PicSorter

Software Design Document

Version 3.0

December 12, 2017

Team Members: Rayne Wang, Paige Hanssen, Jacob Tower, Ryan Sellar

RECORD OF CHANGES

*A - ADDED M - MODIFIED D – DELETED

VERSIO N NUMBER	DATE	NUMBER OF FIGURE, TABLE OR PARAGRAPH	A* M D	TITLE OR BRIEF DESCRIPTION	CHANGE REQUEST NUMBER
1.0	**11/05/2017**		A	Creation of SDD	
2.0	11/15/2017	2.1.2.1-2.1.2.4, 2.2.2.1, 2.2.2.2, 2.5	M	Broke down Activity, Sequence Diagrams, updated Class Diagram, removed Use Case diagrams	01
3.0	12/12/2017	Section 2	M	Updated diagrams	02

Table of Contents

1. Introduction 4
1.1. Purpose 5
1.2. Scope 5
1.3. Summary 5
1.4. Definitions, acronyms, abbreviations, and references 5
2. Design viewpoints 6
2.1. Process viewpoint 6
2.1.1. Design concerns 6
2.1.2. Design view 6
2.1.3. Design rationale 10
2.2. Logical viewpoint 10
2.2.1. Design concerns 10
2.2.2. Design view 10
2.2.3. Design rationale 14
2.3. Development viewpoint 14
2.3.1. Design concerns 14
2.3.2. Design view 14
2.3.3. Design rationale 15
2.4. Physical viewpoint 15
2.4.1. Design concerns 15
2.4.2. Design view 16
2.4.2. Design rationale 16
2.5. Scenarios viewpoint 16

Table of Figures

Figure 2.1.2.1: Activity Diagram #1………………………………………………………………6
Figure 2.1.2.2: Activity Diagram #2………………………………………………………………7
Figure 2.1.2.3: Activity Diagram #3………………………………………………………………8
Figure 2.1.2.4: Activity Diagram #4……………………………………………………….........9
Figure 2.2.2.1: Class Diagram……………………………………………………………………..11
Figure 2.2.2.2: Sequence Diagram#1……………………………………………………………..12
Figure 2.2.2.2: Sequence Diagram#2……………………………………………………………..12
Figure 2.2.2.2: Sequence Diagram#3……………………………………………………………..13
Figure 2.2.2.2: Sequence Diagram#4……………………………………………………………..13
Figure 2.3.2.1: Component Diagram…………………………………………………………....15
Figure 2.4.2.1: Deployment Diagram……………………………………………………………16

1. Introduction

1.1. Purpose

The purpose of the Software Design Document (SDD) is to specify the requirements for the design of the PicSorter application. This includes, but is not limited to, the selection of design languages to be used, as well as the requirements for documenting the design viewpoints. The SDD provides the necessary information and details for the system to be built.

1.2. Scope

The Software Design Document shall produce the base-level design system of the PicSorter application to provide a base level of functionality. Different design viewpoints shall develop the system in more depth. The document's viewpoints follow the 4+1 Viewpoint Model. The Process, Logical, Development, and Physical viewpoints shall describe the different functions of the application so that the implementation and software development process shall be as smooth and planned out as possible. The Scenarios viewpoint shall illustrate and validate the overall design architecture.

1.3. Summary

The Software Design Document provides documentation which will be used to aid in the development process by providing the details for how the software should be built. Within the SDD is the documentation of the software design for the project including use case models, class, sequence and activity diagrams, and more.

1.4. Definitions, acronyms, abbreviations, and references

GUI - Graphical User Interface
SDD - Software Design Document
UI - User Interface

[1] SRS: The Software Requirements Specification document section **2.1 System Context** illustrates both high-level and individual use cases that demonstrate the scenarios viewpoint.

2. Design viewpoints

2.1. Process viewpoint

The process view focuses on the run-time behavior of the software.

2.1.1. Design concerns

The main concerns involved in the process is how to manage decision branches and perform concurrent tasks. In this system, these are mainly deciding whether a given entry is a duplicate or not and making use of the two hash tables. These are the major decision branches. The only tasks which should be performed concurrently, or closely linked together, will be the addition of entries to both hash tables and the deletion and confirmation of the deletion of the entries which have been marked for deletion in the deletion array. The process as a whole is fairly linear and most decision branches lead to a loop until a yes is reached. For example, when finding files or when looking for duplicates. The only decisions which make a serious change to the actions of the system are the decisions made by the user. In indicating the directory the user has control over what images and subdirectories the system will make use of (or not make use of). In selecting an image as a duplicate the image is marked for later deletion (though the loop to look for duplicates continues). In finalizing changes the user has the final say over whether the files marked for deletion will be deleted or not. (For safety and in case of user mistakes, the files are actually moved to the recycle bin, trash bin, etc.)

