Orange Coast College

CS A220: Software Engineering

Student Record Management System

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

Presentation Date: March 20, 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 Sprint Backlog Updated Staging/ Grooming Updated Development Process Updated User Manual	21 22 23 – 24 25 – 31	All	03/20/2021
5				
6				
7				

Project Plan

- What kind of project is that?

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

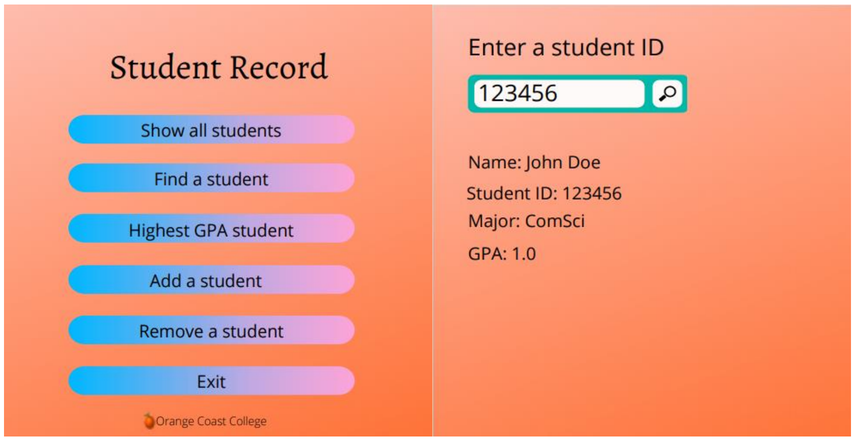
- What is the programming language you need to know?

  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 idea 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 tool that organizes and helps users easily access and manage the student records.

User Stories

Functional

1. **As** a user

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

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

2. **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.

3. **As** a user

   I need a help button

   So that I can learn more about the system.

4. **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.

5. **As** a user

   I want to have a return button

   So that I can return to the main menu.

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 to be able to search by using the student's ID or name

   So that I can choose the most convenient method for me.

5. **As** a user

   I want the system to be fault-tolerable

   So that it won't crash when there is a faulty input, but instead giving an error announcement.

Use Case

Use Case Textual

Use Case: Show all the students in the database.

ID: UC-01

Description:

User selects to see all the students in the database. The system processes the database and displays all the students' information on the screen.

Primary actor:

User

Pre-conditions:

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

User successfully clicks the "Show all students'' button.

Post conditions:

<u>Success end condition</u>

The students' database is successfully loaded by the system and showed up on the screen.

<u>Failure end condition</u>

Users are unable to see the list of all students. No students' information is displayed by the system.

Trigger

User clicks the "Show all students'' button.

Main Success Scenario

6. User launches the system.
7. System displays a menu of functions.

8. User clicks the “Show all students” button.
9. System processes the database.
10. System displays the students’ information on the screen.

Extensions

5a. In step 5, if the user decides to close the students’ information window:

1. User selects to close the data window.
2. Use case resumes on step 2.

Use Case: Show the highest GPA students.

ID: UC-02

Description:

Users select to see the highest GPA students in the database. The system processes the database and displays the students' information with the highest GPA on the screen.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks the "Highest GPA students'' button.

Post conditions:

<u>Success end condition</u>

The system successfully searches through the database and displays the students' information with the highest GPA on the screen.

<u>Failure end condition</u>

Users are unable to see the students' information with the highest GPA. No students' information is displayed by the system.

Trigger

User clicks the "Highest GPA students" button.

Main Success Scenario

1. User launches the system.
2. System displays a menu of functions.
3. User clicks the "Highest GPA students" button.

4. System searches for the students with the highest GPA in the database.
5. System displays the students' information with the highest GPA on the screen.

Extensions

5a. In step 5, if the user decides to close the students' information window:

1. User selects to close the window.
2. Use case resumes on step 2.

Use Case: Find a student in the database.

ID: UC-03

Description:

Users select to find a student in the database. The system processes the database and displays the queried students’ information on the screen

Primary actor:

User

Pre-conditions:

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

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

Post conditions:

<u>Success end condition</u>

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

<u>Failure end condition</u>

The system cannot find the student’s name in the database. User is unable to see the queried student’s information. A warning message that the student’s name cannot be found is displayed by the system.

Trigger

User clicks the “Find a student’’ button.

Main Success Scenario

1. User launches the system.
2. System displays a menu of functions.

3. User clicks the “Find a student’’ button.
4. User inputs the student’s ID.
5. User clicks the OK button.
6. System searches through the database.
7. System displays the queried student’ information on the screen.

Extensions

4a. In step 4, if the user decides to cancel finding a student:

1. User will select the cancel button or close the window.
2. The use case resumes on step 2.

Use Case: Use “Import file” button

ID: UC-04

Description:

Users can import the file containing the data of students into the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks the “Import file” button.

