Orange Coast College

CS A220: Software Engineering

Student Record Management System

3rd Sprint

Group Name: The Uprising

Group Members: Harry Nguyen, Anh Vuong, Phu Nguyen, Khang Nguyen

Submission Date: May 23, 2021

Instructor: Sara Ghadami

Table of Contents

Document Revision History Table

REVISION HISTORY				
Rev.	**Description of Change**	**Page No.**	**Author**	**Date**
0	Created one-page summary of Project Plan	4	All	02/21/2021
1	Created 5 functional User Stories	5	All	03/06/2021
2	Added 5 non-functional User Stories Created Use-Case (textual) descriptions Created Use-Case (diagram) descriptions	6 7 – 19 20	All	03/15/2021
3	Updated Use-Case (textual) descriptions Updated Use-Case (diagram) descriptions Added Staging/ Grooming Added Development Process Created User Manual	7 – 19 20 22 23 – 24 25 – 31	All	03/18/2021
4	Added Use Case Diagram Added Sprint Backlog Updated Staging/ Grooming Updated Development Process Updated User Manual	20 21 22 23 – 24 25 – 31	All	03/20/2021
5	Updated User Stories Updated Use Case Textual Updated Use Case Diagram	5 – 7 8 – 29 30	All	04/04/2021
6	Added more functional User Stories Added more Use Case Textual Updated Use Case Diagram Updated Sprint Backlog	5 – 6 8 – 29 30 31 – 33	All	04/11/2021
7	Added Class Diagram Added CRC Cards Added 5 Sequence Diagrams Updated User Manual	37 38 – 40 41 – 43 44 – 58	All	04/19/2021
8	Updated Staging/ Grooming Updated Development Process Updated Sequence Diagrams	34 35 – 36 41 – 43	All	04/20/2021
9	Updated Project Plan Added 5 more User Stories Added 5 more Use Case Textual Updated Use Case Diagram	4 8 – 9 11 – 43 44	All	05/8/2021
10	Updated functional User Stories Updated Use Case Textual Updated Sprint Backlog Updated Development Process Updated Class Diagram	8 – 9 11 – 43 45 – 49 51 – 53 54	All	05/16/2021

11	Updated CRC Cards Added 10 Test Cases Added 5 State Diagrams Updated User Manual	55 – 60 61 – 75 76 – 77 81 – 109	All	05/19/2021
12	Updated Revision History Updated functional User Stories Updated Use Case Textual Updated Use Case Diagram Updated Test Cases Updated State Diagrams Updated User Manual	2 – 3 8 – 9 11 – 43 44 61 – 75 76 – 77 81 – 109	All	05/22/2019

Project Plan

- What kind of project is that?

  A program that can query, interact, and manage students' records in the database.

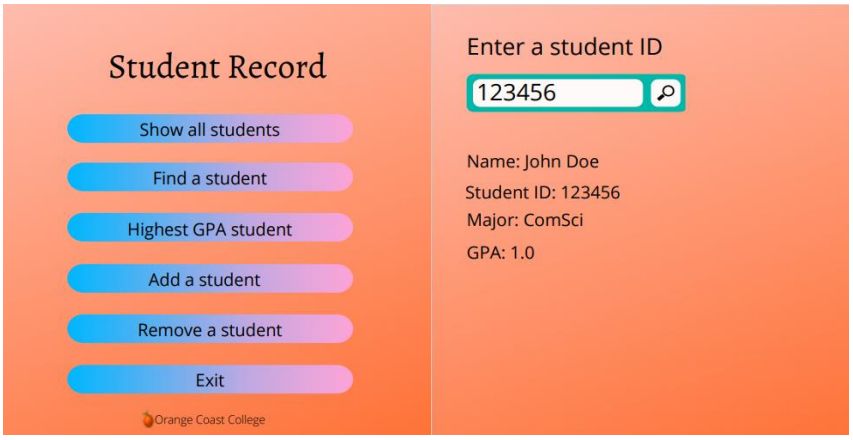
- What is the programming language you need to know?

  Java.

- How is the GUI? (sketches)

  Interactive and straightforward enough for users to interact and use.

  A mockup GUI is temporarily created and will be changed later in the project.



- Why this project?

  This project helps the users efficiently query and manage the students' database with the help of an intuitive and user-friendly GUI.

- What are the main features needed during the first phase? (we have 3 phases)

  Brainstorming the ideas and features that need to be implemented.

  Creating topics and writing up the documentation.

  Considering the user's experience to make a friendly mockup GUI then improving the system gradually.

- Who will the users be?

Database administrators and office use.

- What is the goal of this project?

  To create a system that organizes a student database and helps the users easily access and manage the student records.

- What are the changes on the planned project since 2nd iteration?

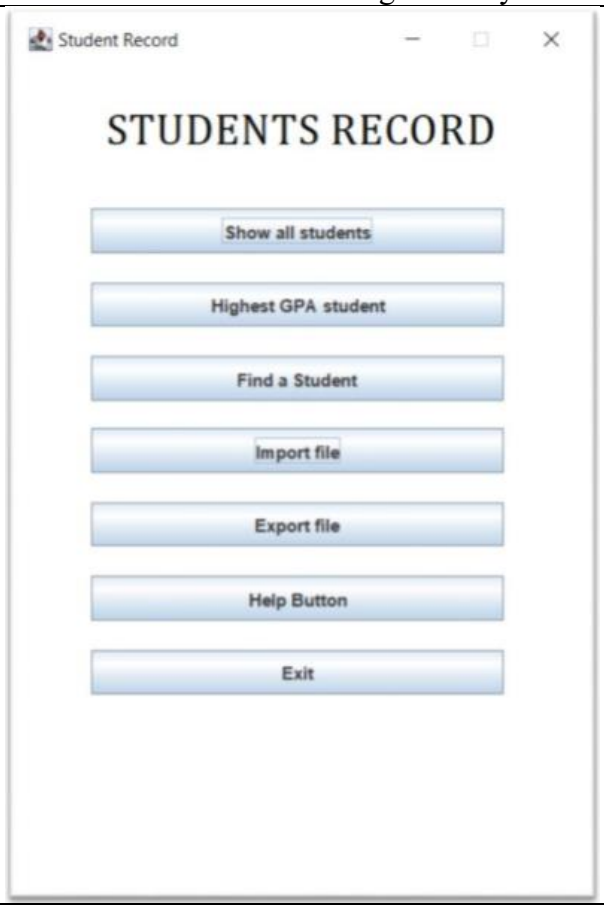
  From the beginning, this project was planned to query and interact with the student database by using only simple functions such as "Show all students," "Find a student/Find the highest GPA students," and "Add/Remove a student." Then, when the instructions of the first sprint phase regarding user stories, use cases, and other essential steps were released, a few creative thoughts came out. The specific management functions such as importing and exporting data were mentioned in the first phase.

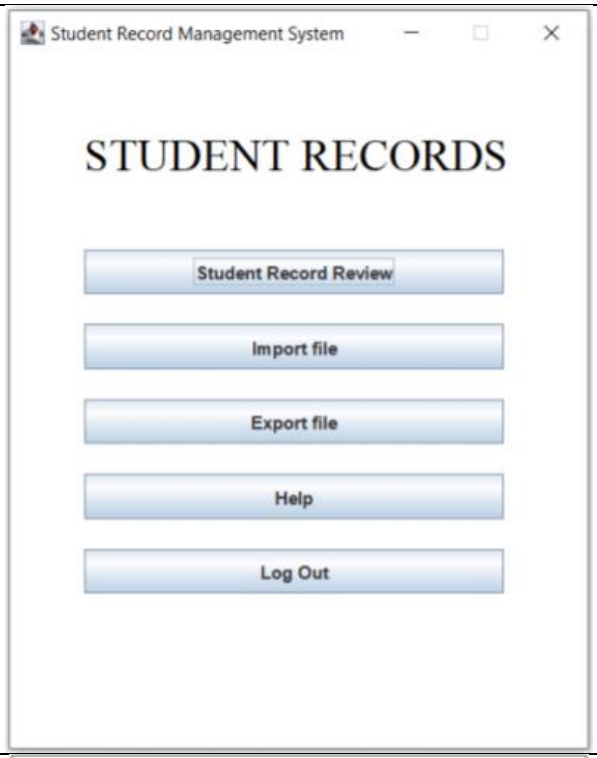
  After the 1st iteration, more useful functions were created to help the users query, interact, and manage flexibly the student record system. For instance, the functions of reviewing the student records as well as adding and finding a student were put together inside the "Student Record Review" window for the intuitive and straightforward usage of the users. The "Sort Student List" function was also added in this part. Moreover, in this 2nd iteration, the security functions, "Log in/out" and "Reset password," were intentionally created to keep the privacy of the system's database from strangers.

  The process of implementing the 3rd sprint phase became clearer after 15 unique functional user stories were determined. Since the 2nd sprint, to enhance the security of the system, a "Sign up" button and a "Registration Form" have been added for the new users to register their accounts before they can use the system. In the "Student Record Review" window, a "Filter Student List" button was also added so that the users can

manipulate the table containing the list of student data based on the filter's field and value. Moreover, more innovative features such as submitting a help form, accessing the activity log, or viewing system announcements were created to allow users to query, control, and approach the system quickly and easily.

The changes of the functions and the GUI evolution throughout 3 phases can be seen clearly in the following main-menu interfaces of "Student Record Management System":

The GUI evolution of "Student Record Management System"	
1st iteration	

2nd iteration	
3rd iteration	

User Stories

Functional

1. **As** a user

   I need a login button

   So that I can keep the privacy of the system's database.

2. **As** a user

   I need a sign-up button

   So that I can register an account to start using the system.

3. **As a** registered user

   I want to reset my password

   So that I can create a new password in case I forget my old password.

4. **As** a user

   I need to review the records of the students

   So that I can view or modify their data.

5. **As** a user

   I want to add a student record manually

   So that I can add student data into the database without having to import a file.

6. **As** a user

   I want to be able to find a student by using his/her student's ID or name

   So that I can view the queried student’s record.

7. **As** a user

   I need to sort the database

   So that I can have an overview of the data based on the sort methods.

8. **As** a user

I need to filter the database

So that I can have a reduced list of the data based on the filter's field and value.

9. **As** a user

   I want to import the file containing the data of the students

   So that I can save a large amount of time and effort for inputting data.

10. **As** a user

    I want to export the file containing the data of the students

    So that I can conveniently transfer data or help with statistics based on some criteria.

11. **As** a user

    I need an activity log

    So that I can manage the activities of all users in the system.

12. **As** a user

    I need to view the system announcements

    So that I can acknowledge the system's issues or maintenance events.

13. **As** a user

    I need a "Guidelines" button

    So that I can learn more about the system's functionality.

14. **As** a user

    I need to submit a help form

    So that I can report the system's issues to the admin.

15. **As** a user

    I need a log-out button

    So that I can prevent others from accessing the system without beforehand permissions.

Non-Functional

1. **As** a user

   I want to access the students' data within seconds

   So that I can improve the efficiency of working time.

2. **As** a developer

   I need to know all the tasks in advance

   So that I can make a plan in time properly.

3. **As** a developer

   I want the system to be easily maintainable

   So that it can be upgraded with more features in the future.

4. **As** a user

   I want the system to be fault-tolerable

   So that it will not crash when there is a faulty input, but instead giving an error announcement.

5. **As** a user

   I need to be able to access the system offline by using the computers in the school’s office

   So that I can interact with the system without the need of the Internet or in an emergency case.

Use Case

Use Case Textual

Use Case: Login

ID: UC-01

Description:

A login button to let the user enter the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully enters the login info and clicks the “Sign in” button.

Post conditions:

Success end condition

User logs in the system successfully.

Failure end condition

User cannot log in the system.

Trigger

User enters the login info and clicks the “Sign in” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the “Sign in” button.

5. System confirms the login info with the saved login details and lets the user log in the system.

Extensions

5a. In step 5, if the user enters the wrong login info:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

Use Case: Sign up

ID: UC-02

Description:

A “Sign up” button to let the user create an account to begin using the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks the “Sign up” button.

Post conditions:

Success end condition

The user successfully creates an account in the system.

Failure end condition

User is unable to create an account in the system.

Trigger

User clicks the “Sign up” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User clicks the “Sign up” button.
4. The system displays the registration form.
5. User fills in the required information.
6. User clicks the “Register” button.
7. System records the new account of the user.

Extensions

5a. In step 5, if the user wants to cancel creating an account:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 2.

6a. In step 6, if the user does not fill in all the required information but clicks the “Register” button:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 5.

6b. In step 6, if the confirm password does not match with the password that user inputted, but user clicks the “Register” button:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 5.

Use Case: Reset password

ID: UC-03

Description:

A “Forgot password” button to verify the user’s security info before letting the user reset the password.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks the “Forgot password” button.

Post conditions:

Success end condition

The user successfully changes the password.

Failure end condition

User is unable to reset the password to get a new one.

Trigger

User clicks the “Forgot password” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User clicks the “Forgot Password?” button.
4. A “Forgot Your Password?” dialog pops up.

5. User fills in the username, then chooses the security question and its answer that user set up in the past.
6. User clicks the “Submit” button.
7. A “Reset password” window appears to let user type in a new password and retype it to confirm.
8. User clicks the “Submit” button.
9. System records the new password of the user.

Extensions

5a. In step 5, if the user remembers the password:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 2.

6a. In step 6, if the inputted security info is incorrect:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 5.

7a. In step 7, if the user remembers the password:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 2.

8a. In step 8, if the password and the confirm password do not match:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 7.

Use Case: Student record review.

ID: UC-04

Description:

A “Student Record Review” button to show a table of all the student records so that the user can view the student records visually.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Student Record Review’’ button.

Post conditions:

Success end condition

The students’ database is successfully loaded and displayed on the screen by the system.

Failure end condition

The system is unable to display the students’ information on the screen.

User is unable to see the student records.

Trigger

User logs in the system and clicks the “Student Record Review’’ button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.

5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Student Record Review” button.
8. A “Student Record Review” window appears and displays a table of all the student’s information on the screen.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

Use Case: Add student

ID: UC-05

Description:

An “Add a student” button to let the user manually add the data of students into the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Student Record Review” button.

User successfully clicks the “Add a student” button.

Post conditions:

Success end condition

User successfully adds the data of the students into the system.

Failure end condition

User did not fill in all the required data of the student.

User did not click the “Register” button after filling in the student’s data.

No new student’s data is added to the system.

Trigger

User logs in the system and clicks the “Student Record Review” button first, then clicks the “Add a student” button.

Main Success Scenario

1. User launches the system.

2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Student Record Review” button.
8. A “Student Record Review” window appears and displays a table of all the students’ information on the screen.
9. User clicks the “Add a student” button.
10. A “Student Registration Form” dialog appears.
11. User fills in all required information.
12. User clicks the “Register” button.
13. A “Student Information” dialog will pop up to let user double-check the data of the student.
14. User clicks “Yes” button to add the data of the student to the system.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

10a. In step 10, if the user decides to cancel adding a student’s data:

1. User clicks “Cancel” button or just closes the window.

2. The use case resumes on step 8.

12a. In step 12, if the user does not fill in all the required data boxes:

1. The system displayed a warning message.

2. User clicks “OK” button or just closes the window.

3. The use case resumes on step 11.

13a. In step 13, if user finds out that the inputted data is incorrect, or user decides to cancel adding the student’s data:

1. User clicks “Cancel” button or just closes the window.

2. The system will display a “Data unsaved” message.

3. User clicks “OK” button or just closes the window.

4. The use case resumes on step 11.

Use Case: Find student

ID: UC-06

Description:

A “Find a student” button to let the user search for a student in the database.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Student Record Review’’ button.

User successfully clicks the “Find a student’’ button.

Post conditions:

Success end condition

The system successfully searches in the database and displays the queried student’s information on the screen.

Failure end condition

The system is unable to find the student’s ID or name in the database.

User is unable to see the queried student’s information.

A warning message that the student cannot be found is displayed on the screen.

Trigger

User logs in the system and clicks the “Student Record Review” button first, then clicks the “Find a student’’ button.

Main Success Scenario

1. User launches the system.

2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Student Record Review’’ button.
8. A “Student Record Review” window appears and displays a table of all the students’ information on the screen.
9. User clicks the “Find a student’’ button.
10. An “Input” window appears on the screen.
11. User inputs the student’s ID or name.
12. User clicks “OK” button.
13. System searches through the database.
14. System displays the queried student’ information on the screen.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

10a. In step 10, if the user decides to cancel finding a student:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 8.

13a. In step 13, if the system cannot find the inputted student's ID or name:

1. The system will display a warning message.
2. User clicks "OK" button or just closes the window.
3. Use case resumes on step 8.

Use Case: Sort data

ID: UC-07

Description:

A “Sort Student List” button to sort the students’ data based on some fixed criteria.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Student Record Review” button.

User successfully clicks the “Sort Student List” button.

Post conditions:

Success end condition

The system successfully displays the students’ information in the requested order.

Failure end condition

The order of students’ information is not displayed correctly per user’s request.

Trigger

User logs in the system and clicks the “Student Record Review” button first, then clicks the “Sort Student List” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.

5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Student Record Review” button.
8. A “Student Record Review” window appears and displays a table of all the students’ information on the screen.
9. User clicks the “Sort Student List” button.
10. A “Sort Options” dialog will appear on the screen.
11. User chooses the student’s data column to sort.
12. User chooses the type of sort which depends on the value data type of the selected column.
13. User clicks “OK” button to confirm.
14. System displays the students’ information in the requested sort order.

Extensions

5a. In step 5, if the user enters the wrong login info:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

10a. In step 10, if the user decides to cancel sorting:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 8.

Use Case: Filter data

ID: UC-08

Description:

A “Filter Student List” button to filter the students’ data based on the filter’s field and value.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Student Record Review” button.

User successfully clicks the “Filter Student List” button.

Post conditions:

Success end condition

The system successfully displays the reduced list of students’ information based on the filter’s field and value.

Failure end condition

The students’ information is not displayed correctly per user’s filter’s field and value.

Trigger

User logs in the system and clicks the “Student Record Review” button first, then clicks the “Filter Student List” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.

3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Student Record Review” button.
8. A “Student Record Review” window appears and displays a table of all the students’ information on the screen.
9. User clicks the “Filter Student List” button.
10. A “Filter Options” dialog will appear on the screen.
11. User chooses the student’s data column to filter.
12. User enters the filter value.
13. User clicks “OK” button to confirm.
14. System displays a reduced list of students’ information per user’s filter field and value.

Extensions

5a. In step 5, if the user enters the wrong login info:

1. The system will display a warning message.
2. Use case resumes on step 3.

10a. In step 10, if the user decides to cancel filtering:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 8.

13a. In step 13, if the user forgets to choose the student’s data column or value to filter:

1. The system will display a warning message.

2. User clicks “OK” button or just closes the window.

3. Use case resumes on step 10.

Use Case: Import file

ID: UC-09

Description:

An “Import file” button to let the user import the file containing the data of students into the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Import file” button.

Post conditions:

Success end condition

User successfully imports the file containing the data of students into the system.

Failure end condition

No file is imported into the system.

Trigger

User logs in the system and clicks the “Import file” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.

5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Import file” button.
8. The import file window will appear on the screen.
9. User clicks on “Look in” to find the location of the file needed.
10. User selects the appropriate data file, then clicks the “Open” button.
11. System displays an info window with the selected filename.
12. User clicks the “Open” button.
13. Data of the students in the file will be added into the system.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

9a. In step 9, if the user decides to cancel importing a student:

1. User will click “Cancel” button or just close the window.
2. The use case resumes on step 6.

10a. In step 10, if the user chooses an inappropriate file:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. The use case resumes on step 9.

Use Case: Export file

ID: UC-10

Description:

An “Export file” button to back up the current database into a file and save it to a designated folder.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Export file” button.

Post conditions:

Success end condition

The system successfully backs up the current database into a file and saves them to a designated folder.

Failure end condition

The system is unable to back up the current database into a file to save it to a designated folder.

User cannot back up the database as he/she demands.

Trigger

User logs in the system and clicks the “Export file” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.

3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.
7. User clicks the “Export file” button.
8. The export file window will show up.
9. User clicks on “Look in” to find the location to save the file.
10. User types in the name of the backup file.
11. User clicks the “Save” button.
12. System backs up the database into a file and saves it to the designated folder.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

9a. In step 9, if the user decides to cancel exporting the database:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 6.

Use Case: Activity Log

ID: UC-11

Description:

An “Activity Log” button to let the user view the timeline of all the activities of the users in the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Activity Log” button.

Post conditions:

Success end condition

The system successfully displays the timeline of every activity in the system.

Failure end condition

User is unable to view the timeline of every activity in the system.

Trigger

User logs in the system and clicks the “Activity Log” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.

6. System displays a menu of functions.
7. User clicks the “Activity Log” button.
8. An “Activity Log” window appears and displays the timeline of every activity of the users on the screen.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

8a. In step 8, if the user wants to search specific activity logs:

1. User inputs a specific key word.
2. System display the activity log related to the inputted key word.

Use Case: Announcements

ID: UC-12

Description:

An “Announcements” button to let the user view the announcements from the admin about the system’s issues or the maintenance events.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Announcements” button.

Post conditions:

Success end condition

User successfully view the announcements from the admin.

Failure end condition

User is unable to view the announcements from the admin.

Trigger

User logs in the system and clicks the “Announcements” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.

6. System displays a menu of functions.
7. User clicks the “Announcements” button.
8. An “Announcements” window appears and displays a list of announcements from the admin regarding the system’s issues or the maintenance events.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

Use Case: Guidelines

ID: UC-13

Description:

A “Guidelines” button to guide the user through the functions in the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Guidelines” button.

Post conditions:

Success end condition

User is presented with the guidelines about each button’s functionality.

Failure end condition

No guidelines are displayed to the user.

Trigger

User logs in the system and clicks the “Guidelines” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.

6. System displays a menu of functions.
7. User clicks the “Guidelines” button.
8. System displays a “Guidelines” dialog about each button’s functionality.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

Use Case: Submit help form

ID: UC-14

Description:

A “Contact us” button to let the user report issues related to the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Contact Us” button.

Post conditions:

Success end condition

User successfully reports issues related to the system to the admin.

Failure end condition

User is unable to report issues related to the system to the admin.

Trigger

User logs in the system and clicks the “Contact Us” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.
6. System displays a menu of functions.

7. User clicks the “Contact Us” button.
8. A “Contact Form” window appears and displays a drop-down list of subjects and a message area to describe the issues in detail.
9. User chooses a subject and fills in the message area.
10. User clicks “Submit” button.
11. The system displays a message that the form has been successfully submitted.

Extensions

5a. In step 5, if the user enters the wrong login information:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

8a. In step 8, if the user wants to cancel the feedback:

1. User clicks “Cancel” button or just closes the window.
2. Use case resumes on step 6.

10a. In step 10, if the user does not choose a subject or fill in the message:

1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 8.

Use Case: Log out

ID: UC-15

Description:

A log out button to let the user log out and exit the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded and logged in by the user.

User successfully clicks the “Log out” button.

Post conditions:

Success end condition

User logs out and exits the system successfully.

Failure end condition

User cannot log out and exit the system.

Trigger

User logs in the system and clicks the “Log out” button.

Main Success Scenario

1. User launches the system.
2. Application displays the login window.
3. User types in the username and password.
4. User clicks the login button.
5. System confirms the login info with the saved login details and lets the user log in the system.