2.1.2. Design view

Figure 2.1.2.1: Activity Diagram #1

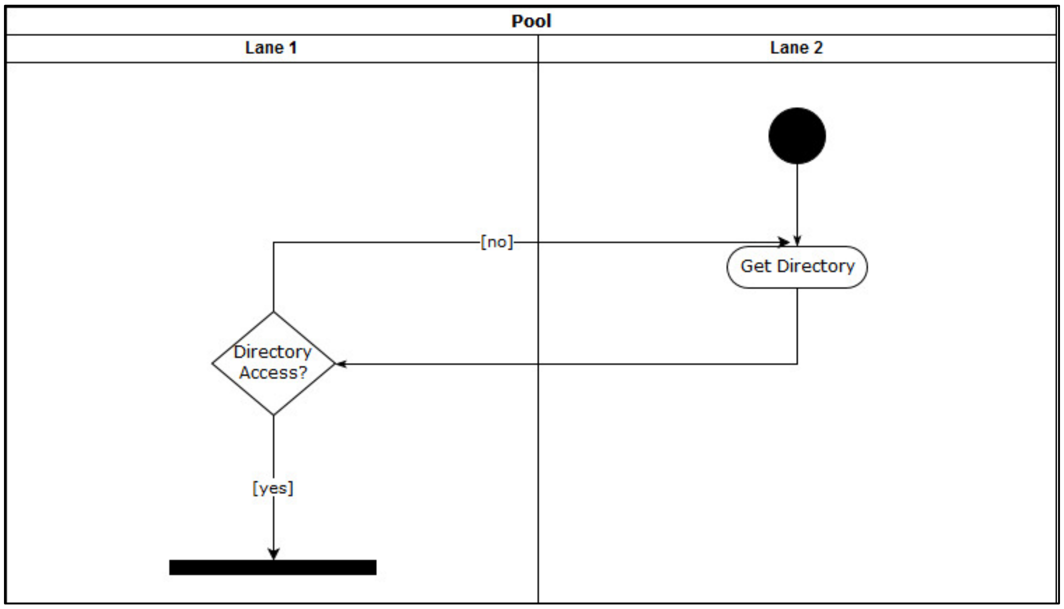


