Requirements and Analysis Document for Desktop Hipster

Version: Final
Date: 2014-05-20
Author: Edvard Hübinette

This version overrides all previous versions

1. INTRODUCTION

Desktop software for managing pictures, applying basic filters and upload to social media sites.

1.1. PURPOSE OF APPLICATION

To simplify editing and managing pictures on the users computer and offer a streamlined solution for uploading the finished images to social media websites like tumblr or Twitter. Currently the most widely-used key paths to do so uses several applications in the chain of events as well as separate, online uploading tools for each site.

1.2. GENERAL CHARACTERISTICS OF APPLICATION

A minimalistic and streamlined application designed for simple usage, with a gentle learning curve designed with a main focus on simplicity for the user.

1.3. SCOPE OF APPLICATION

The application only provides simple, pre-designed filters and uploading to sites of the developers' choice. The application is developed for OS X.

1.4. OBJECTIVES AND SUCCESS CRITERIA

- It should be possible to browse imported pictures and add tags for identification
- It should be possible to upload a picture to at least one image website
- It should be possible to add at least basic image enhancements to a picture

1.5. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Filter – A graphical image enhancement applied to a given image.

Host – In this context a host is the web service to which we upload the image.

Butcons – An icon that also is a button.

Java – Commonly used programming language that is supported on several platforms.

GUI – Graphics User Interface, all the graphics that are presented to the user.

.jar – A Java executable file.

JRE – Java Runtime Environment, the file needed to run a Java application.

OS X – The operating system on a Mac.

.dmg – A installation package used by computers with OS X.

.app – An application file that could be ran on a computer with OS X.

GPL – General Public License, a public copyright license.

Tumblr – A social media website.

Twitter – A social media website.

API – Application Programming Interface, documentation on how to use a software package.

2. REQUIREMENTS

In this section we specify all requirements.

2.1. FUNCTIONAL REQUIREMENTS

The user should be able to:

- Import an image form the file system via system dialog
- Browse imported images' thumbnails
- Select an image and take it to the editing stage
- Apply a basic filter
- Either:
    - Save image locally
    - Upload image to online host
- Exit application will save the image library

2.2. NON-FUNCTIONAL REQUIREMENTS

2.2.1. USABILITY

Usability is prioritized; application should be self-explanatory and have an easy learning curve. Simple UI and helpful butcons will contribute to fulfilling this. Safe exploration enabled through filter previews and preserved original images.

2.2.2. RELIABILITY

There should not for the user to lose his/her work, the image library should always be saved when application exits. Shutdown hook will ensure application can always exit gracefully.

2.2.3. PERFORMANCE

Any actions should not lead to major perceived lag. Uploading stage and importing images are allowed to differ from this since disk read/write times and Internet connectivity are parameters out of reach.

2.2.4. SUPPORTABILITY

Application is developed in Java, which makes portability to other platforms simple, the application is however developed for OS X because the lack of a testing environment for Windows and interest in optimizing for OS X system controls.

There will be automated test for verifying use cases. GUI testing performed manually.

2.2.5. IMPLEMENTATION

Application will create its own library folder for output files in the users file system. Reasoning for this is that we never interfere with the original image, we don't want the .jar to get bigger and we want central image storage. Users need JRE installed and configured for the application to work.

2.2.6. PACKAGING AND INSTALLATION

The aim is to follow OS X standards application will be delivered as a .dmg volume, containing an .app file for the user to drag into the Applications folder.

2.2.7. LEGAL

There are no apparent legal issues regarding the application if the filters are developed form scratch and/or use GPL-licensed software. Tumblr, Twitter or similar logos are fine to use when it's in combination with their API and their services.

2.3. APPLICATION MODELS

2.3.1. USE CASE MODEL

Use case UML for application:

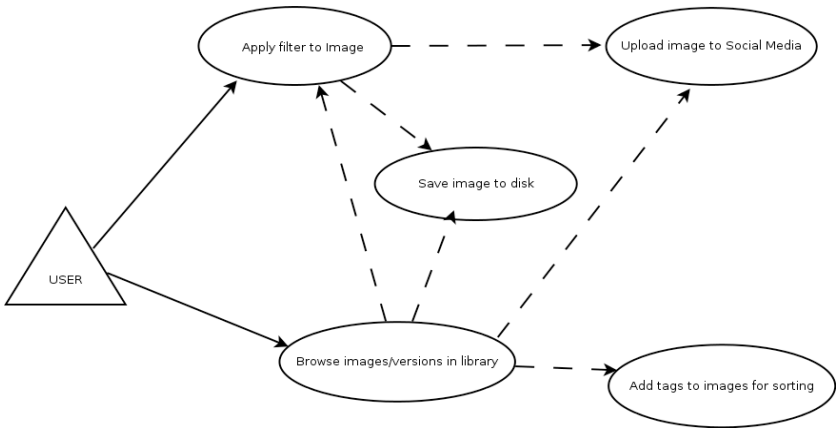

Figure 1: Diagram over the use case model

2.3.2. USE CASE NAMES:

- Normal application flow
- User not authenticated for upload
- User saves image to default library
- User saved image to custom directory
- User imports file through file-system browser
- User directly selects image
- Application welcome tutorial

2.3.3. USE CASES PRIORITY

1. User directly selects image
2. User imports file through file-system browser
3. User saves to default library
4. User saved image to custom directory
5. Application welcome tutorial
6. Normal application flow

2.3.4. DOMAIN MODEL

The domain model is a low detailed model of the application.

