Deliverable 2
Design Document

404 NOT FOUND

Matthew Chan
(matthew417@gmail.com)

Zhe Li
(lz897825130@gmail.com)

Gordon McCreary
(gord@uw.edu)

Ken Gil Romero
(kgmr@uw.edu)

Tammy Vo
(votammy97@gmail.com)

https://404github.github.io/404notfound_TCSS360/

Introduction

The purpose of our application is to allow DIYers to manage their projects. We will accomplish this by allowing the DIYer to create and delete projects which will be stored in a list that can be sorted by desired categories. Each project will store information about total cost and time cost as well as energy efficiency and required materials. This will enable the DIYer to compare and prioritize their projects.

The goal of our teams design is to keep the application simple and intuitive while providing all the desired functionality. We will do this by basing our user interface around a 'home page' that can't be navigated away from, while minimizing pop up windows. The purpose of this is to keep the navigation of our app easy and keeping the DIYer from getting lost in the application. We ultimately want to make performing any task on our application instinctive. We will accomplish this by using common user interface elements and visual composition that guides use.

Table of Contents

1. **Rationale Summary:**

Front End:

We decided to combine the GUI_NewProject class and GUI_EditProject class as one GUI_NewOrEditProject class. There are several reasons for that.

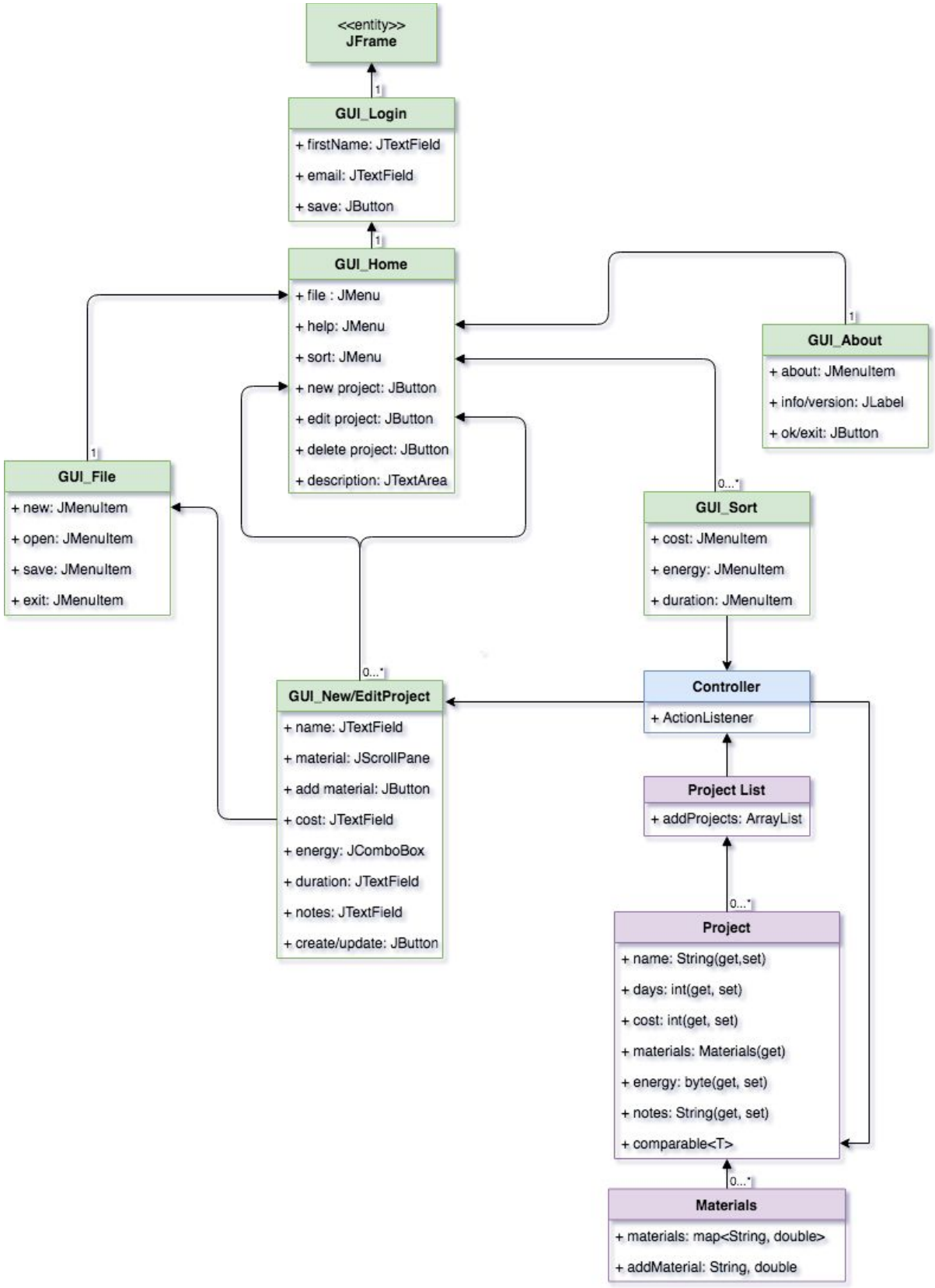
1. **Contents**
    a. They have the exact same number of input fields. Consider the following, the user has to input the name, material, cost, energy, duration, and notes as the data of a project. When it comes to editing, basically the user is going to change one or more of the previous mentioned data, so we can just fill up the same form with the existing data and let the user make the changes.
2. **Appearance**
    a. Since they share the input fields, their appearance should look similar or even the same as well. When it comes to the design of GUI, the alignment of those input components should be placed in the exact same spot with same dimension.
    b. The only differences are the title of the form and the text representation of the JButton. While creating a new project, the title of the form will be "Create a New Project" and the text representation of the JButton will be "Create". Similarly, "Edit an Existing Project" and "Update" will be displayed while on edit mode.
3. **Maintenance**
    a. Given that they are high in similarity (contents and appearance), it should be a good idea to combine them as one class. By having overloaded constructors, we can easily determine which mode is being used, and we can assign the text representation to the form title and the JButton. Having one class instead of two can decrease the project complexity, and it will be easier for code maintenance.