Figure 2.1.2.2: Activity Diagram #2

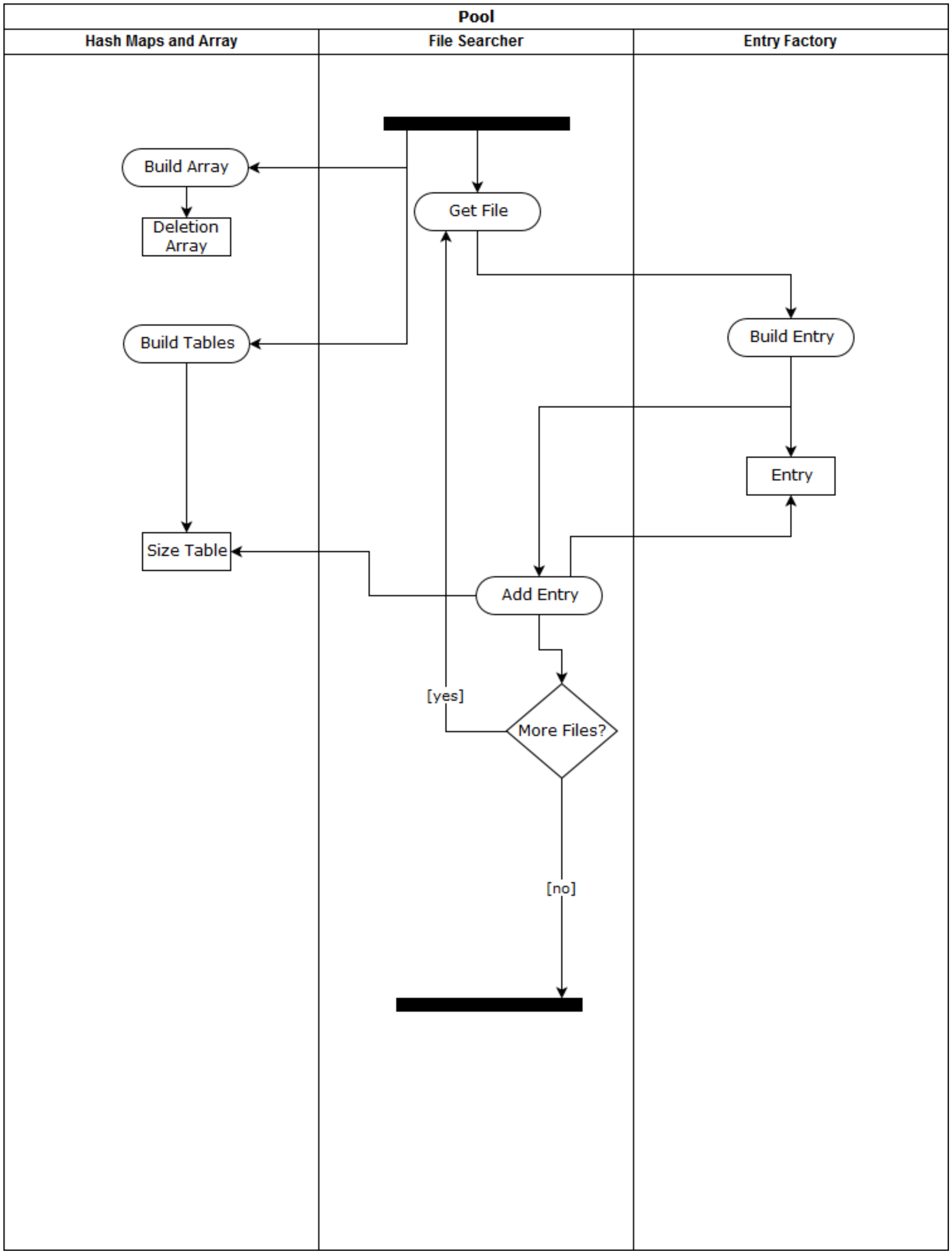


Figure 2.1.2.3: Activity