Orange Coast College

CS A220: Software Engineering

Student Record Management System

Team members: Harry Nguyen, Anh Vuong, Phu Nguyen, Khang Nguyen

Presentation Date: April 21, 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	3	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 Cases 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 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

Project Plan

- What kind of project is that?

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

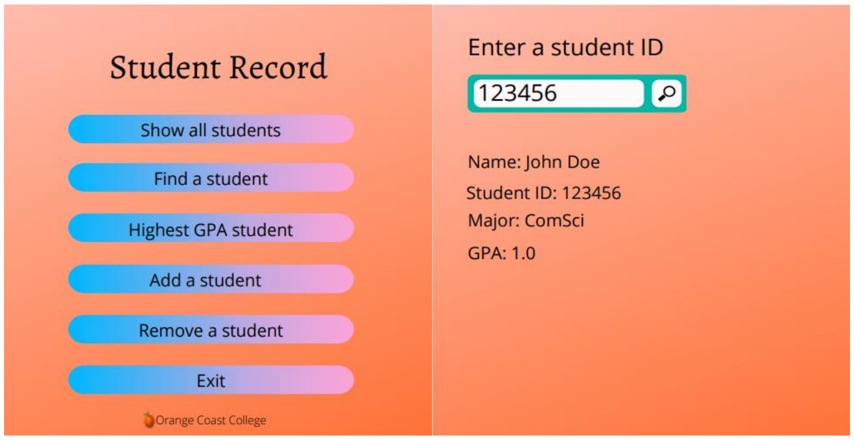
- What is the programming language you need to know?

  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.



Name: John Doe
Student ID: 123456
Major: ComSci
GPA: 1.0

- 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.

- 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.

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** registered user

   I want to reset my password

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

3. **As** a user

   I need to review the records of the students

   So that I can view or modify their data.

4. **As** a user

   I want to add a student record manually

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

5. **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.

6. **As** a user

   I need to sort the database

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

7. **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.

8. **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.

9. **As** a user

   I need a help button

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

10. **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:

<u>Success end condition</u>

User logs in the system successfully.

<u>Failure end condition</u>

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. Use case resumes on step 3.

Use Case: Reset password

ID: UC-02

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:

<u>Success end condition</u>

The user successfully changes the password.

<u>Failure end condition</u>

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

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

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

Use Case: Student record review.

ID: UC-03

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:

<u>Success end condition</u>

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

<u>Failure end condition</u>

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. Use case resumes on step 3.

Use Case: Add student

ID: UC-04

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 on the "Student Record Review" button.

User successfully clicks on the "Add a student" button.

Post conditions:

<u>Success end condition</u>

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

<u>Failure end condition</u>

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

User did not click on 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 on the "Student Record Review" first, then clicks on 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 on 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 on 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 on “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. Use case resumes on step 3.

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

1. User clicks the “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. 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 the "Cancel" button or just closes the window.
2. The system will display a "Data unsaved" message.
3. The use case resumes on step 11.

Use Case: Find student

ID: UC-05

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:

<u>Success end condition</u>

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

<u>Failure end condition</u>

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 the "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. Use case resumes on step 3.

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

1. User clicks the "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. Use case resumes on step 8.

Use Case: Sort data

ID: UC-06

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 on the “Student Record Review” button.

User successfully clicks on the “Sort Student List” button.

Post conditions:

<u>Success end condition</u>

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

<u>Failure end condition</u>

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 sort type which depends on the value data type of the selected column.
13. User clicks on the “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. Use case resumes on step 3.

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

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

Use Case: Import file

ID: UC-07

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:

<u>Success end condition</u>

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

<u>Failure end condition</u>

No file is imported into the system.

Trigger

User logs in the system and clicks on 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 on 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. Use case resumes on step 3.

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

1. User will click the “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. The use case resumes on step 9.

Use Case: Export file

ID: UC-08

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:

<u>Success end condition</u>

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

<u>Failure end condition</u>

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. Use case resumes on step 3.

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

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

Use Case: Use help button

ID: UC-9

Description:

A help 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 on the “Help” button.

Post conditions:

<u>Success end condition</u>

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

<u>Failure end condition</u>

No help guide is displayed to the user.

Trigger

User logs in the system and clicks on the “Help” 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 on the “Help” 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. Use case resumes on step 3.

Use Case: Log out

ID: UC-10

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:

<u>Success end condition</u>

User logs out and exits the system successfully.

<u>Failure end condition</u>

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.

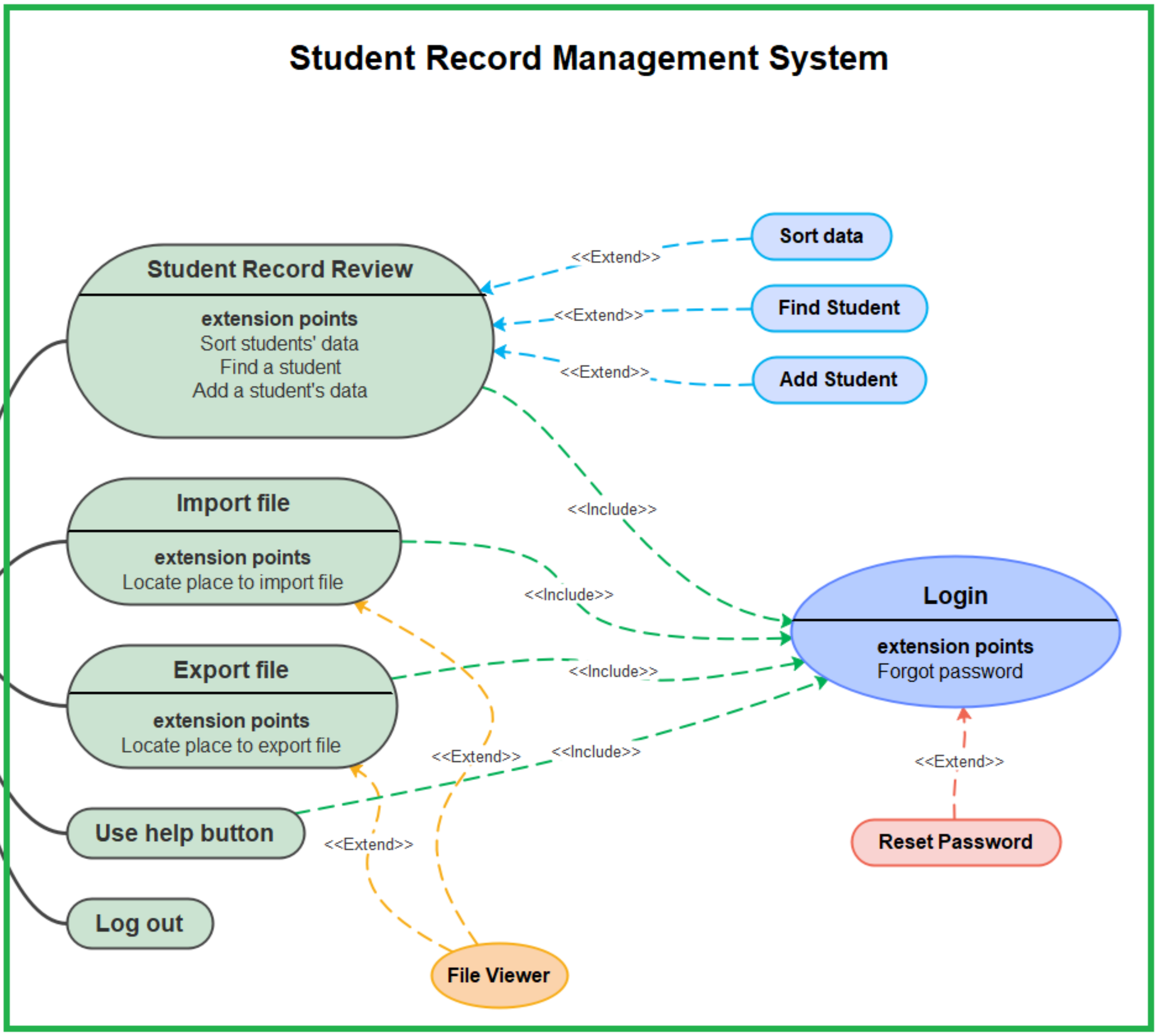
6. System displays a menu of functions.
7. User clicks the “Log out” button or just closes the window.
8. The user is logged out of the system, and the system is closed.

Extensions

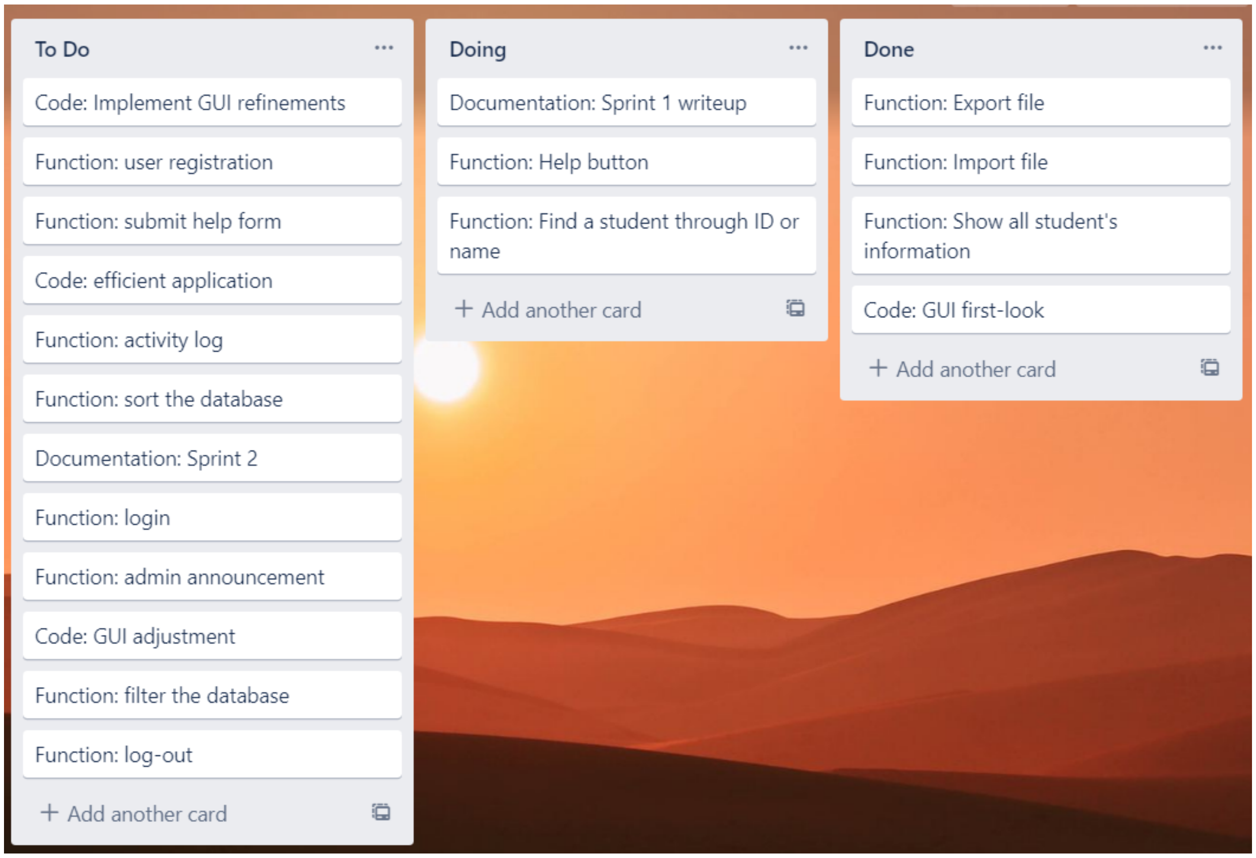
5a. In step 5, if the user enters the wrong login information:

