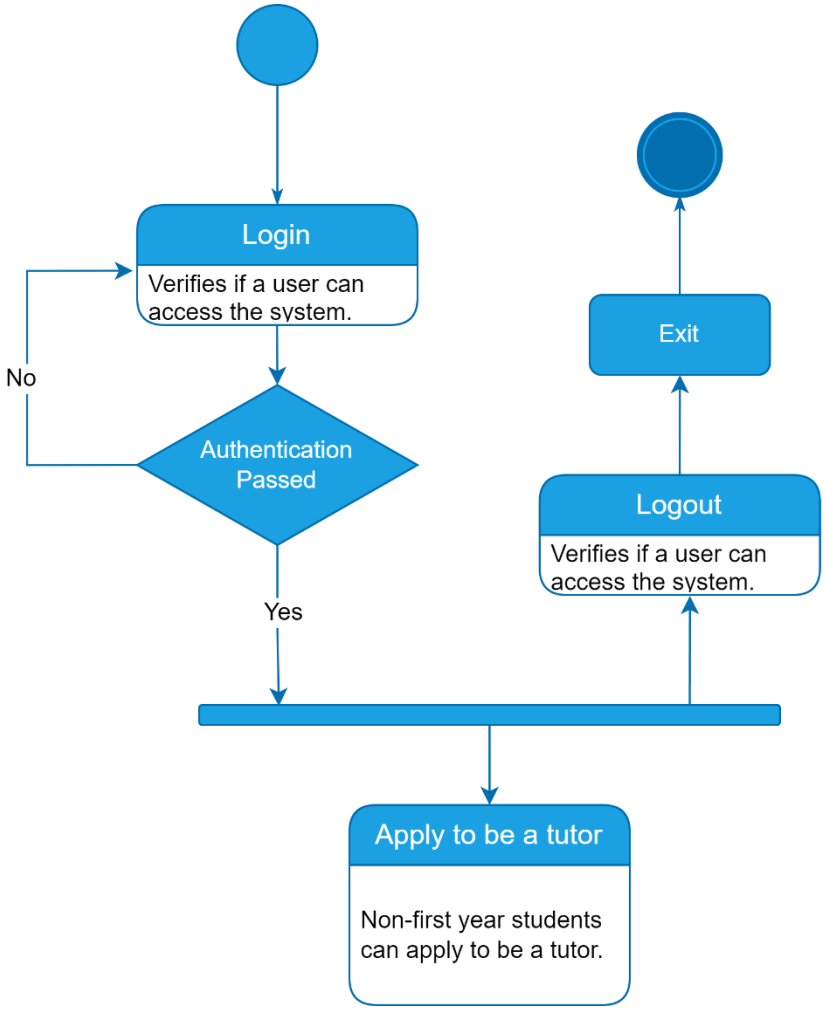


Figure 144: 2nd or 3rd Student State Chart

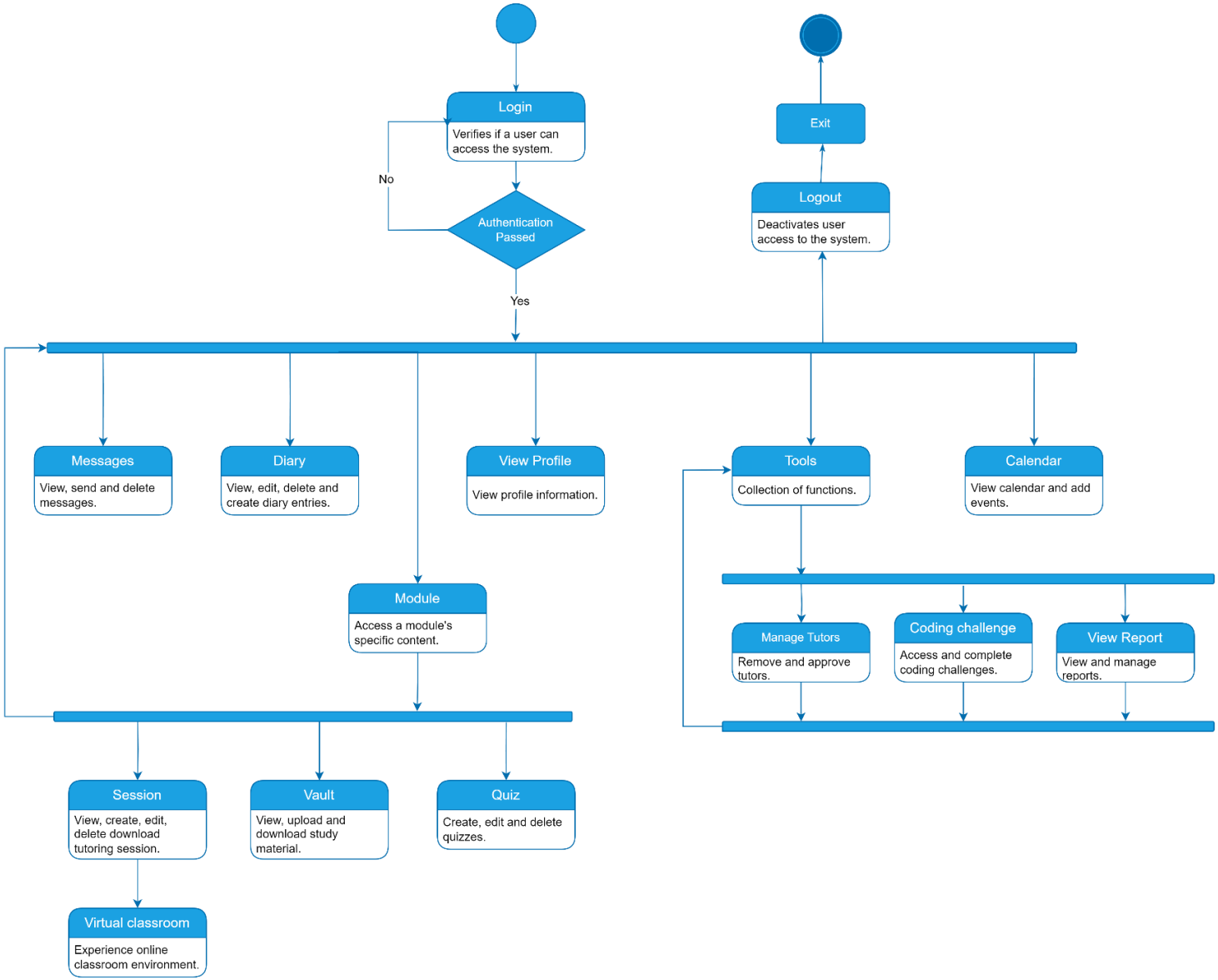


Figure 145: Lecturer State Chart

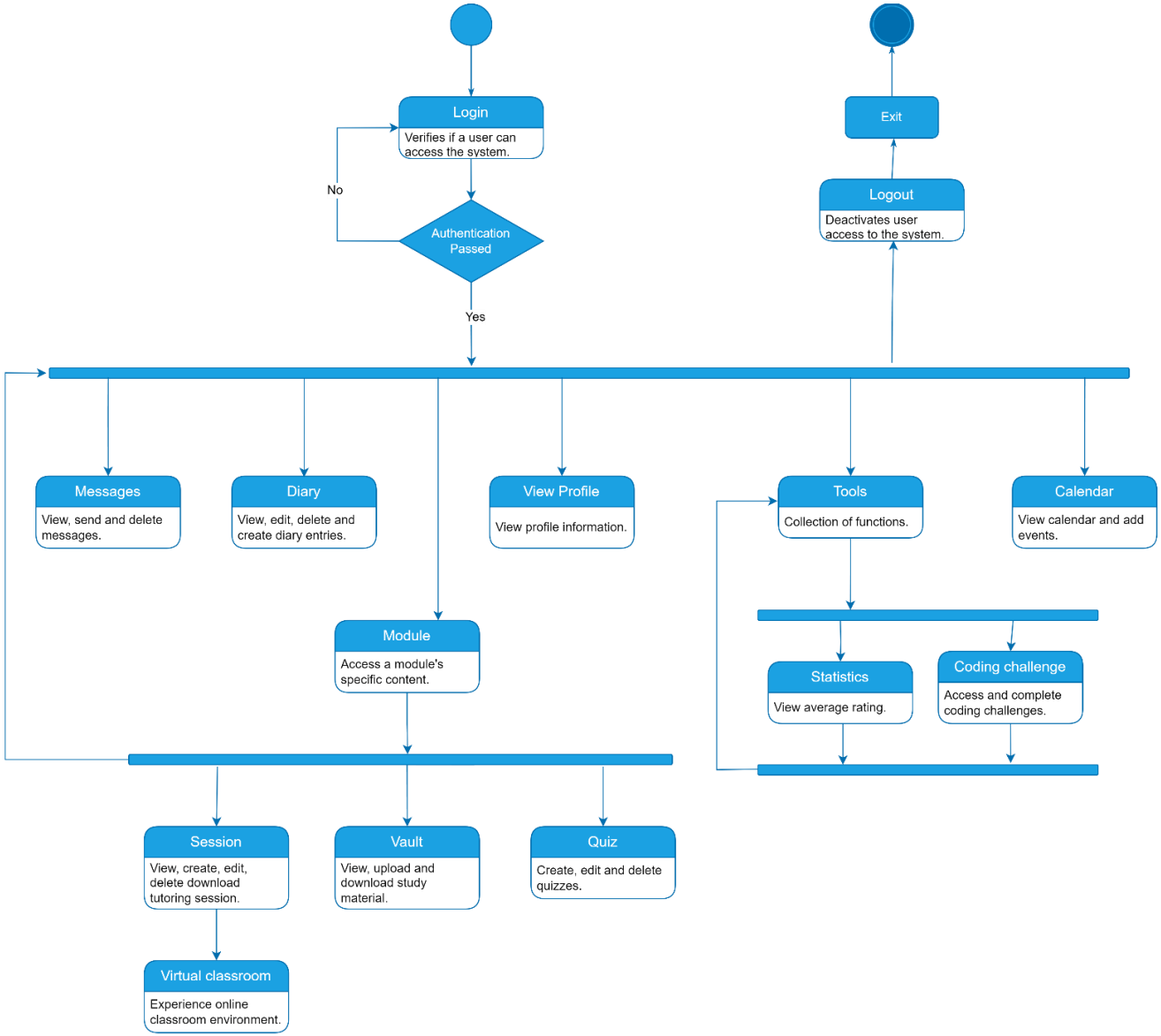


Figure 146: Tutor State Chart

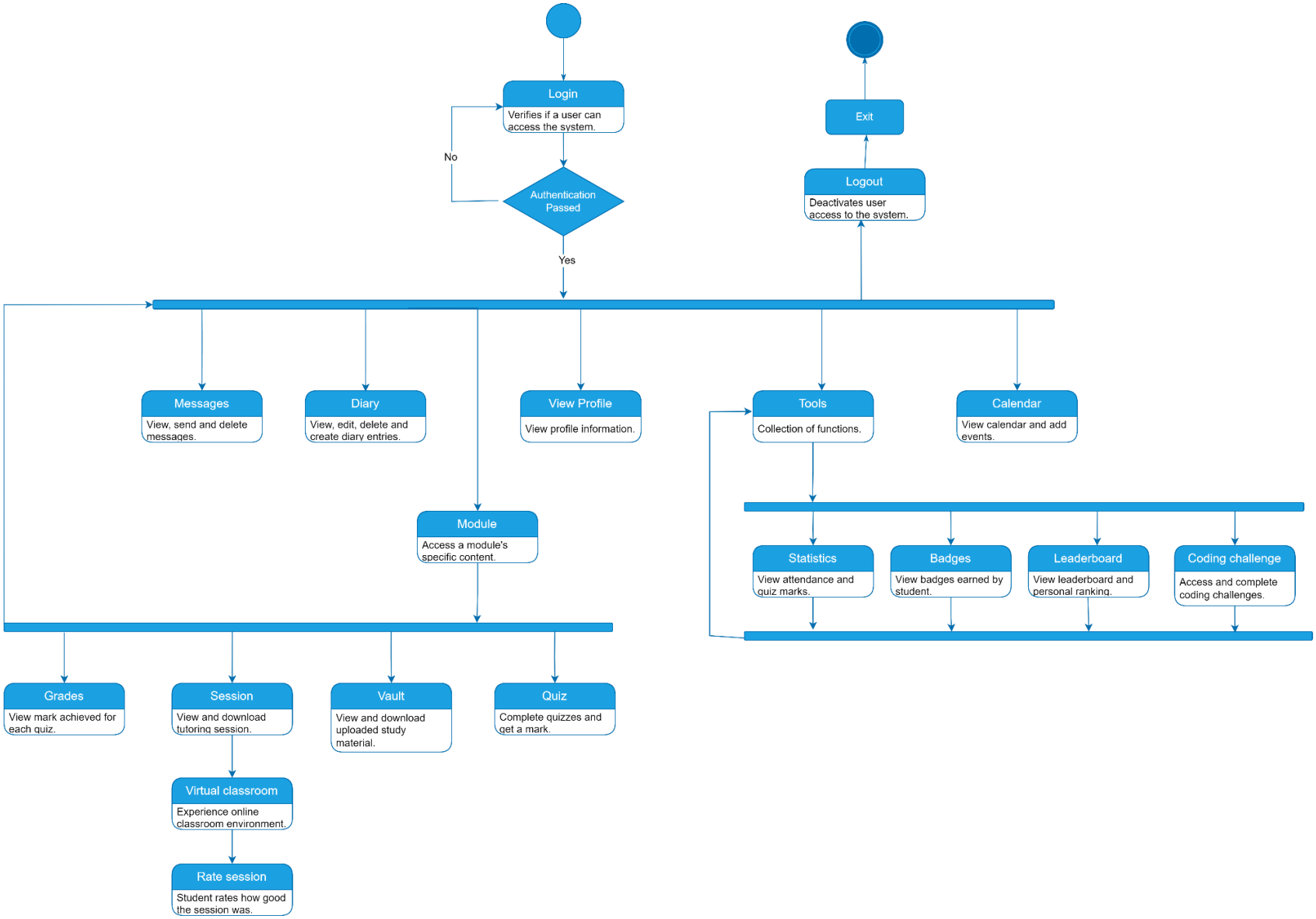


Figure 147: Student State Chart

Noun Extraction

1. Product Description

A user logs into the system. A user can be a student, tutor, lecturer, or admin. Once a user logs-in, he/she is presented with a dashboard and a side-menu. All users have a dashboard that display announcements and a series of tools. These tools include a leaderboard, statistics, code compiler with different difficulty problems. Furthermore, if the user is a student, they will also be able to view the badges they were awarded. If the user is a lecturer, they will have the additional tools of reports and applications.

On the side-menu a user can navigate to a calendar, chat, a diary, courses or log-out of the system. A calendar will contain events. Chat will facilitate sending, receiving and storage of messages between users. A diary with entries containing text, voice recordings and to-do lists. Courses contains a list of modules. When a user opens a module a new side-menu is created. Using the new side-menu a user can navigate to vault, quiz, grade and session. Vault contains study items. Quiz contains a list of available quizzes. Grades contains a list of marks obtained. Session contains a list of archived and upcoming sessions. When a user joins an ongoing session, they will have the following functionalities: live polls, hand raising, breakout groups as well as microphone and camera control.

Throughout the system there is a filter or search option available.

2. Identify the Nouns