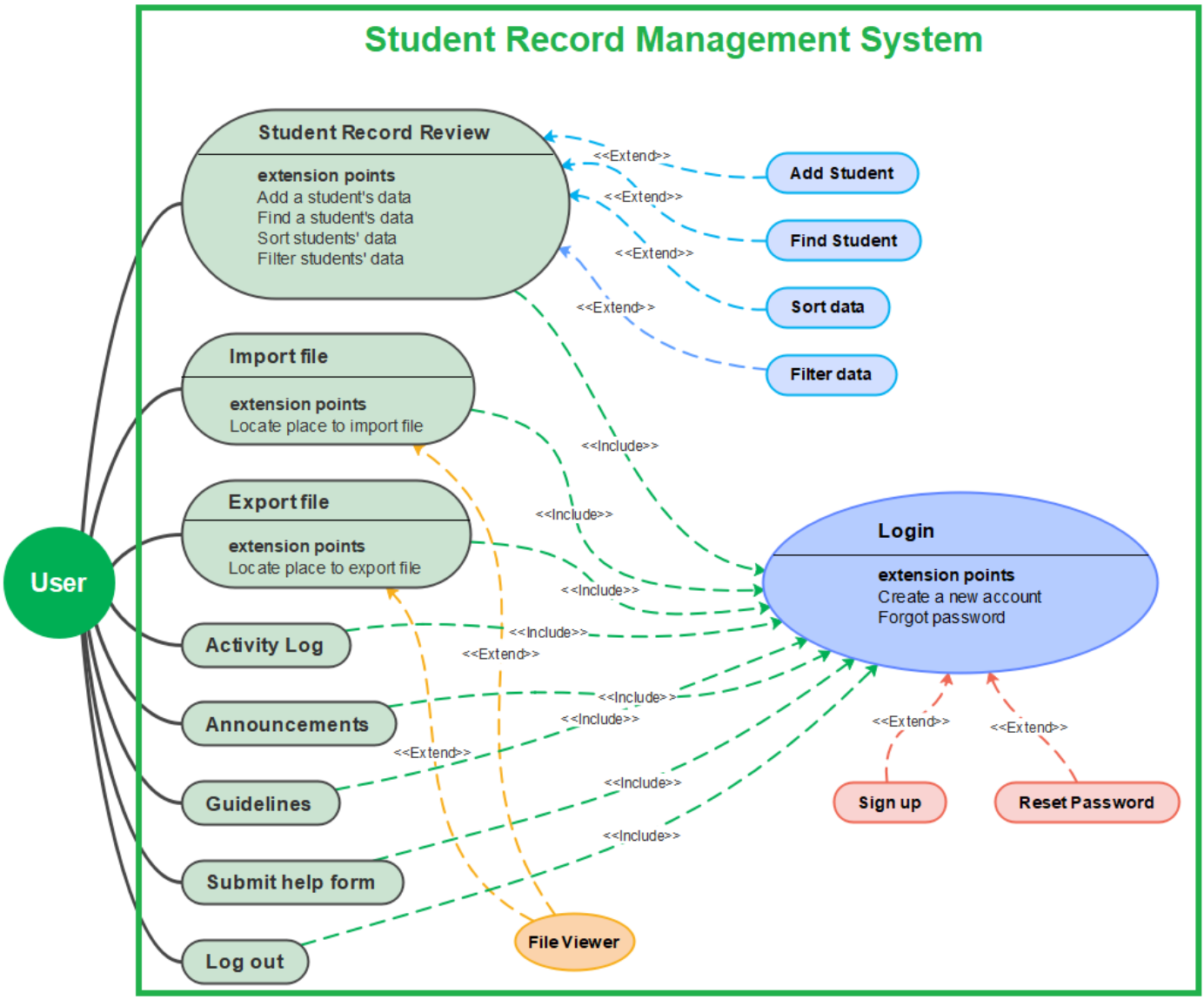
6. System displays a menu of functions.
7. User clicks the “Log out” button or just closes the window.
8. The user is logged out of the system, and the system is closed.

Extensions

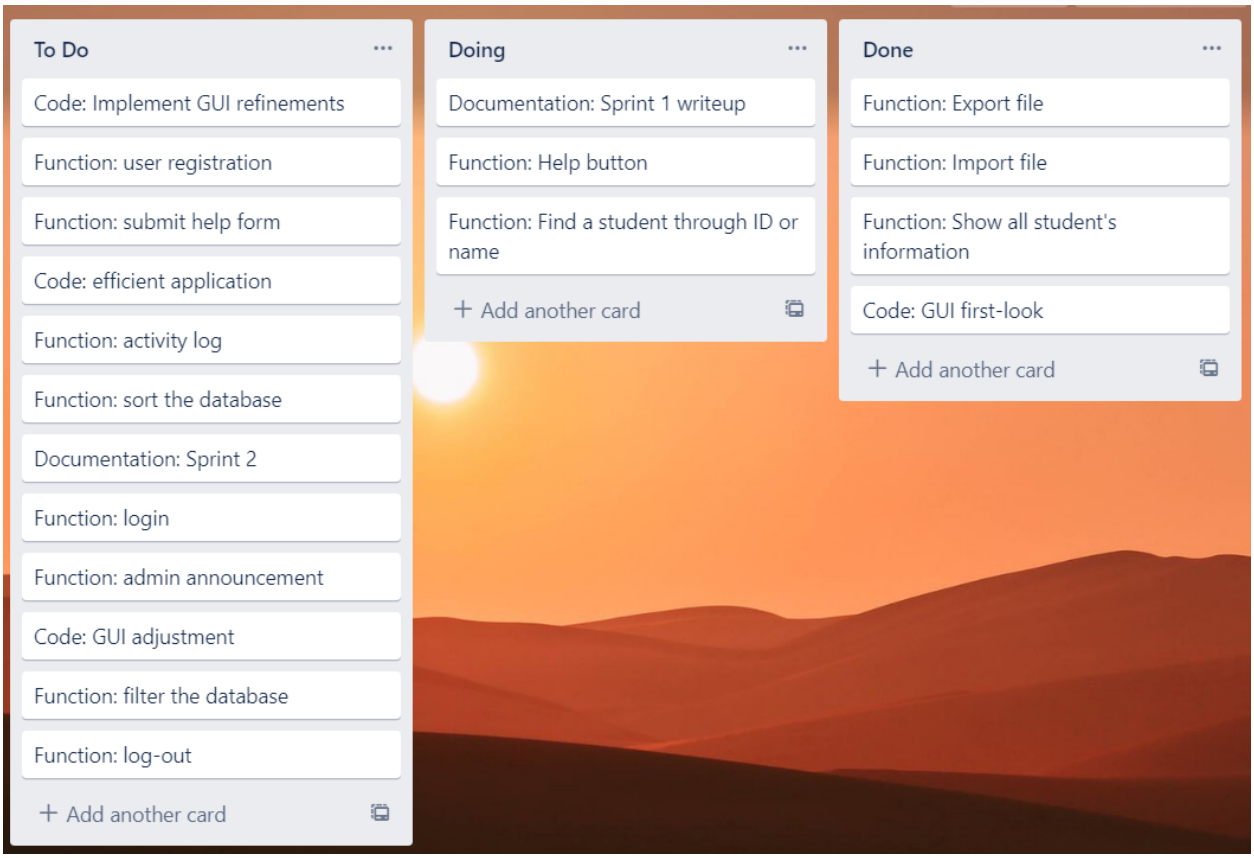
5a. In step 5, if the user enters the wrong login information:

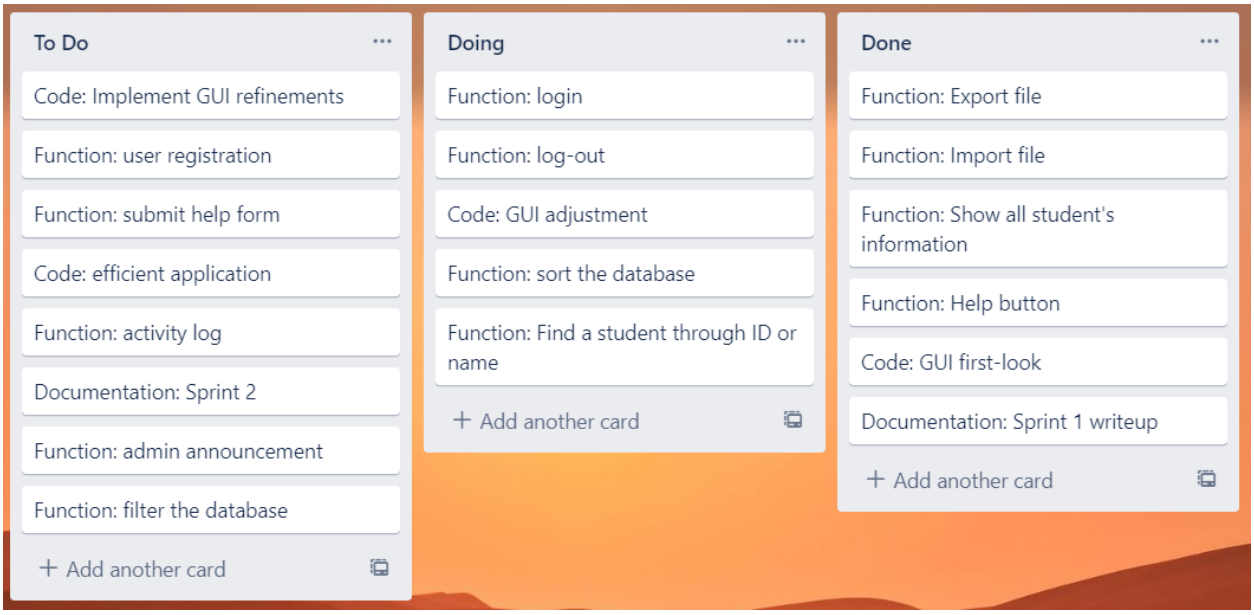
1. The system will display a warning message.
2. User clicks “OK” button or just closes the window.
3. Use case resumes on step 3.

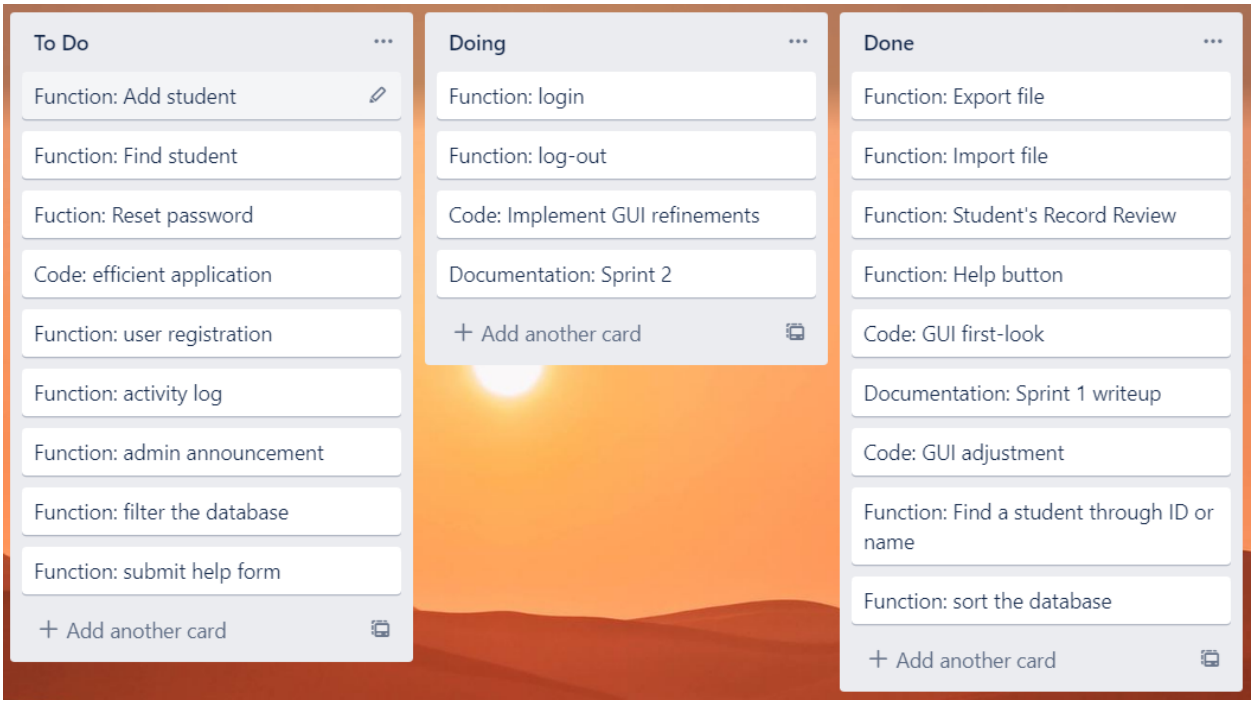
Use Case Diagram

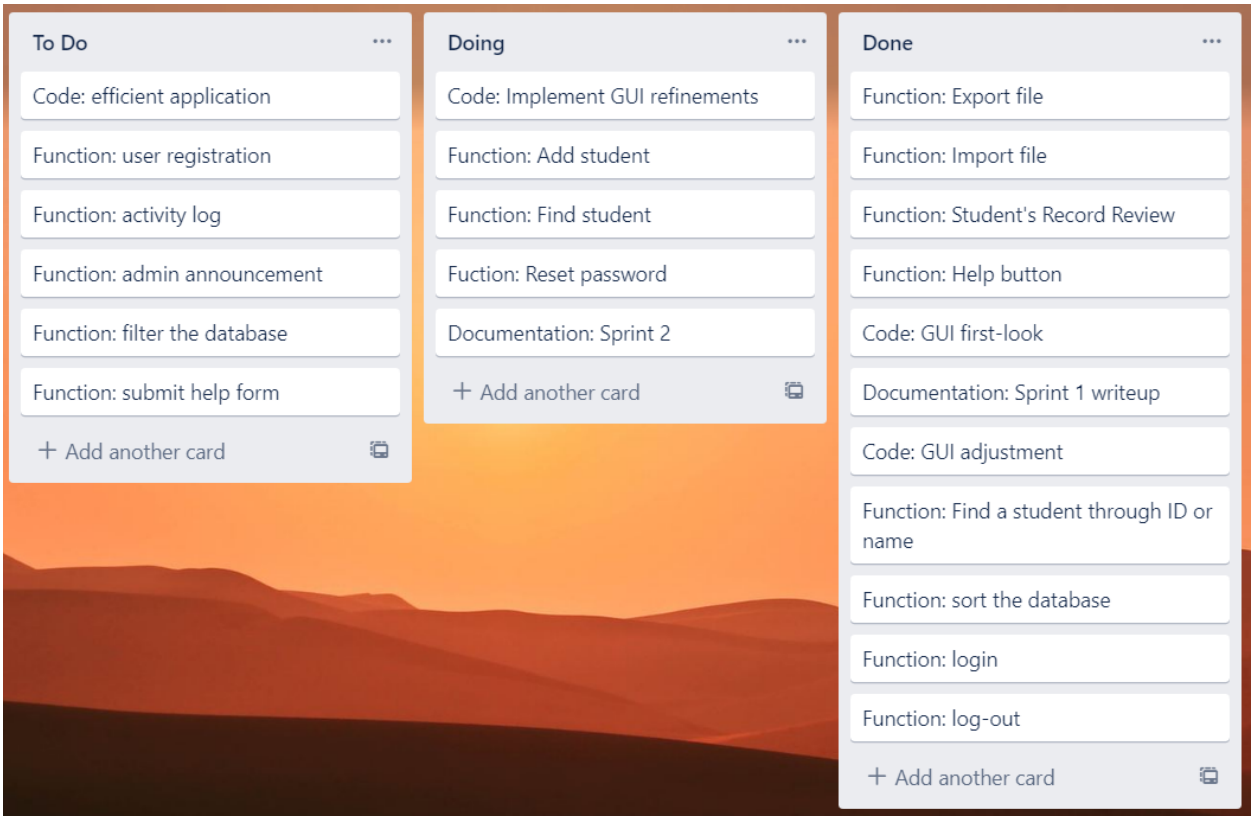


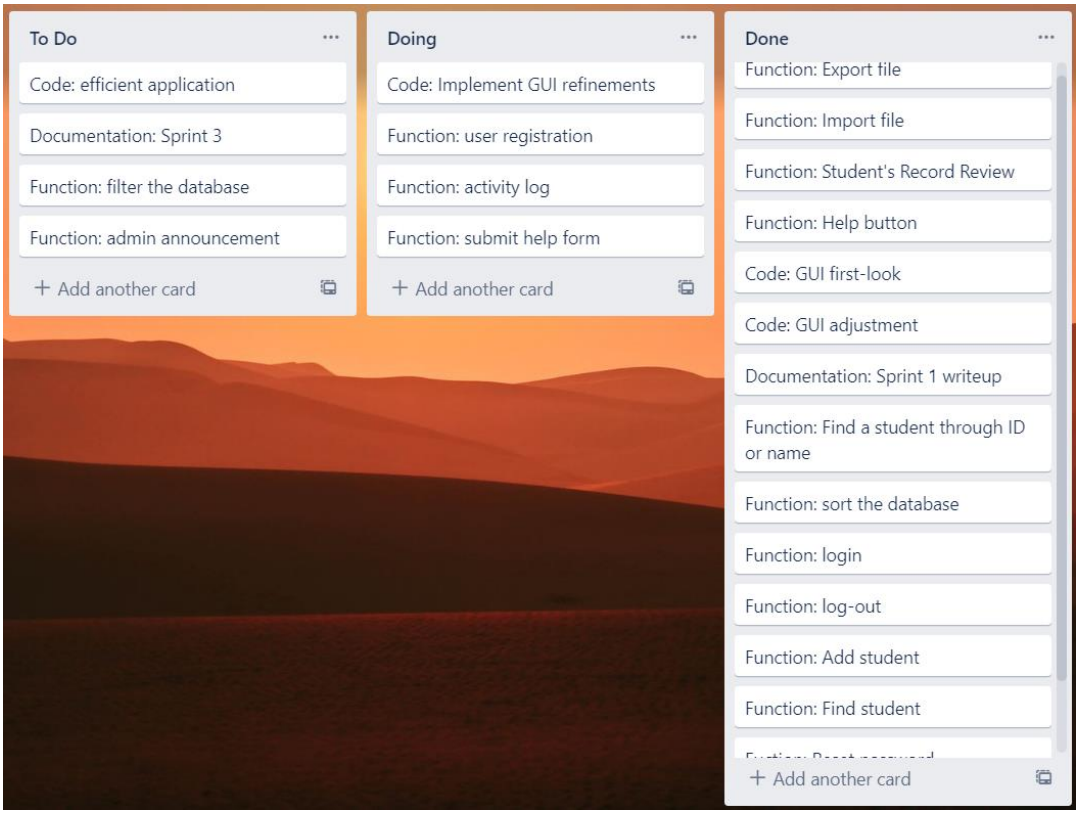
Sprint Backlog

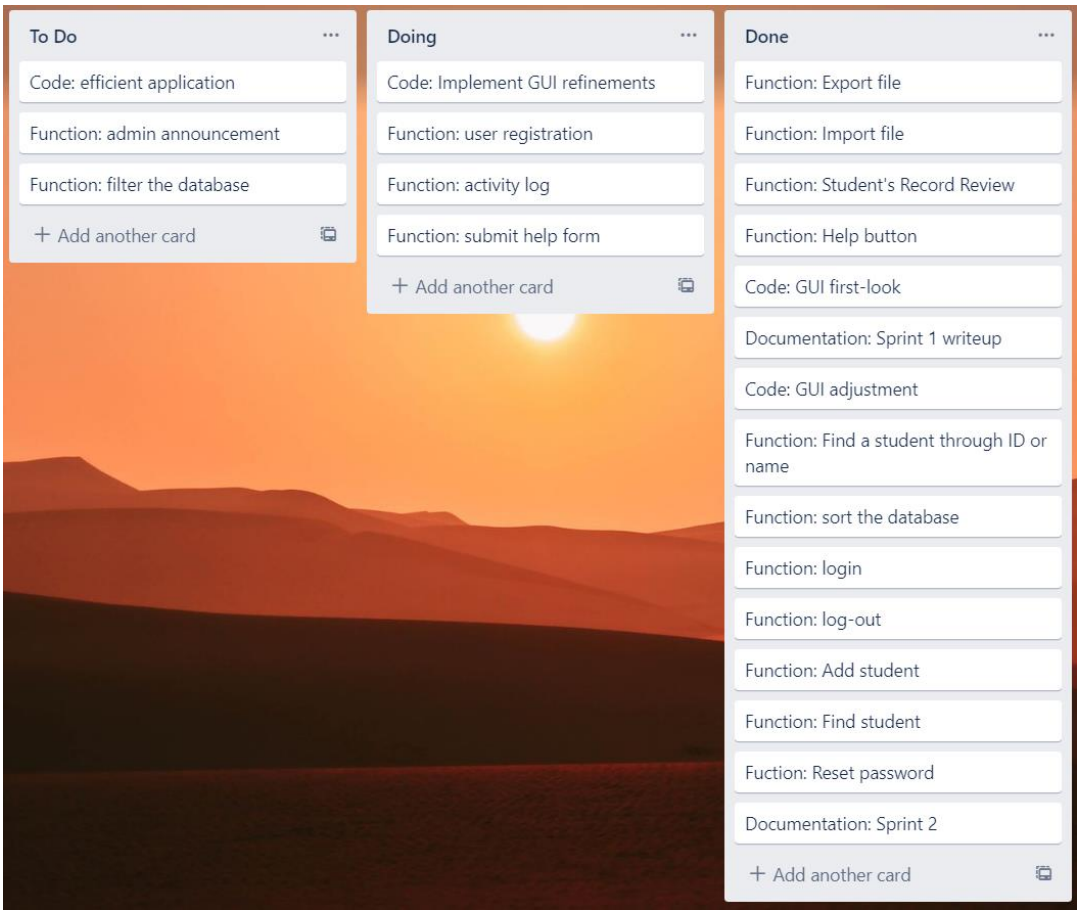


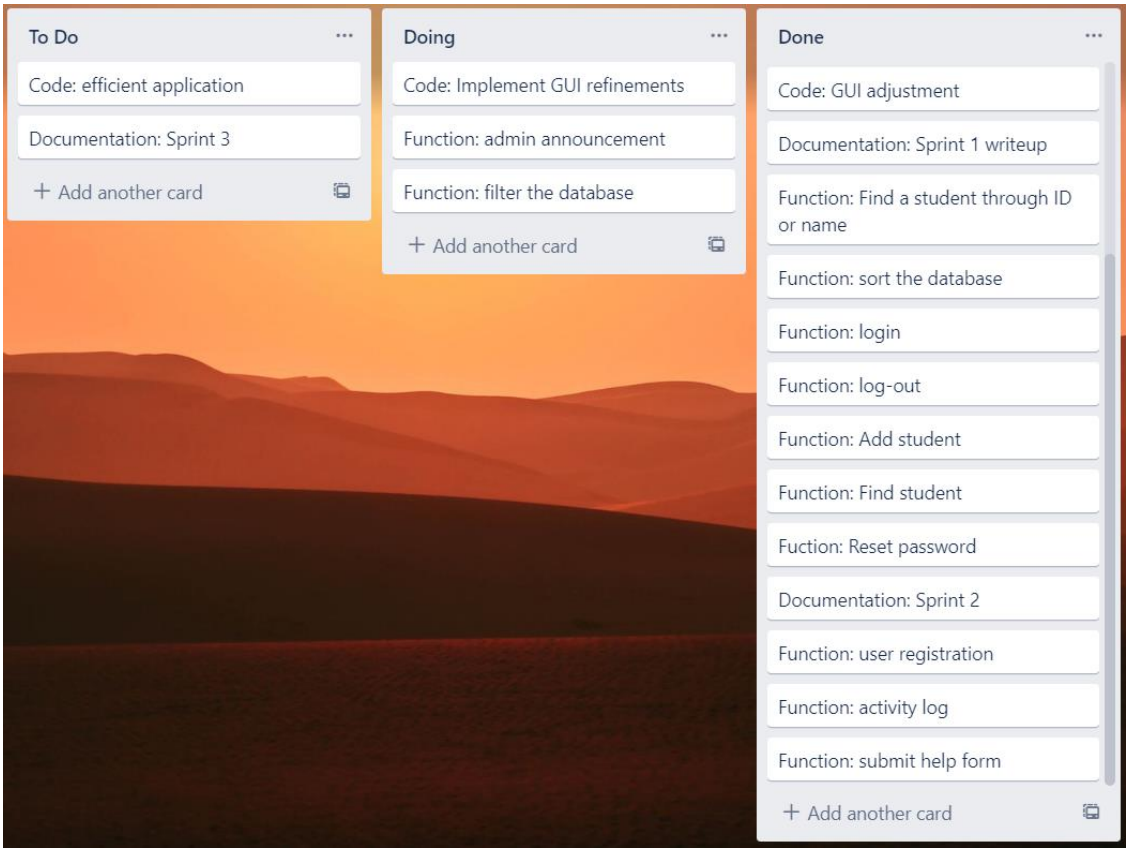


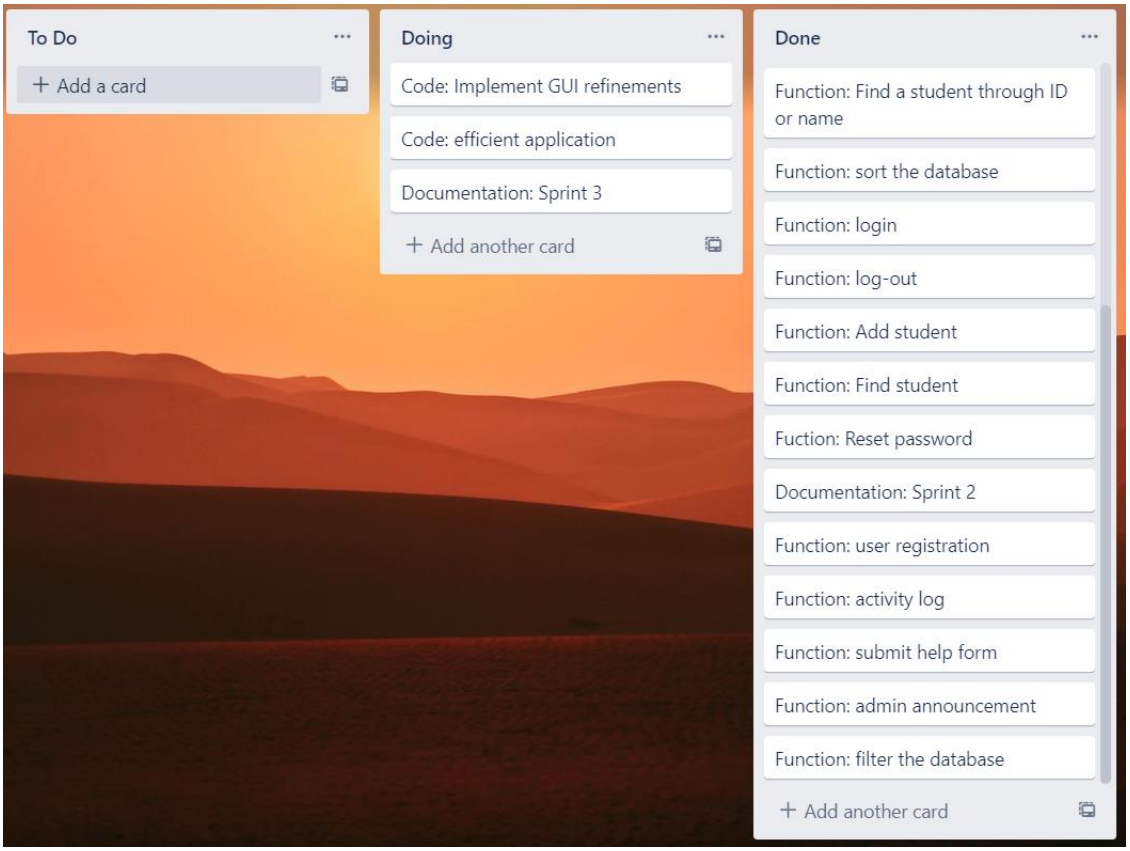

To Do
Function: Add student
Function: Find student
Fuction: Reset password
Code: efficient application
Function: user registration
Function: activity log
Function: admin announcement
Function: filter the database
Function: submit help form
Add another card
Doing
Function: login
Function: log-out
Code: Implement GUI refinements
Documentation: Sprint 2
Add another card
Done
Function: Export file
Function: Import file
Function: Student's Record Review
Function: Help button
Code: GUI first-look
Documentation: Sprint 1 writeup
Code: GUI adjustment
Function: Find a student through ID or name
Function: sort the database
Add another card