Post conditions:

<u>Success end condition</u>

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

<u>Failure end condition</u>

No file is imported into the system.

Trigger

User clicks on the “import file” button.

Main Success Scenario

1. User launches the system.
2. System displays a menu of functions.
3. User clicks on the “import file” button.
4. The import file window will be opened
5. User clicks on “Look in” to find the location of the file needed.
6. User selects the appropriate data file, then clicks the open button.

7. System displays the info window with the file name that was selected.
8. User clicks the “OK” button.
9. Data of the students will be added into the system.

Extensions

6b. In step 6, if the user decides to cancel importing a student:

1. User will select the cancel button or close the window.
2. The use case resumes on step 2.

7a. In step 7, if the user chooses an inappropriate file:

1. The system displayed a warning message.
2. The use case resumes on step 2.

Use Case: Use “Export file” button.

ID: UC-05

Description:

User selects the “Export file” button. The system backs up the database into a file and saves them into a designated folder.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks the “Export file” button.

Post conditions:

<u>Success end condition</u>

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

<u>Failure end condition</u>

The system is unable to back up the 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. System displays a menu of functions.
3. User clicks the “Export file” button.

4. The Export File Window will show up.
5. User chooses a location and types in the name of the backup file.
6. User clicks the Save button.
7. System backs up the database into a file and saves it to the designated folder.

Extensions

5a. In step 5, if the user decides to cancel saving the database:

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

Use Case: Use help button

ID: UC-06

Description:

A help button to guide the user through the functions in the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks 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 clicks the help button.

Main Success Scenario

1. User launches the system.
2. System displays a menu of functions.
3. User clicks the help button.
4. System displays the help guide about each button's functionality.

Extensions

4a. In step 4, if the user decides to close the help window:

1. User selects to close the window.
2. Use case resumes on step 2.

Use Case: Use Exit button

ID: UC-07

Description:

An exit button to let the user exit the system.

Primary actor:

User

Pre-conditions:

The system is successfully loaded by the user.

User successfully clicks the exit button.

Post conditions:

<u>Success end condition</u>

User exits the system successfully.

<u>Failure end condition</u>

User cannot exit the system.

Trigger

User clicks the exit button.

