Bilkent University

Department of Computer Engineering

Senior Project Design

Autoshop

Analysis Report

Efe Acer
Hikmet Demir
Mehmet Mert Duman
Talha Murathan Göktaş
Burak Yaşar

Supervisor	: Fazlı Can
Jurt Members	: Cevdet Aykanat, Ercüment Çiçek
Innovation Expert	: Duygu Gözde Tümer
Website	: https://beastunited.github.io/

Analysis Report
Nov 11, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction

With social media becoming prominent as a byproduct of the technology era, we began to see a very intense online photo traffic. To be more specific, over 300 million photos are uploaded to Facebook daily and there are around 40 billion photos shared on Instagram since its creation [1]. The direct implication of these numbers is that people love to take and share photos. They spend hours trying to capture effectual moments or just to look good. In an effort to make it easier for people to be happy with their photos; a new industry, "Photoshop" had emerged.

Photoshop itself, is a demanding task. People train themselves to become good photoshoppers and spend heaps of time in front of the screen to make realistic photo edits. The departure point of our project, "Autoshop", is to make this arduous task accessible to everyone, even the people knowing the absolute minimum of Photoshop.

The nature of innovation behind Autoshop is to create a new and powerful photo editing tool using state of the art Computer Science techniques. Autoshop will help people add objects to and remove objects from their photos without requiring any photoshop knowledge. Autoshop makes advanced technologies accessible in multiple platforms, aiming to dilate its user profile.

Imagine yourself missing a friend gathering, you would probably say "I wish I was there.". Autoshop helps you exactly at these moments. Using it, you will be able to effortlessly add yourself into the moment. You may argue that this is possible using tools such as Adobe's Photoshop, however, you would have to work quite a bit to do that. Autoshop, as its name suggests, automizes the process and makes your computer or your mobile phone your personal photoshopper.

Tons of use cases can be found regarding photo editing. For instance, one may want to purchase some furniture for her living room, but she may be indecisive about how well the furniture will fit into the room. With the help of Autoshop; she can take the photo of the piece of furniture and the living room, combine these as if they are actually in the same place, and then decide. Also, people can add their faces on top of any photo and any person's face, for example, their favorite rock band Queen's poster or the famous movie Marvel Avengers' cover photo.

In this report, we intend to provide an overall analysis of the system. First, existing systems together with their scopes, differences and missing features will be discussed. Later, a brief description of the proposed system, Autoshop, will be given by emphasizing its unique features. Then functional, non-functional and pseudo requirements will be listed. Afterwards, system models of our system will be examined. Scenarios will be derived from generalized use-cases whose diagrams will be provided. These scenarios will be discussed in detail taking the possible user choices and conditions into account. Object and class model, and dynamic models will also be provided and explained. The report will be continued with screen mock-ups and navigational paths. Lastly, we will conclude the report with a discussion on the social dimensions of the project.

2 Current System

Although NVIDIA open-sourced sophisticated deep learning models to remove objects from and add objects to images, there is no comprehensive commercial product that makes use of these technologies. There are also web-applications that offer the removal of unwanted objects from photos [2], however, these applications corrupt the inherent style of the photos while removing the objects from them. Additionally, there are industry-standard tools such as Adobe's Photoshop [3], albeit, these tools require advanced photoshop knowledge to perform addition and removal of objects. On top of that, Adobe products are relatively expensive [4] and not fully-functional on mobile platforms; which implies that these products have serious accessibility issues.

3 Proposed System

3.1 Overview

Autoshop is a photo-editing tool that makes use of cutting edge deep learning techniques to add objects to and remove objects from images. The tool is innovative in the sense that there is no commercial product that can perform context-aware addition and removal of objects. The task of context-aware object addition and removal is challenging since the texture, opacity, brightness and other photographical details of the added object should match those of the background image. Currently, this process is handled by professional photoshoppers and it is quite time-consuming. Hence, Autoshop will allow people to edit their photos effortlessly.

Advanced models and techniques such as neural networks, smart-cropping, padding, and image inpainting will constitute the backbone of Autoshop. The software will mostly revolve around the "sub" Computer Science topics; Deep Learning and Image Processing. Autoshop will demonstrate that such advance technologies can indeed be useful as solutions to practical problems.

Autoshop aims to have an extensive user profile. Thus, the software will have two versions; desktop and mobile. The desktop will enable professionals to use Autoshop in unison with the other photoshopping software they have. Besides that, the rationale behind the idea was to make photoshopping easy and accessible to everyone, so a mobile version is a must for us.

3.2 Functional Requirements

In this section, functional requirements will be discussed. Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform. [5]. As part of functional requirements, we will discuss data resources requirements and user-specific requirements.

3.2.1 System Functionality

The system should:

- ask the user's permission to access the local machine's resources (photo gallery, camera, etc.).
- display a gallery of user photos from which the user can make selections.
- provide an option to receive camera input.
- receive a background image from the user as an input, the background image refers to the image to be edited.
- receive additional images, which are the image(s) to add on the background image, as a user input.
- get the position of the additional images over the background image by drag and drop.
- get the size of the additional images by resizing.
- get the borders of the additional images by cropping and scanning; if the user does not crop or scan, then use smart-cropping methods to extract the object of interest.
- get the borders of the background image by cropping.
- get the scaling factor of the background image by rescaling.
- display the images to edit, in other words, the background image and the additional images on top of it, in a frame.
- communicate with a server, which runs a pre-trained neural network. In the desktop application, the server will be the localhost itself since it can run GPU-heavy tasks. However, in the mobile application, the server will be a remote machine that is capable of running the neural network efficiently.
- send the background images, the additional images; the positions, sizes, and borders of those images to the server.

- use the pre-trained neural network to adjust the photographic properties (opacity, texture, brightness, color, etc.) of the additional images according to the background image, and generate a photoshopped output.
- receive the photoshopped output from the server.
- display the photoshopped image in another frame. This will be done in real-time, as the system receives input, in the desktop application since the server is the local machine itself; but the mobile application will require some time to establish server communication and perform data transfers.
- provide an option that allows the user to save the photoshopped output to the local machine's photo gallery.
- provide an option that allows the user to share the photoshopped output using the various social media and messaging platforms (Instagram, Whatsapp, E-mail, etc.) in the local machine.
- ask the user's permission to use the photoshopped output for further training of the neural network.
- delete the photoshopped output and the inputs from the server-side, if the user refuses the use of his/her data.
- store the photoshopped output and the inputs in the server-side, if the user grants permission for the system to use his/her data.
- periodically download "photo - photoshopped photo" data from certain sources; and use this data, together with the stored user data to retrain the neural network. Period is to be determined.

3.2.2 User Functionality

The user should/can:

- allow or deny permission for the system to use the local machine's resources.
- select or take a background photo and upload it to the system.
- select or take additional images and upload them to the system.
- drag and drop the additional images over the background image to properly position them.
- rotate the additional images and the background image.
- resize the additional images.
- rescale (zoom in or out) the background image.
- crop or scan the additional images.
- crop the background image.
- see the original image and the photoshopped output. The user can also watch the process of neural network photoshopping the images in action in the desktop application.

- save the photoshopped output to his/her photo gallery.
- share the photoshopped output directly in his/her preferred social media or messaging platform.
- allow or deny permission for the system to use his/her data (original and photoshopped image) for system improvement purposes.

3.3 Nonfunctional Requirements

In this section, non – functional requirements will be discussed. There are many definitions of non – functional requirements. "A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system." [5]. Because of this, we will spend a lot effort to make them as good as possible. Below you can find information about Autoshop's non-functional requirements such as extensibility, reliability, usability, accessibility, portability, and efficiency.

3.3.1 Extensibility

The system should:

- be easy to maintain, in other words, open to updates.
- be available on multiple platforms (desktop, mobile, etc.).

3.3.2 Reliability

The system should:

- ensure that the user data (the additional images and the background image) is not stored in the server-side unless the user gives permission.
- roll-back in case of a failure in the server communication. To be clearer, delete any data that is not completely processed.
- be resistant to adversarial cyber-attacks on the server-side. Methods such as hashing, signing and encrypting can be used to ensure data confidentiality and integrity.
- generate a realistic, hence reliable, output. For this, NVIDIA's libraries such as FastPhotoStyle and Image Inpainting will be integrated into the system [6].

3.3.3 Usability

The system should:

- be self-explanatory and user-friendly.

- be clear in terms of display and language when prompting.
- present a neat and well-organized user interface with themes that the user can select.
- require the absolute minimum photoshop knowledge from users.

3.3.4 Accessibility

The system should:

- be downloadable for free.
- be downloadable from the Autoshop official website for the Desktop version.
- be downloadable from the GooglePlay Store for the mobile version.

3.3.5 Portability

The system should:

- run in hardware and OS independent manner. Here, OS independent refers to the flexibility to run on any OS that has a Java Virtual Machine (JVM) and can run Python scripts.
- be accessible from multiple platforms (desktop and mobile).

3.3.6 Efficiency

The system should:

- not lag when communicating with the server. In case of a lag, there should be a time-out event, possibly 5 seconds (subject to change), that leads to server roll-back.
- not delay much in the mobile version when receiving touch-input from the user. A long delay is conventionally considered as 2-3 frames.
- not delay much in the desktop version when receiving keyboard and mouse input. A long delay is usually considered as 100 milliseconds.
- not wait longer than a second (subject to change) for the neural network to generate the photoshopped output.

3.4 Pseudo Requirements

3.4.1 Issue Tracking

- Git and Github will be used for version control.
- The project board of Github will be used for issue tracking. The Agile strategy will be adopted in the development process. Short intervals of development; in

other words, Sprints will be executed to ensure progressive improvement. Developers in the team will be equally assigned tasks and issues through Github's project board.

3.4.2 Testing

- Travis CI will be used as a continuous integration service used to build and test Autoshop directly from its Github repository.[7]
- CodeClimate will be used as an automated code review service to ensure that the style conventions determined by the team are followed.[8]
- A certain threshold of test-coverage will be enforced to merge any code with the Master branch using Travis CI. This will make sure that the Master branch is always a functional version of Autoshop.

3.4.3 External Tools and Technologies

- Android Studio will be used for the development of the mobile application [9].
- NVIDIA's technologies for GPU accelerated Python programming will be potentially used, depending on how GPU-dependent the performance of the neural network will be.
- Grafana[10] will be used as an observability platform to monitor and analyze database analytics.

3.5 System Models

To start off the system model, the report will be explaining the scenarios that a potential user may go through. In this report twenty-six scenarios that a user may encounter are included. To explain each scenario in detail, they are divided into subsections as: Use Case Name, Actors, Entry Conditions, Exit Conditions and Main Flow of Events. The system model will be summarized by a use-case diagram. The use-case diagram will be used to describe the events and actions between the user and the system in a visual manner. Furthermore, each use-case will be explained in detail for further insight.

3.5.1 Scenarios

In this section, we will discuss how our app will function during different user-program interactions. Importance of this section is demonstrating the story-line of the application, as well as finding possible alternative solutions to usability problems we are trying to solve.

Scenario 1

Use Case Name	`Select a Background Image from Gallery`
Participating Actors	`Steve`
Entry Conditions	• The application must be open.
Exit Conditions	• User either clicks on the *Select* button or the *Cancel* button.
Main Flow of Events	User: 1. clicks on the *Add Background Image* button. 2. chooses a picture to add from the gallery. 3. clicks on the *Select* button or the *Cancel* button.

Scenario 2

Use Case Name	`Use the Camera to Take a Background Image`
Participating Actors	`Steve`
Entry Conditions	• The application must be open.
Exit Conditions	• User either clicks on the *Accept* button or the *Cancel* button.
Main Flow of Events	User: 1. clicks on the *Camera* button. 2. takes a picture with the phone's camera. 3. either clicks on the *Accept* button or the *Cancel* button to retake the photo.

Scenario 3

Use Case Name	`Select Additional Images`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image.
Exit Conditions	• User selects a photo or taps on an asset.
Main Flow of Events	User: 1. clicks on the *Add* button. 2. either selects a photo from the gallery or taps on an asset from assets. 3. crops or scans the additional image. 4. clicks on the *Select* button or the *Cancel* button.

Scenario 4

Use Case Name	`Crop the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User is selecting a background image. • User has clicked on an image from the gallery.
Exit Conditions	• User either clicks on the *Select* button or the *Cancel* button.
Main Flow of Events	User: 1. drags the corners of the image to crop it. 2. clicks on the *Select* button or the *Cancel* button.

Scenario 5

Use Case Name	`Crop and Scan the Additional Images`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User is selecting additional images. • User is selecting from the gallery instead of from the assets.
Exit Conditions	• User either clicks on the *Select* button or the *Cancel* button.
Main Flow of Events	User: 1. clicks on the *Add* button. 2. chooses a picture to add from the gallery. 3. crops the picture by dragging its corners. 4. scans the image by drawing with gestures. 4. clicks on the *Select* button or the *Cancel* button.

Scenario 6

Use Case Name	`Use Smart-Cropping to Crop Additional Images`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User is selecting additional images. • User is selecting from the gallery instead of from the assets.
Exit Conditions	• User either clicks on the *Select* button or the *Cancel* button.
Main Flow of Events	User: 1. clicks on the *Add* button. 2. chooses a picture to add from the gallery. 3. clicks on the *Smart Crop* button. 4. clicks on the object to crop out of the picture. 4. clicks on the *Select* button or the *Cancel* button.

Scenario 7

Use Case Name	`Scale the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image.
Exit Conditions	• User stops doing gestures.
Main Flow of Events	User: 1. gestures on the background image with two fingers to move the image in all directions and to rescale.

Scenario 8

Use Case Name	`Scale and Resize the Additional Images`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already added additional images.
Exit Conditions	• User stops doing gestures.
Main Flow of Events	User: 1. gestures on the additional image with two fingers to rescale and resize.

Scenario 9

Use Case Name	`Rotate the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User is gesturing over the background image.
Exit Conditions	• User stops doing gestures.
Main Flow of Events	User: 1. gestures on the additional image with two fingers to rotate.

Scenario 10

Use Case Name	`Rotate the Additional Images`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already added additional images. • User is gesturing over the additional images.
Exit Conditions	• User stops doing gestures.
Main Flow of Events	User: 1. gestures on the additional image with two fingers to rotate.

Scenario 11

Use Case Name	`Reposition the Additional Images over the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already added additional images.
Exit Conditions	• User stops moving the additional image.
Main Flow of Events	User: 1. gestures on the additional image with one finger to reposition it.

Scenario 12

Use Case Name	`Remove an Additional Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already added additional images. • User is dragging an additional image.
Exit Conditions	• User stops dragging the additional image or drags the image on the trash can icon at the bottom of the screen.
Main Flow of Events	User: 1. gestures on the additional image with one finger. 2. drags the image on top of the trash can icon and releases it.

Scenario 13

Use Case Name	`Scan the Background Image for Object Removal`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image.
Exit Conditions	• User clicks on the *Save* button or the *Cancel* button.
Main Flow of Events	User: 1. clicks on the *Remove* button. 2. gestures over the background image to mark the regions to be deleted. 3. clicks on the *Save* button or the *Cancel* button.

Scenario 14

Use Case Name	`Change Brush Size for the Scan Operation`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User is currently scanning the background image.
Exit Conditions	• User chooses a brush size.
Main Flow of Events	User: 1. changes the slider to adjust the brush size.

Scenario 15

Use Case Name	`Deselect the Area Scanned on the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already scanned the background image for object removal.
Exit Conditions	• User clicks on the *Undo* button.
Main Flow of Events	User: 1. clicks on the *Undo* button to deselect the scanned areas.

Scenario 16

Use Case Name	`Undo the Additions Made to the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already added additional images. • User has already performed the photoshop operation.
Exit Conditions	• User clicks on the *Undo* button.
Main Flow of Events	User: 1. clicks on the *Autoshop* button. 2. receives the photoshopped image. 3. clicks on the *Undo* button.