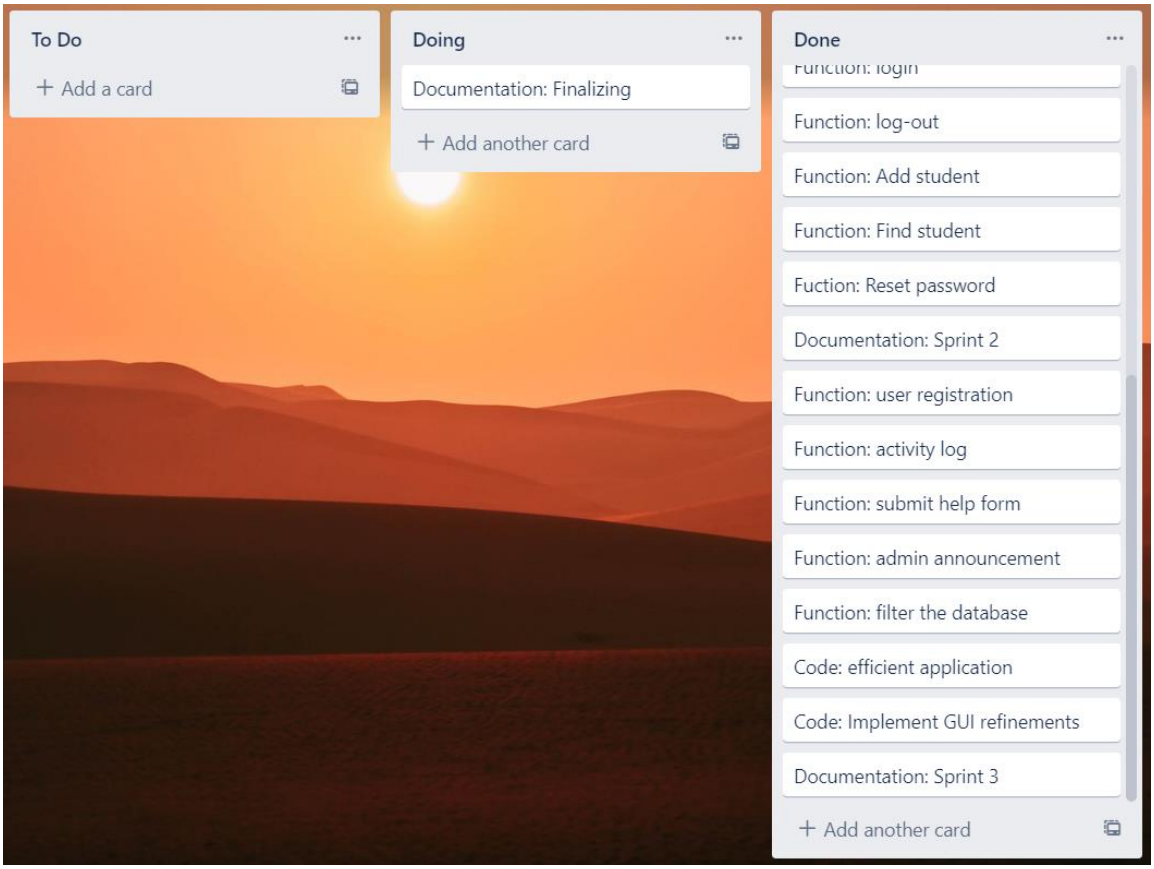

To Do
Code: efficient application
Function: user registration
Function: activity log
Function: admin announcement
Function: filter the database
Function: submit help form
Add another card
Doing
Code: Implement GUI refinements
Function: Add student
Function: Find student
Fuction: Reset password
Documentation: Sprint 2
Add another card
Done
Function: Export file
Function: Import file
Function: Student's Record Review
Function: Help button
Code: GUI first-look
Documentation: Sprint 1 writeup
Code: GUI adjustment
Function: Find a student through ID or name
Function: sort the database
Function: login
Function: log-out
Add another card

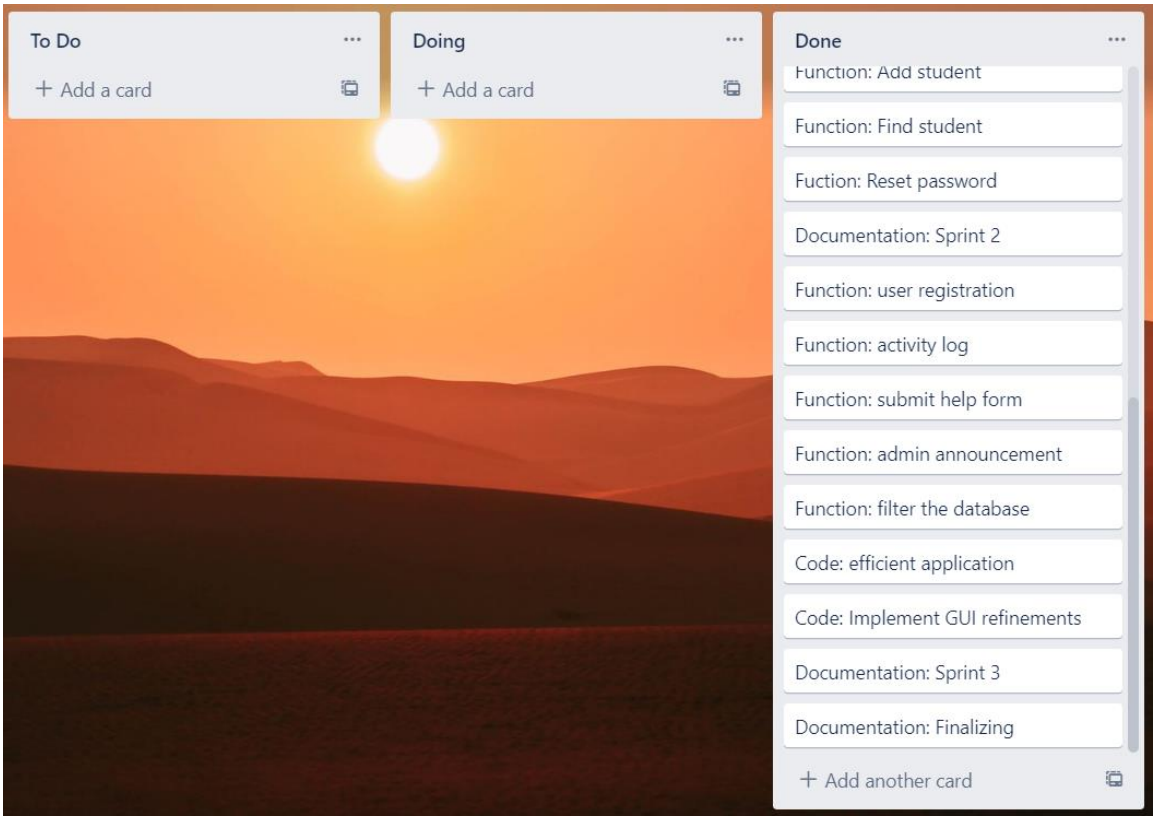

To Do
Code: efficient application
Documentation: Sprint 3
Function: filter the database
Function: admin announcement
Add another card
Doing
Code: Implement GUI refinements
Function: user registration
Function: activity log
Function: submit help form
Add another card
Done
Function: Export file
Function: Import file
Function: Student's Record Review
Function: Help button
Code: GUI first-look
Code: GUI adjustment
Documentation: Sprint 1 writeup
Function: Find a student through ID or name
Function: sort the database
Function: login
Function: log-out
Function: Add student
Function: Find student
Add another card


To Do
Code: efficient application
Function: admin announcement
Function: filter the database
Add another card
Doing
Code: Implement GUI refinements
Function: user registration
Function: activity log
Function: submit help form
Add another card
Done
Function: Export file
Function: Import file
Function: Student's Record Review
Function: Help button
Code: GUI first-look
Documentation: Sprint 1 writeup
Code: GUI adjustment
Function: Find a student through ID or name
Function: sort the database
Function: login
Function: log-out
Function: Add student
Function: Find student
Fuction: Reset password
Documentation: Sprint 2
Add another card


To Do
Code: efficient application
Documentation: Sprint 3
+ Add another card
Doing
Code: Implement GUI refinements
Function: admin announcement
Function: filter the database
+ Add another card
Done
Code: GUI adjustment
Documentation: Sprint 1 writeup
Function: Find a student through ID or name
Function: sort the database
Function: login
Function: log-out
Function: Add student
Function: Find student
Fuction: Reset password
Documentation: Sprint 2
Function: user registration
Function: activity log
Function: submit help form
+ Add another card


To Do
+ Add a card
Doing
Code: Implement GUI refinements
Code: efficient application
Documentation: Sprint 3
+ Add another card
Done
Function: Find a student through ID or name
Function: sort the database
Function: login
Function: log-out
Function: Add student
Function: Find student
Fuction: Reset password
Documentation: Sprint 2
Function: user registration
Function: activity log
Function: submit help form
Function: admin announcement
Function: filter the database
+ Add another card


To Do
Add a card
Doing
Documentation: Finalizing
Add another card
Done
Function: login
Function: log-out
Function: Add student
Function: Find student
Fuction: Reset password
Documentation: Sprint 2
Function: user registration
Function: activity log
Function: submit help form
Function: admin announcement
Function: filter the database
Code: efficient application
Code: Implement GUI refinements
Documentation: Sprint 3
Add another card


To Do
Add a card
Doing
Add a card
Done
Function: Add student
Function: Find student
Fuction: Reset password
Documentation: Sprint 2
Function: user registration
Function: activity log
Function: submit help form
Function: admin announcement
Function: filter the database
Code: efficient application
Code: Implement GUI refinements
Documentation: Sprint 3
Documentation: Finalizing
Add another card

Staging/ Grooming

As we are planning and grooming, we consider breaking down large user stories into smaller ones based on their priorities. We first begin with functions that are required for the basic implementation of our program, thereby allowing us to remove user stories that are no longer relevant to our software. As a result, we can focus more on non-functionality and any other elements. To begin, we will create a text file containing the vital data that we can import to the system for the management and edit of the data in the student record system.

GUI is our next important task because it plays an essential role in the interaction of users with the system, allowing them to manipulate elements and access available functions. GUI also gives users a clear visual view of the system's data and functionality which makes the system's operation more intuitive, and thus easier to learn and use. We will be working to ensure that our GUI is user-friendly and attractive so that the users can easily access the functions and quickly get used to the system. We will also work on non-functional requirements to make the application more efficient and the GUI work properly. When we finish implementing the GUI and the functions, we will import the data to the system and move on with other features that might be useful for the application.

Development Process

Our project is an application that queries and manages student records in the database. The program lets the users access the student's database easily and provides them with the tools for editing and managing students' information.

For Agile Development, our project is divided into sprints, each sprint comprising a full software development cycle which includes planning, requirement analysis, designing, coding, testing, deployment, and maintenance. A sprint is normally completed in an estimated time of 1-3 weeks. We also allocate and determine team member's strengths and weaknesses for each task to achieve the best outcomes. Team members with a good understanding of Java and logic programming are required for this project, and the persons who are specialized in writing software documentation are also necessary.

According to the user stories, we first create a function that displays a table of student data from an imported database together with the assisting tools to manage the student records. Second, we want to add a security layer so that only authorized users can sign up and log in to the SRMS to begin using the system. In case of forgetting the password, the user can easily reset it by submitting the correct security info. The third milestone is to add extra functions that assist the users in dealing with the system's issues or logging out of the system completely. Finally, we want to perfect the GUI so that the users can access and interact easily and intuitively with the GUI and all the system's features.

Requirements:

- Ability to access and log in the system.
- Ability to display the student database.
- Ability to manage and edit students' information and records.

- Ability to import the database file.
- Ability to export the current database file.
- Ability to access the activity log.
- Ability to display the announcements from the system.
- Ability to provide the guidelines for the system's functionality.
- Ability to contact technical support.
- Ability to log out and exit the system.

Project deliverables:

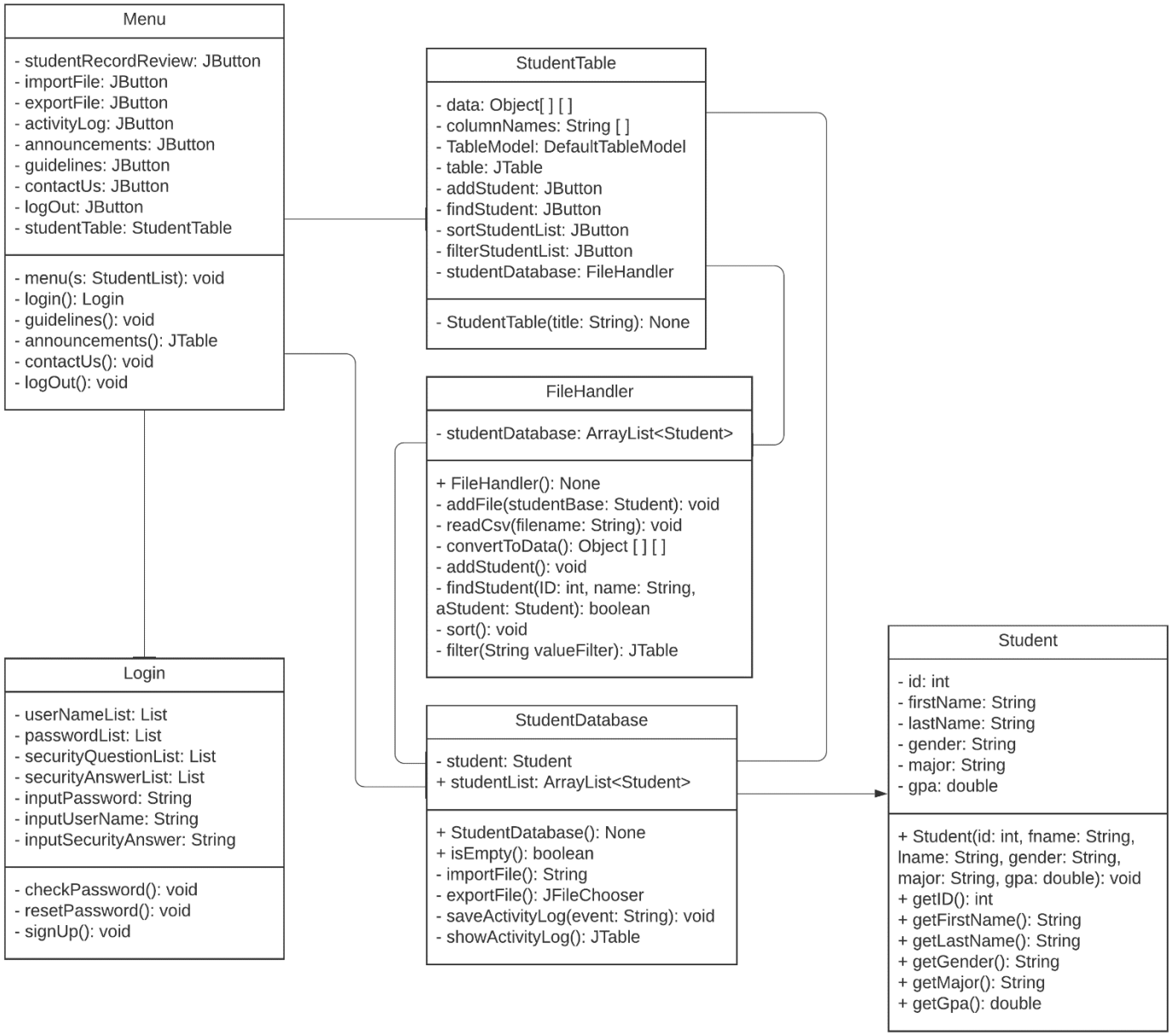
- SRMS coded by Java language.
- Fully functional GUI implemented by Java language.
- User manual.

Design and Prototyping

- Architecture: Java language.
- User Interface: The GUI will make the application's operation more intuitive by letting the user interact with the application by using the mouse and the keyboard. When a user uses the mouse to click on the function buttons, the application will display responsive information so that the user can manipulate the system and edit the data. Example: When a user clicks on the "Student Record Review" button, the application will automatically display a table of all student records in the system.
- Platforms: Windows
- Programming: Java is the main programming language to code the system. It will be utilized with JFrame, JOptionPane, JFileChooser, JTable, JScrollPane,

JTextField, JComboBox, and JDialog. For JFrame, it works like the main window where labels, buttons, and text fields are added to produce a graphical user interface (GUI). JOptionPane and JDialog allow us to create and display dialog boxes on the screen to request the user's input or make an announcement. Meanwhile, a simple and efficient method for prompting the user to select a file or directory is JFileChooser. JTable is used to display a table of student's information and JTextField enables the user to type text which we can edit and manipulate later. Finally, JScrollPane provides a scrollable view of a component and JComboBox shows a box list so that the user can choose from those choices of the specified lists.

Class Diagram



CRC Cards

Class Name: Menu	**ID:** 1	**Type:** Concrete, Domain
Description: This class displays the main menu with all the GUI buttons to let user access the system's functionality.		**Associated Use Cases:** Log in Student record review Import file Export file Activity log Announcements Guidelines Submit help form Log out

Responsibilities	**Collaborators**
• Call login() to check user login information. • Let user access the "Student Record Review" functionality. • Let user access the "Import" functionality. • Let user access the "Export" functionality. • Let user access the "Activity log" functionality. • Let user access the "Announcements" functionality. • Let user access the "Guidelines" functionality. • Let user access the "Contact us" functionality. • Call announcements() to display the messages from the admins. • Call guidelines() to display guidelines for a quick reference guide. • Call contactUs() to let the user report issues related to the system. • Call logOut() to sign out of the system.	StudentTable Login StudentDatabase

Attributes:
studentRecordReview: JButton importFile: JButton exportFile: JButton activityLog: JButton announcements: JButton guidelines: JButton contactUs: JButton logOut: JButton studentTable: StudentTable
Relationships: **Generalization (a-kind-of):** **Aggregation (has-parts):** **Other Associations:** Login StudentTable StudentDatabase

Class Name: Login	**ID:** 2	**Type:** Concrete, Domain
Description: This class allows user to sign up or log in to the system or to reset user's password.		**Associated Use Cases:** Sign up Log in Reset Password
Responsibilities • Let user create an account to begin using the system. • Check the inputted username and password before letting user log in to the system. • Let user reset the password in case user forgets the current one.		**Collaborators** Menu

Attributes:
userNameList: List passwordList: List securityQuestionList: List securityAnswerList: List inputPassword: String inputUserName: String inputSecurityAnswer: String
Relationships: **Generalization (a-kind-of):** **Aggregation (has-parts):** **Other Associations:** Menu

Class Name: StudentTable	**ID:** 3	**Type:** Concrete, Domain
Description: This class displays a table of all the student records in the system database along with some GUI buttons to let user modify the visual view of the data in the table.		**Associated Use Cases:** Student record review Add student Find student Sort data Filter data

Responsibilities	**Collaborators**
• Display a table of all the student records in the system database with the data separated by its fields. • Let user access the "Add Student" functionality. • Let user access the "Find Student" functionality. • Let user access the "Sort data" functionality. • Let user access the "Filter data" functionality.	Menu FileHandler StudentDatabase

Attributes:
data: Object[] []
columnNames: String []
TableModel: DefaultTableModel
table: JTable
addStudent: JButton
findStudent: JButton
sortStudentList: JButton
filterStudentList: JButton
studentDatabase: FileHandler

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts):

Other Associations: Menu, FileHandler, StudentDatabase

Class Name: FileHandler	**ID:** 4	**Type:** Concrete, Domain
Description: This class deals with database file to import or export student records. It also modifies the visual display of the table of student records. Based on user's button selection, it will do the according actions such as adding a new student, finding a student, sorting the table, or filtering the table.		**Associated Use Cases:** Import file Export file Add student Find student Sort data Filter data

Responsibilities	**Collaborators**
• Choose the database file and import it to the system. • Export the current system database to a file. • Convert the current student database to a table so that user can have a visual display of the database. • Add a student record to the system's database. • Find a student in the database. • Sort the table of student records based on the user's chosen sort method. • Filter the table of student records based on the filter's field and value.	StudentTable StudentDatabase

Attributes:
studentDatabase: ArrayList<Student>
Relationships: **Generalization (a-kind-of):** **Aggregation (has-parts):** **Other Associations:** StudentTable, StudentDatabase

Class Name: StudentDatabase	**ID:** 5	**Type:** Concrete, Domain
Description: This class holds a list of student records and manages the system's database.		**Associated Use Cases:** Student record review Import file Export file Add student Find student Activity log

Responsibilities	**Collaborators**
• Contain a list of student's record information (First name, Last name, Gender, ID, Major, GPA.) which serves as the system's database. User can choose to import to add more student records to the list or export to save the current list to a file. • Check if current student list is empty. • Call importFile() to begin importing process. • Call exportFile() to begin exporting process. • Call saveActivityLog(event: String) to save information such as time, username, category, and event into the designated log file. • Call showActivityLog() to convert the saved log file into a table and display it to the user.	Menu StudentTable FileHandler Student

Attributes:
student: Student studentList: ArrayList<Student>
Relationships: **Generalization (a-kind-of):** **Aggregation (has-parts):** Student **Other Associations:** Menu StudentTable FileHandler

Class Name: Student	**ID:** 6	**Type:** Concrete, Domain
Description: This class holds a single student record.		**Associated Use Cases:** Student record review Add student Find student Sort data Filter data

Responsibilities	**Collaborators**
• Hold the student's record information (First name, Last name, Gender, ID, Major, GPA.) • Release student's record information when being queried: getID() getFirstName() getLastName() getGender() getMajor() getGPA()	StudentDatabase

Attributes:
id: int.
firstName: String.
lastName: String.
gender: String.
major: String.
gpa: double.

Relationships:
Generalization (a-kind-of):
Aggregation (has-parts):
Other Associations: StudentDatabase

Test Cases

Test Case (Doc: T_01)

Test Case #: 1	**Test Case Name:** Login	**Page: 1 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** Login	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Harry Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test the login function of the system.		