Back End:

There will be a dedicated class for the object type "Project".

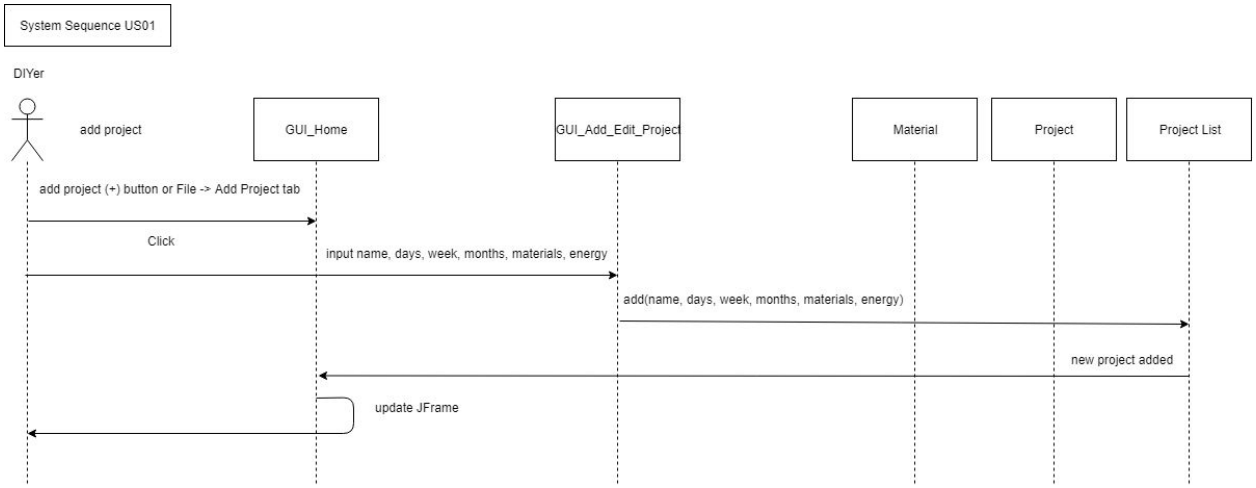
1. **Project**
    a. The system has to store all detailed data of a project, having a dedicated class for that can make things handy. Also, encapsulation is one of the four fundamental OOP concepts in Java. By having getters and setters, we can have access to those data fields easily while avoiding any illegal direct access to the private fields.
2. **Map Data Structure**
    a. We decided to use Map data structure to store all materials needed for a project. Since Map provides key and value pairs, it is handy for us to store the name of a material as a String and the cost as a double at the same time. Consider that there will not be any duplicated material names, we can store the names as the key of a map entry, and the cost as the corresponding map entry value.