Scenario 17

Use Case Name	`Photoshop the Finalized Input (The Additional Images and Removed Parts) on the Background Image`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image. • User has already added additional images or scanned some parts to be removed.
Exit Conditions	• User clicks on the *Autoshop* button.
Main Flow of Events	User: 1. finalizes the modifications (crop, scale, resize, rotate) of the additional images. 2. finalizes scanning the background image to remove objects. 3. clicks on the *Autoshop* button.

Scenario 18

Use Case Name	`Reset the Background Image to its Initial Version`
Participating Actors	`Steve`
Entry Conditions	• User has already added a background image.
Exit Conditions	• User clicks on the *Reset* button.
Main Flow of Events	User: 1. clicks on the *Reset* button.

Scenario 19

Use Case Name	`Export the Final Image`
Participating Actors	`Steve`
Entry Conditions	• User has already performed the photoshop operation.
Exit Conditions	• User either exports the image by clicking on the *Ok* button or cancels by clicking on the *Cancel* button.
Main Flow of Events	User: 1. clicks on the *Export* button. 2. chooses where to export (Instagram, Twitter, WhatsApp, Gallery etc.). 3. clicks on the *Ok* button to export the image or *Cancel* button to cancel the operation.

Scenario 20

Use Case Name	`Change Settings`
Participating Actors	`Steve`
Entry Conditions	• The application must be open.
Exit Conditions	• User either select one of settings option or go back to main page with back button
Main Flow of Events	User: 1. clicks to settings button 2. change settings 3. clicks on the Ok button to apply changes or Cancel button to cancel changes

Scenario 21

Use Case Name	`Share the Application`
Participating Actors	`Steve`
Entry Conditions	• User must be in settings page
Exit Conditions	• User share the application over social media or cancel the sharing with back button
Main Flow of Events	User: 1. User clicks to the share the application button 2. User share the application over social media or cancel sharing with back button

Scenario 22

Use Case Name	`Read Frequently Asked Questions`
Participating Actors	`Steve`
Entry Conditions	• User must be in settings page
Exit Conditions	• User go back to settings page with back button
Main Flow of Events	User: 1. User clicks to FAQ button 2. User read questions and exit the page with back button

Scenario 23

Use Case Name	`Tutorials`
Participating Actors	`Steve`
Entry Conditions	• Enter the application for the first time or click the ? button
Exit Conditions	• User click cancel button to exit tutorial
Main Flow of Events	User: 1. User clicks to ? button or user start using application for the first time 2. User complete the tutorial and click exit button to finish tutorial

3.5.2 Use Case Model

In the following section, we will introduce the use-case models of our system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal [11]. This is important for showing our user-system interactions in a compact manner.

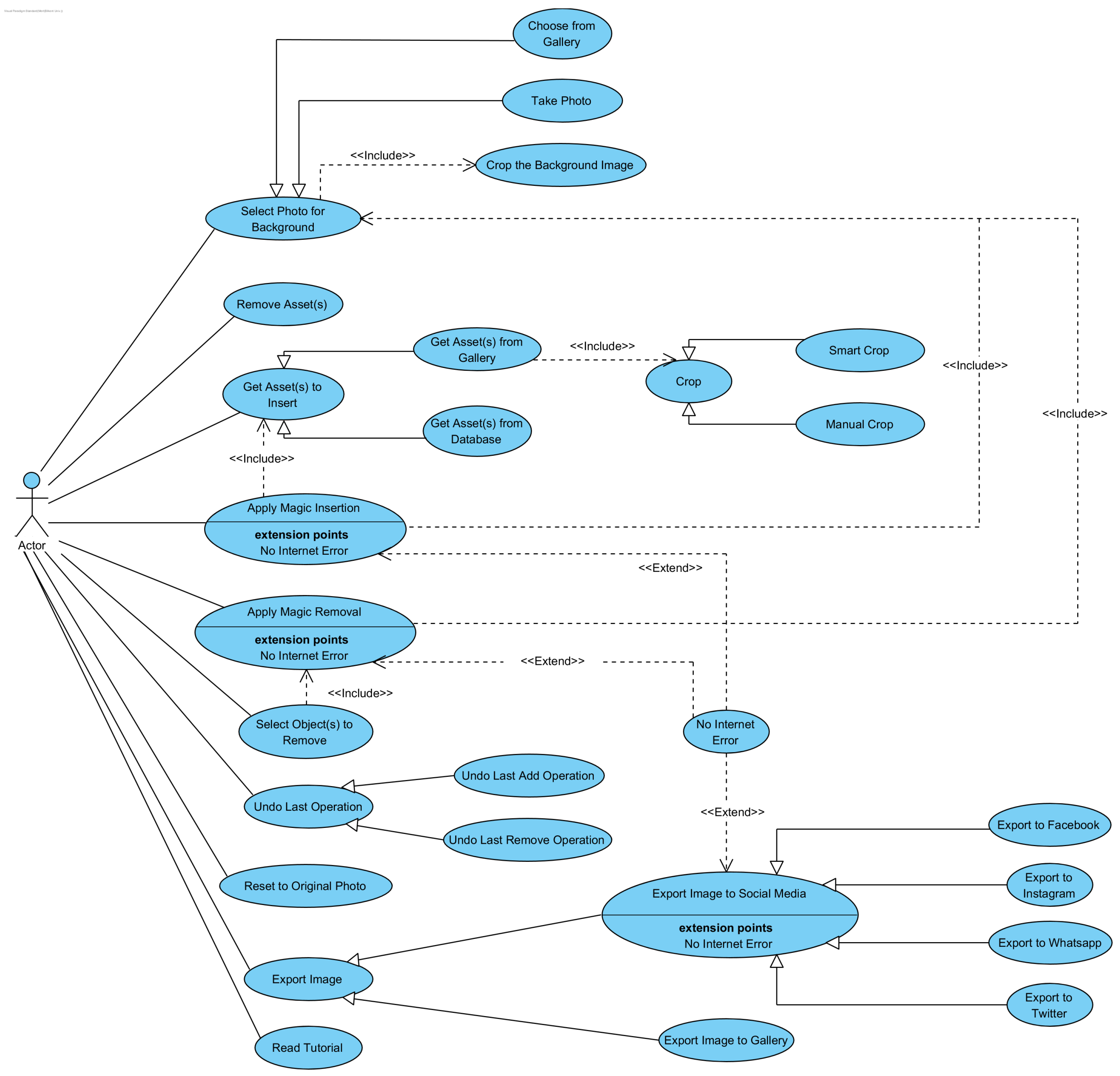


Figure 1: Use-Case Model

3.5.3 Object and Class Model

To demonstrate our initial design for program architecture, we will introduce object class diagram in this section. The purpose of the object class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages, hence they are widely used at the design phase of an application [12]. Below are brief descriptions of the classes we have put into our object and class model:

- **Image**: The data type to model a photo.
- **Asset**: A specific type of **Image** with a well-defined boundary that is not necessarily rectangular. This class is used to model the additional images.
- **Cropper**: An interface that provides image cropping functionality.
- **SmartCropper**: A specific **Cropper** that provides functionality to automatically detect an object inside a rectangular selection area and extract the object by computing a well-fitted boundary for it.
- **SmoothCropper**: Another **Cropper** that allows for defining a boundary using gestures. The functionality of this class corresponds to a user preparing an asset by cropping an image with the brush.
- **ManualCropper**: Yet another **Cropper** that lets cropping of an image according to a rectangular selection area.
- **Rotator**: A class to rotate images by a specified angle, which is calculated from user gestures.
- **Magnifier**: A class to resize or rescale images by a specified zoom factor, which is calculated from user's pinching gestures.
- **Boundary**: A data type to model an image boundary, meaning a potentially non-rectangular boundary that may look like any shape.
- **Model**: An interface that provides automated photoshopping functionality by making use of certain deep learning models.
- **Adder**: A specific **Model** that contains a neural network trained for context aware object addition, in other words automated image styling.

- **Remover**: Another **Model** that includes a neural network trained for context aware object removal. The class provides functionality to remove certain pixels from an image and put realistic artificial pixels in place.
- **ObjectIdentifier**: The last **Model**, which benefits from a neural network trained to detect the primary object inside a rectangular area and extract the object's **Boundary**.
- **Exporter**: A class that provides functionality to save an "autoshopped" image to a platform of choice (WhatsApp, Instagram, photo gallery, etc.)
- **AssetStorage**: A data type to store and search for **Asset**s.

In the next page, we provide the object class diagram which is the overall scheme that combines the classes described above, using the appropriate relationships. We specify the significant methods and attributes of the classes, which are subject to change. We introduce the notation *<DataType>* to refer to a platform/library specific data type; for instance Android's image representation or Java's prominent area representation.

Note that there will obviously be other classes that are necessary to carry out server communication, and others for the graphical user interface. Since this is an Analysis report we focused on the keystone class representations instead of the classes for the technical details. In the upcoming High-Level Design report, we will elaborate on these classes that capture the technical details.

The object class model follows in the next page.

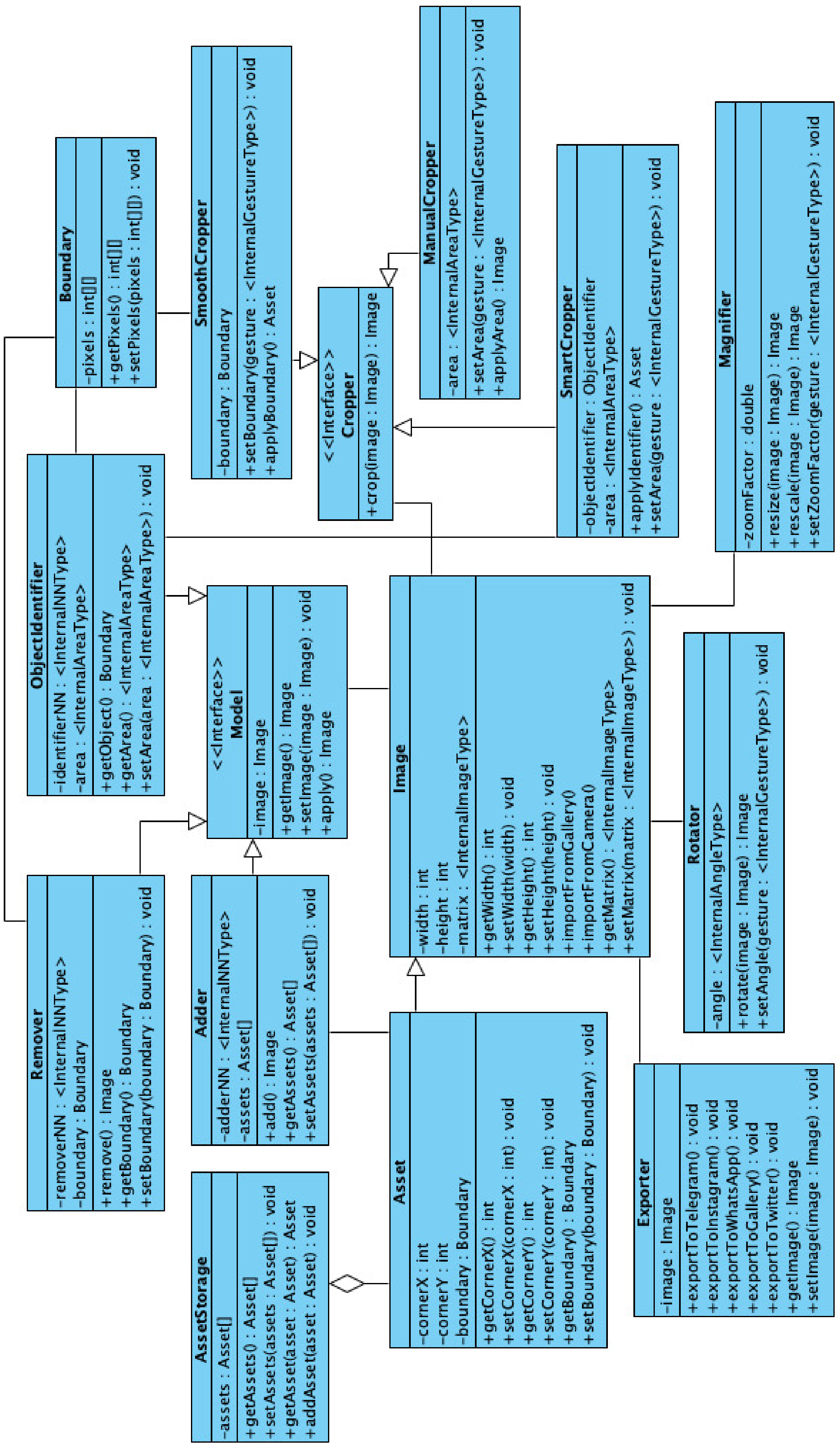

Boundary
-pixels : int[][]
+getPixels() : int[][]
+setPixels(pixels : int[][]) : void
SmoothCropper
-boundary : Boundary
+setBoundary(gesture : <InternalGestureType>) : void
+applyBoundary() : Asset
ManualCropper
-area : <InternalAreaType>
+setArea(gesture : <InternalGestureType>) : void
+applyArea() : Image
<<Interface>>
Cropper
+crop(image : Image) : Image
SmartCropper
-objectIdentifier : ObjectIdentifier
-area : <InternalAreaType>
+applyIdentifier() : Asset
+setArea(gesture : <InternalGestureType>) : void
Magnifier
-zoomFactor : double
+resize(image : Image) : Image
+rescale(image : Image) : Image
+setZoomFactor(gesture : <InternalGestureType>) : void
ObjectIdentifier
-identifierNN : <InternalNNType>
-area : <InternalAreaType>
+getObject() : Boundary
+getArea() : <InternalAreaType>
+setArea(area : <InternalAreaType>) : void
<<Interface>>
Model
-image : Image
+getImage() : Image
+setImage(image : Image) : void
+apply() : Image
Image
-width : int
-height : int
-matrix : <InternalImageType>
+getWidth() : int
+setWidth(width) : void
+getHeight() : int
+setHeight(height) : void
+importFromGallery()
+importFromCamera()
+getMatrix() : <InternalImageType>
+setMatrix(matrix : <InternalImageType>) : void
Rotator
-angle : <InternalAngleType>
+rotate(image : Image) : Image
+setAngle(gesture : <InternalGestureType>) : void
Remover
-removerNN : <InternalNNType>
-boundary : Boundary
+remove() : Image
+getBoundary() : Boundary
+setBoundary(boundary : Boundary) : void
Adder
-adderNN : <InternalNNType>
-assets : Asset[]
+add() : Image
+getAssets() : Asset[]
+setAssets(assets : Asset[]) : void
Asset
-cornerX : int
-cornerY : int
-boundary : Boundary
+getCornerX() : int
+setCornerX(cornerX : int) : void
+getCornerY() : int
+setCornerY(cornerY : int) : void
+getBoundary() : Boundary
+setBoundary(boundary : Boundary) : void
Exporter
-image : Image
+exportToTelegram() : void
+exportToInstagram() : void
+exportToWhatsApp() : void
+exportToGallery() : void
+exportToTwitter() : void
+getImage() : Image
+setImage(image : Image) : void
AssetStorage
-assets : Asset[]
+getAssets() : Asset[]
+setAssets(assets : Asset[]) : void
+getAsset(asset : Asset) : Asset
+addAsset(asset : Asset) : void

Figure 2: Object Class Diagram

3.5.4 Dynamic Models

Dynamic modeling of a system demonstrates the function calls and interactions between the system components, thus shows the inner processing of the system.

Sequence Diagrams

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order with which function calls [13]. We introduce following eight example sequences for our system.

Main Page Sequence Diagram

This diagram displays how Steve loops inside the main page by either choosing a photo from the gallery or going to the settings page.

