**Software Requirements Specification**

**for**

Photo Editor

**Version 1.0 approved**

**Prepared by Nikhil KR (PES2201800044)**

**Worked on Introduction (Section 1) and Analysis Models (Section 4)**

**PES University Electronic City**

**1st February 2021**

**Table of Contents**

**Table of Contents ii**

**Revision History i**[**ii**](#_heading=h.30j0zll)

**1. Introduction 1**

1.1 Purpose 1

1.2 Intended Audience 1

1.3 Product Scope 1

1.4 Document Convention 1

1.5 References 2

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 Assumptions and Dependencies 3

**3. External Interface Requirements 4**

3.1 User Interfaces 4

3.2 Software Interfaces 4

3.3 Communications Interfaces 4

3.4 Hardware Interfaces 4

**4. Analysis Models 5**

**5. System Features 6**

5.1 Load Image 6

5.2 Crop 7

5.3 Rotate 8

5.4 Color Correction 9

5.5 Contrast Correction 11

5.6 Text Annotation 13

5.7 Zoom In/Out 14

**6. Other Nonfunctional Requirements 16**

6.1 Performance Requirements 16

6.2 Safety Requirements 16

6.3 Security Requirements 16

6.4 Software Quality Attributes 16

6.5 Business Rules 16

**Appendix A: Glossary 17**

**Appendix B: Field Layouts 17**

**Appendix C: Requirement Traceability Matrix 17**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Hemant Sathish | 1/2/2021 | First Edit | 1.0 |
| Ruchira Vadiraj | 1/2/2021 | First Edit | 1.0 |
| Nikhil KR | 1/2/2021 | First Edit | 1.0 |

# 1. **Introduction**

## **Purpose**

The purpose of this document is to capture, in natural language and at a functional level, the description and requirements of a photo editor. This is a functional description of those features required for a basic photo editor. This includes features such as cropping, rotating, zooming in/out, adding text, color correction and differing contrast. The description for the requirements of the software can help improve the understanding of the system for a developer in a better fashion.

## **Intended Audience**

This document is intended for design and project developers who are looking to work on the Photo Editing Application. Understanding the document in a chronological order will be very beneficial to the reader. The document starts with a brief introduction of the product and dives into the functionalities and features moving forward.

## **Product Scope**

The system will be user-friendly and hence highly beneficial to the people working with a lot of pictures on a daily basis. They could be photographers, graphic designers, or even people who are highly active on social media. There are many problems that can be fixed or elements that can be enhanced with a photo editing software. The user will be able to modify the image in terms of size, angle and contrast. Functionalities such as adding text and color correction can also be done by the user.

## Document Conventions

**LI** Load the image

**CI** Crop Image

**GUI** Graphic User Interface

**CSCI** Computer Software Configuration Item

**RI** Rotate Image

**CRC** Color Correction

**CSC** Contrast Correction

**TA** Text Annotation

**PNG** Portable Network Graphics

**OS** Operating System

**ZIA** Zoom In/Out

## **References**

1. IEEE 830 Template
2. The Photo Editor Application - Chapter 5
3. IEEE Recommended Practice for Software Requirements Specifications IEEE Computer Society, 1998

# **Overall Description**

## **Product Perspective**

An image is an artifact that depicts visual perception, such as a photograph or a two-dimensional picture, that resembles a subject—usually a physical object—and thus provides a depiction of it. A photo editor takes in an image and performs user-specific functions on it to transform it into how the user perceives it to be viewed. The product mimics some parts of the Google-owned application, Snapseed. As the intended application has only a handful of developers, a small set of functions based on priority of the Snapseed application are implemented.

## **Product Functions**

There are essentially six functions(CSCI’s) that the user can perform after loading the application :

1. Crop Image (**CI**): The image, after being loaded into the application, can be cropped, which essentially corresponds to changing the aspect ratio (dynamic change), done using the height and width bars provided on selecting the “Crop” option.
2. Rotate Image(**RI**) : The image, after being loaded into the application, can be rotated, around a fixed axis. The rotating function can be used by moving the vertical bar which changes the orientation of the image in question, on selecting the “Rotate” option.
3. Color Correction (**CRC**): The image, after being loaded into the application, can be color-corrected, meaning the image can be passed through filters, which in turn changes the theme/color of light, reflection and shadow light emission seen by the user.
4. Contrast Correction (**CSC**): The image, after being loaded into the application, can be contrast-corrected, wherein the image brightness can be enhanced or decreased based on the movement of the vertical bar provided to the user. White balancing is an integral part of this function.
5. Text Annotation (**TA**): The image, after being loaded into the application, can be modified by having labels/text symbols being inserted/imprinted on it. A text box, which can be used after selecting the **TA** option, can be dragged through the frame of the image by the user and the text symbols can be written in after the user has reached a desired spot to place the text box.
6. Zoom in/out(**ZIA**): Zooming refers to enlarging or reducing into the pixels of an image. This can be done using the necessary buttons(zoom in, zoom out), and the necessary changes must be seen on the input image.

