**PicSorter**

**Software Design Document**

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

**Version: 1.0** **Date: (11/05/2017)**

Table of Contents

[1. Introduction 2](#_Toc497649508)

[1.1. Purpose 3](#_Toc497649509)

[1.2. Scope 3](#_Toc497649510)

[1.3. Summary 3](#_Toc497649511)

[1.4. Definitions, acronyms, and abbreviations 3](#_Toc497649512)

[2. Design viewpoints 4](#_Toc497649513)

[2.1. Process viewpoint 4](#_Toc497649514)

[2.1.1. Design concerns 4](#_Toc497649515)

[2.1.2. Design view 5](#_Toc497649516)

[2.1.3. Design rationale 6](#_Toc497649517)

[2.2. Logical viewpoint 6](#_Toc497649518)

[2.2.1. Design concerns 6](#_Toc497649519)

[2.2.2. Design view 7](#_Toc497649520)

[2.2.3. Design rationale 8](#_Toc497649521)

[2.3. Development viewpoint 9](#_Toc497649522)

[2.3.1. Design concerns 9](#_Toc497649523)

[2.3.2. Design view 9](#_Toc497649524)

[2.3.3. Design rationale 10](#_Toc497649525)

[2.4. Physical viewpoint 10](#_Toc497649526)

[2.4.1. Design concerns 10](#_Toc497649527)

[2.4.2. Design view 11](#_Toc497649528)

[2.4.2. Design rationale 11](#_Toc497649529)

[2.5. Scenarios viewpoint 12](#_Toc497649530)

Table of Diagrams

Diagram 2.1.2.1: Activity Diagram………………………………………………………………………5

Diagram 2.2.2.1: Class Diagram.…….…………………………………………………………………7

Diagram 2.2.2.2: Sequence Diagram…..………………………………………………………………8

Diagram 2.3.2.1: Component Diagram…………………………………………………………………9

Diagram 2.4.2.1: Deployment Diagram…….…………………………………………………………11

Diagram 2.5.1: Use Case Diagram #1 ………………………………………...…………..…………12

Diagram 2.5.2: Use Case Diagram #2 …………………………………………...………..…………12

Diagram 2.5.3: Use Case Diagram #3………………………………..………………………………13

Diagram 2.5.4: Use Case Diagram #4..………………………………………………………………13

# 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, and abbreviations

GUI - Graphical User Interface

SDD - Software Design Document

UI - User Interface

# 2. Design viewpoints

## 2.1. Process viewpoint

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

### 2.1.1. Design concerns

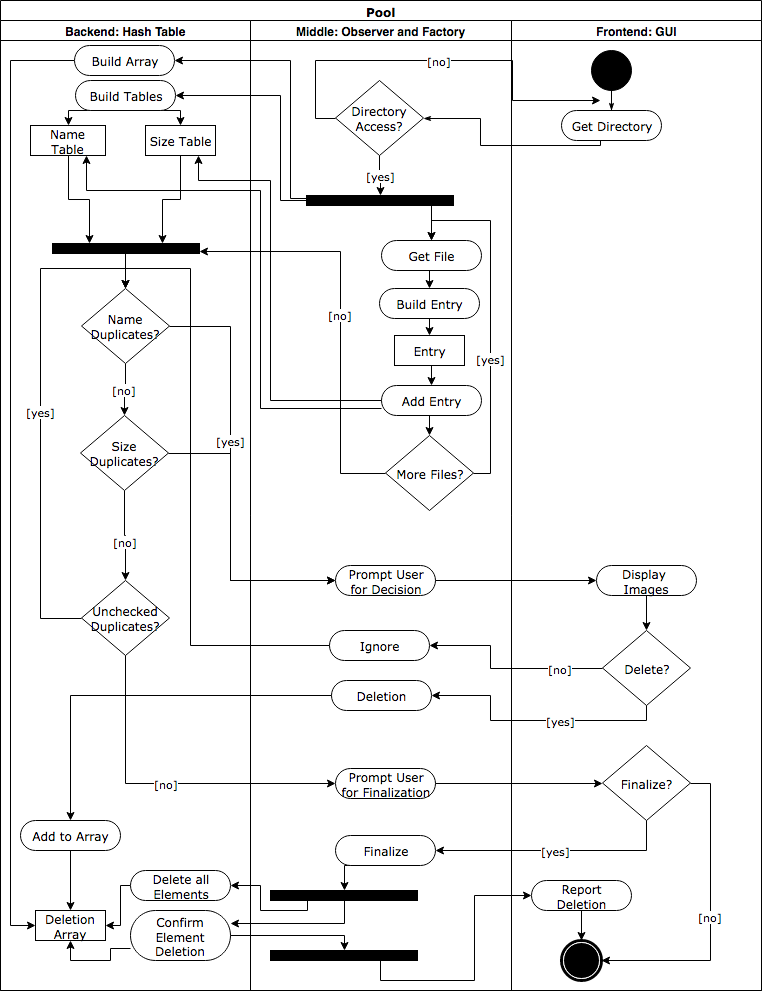
The main concerns involved in the process is how to manage decision branches and perform concurrent tasks. In this system, these are mainly deciding whether a given entry is a duplicate or not and making use of the two hash tables. These are the major decision branches. The only tasks which should be 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

Diagrams 2.1.2.1: Activity Diagram



### 2.1.3. Design rationale

The 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 filename, 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.