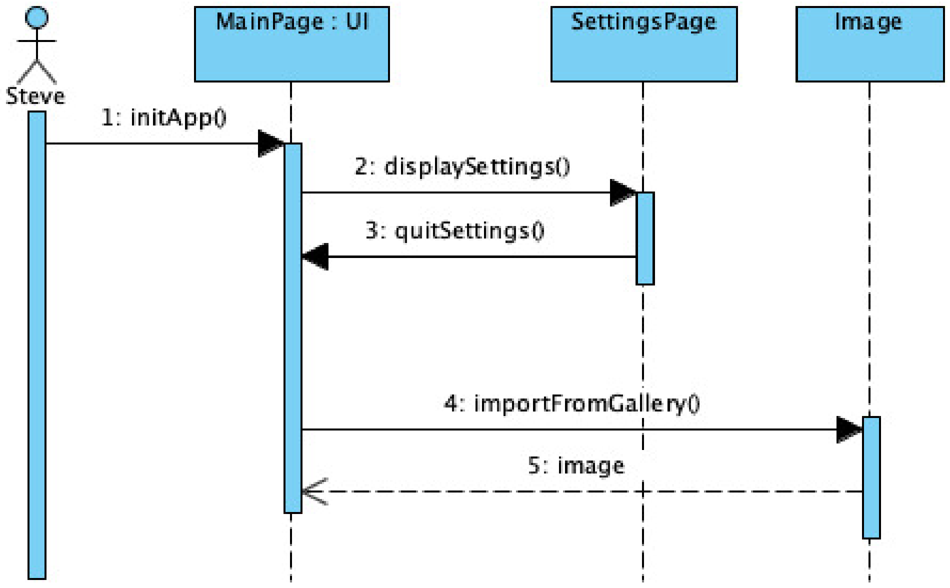


Figure 3: Main Page Sequence Diagram

Add Asset Sequence Diagram

This diagram shows how Steve can recursively add assets, in other words additional images, on top of the background image; by either selecting from the asset storage or generating a new asset.

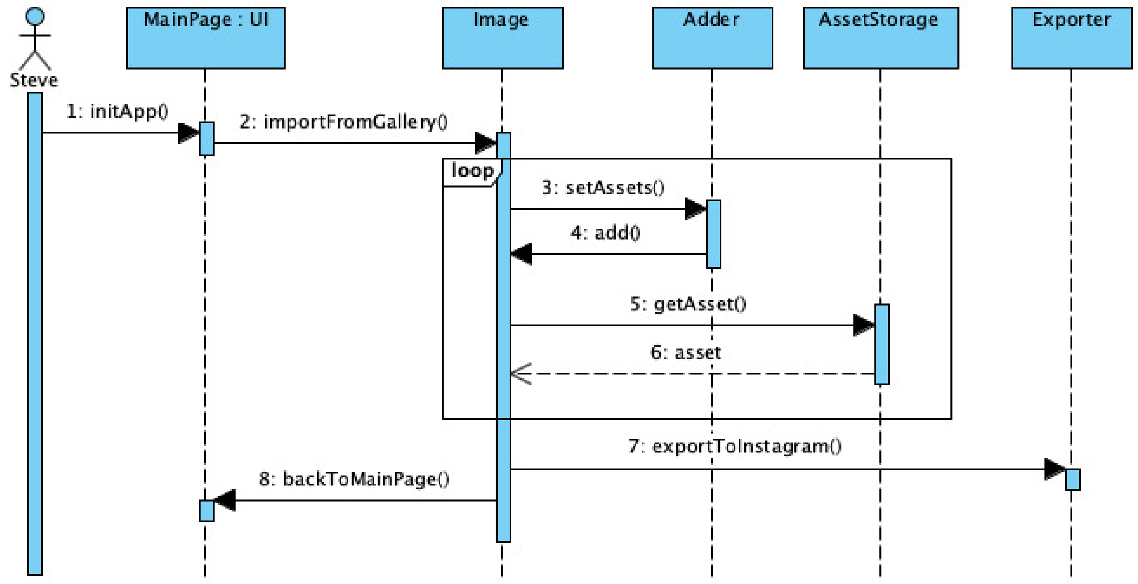


Figure 4: Add Asset Sequence Diagram

Add with Smooth Crop Sequence Diagram

This diagram together with the following two diagrams demonstrates how Steve can add objects to his images by cropping the objects with different methods in order to turn them into assets. The cropping methods include *Smooth*, *Smart* and *Manual* cropping.

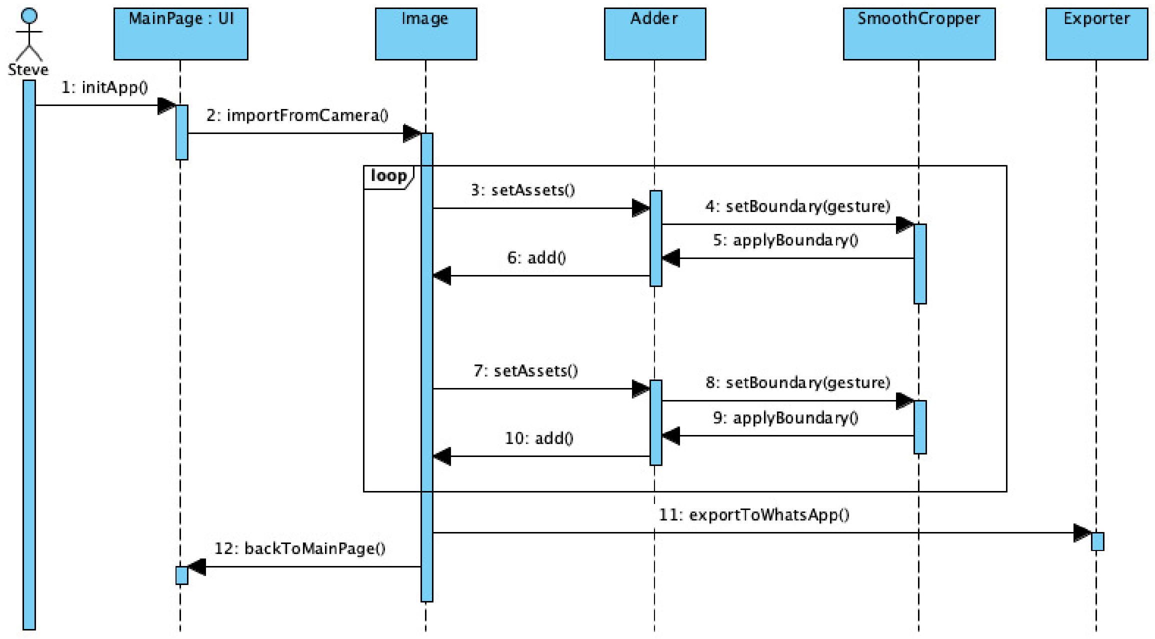


Figure 5: Add with Smooth Crop Sequence Diagram

Add with Smart Crop Sequence Diagram

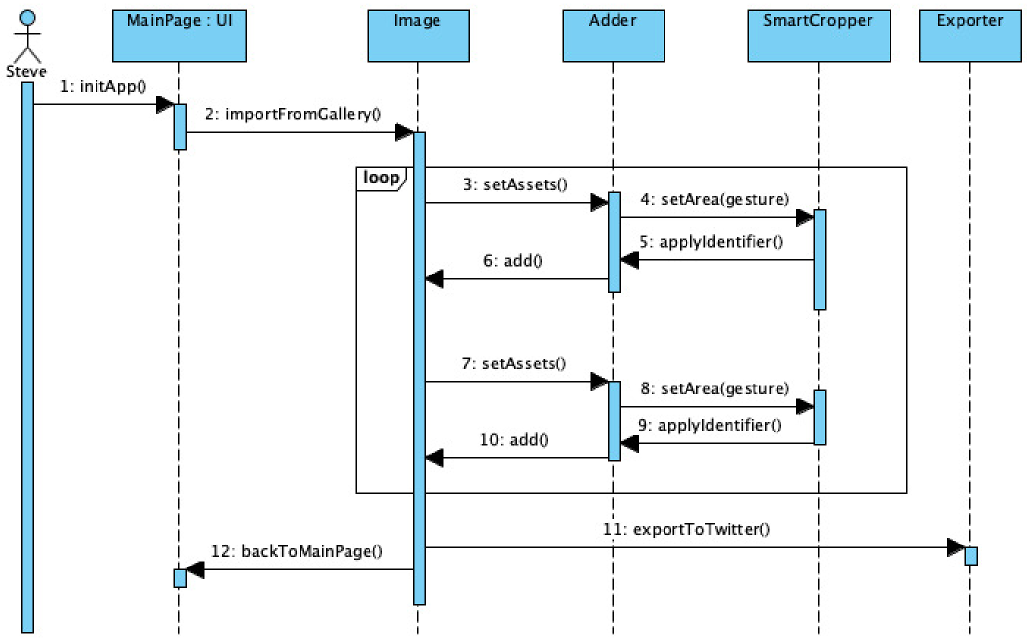


Figure 6: Add with Smart Crop Sequence Diagram

Add with Manual Crop Sequence Diagram

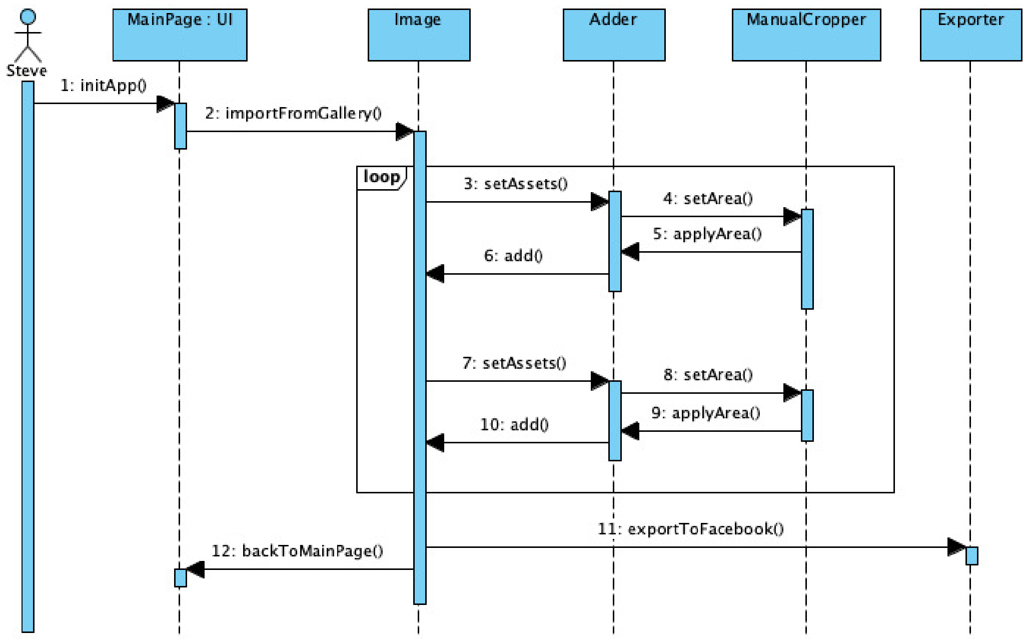


Figure 7: Add with Manual Crop Sequence Diagram

Remove with Smooth Crop Sequence Diagram

This diagram together with the following two diagrams demonstrates how Steve can remove objects from his images by cropping the objects with different methods in order to define removal boundaries. The cropping methods again include *Smooth*, *Smart* and *Manual* cropping.