*The Use Case Diagram (****Figure 4.1****)**summarizes all the features provided by the application.*

## **User Classes and Characteristics**

No special knowledge or skills shall be assumed on the part of the users. Users shall not be expected to learn a set of commands in order to start using the application. Users shall not be expected to remember a list of commands while using the software – these shall be provided via menus, tool palettes. Users shall not be protected from data loss, and must download the image after the necessary edit operations are applied.

## **Operating Environment**

A Java application / Android application is currently in the works, which will work on any JDK-installed/Android platform. Any type of image input file, can be loaded into the application and the output file that can be downloaded is of only one type, PNG.

## **Design and Implementation Constraints**

There are currently two options that are considered for implementing the application :

1. Android Studio : [Meet Android Studio  |  Android Developers](https://developer.android.com/studio/intro)
2. Java GUI : [Java Platform Standard Edition 8 Documentation (oracle.com)](https://docs.oracle.com/javase/8/docs/)

There are no specific constraints on the developer otherwise.

## **2.6 Assumptions and Dependencies**

It is assumed that the requirements described in this document have the same levels of priority. Although no specific priorities have been documented, it is assumed that software developers may need to scrub priority requirements in the face of schedule and resource pressures. Requirements should only be scrubbed by agreement with all the developers, and only if it can be clearly demonstrated that there is no other high priority (non-scrubbed) requirement that depends upon the scrubbed requirements. No other special assumptions or dependencies have been identified.

# **External Interface Requirements**

## **User Interfaces**

The interaction with the user is through the GUI. The interaction is mainly through the use of button clicks and drags. The interface will have a blank slate before the user loads in the image and until the user decides to exit out of the application, the image shall remain in point of view, with the exception when the user removes/replaces the image that is currently being worked on. On downloading the image after editing, the blank slate returns for further images to be loaded and worked on, or until the user closes the application.

## **Software Interfaces**

The application uses the file system residing in the **OS,** for looking after the downloaded image/loading the image that is residing in the file system to the application. No guarantees are given by the software for security assurances with regards to saving the loaded image during/after a security breach.

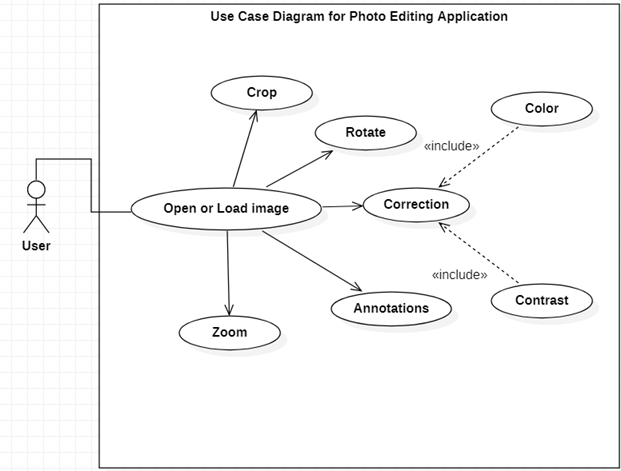
## **Communications Interfaces**

The application will not interact with any other application/over the web transfers and will solely reside on the edge where it has been installed on. No other software/application interaction is a guarantee.

## Hardware Interfaces

On the client side, a device that runs Android OS will be required. The device will contain the application installed on it for the usage.

# **Analysis Models**

**

***Figure 4.1.*** *Use Case Diagram*

# **System Features**

The specifications of the system features with their priorities and sequences are mentioned below:

## Load Image

Loading the image to the application.

5.1.1 Description and Priority

The first feature of the application is to get the required image from the user. This is done using button clicks to find the required file in the current system the application is running on and loading it on to the application. This is the most important feature of the application as all the other system features depend on this for their working, therefore this is a high priority feature.

Benefit (9) – Almost all of the features in the application depend on it.

Penalty (9)– Without the feature the application wouldn’t run, therefore this will incur the highest penalty.

Cost (5) – A relatively low cost is assigned as it doesn’t take much time to be implemented.

Risk (8) – There is a lot of risk in finishing the application without this feature.

