Deliverable 2 - Requirements Document

D.N.A

Our website:

https://github.com/dkt8-1574124/tcss360dana

Contact Information:

Duy: dkt8@uw.edu Amelia: amelieb@uw.edu Nick:ncaron001@gmail.com

B. Introduction and Table of Contents

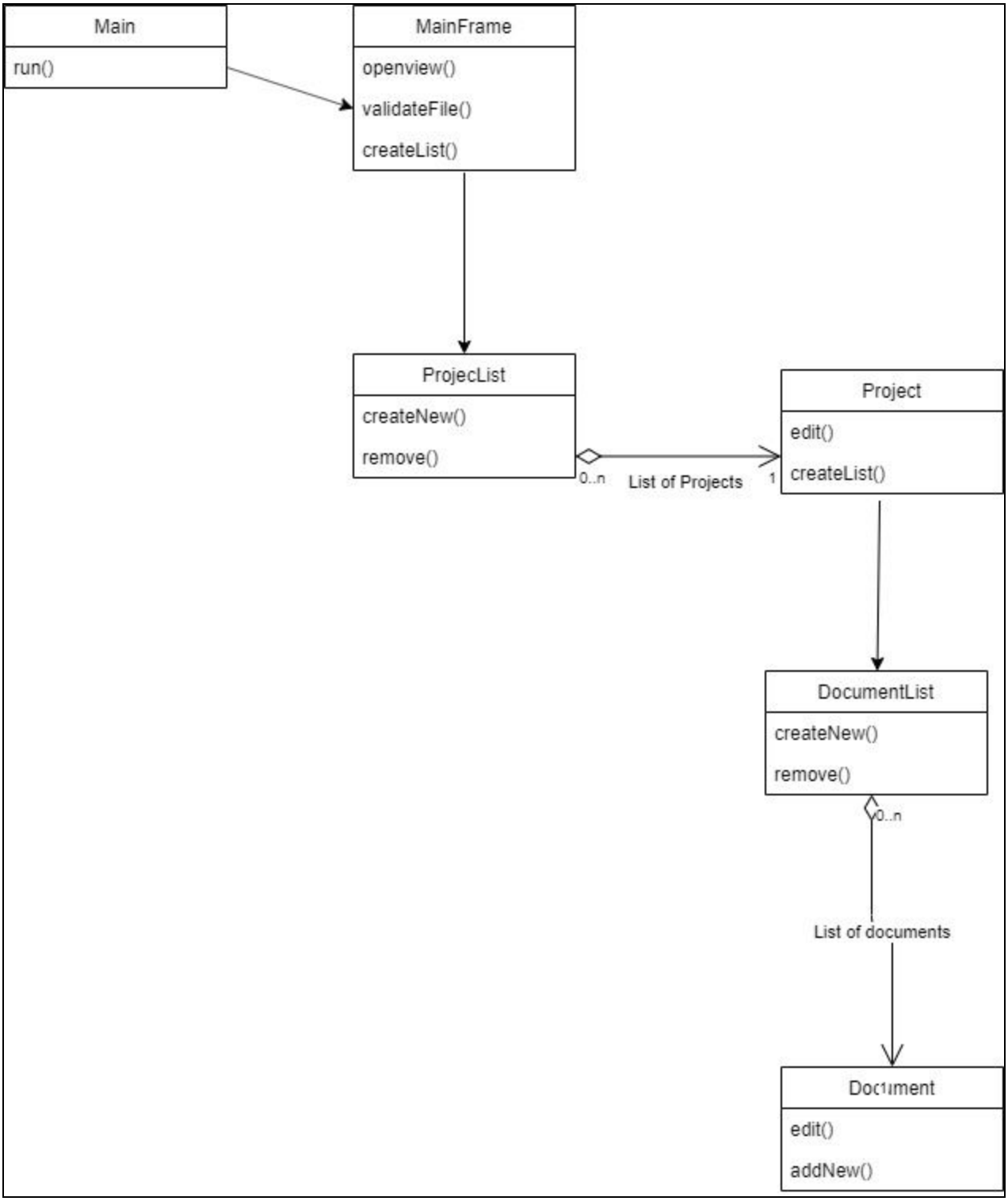
C. Rationale Summary

Our project is primarily using three different structures to represent the different parts of our program. The first part of our program that the user will interact with will be a list of projects. This list will be a list of Project types items which will hold all of the relevant information about each different project. We chose to implement this idea as a list so that it will be very easy to add and remove items as the user wants. This will also allow us to switch the nodes around and iterate through them when we are doing our searches. The relevant information inside these projects will be the name and description of each project.