Two main loops can be seen: 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 first checks for filename duplicates using the file name hash table. If it finds duplicates, then it presents them to the user. The user then has the final say for whether to delete or not. The reason for filename going straight to the prompt without checking file size is too allow for pictures that are similarly named and could be near duplicates. In addition, if the files are in different subdirectories, they could be exact duplicates even in name. This is then a quick way of determining potential duplicates. Once there are no more name duplicates the loop will check for size duplicates. This is another easy way to check for exact duplicates. 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 two hashmaps to have quick access to duplicates.

### 2.2.2. Design view

Diagram 2.2.2.1: Class Diagram

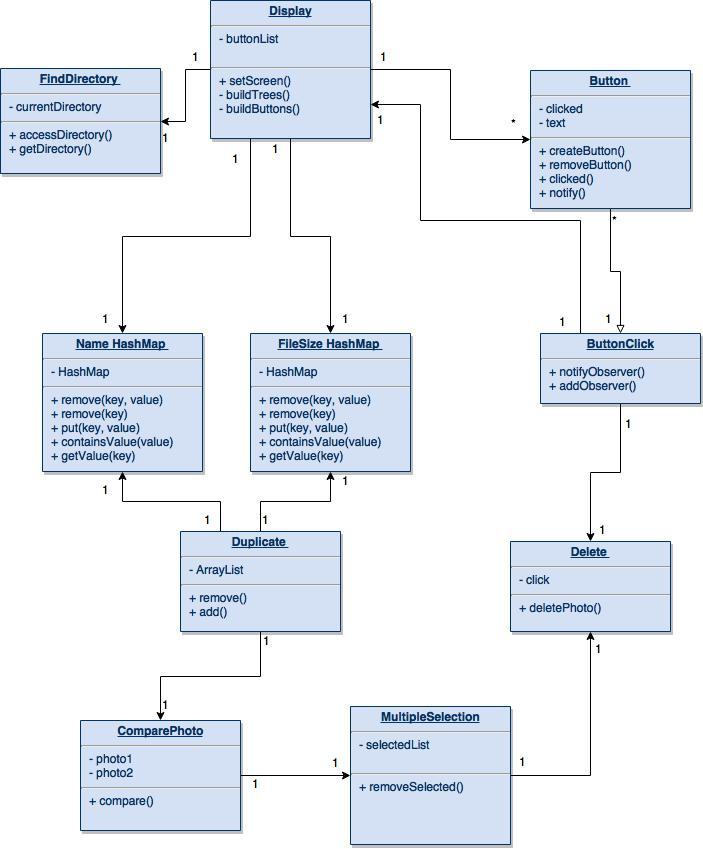
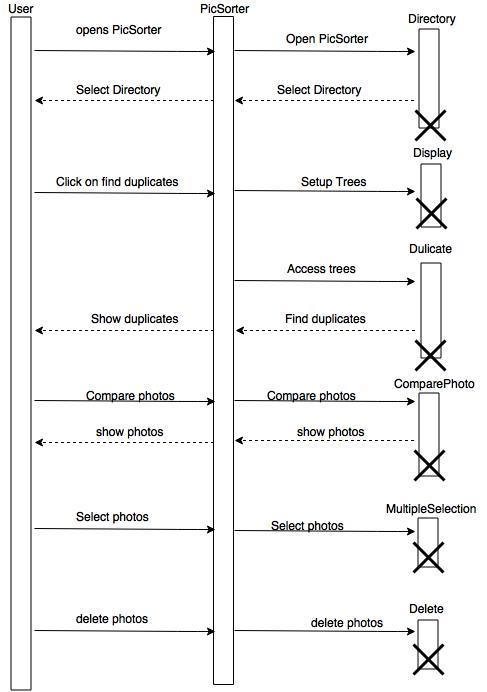


Diagram 2.2.2.2: Sequence Diagram



### 2.2.3. Design rationale

The software classes are split into 2 main sections: user facing and background classes. User facing classes are Display, Directory, and Button. These classes would allow the user to see the software, select a directory, and click on buttons. The background classes are Name and FileSize HashMaps, Duplicate, ComparePhoto, Delete, MultipleSelection, and ButtonClick. The two hashmaps are used to sort the photos by name and file size. The duplicate class will then use the hashmaps to determine which photos are exact duplicates. ComparePhoto would then allow the user to compare duplicates side by side and decide if they want to delete or keep the photos. The MultipleSelection class will allow the user to select multiple photos and the Delete class will delete the selected photos. All button clicks will be tracked by ButtonClick class who will notify observing classes to act when the appropriate button is clicked.