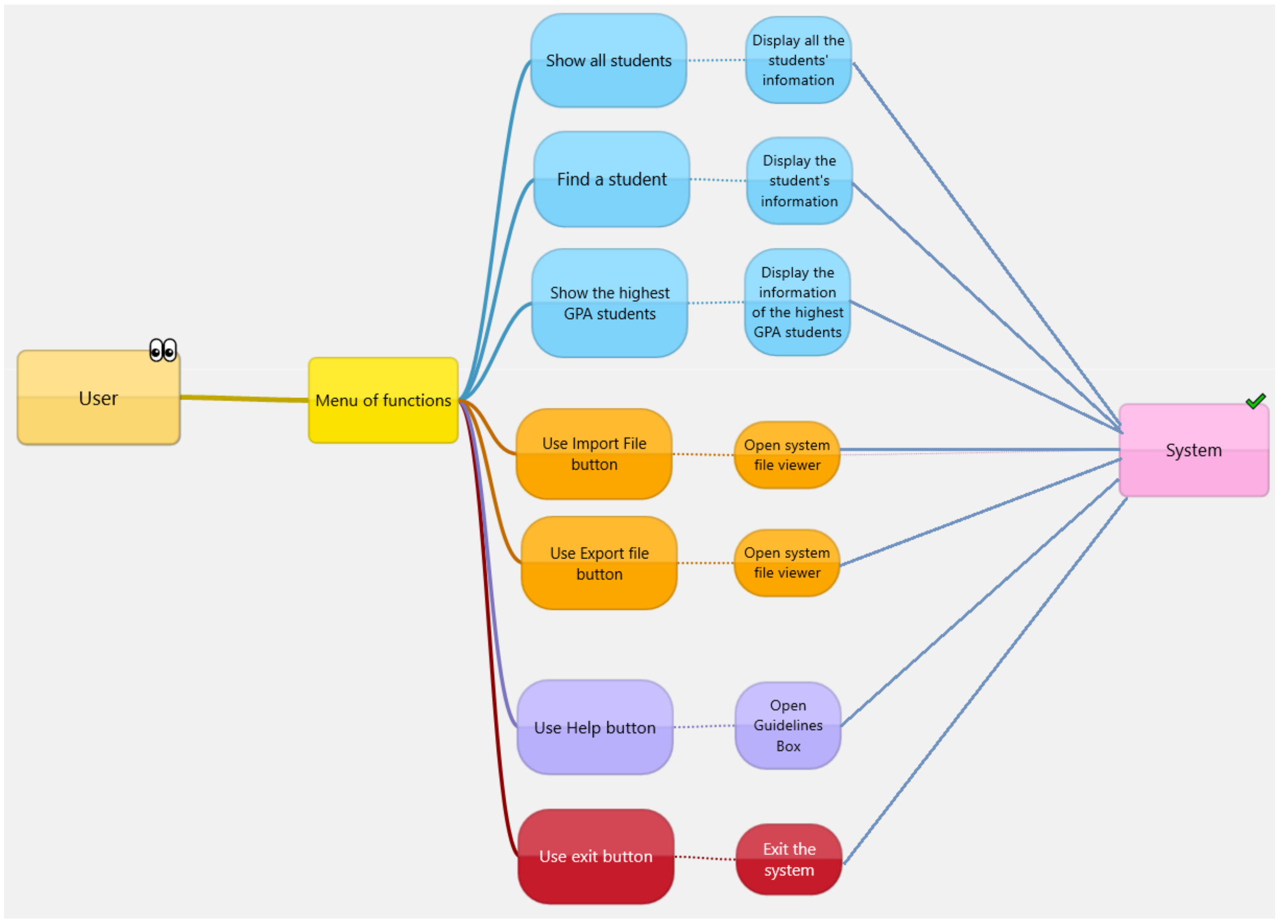
Main Success Scenario

1. User launches the system.
2. System displays a menu of functions.
3. User clicks the exit button.
4. The system is closed.