These projects will each have the ability to create its own list of documents that pertain to the project. This list of document objects will also be easy to iterate through and easily changed. Being able to remove and change any of the documents and the projects is one of the primary characteristics of our design, so using Lists to represent each of these was an easy choice. The information stored inside the document class will be the location of the document on the machine, the description of the document, and the title of the document.

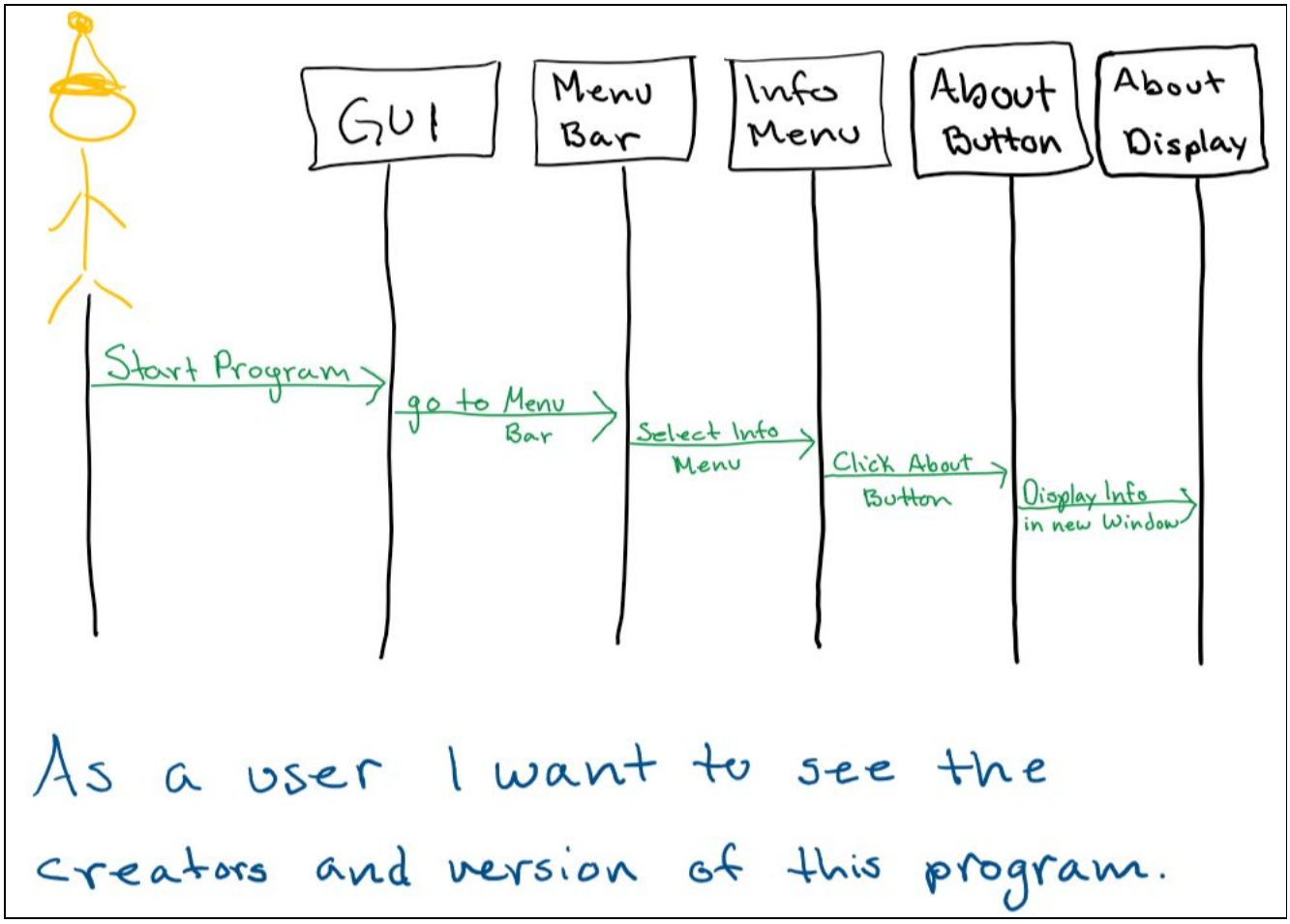
We chose these design decisions to try and reflect our object-oriented design. We wanted each part of our code to be as modular as possible so that when things are changed by the user, the changes are small and do not affect every aspect of the program. This will prevent a god class from happening because the top class will only be able to create the project list, then that project will be able to create a list of documents. This will also allow all of the data stored inside both of the lists to be protected. This data can only be accessed by the getter functions for the display in the GUI.