Diagram #3

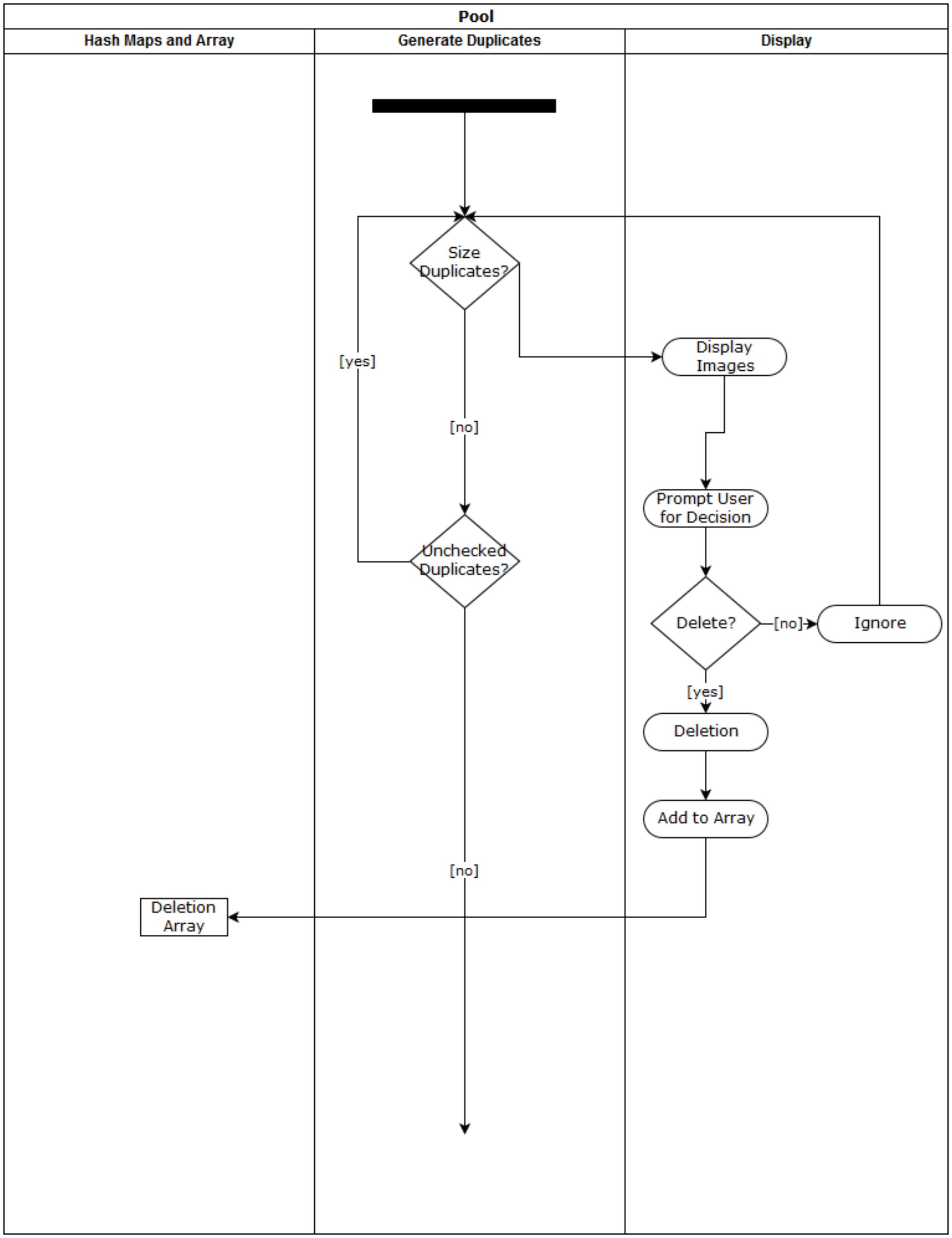
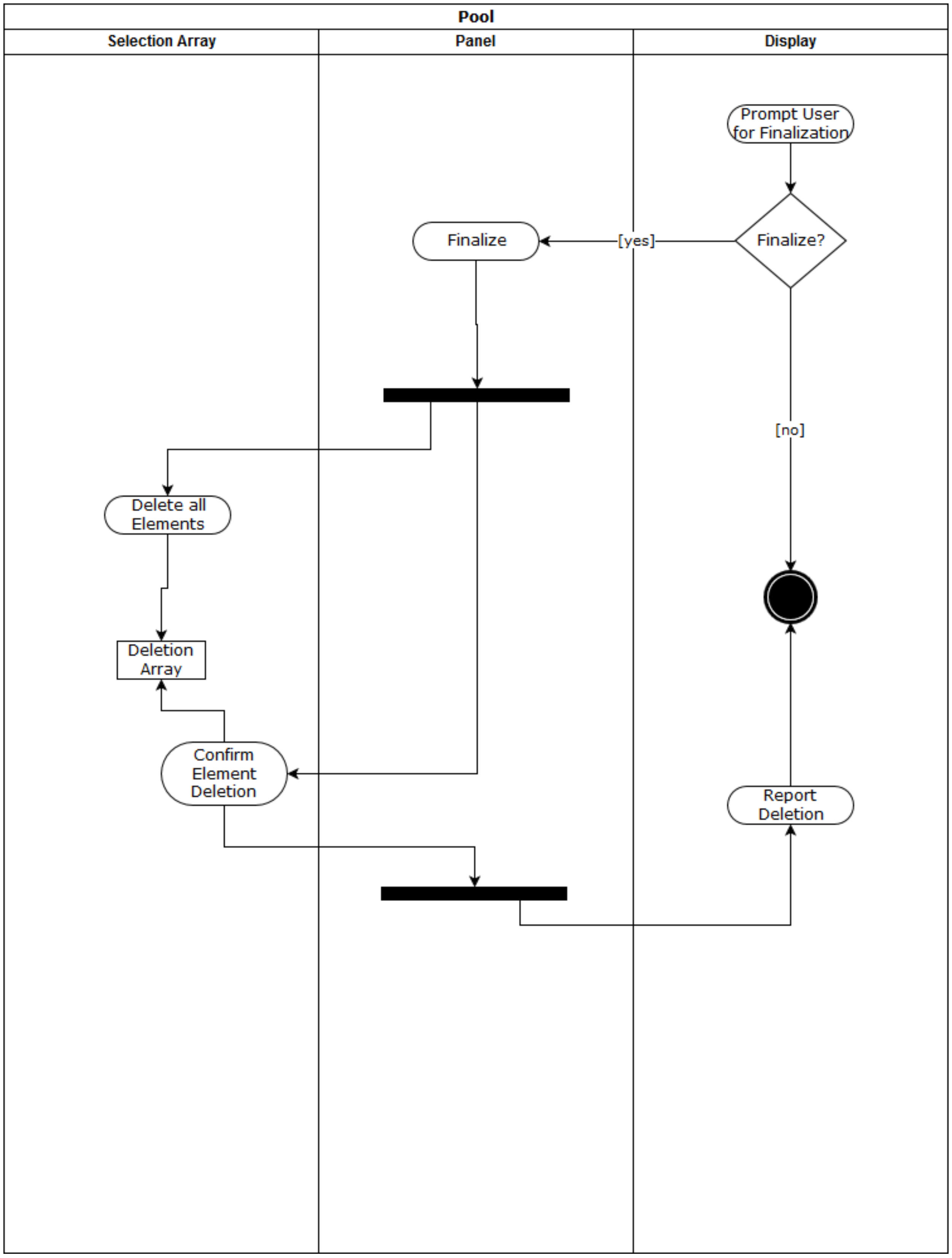