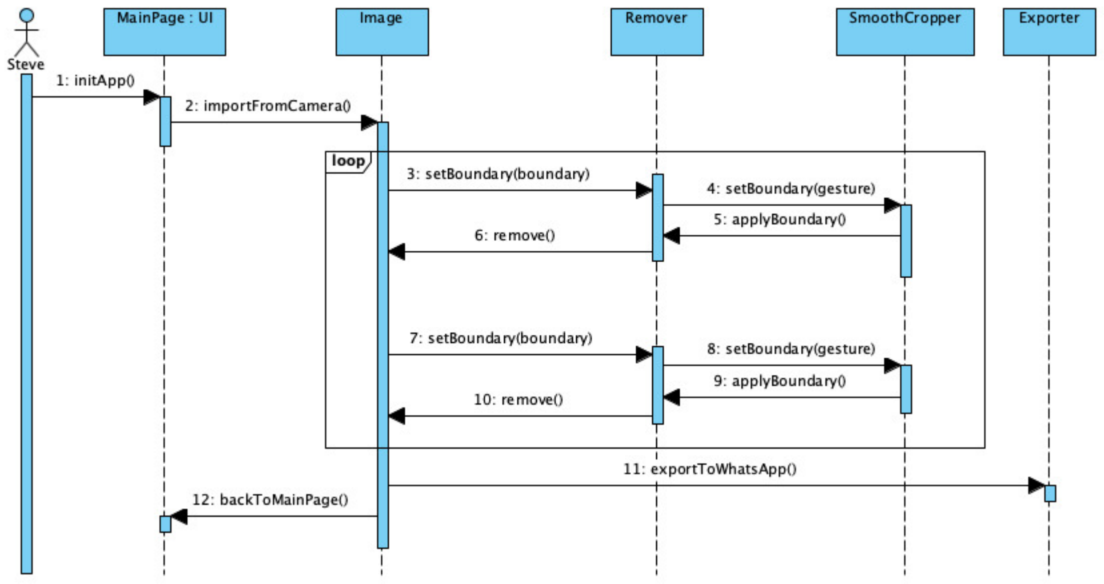


Figure 8: Remove with Smooth Crop Sequence Diagram

Remove with Smart Crop Sequence Diagram

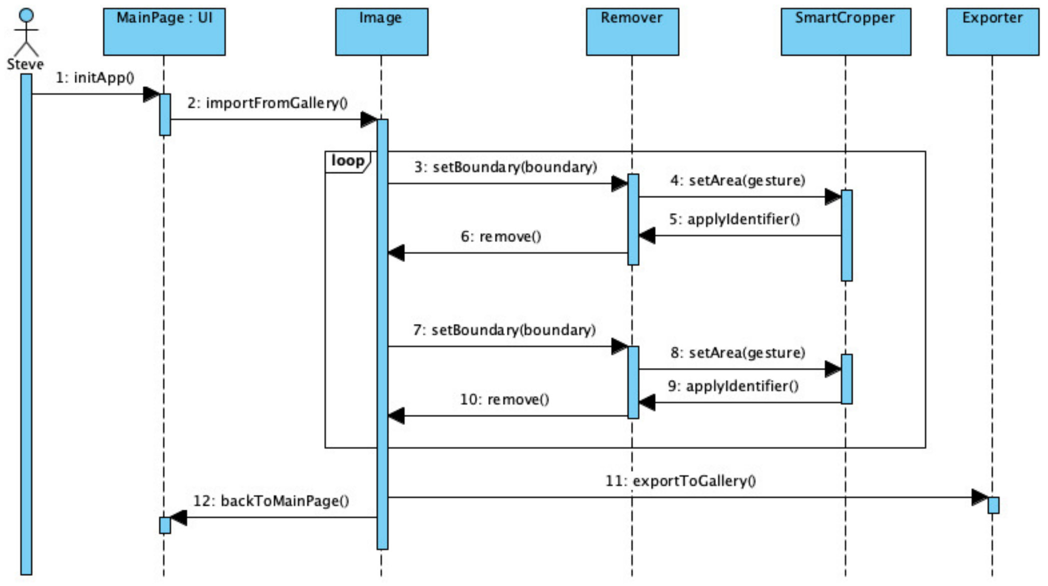


Figure 9: Remove with Smart Crop Sequence Diagram

Remove with Manual Crop Sequence Diagram

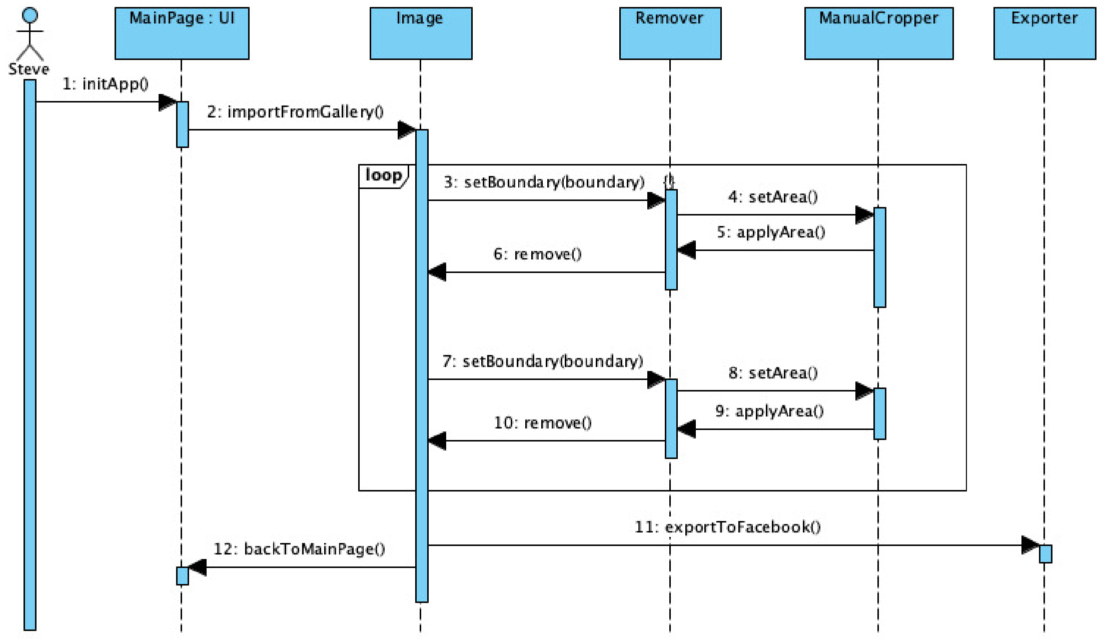


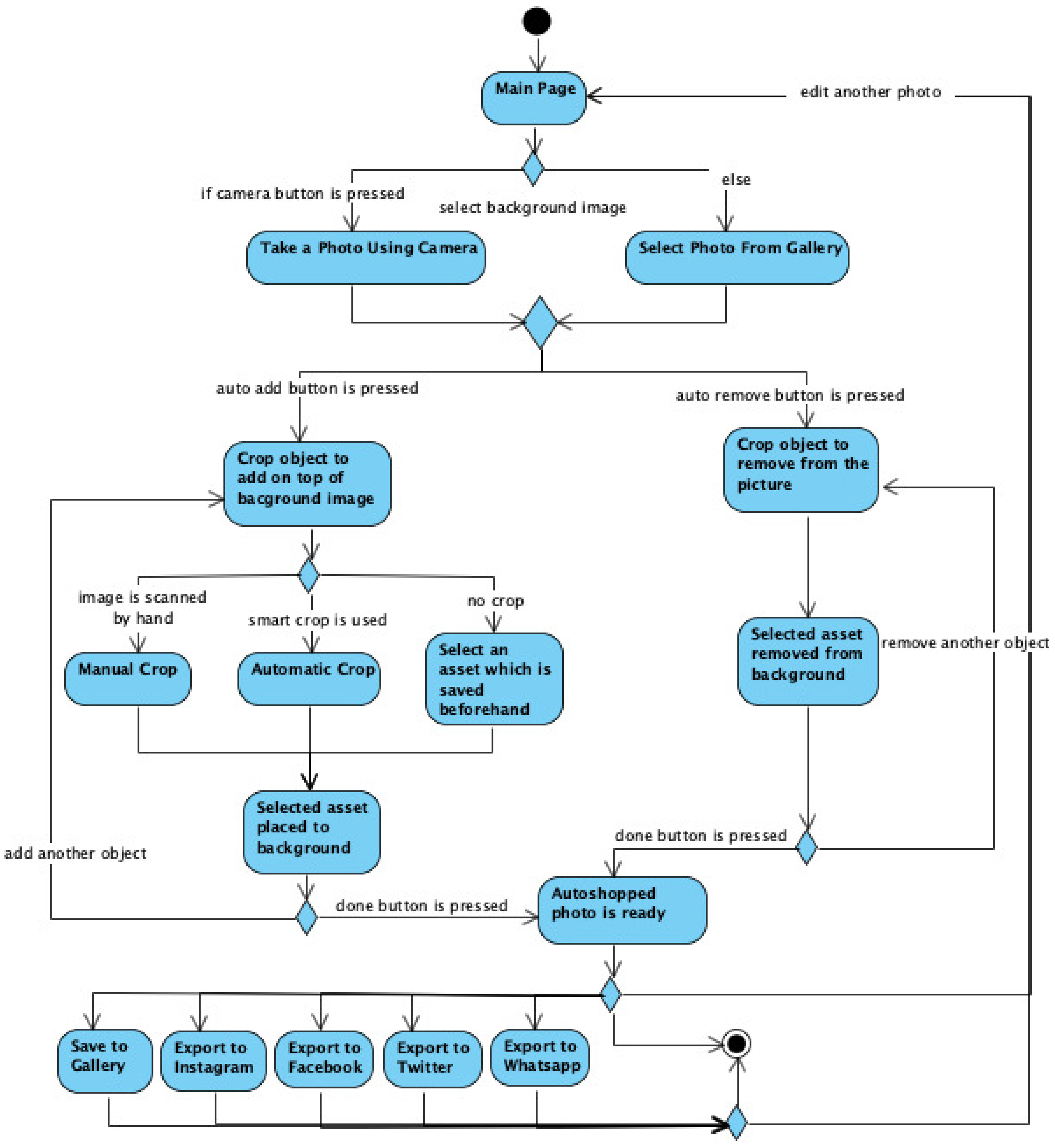
Figure 10: Remove with Manual Crop Sequence Diagram

Activity Diagrams

Activity diagram is another important diagram in unified modeling language to describe the dynamic aspects of the system that represents the flow from one activity to another activity [14]. We included following two examples of activity sequences for our system, which are provided in the next two pages.

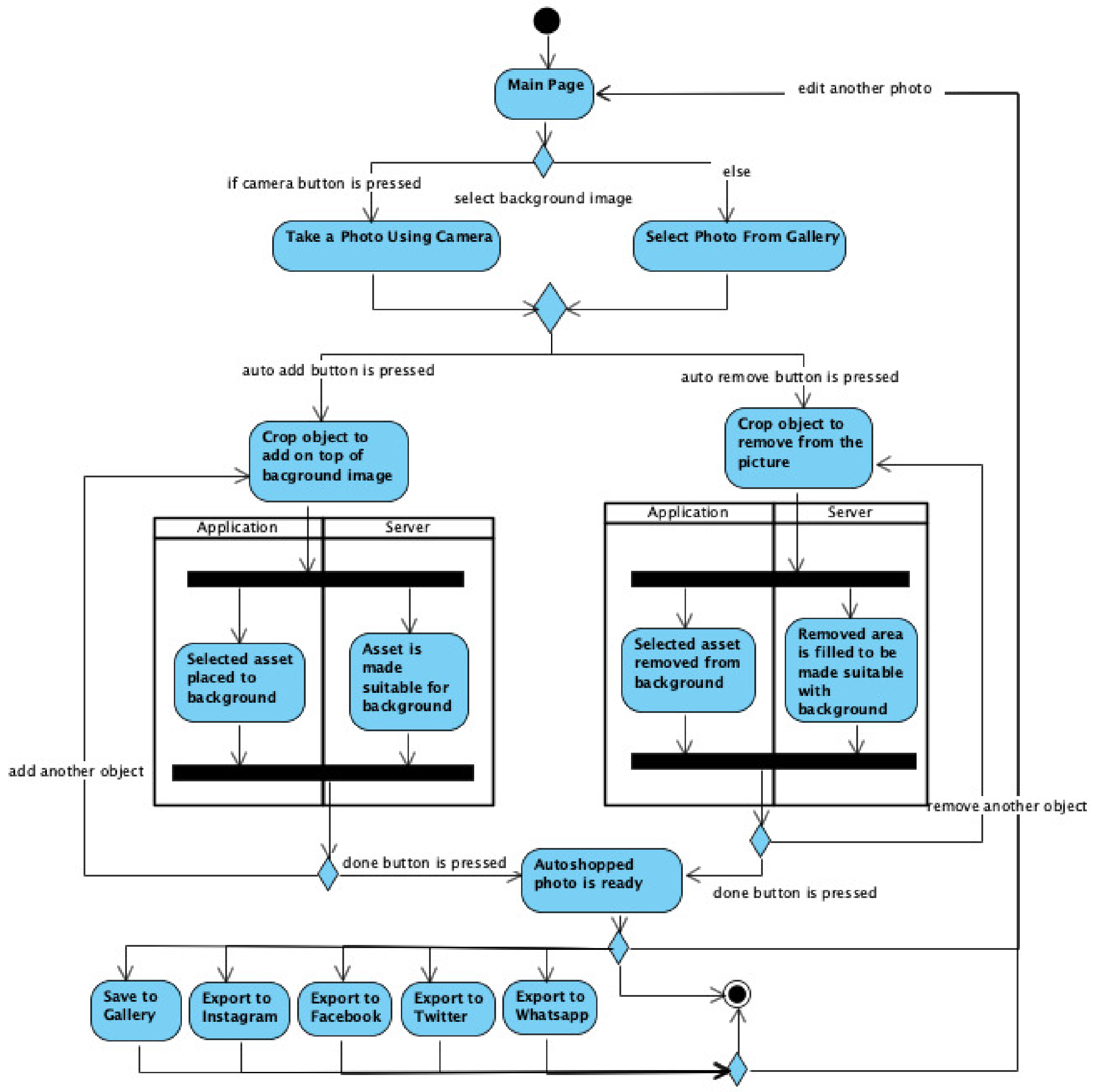
The first activity diagram is the *Main Flow Activity Diagram*. It provides an algorithmic flow-chart like view of the overall application. To be clearer, the diagram specifies which actions to take in order from the point you open the application to the point you export the "autoshopped" photo.

The second activity diagram *Object Addition/Removal Activity Diagram* which focuses specifically on the context aware object addition and removal processes. The diagram depicts the concurrent execution of the server and the application using vertical swimlanes. Note that concurrent execution is a must for us for performance purposes.



Main Page
edit another photo
if camera button is pressed
select background image
else
Take a Photo Using Camera
Select Photo From Gallery
auto add button is pressed
auto remove button is pressed
Crop object to add on top of bacground image
Crop object to remove from the picture
image is scanned by hand
smart crop is used
no crop
Manual Crop
Automatic Crop
Select an asset which is saved beforehand
Selected asset removed from background
remove another object
Selected asset placed to background
add another object
done button is pressed
done button is pressed
Autoshopped photo is ready
Save to Gallery
Export to Instagram
Export to Facebook
Export to Twitter
Export to Whatsapp

Figure 11: Main Flow Activity Diagram



Main Page
edit another photo
if camera button is pressed
select background image
else
Take a Photo Using Camera
Select Photo From Gallery
auto add button is pressed
auto remove button is pressed
Crop object to add on top of bacground image
Crop object to remove from the picture
Application
Server
Selected asset placed to background
Asset is made suitable for background
Selected asset removed from background
Removed area is filled to be made suitable with background
add another object
remove another object
done button is pressed
Autoshopped photo is ready
done button is pressed
Save to Gallery
Export to Instagram
Export to Facebook
Export to Twitter
Export to Whatsapp

Figure 12: Object Addition/Removal Activity Diagram

3.5.5 User Interface - Navigational Paths and Screen Mock-ups

In this part of the report, the user-interface of the application will be presented. Each sketch is provided with an explanation to better specify how it addresses the requirements mentioned above.

Logo Page

The user will be welcomed with the *Logo Page* having the logo of *Autoshop* and production remarks noted at the bottom of the page as shown in Figure 13. User cannot interact with the application at this page while the application is initializing the network with the servers.

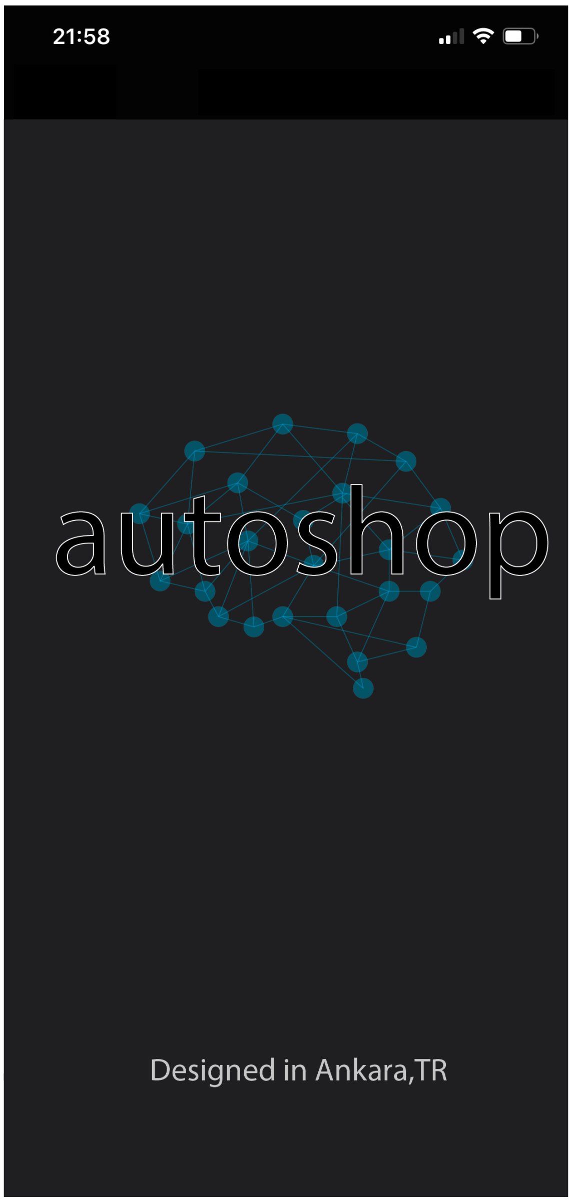


Figure 13: Logo Page

Add Background Photo Page

After *Logo Page*, user will be directed to *Add Background Photo Page* for choosing a background photo that is required for both *Auto Remove* and *Auto Add* functionalities of *Autoshop*. At this page user can choose the background image from the gallery or can take a photo after taping the photo icon on the top of the page, as shown in Figure 14. On the bottom-left corner of this page user can tap *information* icon to see the tutorial about the usage of the application and on the bottom-right corner of the page user can tap *settings* icon to be directed to *Settings Page*.

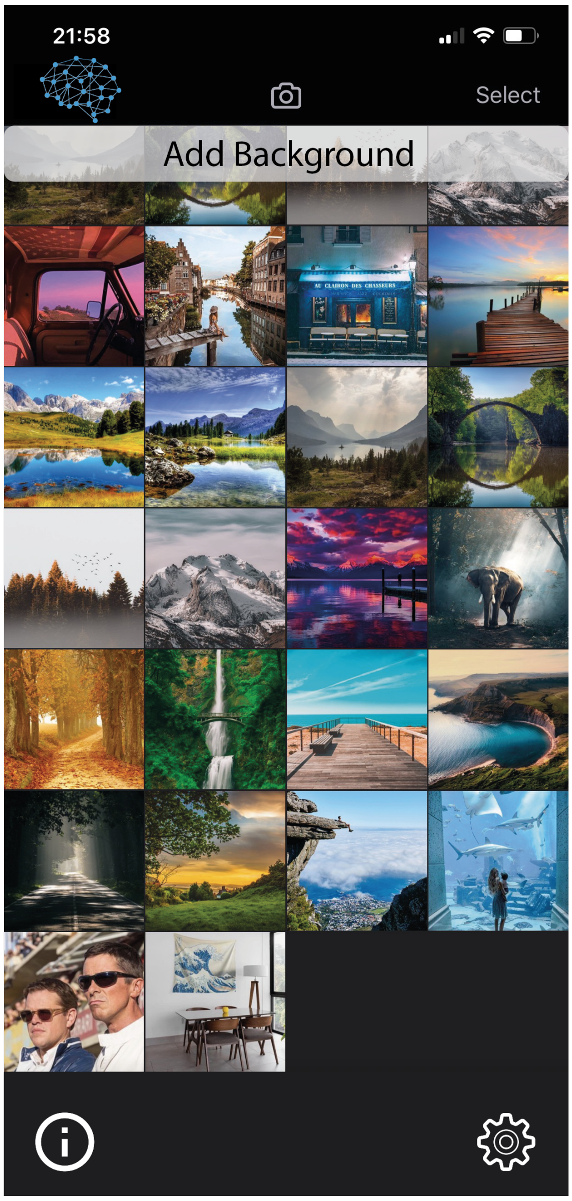


Figure 14: Add Background Photo Page

Add or Remove Page

After selecting the background image user will be asked for the *Autoshop* functionality she will proceed as shown in Figure 15. If she chooses *Auto Add*, she will be directed to *Add Additional Image Page* or if she chooses *Auto Remove* she will be directed to *Auto Remove Page*. Before going to next stage, user can crop the background image after tapping the *crop* icon on the top-right corner of this page. Also the user can go to the previous page by tapping the *left arrow* icon on the top-left corner.