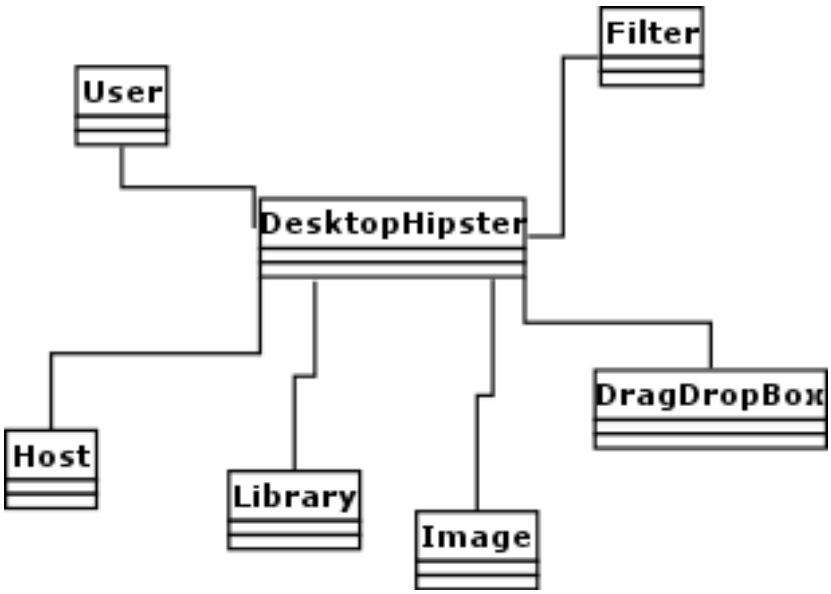

Figure 3: Diagram over the domain model

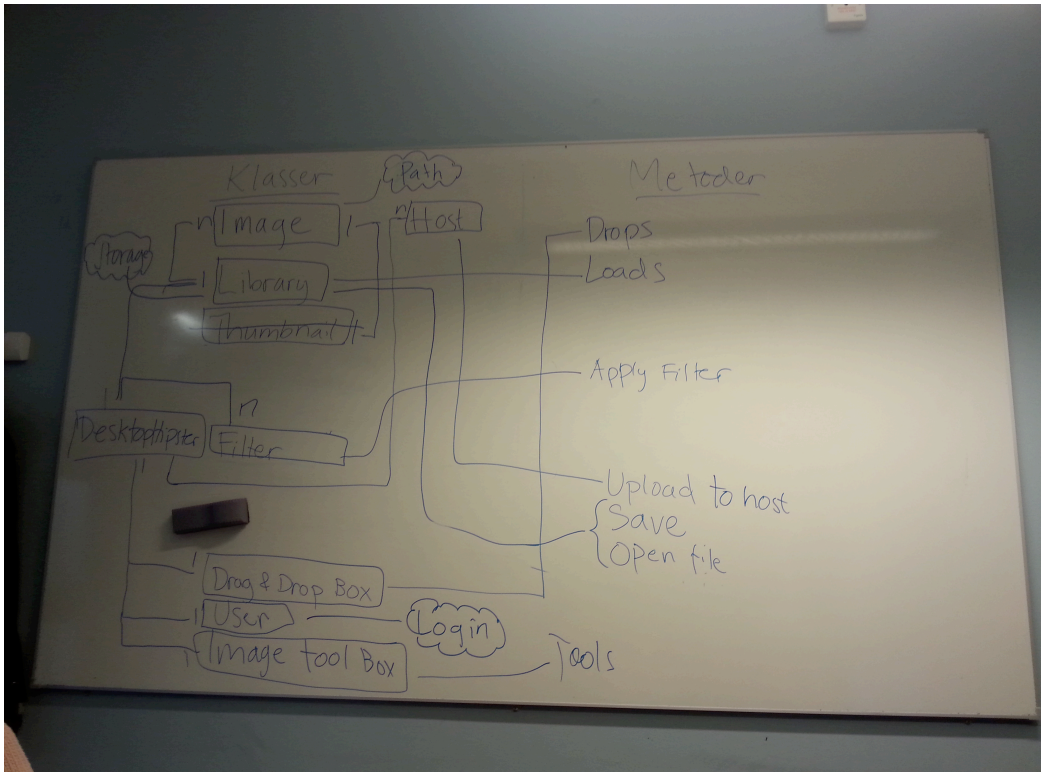

Figure 2: A sketch from the early stages

2.3.5. USER INTERFACE

The user interface should look similar like the picture below, a streamlined interface with a clear flow. Small amount of controls will lead to less clutter.