A user logs into the system. A user can be a student, tutor, lecturer, or admin. Once a user logs-in, he/she is presented with a dashboard and a side-menu. All users have a dashboard that display announcements and a series of tools. These tools include a leaderboard, statistics, code compiler with different difficulty problems. Furthermore, if the user is a student, they will also be able to view the badges they were awarded. If the user is a lecturer, they will have the additional tools of reports and applications.

On the side-menu a user can navigate to a calendar, chat, a diary, courses, or log-out of the system. A calendar will contain events. Chat will facilitate sending, receiving and storage of messages between users. A diary with entries containing text, voice recordings and to-do lists. Courses contains a list of modules. When a user opens a module a new side-menu is created. Using the new side-menu a user can navigate to vault, quiz, grade, and session. Vault contains study items. Quiz contains a list of available quizzes. Grades contains a list of marks obtained. Session contains a list of archived and upcoming sessions. When a user joins an ongoing session, they will have the following functionalities: live polls, hand raising, breakout groups as well as microphone and camera control.

Throughout the system there is a filter or search option available.

3. Identify the Classes

From the above paragraph we identify the following classes:

Nouns
1. Study Item 2. Question 3. Quiz 4. Abstract Session 5. Upcoming Session 6. Archived Session 7. Module 8. Abstract User 9. Calander 10. Event 11. Dairy 12. Entry 13. Chat 14. Message 15. Badge 16. Student 17. Lecturer 18. Tutor 19. Application 20. Report 21. Student Grade

4. Class-responsibility-collaboration Cards

We will identify different types of classes as follows:

- Abstract
- Child
- Normal

Class-responsibility-collaboration Cards