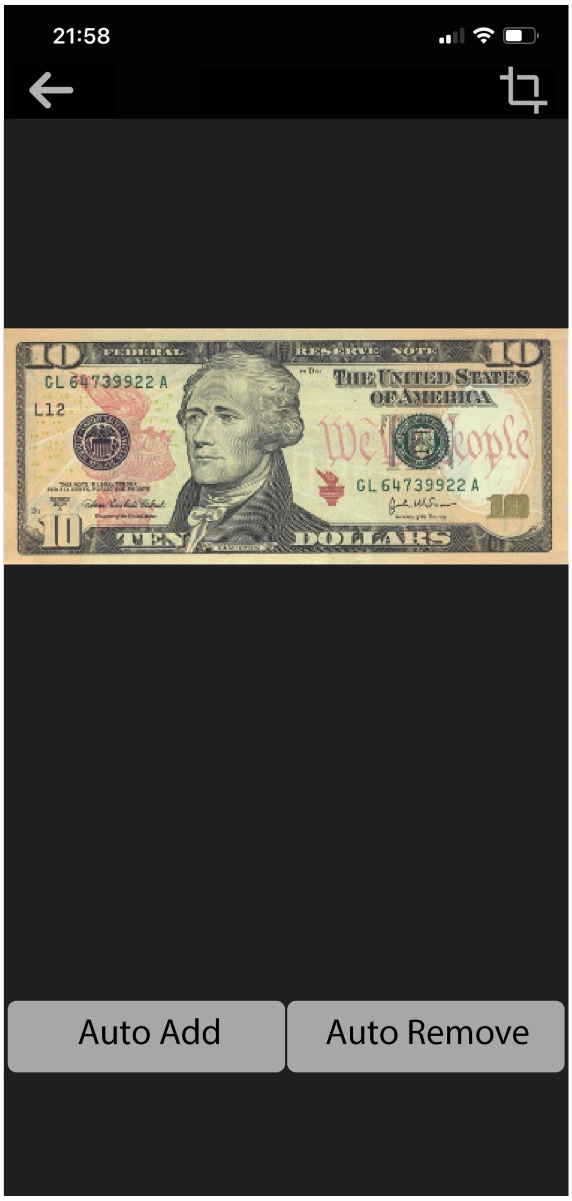


Figure 15: Add or Remove Page

Add Additional Image Page

User will be directed to this page if she chooses *Auto Add* option in *Add or Remove Page* as she will need an additional image. She can choose a photo from gallery directly or can tap the *Choose from Assets* button on the bottom of the page to import from previously saved assets. If the user chooses a photo from gallery she will be directed to *Crop Page* to crop the desired part of the photo. Also the user can go to the previous page with tapping the *left arrow* icon on the top-left corner.

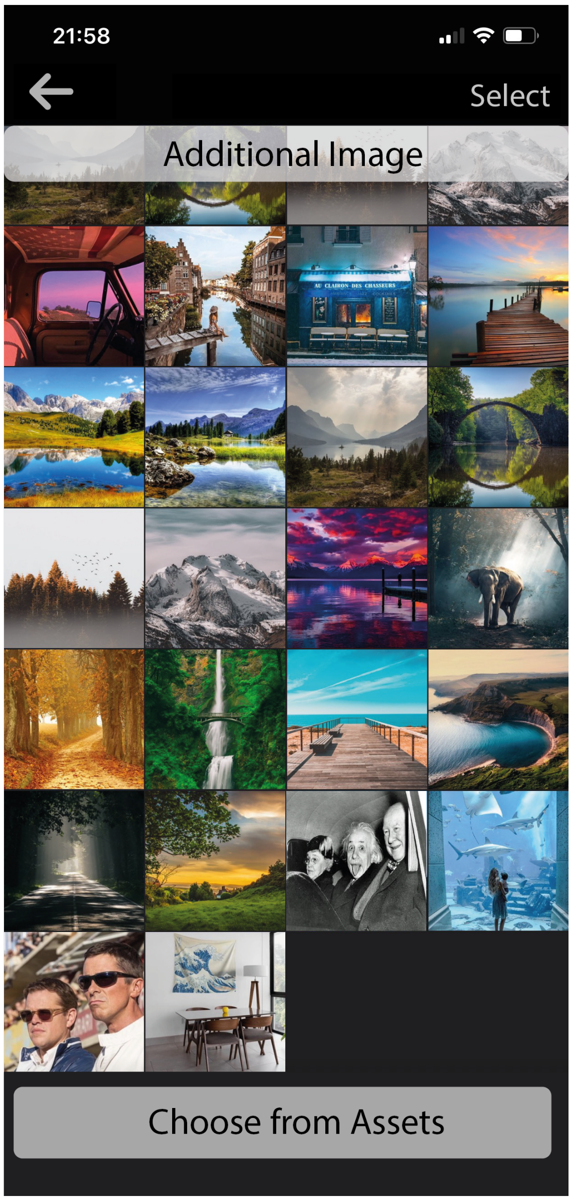


Figure 16: Add Additional Image Page

Crop Page

User will be directed to this page if she chooses a photo from gallery to further crop the image. In this page the user will choose a method to crop the image using the *Manual*, *Smooth* and *Smart* buttons on the bottom of the page as shown in Figure 17. After choosing the method she can crop the photo by tapping the *Crop* icon on the top-right corner of the page. User can also go to the previous page by tapping *left arrow* on the top-left corner of the page.

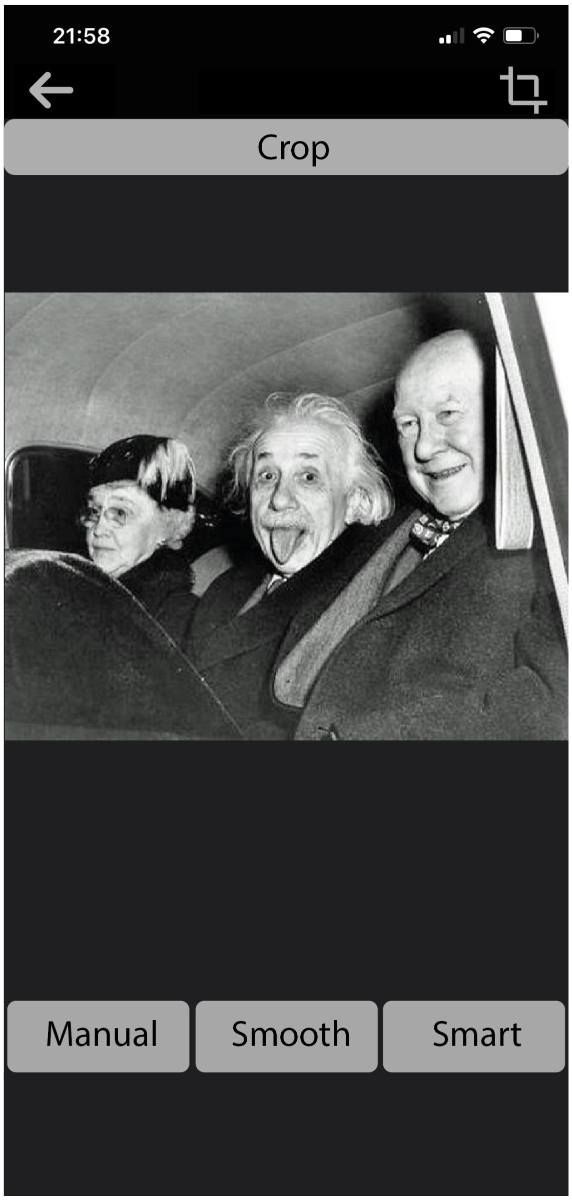


Figure 17: Crop Page

Manual Crop Page

User will be directed to this page if she chooses *Manual* option in *Crop Page*. *Manual Crop* requires user to select a rectangular region as shown in Figure 18. While cropping user can undo her last move by tapping *Undo* button on bottom-left corner. After she decides on the rectangular region she taps the *Done* button on bottom-right corner to be directed to the *Merge Page*. Also the user can go to the previous page by tapping the *left arrow* icon on the top-left corner.

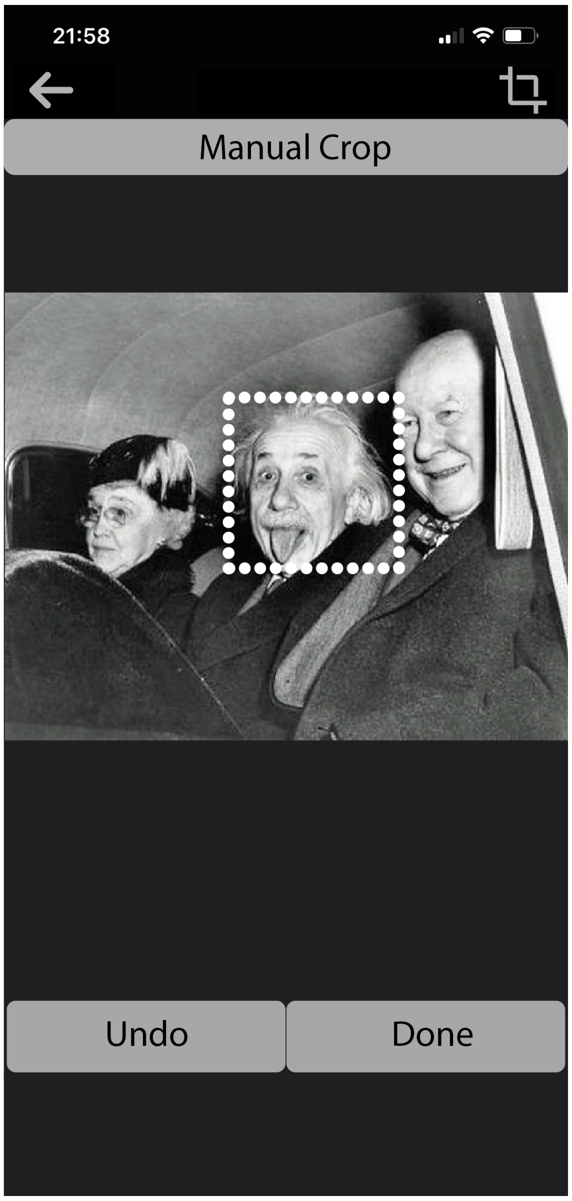


Figure 18: Manual Crop Page

Smooth Crop Image

User will be directed to this page if she chooses *Smooth* option in *Crop Page.* For *Smooth Crop* user is required to draw the rough edges of the object. The application will further process this input to perform a more optimized cropping. While drawing the edges, user can undo her last move by tapping *Undo* button on the bottom-left corner as shown in Figure 19. After she is satisfied with her drawing she taps the *Done* button on bottom-right of the page to be directed to the *Merge Page.* Also the user can go to the previous page with tapping the *left arrow* icon on the top-left corner.

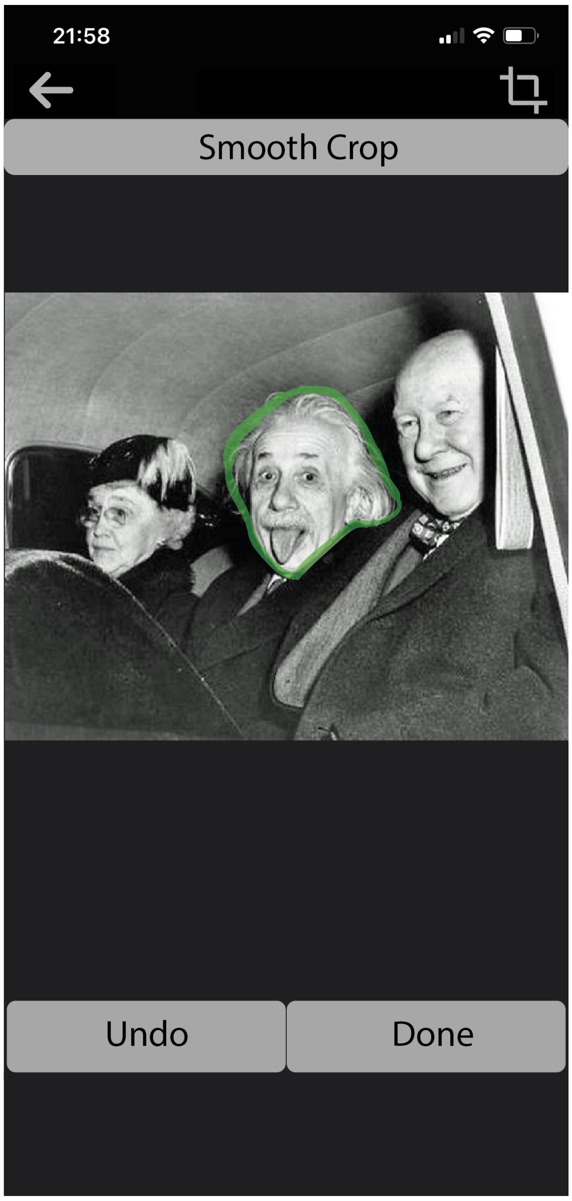


Figure 19: Smooth Crop Page

Smart Crop Image

User will be directed to this page if she chooses *Smart* option in the *Crop Page*. *Smart Crop* requires a rectangular region from user that includes the desired object. This input will be processed by *Autoshop* to detect the object inside the region and it will be cut from its edges. While drawing the rectangle, user can undo her last move by tapping *Undo* button on bottom-left corner as shown in Figure 20. After she is satisfied with the rectangular region, she taps the *Done* button on bottom-right of the page to be directed to the *Merge Page*. Also the user can go to the previous page by tapping the *left arrow* icon on the top-left corner.