5.1.2 Stimulus/Response Sequences

The user has only one button to activate this feature and on-click, a window pane with options to navigate through the file system to find the image to be worked on is seen, and the final conclusion of this sequence ends when the image is loaded into the application.

5.1.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for Android.

* get\_input(image\_input): The image input must be taken in from the user. The input may or may not necessarily be an image. The handling of such errors which keeps the application in a valid state must be given priority.

* error\_on\_invalid\_entry(invalid\_input): The application must respond to an invalid input after the window pane that is used to select the image is done with its job. The error can be a prompt or a message entry at one of the text boxes. The user has to understand that there is an error and must act on it to rectify it. Therefore, the message must be clear and loud and should take a substantial amount of space in the available frame.
* On\_success(image\_input): On a successful load of the image, necessary features open up to the user to act on editing the image. These features must not be available to the user before an image is selected. Any more movement in selecting the image must delay these features being shown on the frame.

## Crop

Cropping the input image.

5.2.1 Description and Priority

This feature is available to the user right after the load image feature gives a valid output. The “Crop” option will be available after a successful load of the image. Cropping is essentially a height and width change therefore the whole image is selected before any change is done by the user.

Benefit (5) – A standard value as it is pertinent in building a successful product.

Penalty (7)– Without the feature the application might feel that it is missing something, therefore a high penalty is given to it.

Cost (6) – A moderate cost is given to it as it is one of the core features of a photo editing application and can take time to be developed.

Risk (6) – There involves a slight risk as there are no other products on the market which have the same application and don’t have this feature.

5.2.2 Stimulus/Response Sequences

Image crop allows users to extract a rectangular part of the image. When the Crop button is pressed, a dialog window opens and the user is asked to type in the width and height of the cropped image. The image is cropped to the size specified by the user if the size specified is smaller than the actual image; otherwise, it is ignored. The cropping is done in a way that the midpoint of the original image is the midpoint of the cropping rectangle defined by the user. The area outside the cropping rectangle is deleted, and the new image in the new size is shown. If the defined cropping rectangle is larger than the image, no cropping is done.

5.2.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for android.

* crop\_image(height,width): The application must respond to the crop image option being picked and must concur with the necessary features mentioned in section 4.1.2.
* out\_of\_bounds(height,width): The height and the width inputs are limited to a specific size(TBD) and any out of bounds entry must be dealt with a prompt of the error message stating the same.
* on\_complete(final\_image): On a successful attempt of the required operation to be done on the image, the user now can select any other feature to make the necessary changes.

NOTE : All the other features(buttons) must be hidden when a feature is already selected. This ensures that the image is always in a valid state on jumping from one feature to another.

## 5.3 Rotate

Color correction with color grading on applied to an image.

5.3.1 Description and Priority

This feature is available to the user right after the load image feature gives a valid output. The rotate image option is given by two buttons (one for left, one for right). On selecting the “Rotate” option, the two buttons appear on the UI and the user can use them to flip the input images. (360 degree rotation).

Benefit (7) – An important feature therefore a high benefit value is specified.

Penalty (8)– A high penalty based on the benefit.

Cost (6) – A moderate cost as not a lot of time needs to be spent to implement this feature.

Risk (6) – There is a moderate amount of risk in implementing this feature.

5.3.2 Stimulus/Response Sequences

Rotation is an important aspect of augmentation and generally almost all the editors implement this feature. The user upon selecting the “Rotate” option, has two choices, either rotate left, or rotate right. These choices are made using button clicks and the necessary change is seen in the input image. An “exit” option is used to go back to the main list of features after using this feature.

5.3.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for android.

* rotate(button\_click): On selecting the rotate option, the user is provided with two buttons, which indicate their features through their labels.
* rotate\_left(augment\_image): Rotate the image 90 degrees to the left. Necessary changes must be seen in the window pane containing the input image.
* rotate\_right(augment\_image): Rotate the image 90 degrees to the right. Necessary changes must be seen in the window pane containing the input image.
* on\_exit(out\_of\_rotate): On clicking this button, the input image holds the changes made and exits out of **RI**. All the other features will now be available.

NOTE : All the other features(buttons) must be hidden when a feature is already selected. This ensures that the image is always in a valid state on jumping from one feature to another.

## 5.4 Color Correction

Color correction with color grading on applied to an image.

5.4.1 Description and Priority

This feature is available to the user right after the load image feature gives a valid output. The color correction option is generally used to enhance tone and build up contour. Currently, a three way color corrector is being considered to be an option. The drag option lets the user pick up the color balance.

Benefit (3) – A low benefit value is given to it, as it does not specially entail in a photo editor.