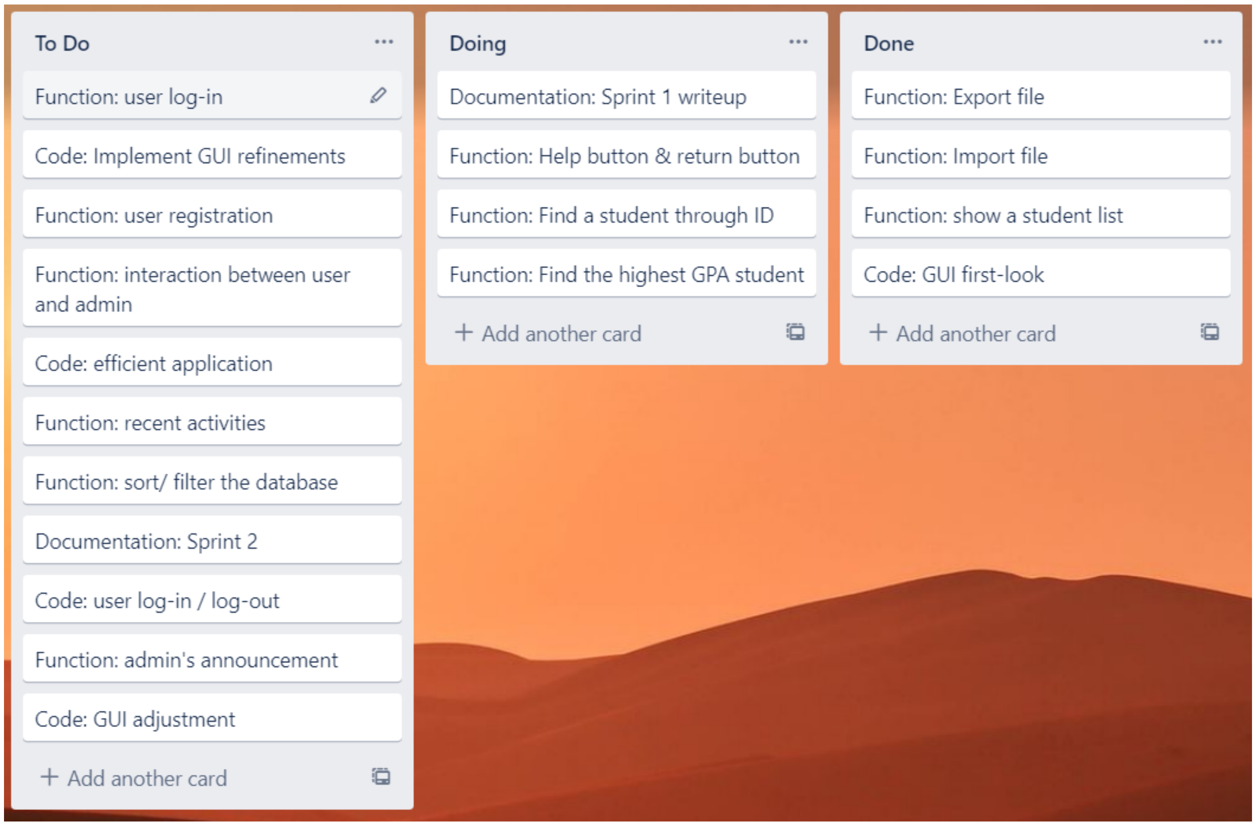
Extensions

N/A

Use Case Diagram



Sprint Backlog



To Do

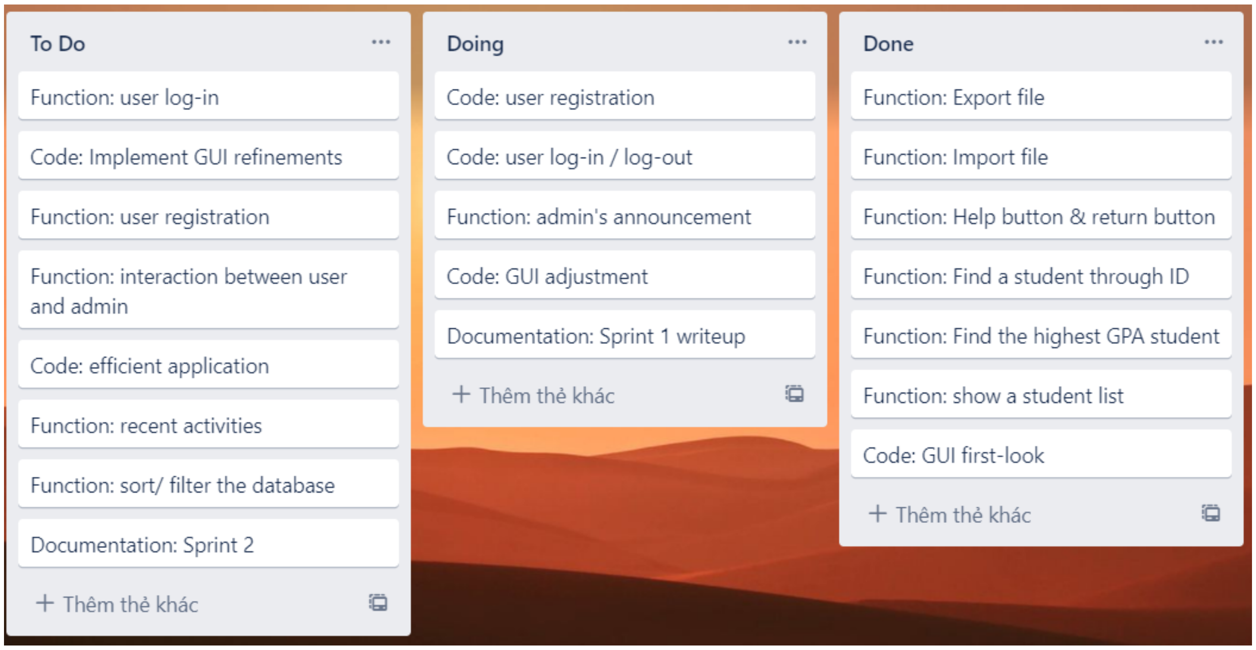
- Function: user log-in
- Code: Implement GUI refinements
- Function: user registration
- Function: interaction between user and admin
- Code: efficient application
- Function: recent activities
- Function: sort/ filter the database
- Documentation: Sprint 2
- Code: user log-in / log-out
- Function: admin's announcement
- Code: GUI adjustment
- + Add another card

Doing

- Documentation: Sprint 1 writeup
- Function: Help button & return button
- Function: Find a student through ID
- Function: Find the highest GPA student
- + Add another card

Done

- Function: Export file
- Function: Import file
- Function: show a student list
- Code: GUI first-look
- + Add another card



To Do

- Function: user log-in
- Code: Implement GUI refinements
- Function: user registration
- Function: interaction between user and admin
- Code: efficient application
- Function: recent activities
- Function: sort/ filter the database
- Documentation: Sprint 2
- + Thêm thẻ khác

Doing

- Code: user registration
- Code: user log-in / log-out
- Function: admin's announcement
- Code: GUI adjustment
- Documentation: Sprint 1 writeup
- + Thêm thẻ khác

Done

- Function: Export file
- Function: Import file
- Function: Help button & return button
- Function: Find a student through ID
- Function: Find the highest GPA student
- Function: show a student list
- Code: GUI first-look
- + Thêm thẻ khác

Staging/ Grooming

As we are planning and grooming based on our user stories, we break down large user stories into smaller tasks that are based on priorities. We also decide to work on functions that are required for the basic implementation of our program. This will let us start with the basic functions, allowing us to remove uncertain user stories that no longer appear to be relevant to our software. Then we can focus on non-functionality and any other elements. To begin, we will create a text file that includes the vital data that is required for the management and editing features of the student record system. We then begin to import the file to the system for later managing and editing the data.