2. Class Diagram:



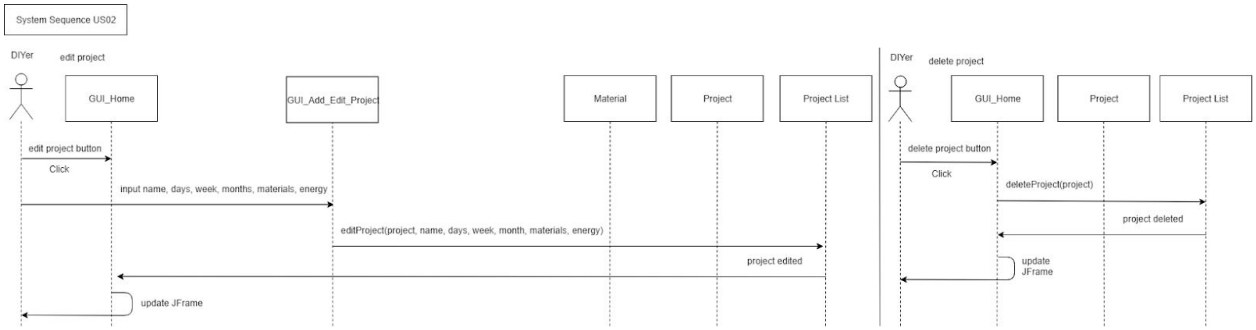
3. User Story Sequence Diagrams:

US01: As a DIYer, I want an app that collects project costs, duration, and other data that I want to put in.



The DIYer will initiate the create new project process by either selecting the add project button on the home page or through the File menu and selecting new project on the GUI_Home. We chose to have this option in two places because we want the user to have their own style of using our application. The GUI_Add_Edit_Project will show up where the user will input a project name, the day, week, and months of the project's duration, the materials and their cost needed. The user will also choose an energy efficiency rating. The GUI_Add_Edit_Project will then send the information given to the GUI_Home. It will then call the addProject method and pass all the project information through it. The project will be created and added to the Project List. After that it will notify the GUI_Home to update and display the updated Project List. The diagram shows us going through the Material and Project class because we will be storing the materials into the Material class and the project in the Project class, and then storing an instance of Material into a Project and that Project will be stored into the Project List.

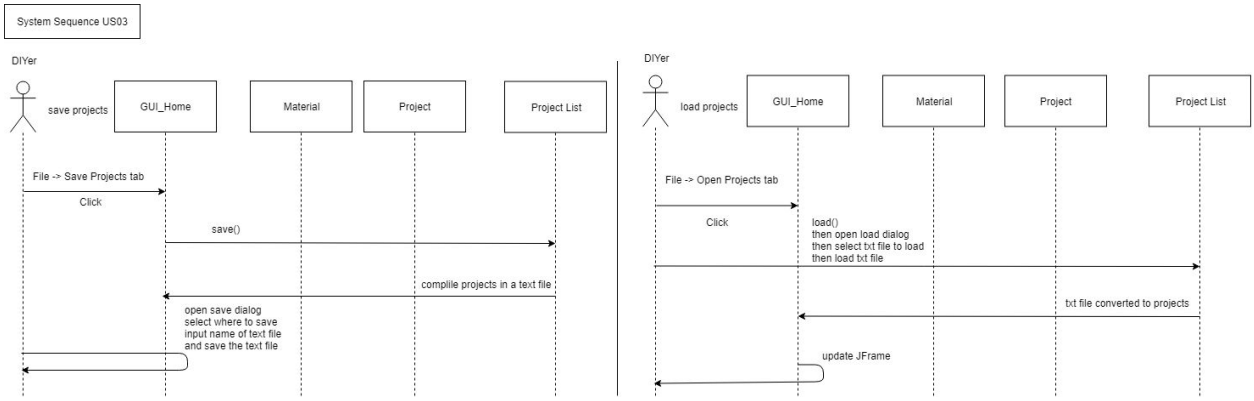
US02: As a DIYer, I want an app that organize my projects to compare many projects at once so I can choose one.