Pre-conditions:
The system is successfully loaded by the user.
The system displays the login window.
The correct username is user1.
The correct password is 1234.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	From the login window, fill in the correct username and password, then click the "Sign in" button.	The system confirms the login info with the saved login details and lets the user log in the system.	Pass	
2	**Check post-condition 1**			
3	Repeat step 1 but use an incorrect password.	The system displays a warning message that the username or password is incorrect.	Pass	
4	Click "OK" button.	The system returns to the login window.	Pass	
5	**Check post-condition 2**			
6	Repeat step 1 but use a made-up username.	The system displays a warning message that the username or password is incorrect.	Pass	
7	Click "OK" button.	The system returns to the login window.	Pass	
8	**Check post-condition 3**			
9	Repeat step 1 but fill in only the password and leave the username box empty.	The system displays a warning message that the username is required.	Pass	
10	Click "OK" button.	The system returns to the login window.	Pass	
11	**Check post-condition 4**			
12	Repeat step 1 but click "X" button	The system is exited.	Pass	
13	**Check post-condition 5**			

Post-conditions:
1. User logs in to the system successfully.
2. User cannot log in to the system.
3. User cannot log in to the system.
4. User cannot log in to the system.
5. The system exits completely. User cannot log in to the system.

Test Case (Doc: T_02)

Test Case #: 2	**Test Case Name:** Sign up	**Page: 3 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** Login	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Harry Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test the sign-up function of the system.		

Pre-conditions:
The system is successfully loaded by the user.
The system displays the login window.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	From the login window, click the “Sign Up” button.	The system displays the “Registration Form” window.	Pass	
2	Fill in all the required information and click the “Submit” button.	The system displays a message of successful operation.	Pass	
3	Click “OK” button.	The system displays the login window.	Pass	
4	**Check post-condition 1**			
5	Repeat step 1, 2 but click “Cancel” or “X” button.	The system displays the login window.	Pass	
6	**Check post-condition 2**			
7	Repeat step 1, 2 but do not fill in all the required information.	The system displays a warning message that all required information must be inputted.	Pass	
10	Click “OK” button.	The system returns to the “Registration Form” window.	Pass	
11	**Check post-condition 3**			
12	Repeat step 1, 2 but make the confirmation password different from the password	The system displays a warning message that the passwords do not match.	Pass	
13	Click “OK” button.	The system returns to the “Registration Form” window.	Pass	
14	**Check post-condition 4**			

Post-conditions:
1. User successfully creates an account in the system.
2. No new account is added in the system.
3. No new account is added in the system.
4. No new account is added in the system.

Test Case (Doc: T_03)

Test Case #: 3	**Test Case Name:** Reset password	**Page: 4 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** Login	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Harry Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test the reset-password function of the system.		

Pre-conditions:
The system is successfully loaded by the user.
The system displays the login window.
The correct username is user1.
The correct security question is "What is your date of birth?"
The correct security answer is Jan 1 1990.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	From the login window, click the "Forgot password?" button.	The system displays the "Forgot Your Password?" window.	Pass	
2	Fill in the correct username, choose the correct security question and fill in the correct security answer. Finally, click the "Submit" button.	The system displays the "Reset password" window.	Pass	
3	Type in a new password then type it again to confirm. Finally, click the "Submit" button.	The system displays a message of successful operation. The system asks the user if he wants to re-login or to exit.	Pass	
4	Click "Yes" button.	The system displays the login window.	Pass	
5	**Check post-condition 1**			
6	Repeat step 1, 2 but use a made-up username.	The system displays a warning message that the username or security answer is incorrect.	Pass	
7	Click "OK" button.	The system returns to the "Forgot Your Password?" window.	Pass	
8	**Check post-condition 2**			
9	Repeat step 1, 2 but use an incorrect security question.	The system displays a warning message that the username or security answer is incorrect.	Pass	
10	Click "OK" button.	The system returns to the "Forgot Your Password?" window.	Pass	
11	**Check post-condition 3**			
12	Repeat step 1, 2 but use an incorrect security answer.	The system displays a warning message that the username or security answer is incorrect.	Pass	

13	Click “OK” button.	The system returns to the “Forgot Your Password?” window.	Pass	
14	**Check post-condition 4**			
15	Repeat step 1, 2, 3 but make the confirmation password different from the new password	The system displays a warning message that the password does not match.	Pass	
16	Click “OK” button.	The system returns to the “Reset password” window.	Pass	
17	**Check post-condition 5**			
18	Repeat step 1, 2, 3, 4 but click “No” button.	The system is exited.	Pass	
19	**Check post-condition 6**			
20	Repeat step 1 and click “Cancel” or “X” button.	The system returns to the login window.	Pass	
21	**Check post-condition 7**			

Post-conditions:
1. The new password is saved in the system.
2. User cannot reset the password.
3. User cannot reset the password.
4. User cannot reset the password.
5. User cannot reset the password.
6. The new password is saved in the system.
7. User cannot reset the password.

Test Case (Doc: T_04)

Test Case #: 4	**Test Case Name:** Student Record Review	**Page: 6 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** StudentTable	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Harry Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test the "Student Record Review" function of the system		

Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	From the main menu of SRMS, click the "Student Record Review" button.	The system displays a table of all student's information on the screen.	Pass	
2	Click "X" button.	The system returns to the main menu	Pass	
3	**Check post-condition 1**			

Post-conditions:
1. The students' database is successfully loaded by the system and displayed on the screen.

Test Case (Doc: T_05)

Test Case #: 5	**Test Case Name:** Add student	**Page: 7 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** StudentTable	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Harry Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test "Add a student" function of the system.		

Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.
The user clicks the "Student Record Review" button.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	In the "Student Record Review" window, click the "Add a student" button.	System displays a "Student Registration Form" dialog.	Pass	
2	Fill in all the boxes in the "Student Registration Form," then click the "Register" button.	System displays a "Student Information" dialog to ask the user to confirm the student's data.	Pass	
3	Click "Yes" button to confirm.	The system displays a message of successful operation.	Pass	
4	**Check post-condition 1**			
5	Repeat step 1, 2 but do not fill in all the boxes.	The system displays a warning message that the user has to fill in all the information.	Pass	
6	Click "OK" button.	The system returns to the "Student Registration Form" dialog.	Pass	
7	**Check post-condition 2**			
8	Repeat step 1, 2 but click "Cancel" or "X" button	The system returns to the "Student Record Review" window.	Pass	
9	**Check post-condition 3**			
10	Repeat step 1, 2, 3 but click "No" button.	The system displays a warning message that the data of the new student is not saved. The system returns to the "Student Registration Form" dialog.	Pass	
11	**Check post-condition 4**			
12	Repeat step 1 and click "Cancel" or "X" button	The system returns to the "Student Record Review" window.	Pass	
13	**Check post-condition 5**			

Post-conditions:
1. User successfully adds the data of a student into the system.
2. The data of the new student is not added into the system.
3. The data of the new student is not added into the system.
4. The data of the new student is not added into the system.
5. The data of the new student is not added into the system.

Test Case (Doc: T_06)

Test Case #: 6	**Test Case Name:** Find student	**Page: 9 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** StudentTable	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Khang Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test "Find a student" function of the system.		

Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.
The user clicks the "Student Record Review" button.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	In the "Student Record Review" window, click the "Find a student" button.	The system displays an input window.	Pass	
2	Fill in a student ID and click "OK" button.	The system displays an information window with the queried student's data.	Pass	
3	Click "OK" button.	The system returns to the "Student Record Review" window.	Pass	
4	**Check post-condition 1**			
5	Repeat step 1, 2 but fill in a student's name	The system displays an information window with the queried student's data.	Pass	
	Click "OK" button.	The system returns to the "Student Record Review" window.	Pass	
6	**Check post-condition 2**			
7	Repeat step 1, 2 but enter a wrong student ID	The system displays a warning message that the student ID or name does not exist.	Pass	
8	Click "OK" button.	The system returns to the input window.	Pass	
9	**Check post-condition 3**			
10	Repeat step 1, 2 but enter a wrong student's name	The system displays a warning message that the student ID or name does not exist.	Pass	
11	Click "OK" button.	The system returns to the input window.	Pass	
12	**Check post-condition 4**			
13	Repeat step 1 and click "Cancel" or "X" button	The system returns to the "Student Record Review" window.	Pass	
14	**Check post-condition 5**			

Post-conditions:
1. The system successfully displays the queried student's information on the screen.
2. The system successfully displays the queried student's information on the screen.
3. The system cannot display the queried student's information on the screen.
4. The system cannot display the queried student's information on the screen.
5. The system has not done anything as the user cancels finding a student's info.

Test Case (Doc: T_07)

Test Case #: 7	**Test Case Name:** Sort data	**Page: 11 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** StudentTable	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Khang Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test "Sort data" function of the system.		

Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.
User successfully clicks the "Student record review" button.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	In the "Student Record Review" window, click the "Sort Student List" button.	A "Sort Options" window appears on the screen.	Pass	
2	Choose the students' data column and sort type, then click "OK" button.	System displays the students' information in the requested sort order.	Pass	
3	**Check post-condition 1**			
4	Repeat step 1 and click "Cancel" or "X" button.	The system returns to the "Student Record Review" window.	Pass	
5	**Check post-condition 2**			

Post-conditions:
1. The system successfully displays the students' information in the requested order.
2. The system has not done anything as the user cancels sorting the student-data Table.

Test Case (Doc: T_08)

Test Case #: 8	**Test Case Name:** Import file	**Page: 12 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** Import	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Khang Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test "Import file" button to let the user import the file containing student data into the system.		

Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	From the main menu of SRMS, click the "Import file" button.	The import file window appears on the screen.	Pass	
2	Click on "Look in" to find the location of the file.	The system displays the folder tree in the computer so that user can browse to look for the file.	Pass	
3	Select the appropriate data file and clicks the "Open" button.	The system displays an "Info" window with the filename of the selected file.	Pass	
4	Click "OK" button.	The system returns to the main menu	Pass	
5	**Check post-condition 1**			
6	Repeat step 1, 2, 3 but select an inappropriate file.	The system displays a warning message that the wrong file was chosen.	Pass	
7	Click "OK" button.	The system returns to the import file window	Pass	
6	**Check post-condition 2**			
7	Repeat step 1, 2, 3 but click "Cancel" button.	The system returns to the main menu	Pass	
8	**Check post-condition 3**			
9	Repeat step 1 and click "Cancel" or "X" button.	The system returns to the main menu	Pass	
10	**Check post-condition 4**			

Post-conditions:
1. The data of students in the file is imported into the system.
2. No student data is imported into the system.
3. No student data is imported into the system.
4. No student data is imported into the system.

Test Case (Doc: T_09)

Test Case #: 9	**Test Case Name:** Export file	**Page: 13 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** Export	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Khang Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test "Export file" button to let the user export the student database into a file.		

Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	From the main menu of SRMS, click the "Export file" button.	The export file window appears on the screen.	Pass	
2	Click on "Save in" to find the location of the file.	The system will display the folder tree in the computer so that user can browse to choose the location to save the file.	Pass	
3	Select the preferred location to save, then type the filename and click the "Save" button.	The system displays a message of successful operation.	Pass	
4	Click "OK" button.	The system returns to the main menu	Pass	
5	**Check post-condition 1**			
6	Repeat step 1, 2, 3 but do not type in a filename.	The system displays a warning message that the filename is needed.	Pass	
7	Click "OK" button.	The system returns to the export file window	Pass	
6	**Check post-condition 2**			
7	Repeat step 1, 2, 3 but click "Cancel" or "X" button.	The system returns to the main menu	Pass	
8	**Check post-condition 3**			
9	Repeat step 1 but click "Cancel" or "X" button.	The system returns to the main menu	Pass	
10	**Check post-condition 4**			

Post-conditions:
1. The system successfully backs up the current database into a file and saves them to a designated folder.
2. The data of students in the system is not exported to a file.
3. The data of students in the system is not exported to a file.
4. The data of students in the system is not exported to a file.

Test Case (Doc: T_10)

Test Case #: 10	**Test Case Name:** Log out	**Page: 15 of 15**
System: Student Record Management System (SRMS)	**Subsystem:** LogOut	
Designed By: Coding Team	**Design Date:** April 2021	
Executed By: Khang Nguyen	**Execution Date:** 05/07/2021	
Short Description: Test "Log Out" button if it lets user log out of the system.		

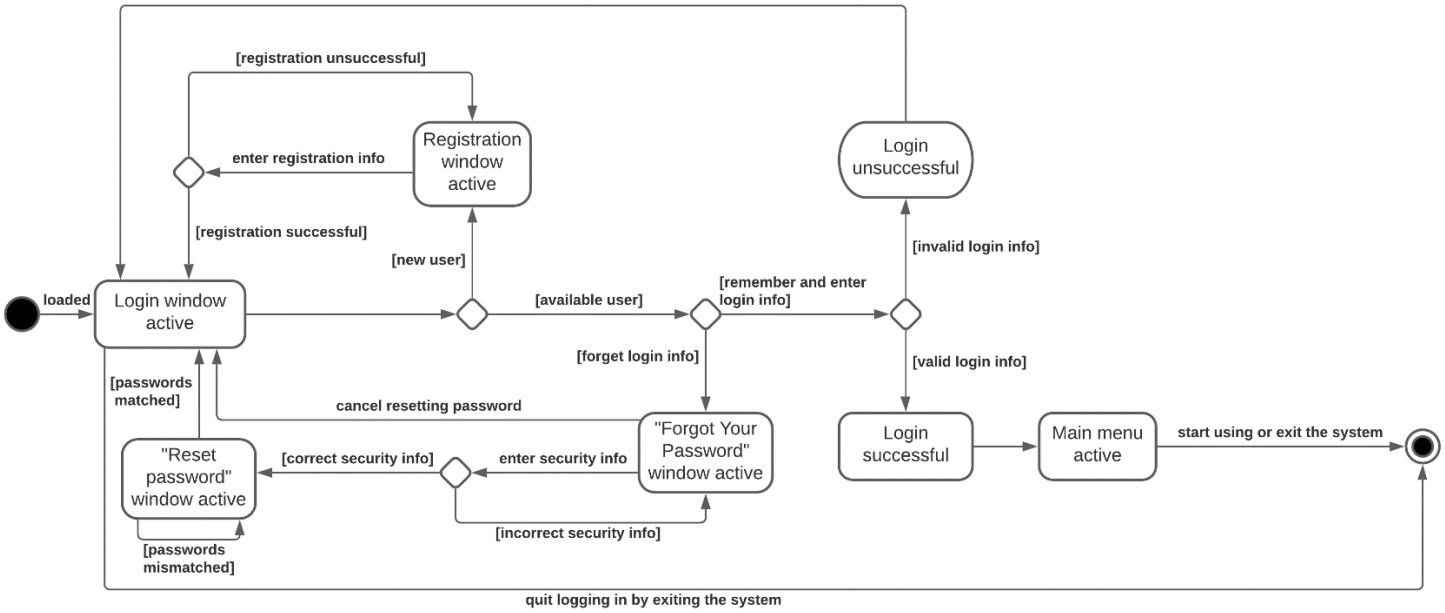
Pre-conditions:
The system is successfully loaded and logged in by the user.
The system displays the main menu.

Step	**Action**	**Expected System Response**	**Pass/ Fail**	**Comment**
1	From the main menu of SRMS, click the "Log Out" button.	The user is logged out of the system, and the system is exited.	Pass	
2	**Check post-condition 1**			

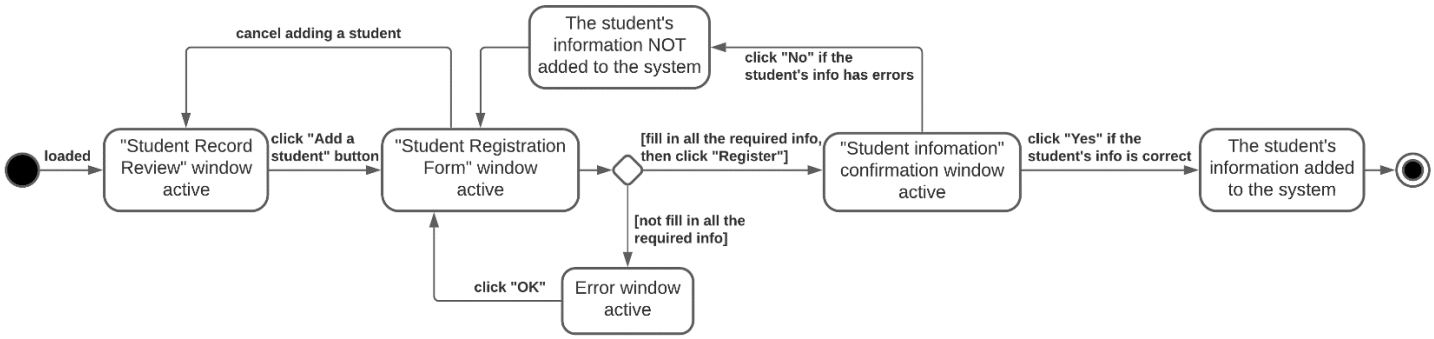
Post-conditions:
1. The user is successfully logged out of the system, and the system is exited.