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

Use Case Diagram



Sprint Backlog



To Do

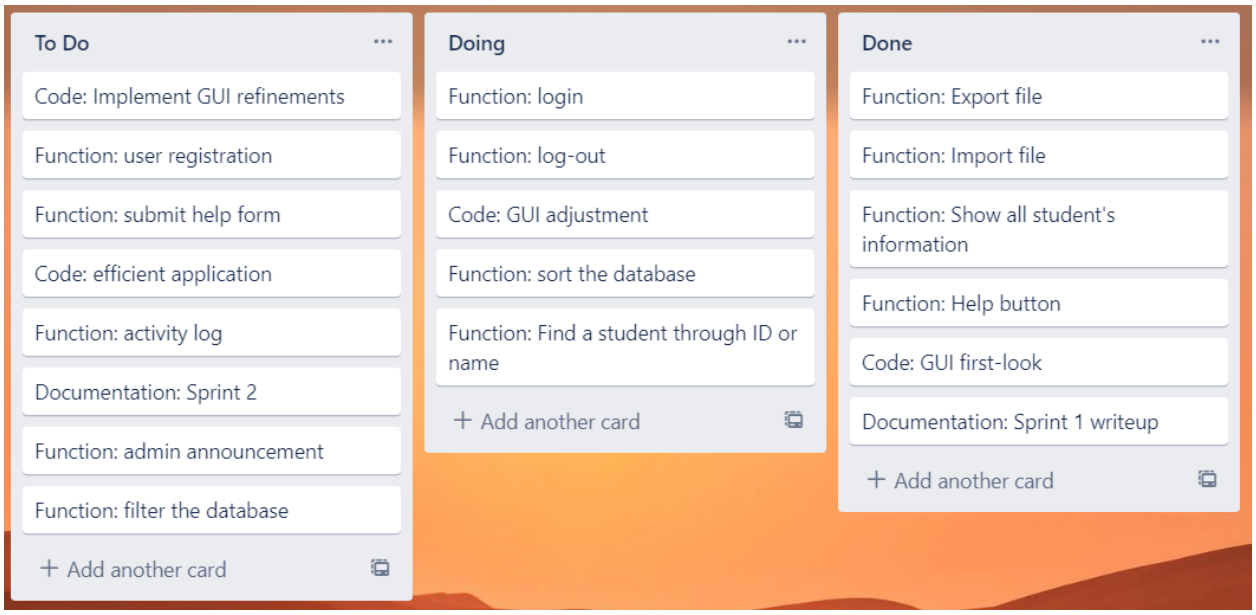
- Code: Implement GUI refinements
- Function: user registration
- Function: submit help form
- Code: efficient application
- Function: activity log
- Function: sort the database
- Documentation: Sprint 2
- Function: login
- Function: admin announcement
- Code: GUI adjustment
- Function: filter the database
- Function: log-out
- + Add another card

Doing

- Documentation: Sprint 1 writeup
- Function: Help button
- Function: Find a student through ID or name
- + Add another card

Done

- Function: Export file
- Function: Import file
- Function: Show all student's information
- Code: GUI first-look
- + Add another card



To Do

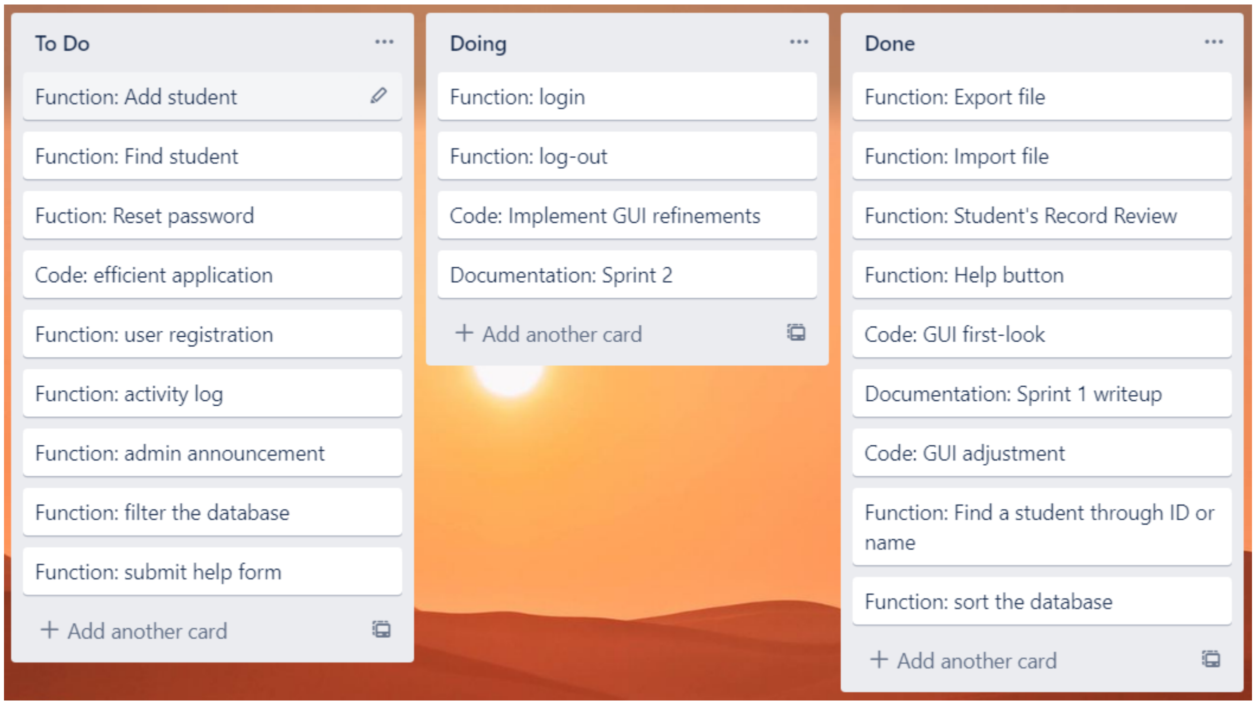
- Code: Implement GUI refinements
- Function: user registration
- Function: submit help form
- Code: efficient application
- Function: activity log
- Documentation: Sprint 2
- Function: admin announcement
- Function: filter the database
- + Add another card

Doing

- Function: login
- Function: log-out
- Code: GUI adjustment
- Function: sort the database
- Function: Find a student through ID or name
- + Add another card

Done

- Function: Export file
- Function: Import file
- Function: Show all student's information
- Function: Help button
- Code: GUI first-look
- Documentation: Sprint 1 writeup
- + Add another card



To Do

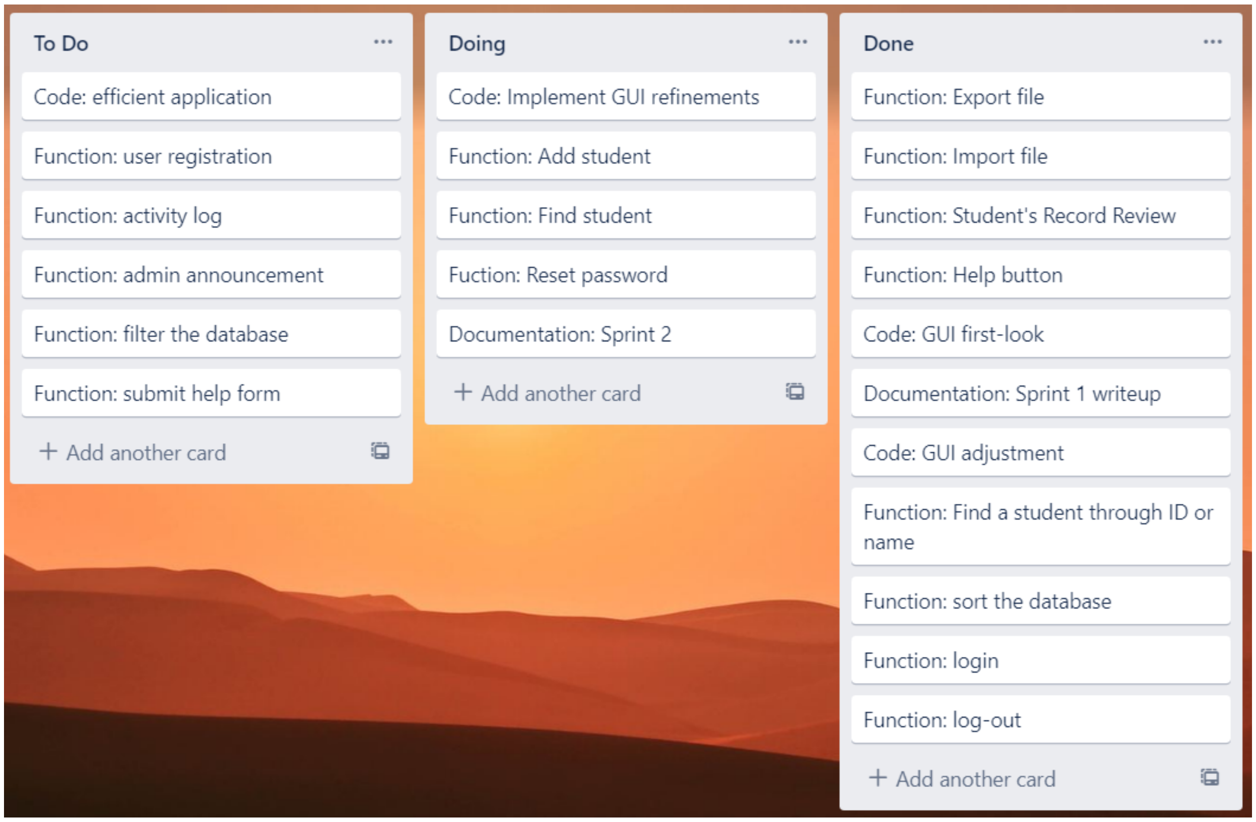
- Function: Add student
- Function: Find student
- Fuction: Reset password
- Code: efficient application
- Function: user registration
- Function: activity log
- Function: admin announcement
- Function: filter the database
- Function: submit help form

Doing

- Function: login
- Function: log-out
- Code: Implement GUI refinements
- Documentation: Sprint 2