D. Class Diagram



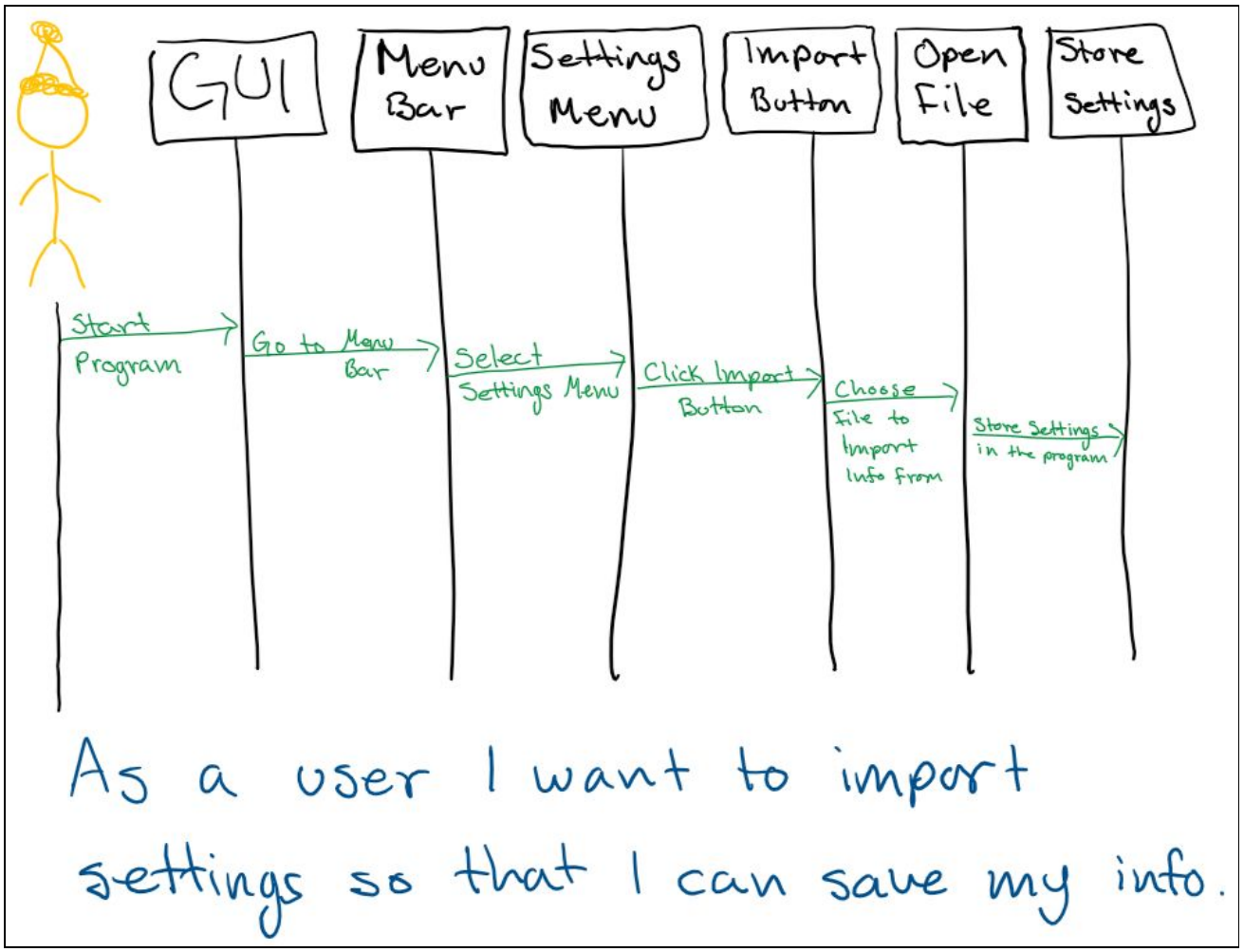
E. User Story Sequence Diagram

Diagram 1. About Screen Sequence Diagram



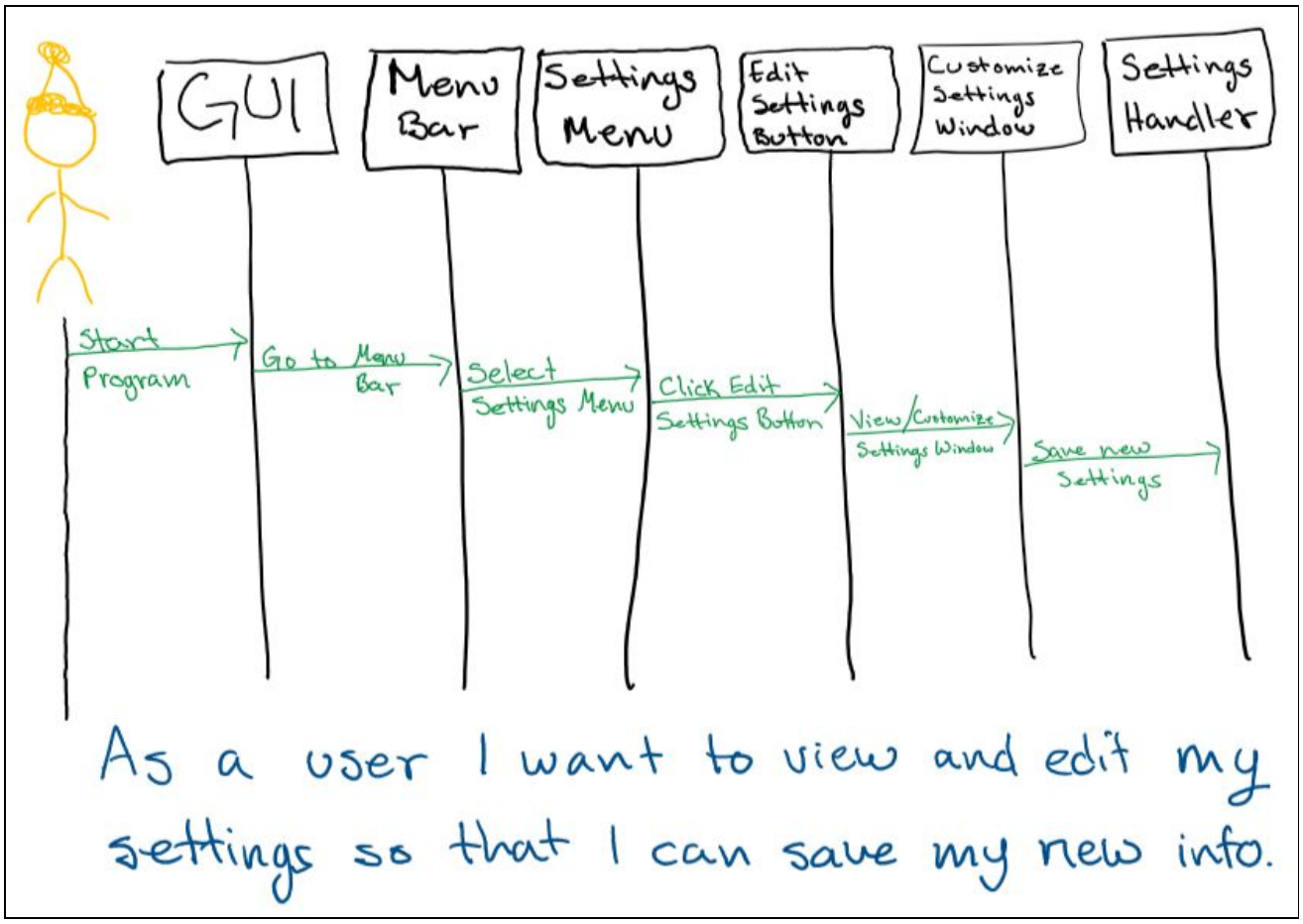
The user opens the program and navigates to the menu bar where they will find the word ‘Info.’ The user then clicks on the word ‘Info’, and a menu drops down. On that menu, the user will find the About Button which they will then click to bring up the About Screen. On the About Screen, the user will see the names of the creators of the program and the program version number.