State Diagrams

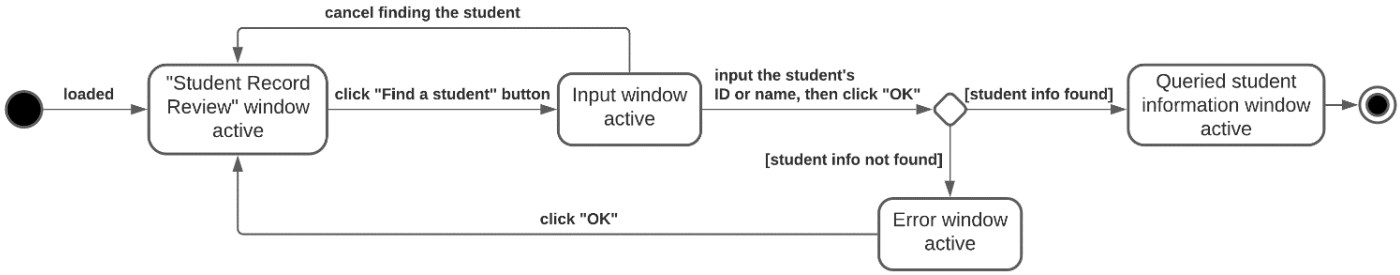
Login



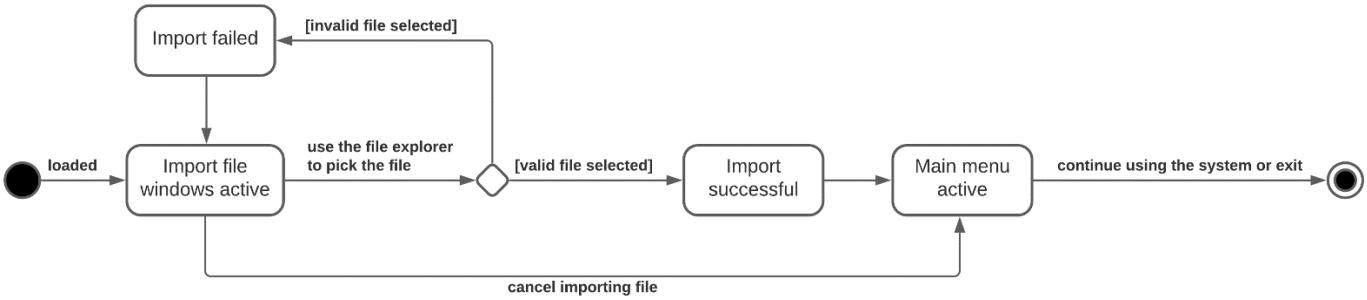
Add a student



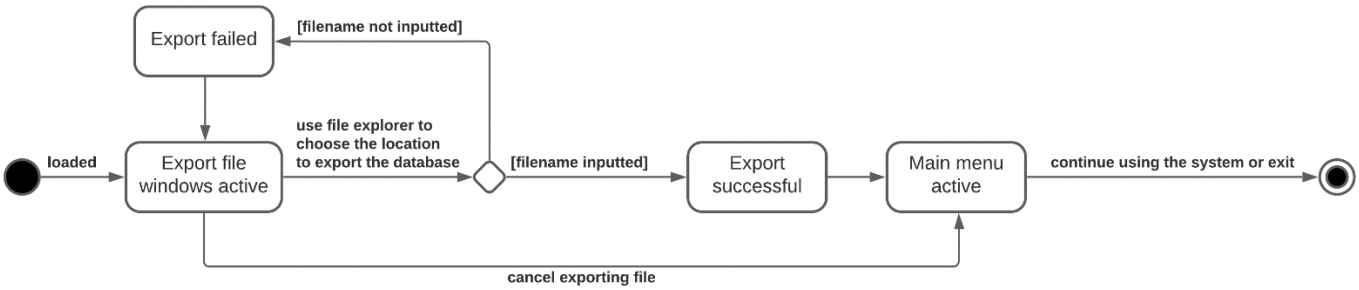
Find a student



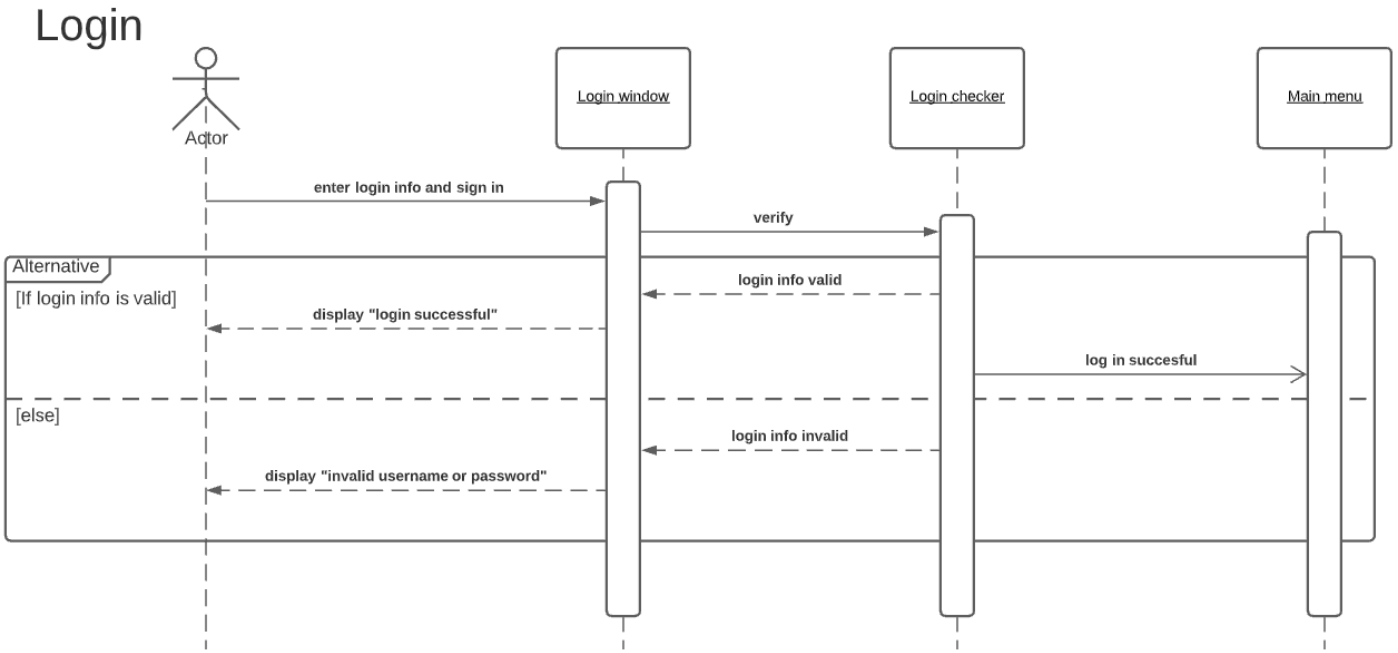
Import

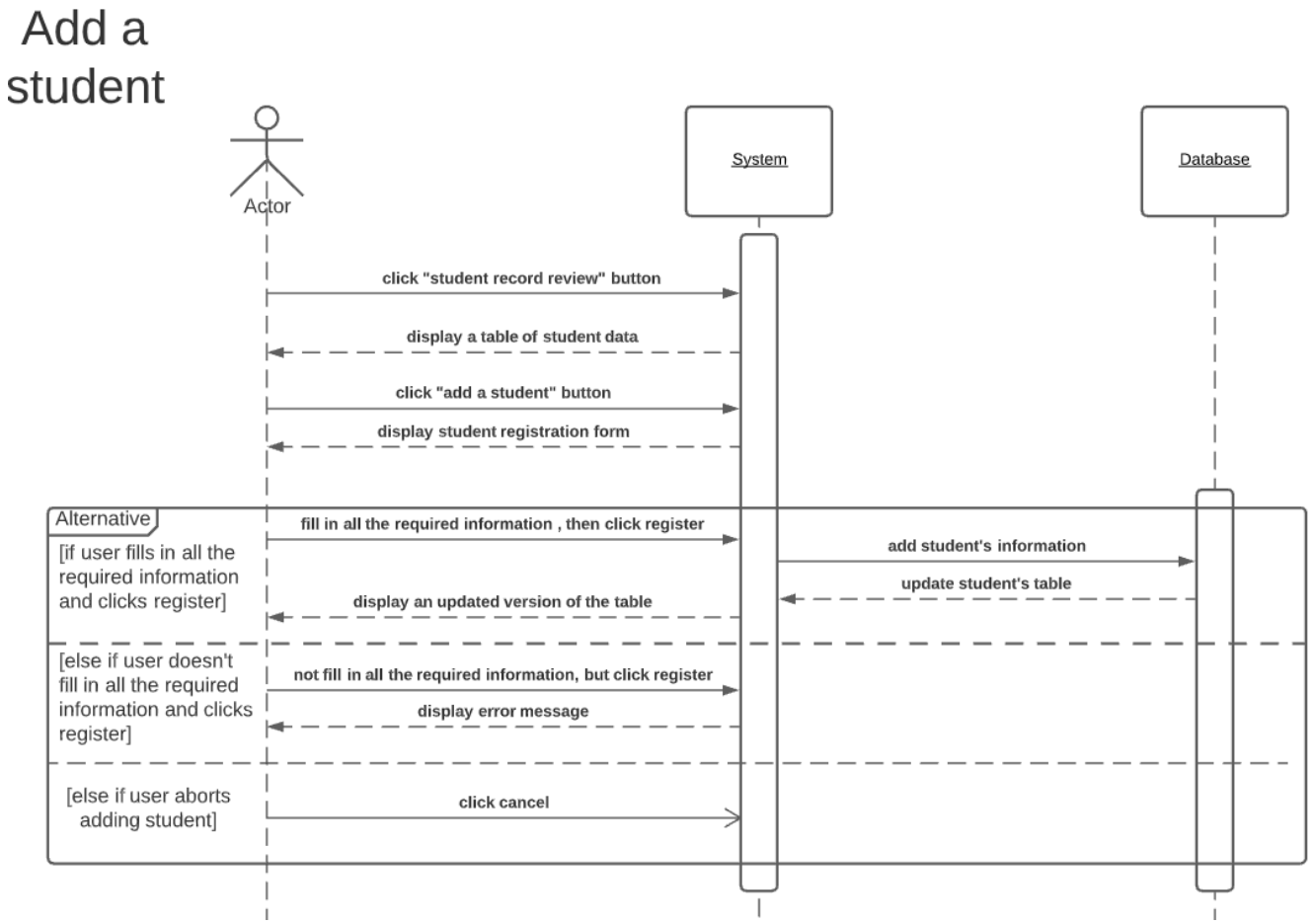


Export

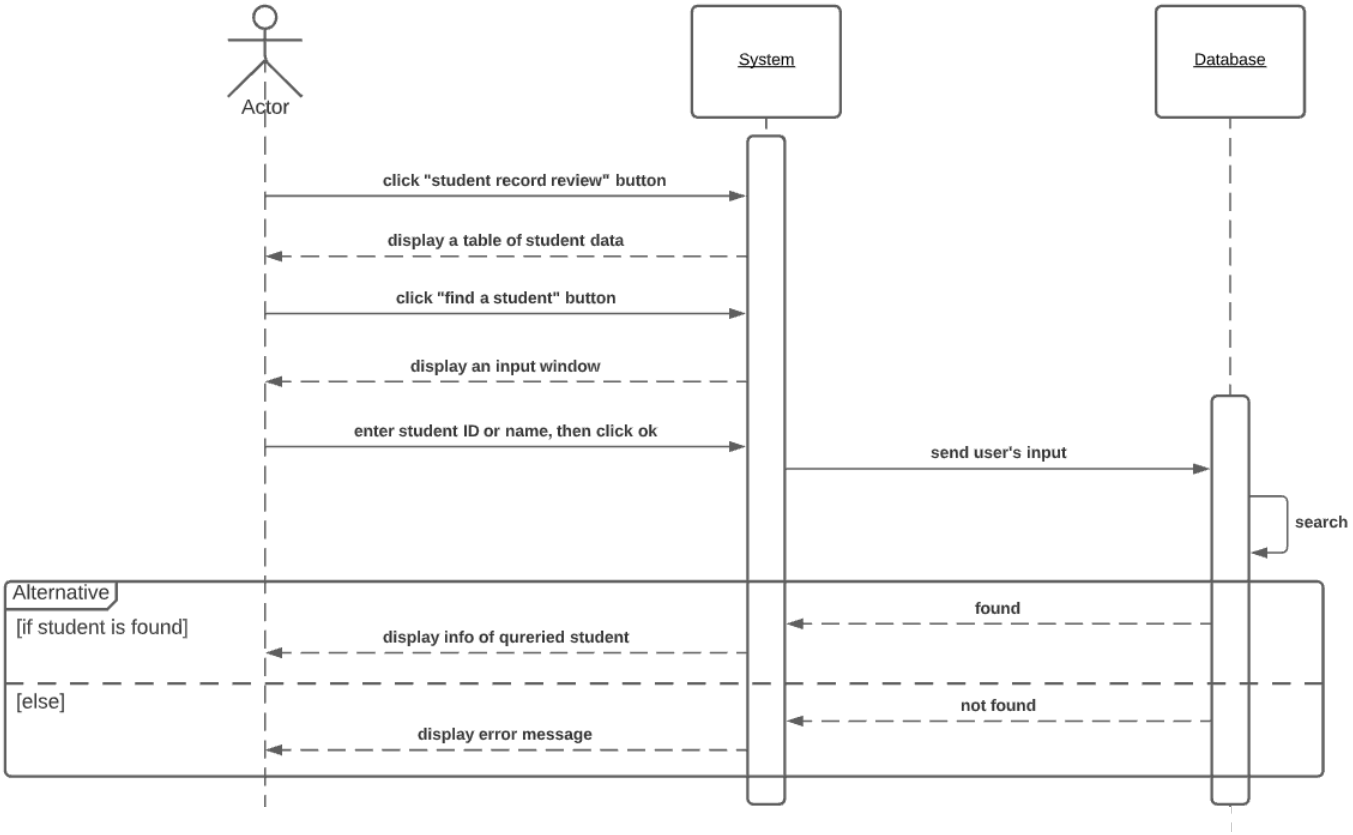


Sequence Diagrams

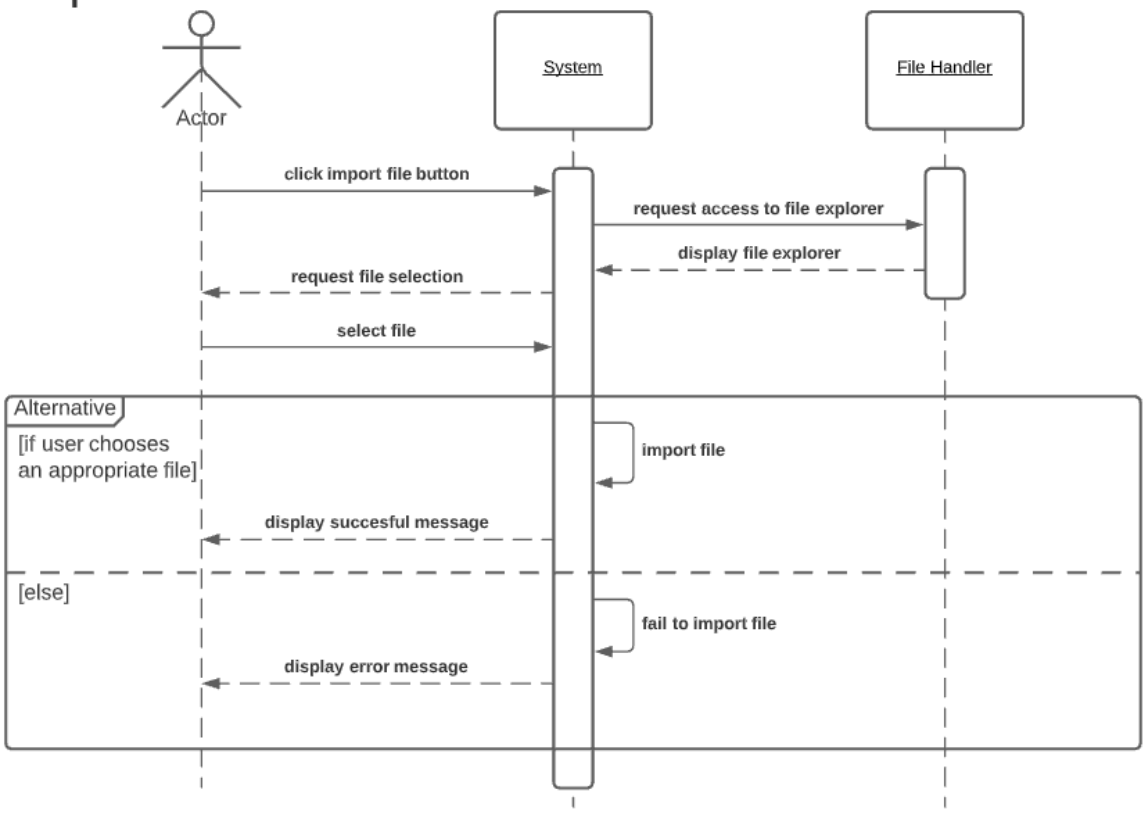




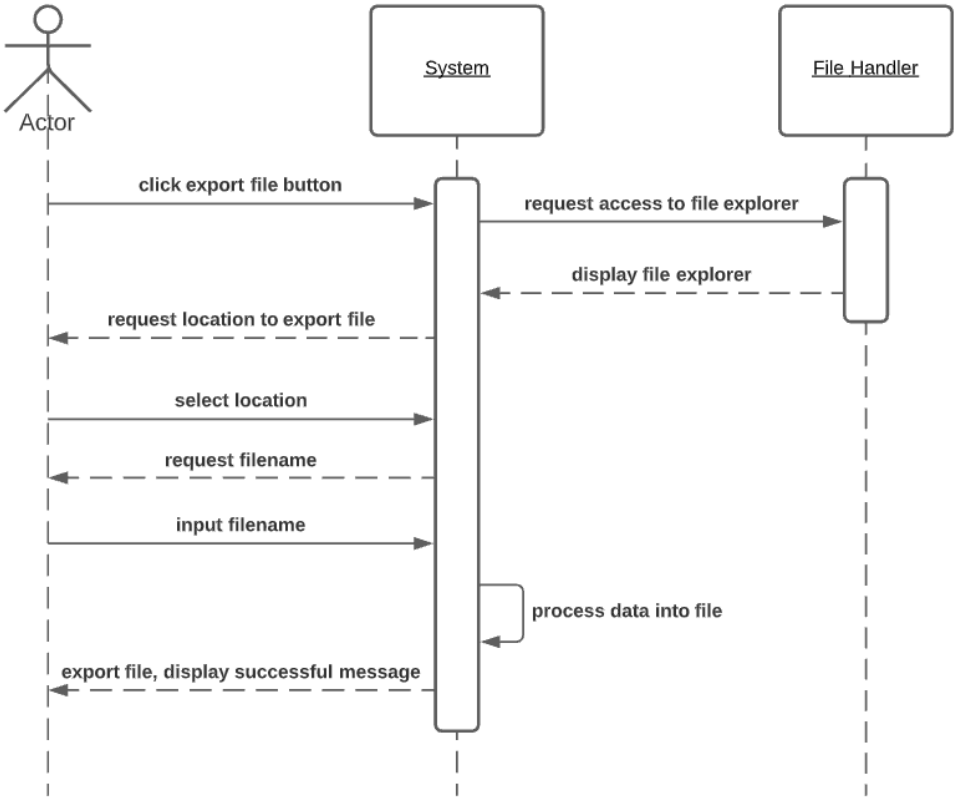
Find a student



Import



Export


Actor
System
File Handler
click export file button
request access to file explorer
display file explorer
request location to export file
select location
request filename
input filename
process data into file
export file, display successful message

User Manual

Getting Started by Login Account

1. Load the application: Student Record Management System (SRMS).
2. The login dialog of SRMS will be loaded first as the below figure 1.

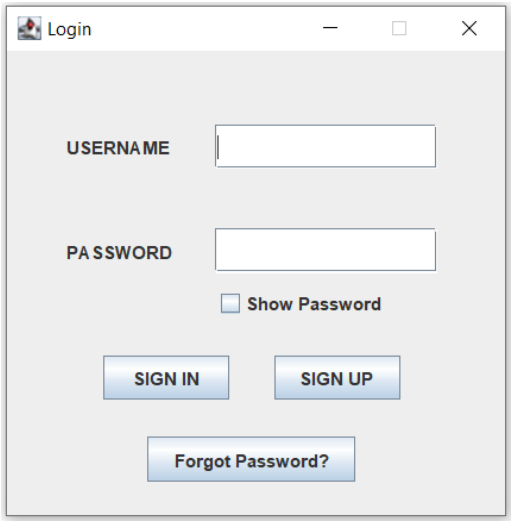


Figure 1

3. Enter the username and password, then click the “Sign in” button to login to Student Record Management System (SRMS). User can click the “Show Password” option to check the spelling of the password text field if necessary (Figure 2).

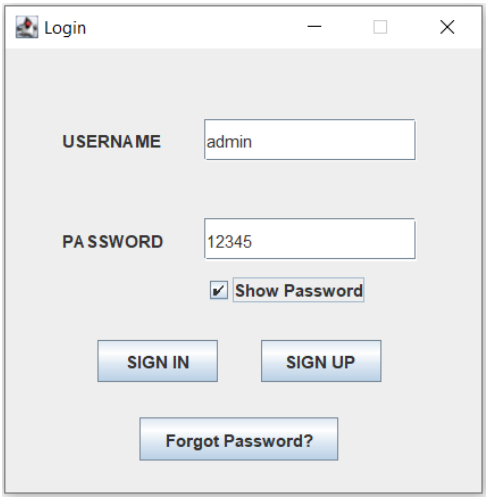


Figure 2

4. If the username or password is invalid, an error message will pop up (Figure 3). If both of them are valid, a successful login message will appear (Figure 4).

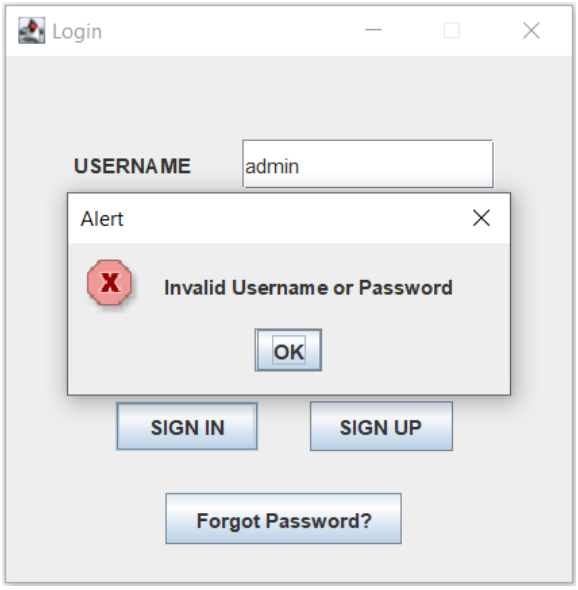


Figure 3

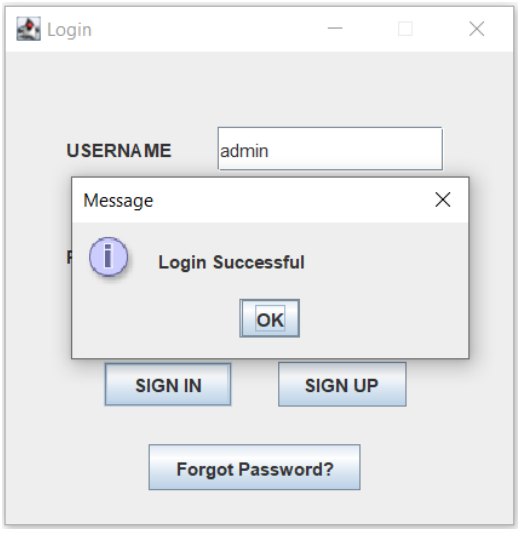


Figure 4

Signing up new account

1. For a new user, select “Sign up” button from figure 1. A “Registration Form” window will be displayed to let new users fill in their information as the below figure 5.

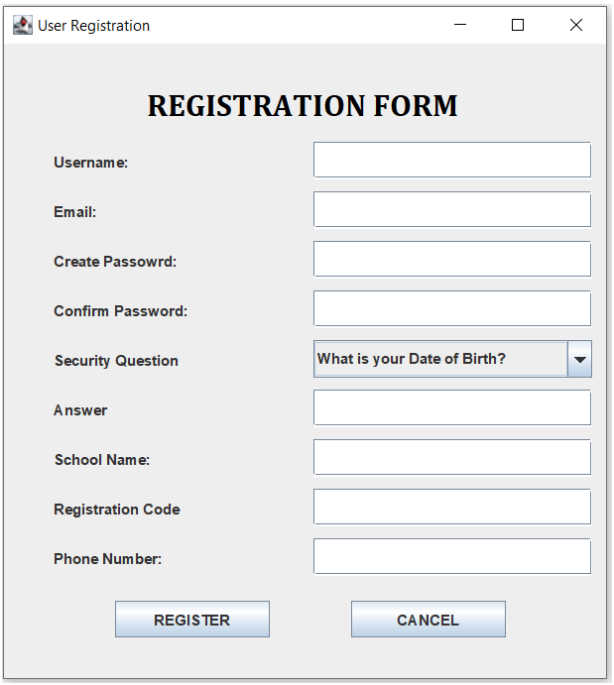


Figure 5

2. An error message will show up on the screen if the user does not fill in all the required information of the registration form but clicks the “Register” button (Figure 6).

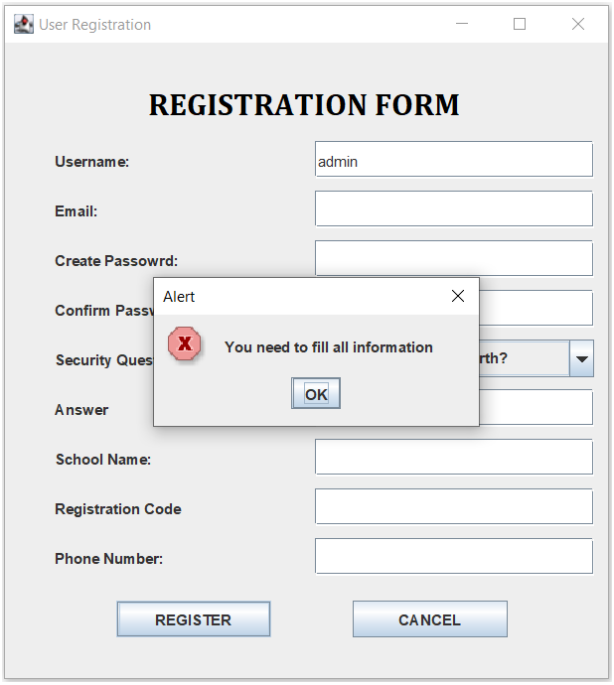


Figure 6

3. In case of a mismatch between the password and confirm password, a warning message will pop up when the user clicks the “Register” button (Figure 7).

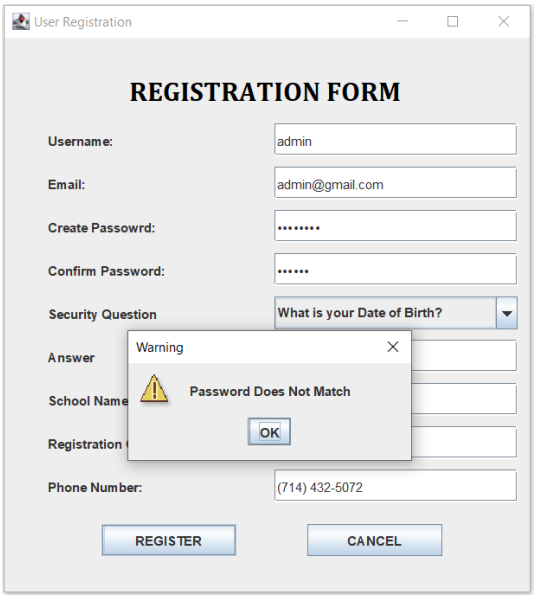


Figure 7

4. When the user clicks the “Register” button, a successful message shows up if the user has filled in all the required information and both lines of password are matched as shown in the figure 8 below.

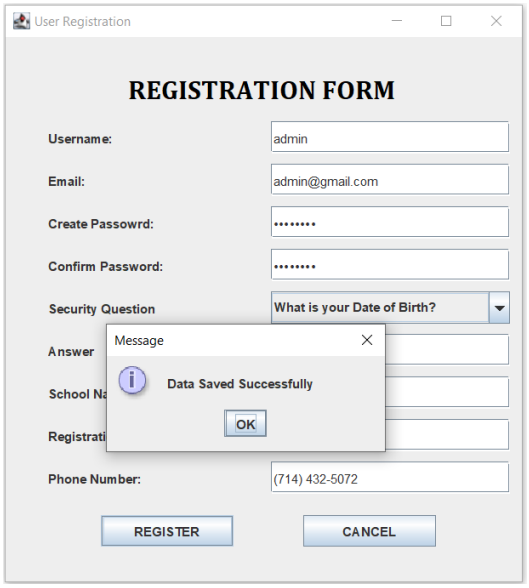


Figure 8

Resetting the password

1. From figure 1, click the “Forgot Password?” button. Fill in the security information (username, security question, and its answer) as the below figure 9 and 10, then click the “Submit” button to get the approval to reset the password.

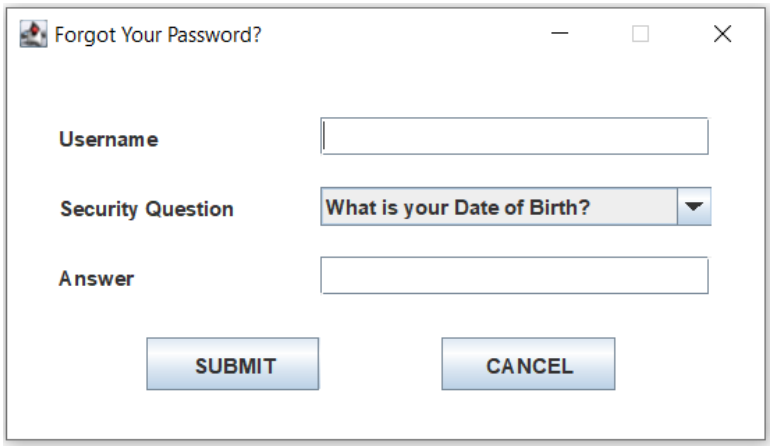


Figure 9

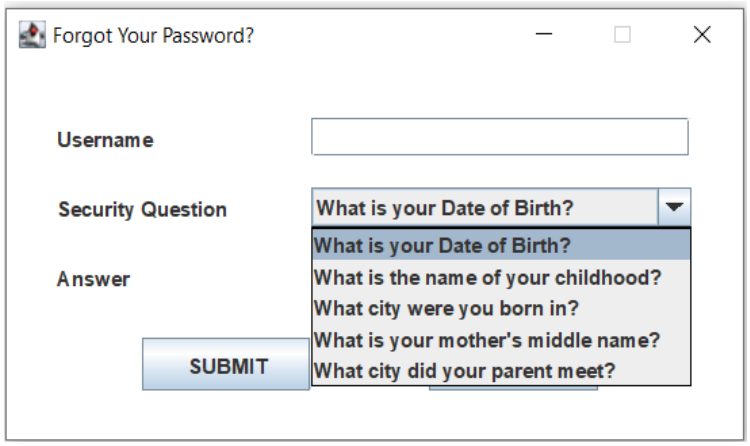


Figure 10

2. After submitting the security information, if the inputted information fails the verification, an error message dialog will pop up as the below figure 11.