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



To Do

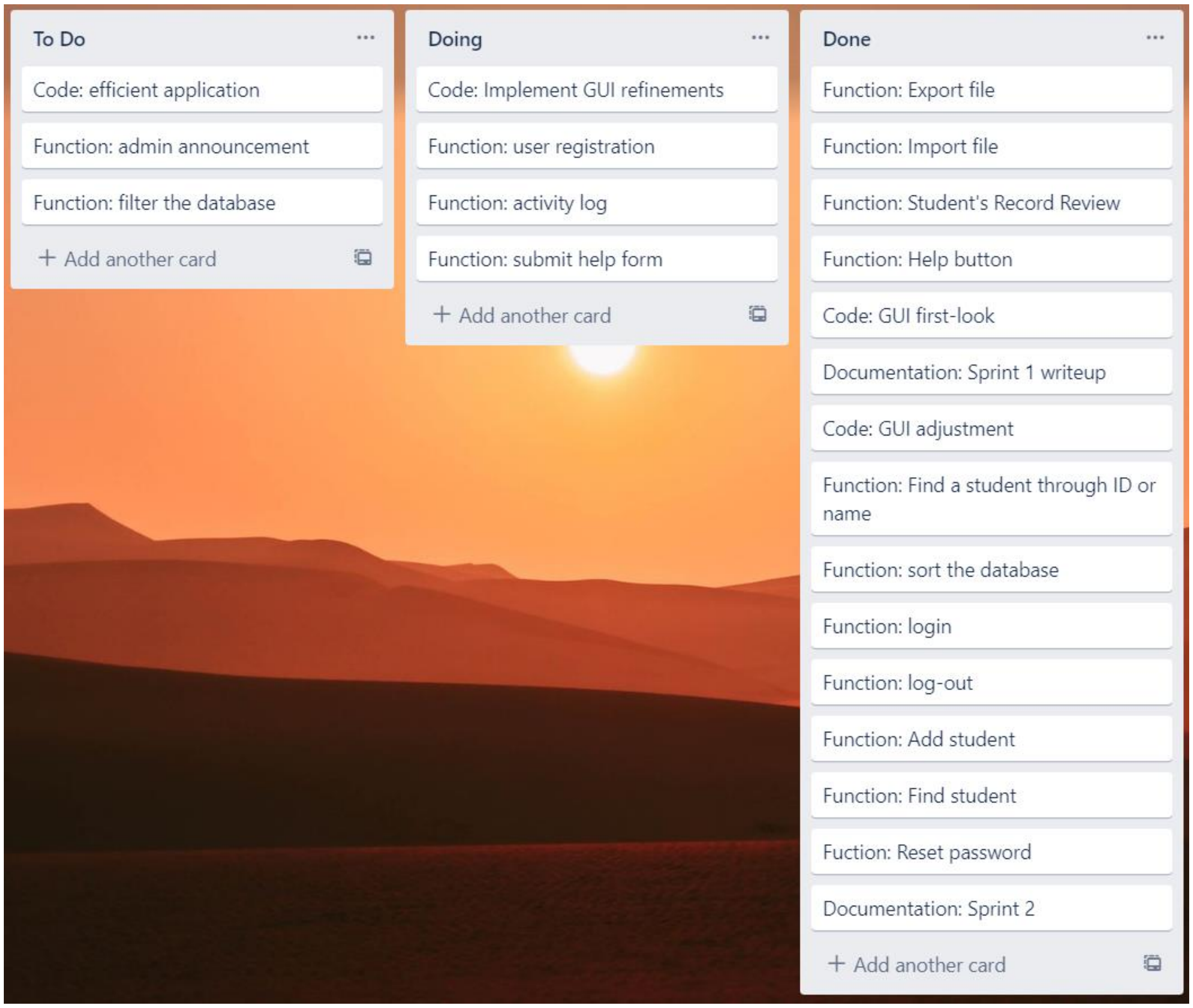
- Code: efficient application
- Function: user registration
- Function: activity log
- Function: admin announcement
- Function: filter the database
- Function: submit help form

Doing

- Code: Implement GUI refinements
- Function: Add student
- Function: Find student
- Fuction: Reset password
- Documentation: Sprint 2

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



To Do

- Code: efficient application
- Function: admin announcement
- Function: filter the database

Doing

- Code: Implement GUI refinements
- Function: user registration
- Function: activity log
- Function: submit help form

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

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 student's 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, creating a function that displays a table of student data from an imported database, together with the assisting tools to manage the student records, is our first priority. The second milestone is to make sure that users are able to access and interact easily with the GUI and all the features.

Requirements:

- Ability to access and display the student's database.
- Ability to manage and edit student records.
- Ability to import the database file.
- Ability to export the current database file.

Project deliverables:

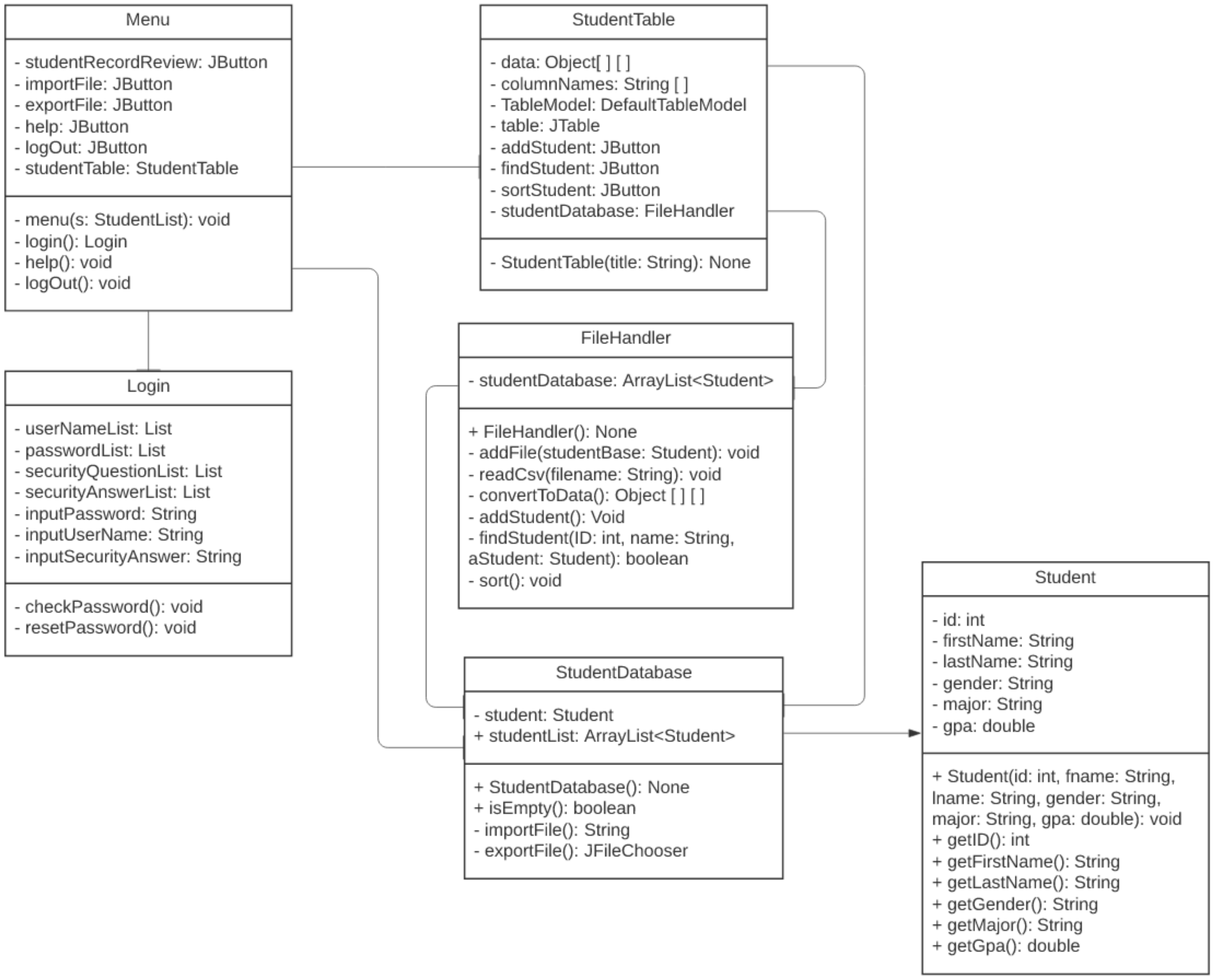
- User manual.

- Coded with Java language.
- Fully functional GUI.

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 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 that 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 Help Log out

Responsibilities	**Collaborators**
• Call login() to check user login information • Let user access the student record review functionality. • Let user access the Import fuctionality. • Let user access the Export fuctionality. • Let user access the Help fuctionality. • Call help() to display guidelines for a quick reference guide. • Call logOut() to sign out of the system.	StudentTable Login StudentDatabase

Attributes: studentRecordReview: JButton importFile: JButton exportFile: JButton help: 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 log in to the system or to reset user's password.		**Associated Use Cases:** Log in Reset Password

Responsibilities	**Collaborators**
• Check the inputed username and password before letting user log in to the system. • Let user reset the password in case user forgets the current one.	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
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 fuctionality. • Let user access the Find Student fuctionality. • Let user access the Sort data fuctionality.	Menu FileHandler StudentDatabase	

Attributes: data: Object[] [] columnNames: String [] TableModel: DefaultTableModel table: JTable addStudent: JButton findStudent: JButton sortStudent: 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, or sorting the table.		**Associated Use Cases:** Import file Export file Add student Find student Sort 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.	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

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	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