The following Class-responsibility-collaboration Cards were drawn up from the identified classes.

Study Item

Stereotype	Entity
Description	A ‘study item’ is a type that is used to store information in a ‘vault’.
Responsibilities	• A lecturer/tutor must be able to add, edit and delete a ‘study item’. • A user must be able ‘view’ the study item. They must also be able to ‘download’ the item which will download the item to a file created by the system and automatically open the file to be viewed by the user.
Collaborators	Vault
Attributes	• `int Study_Item_ID` • `string Name` • `upload Upload_Item` (e.g. .pdf, .docx, .ppt, etc.) • `double Size`
Relationships	Aggregation

Vault

Stereotype	Entity
Description	The vault connects a module and its associated study item’s.
Responsibilities	• All users must be able to view a list of the study items specific to a module.
Collaborators	Study Item, Module
Attributes	• `list<Study_Item_ID> Vault` • `int Module_Code`
Relationships	Aggregation

Question

Stereotype	Entity
Description	A ‘question’ is a type that is stored within a ‘quiz’. Students can then choose the option they believe to be the correct answer.
Responsibilities	• A lecturer/tutor should be allowed to view, edit and add a ‘question’. • A student should be allowed to answer a question when he submits the quiz. The students answer to the question should be compared with the answer that the tutor/lecturer added.
Collaborators	Quiz
Attributes	• `int Question_ID` • `string Description` • `enum Difficulty` • `int Points` • `string Option_1` • `string Option_2` • `string Option_3` • `string Option_4` • `string Answer` We acknowledge that the number of options should not have a fixed length of 4 but should rather allow the lecturer or tutor to set the length, however for the purposes of this assignment we will delimit the scope of this feature.
Relationships	Aggregation