GUI is the next thing that we will be working on because the interface is the most important feature that allows users to manipulate elements and easily access available functions. GUI also gives us more convenience with how they display information and images, which makes tasks more accessible to the users. We will be working to ensure that the GUI is user-friendly and attractive so that the users will easily manipulate the functions and spend less time discovering the application. We will also be working with non-functional requirements to make the application more efficient and the GUI to work properly. When we are done with the GUI and the functions, we will start to import the data and move on with other features.

Development Process

Our project is an application to query and manage students' records in the database. The program allows the users to easily access the student's database and also to edit and manage student's information.

For Agile Development, our project is divided into sprints, and each sprint comprises a full software development cycle including planning, requirements analysis, design, coding, testing, and release for each sprint. Each sprint is completed in an estimated time of 1-3 weeks.

We also allocate and determine team member's strengths and weaknesses for each task for the best outcomes. Team members with a good understanding of Java and logic programming are required for this project, and people who are specialized in writing software documentation are also necessary.

According to user stories, importing and exporting files with ease is our first priority. The second milestone is to make sure that users can easily interact with the GUI and be able to access all the features.

Requirements:

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

Project deliverables:

- User manual.
- Coded with Java language.

- Fully functional GUI.

Design and Prototyping

- Architecture: Java language.
- User Interface: The GUI will be interacted by mouse and keyboard and will be user-friendly as much as possible. When a user uses a mouse to click on the function buttons, the application will display the necessary information so that the user can check and edit the data. Example: When a user clicks the "Show all students" button, the application will automatically display all students and their information in the system.
- Platforms: Windows
- Programming: Java is the main programming language that will be utilized with JFrame, JOptionPane, JFileChooser, and JTable. For JFrame, it works like the main window, where functions such as labels, buttons and textfields are added to produce a graphical user interface (GUI). Since JFrame is a Swing component, it provides us with lots of functionality and flexibility so that we can easily implement them. JOptionPane includes facilities that allow us to create dialog boxes which appear on the screen to request the users to input or display messages for announcement. JFileChooser is a simple and efficient method for prompting the user to select a file or directory. Therefore, users can find the location to import/export student's data. Finally, we use JTable to display a table of student's information (Student ID, First Name, Last Name, Major, GPA) and allow users to modify the data.

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.

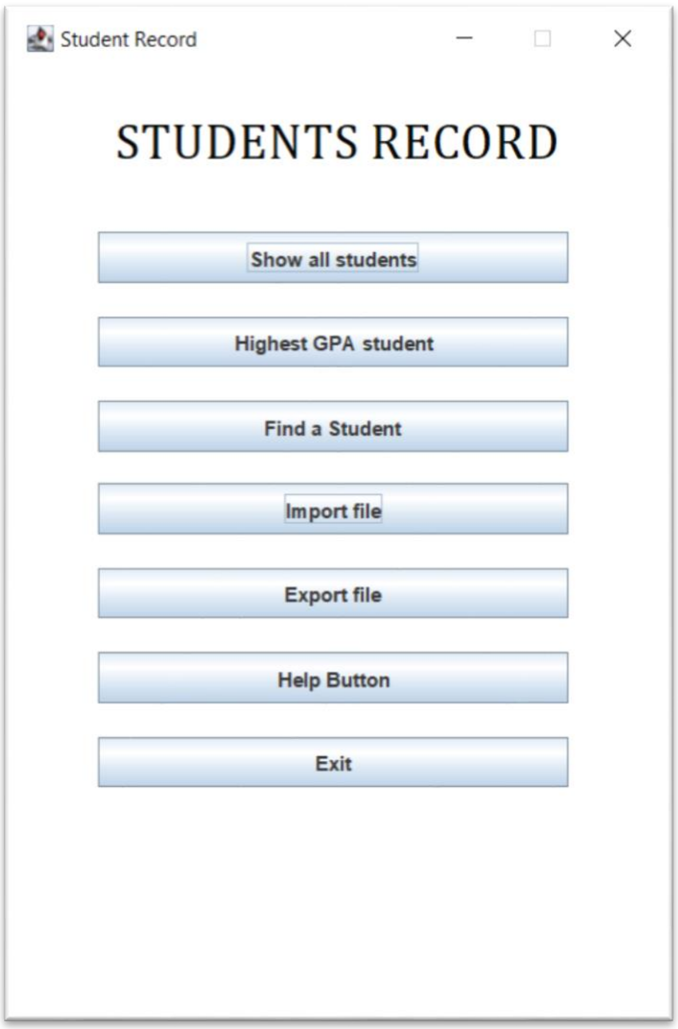
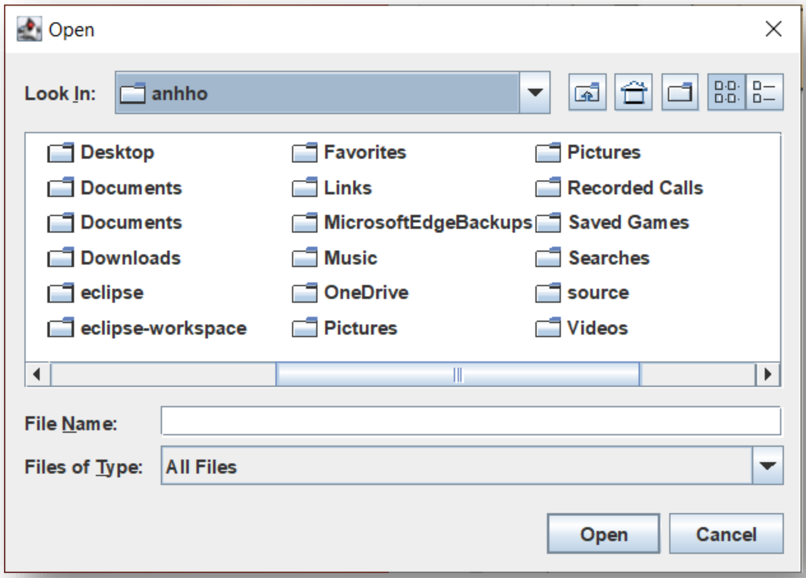