## 2.3. Development viewpoint

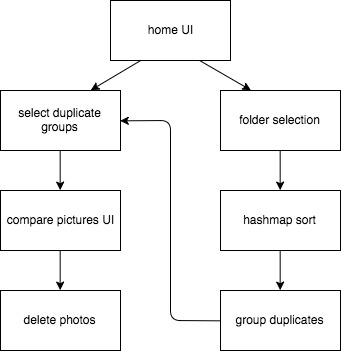
The Development viewpoint focuses on the static organization of the software in its development environment.

### 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

Diagram 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 interface, the user can access the duplicate folder group components, which opens the compare pictures GUI. Inside that component, a user can delete photos.

From the home component, a user can also 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 are stored in groups. Those groups can then be accessed from the home component in the selection of duplicate groups.

## 2.4. Physical viewpoint

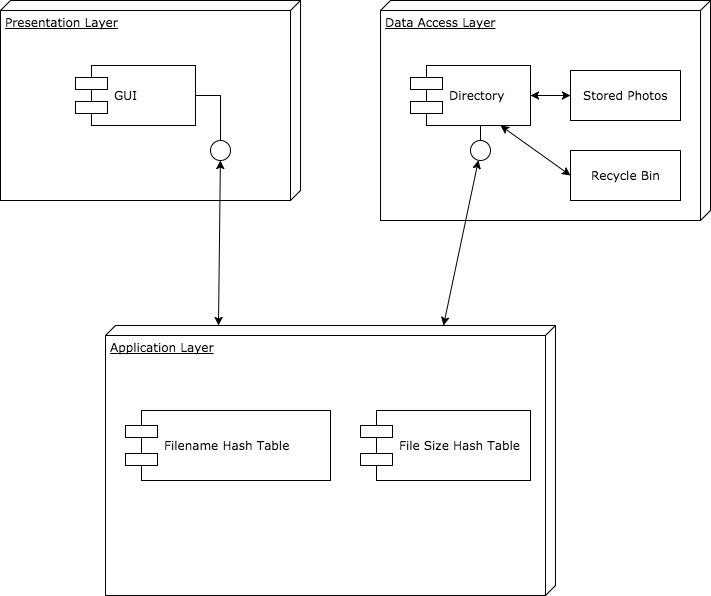
The physical view focuses on the system’s topology, communication, and deployment.

### 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

Diagram 2.4.2.1: Deployment Diagram

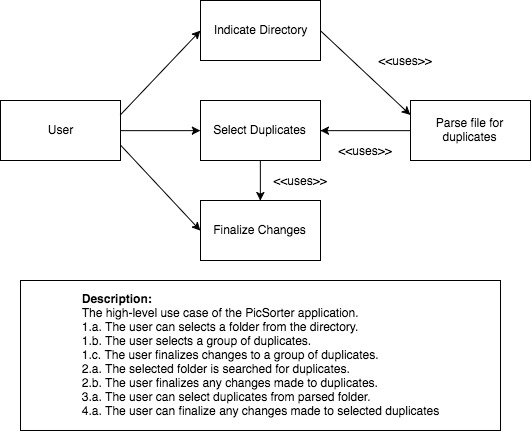


### 2.4.2. Design rationale

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 two separate 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 photos through the application, so the application must also be able to notify the directory of this and delete a specific photo. The GUI must allow the user to interact with the software by displaying the photos or duplicates that the user wishes to view. 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.

Diagram 2.5.1: Use Case Diagram #1

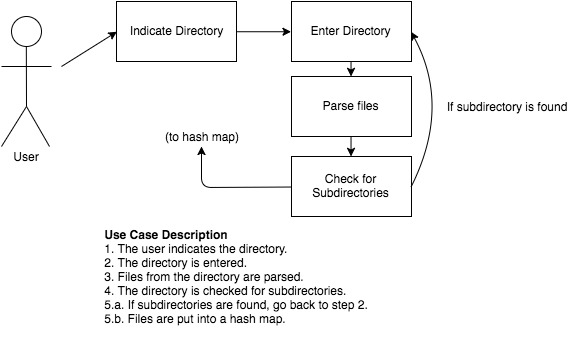
Diagram 2.5.2: Use Case Diagram #2

Diagram 2.5.3: Use Case Diagram #3

### Use Case Diagram 2.1.2 and 2.1.3.pngUse Case Diagram 2.1.2 and 2.1.3.pngDiagram 2.5.4: Use Case Diagram #4