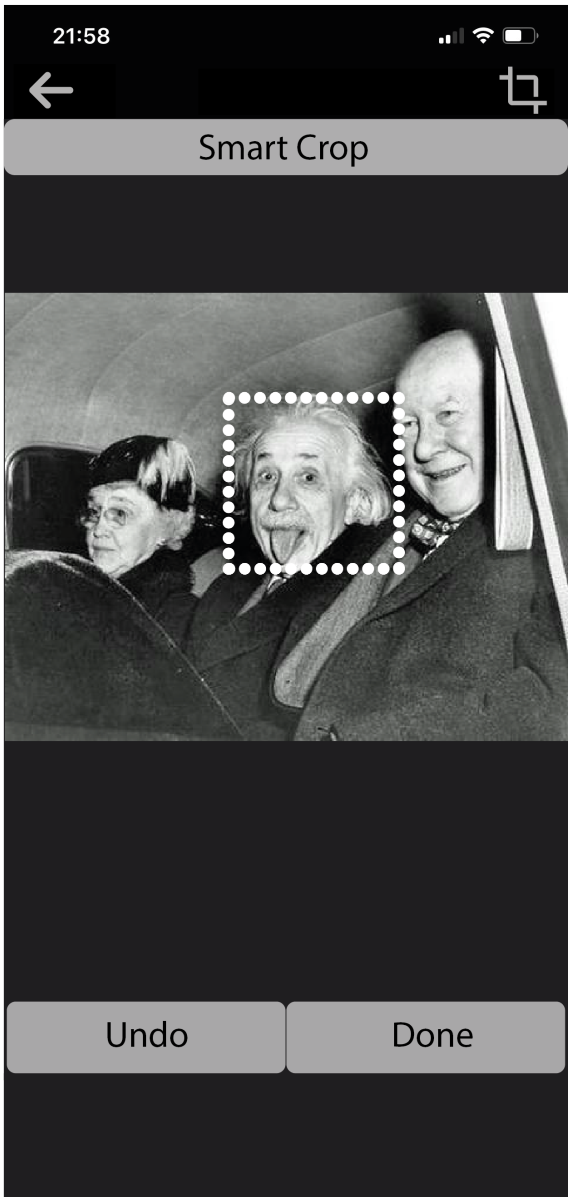


Figure 20: Smart Crop Page

Choose from Assets Page

User will be directed to this page if she taps *Choose from Assets* button in the *Add Additional Image Page*. The user will be able to access her previously created assets as shown in Figure 21 and will be able to use them without further cropping. The user can go to the previous page by tapping the *left arrow* icon on the top-left corner.

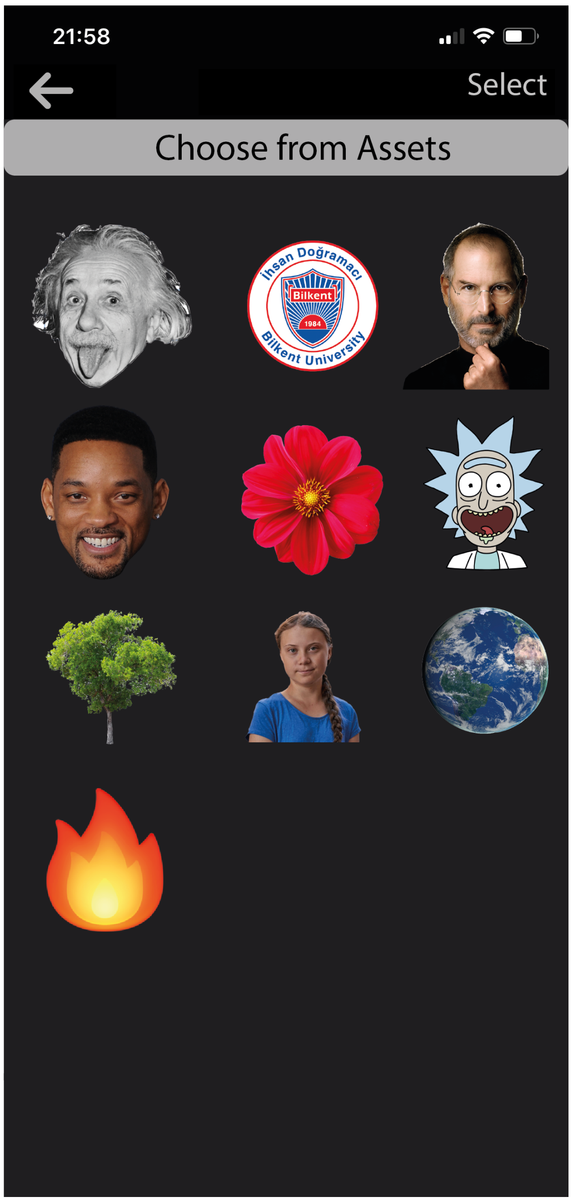


Figure 21: Choose From Assets Page

Merge Page

After having the *Background Image* and *Additional Image* user will be directed to this page to decide on the scales of *Background Image* and *Additional Image* also the relative positions of them, using tap gestures. As shown in Figure 22, the user will tap *Apply Auto Add* button to get the final image. The user can go to the previous page by tapping the *left arrow* icon on the top-left corner.

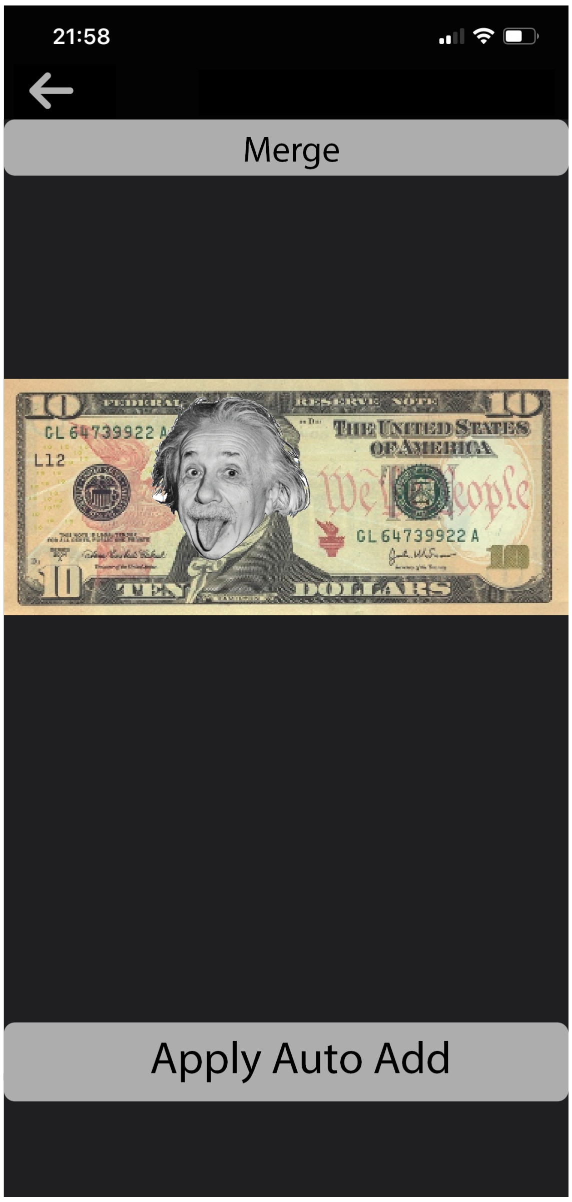


Figure 22: Merge Page

Export Page

This page will show the final output to the user. User can export the final output to social media platforms such as *Facebook*, *Instagram*, *Whatsapp* and *Twitter* with the buttons on the bottom of the page as shown in Figure 23. User can save the photo to gallery by tapping the *Save* icon on the top-right corner of the page. The user can go to the previous page by tapping the *left arrow* icon on the top-left corner.

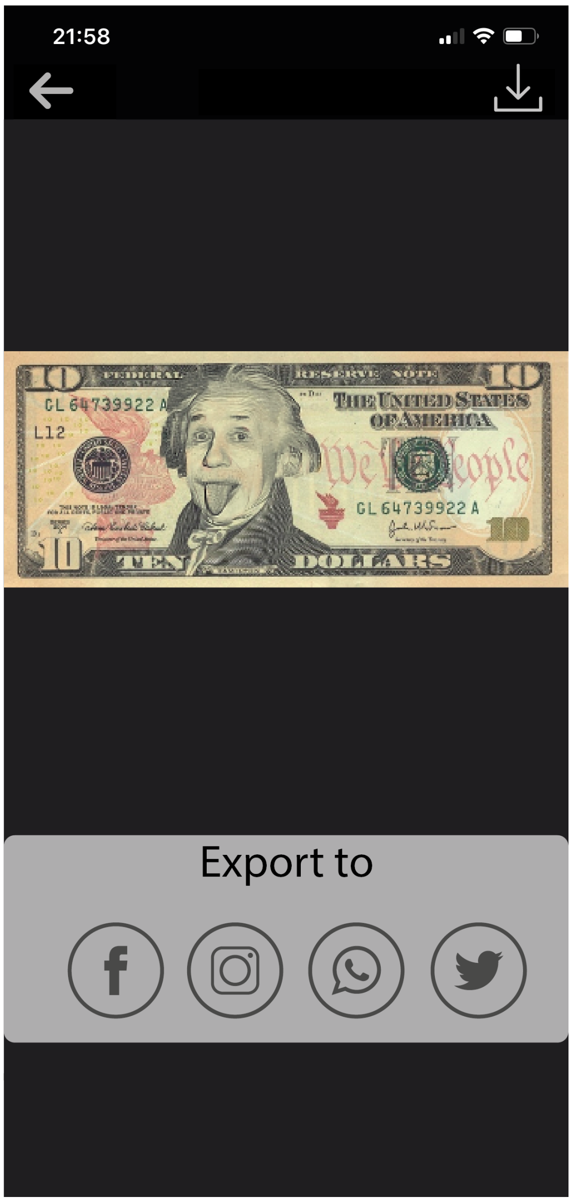


Figure 23: Export Page

Settings Page

User will be directed to this page if she taps *settings* icon in *Add or Remove Page*. This page will have buttons to direct user to *Contact*, *Rate the App*, *Share the App*, *Team* and *FAQ* links as shown in Figure 24. User can go to the previous page by tapping the *left arrow* icon on the top-left corner.

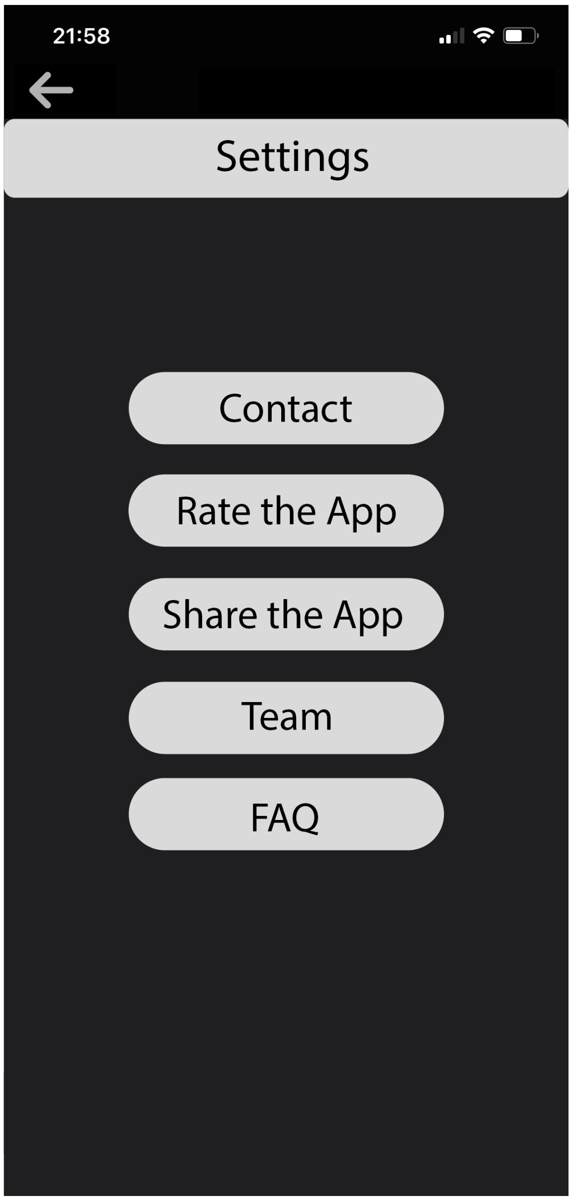


Figure 24: Settings Page

Auto Remove Page

User will be directed to this page if she chooses *Auto Remove* option in *Add or Remove Page*. The user will be required to scan the object to be removed with the brush at this page. The user can change brush size as shown in Figure 25. After scanning the object with the brush, user will tap the *Apply Auto Remove* button to get the final image, then the user will be directed to the *Export Page*. User can go to the previous page by tapping the *left arrow* icon on the top-left corner.

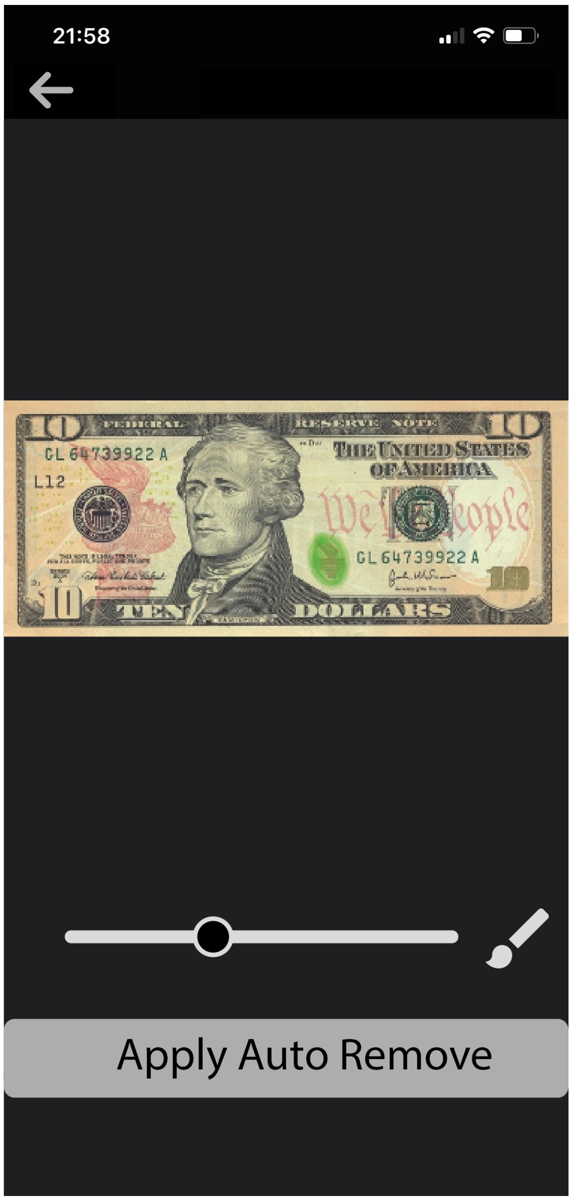


Figure 25: Auto Remove Page

Camera Page

If user taps the *camera* icon in the *Add Background Image Page* she will be directed to this page. Here the user can switch the cameras, turn on-off the flash and take a photo as shown in Figure 26. The user can go to the previous page with tapping the *left arrow* icon on the top-left of the corner.

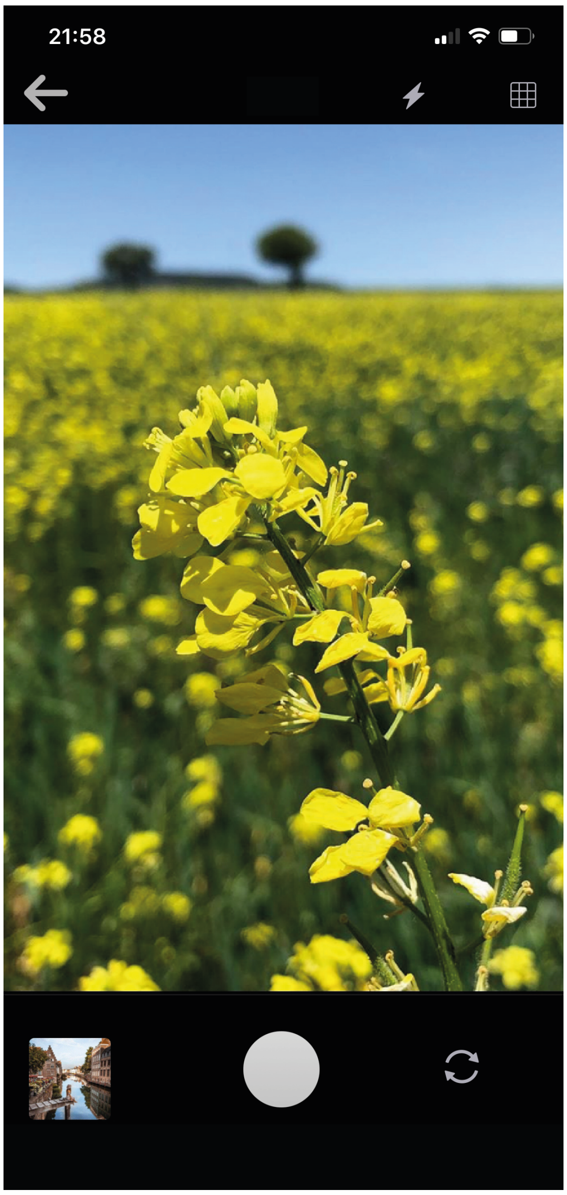


Figure 26: Camera Page

Navigational Path

The user navigational path for the pages of the application is shown below. User can also go back to the previous page with *left-arrow* button, that relation is not shown in the figure for simplicity.