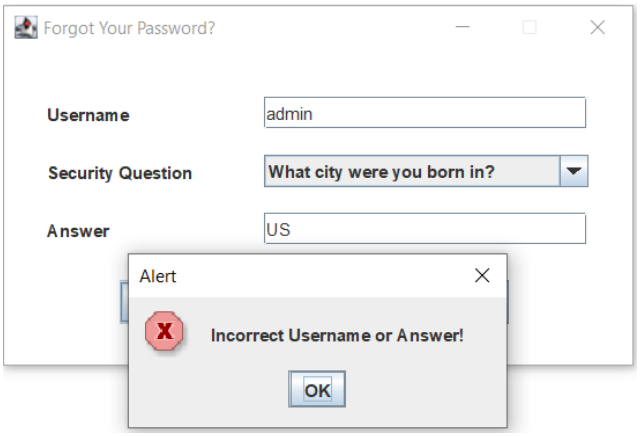


Figure 11

3. After the security information is verified, the "Reset Password" dialog will show up (Figure 12). If the password and the confirm password do not match, an error message appears (Figure 13), and vice versa, a successful message is displayed as figure 14.

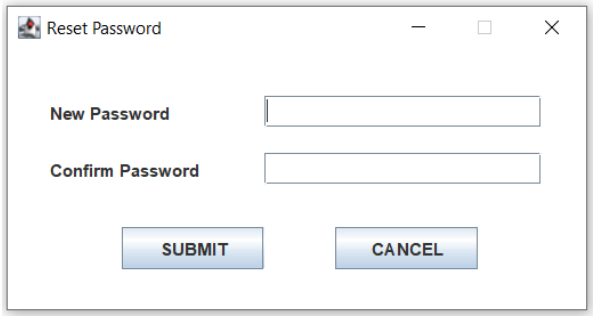


Figure 12

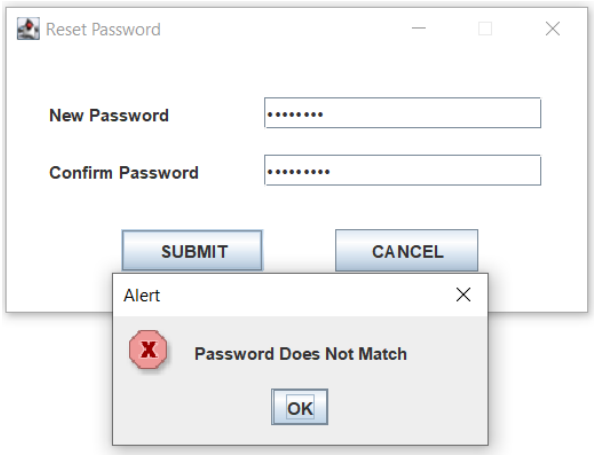


Figure 13

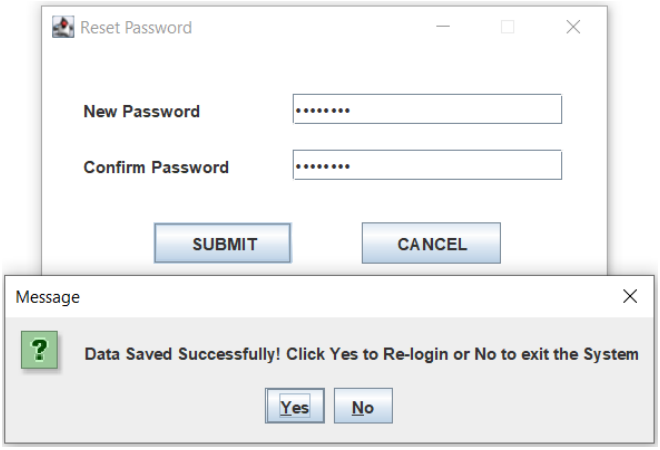


Figure 14

4. Select “No” to exit the system or “Yes” to return to the “Login” window.

Getting Started with an Existing Database file

1. The menu window of SRMS will be loaded after the user logs in to the system successfully as the below figure 15.

Figure 15

Using Student Record Review

1. From the menu window of SRMS as in the above figure 15, click the “Student Record Review” button.
2. A data table of all the students in the database with columns of ID, first name, last name, gender, major, and GPA will appear as figure 16.

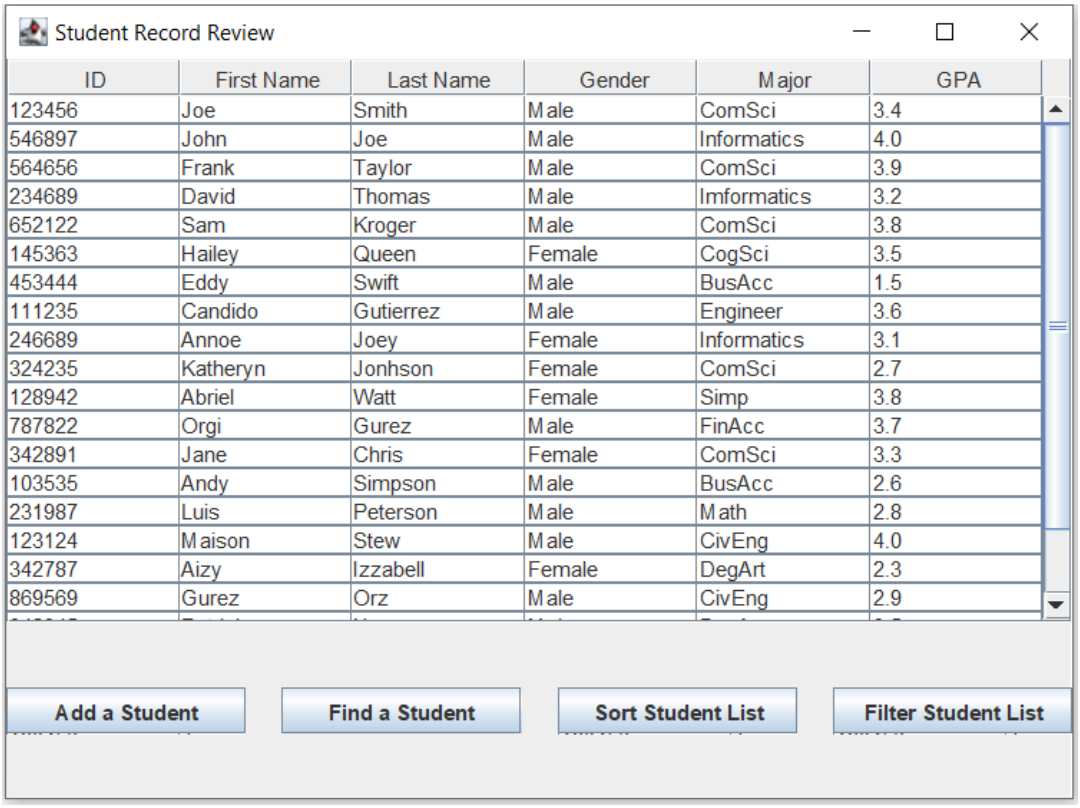


Figure 16

Adding a Student

1. In the “Student Record Review” window as the above figure 16, click the “Add a Student” button to add more students into the existing database. A “Student Registration Form” dialog will appear (Figure 17 and 18).

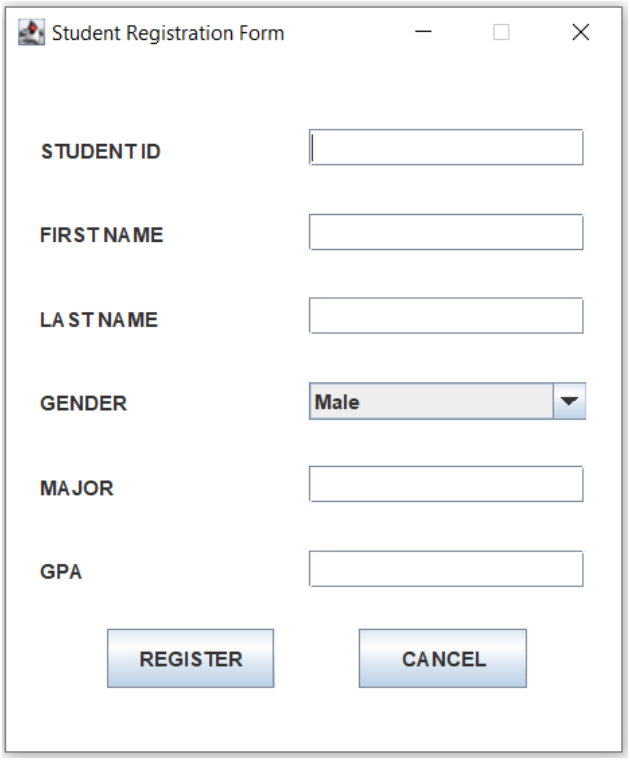


Figure 17

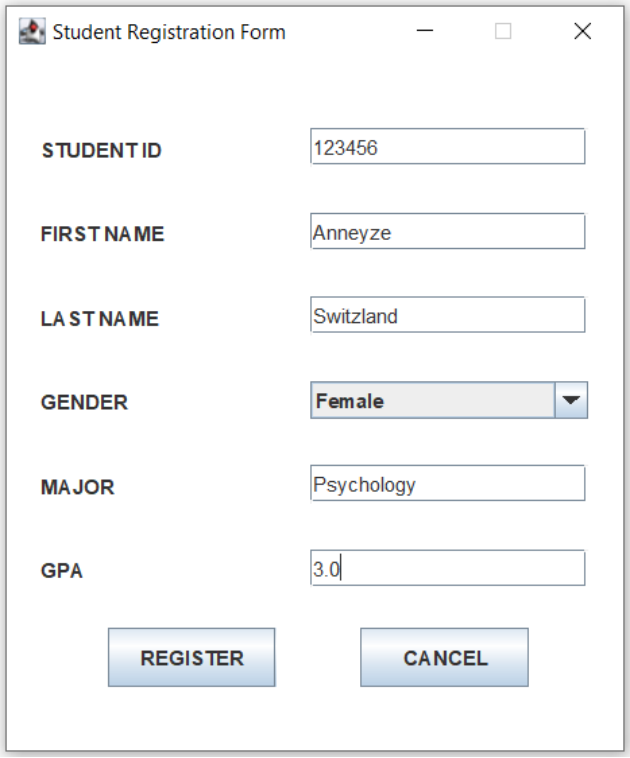


Figure 18

2. Click “Cancel” button to cancel adding students and go back to the “Student Record Review” window.
3. Click the “Register” button to register a new student after filling in all the boxes in the “Student Registration Form.”
4. If the user does not fill in all the boxes, an error message will be displayed on the screen as figure 19. Otherwise, a “Student Information” confirmation dialog will pop up (Figure 20).

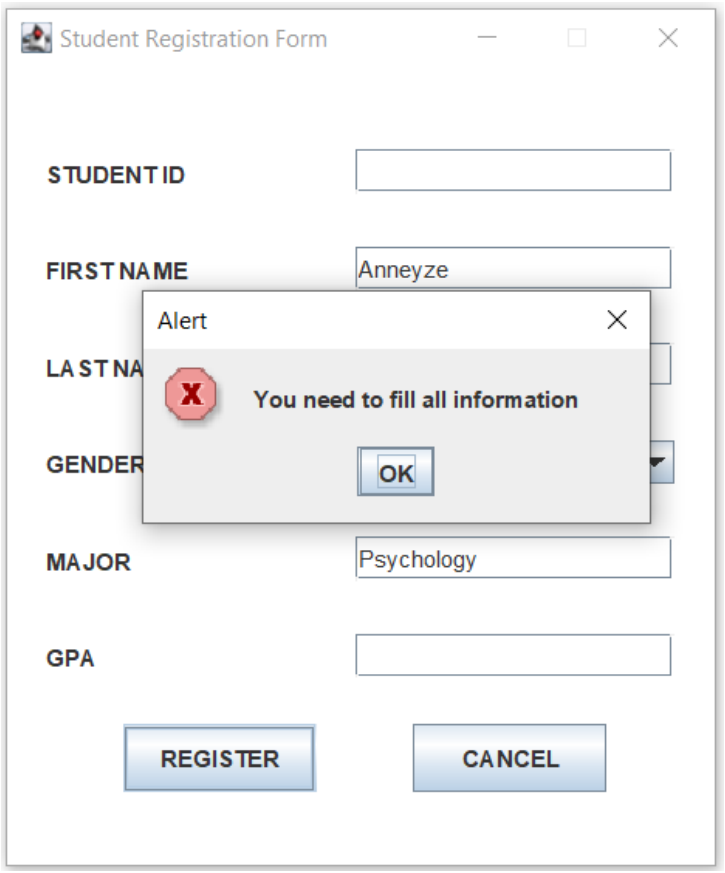


Figure 19

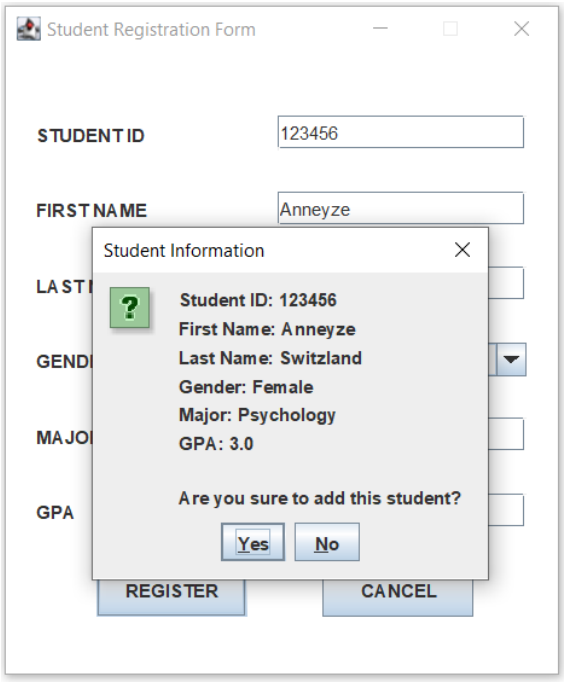


Figure 20

5. Click “No” to review and correct any incorrect information (Figure 21) or click “Yes” to confirm adding the new student to the database (Figure 22). The system will display a corresponding message based on the user’s choice.

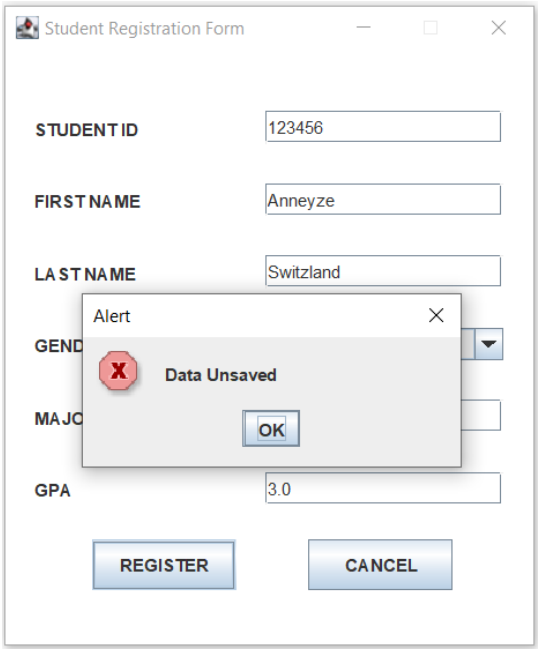


Figure 21

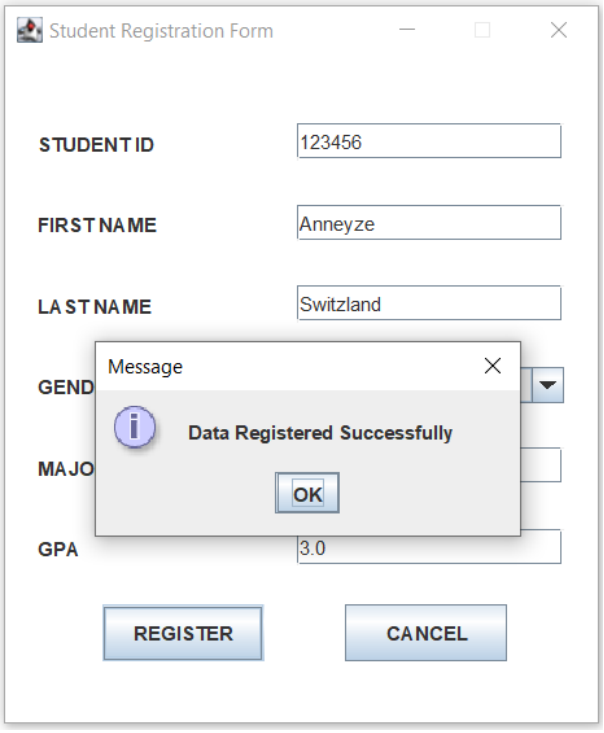


Figure 22

6. Click “OK” or “X” button to return to the “Student Registration Form” dialog.

Finding a Student

1. Click the “Find a Student” button from the “Student Record Review” window (Figure 16).
2. An input window will appear (Figure 23).

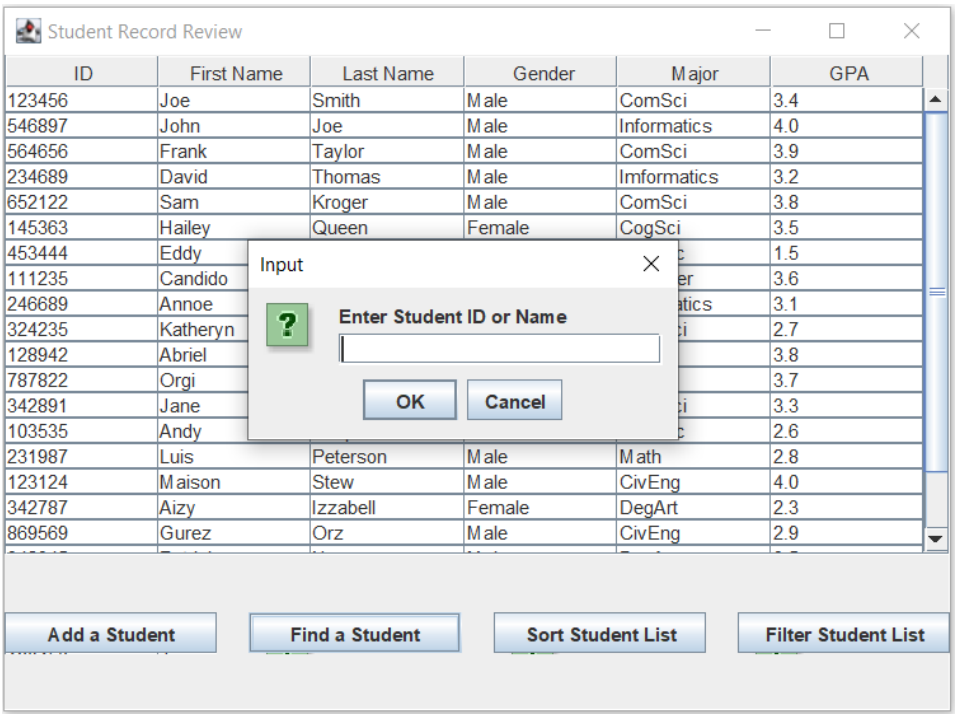


Figure 23

3. Enter student ID or name.
4. Click “OK” button to begin searching for the student or click “Cancel” button to go back to the “Student Record Review” window.
5. A window with the information of the queried student will appear if the inputted student ID or name matches a student’s information in the database as the below Figure 24. Otherwise, if the student cannot be found, an error message will be displayed on the screen as figure 25.

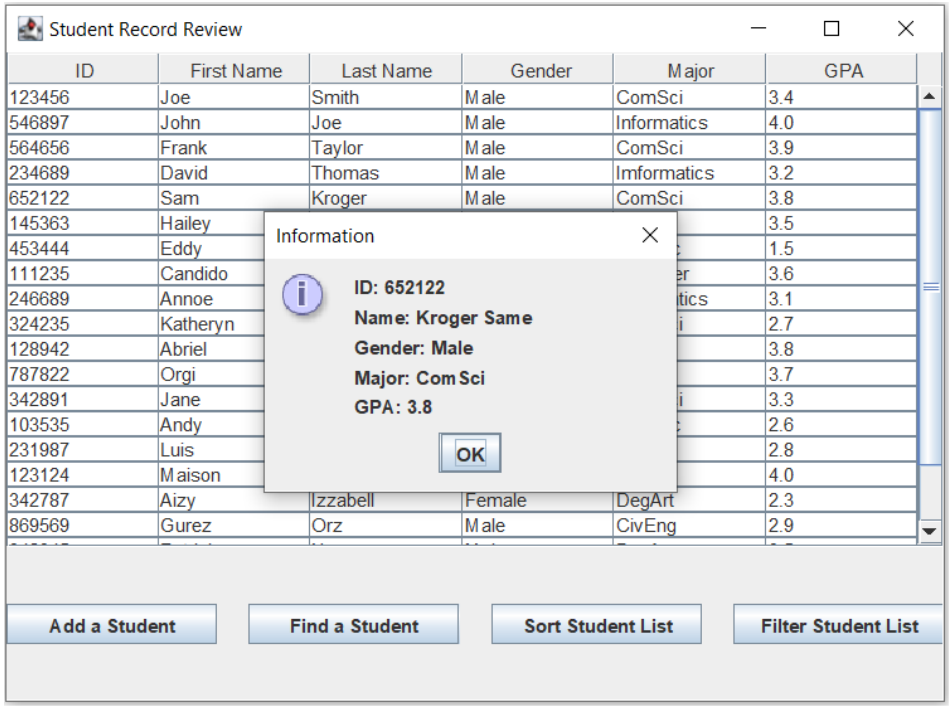


Figure 24

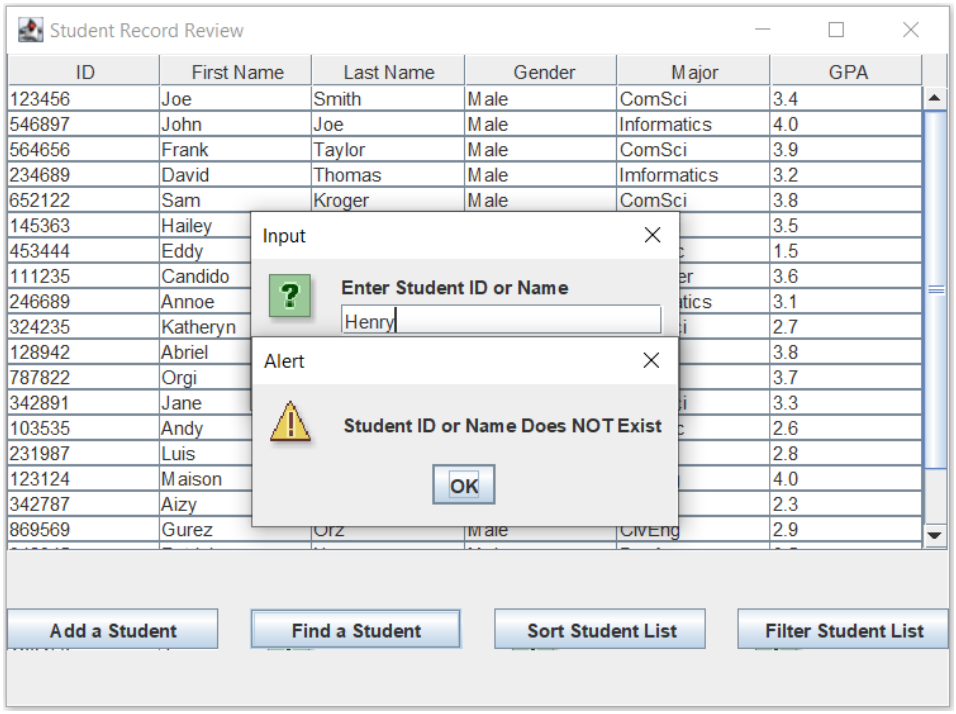


Figure 25

6. Click “OK” or “X” button to return to the “Student Record Review” window.

Sorting Student List

1. In the “Student Record Review” dialog (Figure 16), click the “Sort Student List” button.

The “Sort Options” dialog will appear as the following figure 26, 27, and 28.

