Requirements and Analysis Document

Version: 2.0
Date: 2017-05-15
Authors: Anton Hagvall, Niclas Johansson, Hanna Jacobsson, Kristina Markan
This version overrides all previous versions.

1 Introduction

Are you in need of a new profile picture? Do you sometimes wish your photos would look just a little bit different? Most young people take photos on a daily basis. Making these photos look the way you want them to often takes some skill and experience and can easily become a time-consuming process. Layers is made for quick and simple editing. No time should be wasted on understanding how a complicated program works. Filters and text can be added, sharpen or blur. With all the basic tools close at hand you can give your photos the look you want.

1.2 Definitions, acronyms and abbreviations

- **GUI -** graphical user interface
- **Java** - a platform independent programming language.
- **Layer** - We save the filters/color/text transformation for easy control and access
- **Layer stack** - List containing all the layers applied on the image
- Filter
- Center stage
-

---

2 Requirements

2.1 User interface

The GUI of Layers is simple. The main view uses the design pattern *Canvas plus Palette* which consists of a large center stage accompanied by a palette at the bottom of the screen with options for editing the photo in question. The palette, or the toolbar, is a simple and easy way to find the function you want to use. When a category is chosen, the options in that category are shown. When an option is picked by the user, the specific settings for that option is shown in the toolbar instead of the different options.

The panel on the right side of the center stage helps the user get an overview of the photo as well as providing a simple way to zoom in and out. It also provides a list of all the layers and all the options related to their visibility and existence.

At the top of the screen all options can be reached through a menu. It can be used for getting an overview of the different options or as a quicker way of finding precisely what the user is looking for.

The interface does not consist of many different pages to navigate between. Instead everything can be reached from the main page. This is to provide a less confusing experience for the user and keep the focus on the photo that is being edited. Therefore, no other view than the main view is needed.

2.2 Functional requirements

What will the user be able to do ? Write a list of use case names (id's) in the language of the customer. The specific flows for each use case is recorded below. Specify a use cases in priority order.

General functionality:
Open photo - allows user to open a photo from computer
Open project - allows user to open a previous project
Save project - save project for future modifications
Export photo - export edited photo to .jpg
Close program - close the program
Undo - undo most recent change
Redo - redo change that has been undone
Reset photo - remove all layers

Crop, rotate and flip:
Crop - remove parts of the photo
Flip vertically - flip the photo around the vertical axis
Flip horizontally - flip the photo around the horizontal axis
Rotate 90° right - rotate photo 90° to the right
Rotate 90° left - rotate photo 90° to the left

Navigation tools:
Zoom in - view the photo bigger
Zoom out - view the photo smaller
The user drags the photo in chosen direction to navigate around it.

Filters:
Exposure - change of exposure
Contrast - change the contrast
Levels - make colors more muted
Noise - add noise to photo
Blur - make photo blurry
Gaussian blur - make photo blurry, but keep the edged less blurry
Sharpen - sharpen the photo
Edges - enhance the edges
Text - add text
Colorshift - increase a certain color in the photo
Black and white - make photo black and white

Grayscale - make photo grayscale
White balance - change the white balance
Add own filter - create own filter by filling a kernel with values

2.3 Non-functional requirements

Any special considerations besides functionality? Usability, reliability, performance, supportability, legal, implementation, … NOTE: Testability mandatory (must have tests)

Programmet ska vara på svenska och vara responsivt i renderingen av bilder.

The program is in Swedish and is responsive in the rendering of photos.