Quiz

Stereotype	Entity
Description	A ‘quiz’ contains a list of ‘question’.
Responsibilities	• All students belonging to a specific module must be able to view a list quizzes. • A student must be able to select the difficulty of the quiz and attempt the quiz. The student should be allowed to attempt a specific quiz multiple time. • A lecturer/tutor should be allowed to view, edit and add a ‘quiz’. • Award points to a student on successful completion.
Collaborators	Question, Module

Attributes	• `int Quiz_ID` • `list<Question_ID>` • `dictionary<enum, bool> Difficulty` • `Timer Quiz_Timer` We acknowledge that the lecturer/tutor should be able to set a time limit for the quiz as well as a start / end date, however for the purposes of this assignment we delimit the scope.
Relationships	Aggregation

(Abstract) Session

Stereotype	Entity
Description	A ‘session’ is a function of a virtual classroom specific to a module. If the ‘session’ has not already occurred, it has a type ‘upcoming’ which contains the scheduled event details. If it has occurred, it has type ‘archived’ and contains the recording and related details.
Responsibilities	• Determine if the ‘session’ is ‘upcoming’ or ‘archived’. • Switch between the different ‘session’ types. • When a user joins an ongoing session, they will have the following functionalities: live polls, hand raising, breakout groups as well as microphone and camera control. • A lecturer/tutor should be allowed to view, edit and add a ‘session’. When a lecturer/tutor adds a ‘session’ it is automatically set to type ‘upcoming’ as it has not yet occurred.
Collaborators	Module
Attributes	• `int Session_ID` • `string Name`
Relationships	Aggregation

Archived Session

Stereotype	Entity
Description	An ‘archived session’ is a type of ‘session’ that has already occurred.
Responsibilities	N/A
Collaborators	Session
Attributes	• `double Size` • `datetime Date_Uploaded` • `video Uploaded_Recording`
Relationships	Generalization

Upcoming Session

Stereotype	Entity
Description	A ‘upcoming session’ is a type of ‘session’ that has not occurred.
Responsibilities	N/A
Collaborators	Session
Attributes	• `datetime Upcoming_datetime`
Relationships	Generalization

(Abstract) User

Stereotype	Entity
Description	A user that can access the system. Each user has access to a calendar, chat diary and the modules they are enrolled for/teaching.

Responsibilities	• A user should be authenticated. If a student is a 1st year in the department of computer science and informatics, then he/she can be allowed access to the system. If the student is a 2nd/3rd year, they are only allowed access to a dashboard with a single function, to apply to be a tutor. Similarly, only valid tutors and lecturers should be allowed access to the system.
	• A user has different types, including:
	  ○ Student,
	  ○ Lecturer,
	  ○ Tutor,
	  ○ Admin.

	Although we acknowledge that an admin should be part of the system, for the purposes of this assignment we will not implement the 'admin' role. • A user should be allowed to access their modules, calendar, diary, and Chat. as individual features based on their role. Details of the other features are discussed under the relevant user role's CRC card.
Collaborators	Module, Calendar, Chat, Diary, Leaderboard
Attributes	• `int User_Code` • `string Name` • `string Surname` • `string Email_Address` • `list<Module> Modules` • `list<Event> Calander` • `list<Entry> Diary` • `list<Message> Chat`
Relationships	Aggregation

Student

Stereotype	Entity
Description	A 'student' is a type of user. A student has access to the features of a user (details are available under the User CRC card). A student has additional features including reporting a tutor, being awarded badges, and viewing badges and lastly, viewing their statistics.

Responsibilities	• A student should be able to be awarded badges for excellent work. A student should also be able to view these badges.
	• A student's statistics should be updated by the system. The student should also be able to view their statistics including:
	  ○ Average Attendance,
	  ○ Average Quiz Grade.
	• A student should be able to report a tutor.

Collaborators	Tutor Report, Tutor Application, Badges, Statistics
Attributes	• `string Student_Number` • `double Average_Grade` • `list<Tutor> Report_Tutor` • `list<Badges> Badges` • `list<Module> Modules` • `double Average_Attendance`
Relationships	Generalization, Aggregation

Lecturer

Stereotype	Entity
Description	A 'lecturer' is a type of user. A lecturer has access to the features of a user (details are available under the User CRC card).

Responsibilities	• A lecturer should be able to manage their tutors, including:
	  ○ Approving applications,
	  ○ Removing a tutor.
	• A lecturer should be able to manage reports made against tutors, including:
	  ○ View Report,
	  ○ Message.
	• A tutor should be able see the statistics of their students and tutors. We acknowledge that a lecturer should also be able to view the statistics for all his students and tutors, however for the purposes of this assignment we will delimit the scope of this feature.
	• A lecturer should be able to add, remove and edit announcements.
	• A lecturer should be able to manage a module including the vault and sessions. The details of the vault and session is discussed in the relevant CRC cards.
	• A lecturer should be able to see the students' badges. However, purposes of this assignment we will delimit the scope of this feature.

Collaborators	Tutor Report, Tutor Application
Attributes	• `string Staff_Number` • `list<Report> Reports` • `list<Application> Applications`
Relationships	Generalization, Aggregation

Tutor

Stereotype	Entity
Description	A 'tutor' is a type of user. A tutor has access to the features of a user (details are available under the User CRC card).
Responsibilities	• A tutor's rating should be updated by the system. The tutor should also be able to view their average rating. • A tutor should be able see the statistics of their students.