Figure 2.1.2.4: Activity Diagram #4



2.1.3. Design rationale

The overall process is split into three lanes: the frontend, the middle, and the backend. The frontend consists generally of any GUI features which take input from the user which is critical to the rest of the program. The middle covers the in-between details between the frontend and the backend as well as dealing with the directory and the creation of entries. The backend governs three data structures: one hash table ordered by file size and an array storing the addresses of entries marked for deletion. The process has been split up in this way to maximize specialization.

There are two main loops in our design: one in the middle when getting files from the directory and building entries, the second in the backend when checking for duplicates. The first is fairly simple: it continues getting new files from the directory until all files are exhausted. The second is a bit more complicated. It checks for duplicates using the file size hash table. If it finds duplicates, it presents all duplicates groups to the user. The user then has the final say for whether to delete any photos or not. The reason for using file size is because it is extremely unlikely that two files with the exact same number of bytes are not duplicates.

The use of an array to keep track of the duplicates marked for deletion is an easy way to save the entries for later use. The entries can be easily deleted simply by looping over the array and their deletion can be confirmed by checking their address on the computer (since the files themselves are ‘deleted’ by moving their location to the recycle bin). Finally, it also makes it easier for the user to examine all the images which have been marked for deletion when finalizing changes. The contents of the array (in a more appealing visual format) can be shown to the user so that he can make sure that he made the correct decisions. In case he has decided not to delete the files, the finalization may be cancelled and the system exits.

2.2. Logical viewpoint

The logical viewpoint shows the development and implementations of the software. It focuses on the main functional requirements of the system.

2.2.1. Design concerns

The main concern is the interaction between classes and how to make the software efficient. There are many classes and coupling is a concern. We want high cohesion and low coupling, but some classes such as the Display Class must have connections with multiple classes. We are also concerned with the speed of the software and decided to limit sorting to a size hash map to have quick access to duplicates.