---

3 Use cases

3.1 General functionality

Use case: Open a photo
Priority: High

#	**Actor**	**System**
1	The user chooses “Öppna bild”	
2		File chooser appears
3	The user chooses a photo to open	
4		The photo appears on center stage

Use case: Open a project
Priority: High

#	**Actor**	**System**
1	The user chooses “Öppna projekt”	
2		File chooser appears
3	The user chooses a project to open	
4		The photo appears on the center stage and the filters used appear in the layer stack

Use case: Save project
Priority: High

#	Actor	System
1	The user chooses “Spara project”	
2		File chooser appears
3	User chooses a name and location for the project and pushes “Spara”	
4		The project is saved in the location

Use case: Export photo
Priority: High

#	Actor	System
1	The user chooses “Exportera bild”	
2		File chooser appears
3	The user chooses a name and location	
4		The photo is saved in the location

Use case: Close program
Priority: High

#	Actor	System
1	The user chooses to close program	
2.1		Pop-up asking user if they want to save project or close without saving
2.1.1	User chooses “Spara”	
2.1.2		See use case “Save program”
2.1.3		Program is closed
2.2.1	User chooses “Avsluta”	
2.2.2		Program is closed
2.3		Program is closed

Use case: Undo
Priority: Medium

#	Actor	System
1	User chooses “Ångra”	
2		Latest added layer is removed from layer stack

Use case: Redo
Priority: Medium

#	Actor	System
1	User chooses “Gör om”	
2		Removed layer reappears in layer stack

Use case: Reset photo
Priority: Medium

#	Actor	System
1	User chooses “Återställ bild”	
2		Photo is centered on center stage

Use case: Deleting the layer in the layer view
Priority: High

#	Actor	System
1	User presses the trash can on a layer	
2		System removes the layer from layer stack
3		System re-renders the image
4		System re-renders the layer view

3.2 Crop, rotate and flip

Use case: Crop
Priority: Medium

#	Actor	System
1	User chooses “Beskärning”	
2	User drags from one point to another on photo	

3		Black square is drawn on photo
4		Pop-up “do you only want to keep this part of the picture?”
5.1.1	User picks “Ja”	
5.1.2		Picture is cropped
5.2.1	User picks “Nej”	
5.2.2		Pop-up is closed

Use case: Flip vertically
Priority: Medium

#	**Actor**	**System**
1	User chooses “Spegla vertikalt”	
2		Picture is flipped vertically

Use case: Flip horizontally
Priority: Medium

#	**Actor**	**System**
1	Use chooses “Spegla horisontellt”	
2		Picture is flipped horizontally

Use cases: Rotate 90° right
Priority: Medium

#	**Actor**	**System**
1	User chooses “Rotera 90 grader åt höger”	
2		Picture is rotated 90° to the right

Use cases: Rotate 90° left
Priority: Medium

#	**Actor**	**System**
1	User chooses “Rotera 90 grader åt vänster”	
2		Picture is rotated 90° to the left

3.3 Navigation

Use case: Zoom in
Priority: High

#	Actor	System
1	User chooses “Zomma in”	
2		The photo is repainted in a scaled up version

Use case: Zoom out
Priority: High

#	Actor	System
1	User chooses “Zomma ut”	
2		The photo is repainted in a scaled down version

Use case: Moving the image on the canvas
Priority: High

#	Actor	System
1	User presses the mouse 1	
2		System stores the coordinates
3	User moves the mouse	
4	User releases the mouse 1	
5		System stores the coordinates and calculate the difference in x and y
6		System check if the image needs to be moved if it is jump to case
7.1		System re renders the image on moved coordinates
7.2		System re renders the image as it was

3.4 Filters

Use case: Blur

Priority: Medium

#	**Actor**	**System**
1	User chooses “Oskärpa”	
2		Image is blurred
3		“Oskärpa” is added to the layer stack

Use case: Gaussian blur
Priority: Medium

#	**Actor**	**System**
1	User chooses “Gaussisk oskärpa”	
2		Image is blurred but edges are kept sharp
3		“Gaussisk oskärpa” is added to the layer stack

Use case: Sharpen
Priority: Medium

#	**Actor**	**System**
1	User chooses “Skärpa”	
2		The image is sharpened
3		“Skärpa” is added to the layer stack

Use case: Edges
Priority: Medium

#	**Actor**	**System**
1	User chooses “Kanter”	
2		Only the edges in the photo is kept
3		“Kanter” is added to the layer stack

Use case: Text
Priority: Low

#	**Actor**	**System**
1		

Use case: Colorshift
Priority: Medium

#	**Actor**	**System**
1	User chooses “Färgfilter”	
2		Setting appear in toolbar
3		“Färgfilter” is added to layer stack
4		Color is applied to picture