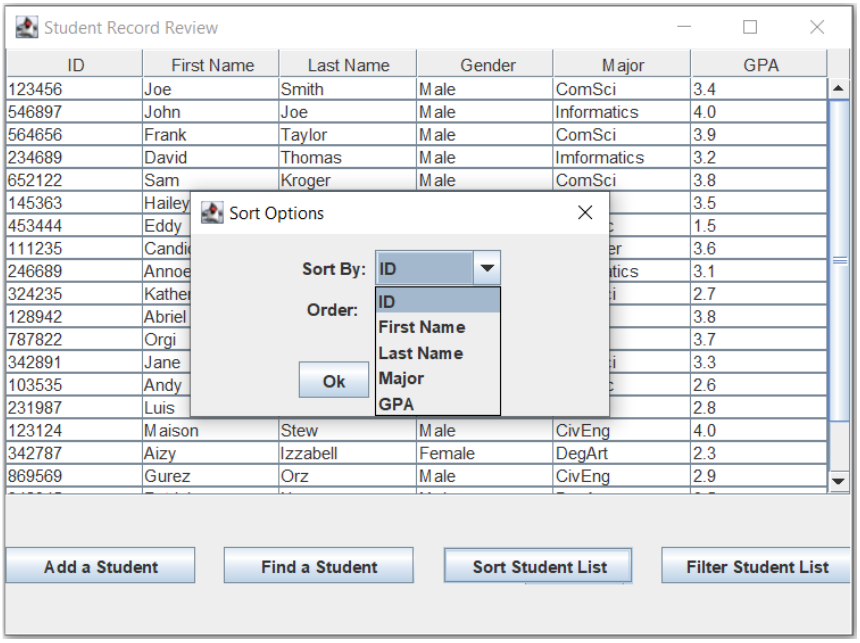


Figure 26

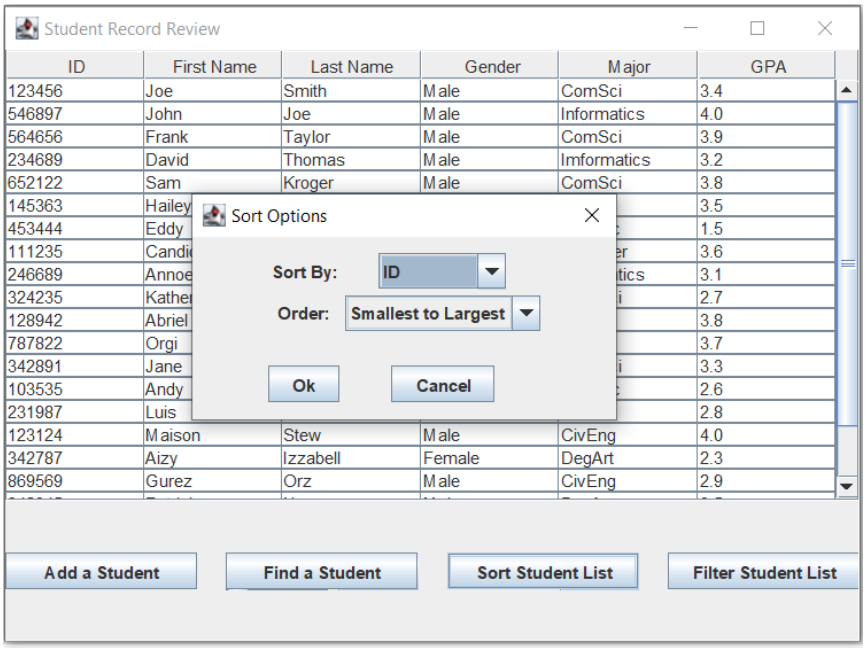


Figure 27

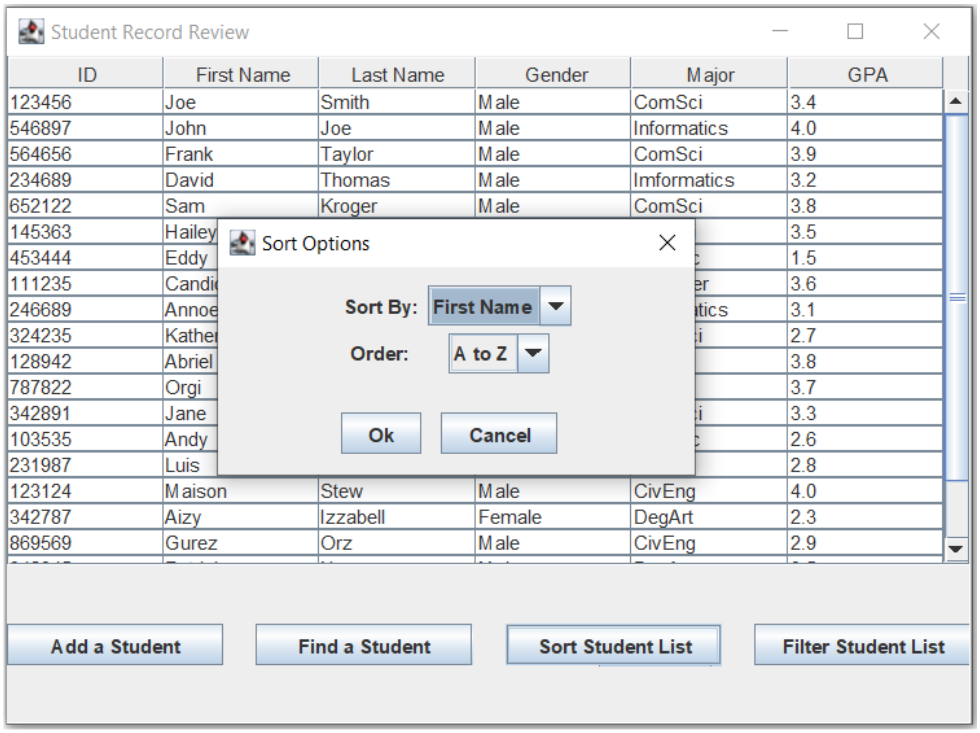


Figure 28

2. Choose the column and select the preferred order to sort.
3. Click “OK” button to sort or “Cancel” button to go back to the “Student Record Review” window (Figure 16).

Filtering Student List

1. In the “Student Record Review” window (Figure 16), select “Filter Student List” button. A “Filter Option” dialog will be displayed as the below figure 29 and 30.

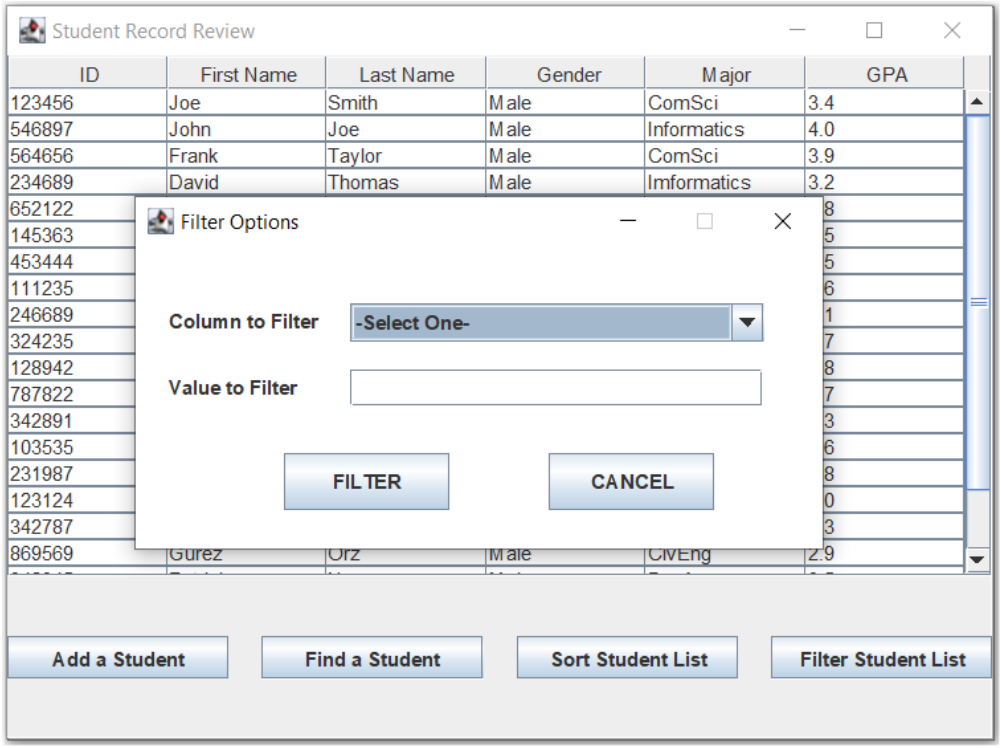


Figure 29

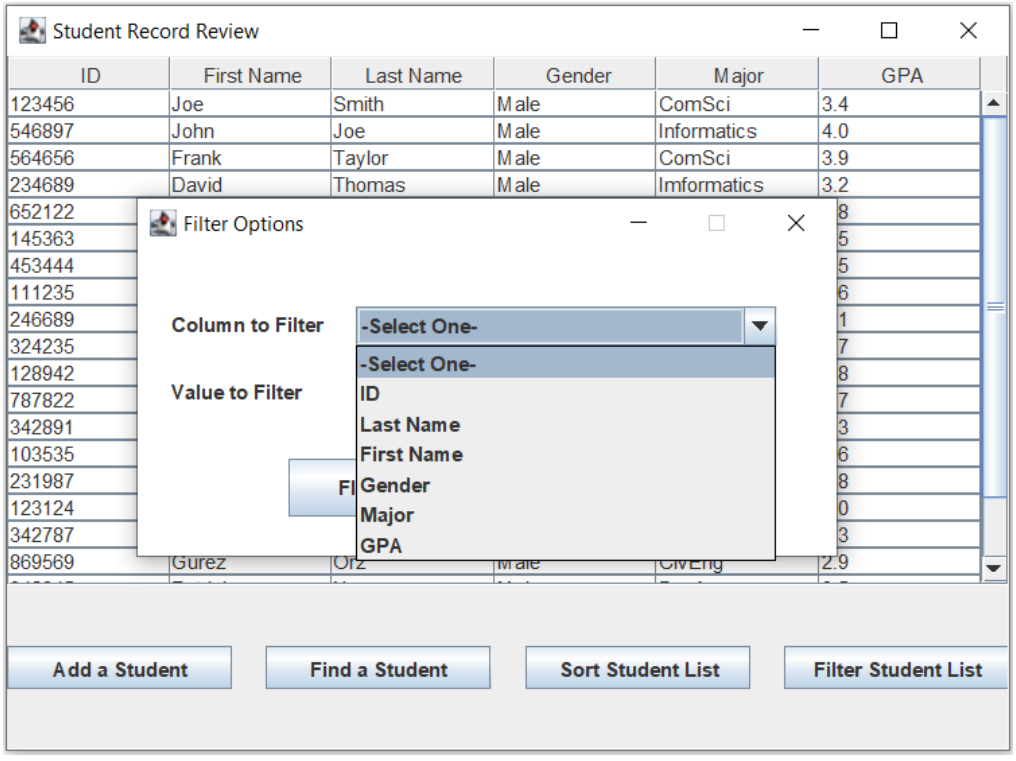


Figure 30

2. Choose the column and fill in the value to filter (Figure 31). If the user does not choose the column or fill in the value to filter, a corresponding error message will be displayed on the screen as figure 32 and 33.

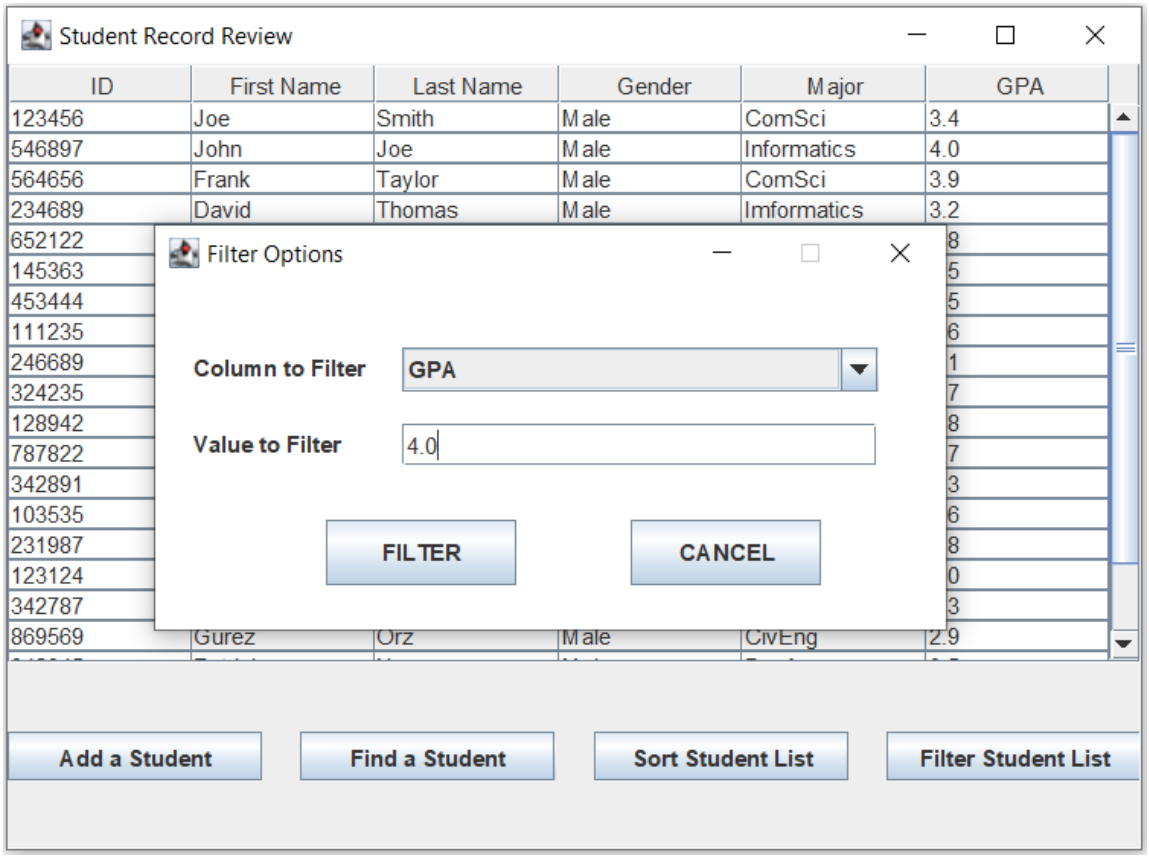


Figure 31

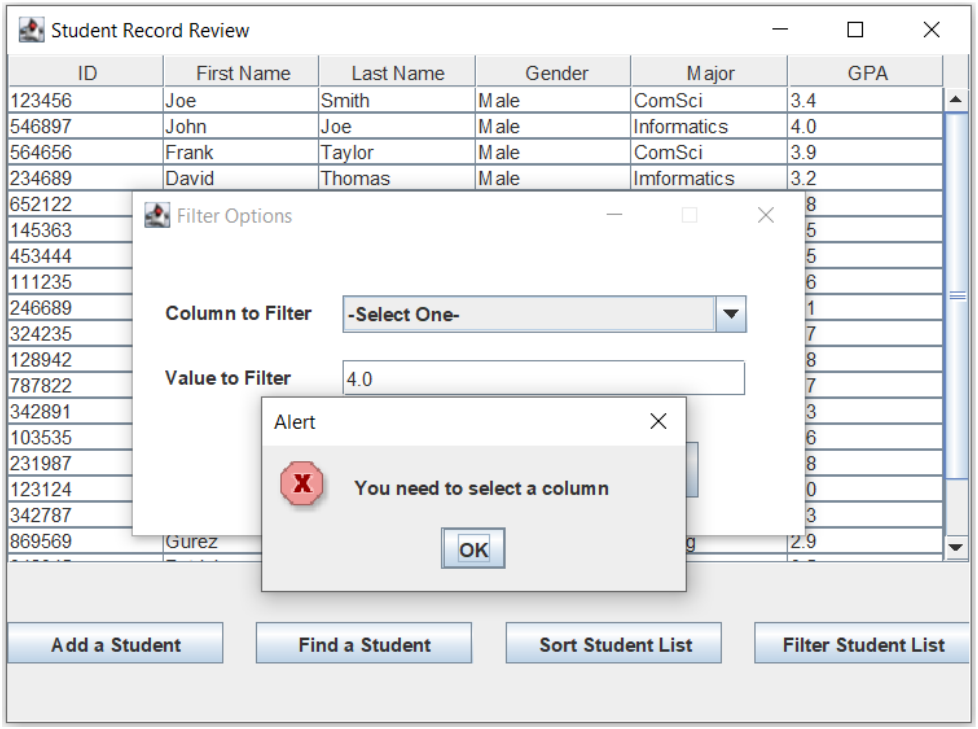


Figure 32

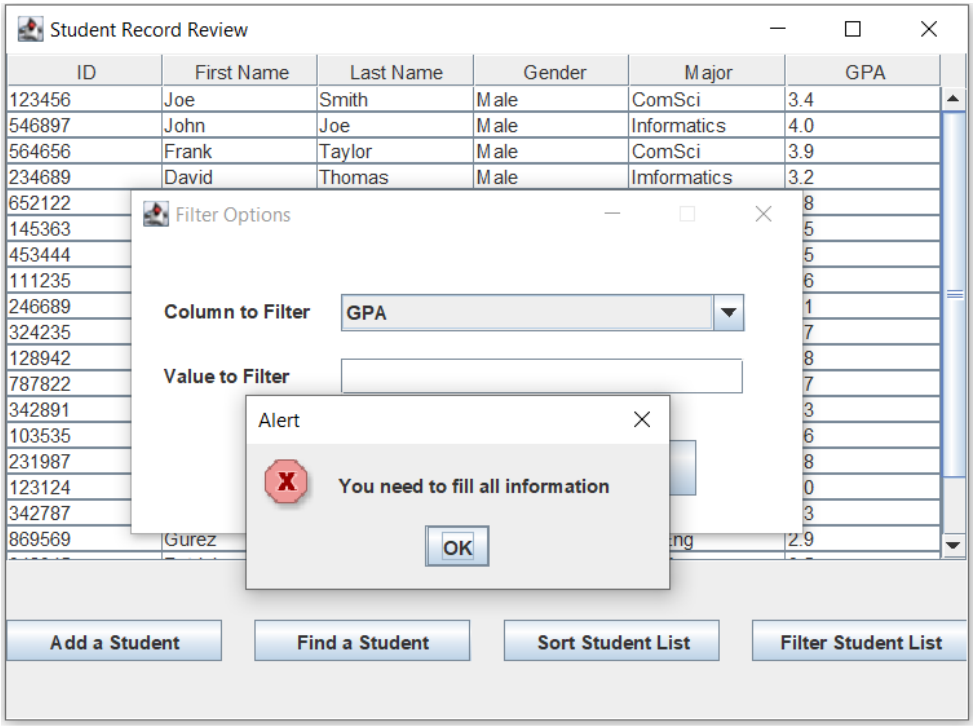


Figure 33

3. Click the “Filter” button in figure 31 to get the filtered table as the following figure 34.

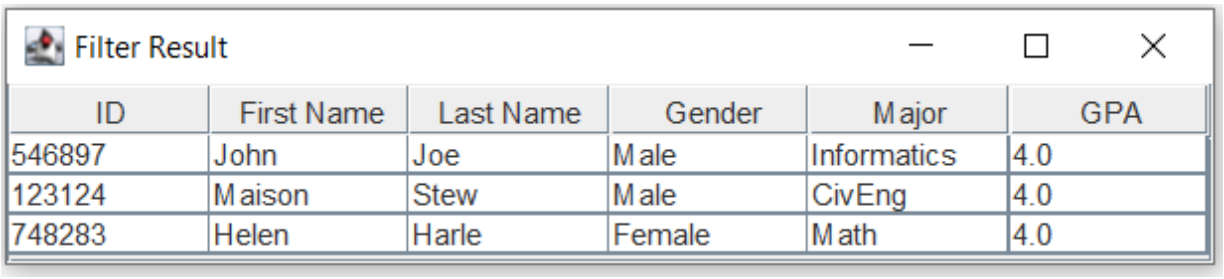


Figure 34

4. Click “X” button to close the filtered table and return to the “Student Record Review” window (Figure 16).

Importing the File

1. Click the “Import File” button from the main menu window of SRMS as in figure 15.
2. The import file window will show up as figure 35.

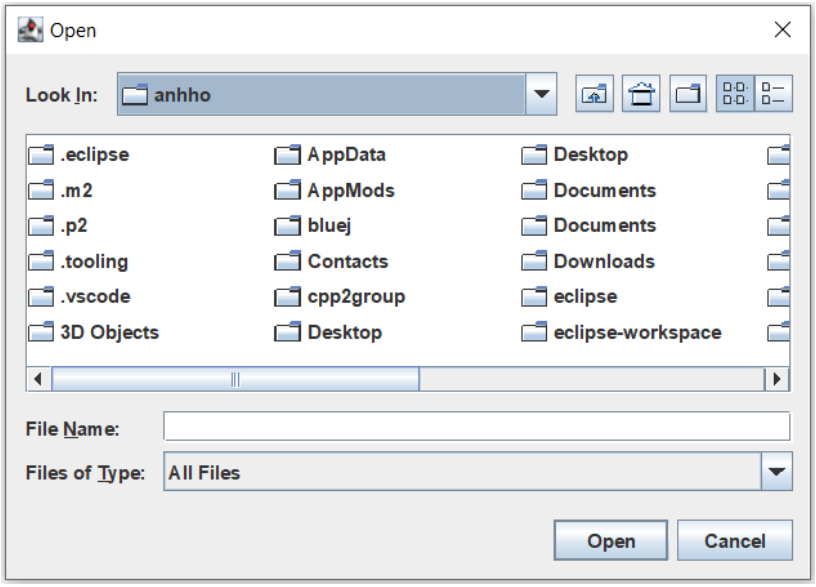


Figure 35

3. Click on “Look in” to find the location of the database file.
4. Select the appropriate data file, then click the “Open” button as the below figure 36.

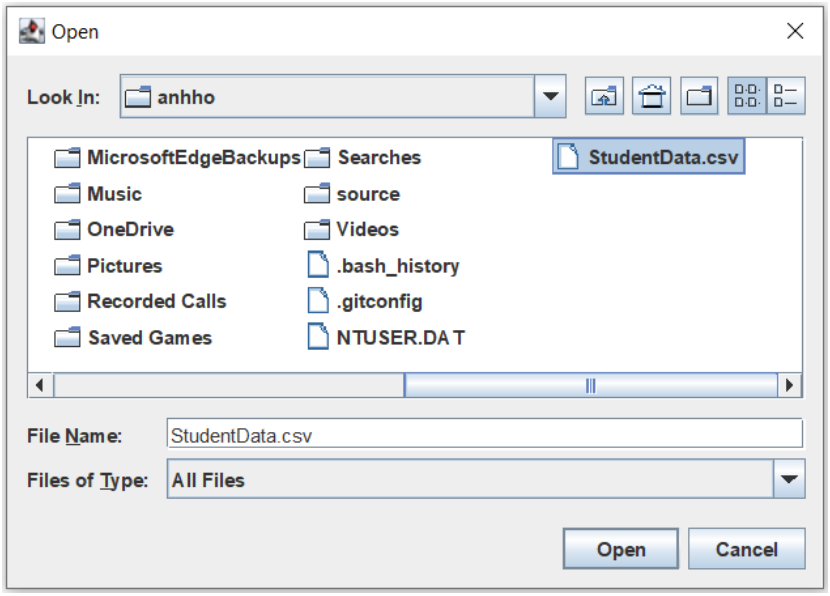


Figure 36

5. An error message will pop up if the user opens the wrong file (Figure 37)

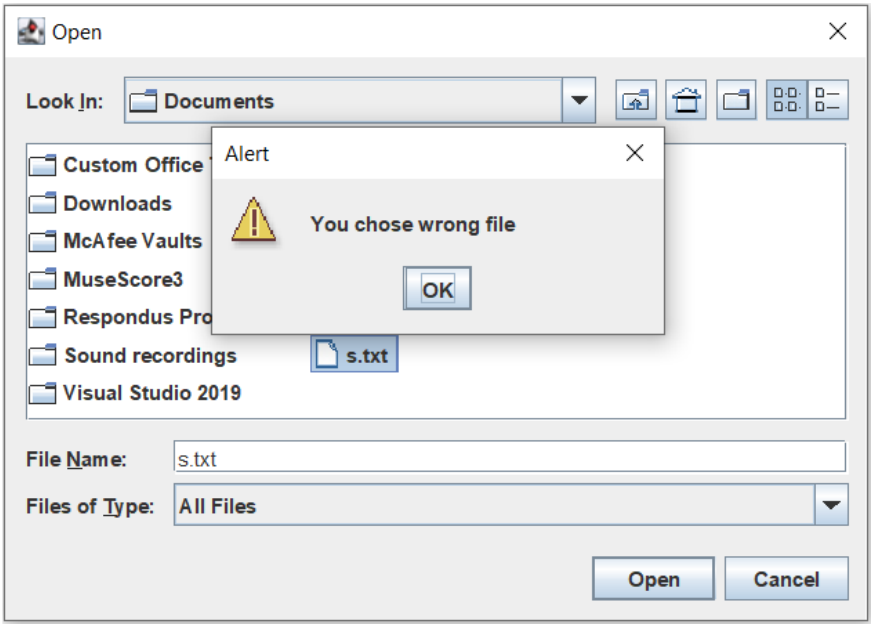


Figure 37

6. An info window will appear with the information of the selected file if user opens a correct file (Figure 38).

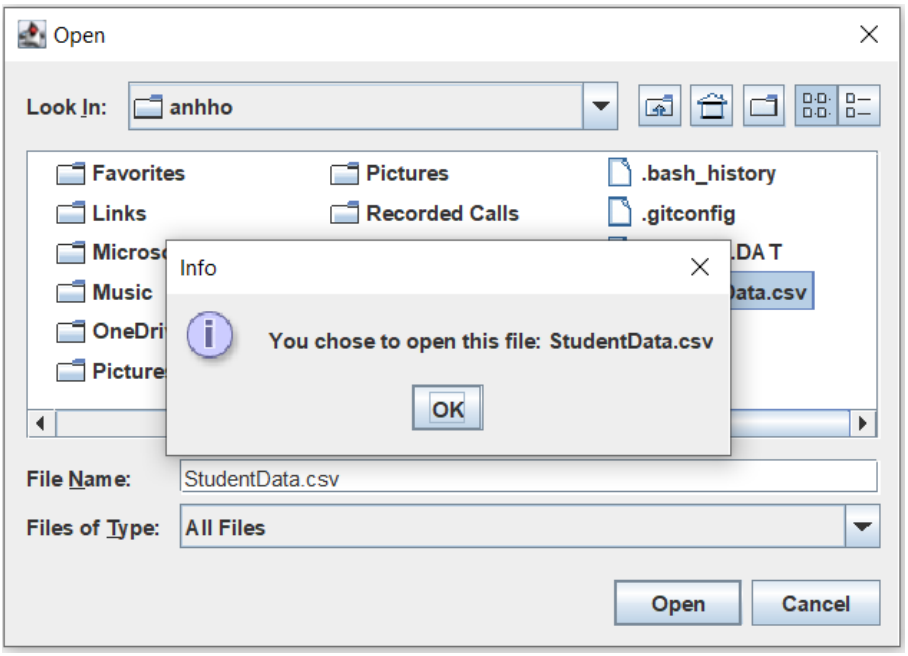


Figure 38

7. Click “OK” or “X” button to return to the main menu window.

Exporting the File

1. From the main menu window of SRMS as in figure 15, click the “Export File” button.
2. The export file window will appear (Figure 39).