2.2.2. Design view

Figure 2.2.2.1: Class Diagram

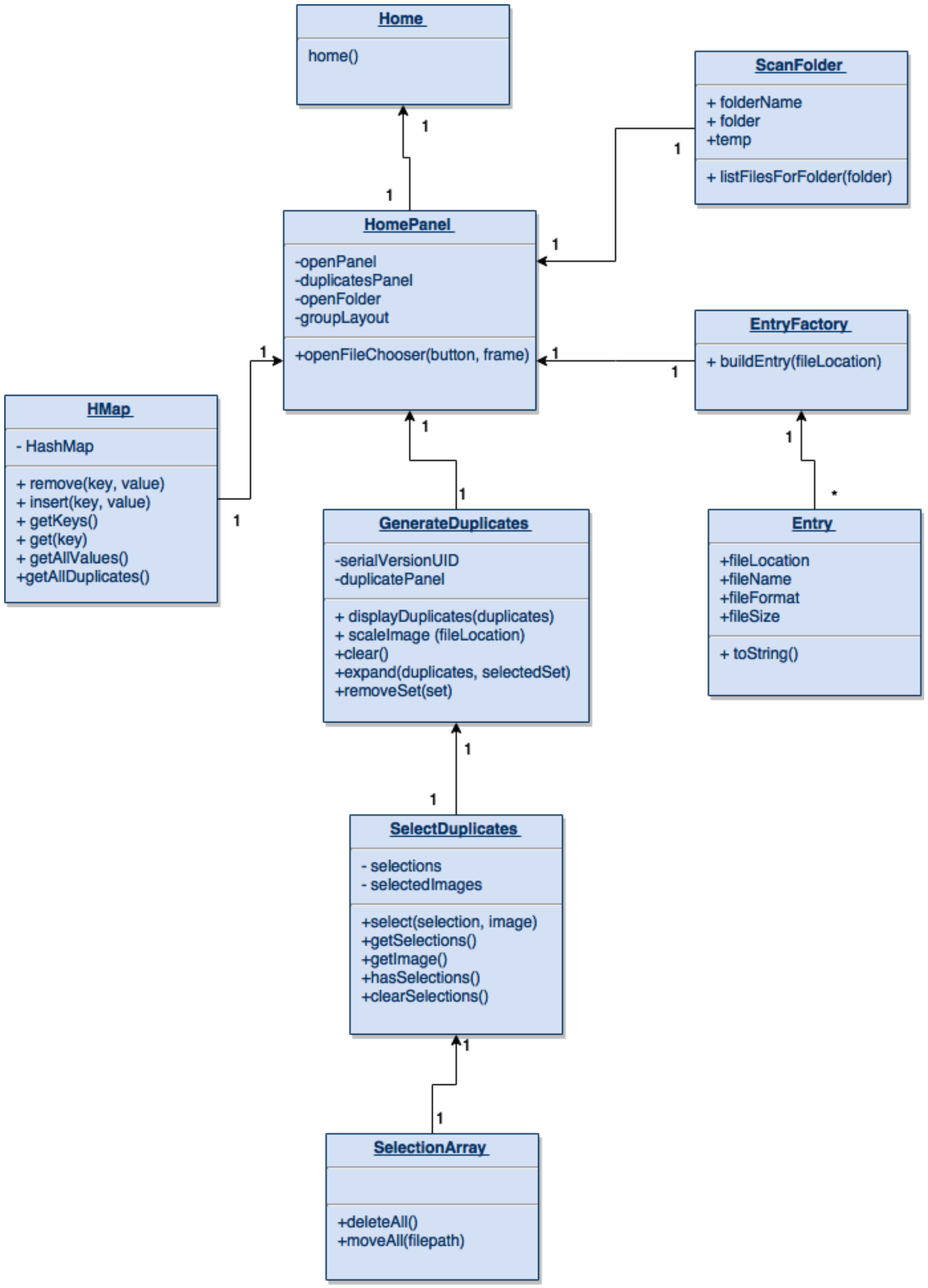


Figure 2.2.2.2: Sequence Diagram#1

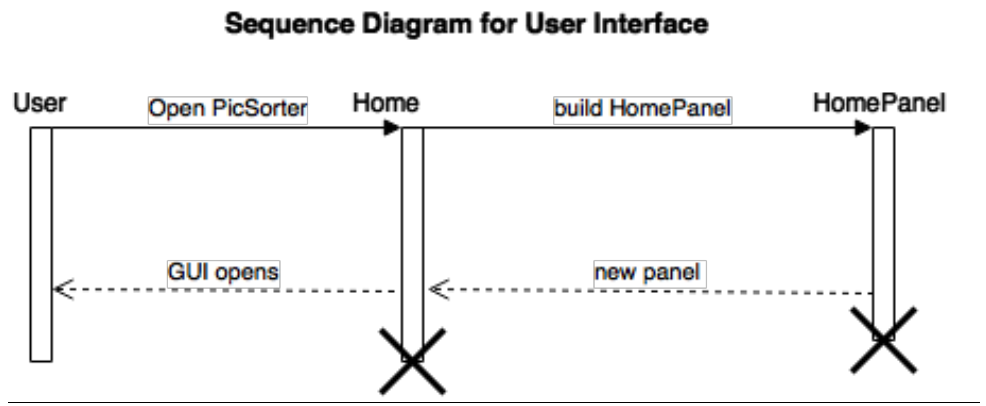


Figure 2.2.2.3: Sequence Diagram#2

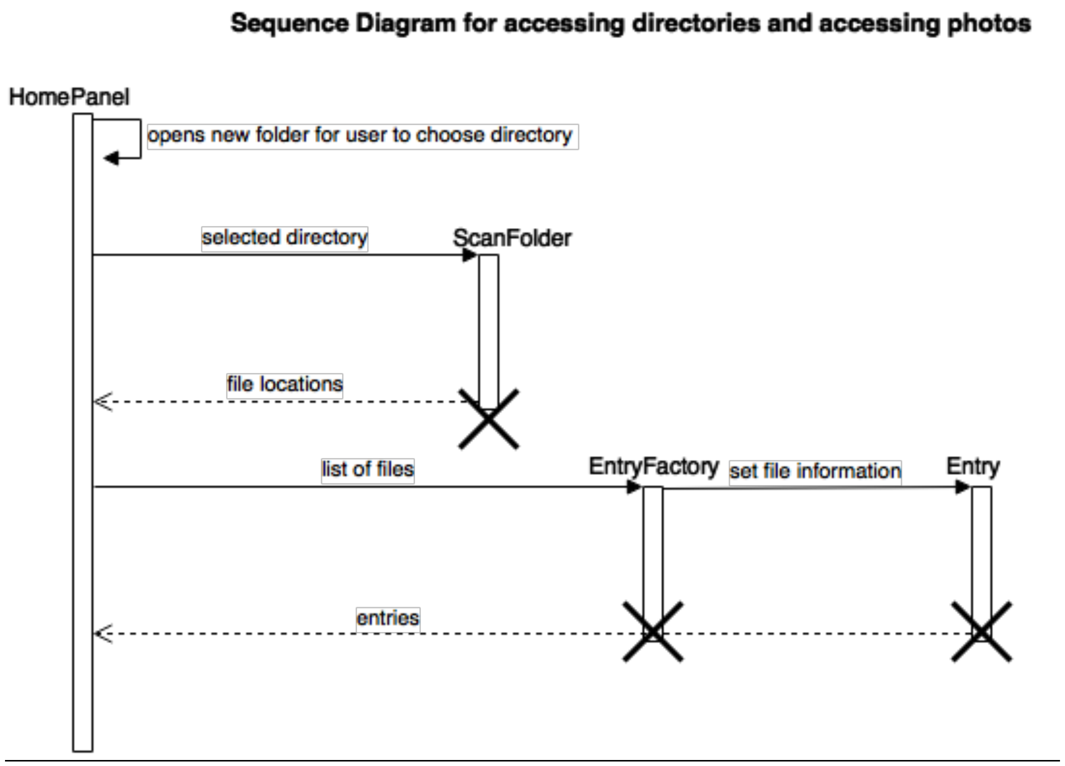

Sequence Diagram for accessing directories and accessing photos
HomePanel
opens new folder for user to choose directory
selected directory
ScanFolder
file locations
list of files
EntryFactory
set file information
Entry
entries