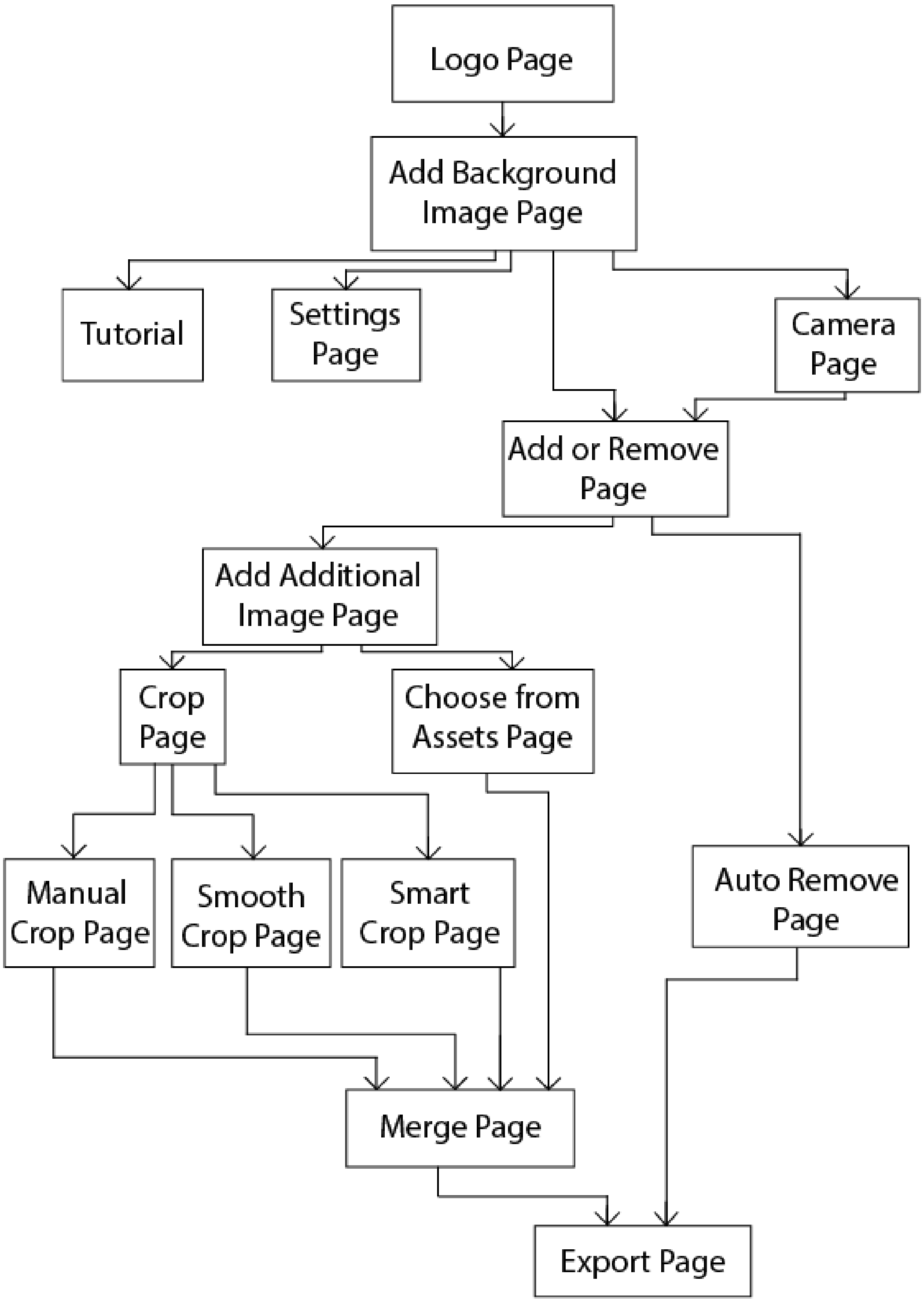


Figure 27: Navigational Page

4 Other Analysis Elements

In this part of the report, we present an extensive discussion which is related to certain soft-skills in software engineering and social dimensions of the project.

4.1 Consideration of Various Factors

There exists limiting factors in our project that requires an extensive discourse. Below are the factors limiting our design and brief discussions on the potential effects of these factors:

Public Health

There is no physical illness that can be caused by our application, at least we cannot think of one right now. However, our application may trigger certain psychological problems. Those psychological problems are closely related with social media. Research show that social media stimulates a wide range of mental problems that are closely related with anxiety and inadequacy [15]. People are observed to be comparing themselves with macro-influencers and feel as if they cannot measure up with the "ideal" look depicted by them. One use of *Autoshop* is to modify a personal photo such that the photo becomes aligned with the social norms of beauty. This use case may potentially be a primer to an upcoming use of social media where everyone tries so hard to comply with social norms of beauty, which in turn causes self-dissatisfaction and originate even more mental problems. To prevent this to some extend, we made the design choice where the "autoshopping" tutorial used in the application is periodically changed and the tutorial photos are designed to be inclusive as possible.

Public Safety

Autoshop may be used to depicture unreal people, events or locations as if they are real. Consider an extreme use case where a burglar robs a shop, accesses the security camera recordings and uses *Autoshop* to change the frames of the recordings by putting another face on top of his. Assuming that such a use case is probable, our application can be used as a tool to corrupt evidence that is necessary to charge certain crimes and insure public safety. To defend against cases like this, we are planning to add some type of digital signature that indicates the image is "autoshopped".

Public Welfare

Since our application is planned to be free to download and contain no in-app purchases, it has no effect on public welfare. Anyone owning a mobile phone can download the application for free and downloading the application, obviously, will not affect who owns a mobile phone and who does not. Thus, a discussion on public welfare is not applicable to our context.

Global Factors

In our discussion, global factors refers to rules, regulations and norms that nearly every country come to a mutual agreement on. Such factors include well-defined standards like General Data Privacy Regulation (GDPR) [16], Google Play Terms of Service [17] and other explicitly written or implicit agreements between the developers, users and authorities. We inevitably limit ourselves to these factors. Since these factors, in some sense, reflect the public-opinion, following them is essential for us to build an inclusive application.

Cultural Factors

We have mentioned that we will retrain the generative models that perform the automatized photoshopping using the data accumulated in the application, meaning the input, output pairs which are produced by the user. Of course, users permission is taken before this data is stored and fed into the model. The data gathered from different cultures may introduce an unexpected bias into our model. For instance, Asian countries may use *Autoshop* more frequently than other countries, and the data collected from these countries may result in a model that favors slanted eyes over regular eyes. To prevent such a cultural bias incorporated into our model, we will sample our training data equally among a variety of locations.

Social Factors

There is no indubitable definition of a social factor, but factors such as age, religious orientation, family history, race and ethnicity, education, and economic status are some examples of social factors. We cannot identify a clear link between these social factors and our application. We can only indicate that the argument of cultural bias presented in the cultural factors section above is also applicable for an argument of an ethnic bias.

In the table below, we indicate the significance, or equivalently the level of effect, on a scale of 0 (none) to 10 (maximum) of each factor we have mentioned above:

	Effect Level	Effect
Public Health	7	Psychological Problems of Anxiety and Inadequacy among Users
Public Safety	7	Corrupted Evidences
Public Welfare	0	Not Applicable
Global Factors	9	An Exclusive Application that is Far from Global Standards
Cultural Factors	7	Cultural Biases Incorporated into the Models
Social Factors	4	Ethnic Biases Incorporated into the Models

Table 1: Factors that can effect analysis and design

4.2 Risks and Alternatives

Autoshop contains several implementation risks that may be crucial in the development phase. Mentioning these risks and coming up with B plans is a must for us. Below are these risks and our plan B's to overcome them:

1. NVIDIA Libraries Becoming Inaccessible to Public:

   As we have mentioned before, our application depends highly on two NVIDIA's libraries, namely `PartialConv` and `FastPhotoStyle`. `PartialConv` is used for image impainting, in other words context aware removal of objects from images; whereas, `FastPhotoStyle` is used for blending objects into an image. These libraries are currently open source, and readily importable in `Python` language. As a backup plan, we have already downloaded these libraries and in case of a restrictive license, we will practice on the libraries' code, then come up with our own architectures and models.

2. Unsatisfactory Server Performance:

   Since the task of model preparation is GPU heavy and the server communication is time sensitive, the server performance is a substantial factor for us. In case the server performance is unsatisfactory, we will move to a better equipped server.

3. Infeasible Server Costs:

   Another risk is the server costs exceeding our economic threshold. In this case, we will try to optimize the models and try to make them operate in the local machine; also we may charge some features of the application. Note that there is a trade of between server performance and server costs.

4. One of Team Members Leaving:

   One of the team members leaving for a certain time by becoming injured or withdrawing the course is also an unexpected but important risk. To overcome this risk with minimal damage, we are keeping the bus factor as low as possible. Bus factor is a measurement of the risk resulting from information and capabilities not being shared among team members, derived from the phrase "in case they get hit by a bus." [18].

5. Google Play Store Banning *Autoshop* Due to Deep Fakes:

   Section 4.5 discusses the issue of Deep Fakes, which may become a serious societal problem. In this scenario, our application may be banned by Google's play store as it can be potentially seen as a deep fake generator. Our plan for this situation is to carry the application to web.

Below is a table that summarizes the risks mentioned above:

	Likelihood	Effect on the Project	B Plan Summary
NVIDIA Libraries Becoming Inaccessible to Public	Low	Non-functional dependencies	Developing our own models
Unsatisfactory Server Performance	Medium	Poor user experience	Move to better equipped servers
Infeasible Server Costs	Medium	Necessity of optimization	Optimization and charging for certain features
One of the Team Members Leaving	Low	Decrease in the work power	Reducing the bus factor
Google Play Store Banning *Autoshop* Due to Deep Fakes	Low	Application getting banned	Web deployment

Table 2: Risks and B Plans

4.3 Project Plan

Planning is the key to produce deliverables while meeting the deadlines. Various project planning paradigms can be used in a software engineering project, however the one we have chosen is based on the decomposition of the project into smaller pieces. The paradigm we are following is called the *Work Breakdown Structure (WBS)*. Formally, it is a process which fundamentally subdivides the project deliverables and the project into more manageable, minor components. The rationale behind this paradigm is to recursively decompose the components and arrive at a base level, where the smallest component is called a *Work Package*. In order to apply WBS, we need to identify clear project goals, which are given below:

✓ Deliver Project Specifications Report

✓ Build a website for the project

✓ Deliver Analysis Report

- Work on the NVIDIA Libraries and produce a proof of concept, i.e. test the models
- Deliver High-Level Report
- Complete the core photoshopping functions such as cropping, rotating, etc.
- Acquire the remote servers and make them functional
- Integrate the NVIDIA libraries into the project
- Build additional models to handle features like smart-cropping, object identification, etc.
- Produce a working demo on desktop
- Deliver Low-Level Report
- Revise the user interface of the application
- Produce a working demo on Android
- Expand the demo into a working project
- Deliver Final Report
- Optimize the project to ensure a smoother user experience
- Prepare the Project Demo and Final Presentation

Based on these goals, we construct our work packages. Each work package has a leader and requires at least two members to work on it. The overall decomposition is presented in Figure 28, and the assignments to the work packages are given in Table 3.

Work Package	Leader	Working Members	Deadline	Deliverables
1.* (Done)	-	-	Oct. 14, 2019	Project Specification Report
2 (Done)	-	-	Oct. 14, 2019	Autoshop Website
3.1	Mert	Burak, Murathan	Nov. 11, 2019	Analysis Report
3.2	Hikmet	Efe, Burak	Nov. 11, 2019	Analysis Report
3.3	Burak	Murathan, Hikmet	Nov. 11, 2019	Analysis Report
3.4	Murathan	Mert, Hikmet	Nov. 11, 2019	Analysis Report
3.5	Efe	Mert	Nov. 11, 2019	Analysis Report Section
4.1	Mert	Efe, Burak	Dec. 20, 2019	Functional Servers
5.1	Hikmet	Burak	Dec. 31, 2019	High-Level Design Report
5.2	Murathan	Mert	Dec. 31, 2019	High-Level Design Report
5.3	Efe	Burak	Dec. 31, 2019	High-Level Design Report
5.4	Burak	Hikmet	Dec. 31, 2019	High-Level Design Report
5.5	Mert	Efe	Dec. 31, 2019	High-Level Design Report
6.1.*	Burak	Murathan, Efe	Jan. 15, 2020	GitHub Repository with Integrated Software
7.1	Efe	Hikmet, Murathan	Feb. 17, 2020	Low-Level Design Report
7.2	Murathan	Burak, Hikmet	Feb. 17, 2020	Low-Level Design Report
7.3	Mert	Murathan, Burak	Feb. 17, 2020	Low-Level Design Report
7.4	Burak	Efe, Hikmet	Feb. 17, 2020	Low-Level Design Report
8.1	Murathan	Hikmet	Mar. 10, 2020	POC using NVIDIA Libraries
8.2	Hikmet	Burak	Mar. 17, 2020	Desktop GUI
8.3.*	Mert	Murathan, Burak	Mar. 17, 2020	Photoshopping Features
8.4.*	Burak	Efe, Hikmet	Mar. 17, 2020	Server Communication
8.5.*	Efe	Mert, Burak	Mar. 17, 2020	ML Models
8.6	Murathan	Burak	Mar. 17, 2020	Unit Tests for Desktop
9.1	Burak	Mert	Apr. 25, 2020	Android Server Integration
9.2	Murathan	Hikmet	Apr. 25, 2020	Android UI Design
9.3	Efe	Burak	Apr. 25, 2020	Optimized Android App
9.4	Hikmet	Efe	Apr. 25, 2020	Unit Tests for Android
9.5	Mert	Murathan	Apr. 25, 2020	Android Demo
10.1	Burak	Mert, Murathan	May. 8, 2020	Final Report
10.2	Murathan	Hikmet, Efe	May. 8, 2020	Final Report
10.3	Mert	Burak, Hikmet	May. 8, 2020	Final Report
10.4	Hikmet	Efe, Burak	May. 8, 2020	Final Report
11.*	Efe	Mert, Murathan	May. 10, 2020	Final Presentation