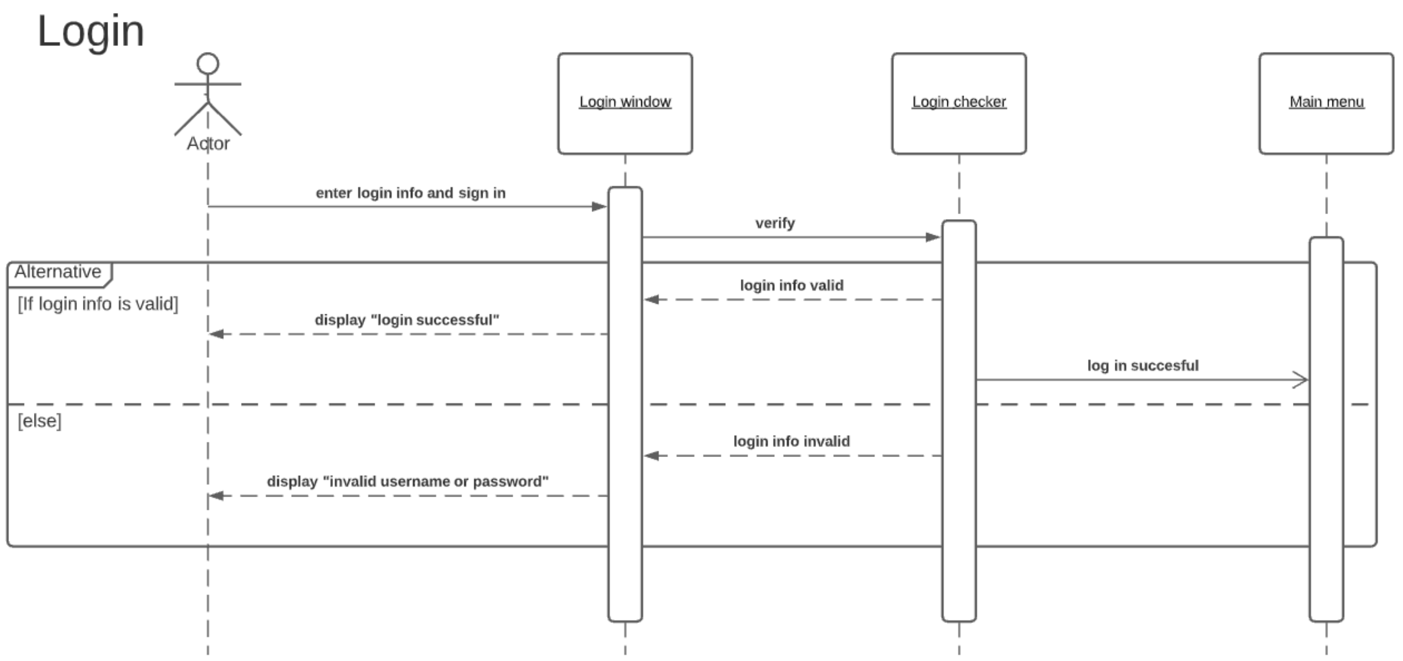
Responsibilities	**Collaborators**

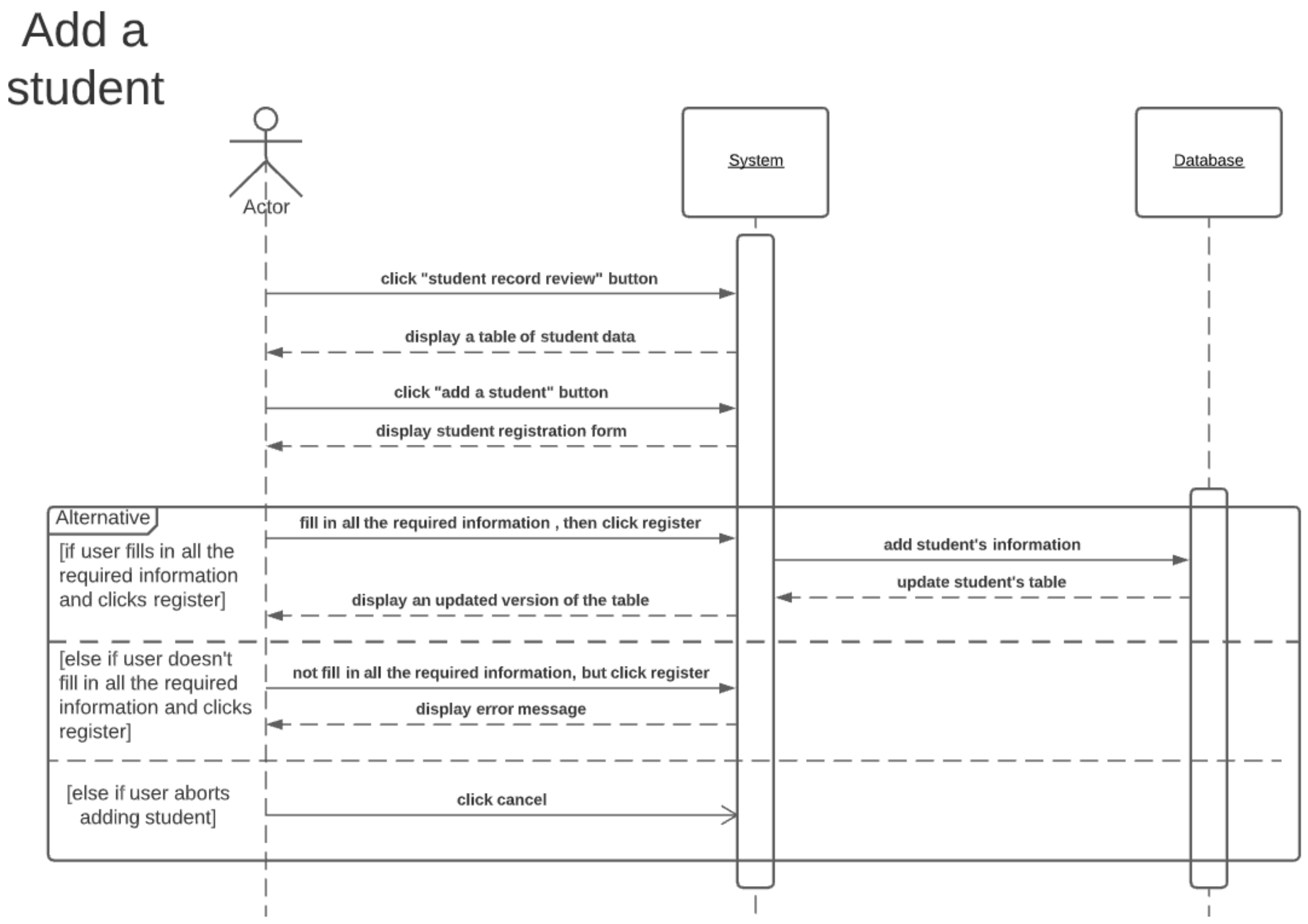
• Hold the student's record information (First name, Last name, Gender, ID, Major, GPA.)
• Release student's record information when being queried:
getID()
getFirstName()
getLastName()
getGender()
getMajor()
getGPA()
	StudentDatabase

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

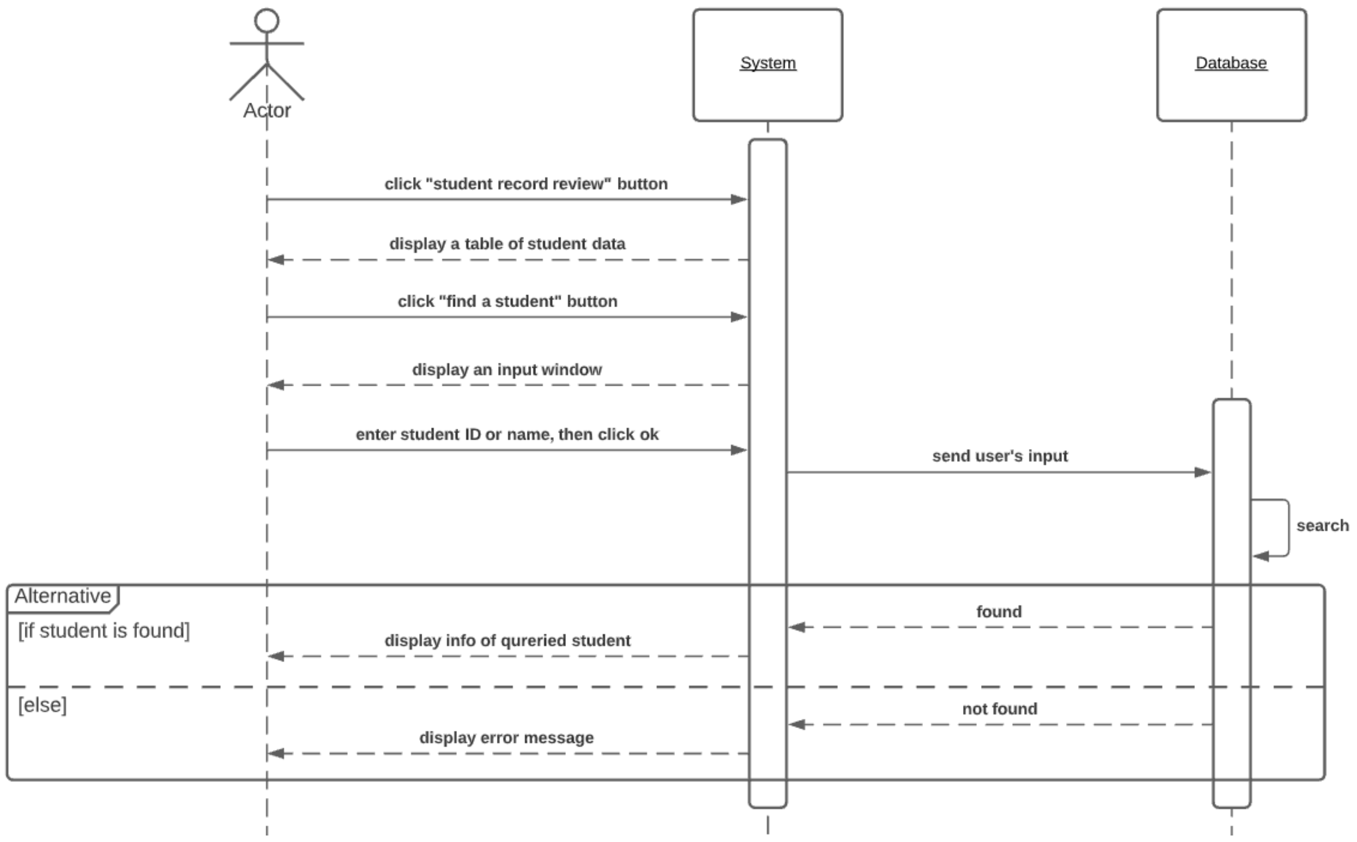
Relationships:
Generalization (a-kind-of):
Aggregation (has-parts):
Other Associations: StudentDatabase

Sequence Diagrams

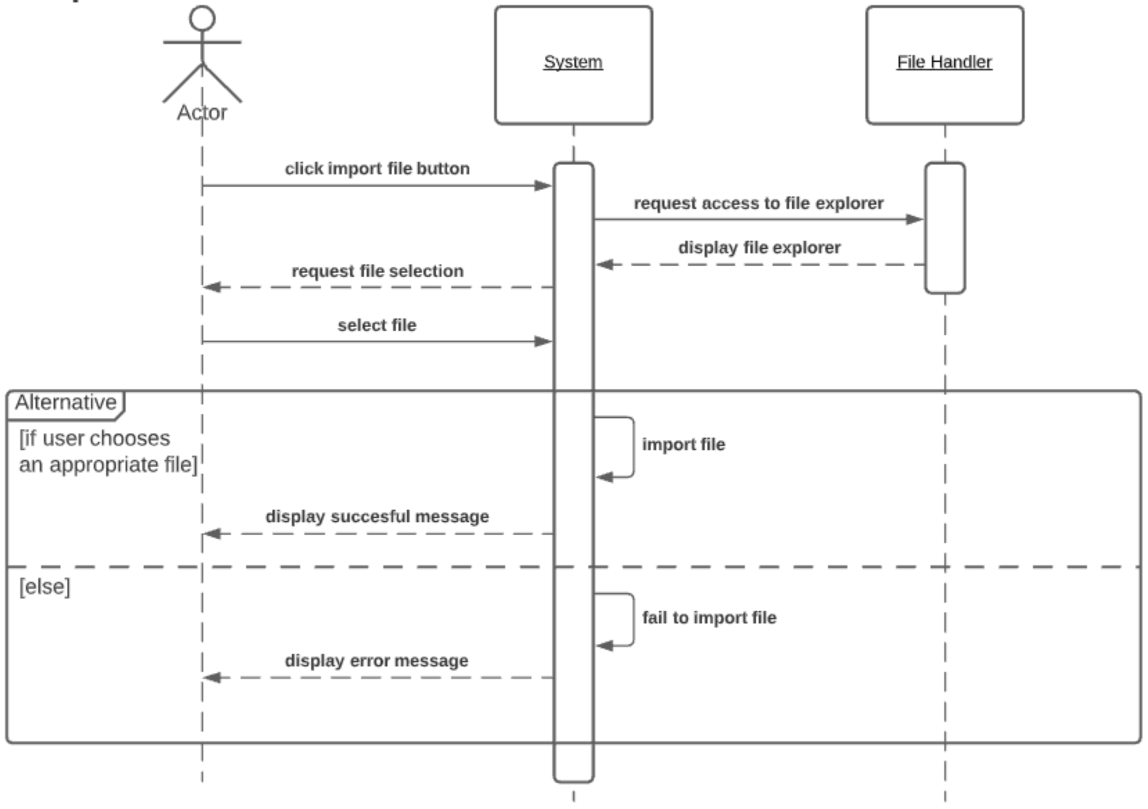



Add a student
Actor
System
Database
click "student record review" button
display a table of student data
click "add a student" button
display student registration form
Alternative
[if user fills in all the required information and clicks register]
fill in all the required information , then click register
add student's information
update student's table
display an updated version of the table
[else if user doesn't fill in all the required information and clicks register]
not fill in all the required information, but click register
display error message
[else if user aborts adding student]
click cancel

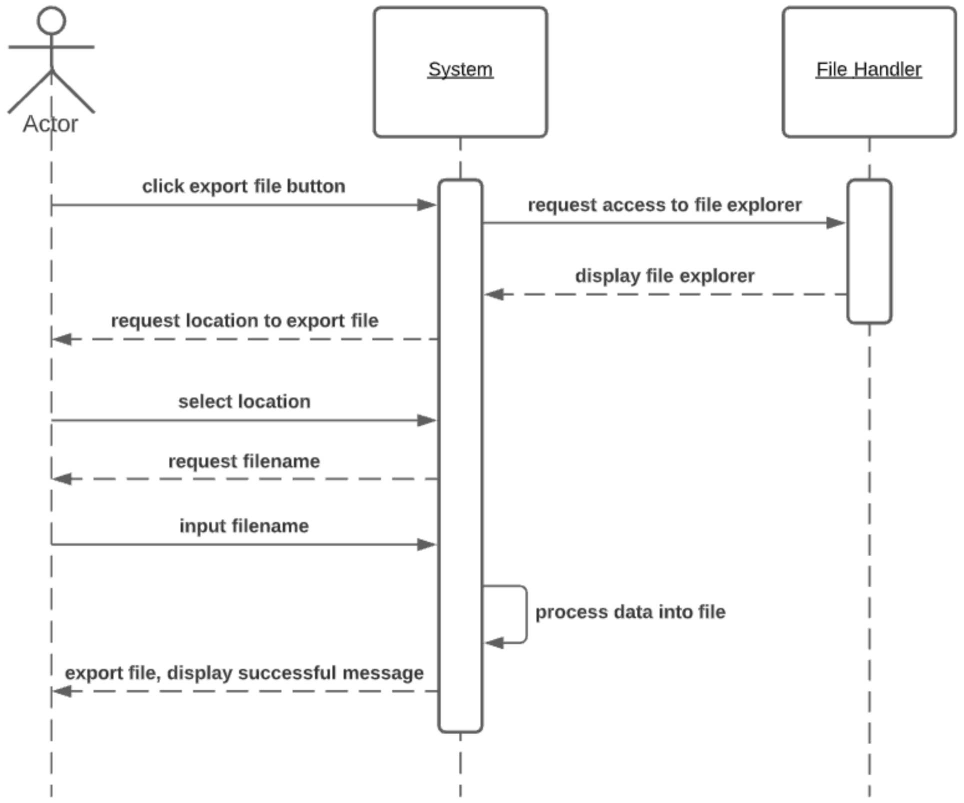
Find a student


Actor
System
Database
click "student record review" button
display a table of student data
click "find a student" button
display an input window
enter student ID or name, then click ok
send user's input
search
Alternative
[if student is found]
found
display info of queried student
[else]
not found
display error message

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.