Figure 2.2.2.4: Sequence Diagram#3

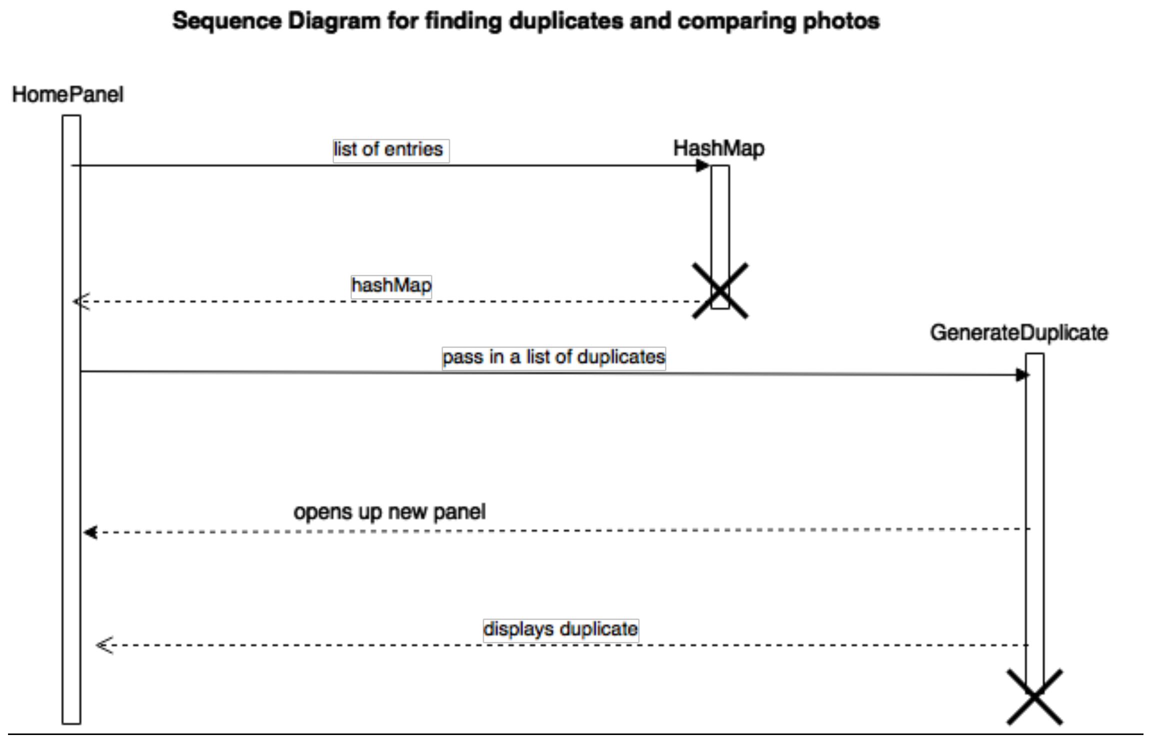
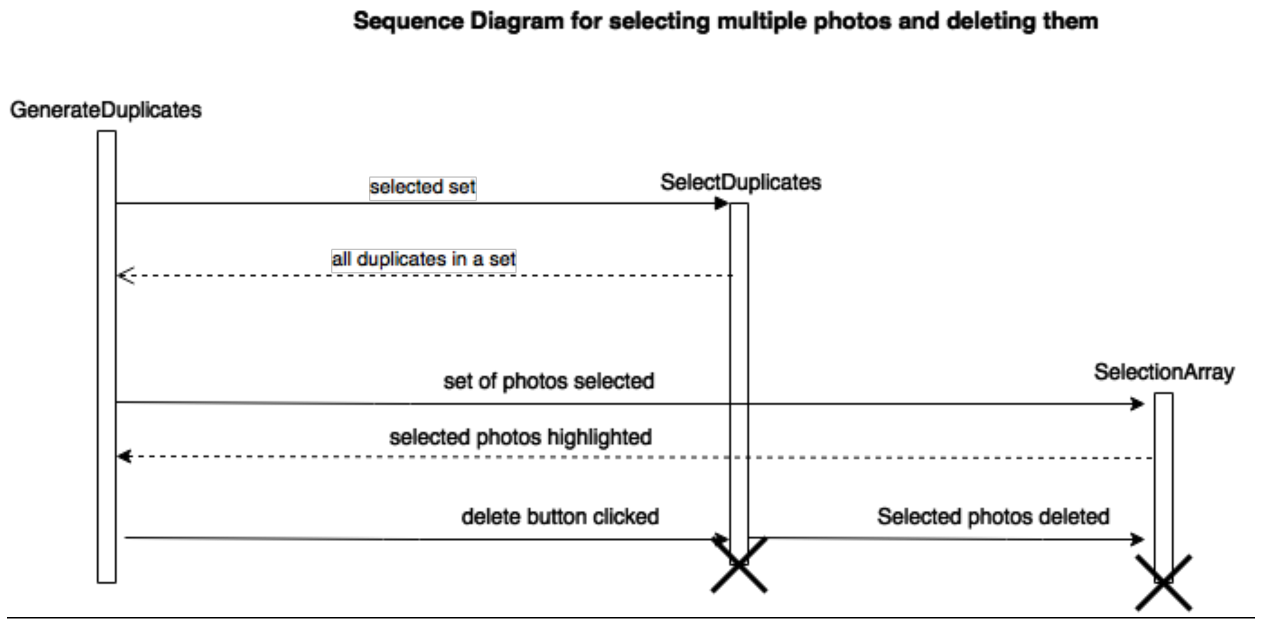


Figure 2.2.2.5: Sequence Diagram#4



2.2.3. Design rationale

The software classes are split into 2 main sections: user facing and background classes. User facing classes are Home, HomePanel, and GenerateDuplicates. These classes would allow the user to see the home screen, select a directory, and see the duplicates. The background classes are HashMaps, SelectDuplicate, SelectionArray, ScanFolder, Entry, and EntryFactory. The ScanFolder will extract all the file locations in the selected directory. Using the file locations, the EntryFactory creates Entriees that saves the photo information such as file address and file size. The hashmap is used to sort the photos by file size. The SelectDuplicate class will allow users to select photos to delete and SelectionArray will then keep track of the photos selected. These background classes will interact with the user facing classes to make the program functional.

2.3. Development viewpoint

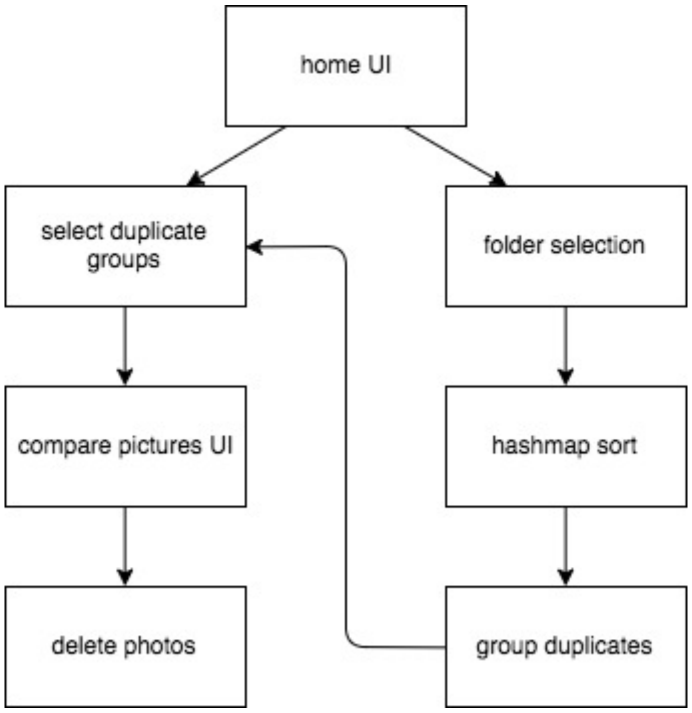
The Development viewpoint focuses on the static organization of the software in its development environment.