	We acknowledge that a tutor should also be able to view the statistics for all his students, however for the purposes of this assignment we will delimit the scope of this feature. • A tutor should be able to add, remove and edit announcements. • A tutor should be able to see the students' badges. However, purposes of this assignment we will delimit the scope of this feature. • A tutor should be able to manage a module including the vault and sessions. The details of the vault and session is discussed in the relevant CRC cards.
Collaborators	Statistics
Attributes	• `int Student_Number` • `double Average_Rating`
Relationships	Generalization, Aggregation

Module

Stereotype	Entity
Description	• A module can be accessed by any student who is currently enrolled for the module. It can also be accessed by a tutor/lecturer teaching the module. • A module contains a vault, quizzes, sessions, and grades. The details of the implementation for the vault, quizzes, sessions, and grades are discussed under their relevant CRC cards.
Responsibilities	• The system should be able to determine and display all the modules that a student is enrolled for / a lecturer or tutor is teaching for the semester. These modules should be displayed on the user dashboard. • The system should be able to determine the modules that the student or tutor/lecturer has access to using the PeopleSoft database. • The system should be able to change the modules that the student or tutor/lecturer has access to each semester and show the new modules on the user dashboard. • The module should store all the information for the vault, quiz, sessions, and the student grades. The details of the vault, quiz, sessions and grades are discussed in the relevant CRC cards.
Collaborators	Vault, Quiz, Session, User

Attributes	• `int Module_Code` • `string Name` • `Vault vault` • `list<Quiz> Quizzes` • `list<Session> Sessions`
Relationships	Aggregation

Calendar

Stereotype	Entity
Description	A calendar acts as a schedule for users to see upcoming events. Events scheduled by tutors/lecturers can be seen by students of their modules. Users can also schedule their own events. A Calander has a daily and monthly view.
Responsibilities	• Tutors/lecturers can add, edit and delete events that are displayed on students' calendar. • Switch between the daily and monthly view.
Collaborators	Event
Attributes	• `int Calander_ID` • `list<Event> Events`
Relationships	Aggregation

Event

Stereotype	Entity
Description	An event is an item of a calendar.
Responsibilities	• Users can add, edit, and delete their own events. • Events can be dragged by the user to a new timeslot and should be scheduled for that time slot.
Collaborators	Calander
Attributes	• `int Entry_ID` • `string Title` • `int Module_Code` • `datetime Start_DateTime` • `datetime End_DateTime`
Relationships	Aggregation

Diary

Stereotype	Entity
Description	An entry is an item contained in a diary. An entry can contain a recording, text, or a to do list.

Responsibilities	• A user should be able to add, edit or delete an entry. To add an entry the user can use the relevant button displayed at the bottom of the page.
	  ○ The user can hold the recording to record their input. The user can release the button they are finished. This will add the recording where the user's cursor is.
	  ○ A user can click the text / to-do lost button which will add a template where the user's cursor is. The user can then edit the template.
	• A user should be able to interact with a recording entry. When a user clicks on the recording icon a popup should appear. Using this popup, the user can play or pause the recording.
	• A user should be able to toggle the checkbox next to a to-do list item to show if the item is completed or not.

Collaborators	User

Attributes	• `int Diary_ID`
	• `list<Entry> Events`
	Recording
	  ○ `upload Recording`
	Text
	  ○ `string Title`
	  ○ `string Body`
	To-Do
	  ○ `string Title`
	  ○ `Dictionary<String, Bool> todo_Item`

Relationships	Aggregation

Chat

Stereotype	Entity
Description	A chat allows users to send messages to one another. It also allows a lecturer to communicate with a student who has made a report against a tutor either anonymously or not anonymously.
Responsibilities	• Allows a user to make a new chat by searching for a user and sending a message. • Store and retrieve chats made by users. • Show the time and date that messages were sent. • Distinguish between reports and chats. • Distinguish between anonymous and non-anonymous reports.
Collaborators	Message, User
Attributes	• `int User_Code` (Sender) • `int User_Code` (Reciever) • `list<Message> Messages`
Relationships	Aggregation

Message

Stereotype	Entity
Description	An item sent in a chat.

Responsibilities	• A user should be able to send a message to another user. The message should contain:
	  ○ A body of text,
	  ○ And a date and time.
	• A user should also be able to delete messages.
	We acknowledge that users should be able to send an attachment with a message, but for the purposes of this project we delimit the scope.

Collaborators	Chat
Attributes	• `int Message_ID` • `string Body` • `datetime Sent_DateTime` • `[Optional] Attachment`
Relationships	Aggregation

Badge

Stereotype	Entity
Description	A badge is awarded to a student on the completion of a goal.

Responsibilities	• The system should automatically award students badges when a goal is met, goals include:
	  ○ 80% attendance of sessions,
	  ○ 75% average for quizzes,
	  ○ The completion of certain coding challenges.
	• Allow a student to view badges that have been awarded to them.
	• Allow lecturers/tutors to see badges that have been awarded to students. Although this feature will not be implemented for this project.
	We acknowledge that lecturers should be able to add, edit and delete badges that can be awarded to students, but for the purposes of this project we will not implement this feature.

Collaborators	Student
Attributes	• `int Badge_ID` • `string Name` • `string Requirement` • `image Image_ID`
Relationships	Aggregation

Tutor Application

Stereotype	Entity
Description	An 2nd or 3rd year student can fill in an application to become a tutor for a specific module.
Responsibilities	• The system should be able to distinguish between student who are allowed to send an application and students who are not. • The system should determine which modules the student can apply to be a tutor for and populate a dropdown box. • A lecturer should be able to approve or deny an application submitted by a student. • The system should be able automatically determine the mark obtained by the student for the module and send this together with the application to the lecturer. We acknowledge that a student should be able to withdraw their application but for the purposes of this assignment we will not include this feature.

Collaborators	Student, Module, Lecturer
Attributes	• `int Application_ID` • `int Student_Number` • `int Module_Code` • `string Description` • `[attachment] Supporting_Document`
Relationships	Aggregation

Tutor Report

Stereotype	Entity
Description	A student should be able to make a report against a tutor for inappropriate behaviour. Thereafter, a lecturer should be able to view and submit comments on a report.
Responsibilities	• A student should be able to submit a report. We acknowledge that if a report is made but not submitted that the system must flag the student and the report must be made available to the lecturer, however for the purposes of the assignment, this feature will not be included. • A lecturer should be able to add comments to a report and be able to change the status of a report. • A lecturer should be able to communicate with the student who has submitted a report. The detail of this implementation is specified in the chat and messages CRC cards.
Collaborators	Student, Lecturer
Attributes	• `int Report_ID` • `int Student_Number` (Reported) • `int Student_Number` (Reportee *if not anonymous) • `string Description` • `[attachment] Supporting_Document` • `bool Remain_Anonymous`
Relationships	Aggregation

Leaderboard

Stereotype	Entity
Description	A leaderboard displaying the current ranking of all students based on their points.

Responsibilities	• A user should be allowed to view the leaderboard.
	• A user should be able to switch between the different types of viewing options:
	  ○ All time,
	  ○ Weekly,
	  ○ Monthly.
	• A student should be shown their current ranking.
	We acknowledge that tutors and lecturers should be able to see the ranking of students enrolled for their modules however for the purposes of this assignment this feature will not be included.