Diagram 2. Import Settings Sequence Diagram



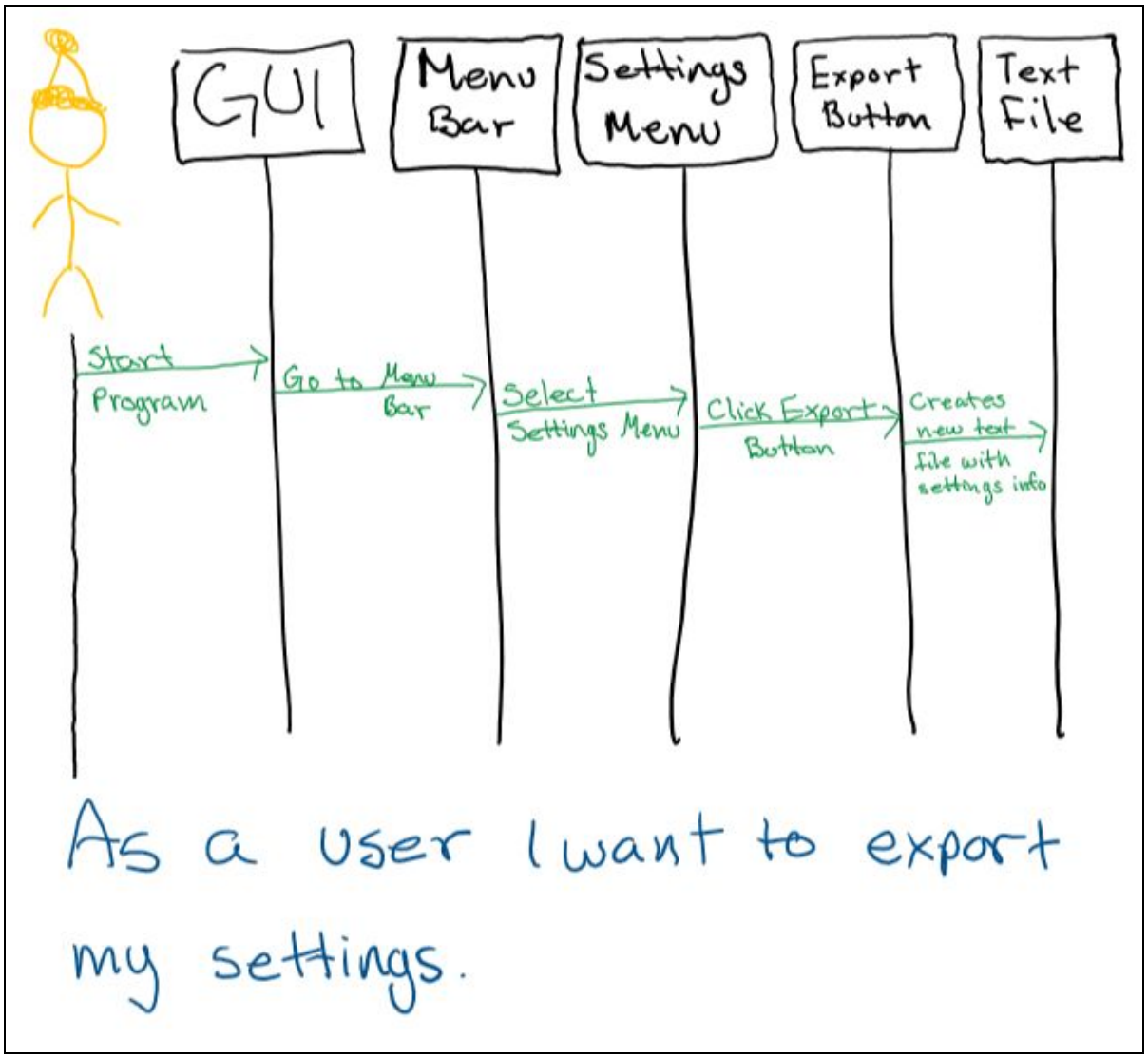
The user opens the program and navigates to the menu bar where they will find the word ‘Settings.’ The user then clicks on the word ‘Settings’, and a menu drops down. On that menu, the user will find the Import Button, which they will then click. A file chooser window will then pop up where the user can choose the text file to import. The program will then read the text file and save the contents as new user settings.