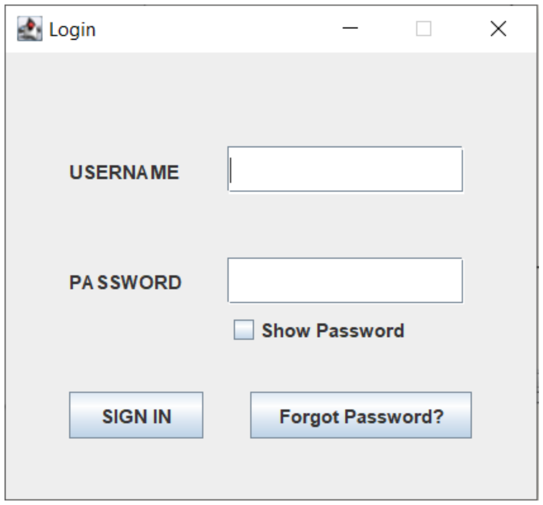
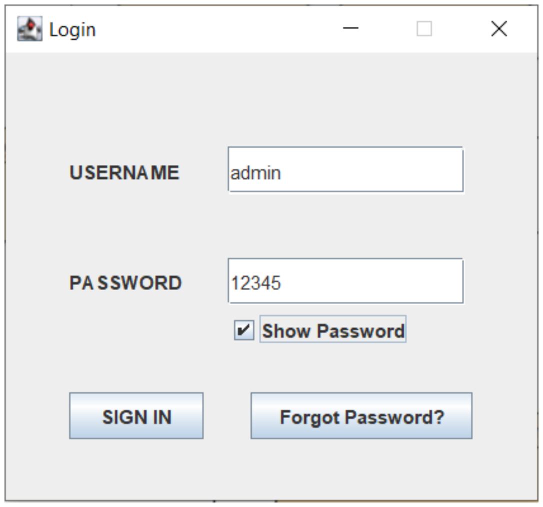


Figure 1

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



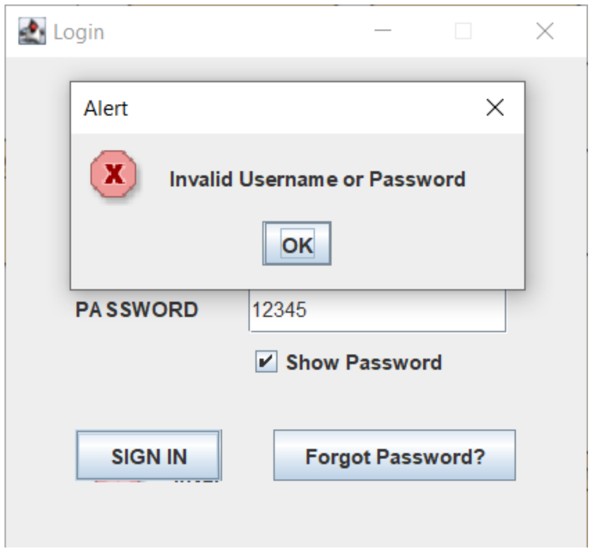
USERNAME admin

PASSWORD 12345

☑ Show Password

Figure 2

4. An error message will pop up if the username or password is invalid (Figure 3) or a successful login message will appear (Figure 4).


Login

Alert

Invalid Username or Password

OK

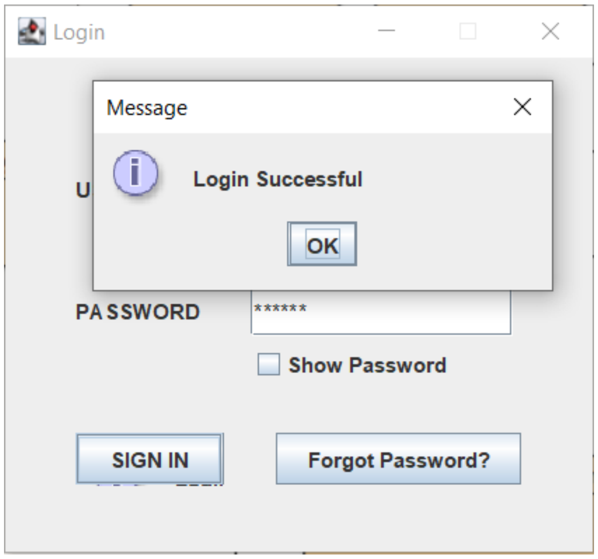
PASSWORD 12345

✓ Show Password

SIGN IN

Forgot Password?

Figure 3


Login

Message

Login Successful

OK

PASSWORD ******

Show Password

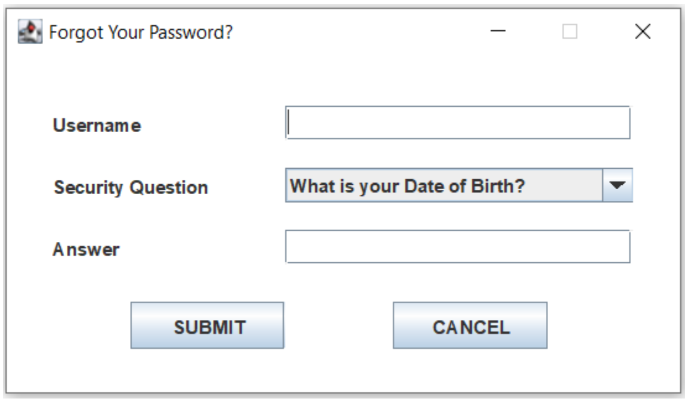
SIGN IN

Forgot Password?

Figure 4

Resetting the password

1. Click on “Forgot Password?” button and fill in the security information (username, security question and answer) as the below figure 5 and 6, then clicks on “Submit” button to get the approval to reset the password.

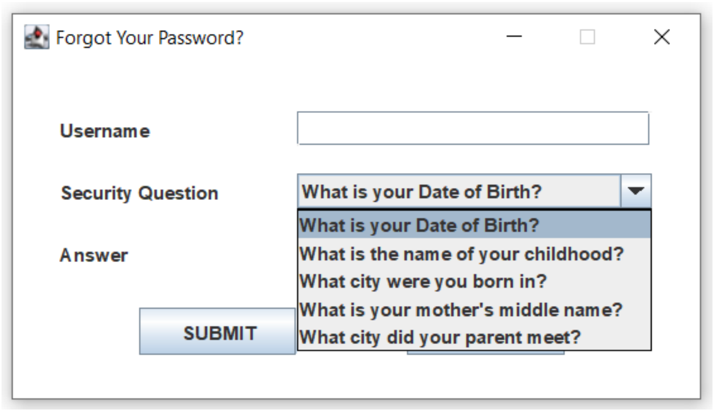


Username

Security Question: What is your Date of Birth?

Answer

Figure 5



Username

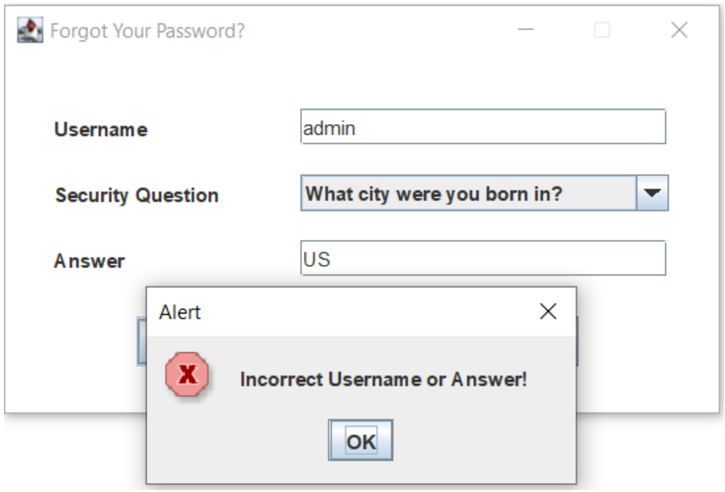
Security Question: What is your Date of Birth?

What is your Date of Birth?
What is the name of your childhood?
What city were you born in?
What is your mother's middle name?
What city did your parent meet?

Answer

Figure 6

2. After submitting the security information, an error message dialog will pop up as the below figure 7.



Username: admin

Security Question: What city were you born in?

Answer: US

Incorrect Username or Answer!

Figure 7

3. After the security information is verified, the “Reset Password” dialog will show up (Figure 8). If the password and the confirm password do not match, an error message appears (Figure 9) and vice versa, a successful message is displayed as figure 10.

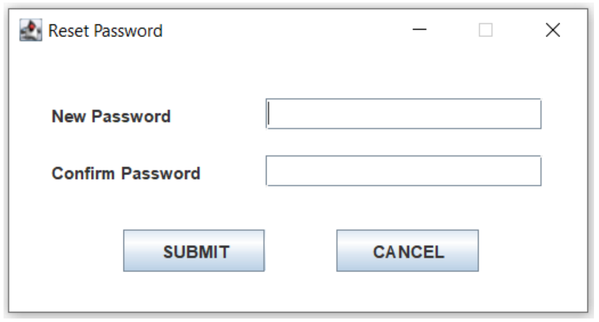
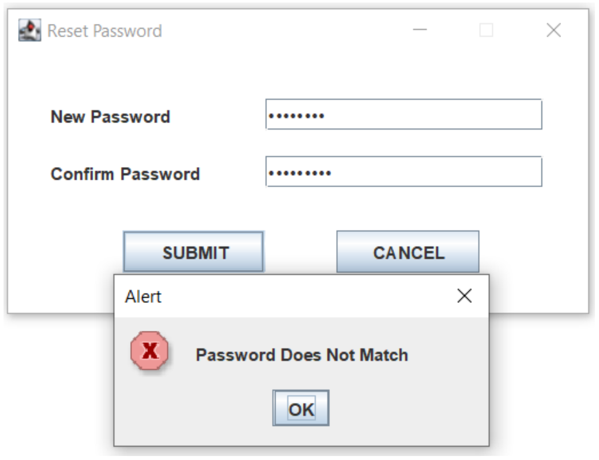
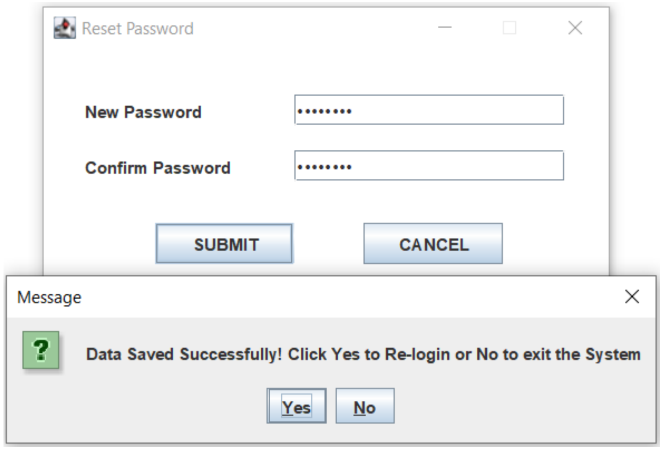