Collaborators	User
Attributes	• `int Student_ID` • `int Number_of_Points` • `int Ranking` • `int Current_View`
Relationships	Aggregation

Code Problems

Stereotype	Control
Description	A coding problem that a user can solve. A problem can belong to many categories.

Responsibilities	• A user should be able to attempt a problem. The system should be able to compile and provide feedback to the user.
	• A student should be able to receive a mark (in %) for a coding problem.
	• A student should be award points on successful completion of a problem.
	• The system should be able to show the user if the problem is:
	  ○ Pending or
	  ○ Completed.
	We acknowledge that a lecturer/tutor should be able to add, edit and delete problems but for the purposes of this assignment problems were added by an admin user and are therefore not included.

Collaborators	User

Attributes	• `int User_ID` • `int Problem_ID` • `string Status` • `string Category` • `string Description` • `string Constraints` • `string Output`
Relationships	Aggregation

Statistics

Stereotype	Control
Description	Calculates the percentage and colour for statistics.
Responsibilities	• Calculates average quiz mark and average attendance for students as well as average rating for tutors. • Calculates the associated colour for that percentage.
Collaborators	Student, Tutor
Attributes	• `int User_ID` • `int Average` • `string Colour`
Relationships	Aggregation

We acknowledge that we should also include a user interface boundary class, however for the purpose of this project we will not include this class.

Unified Modelling Language Diagram

Find below an overview of the consolidated Unified Modelling Language Diagram:

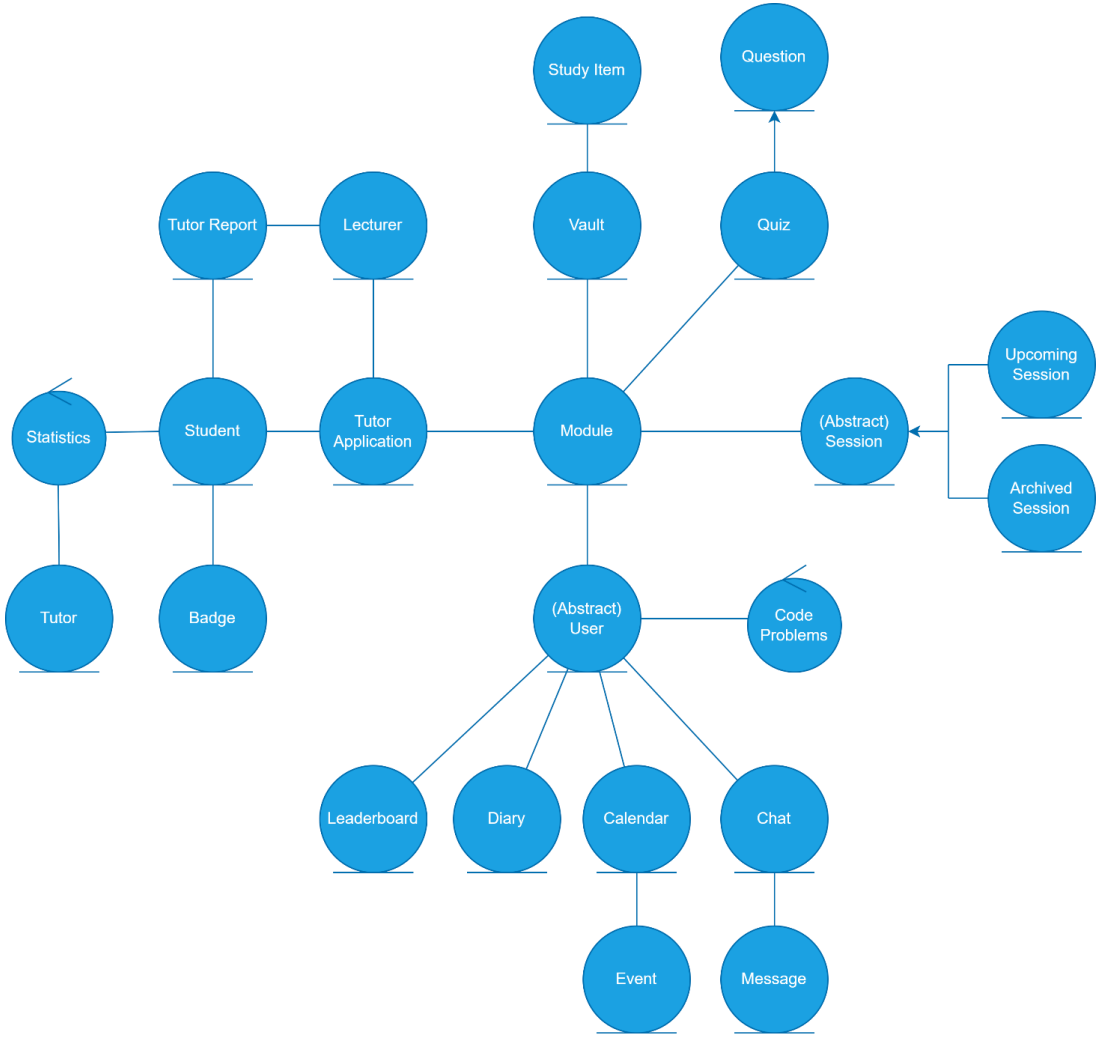


Figure 148: Consolidated Unified Modelling Language Diagram

Unified Modelling Language Classes

Find below the Unified Modelling Language Diagrams for the component classes:

Entity **Study Item**
+ Study_Item_ID : string # Name : string # Upload_Item : upload (e.g. .pdf, .docx, .ppt, etc.) # Size : double
+ addStudyMaterialItem(Name: string, UploadItem: upload, Size: double): bool + editStudyMaterialItem(itemId: int, newName: string, newUploadItem: upload, newSize: double): bool + deleteStudyMaterialItem(itemId: int): bool + viewStudyMaterialItem(itemId: int): string + downloadStudyMaterialItem(): void

Entity **Vault**
+ Vault : list<Study_Item_ID> + Module_Code : int
+ viewVault (Module_Code: int): list<Study_Item_ID>