Diagram 3. View and Edit Settings Sequence Diagram



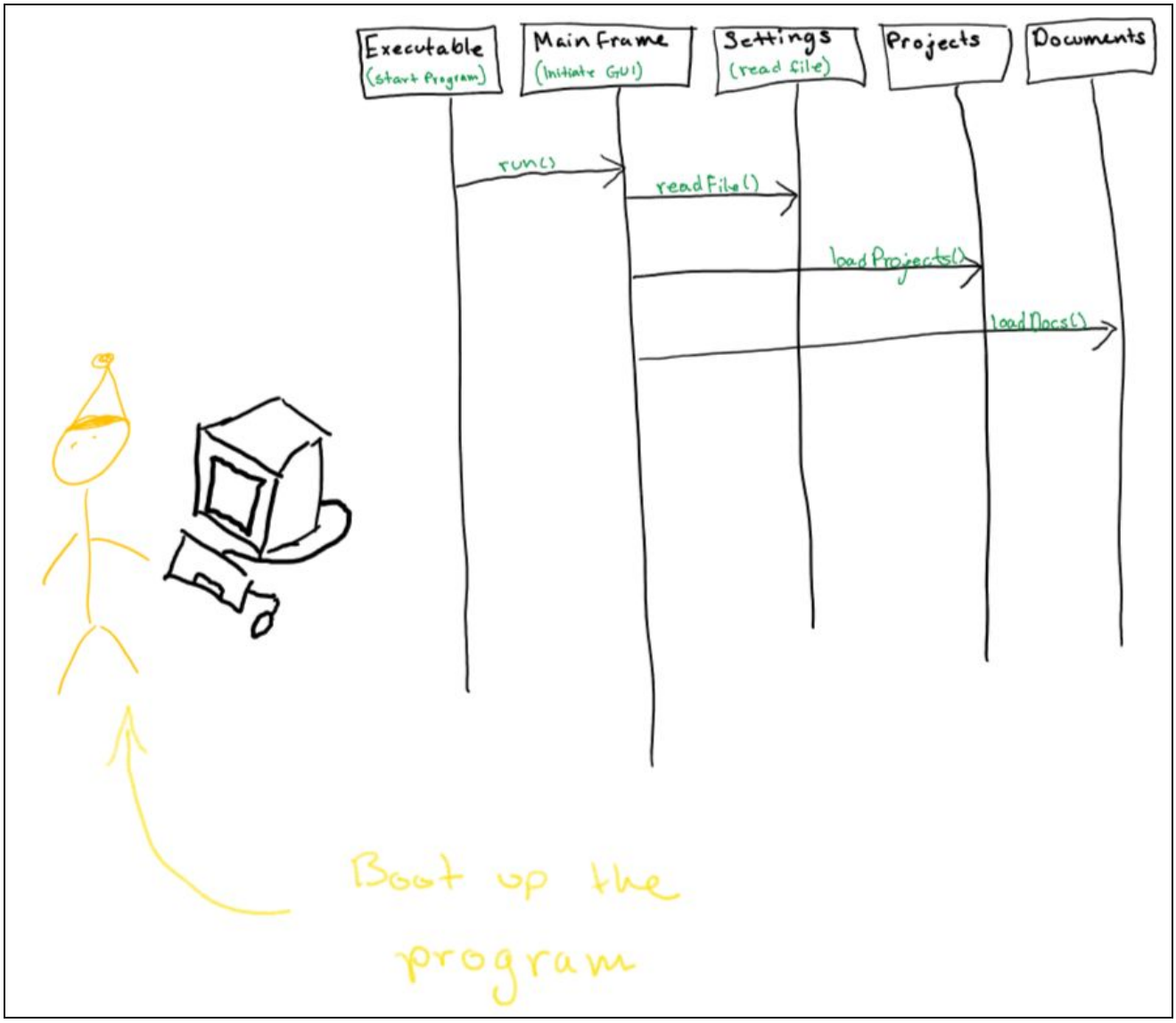
The user opens the program and navigates to the menu bar where they will find the word ‘Settings.’ The user then clicks on the word ‘Settings’, and a menu drops down. On that menu, the user will find the Edit Settings Button which they will then click. A new window will then pop up where the user can view or edit the current settings. The user then clicks ‘OK’ to save the settings as they are currently written in the edit box.

Diagram 4. Export Settings Sequence Diagram



The user opens the program and navigates to the menu bar where they will find the word ‘Settings.’ The user then clicks on the word ‘Settings’, and a menu drops down. On that menu, the user will find the Export Button which they will then click. A text file containing the user’s settings will then be exported and pop up for the user to view.

F. System Startup Sequence Diagram



(Start-Up Sequence Diagram)

When initializing the app, our Main Frame will construct all of the GUI elements in Tidy. As you can see an arrow that is point the Setting entity, this means that the GUI is loading an existing text file to display the About Us information and taking in any saved import files (commonly as a pdf file) so that Tidy can display them. Then Tidy will prepare the left Project panel to store user's projects by order of modified date. The users will be able to see the most recent projects on the topmost position of the vertical panel. Once each of the projects is appropriately created, Tidy then iterates a collection object in the Main Frame to load only

the document that belongs to that one project. It repeats the process until Tidy fills the document for all projects. By default, when loading and displaying any information, the Main Frame will always provide the aesthetic instruction to insert the document in the right position of the GUI frame. Moreover, although we did not insert one arrow from the Document entity that pointing to itself, we are planning to make the Document class to construct the preview information for itself automatically. This means that before a document is set visible, it will query extra information such as one-to-two summary sentences and last opened date.