Figure 8



Password Does Not Match

Figure 9



Data Saved Successfully! Click Yes to Re-login or No to exit the System

Figure 10

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

Getting Started with an Existing Database file

1. The menu window of SRMS will be loaded after user log in to the system successfully as the below figure 11.

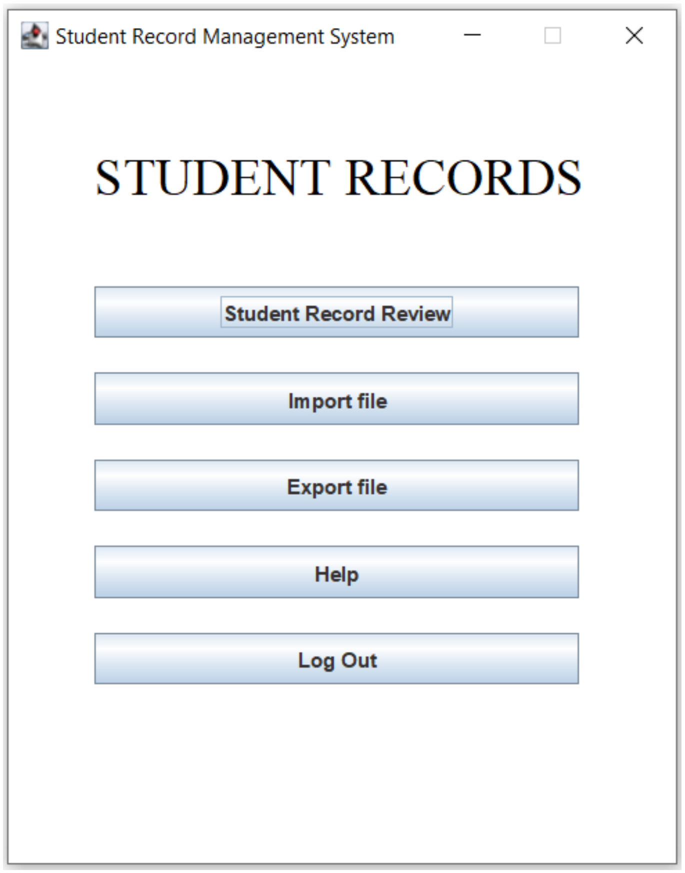
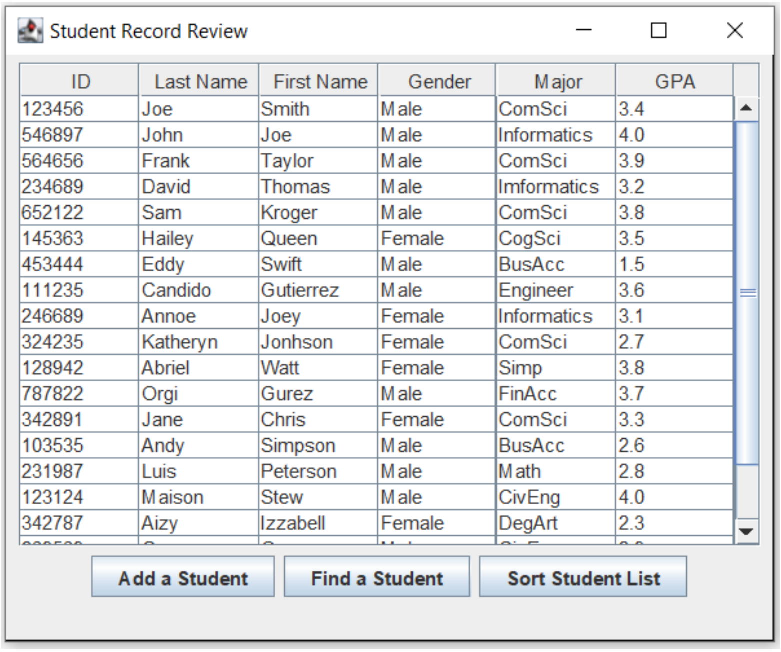


Figure 11

Using Student Record Review

1. From the menu window of SRMS as in the above figure 11, click on “Student Record Review” button.
2. A data table of all students in the database with columns of ID, first name, last name, gender, major, and GPA will appear as figure 12.



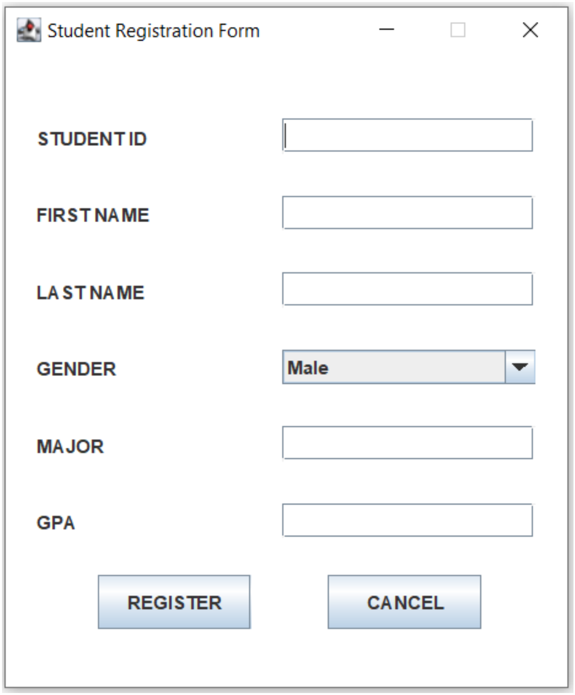
Student Record Review

ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Hailey	Queen	Female	CogSci	3.5
453444	Eddy	Swift	Male	BusAcc	1.5
111235	Candido	Gutierrez	Male	Engineer	3.6
246689	Annoe	Joey	Female	Informatics	3.1
324235	Katheryn	Jonhson	Female	ComSci	2.7
128942	Abriel	Watt	Female	Simp	3.8
787822	Orgi	Gurez	Male	FinAcc	3.7
342891	Jane	Chris	Female	ComSci	3.3
103535	Andy	Simpson	Male	BusAcc	2.6
231987	Luis	Peterson	Male	Math	2.8
123124	Maison	Stew	Male	CivEng	4.0
342787	Aizy	Izzabell	Female	DegArt	2.3

Figure 12

Adding a Student

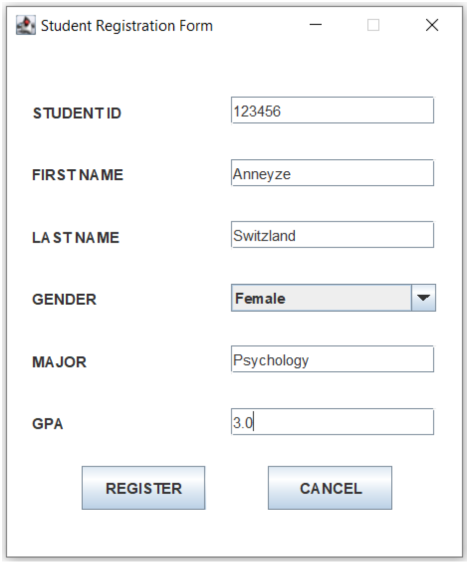
1. From “Student Record Review” as the above figure 12, click on “Add a Student” button to add more students into the existing list. A “Student Registration Form” dialog will appear (Figure 13 and 14).



Student Registration Form

STUDENT ID	
FIRST NAME	
LAST NAME	
GENDER	Male
MAJOR	
GPA	

Figure 13

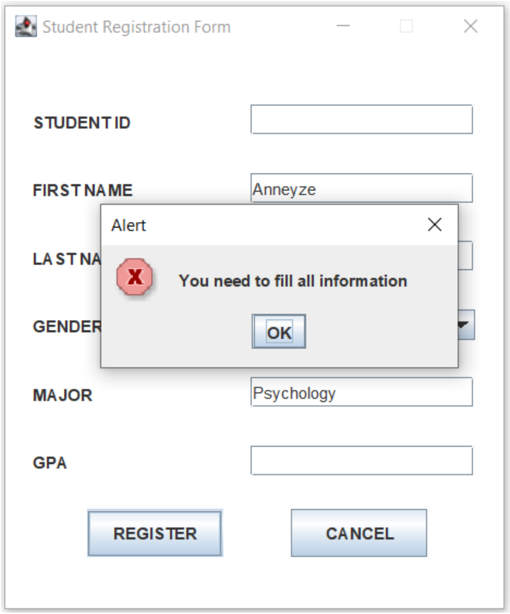


Student Registration Form

STUDENT ID	123456
FIRST NAME	Anneyze
LAST NAME	Switzland
GENDER	Female
MAJOR	Psychology
GPA	3.0

Figure 14

2. Click the “Cancel” button to cancel adding students and go back to the “Student Record Review” window.
3. Click “Register” button 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 15. Otherwise, a “Student Information” confirmation dialog will pop up (Figure 16).



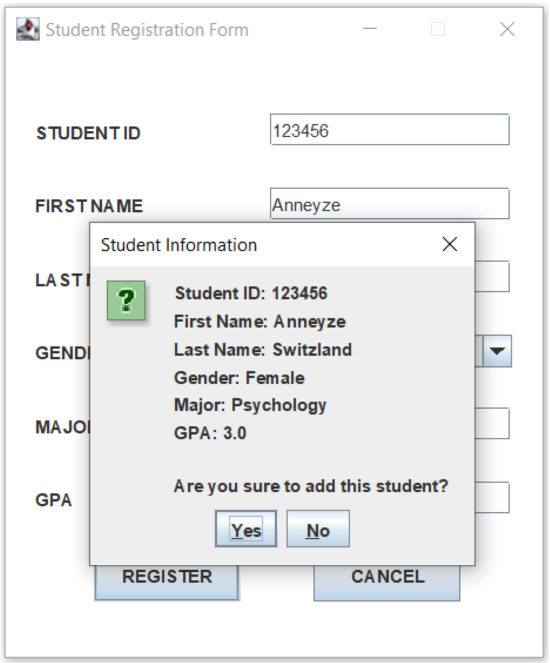
Student Registration Form

STUDENT ID	
FIRST NAME	Anneyze
MAJOR	Psychology
GPA	

Alert

You need to fill all information

Figure 15


Student Registration Form

STUDENT ID 123456

FIRST NAME Anneyze

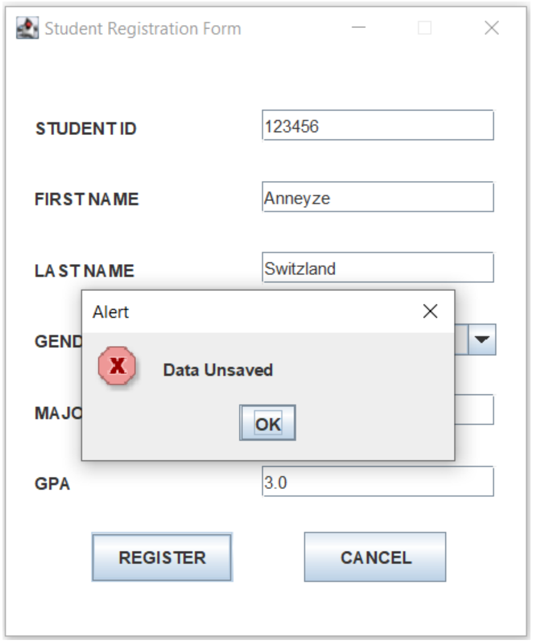
Student Information

Student ID: 123456
First Name: Anneyze
Last Name: Switzland
Gender: Female
Major: Psychology
GPA: 3.0

Are you sure to add this student?