Entity **Question**
+ Question_ID : int # Description : string # Difficulty : enum # Points : int # Option_1 : string # Option_2 : string # Option_3 : string # Option_4 : string # Answer : string
+ addQuestion(Description: string, Difficulty: enum, Points: int, Option1: string, Option2: string, Option3: string,Option4: string, Answer: string): bool + editQuestion(questionId: int, newDescription: string, newDifficulty: enum, newPoints: int, newOption1: string, newOption2: string, newOption3: string, newOption4: string, newAnswer: string): bool + deleteQuestion(questionId: int): bool

Entity **Quiz**
+ Quiz_ID : int # Quiz_Name : string # Question_List : list<Question> # Difficulty : enum # Quiz_Timer : Timer
+ addQuiz(name : string, QuestionList: list<Question>, Difficulty: enum, QuizTimer: Timer): bool + editQuiz(quizId: int, newQuestionList: list<Question>,, newDifficulty: enum, newQuizTimer: Timer): bool + deleteQuiz(quizId: int): bool + startQuiz(quizId: int, Student_Number : int): void + submitQuiz(quizId: int, Student_Number : int, answers: list<int>): void - UnlockNextDifficulty(quizId: int, Student_Number : int, difficulty : dictionary<enum, bool>): void

Entity **(Abstract) Session**
+ Session_ID : int # Name : string
+ addSession(Name : string) + editSession(Session_ID: int, newName: string): bool + deleteSession(Session_ID: int): bool

Entity **Archived Session**
Size : double # Date_Uploaded : datetime # Uploaded_Recording : video
+ changeSession(Session_ID: int): bool

Entity **Upcoming Session**
Upcoming_datetime : datetime
+ changeSession(Session_ID: int): bool

Entity **(Abstract) User**
+ User_Code : int # Name : string # Surname : string # Email_Address : string

Modules : list<Module> # Calander : list<Event> # Diary : list<Entry> # Chat : list<Message>
+ viewModules() : list<Module> + viewCalander() : list<Event> + viewDiary() : list<Entry> + viewChat() : list<Message>

Entity **Student**
+ Student_Number : int # Average_Grade : double # Report_Tutor : list<Tutor> # Badges : list<Badges> # Modules : list<Module> # Average_Attendance : double
Kindly see the relevant UML cards for details of the above.

Entity **Lecturer**
+ Staff_Number : int # Reports : list<Report> # Applications : list<Application>
Kindly see the Tutor Application and Tutor Report UML cards for details of the Manage Reports and Manage Application functionalities.

Entity **Tutor**
+ Student_Number : int # Average_Rating : double
Kindly see the Statistics UML card for details of the above.

Entity **Module**
+ Module_Code : int # Name : string # Vault : Vault # Quizzes : list<Quiz> # Sessions : list<Session>
+ ViewVault() : Vault + viewQuizzes() : list<Quiz> + viewSessions() : list<Session>

Entity **Calendar**
+ Calander_ID : calendar # Events : list<Event>
+ ViewEvents() : list<Event> + ChangeCalendarView() : void + void DragEvent(Event_ID : int, newStartTime : DateTime, newEndTime : DateTime) : void

Entity **Event**
+ Entry_ID : int # Title : string # Module_Code : int # Start_DateTime : datetime # End_DateTime : datetime
+ AddEvent(Title : string, Module_Code : int, StartDateTime : DateTime, EndDateTime : DateTime, Make_Available_to_Students : bool) : bool + DeleteEvent(EventID : int) : bool + EditEvent(EventID : int, NewTitle : string, Module_Code : int, StartDateTime : DateTime, EndDateTime : DateTime, Make_Available_to_Students : bool) : bool

Entity **Diary**
+ Diary_ID : list<Entry> Recording: # Recording : upload Text: # Title : string # Body : string To-Do: # Title : string # todo_Item : Dictionary<string, bool>
+ AddText(Title : string, body: int) : bool + AddToDoListITem(Title : string, todo_Item : Dictionary<string, bool>) : bool + AddRecording(Recording : upload) + DeleteEvent(Diary_ID : int) : bool

+ EditEvent(Diary_ID: int, NewTitle : string, Module_Code : int, StartDateTime : DateTime, EndDateTime : DateTime, Make_Available_to_Students : bool) : bool + ManageRecording() + ToggleToDoList(todo_Item : Dictionary<String, Bool>) : void

Entity **Chat**
+ User_Code : int (Sender) + User_Code : int (Reciever) # Messages : list<Message>
+ ViewMessages() : list<Message> + AddNewChat(User_ID : int) : void

Entity **Message**
Message_ID : int # Body : string # Sent_DateTime : datetime # Attachment : attachment (Optional)
+ SendMessage(body : string, Attachement : attachment) + DeleteMessage(Message_ID: int) : bool

Entity **Badge**
+ Badge_ID : int # Name : string # Requirement : string + Image_ID : image
+ Badges(Student_Number : int) : list<Badge> We acknowledge that lecturers should be able to add, edit and delete badges that can be awarded to students, but for the purposes of this project we will not implement this feature.

Entity **Tutor Application**
+ Application_ID : int # Student_Number : int # Module_Code : int # Description : string # Supporting_Document : attachment

Student:

```
+ ApplyToBeTutor(Module_Code : int, Student_Number :
  int, Description : string, Supporting_Document :
  Upload) : bool
```

Lecturer:

```
+ RemoveTutor(Tutor_Student_Number : int) : bool
+ ManageApplication(Application_ID : int, status : enum)
  : bool
+ DetermineMarkObtained(Student_Number : int,
  Module_Code : int) : double
```

Entity **Tutor Report**
+ Report_ID : int + Student_Number : int (Reported) + Student_Number : int (Reportee *if not anonymous) # Description : string # Supporting_Document : attachment # Remain_Anonymous : bool
Student: + ReportTutor(Tutor_Student_Number : int, Description : string, Supporting_Document : Upload, Remain_Anonymous : bool) : bool Lecturer: + ViewReport(Report_ID : int) : void + MessageReportee(Report_ID: int, Reportee_Student_Number : int) : void + SubmitComment(Report_ID: int, Description : string) : bool + EditReport(Report_ID: int, status : enum) : bool

Entity **Leaderboard**
Student_ID : int # Number_of_Points : int # Ranking : int # Current_View : int
+ ShowCurrentRank(Student_ID : int) : int + ChangeView(currentView : enum) : enum

Control **Code Problems**
Problem_ID : int # Category : string # Description : string # Constraints : string # Output : string
+ GradeProblem(Student_Number : int, Problem_ID : int) : double + CompileProblem() : void + ProvideFeedback() : string We acknowledge that a lecturer/tutor should be able to add, edit and delete problems but for the purposes of this assignment problems were added by an admin user and are therefore not included.