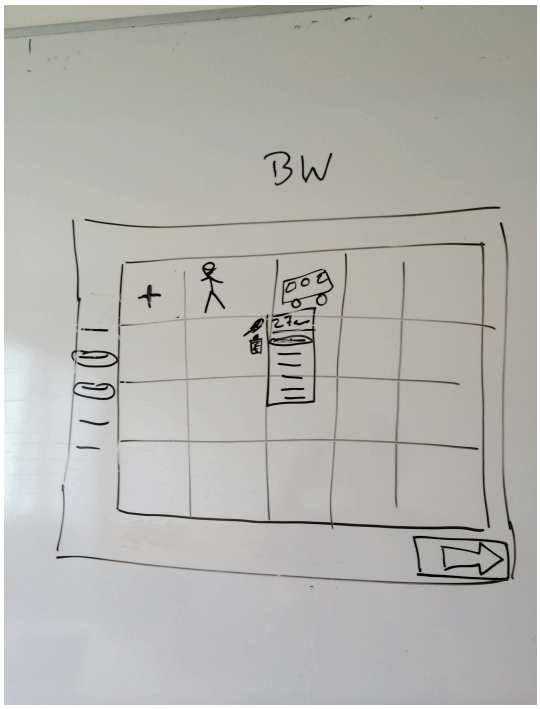

Figure 4: Sketch over the browse view

2.3.6. SEQUENCE DIAGRAMS

Under the implementation, sequence diagrams were made to visualize application MVC calls to identify problems in the software architecture.

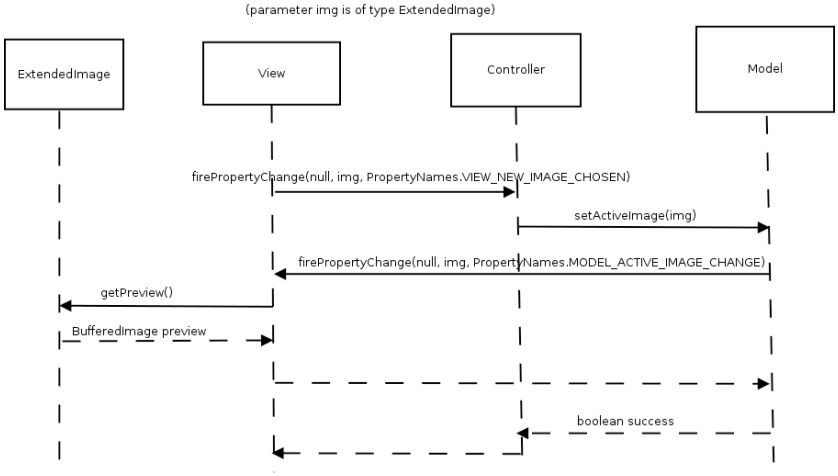

Figure 5: Sequence diagram for when a new image is selected in the UI

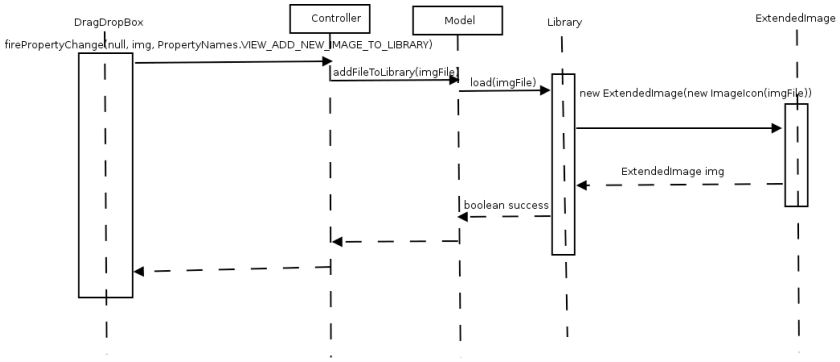

Figure 6: Sequence diagram for dropping an image on the UI

3. APPENDIX

Use Case: Desktop Hipster

Summary: Standard use cases for interacting with the application

Priority: mid

Participators: developers

Normal application flow

Step	Actor	System
1	Drops an image onto program	
2		Loads image to library for editing
3		Adds thumbnail to center stage
4	Chooses image to edit	
5		Toggles to edit stage
6	Chooses a filter from action bar	
7		Filter on the main image
8	Goes to upload stage	
9		Displays choice of image host
10	User selects host	
11		User is authenticated with host
12		Uploads to image host through API
13		Puts URL to upload in clipboard
14		Returns to start view

User saved image to custom directory

Step	Actor	System
8.1	Chooses to save image	
9.1		Displays choice of saving directory, defaults to standard library
10.1	User choses custom directory	

11.2		Displays file system view
12.2	User selects directory of choice	
13.2		Saves file to selected directory
14.2		Returns to start view

User imports file through file-system browser

Step	Actor	System
1.3	User chooses to open file from directory	
2.3		Displays system “open file”-dialog and adds chosen image to library for editing

User directly selects image

Step	Actor	System
1.4	User directly selects an image to edit	
2.4		Toggles to edit stage

Drop in taskbar pop-up

Step	Actor	System
1.5	Drops an image onto taskbar popup	

Chooses to upload an edited version directly

Step	Actor	System
4.6	Selects an earlier made version in the browse view	
5.6		Toggles to upload stage