Penalty (3)– A moderate penalty based on the benefit.

Cost (9) – This has a high cost, as implementation will take a huge chuck of development time.

Risk (9) – There is a high risk in taking up this feature, as it may lead unnecessary development time being wasted on it.

5.4.2 Stimulus/Response Sequences

Color balancing and color correction are considered to extra features when a photo editing application is involved. Although it would help in uplifting the market value of the product, it takes a lot of time and has a low success rate. The user will have to select the correction button and then would be given two options, namely, “Color” and “Contrast”. On picking the “Color” option, a window pane with the three way corrector window would appear and the user would make the necessary color adjustments. After the said adjustments are done, the “Apply” button is selected and the image would reflect the necessary changes that were made.

5.4.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for android.

* correction(button\_click): On selecting this option, two different options can be seen by the user, “Color” and “Correction”.

* color\_correction(coordinates): The color correction feature will have built in features and any necessary errors are handled by the API calls made. The three way color corrector is seen on selecting the color correction button.

* on\_apply(changed\_image): After the apply changes button is selected, the necessary changes are seen on the input image and all the set of features are once again available to the user. Note that after the changes are applied, the application would go back to a valid state and all of the options, correction -> color, must be selected again, for making more changes.

NOTE : All the other features(buttons) must be hidden when a feature is already selected. This ensures that the image is always in a valid state on jumping from one feature to another.

### 5.5 Contrast Correction

Contrast correction with white balancing on applied to an image

.

5.5.1 Description and Priority

This feature is available to the user right after the load image feature gives a valid output. The contrast correction option is generally used to white balance the input image. Currently, a two way color corrector (shades of white to black) is being considered to be an option. The drag option lets the user pick up the white balance.

Benefit (3) – A low benefit value is given to it, as it does not specially entail in a photo editor.

Penalty (3)– A moderate penalty based on the benefit.

Cost (7) – This has a high cost, as implementation will take a huge chuck of development time.

Risk (6) – There is an ample amount of risk in taking up this feature, as it may lead to a moderate amount of development time being wasted on it.

5.5.2 Stimulus/Response Sequences

White balancing is considered to be an important feature in the working of a photo editing application. The user will have to select the correction button and then would be given two options, namely, “Color” and “Contrast”. On picking the “Contrast” option, a window pane with the two way corrector window would appear and the user would make the necessary white balance adjustments. After the said adjustments are done, the “Apply” button is selected and the image would reflect the necessary changes that were made.

5.5.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for android.

* correction(button\_click): On selecting this option, two different options can be seen by the user, “Color” and “Correction”.

* contrast\_correction(coordinates): The contrast correction feature will have built in features and any necessary errors are handled by the API calls made. The two way color corrector is seen on selecting the contrast correction button.

* on\_apply(changed\_image): After the apply changes button is selected, the necessary changes are seen on the input image and all the set of features are once again available to the user. Note that after the changes are applied, the application would go back to a valid state and all of the options, correction -> contrast, must be selected again, for making more changes.

NOTE : All the other features(buttons) must be hidden when a feature is already selected. This ensures that the image is always in a valid state on jumping from one feature to another.

.

### 5.6 Text Annotation

Writing text onto the input image.

5.6.1 Description and Priority

This feature is available to the user right after the load image feature gives a valid output. A text box is provided to the user on selecting the “Annotation” option, which has a drag and drop feature. After placing the box at the desired place, the user can edit the text in the text box by clicking on it and after which the user can anywhere else on the window pane other than the text box, to make the necessary changes stick.

Benefit (7) – A healthy benefit value is given as most editors in the market implement this feature.

Penalty (5)– A moderate amount of penalty is incurred on failing to finish the feature.

Cost (4) – Text Annotation is a low cost, high priority feature.

Risk (5) – A moderate amount of risk is involved in implementation of this feature.

5.6.2 Stimulus/Response Sequences

Text Annotations are integral in any modern photo editor application. The user will have to select the “Annotate” button and upon selecting this button a text box appears, which moves along the window pane with the cursor. The edit option is available as soon as the user clicks on the window pane. The next set of actions are initiated wherein the user can edit the text in the text box and the necessary changes are seen as soon as the user clicks anywhere on the window pane other than the text box after the text is typed in.

5.6.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for android.

* annotate(text\_box): On selecting this feature, a text box appears and the user can start moving around the box and place it anywhere on the image(within the window pane). The user is restricted with the movement of the box within the pane.

* on\_edit(change\_text): After a button click is received from placing the box at the desired place, the user can now edit the text in the text box.