Control **Statistics**
+ User_ID : int + Percentage : double + Colour : string
+ calculatePercentage(markobtained : int, total : int): double + CalculateColour(Percentage : double) : colour Student: + ViewStatistics(Student_Number : int) : Average_Attendance : double, Average_Grade : double Tutor: + ViewStatistics(Student_Number : int) : Average_Rating : double

Psuedocode

1. Method for the Lecturer or Tutor to Add a Quiz:

```
bool AddQuiz(string name, int minutes, enum Difficulty)
{
   if (Lecturer or Tutor)
   {
      try
        set quiz_Name equal to name;
        set quiz_Difficulty equal to Difficulty;
        instantiate new Timer with duration equal to minutes;
        instantiate new list of Question;
        return true;
      else
        return false;
   }
}
```

2. Method for the Lecturer or Tutor to Edit a Quiz:

```
bool EditQuiz(int quiz_ID, string new_name, int new_minutes, enum
              new_difficulty)
{
   if (Lecturer or Tutor)
   {
      try
        find quiz using quiz_ID;
        set quiz_Name equal to name;
        set quiz_Difficulty equal to new_difficulty;
        change Timer duration to new_minutes;
        return true;
      else
        return false;
   }
}
```

3. Method for the Lecturer or Tutor to Delete a Quiz:

```
bool DeleteQuiz(int quiz_ID)
{
   if (Lecturer or Tutor)
   {
      try
        find quiz using quiz_ID;
        delete quiz;
        return true;
      else
        return false;
   }
}
```

Interface Layer and Data Access Layer

The details of how the business objects link with the Interface Layer and Data Access Layer are discussed in length in Chapter 1: Project Description, with specific reference to Scope and Components and Features. Therefore, there is no need to discuss these details in a separate section.

Conclusion

The design artifacts presented in the Design Document serve as the framework for the development of the Peer-to-Peer Tutoring system. The Design document provides a systematic and well-documented approach to development, thus aiding developers in implementing the system effectively. With these artifacts, developers will be able to build a robust and functional software solution that aligns with the project objectives.

Blackboard Tutor®

Appendix A: Overview of development effort

Project Manager, BC Smit, 2020053969

Lead Developer, K Kolanchu, 2020447641

Developer, SQA Manager C Venter, 2019659602

UX Designer, Developer J Smith, 2020046385

Overview of development effort

Requirements Workflow

Firstly, the requirement teams met almost weekly with the University of the Free state Department of Computer Science and Informatics representative (Rouxan Fouche) to discuss the client's needs. The feedback that was provided was incorporated as the development continued.

Analysis Workflow

The analysis team iteratively and incrementally reviewed the requirements. Redundancies were defined and were compiled to streamline development efforts and to improve user experience. The user interface was designed by J Smith. She met regularly with her other team members and the client to improve the user interface.

Thereafter all team members develop use-cases, including descriptions, scenarios, and step-by-steps. Here the iterative-and-incremental modal was crucial. This section took a lot of time as this would provide the basis for the rest of the project.

Design Workflow

During this phase, noun extraction took place. BC Smit and J Smith complied a concise product description. They identified the nouns and extracted the relevant classes. Class-Responsibility-Collaboration (CRC) cards were developed for each class including the responsibilities, collaborators, attributes, and relationships for each class. Using this information, the Unified Modelling Language diagram was drawn.

During this time C Venter used the best-case scenarios of each use-case to draw sequence diagrams.

Implementation Workflow

This phase will not take place for the purposes of this project. However, was planned for in detail in the Software Management Plan.

Testing Workflow

This phase take place continuously throughout the other phases.

Reflection

The scope of the project was quite a bit larger than the team anticipated. We believe that we were caught a little off guard by the amount of work that needed to be completed. As students did not have any real experience designing software. While this is was an issue, we were able to complete the project in time and we are very proud of the work we have produced. We believe that the experience we gained during the completion of this project will help us greatly when we work on real-life software projects in the future.

One of our biggest challenges was meeting regularly. Because of our busy schedules, we couldn't meet with each other as frequently as we wanted to. Finding a time when we were all available to meet our client was difficult as well. The above-mentioned challenges meant that we had integration issues at the end. If we do something similar in the future, we will meet more regularly.

We acknowledge that meeting more regularly would not have eliminated integration issues. We realized that when we divided the work amongst ourselves and completed them separately, that every person had their own ideas about how their finished work should look like. If we do something similar in the future, we should create a "style guide". The style guide should contain an example of what the finished product should look like including font size, style, spacing etc.

While the project was a quite difficult for all of us, we were able to stick together as a team and push through. Having completed the project, all of us can confidently say that we learned a lot. We learned that coding is a just a small part of what goes into software development. Furthermore, this project taught us a lot about working in a team and how to approach large scale projects.

Blackboard Tutor®

Appendix B: Hours Spent

Project Manager, BC Smit, 2020053969

Lead Developer, K Kolanchu, 2020447641

Developer, SQA Manager C Venter, 2019659602

UX Designer, Developer J Smith, 2020046385

Hours Spent

Find below a summary of the average number of hours spent by the team developing the product:

Workflows	Hours Spent	Total
Requirements	J Smith: 17 hours BC Smit: 20 hours C Venter: 23 hours K Kolanchu: 18 hours	80 hours
Analysis	J Smith: 40 hours BC Smit: 45 hours C Venter: 44 hours K Kolanchu: 47 hours	160 hours
Design	J Smith: 103 hours BC Smit: 95 hours C Venter: 93 hours K Kolanchu: 95 hours	400 hours
Implementation	Since this workflow has not yet been completed, we estimate an equal hourly breakdown for each person. J Smith: 600 hours BC Smit: 600 hours C Venter: 600 hours K Kolanchu: 600 hours	Estimated ± 2400 hours
Testing	Since this workflow has not yet been completed, we estimate an equal hourly breakdown for each person. J Smith: 800 hours BC Smit: 800 hours C Venter: 800 hours K Kolanchu: 800 hours	Estimated ± 3200 hours
Grand Total		**6240 hours**