Figure 1

3. Click the import file button.
4. The import file window will be opened as the figure 2.



Look In: anhho

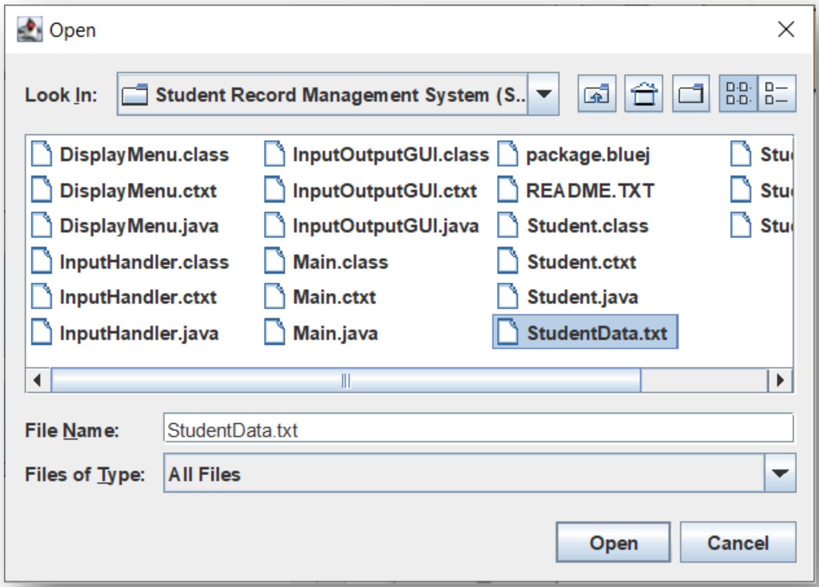
File Name:

Files of Type: All Files

Figure 2

5. Click on “Look In” to find the location of the file needed.

6. Select the appropriate data file, then click the open button as the below figure 3.



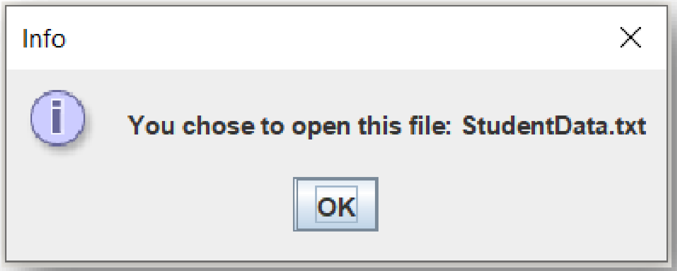
Look In: Student Record Management System (S..

File Name: StudentData.txt

Files of Type: All Files

Figure 3

7. The info window will appear with the file information that was selected (Figure 4).



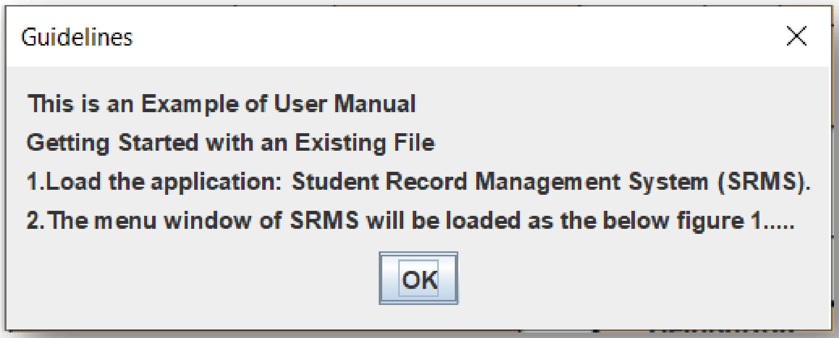
You chose to open this file: StudentData.txt

Figure 4

8. Click “OK” or “X” buttons to return to main menu.

Getting Started with a Help Button

1. Load the application: Student Record Management System (SRMS).
2. The menu window of SRMS will be loaded as the above figure 1.
3. Click “Help Button” and a “Guidelines” box will appear for a quick reference guide (Figure 5).
4. Click “OK” or “X” buttons to return to the main menu.