* on\_click(window\_pane): The final click will ensure that the text is placed on the image at the specified place. This moves the application into a valid state.

NOTE : All the other features(buttons) must be hidden when a feature is already selected. This ensures that the image is always in a valid state on jumping from one feature to another.

### 5.7 Zoom In/Out

Writing text onto the input image.

5.7.1 Description and Priority

This feature is available to the user right after the load image feature gives a valid output. The zoom feature helps in enlarging or reducing the size of the input image. This is done with the help of two buttons (**ZIA**), which are available for selection after the “Zoom” button is selected. Necessary labels indicate the functions of the two buttons.

Benefit (7) – A healthy benefit value is given as most editors in the market implement this feature.

Penalty (6)– A high penalty is incurred on failing to finish the feature.

Cost (3) – Zoom In/Out is a low cost, high priority feature.

Risk (5) – A moderate amount of risk is involved in implementation of this feature.

5.7.2 Stimulus/Response Sequences

The user will have to select the “Zoom” button and upon selecting this button, two new buttons appear on the UI. The two buttons are used for enlarging and reducing the input image size. A set limit is applied to both enlarge and reduce, where exceeding this limit, won’t change anything on the window pane. Note that there will be no prompt when the limit is exceeded. Note that to exit out of **ZIA**, there is an exit button which can be seen probably on the top right of the window pane.

5.7.3 Functional Requirements

The software capabilities that are needed for this feature to work are: the underlying OS must be Windows/Mac or Android, a file system support, user interactive features such as GUI, a pointer that is useful for button clicks/touch input for android.

* zoom(input\_image): A button click will result in two new buttons appearing on the UI, which must be self-explanatory (based on the button label).

* on\_click\_enlarge(input\_image\_size): The image size must increase with this button click. A set limit is specified and on exceeding this limit the button won’t gauge any input from the user.

* on\_click\_reduce(input\_image\_size): The image size must decrease with this button click. A set limit is specified and on exceeding this limit the button won’t gauge any input from the user.

* on\_exit(out\_of\_zoom): On clicking this button, the input image holds the changes made and exits out of **ZIA**. All the other features will now be available.

NOTE : All the other features(buttons) must be hidden when a feature is already selected. This ensures that the image is always in a valid state on jumping from one feature to another.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

An approximation can be made that the output image will be available in 30 seconds or under. Functionalities such as crop and rotate will require a lesser time period for processing as compared to the other functions. The system will be able to handle all images uploaded in any of the standard image formats (.jpeg, .png, .eps etc.).

## **Safety Requirements**

No safety requirements have been identified.

## **Security Requirements**

There are no security concerns with respect to the usage of the system as there will not be any record of the user or the image uploaded on the system side. The client will have an option to download the filtered image.

## **Software Quality Attributes**

The final product will be an Android application and hence will hold all the generic features of an application.

* Reliability : The mentioned features need to be satisfied in the product so as to be of the utmost usage to any user.
* Usability : Shallow learning curve as it only involves dragging, dropping and selecting options from the given menu.
* Portability : Since it is an application, it can be installed on any mobile device anywhere.
* Maintainability : Application will not require any high maintenance.
* Testability : Minimum testing of all the functionalities is necessary.
* As mentioned, the application will be restricted to Android OS devices only.

## **Business Rules**

The application will be open to all for usage. There are no limits on the types of users and since there are no security loopholes in the application, anyone can use it for any purpose at any point in time. Premium services for a set of functionalities can be launched after a period of time.

Knowledge in the following domains are necessary :

* Application Development (Android application in specific)
* Photography
* Minimum color sense

**Appendix A: Glossary**

* GUI : Graphical User Interface
* OS : Operating System
* TBD : To Be Decided
* Android Studio : Unified environment to build applications for Android devices

**Appendix B: Field Layouts**

No information yet.

**Appendix C: Requirement Traceability Matrix**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sl. No** | **Requirement ID** | **Brief Description of Requirement** | **Architecture Reference** | **Design Reference** | **Code File Reference** | **Test Case ID** | **System Test Case ID** |
| 1 | LI | Load the input image into the application. |  |  |  |  |  |
| 2 | CI | Crop the given input image. |  |  |  |  |  |
| 3 | ZIA | Zoom In/Out, change the input image size. |  |  |  |  |  |
| 4 | RI | Rotate the given image in the specified direction. |  |  |  |  |  |
| 5 | CRC | Color grading of the given input image. |  |  |  |  |  |
| 6 | CSC | White Balancing of the given input image. |  |  |  |  |  |
| 7 | TA | Text Annotation on the given input image. |  |  |  |  |  |