For editing a project, the DIYer will be clicking a project's edit button in the GUI_Controller. The GUI_Add_Edit_Project will show up where the user will change an input if he or she wants to change the project name, the day, week, and months of the project's duration, the materials or the cost needed. The user can also choose to change the energy efficiency rating. The GUI_Add_Edit_Project will then send the information given to the GUI_Controller. It will then call the editProject method and pass a project and all the edited project information through it. A project will be edited and updates the list of Projects. After that it will notify the GUI_Controller to update and display the updated list of Projects. The diagram shows us going through the Project class because we need to the match a Project from the Project List to a Project to be edited, then edit that Project's information and Materials.

For deleting a project, the DIYer will be clicking a project's delete button in the GUI_Controller. It will then call the deleteProject method and pass a project through it. A project will be deleted from the Project List. After that it will notify the GUI_Controller to update and display the updated Project List. The diagram shows us going through the Project class because we need to the match a Project from the Project List to a Project to be deleted.

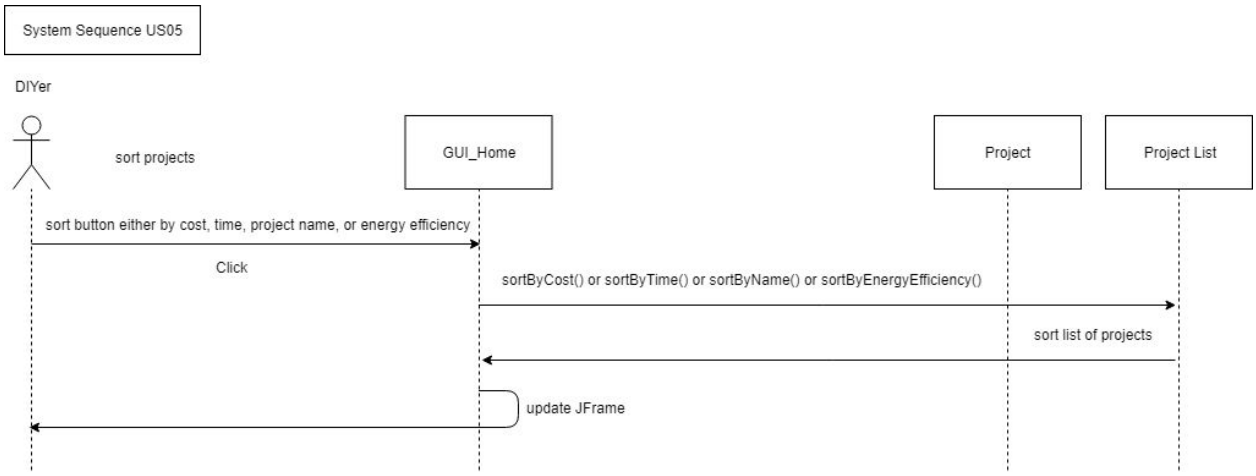
US03: As a DIYer, I want an app that load and saves the projects so I can export and then import to send the information on more than one device.



The DIYer will save the projects by the File menu and selecting save projects on the GUI_Controller. The GUI_Controller will then call the save method. The list of projects will be compiled into a text file. After that the save dialog appears where the DIYer will select where to save the text file. Then, the DIYer will input what the name of the text file will be, and click the save button. The diagram shows us going through the Material and Project class because we need to read every Project in the Project List and every Material in every Projects.