2.3.1. Design concerns

The development view provides an overall picture of the design subject in order to look at the impact of requirements or design changes. It can help people maintaining the application to find components causing system failures and minimize bottlenecks. It can aid in producing the system integration plan by identifying the components that are needed by other components, and the order in which they must be developed.

2.3.2. Design view

Figure 2.3.2.1: Component Diagram



2.3.3. Design rationale

How the system components interact with each other can be represented at a high level, starting with the main component - the GUI that the user encounters when first starting up the application. From the home component, a user can access the folder selection component. After opening the directory and selection a folder, the items inside selected folder are sent to the hashmap sorting function, where duplicates grouped. Those groups can then be accessed from the home component in the selection of duplicate groups. The duplicate folder group components opens another window to display the duplicates in the same group. Inside that component, a user can delete photos.

2.4. Physical viewpoint

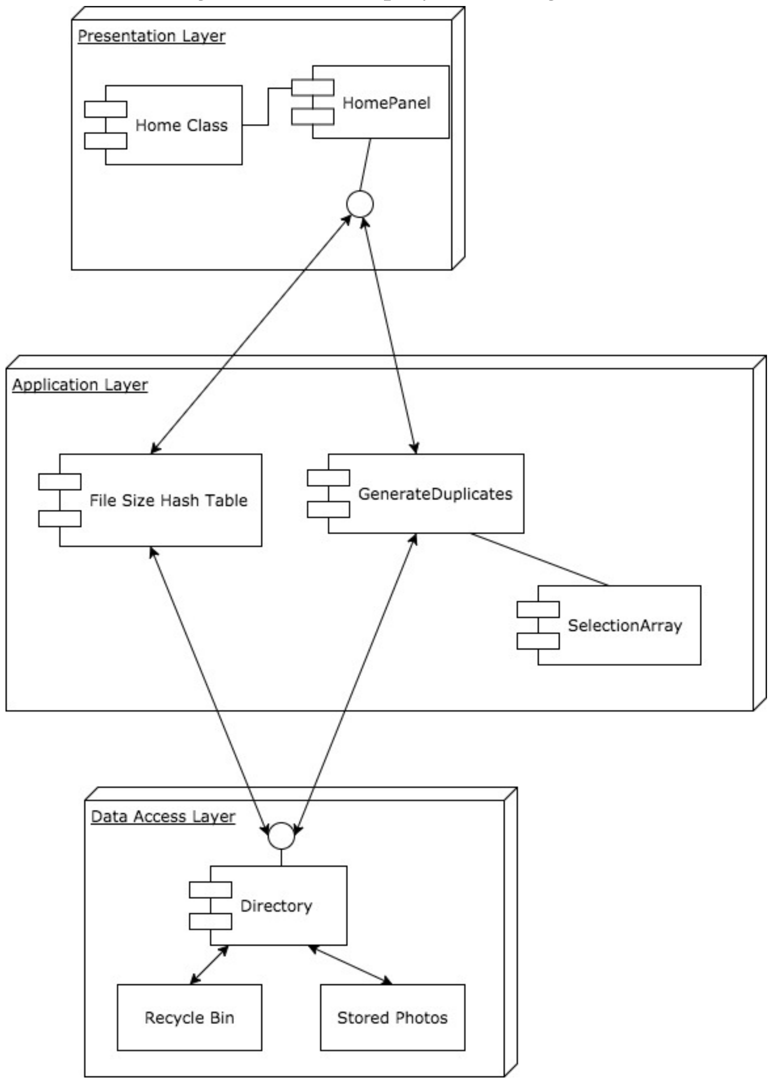
The physical view focuses on the system's topology, communication, and deployment.

2.4.1. Design concerns

The major concerns involved are the interactions between the GUI, the application software, and the directory of the user. The application is responsible for communicating with the directory to retrieve and sort the photos, and then updating the GUI. The GUI will handle user input and display, with its most important functionality being to indicate to the application that the user wishes to search for a duplicate of a selected picture. The application will pull photos from the directory, perform all sorting, and store references to photos in two different hash tables. It must be ready to respond to a user input from the GUI.

2.4.2. Design view

Figure 2.4.2.1: Deployment Diagram



2.4.2. Design rationale

The software follows a three-tier architecture design, as the system can be accurately represented as three distinct interacting components - the GUI, the application software, and the user's directory. The directory contains all of the user's photos, and must be accessed and traversed in order to search for duplicates. The application will handle these actions, and store references to the filename and size of the photos in the file size hash tables. The application must be able to interact with the directory in order to request photos and perform the traversal. In addition to this, the user will have the ability to delete and move photos through the application, so the application must also be able to notify the directory of this and delete a specific photo, or change the file path. The GUI must allow the user to interact with the software by displaying the photos or duplicates that the user wishes to view, and indicating to the application which operations should be performed and on what duplicate. It must be responsive to user input and be able to relay inputs to the application.

2.5. Scenarios viewpoint

The scenarios viewpoint illustrates and validates the overall architecture of the system. Refer to the SRS[1] for use case scenarios.