Use case: Black and white
Priority: Medium

#	**Actor**	**System**
1	User chooses “Svartvit”	
2		Setting appear in toolbar
3		“Svartvit” is added to layer stack
4		The picture turns black and white

Use case: Grayscale
Priority: Medium

#	**Actor**	**System**
1	User chooses “Gråskala”	
2		Setting appear in toolbar
3		“Gråskala” is added to layer stack
4		The picture turns grayscale

Use case: White balance
Priority: Medium

#	**Actor**	**System**
1	User chooses “Vitbalans”	
		Setting appear in toolbar

		“Vitbalans” is added to layer stack
		The white balance is changed

Use case: Add own filter
Priority: Medium

#	**Actor**	**System**
1	User chooses “Lägg till eget filter…”	
2		New window is opened
3.1.1	User fills in name and kernel values	
3.1.2.1	User presses “Spara”	
3.1.2.2		Layer is saved
3.1.2.3		Layer is applied to picture
3.1.2.4		Window is closed
3.1.3.1	User presses “Verkställ”	
3.1.3.2		Layer is applied to picture
3.1.3.3		Window is closed
3.1.4.1	User presses “Avbryt”	
3.1.4.2		Window is closed

Use case: Exposure
Priority: Medium

#	**Actor**	**System**
1	User chooses “Exponering”	
2		The Exposure of the image changes
3		“Exponering” is added to the Layer stack

Use case: Contrast
Priority: Medium

#	**Actor**	**System**
1	User chooses “Kontrast”	

2		Setting appear in toolbar
3		The contrast of the image changes
4		“Kontrast” is added to the Layer stack

Use case: Levels
Priority: Medium

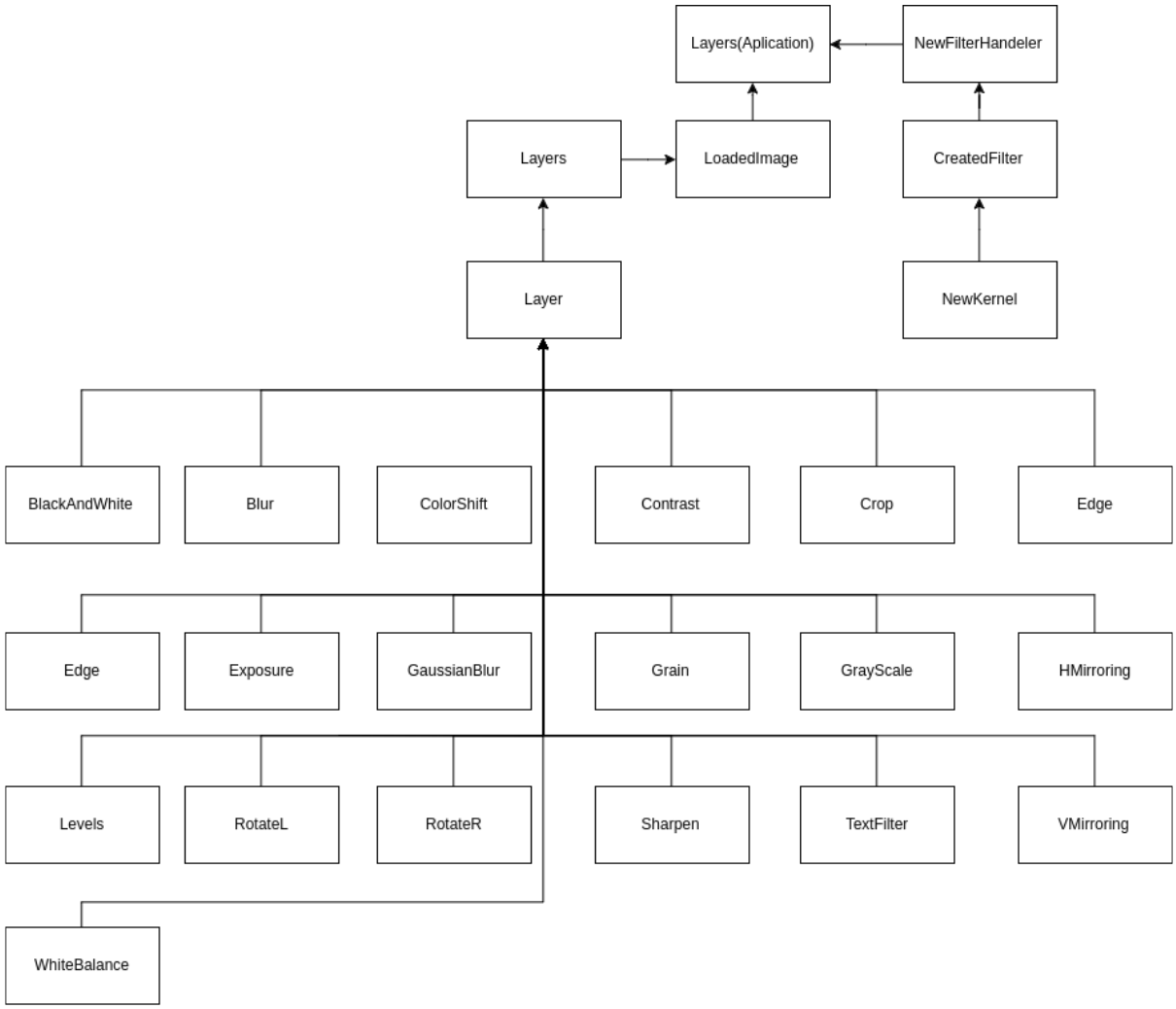
#	**Actor**	**System**
1	User chooses “Nivåer”	
2		Setting appear in toolbar
3		The levels of the image is changed
4		“Nivåer” is added to the layers tack