Figure 16

5. Click “No” to review and correct information (Figure 17) or click “Yes” to save the student information (Figure 18).


Student Registration Form

STUDENT ID 123456

FIRST NAME Anneyze

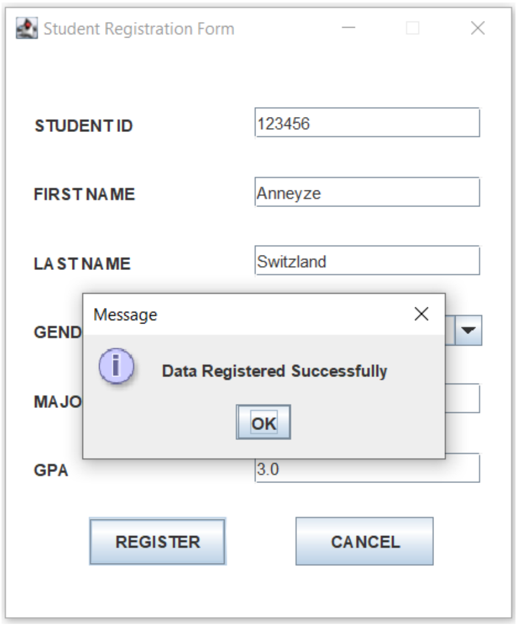
LAST NAME Switzland

GPA 3.0

Alert

Data Unsaved

Figure 17



Student Registration Form

STUDENT ID	123456
FIRST NAME	Anneyze
LAST NAME	Switzland
GEND	
MAJO	
GPA	3.0

Message

Data Registered Successfully

OK

REGISTER CANCEL

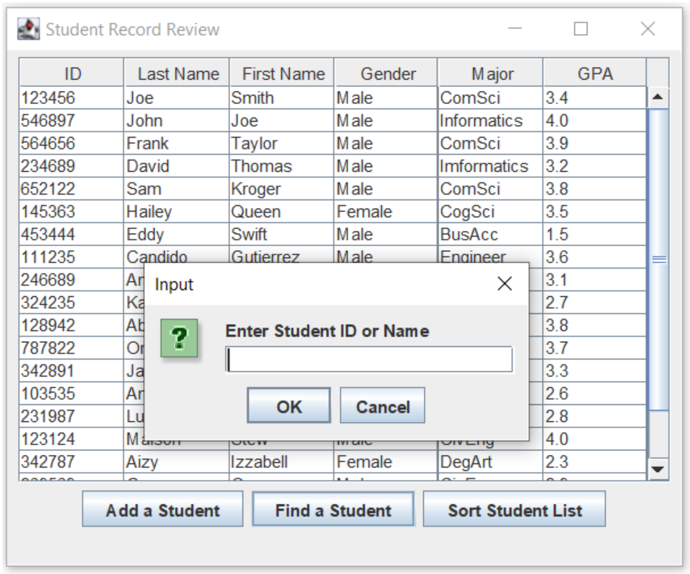
Figure 18

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

Finding a Student

1. Click on “Find a Student” button from the “Student Record Review” window (Figure 12).

2. An input window will appear (Figure 19).



Student Record Review

ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Hailey	Queen	Female	CogSci	3.5
453444	Eddy	Swift	Male	BusAcc	1.5
111235	Candido	Gutierrez	Male	Engineer	3.6
246689	Ar				3.1
324235	Ka				2.7
128942	Ab				3.8
787822	Or				3.7
342891	Ja				3.3
103535	Ar				2.6
231987	Lu				2.8
123124	Maison				4.0
342787	Aizy	Izzabell	Female	DegArt	2.3

Input

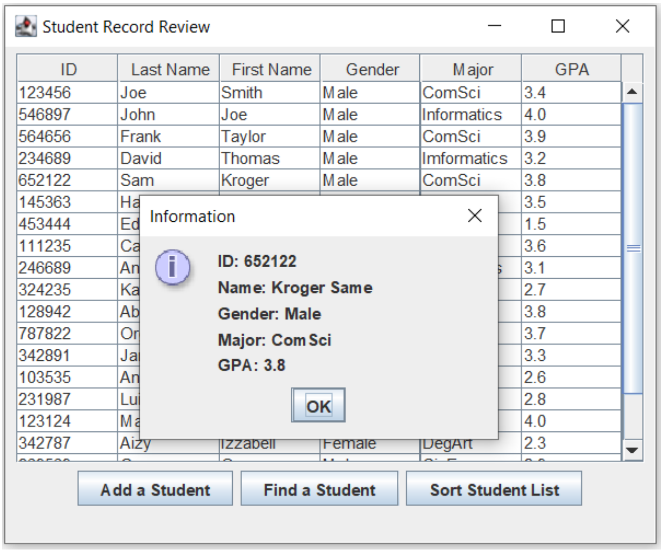
Enter Student ID or Name

OK Cancel

Add a Student Find a Student Sort Student List

Figure 19

3. Enter student ID or name.
4. Click the “OK” button to find the student or the “Cancel” button to go back to “Student Record Review” window.
5. A window with the information of the queried student will appear after entering student ID or student name as the below Figure 20. Otherwise, if the student cannot be found, an error message will be displayed on the screen as figure 21.



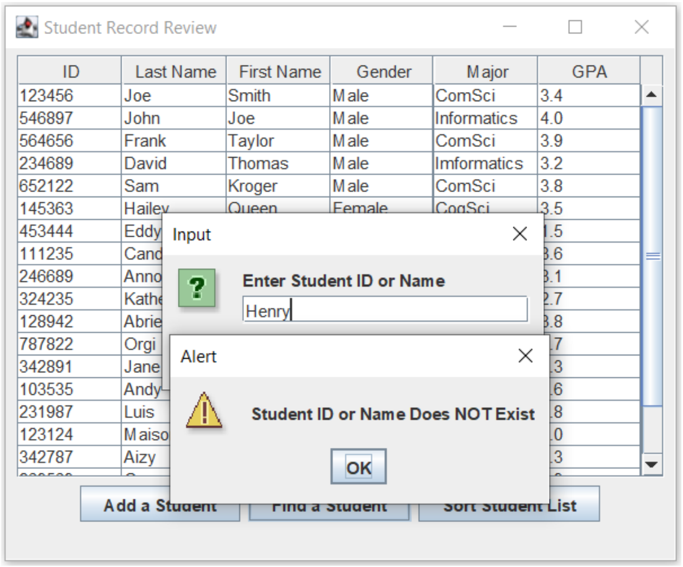
ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Ha				3.5
453444	Ed				1.5
111235	Ca				3.6
246689	An			s	3.1
324235	Ka				2.7
128942	Ab				3.8
787822	Or				3.7
342891	Ja				3.3
103535	An				2.6
231987	Lu				2.8
123124	Ma				4.0
342787	Aizy	Izzabell	Female	DegArt	2.3

Information

ID: 652122
Name: Kroger Same
Gender: Male
Major: ComSci
GPA: 3.8

OK

Figure 20



ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Hailey	Queen	Female	CogSci	3.5
453444	Eddy				1.5
111235	Cand				3.6
246689	Anno				3.1
324235	Kathe				2.7
128942	Abrie				3.8
787822	Orgi				7
342891	Jane				3
103535	Andy				6
231987	Luis				8
123124	Maiso				0
342787	Aizy				3

Input

Enter Student ID or Name

Henry

Alert

Student ID or Name Does NOT Exist

OK

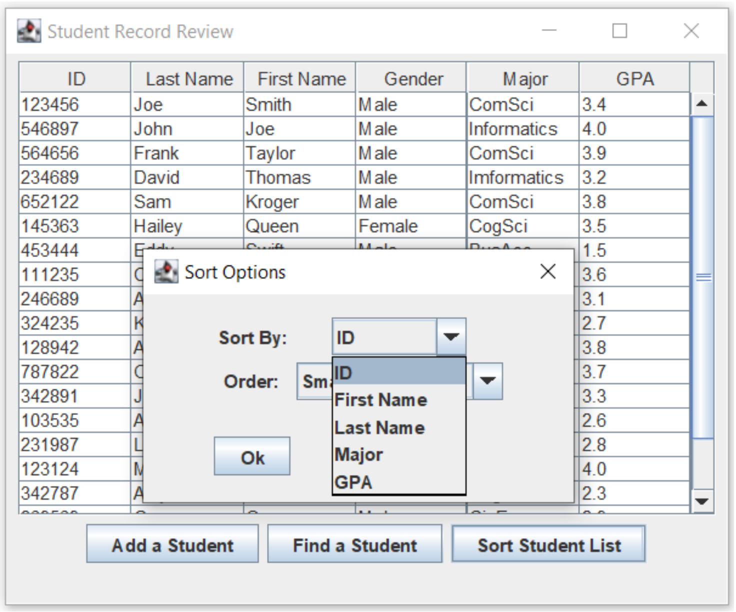
Figure 21

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

Sorting Student List

1. In the "Student Record Review" dialog (Figure 12), click "Sort Student List" button.

   "Sort Options" dialog will appear as the following figure 22, 23, and 24.



Student Record Review

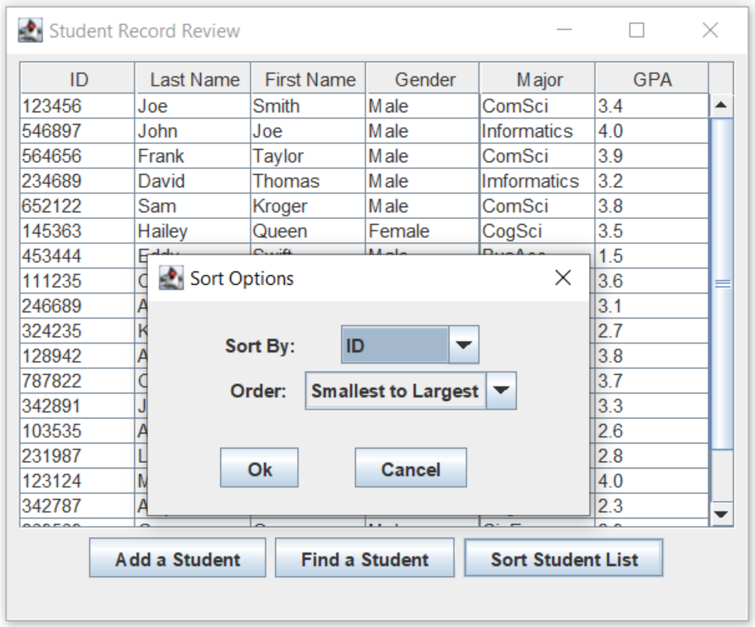
ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Hailey	Queen	Female	CogSci	3.5
453444	E				1.5
111235	C				3.6
246689	A				3.1
324235	K				2.7
128942	A				3.8
787822	C				3.7
342891	J				3.3
103535	A				2.6
231987	L				2.8
123124	M				4.0
342787	A				2.3

Sort Options

Sort By: ID

Order: Sm

Figure 22



Student Record Review

ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Hailey	Queen	Female	CogSci	3.5
453444	E				1.5
111235	C				3.6
246689	A				3.1
324235	K				2.7
128942	A				3.8
787822	C				3.7
342891	J				3.3
103535	A				2.6
231987	L				2.8
123124	M				4.0
342787	A				2.3

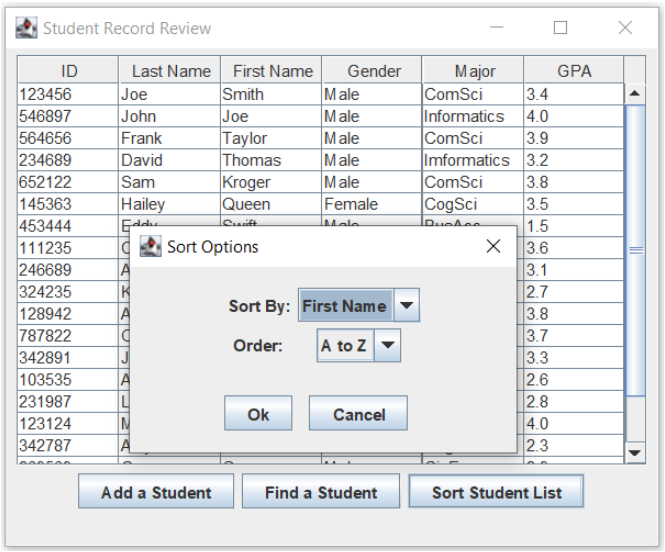
Sort Options

Sort By: ID

Order: Smallest to Largest

Figure 23

2. Choose the column and select the preferred order to sort.



Student Record Review

ID	Last Name	First Name	Gender	Major	GPA
123456	Joe	Smith	Male	ComSci	3.4
546897	John	Joe	Male	Informatics	4.0
564656	Frank	Taylor	Male	ComSci	3.9
234689	David	Thomas	Male	Imformatics	3.2
652122	Sam	Kroger	Male	ComSci	3.8
145363	Hailey	Queen	Female	CogSci	3.5
453444	Eddy	Swift	Male	BusAcc	1.5
111235	C				3.6
246689	A				3.1
324235	K				2.7
128942	A				3.8
787822	C				3.7
342891	J				3.3
103535	A				2.6
231987	L				2.8
123124	M				4.0
342787	A				2.3