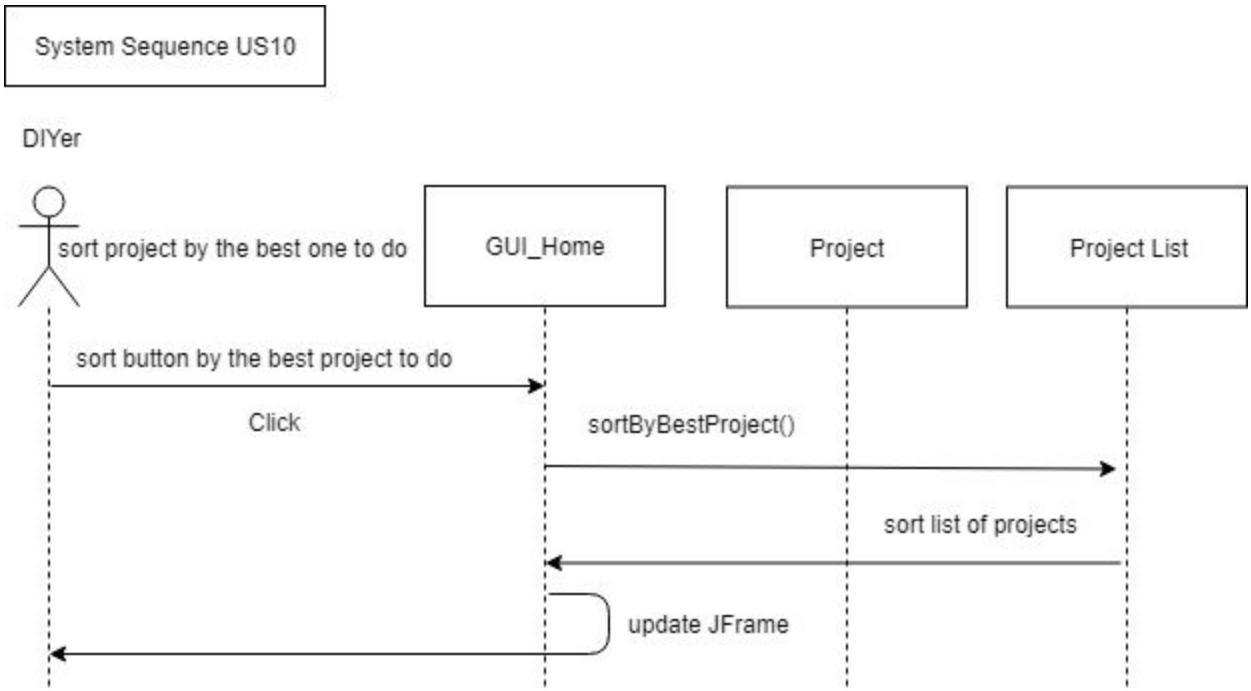
The DIYer will load projects by the File menu and selecting load projects on the GUI_Controller. The GUI_Controller will then call the load method. The load dialog appears where the DIYer can choose a file to load. The file will be converted to a list of projects but could result to an error where none will be loaded. After that the GUI_Controller and the display's list of projects will be updated. The diagram shows us going through the Material and Project class because we will be storing the materials into the Material class and the project in the Project class, and then storing an instance of Material into a Project and that Project will be stored into the Project List.

US05: As a DIYer, I want to be able to sort by the project's costs, duration, and other data that have ranking order.



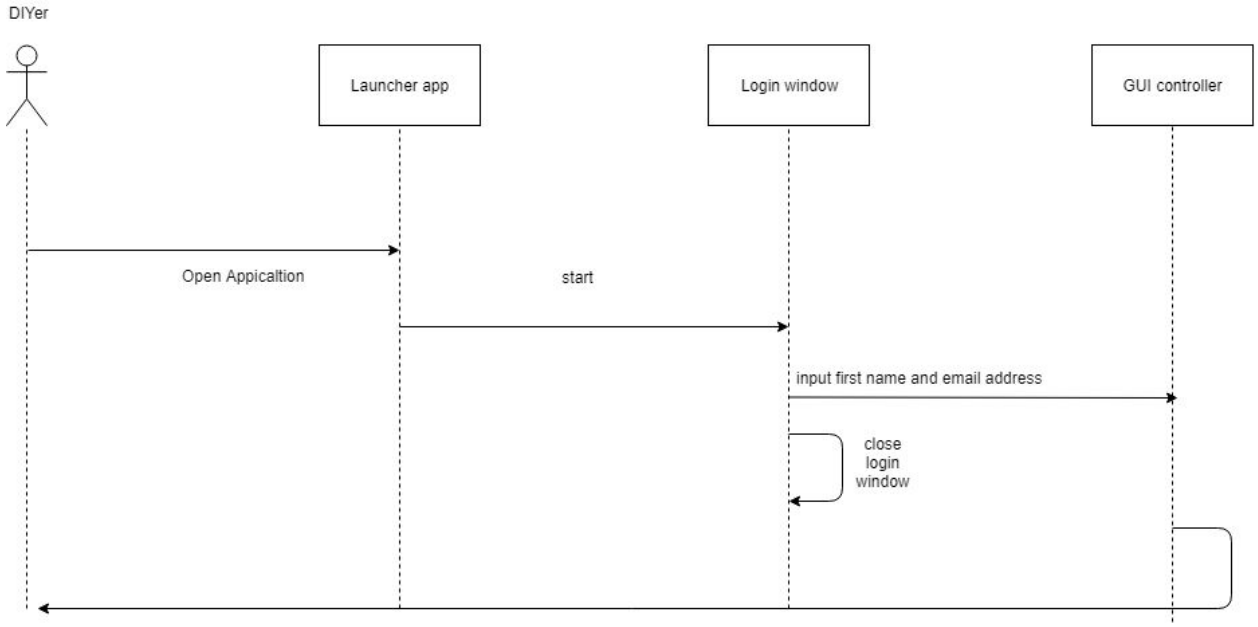
The DIYer will click any of the sort button on the GUI_Controller. The GUI_Controller will then call a sort method. The list of projects will be sorted by type of sorting that the DIYer want. After that it will notify the GUI_Controller to update and display the updated list of Projects. The diagram shows us going through the Project class because to sort the Project List we need every project information.