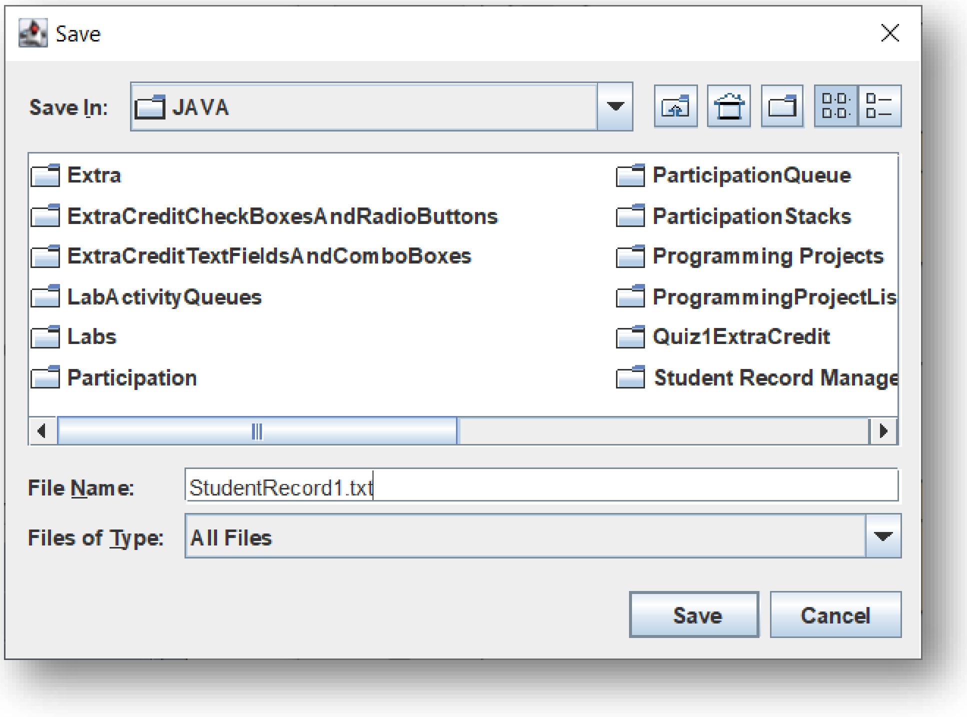
This is an Example of User Manual
Getting Started with an Existing File
1.Load the application: Student Record Management System (SRMS).
2.The menu window of SRMS will be loaded as the below figure 1.....