Sort Options

Sort By: First Name

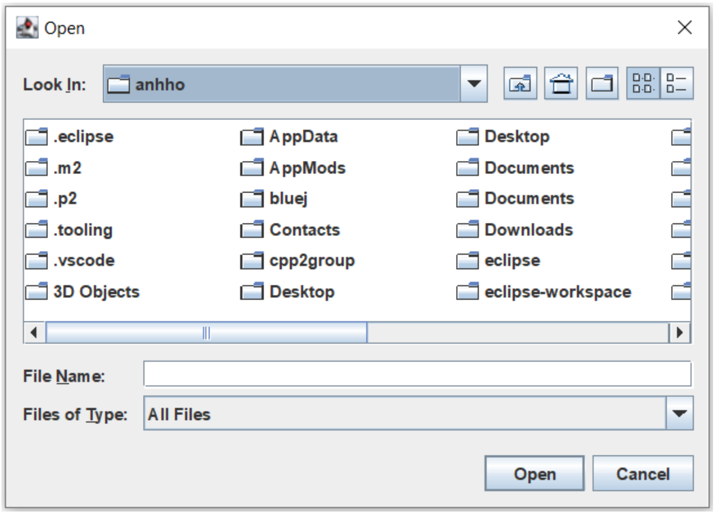
Order: A to Z

Figure 24

3. Click the “Ok” button to sort or the “Cancel” button to go back to the “Student Record Review” window (Figure 12).

Importing the File

1. Click the import file button from the main menu window of SRMS as in the figure 11.
2. The import file window will show up as figure 25.



Open

Look In: anhho

.eclipse	AppData	Desktop
.m2	AppMods	Documents
.p2	bluej	Documents
.tooling	Contacts	Downloads
.vscode	cpp2group	eclipse
3D Objects	Desktop	eclipse-workspace

File Name:

Files of Type: All Files

Figure 25

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 26.

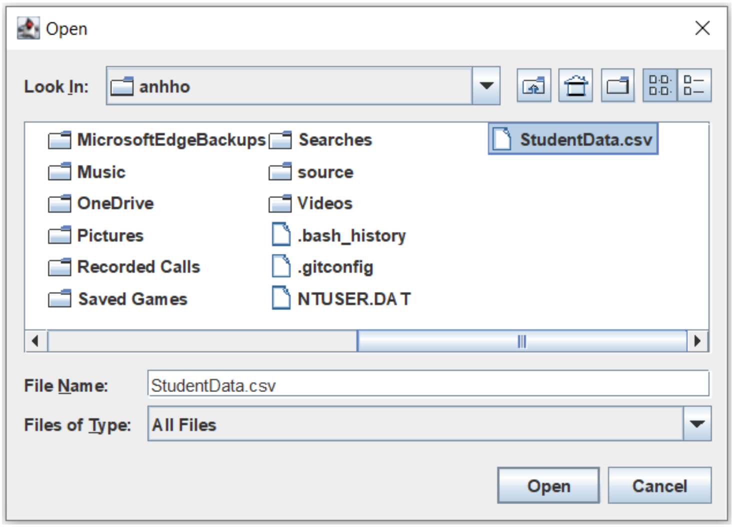


Figure 26

5. An error message will be popped up if user opens wrong file (Figure 27)

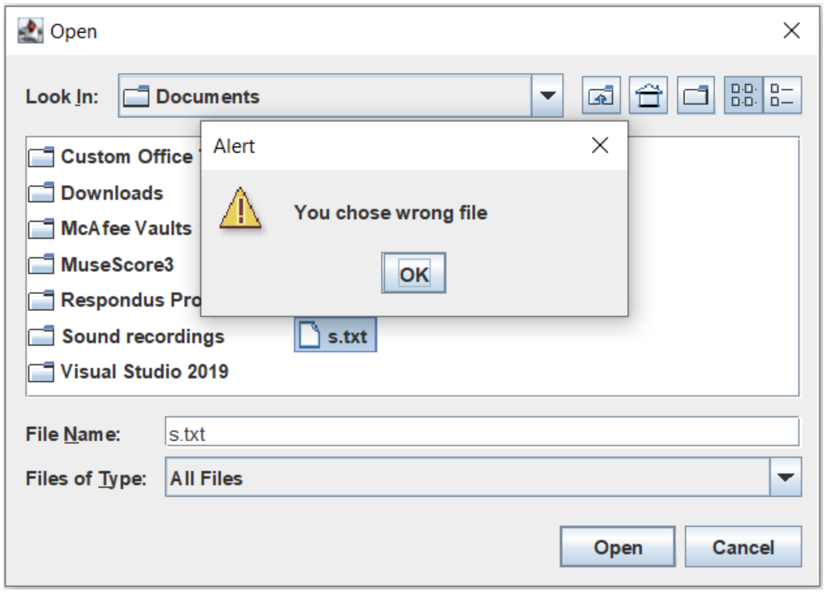


Figure 27

6. The info window will appear with the information of the selected file if user opens correct file (Figure 28).

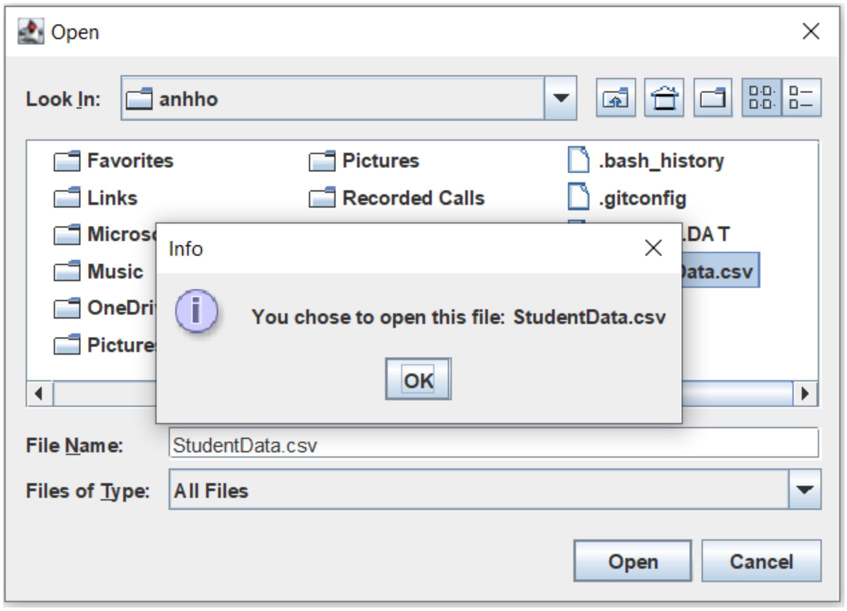


Figure 28

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

Exporting the File

1. From the main menu window of SRMS as in figure 11, click on “Export File” button.

2. The export file window will appear (Figure 29).

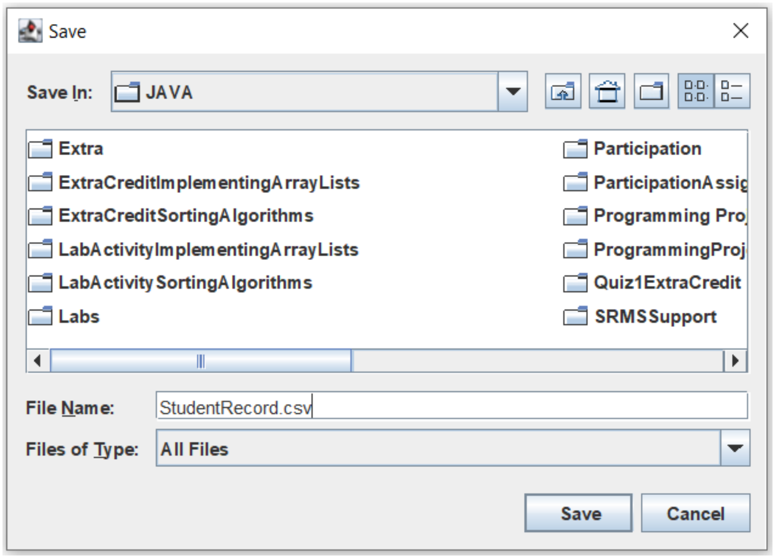
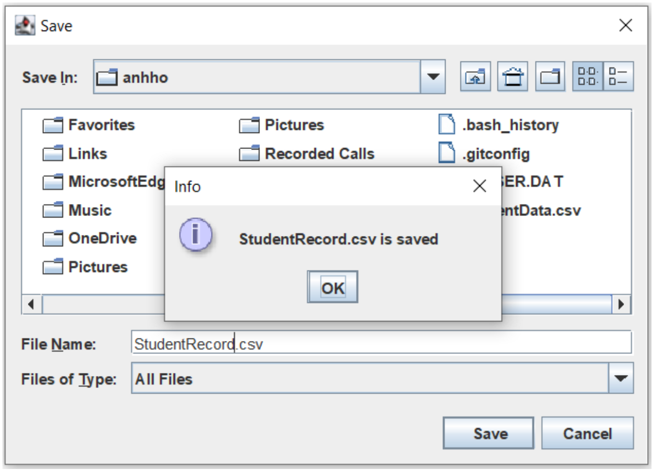

Save
Save In: JAVA
Extra
ExtraCreditImplementingArrayLists
ExtraCreditSortingAlgorithms
LabActivityImplementingArrayLists
LabActivitySortingAlgorithms
Labs
Participation
ParticipationAssig
Programming Proj
ProgrammingProj
Quiz1ExtraCredit
SRMSSupport
File Name: StudentRecord.csv
Files of Type: All Files
Save
Cancel

Figure 29

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 30).
6. Click the “OK” or “X” button to return to the main menu.



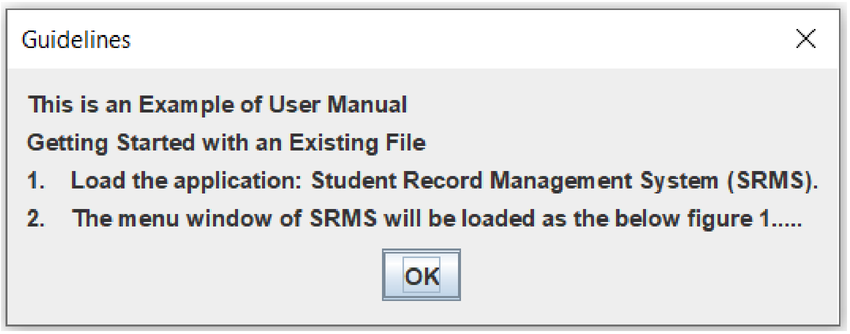
StudentRecord.csv is saved

StudentRecord.csv

Figure 30

Using Help Button

1. In the main menu window of SRMS (Figure 11), click “Help”, then a “Guidelines” dialog will appear for a quick reference guide (Figure 31).
2. Click the “OK” or “X” button to return to the main menu.



This is an Example of User Manual
Getting Started with an Existing File
1. Load the application: Student Record Management System (SRMS).
2. The menu window of SRMS will be loaded as the below figure 1.....

Figure 31

Using “Log Out” button

1. Click the “Log Out” or “X” button in the main menu of SMRS (Figure 11) to log out and exit the program completely.

References

N/A.