US10: As a DIYer,, I want an app to make basic calculations and helps weigh costs versus benefits to size up projects.



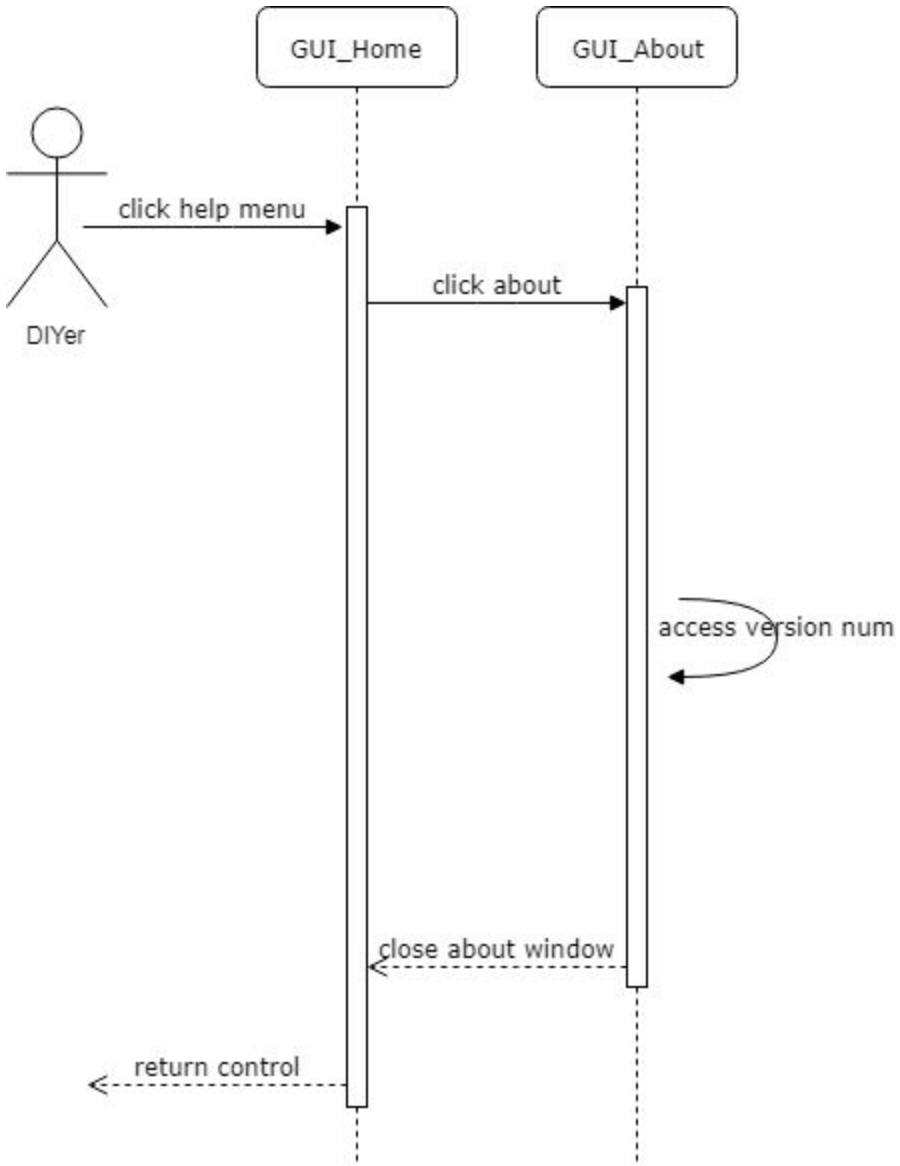
The DIYer will click sort button that sort by the best project on the GUI_Home. The GUI_Home will then call the sortByBestProject method. The Project List will be sorted by the best project. After that it will notify the GUI_Home to update and display the updated list of Projects. The diagram shows us going through the Project class because to sort the Project List we need every project information.