Use case: Noise
Priority: Medium

#	**Actor**	**System**
1	User chooses “Brus”	
2		Setting appear in toolbar
3		Noice is added to the image
4		“Brus” is added to the layer stack

4 Domain model

An UML class diagram.



4.1 Class responsibilities

Explanation of responsibilities of classes in diagram

- MainView
  The main controller, delegates tasks
- CanvasView
  Takes care of everything regarding the center stage of the program
- CropView
  All input regarding the crop function
- LayerRow
  Custom cell for LayerView

- LayerView
  List of the added layers
- MiniCanvasView
  Controlls the small version of the image on the right side of the screen
- NewFilterView
  Handles all input regarding creation of new filters by the user
- CreatedFilter
  Puts together the different components in a new layer
- Layer
  Keeps information regarding a layer
- Layerable
  Interface for all layers
- Layers
  Handles the layer stack
- LoadedImage
  Keeps the data of the loaded image
- NewFilterHandeler
  Keep a list of filters created by the user
- OpenProject
  Takes care of things related to opening previous saved projects
- SaveProject
  Takes care of things related to saving projects
- BlackAndWhite
  Filter takes makes each pixel either black or white depending on its value
- Blur
  Filter that adds a blur to the image
- ColorShift
  Filter that adds a transparent layer of color to the picture
- ColorShiftFactory
  Creates a Colorshift
- ColorShiftType
  ENUM with different Colorshift-options
- Contrast
  Filter that changes the contrast of the picture
- Crop
  All logic behind the crop function
- Edge
  Filter that brings out the edges in the picture while toning down the rest
- Exposure
  Filter that changes the exposure of the picture
- GaussianBlur
  Filter that adds a blur to the picture, while keeping the edges more intact
- Grain
  Filter that adds noise to the photo
- Grayscale
  Filter that makes the photo grayscale

- HMirroring
  Mirrors the image around the horizontal axis
- Levels
  Filter that mutes the color of the picture
- NewKernel
  Saving and storing new kernels
- RotateL
  Rotates the picture 90° to the left
- RotateR
  Rotates the picture 90° to the right
- Sharpen
  Filter that sharpens the picture
- TextFilter
  Adds a text to the picture
- VMirroring
  Mirrors the image around the vertical axis
- WhiteBalance
  Filter that changes the white balance of the picture