* Document the flow of the application and prepare a flow chart
* List the core concepts and algorithms being used to complete this application

Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance

Application should not close, exit, or throw an exception if the user specifies an invalid input.

* Specification document - Product’s capabilities, appearance, and user interactions
* Java concepts being used in the project
* Data Structures where sorting and searching techniques are used.
* CREATE FLOW DIAGRAM WITH ALL THE CLASSES

Documentation

The purpose of this application is to create a prototype to determine the budget needed to create the real application. The prototype will contain the applications basic features such as:

* Generic features and three operations:
* Retrieving the file names in an ascending order
* Business-level operations:
  + Option to add a user specified file to the application
  + Option to delete a user specified file from the application
  + Option to search a user specified file from the application
  + Navigation option to close the current execution context and return to the main context

The appearance of the application will be pretty bare bones as it is only a prototype. The interface will be text base.

* After looking over the document, features of the application were put into user stories, and split into 3 sprints based on the amount of work they’re estimated to take as well as their relationship to other user stories.
  + As there is about a week left at the time of writing this, the duration of each sprint should take no more than 2 days
* For the most part, the requirements were pretty straight forward, but there are a few things that will need clarified for future reference.
  + If a user chooses to display files, are expected to display every file on the computer or simply the files in the current directory?
  + When adding, deleting, or searching for a file it is unclear whether or not case sensitivity is a requirement or merely a suggestion.
* **Sprint 1**
  + The majority of sprint 1 was actually setting up and dealing with documentation which took longer than expected.
  + All of the stories were completed fairly quickly. In hind sight, these stories probably weren’t worth a full point each, but rather a half or even a quarter of a point.
  + In hindsight a lot more could have gotten done in a single sprint
* **Sprint 2**
  + Like the stories in sprint 1, creating another menu as well as a way to get back to the main menu was made quickly without any issues.
  + Story #3 proved to be a bit more challenging. Not only was there a lot more coding involved, but a lot of moving parts as well as some functions I had to familiarize myself with. To save myself some time, I ended up reusing parts of the merge sort algorithm I created in a previous program to sort files for this program.
  + I also had to remind myself to keep functions contained within their own methods. Having a method with multiple functionalities creates dependencies and is overall a bad practice
* **Sprint 3**
  + With the beginning of a new work week, I was able to complete the first story much quicker than I expected. Overall, I’ve found out that working with files is much easier than I expected which has made many of the user stories throughout this entire project a lot more manageable.
  + For now, user story 5 ignores case sensitivity, but this can be changed later if desired.
  + User story 6 is similar to story 5 in a lot of ways so a lot of code could be reused.
  + I also changed both methods to return to the secondary menu regardless of whether or not the operation was successful. This is because I didn’t want users to be stuck adding/deleting files they didn’t intend to add/delete if the file had already been added/deleted