US15: As a user I want to enter settings such as my first name and email address



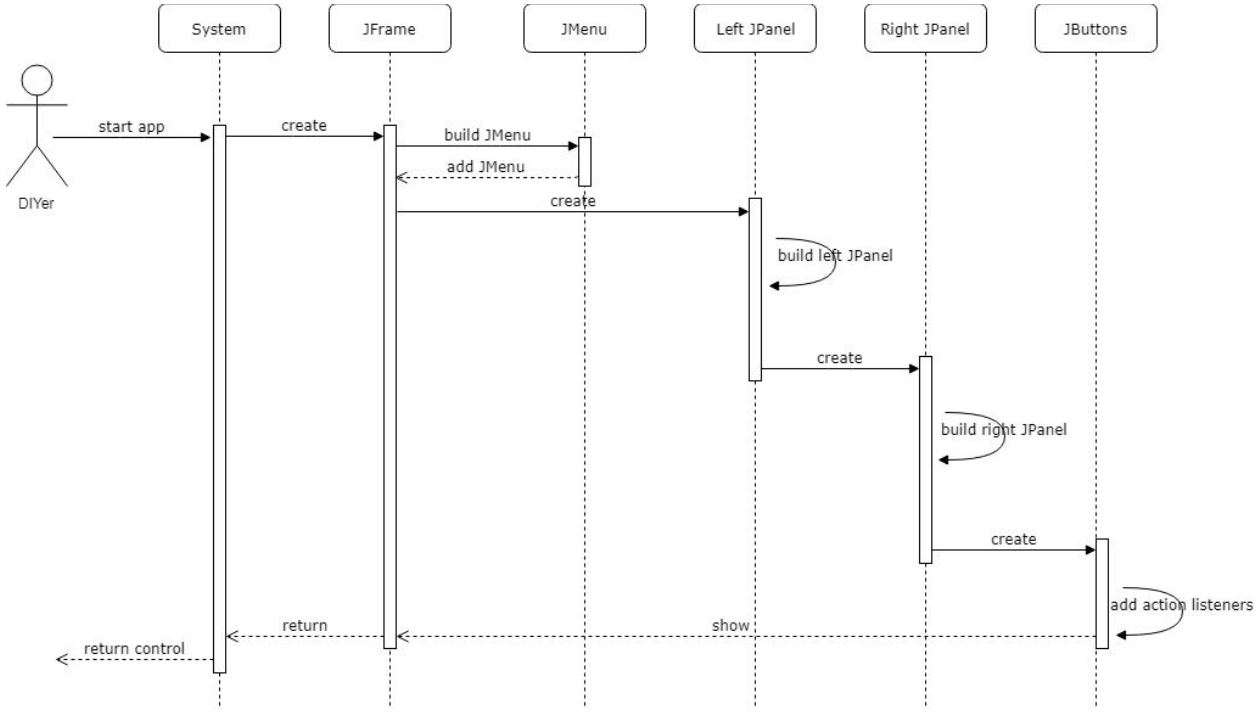
The DIYer will open the application. After launch the application, it will start the login window where the DIyer can input their first name and email address, or they can just close it without typing anything. This information will be sent to the GUI_Controller. The login window will be closed.

US16: As a user I want to see the version of the software and other information such as the names of the developers



From the GUI_Home page, the DIYer will click the help menu. From the help menu, the DIYer will click the about button. This will pop up an about box, which will access the version number in the back end. After the DIYer closes the about window, control will be return to the user of GUI_Home.

4. System Startup Sequence Diagram:



The DIYer will start the application. The System will create a JFrame for the application. The JFrame will build a JMenu, which will be added to the JFrame. The Left JPanel will be created and built, having all of its components created. Then the Right JPanel will be created and built, having all of its components created. Finally, all of the JButtons will be created and have action listeners added to them. The JFrame will be shown, which will return control to the System and then the DIYer.