Figure 5

Exporting the File

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

2. The export file window will be opened (Figure 6).



File Name: StudentRecord1.txt

Files of Type: All Files

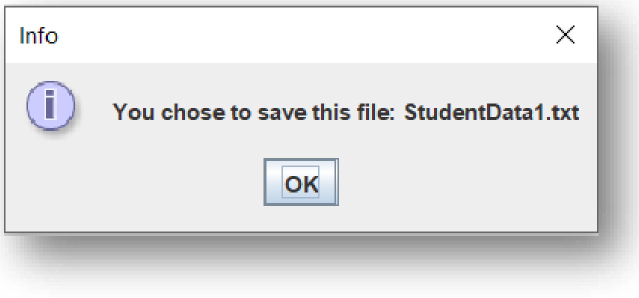
Figure 6

3. Click on “Save In” to choose where to save files.

4. Type file name and click on the “Save” button to save the file.

5. An info window will open and show the saved file name (Figure 7).

6. Click “OK” or “X” buttons to return to the main menu.

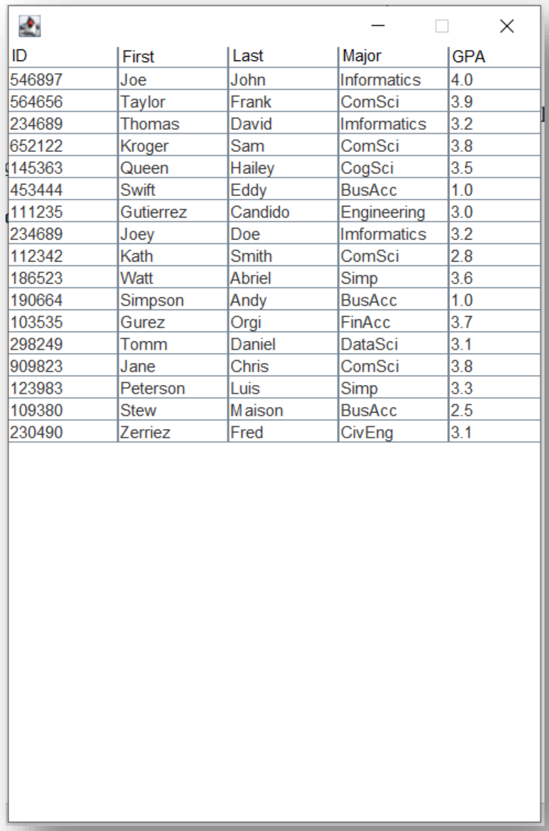


You chose to save this file: StudentData1.txt

Figure 7

Showing All Students

1. From the menu window of SRMS as in the above figure 1, click on “Show all students” button.
2. A data table of all students from the import file will appear including the columns showing ID, first name, last name, major, and GPA as the figure 8.
3. Click the “X” button to return to the main menu.



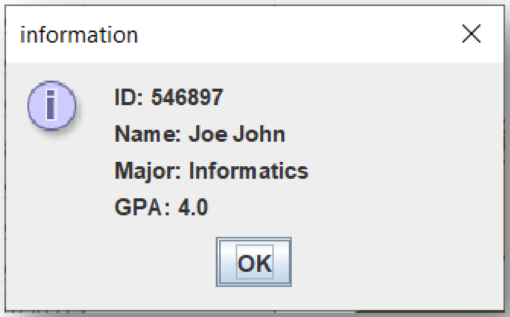
ID	First	Last	Major	GPA
546897	Joe	John	Informatics	4.0
564656	Taylor	Frank	ComSci	3.9
234689	Thomas	David	Imformatics	3.2
652122	Kroger	Sam	ComSci	3.8
145363	Queen	Hailey	CogSci	3.5
453444	Swift	Eddy	BusAcc	1.0
111235	Gutierrez	Candido	Engineering	3.0
234689	Joey	Doe	Imformatics	3.2
112342	Kath	Smith	ComSci	2.8
186523	Watt	Abriel	Simp	3.6
190664	Simpson	Andy	BusAcc	1.0
103535	Gurez	Orgi	FinAcc	3.7
298249	Tomm	Daniel	DataSci	3.1
909823	Jane	Chris	ComSci	3.8
123983	Peterson	Luis	Simp	3.3
109380	Stew	Maison	BusAcc	2.5
230490	Zerriez	Fred	CivEng	3.1

Figure 8

Highest GPA Student

1. From the menu window of SRMS as the above figure 1, click on “Highest GPA student” button.

2. An information dialog will appear and show the highest GPA student (Figure 9).


information

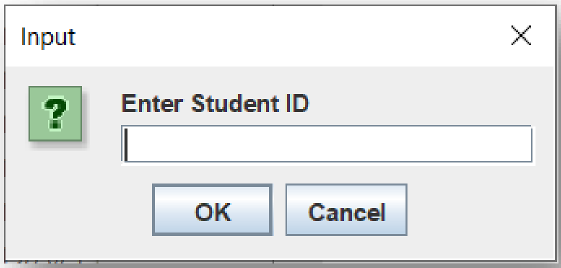
ID: 546897
Name: Joe John
Major: Informatics
GPA: 4.0

Figure 9

3. Click “OK” or “X” buttons to return to the main menu.

Find a Student

1. From the menu window of SRMS as in the above figure 1, click on “Find a Student” button.
2. An input window will appear (Figure 10).

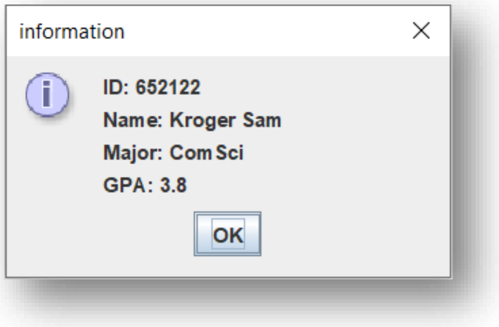

Input

Enter Student ID

Figure 10

3. Enter student ID.
4. Click on the “OK” button to find the student or “Cancel” button to go back to the main menu.
5. A window with the information of the queried student will appear after entering student ID as the below Figure 11.

6. Click “OK” or “X” buttons to return to the main menu.



ID: 652122
Name: Kroger Sam
Major: Com Sci
GPA: 3.8

Figure 11

Using “Exit” button

1. Click on the “Exit” button from the main menu as the above figure 1 to exit the program completely.

References

N/A.