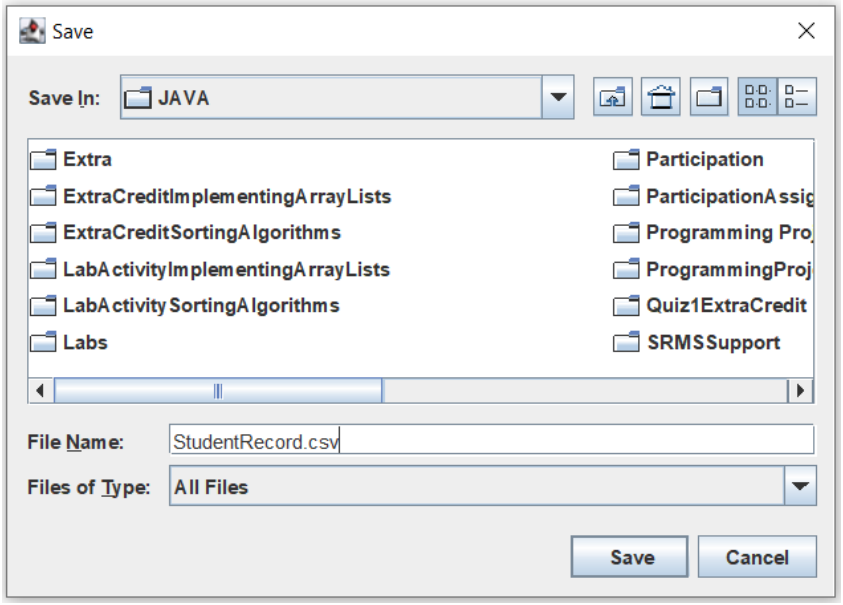


Figure 39

3. Click on “Save In” to choose where to save the database file.
4. Type the filename and click the “Save” button to save the file.
5. An info window will pop up and show the filename to be saved (Figure 40).
6. Click “OK” or “X” button to return to the main menu.

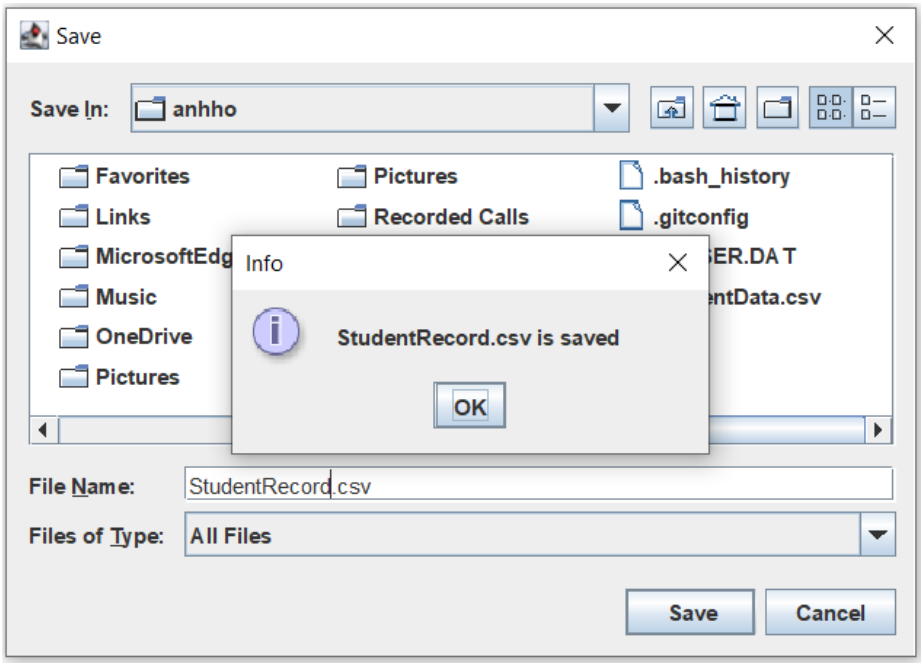


Figure 40

Activity Log

1. In the SRMS main menu (Figure 15), select the “Activity Log” button to access the activity-log table as the below figure 41.

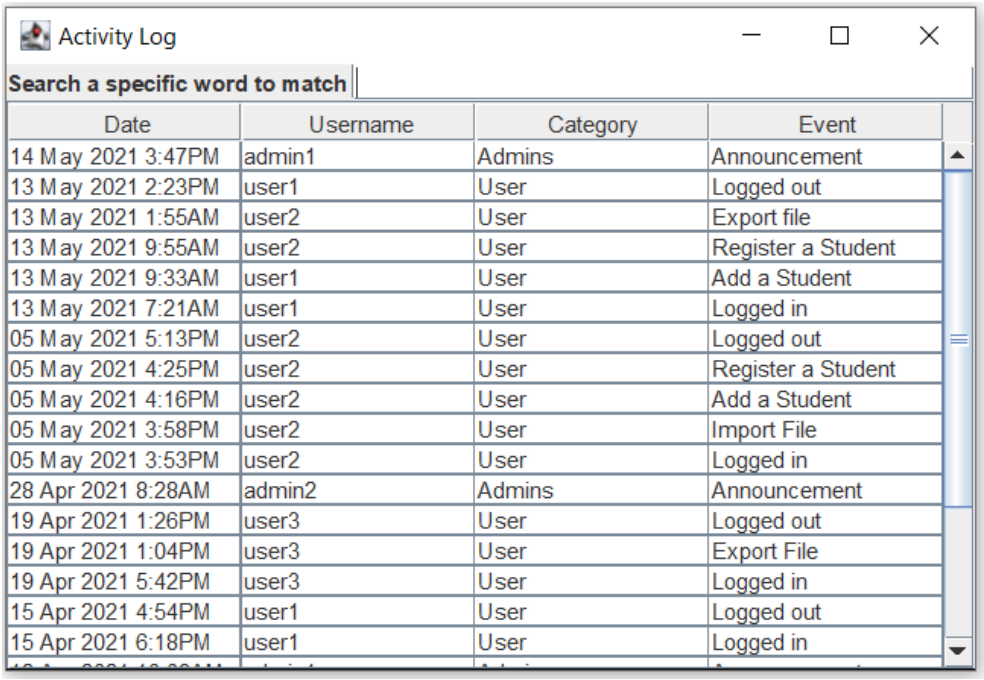


Figure 41

2. Using a key word to search for specific activity logs (Figure 42).

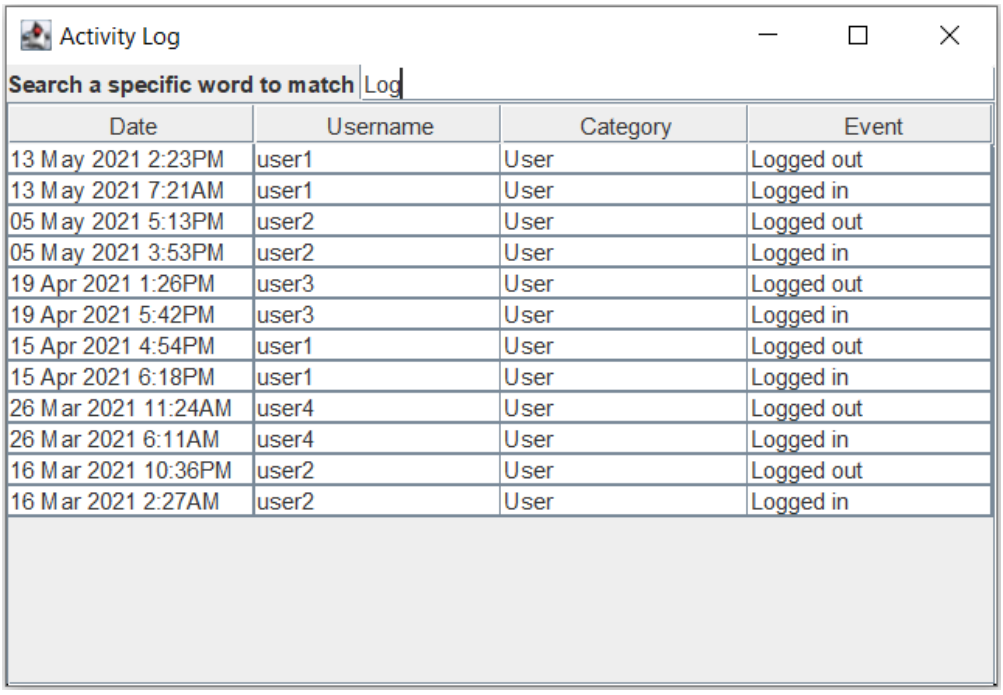


Figure 42

3. Click “X” button to return to the main menu window.

Viewing Announcements

1. When the system releases an announcement, the “Announcements” button in the SRMS main menu window (Figure 15) will be highlighted with red color. Next to it is the number of unread announcements (Figure 43).

Figure 43

2. Click the “Announcements” button.
3. An announcement table will pop up (Figure 44).

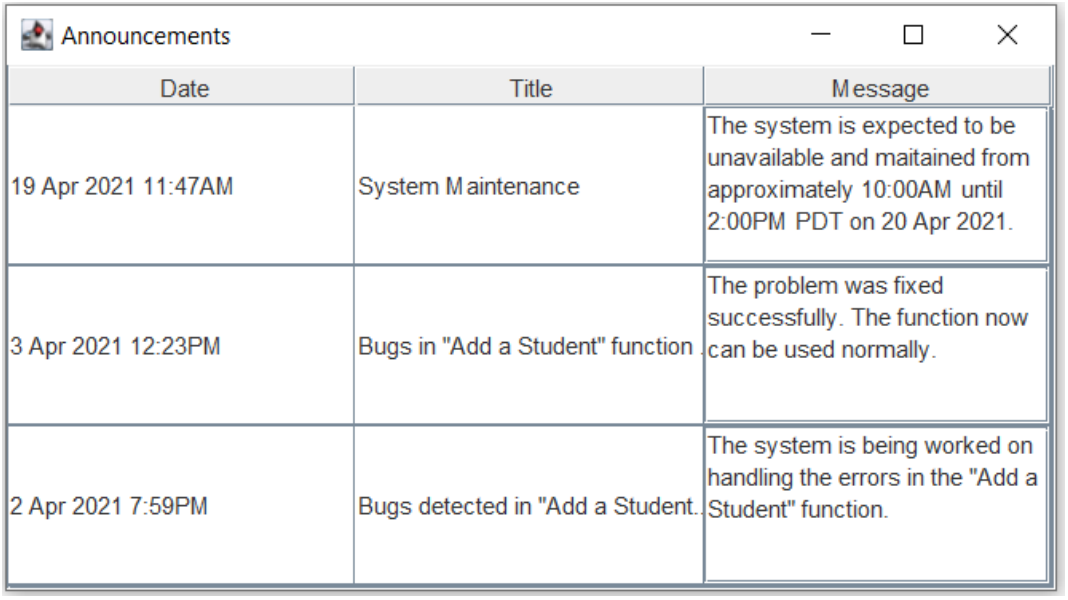


Figure 44

4. Click “X” button to return to the SRMS main menu.

Using Guidelines Button

1. In the main menu window of SRMS (Figure 15), click the “Guidelines” button. A “Guidelines” dialog will appear for a quick reference guide (Figure 45).
2. Click “OK” or “X” button to return to the main menu.

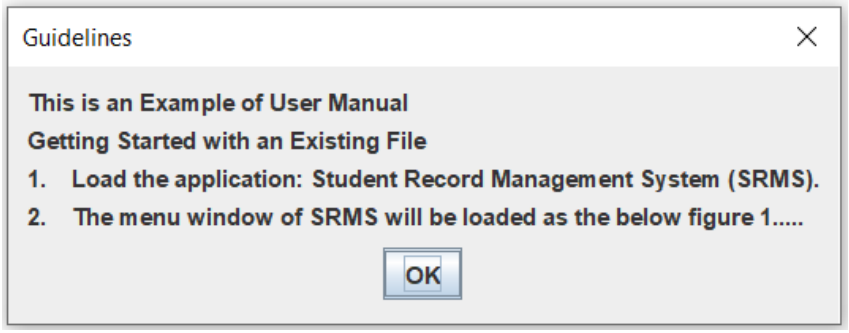


Figure 45

Using “Contact Us” button

1. Click the “Contact Us” button from the SRMS main menu (Figure 15). A “Contact Form” dialog will show up as the following figure 46 and 47.

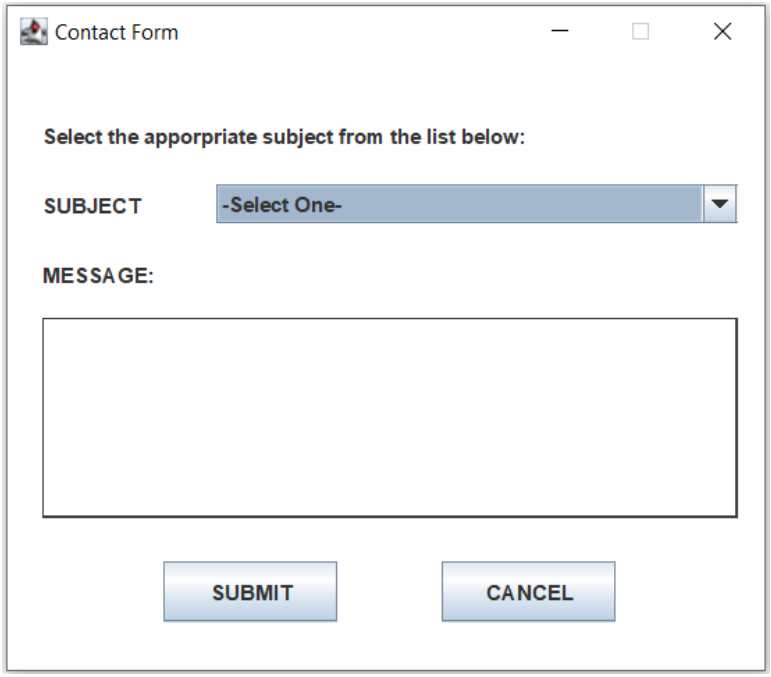


Figure 46

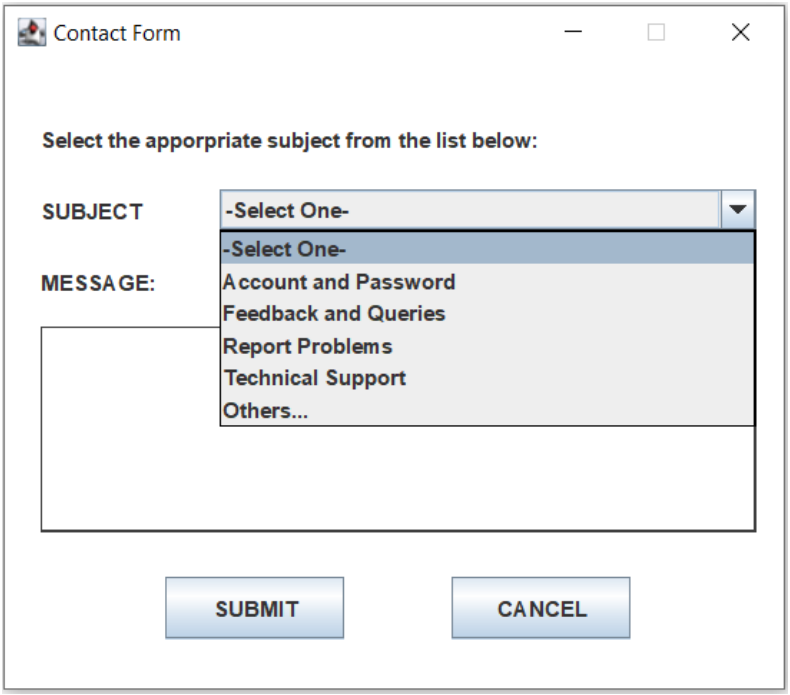


Figure 47

2. If either the “Subject” is not chosen or the “Message” field is not filled in, a corresponding error message will be displayed as figure 48 and 49.

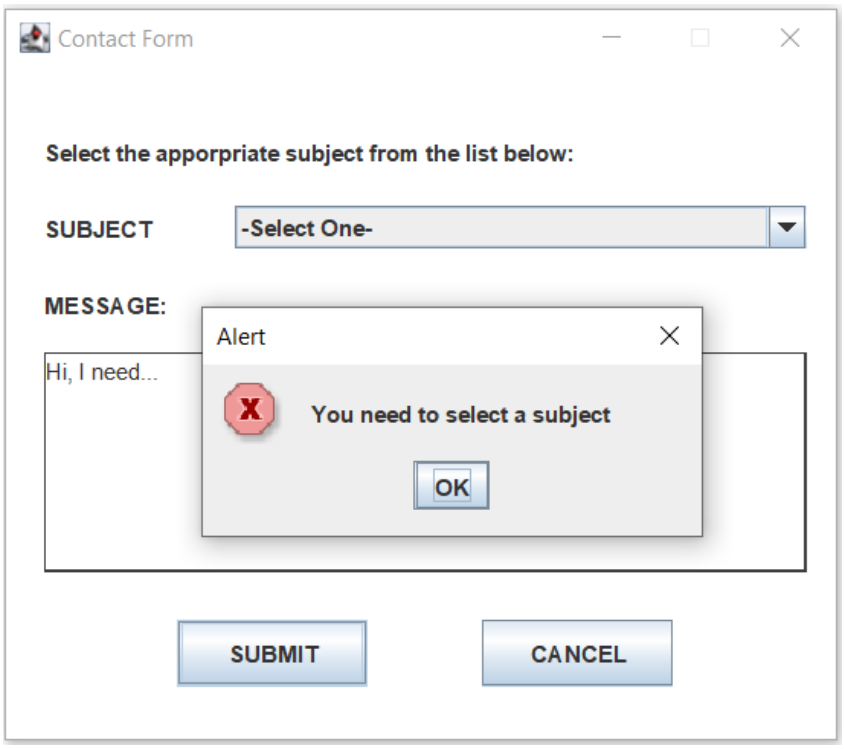


Figure 48

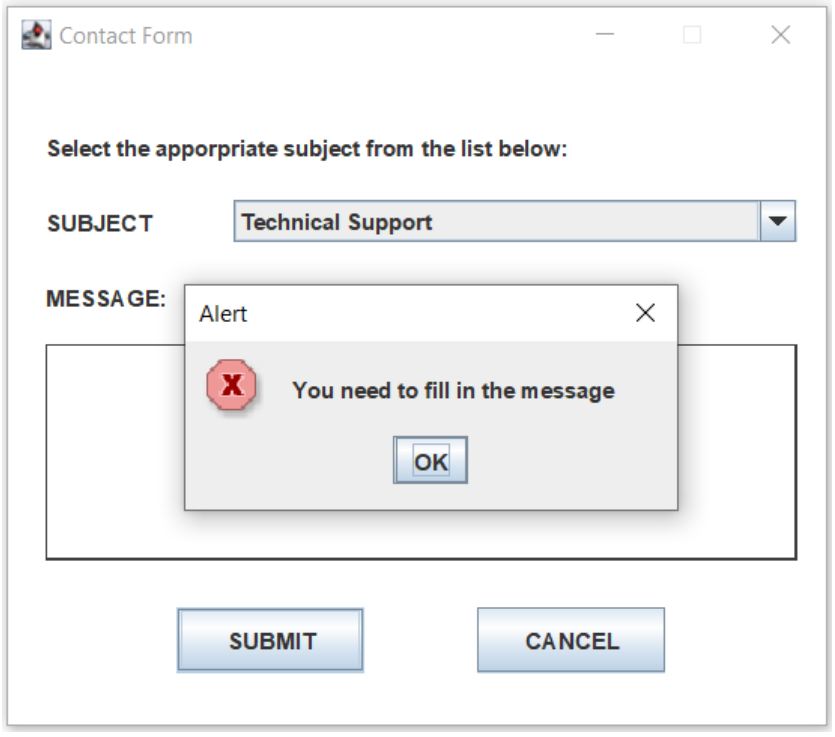


Figure 49

3. After the user fills in all the fields and click the “Submit” button, a successful submission message will appear (Figure 50).

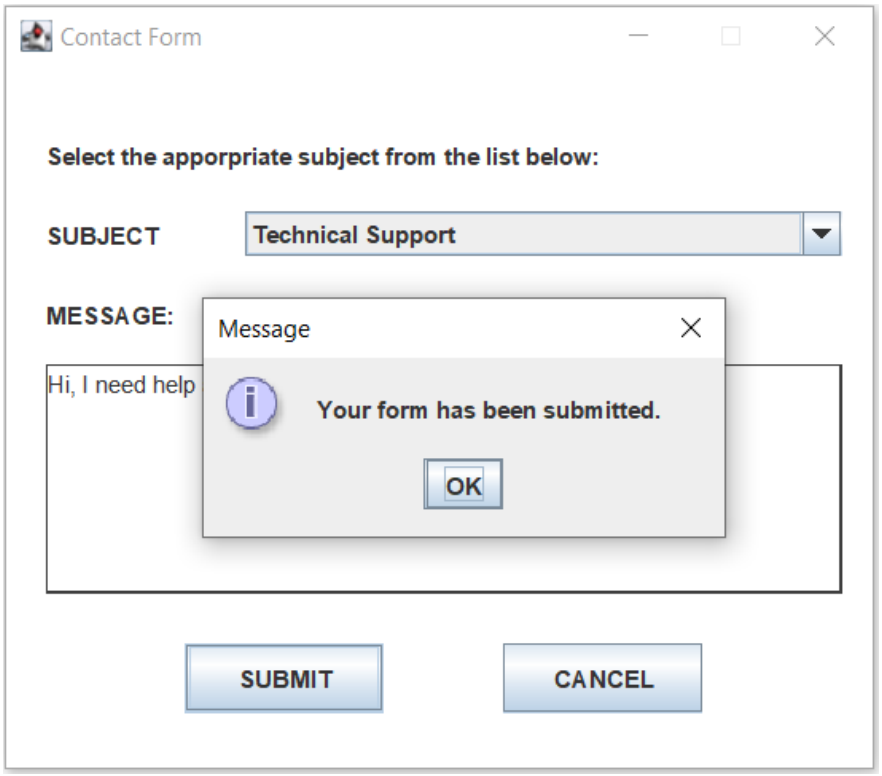


Figure 50

4. Click “OK” or “X” button to return to the main menu.

Using “Log Out” button

1. Click the “Log Out” or “X” button in the main menu of SMRS (Figure 15) to log out and exit the program completely.

References

N/A.