Table 3: Work Breakdown Structure

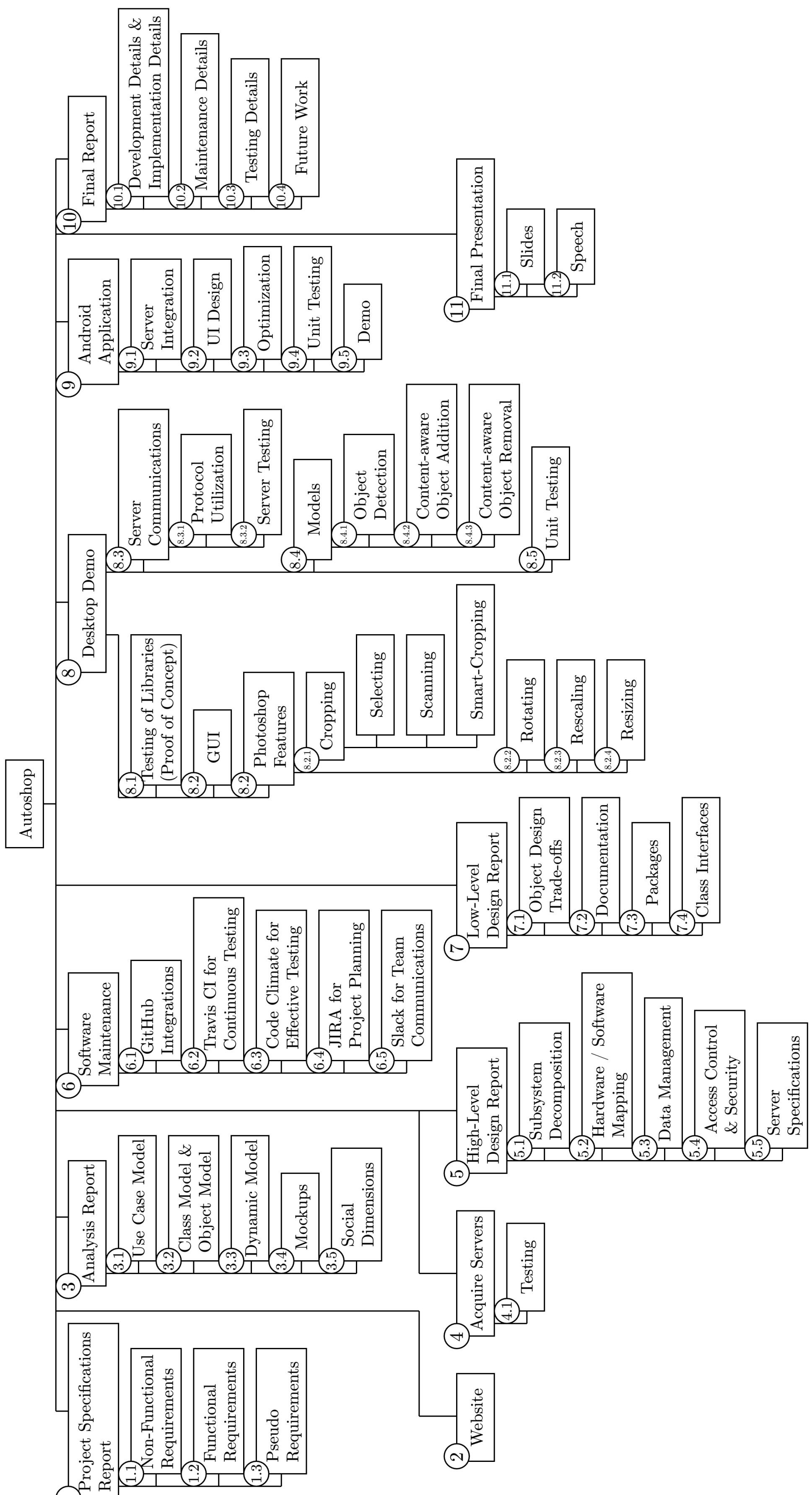


Figure 28: Work Breakdown Structure

A Gantt Chart is useful in visualising the continuous development process. Below is our Gantt Chart that indicates the time rquirements of tasks together with their starting dates and deadlines:

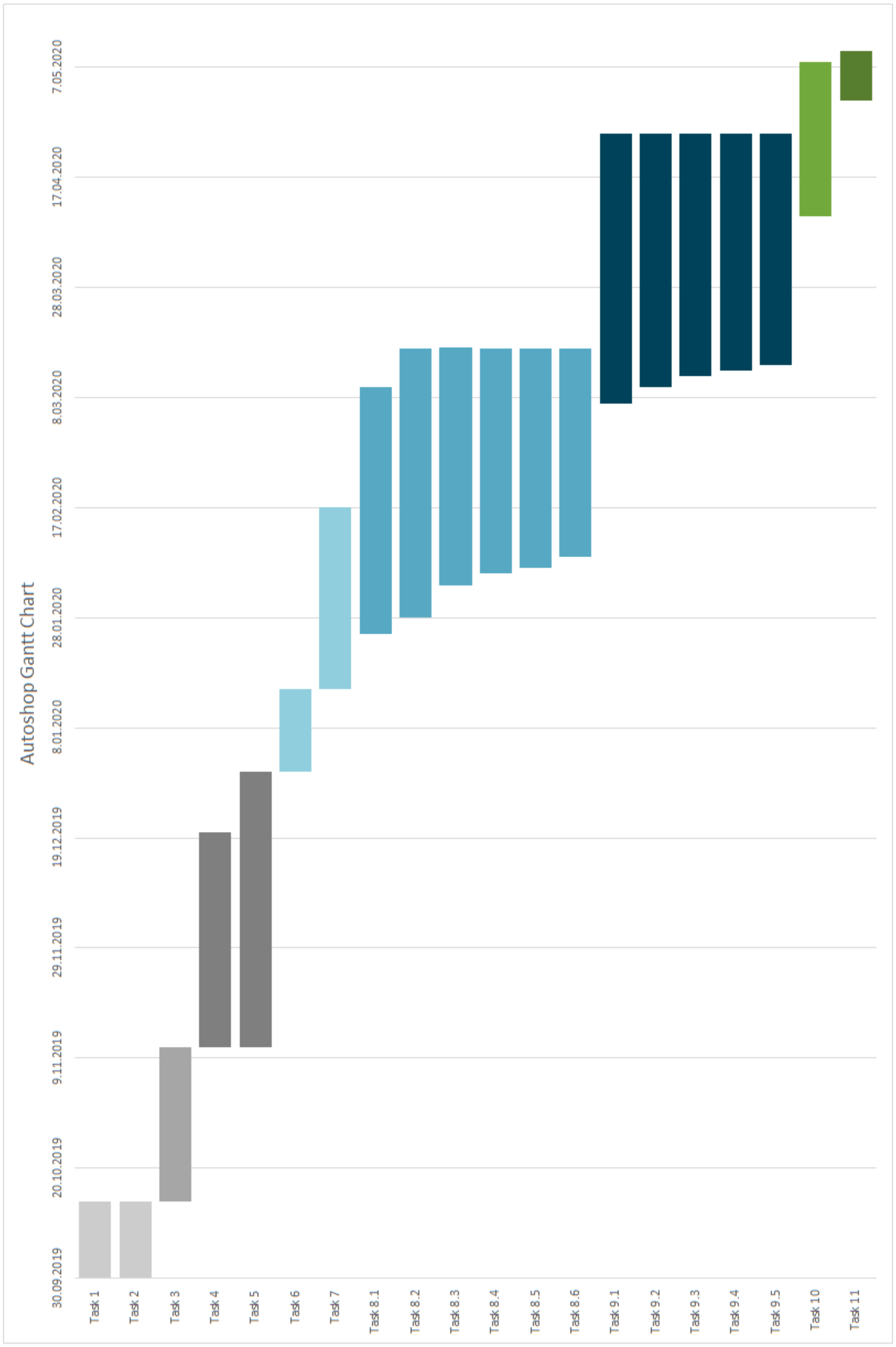


Figure 29: Gantt Chart Visualising the Continuous Development

4.4 Ensuring Proper Team-work

We believe that effective team-work is key to a successful product. In our project, we do not want anyone to be completely responsible from a certain part. Hence we aim to circulate roles and responsibilities so that everyone shares leadership and everyone has an adequate knowledge of the project parts. We postulate a collaborative development environment, where every member is included in all aspects of the project. To ensure and enforce inclusiveness, collaboration and team-work among ourselves we developed certain measures. These measures, in a combined fashion, will act as an indicator of each member's individual collaboration. The measures, or alternatively evidences, of contribution are the following:

- Number of GitHub Commits
- Number of GitHub Issues Resolved
- Rate of Jira Project Leaderships
- Activeness in Jira Projects
- Activeness in Slack Conversations
- Number of Jira Tickets Resolved
- Number of Jira Bugs Solved
- Accuracy to Meet Jira Project Deadlines
- Success to Implement Jira Project Deliverables
- Absence in Real-life Sprint Meetings

We expect everyone to have around the same number of commits and issue resolutions. We also forecast that everyone will be actively speaking in Slack, taking part in Jira projects as leaders and attending real-life meetings. In case certain members fail to contribute as much as others, we will immediately plan meetings where we aim to motivate those particular members under the presence of our supervisor. Again, this is not a case that we expect to happen.

4.5 Ethics and Professional Responsibilities

We begin our discussion on ethics and professional responsibilities with the potential **Global Impact** of our project. The ultimate goal of *Autoshop* is to make the arduous task of photoshop easier and more accessible. Thus, the application will be available to everyone, meaning that it will be downloadable from any application market that agrees to display *Autoshop*. The application will not restrict itself to the use of a certain group of people, country or any social class. As a direct implication of the unrestricted accessibility of *Autoshop* and its goal of making photoshopping easier; we expect the global impact of the application to make photoshopping a regular, effortless task that anyone in the world can perform anywhere, anytime.

Our discussion proceeds with the **Economic** impact and constraints of our project. The photoshop market, namely the graphics market, is currently dominated by Adobe Photoshop, which owns a market share of 57.29% [19]. To estimate the total value of the Graphics market, we look at the value of Adobe. Adobe is reportedly a 95 billion dollar [20] company, so one can estimate the total value of the photoshop market as 170 billion dollars assuming that each percent of the market share possesses the same constant value. *Autoshop* begins its journey as a undergraduate capstone project, hence it will not claim any market share in the near future. In other words, the application will be free to download. In some future time, we may put adds on our application that may potentially generate some amount of profit which in turn lets *Autoshop* to claim a small amount of the market share.

Besides the economic impact mentioned above, there are also economic constraints that affect the development phase. These constraints are basically costs. Some of the costs that we can forecast are:

- AWS Previous Generation GPU Instance Deployment Server Costs:
  \$0.90 per hour $\xrightarrow{\text{100 hours of testing}}$ \$90
- Google App Store Registration Fee: \$25
- GitHub Registration Fee: Free Student Account
- Jira Registration Fee: Free up to 10 people

We can afford these costs, so we do not expect any of them to restrict us in any aspect. Note that we may encounter some unexpected costs that can potentially limit us in certain cases, such as model performance and portability.

Another impact that *Autoshop* has is **Environmental**. There exists a significant correlation between the use of machine learning models and electricity consumption. As data dependent fields such as deep learning and machine learning became more prominent, the need to store and process the data increased. There is a drastic incline in the data centers built, amount of time allocated servers run, and consequently the electricity consumed. According to International Energy Agency's data almost 1% of the global final demand of electricity is the global data center electricity demand [21]. Since *Autoshop* will use NVIDIA's pre-trained generative models the training time required for the servers is relatively low. However, there will still be some amount of training for the additional helper models that we will construct from scratch. *Autoshop* is an application that is completely data dependent, so storage is also a must. Because of the aforementioned model training and data storage requirements there will be an essential negative environmental impact of the application.

The last impact we will touch on is the **Societal Impact** of *Autoshop*. The biggest issue regarding ethics is the issue of deep fakes. Deep fake refers to manipulated photos or videos that yield fabricated images, motions and sounds that appear to be real [22]. The term deep fake combines "deep learning" and "fake", as the name implies, the fake digital representations are powered by sophisticated learning algorithms. Such an algorithm is Generative Adverserial Networks (GANs), which are primarily used to make preliminary fake images look more realistic. The preparation process of a deep fake directly aligns with the photoshopping process in *Autoshop*. To be clearer, the application operates on a preliminary input photo where the user indicates which objects to add or remove, and then proceeds by making this input more believable by benefiting from NVIDIA's GANs and other helper models. As a consequence, *Autoshop* imposes a serious risk of becoming a deep fake generator. The danger, or the ethical issue, here is that deep fakes can be used to make people believe things that are not actually real. For instance, using a deep fake one may decrease the number of audiences in the photo of a crowded concert and depict the concert as if it was unsuccessful, or one can use deep fakes to alternate the facial reactions of people, and so on. To avoid deep fake generation, we may consider adding a small watermark or some other way digitally signing the photoshopped photos so that it becomes a lot simpler to detect them.

Another significant societal issue that we should certainly safeguard is data privacy. *Autoshop* is designed to conform to the General Data Privacy Regulation (GDPR) [16]. Hence the application will never keep user data in the remote servers without explicitly asking for permission. The input photo that the user had prepared on his/her local machine will be transferred to the processing servers, after the processing is done the application will delete the photo if the user had denied permission. In case of an interrupt in the server communication, the servers will roll-back to the initial state where no input photo is received yet. This way we will ensure that no data is kept without permission.

4.6 New Knowledge and Learning Strategies

Strictly speaking, our current knowledge is inadequate to build *Autoshop* right away. Thus, we need to acquire some knowledge, develop new skills and plan appropriate learning strategies to do so. Some topics that we should center upon are:

- Advanced Image Processing
- Deep Learning
- Android Development
- Application Design
- Software Development Planning

Automating the process of photoshopping requires us to build state of the art machine learning models, which may not be covered in Bilkent's courses. Hence, we need to gain in-depth knowledge on deep learning. As photoshopping deals with images, we naturally need to learn about advanced image processing algorithms. As a consequence of our choice of deployment platform we need to have a good understanding of android development and application design. Finally, to meet the deliverables and deadlines we should establish a solid understanding of software development planning.

Below are some methods that we will apply to learn more on the topics above:

- Literature Review
- Online Learning
- Hands-on Experience

Literature review will be beneficial for us to discover cutting edge deep learning architectures and their working principles, same for advanced image processing algorithms. For us, literature review includes a wide range of research from surfing in Stack Overflow to reading ACM papers. Online learning will be necessary, since there are a lot of online tutorials that teach how to write Android programs. Watching these tutorials will likely accelerate the development process. Lastly, hands-on experience is a must for us since we believe that the best teacher for us is our own mistakes.

5 Glossary

`FastPhotoStyle`	An NVIDIA library that is available in `Python`. Given a content photo and a style photo, the library can adjust the content photo according to the style of the style photo [6].
`ImageInpainting`	An NVIDIA library available in `Python`. The library can remove parts of the image and replace the removed parts with realistic artificial parts [2].
WBS	Work Breakdown Structure. A project planning paradigm that recursively decomposes the project into smaller packages.
Work Package	The lowest level packages in the WBS.
GDPR	General Data Privacy Regulation. A rule set concerning the protection of personnel data.
GPU	Graphics Processing Unit. A dedicated electronic circuitry designed to efficiently manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device [23]. GPUs are useful for deep learning applications since they are well-suited for frequently used operations such as matrix-multiplication and convolution.
Neural Networks	A set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns [24].
GAN	Generative Adversarial Networks. A model consisting of a generator and a discriminator, which aims to create new data instances that resemble your training data [25].
Travis CI	A hosted continuous integration service used to build and test software projects hosted at GitHub [7].
Code Climate	Automated code review software for technical debt and test coverage [8].
Grafana	An open-source analytics-monitoring software for a database [10].
Jira	A software that helps manage agile and software development projects.
Slack	Cloud-based instant messaging platform specifically designed for professionalized team communication.
AWS	Amazon Web Services. Reliable, scalable, and inexpensive cloud computing services provided by Amazon. Remote servers can be rented through this service [26].

Agile Development	Agile software development refers to software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams [27].
Sprint	Sprint is a unit iteration of the continuous software development cycle. Within a Sprint, planned amount of work has to be completed by the team and made ready for review [28].
Deep Fake	Manipulated photos or videos that yield fabricated images, motions and sounds that appear to be real [22]. The term deep fake combines "deep learning" and "fake". The fake digital representations are powered by sophisticated learning algorithms.
Git	Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency [29].
GitHub	An online platform that provides hosting for software development version control using Git.
UML	Unified Modelling Language. A standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems [30].

References

[1] D. W. Stout, Claire, J. Lyles, James, A. Sarkar, Barbora, Kyle, Betty, A. Brown, Robison, and et al., "Social media statistics: Top social networks by popularity," Jul 2019.

[2] "Remove unwanted objects." `https://online.theinpaint.com/`. [Accessed: 3- Nov- 2019].

[3] "New and enhanced features: Latest release of photoshop." `https://helpx.adobe.com/photoshop/using/whats-new.html`. [Accessed: 3- Nov- 2019].

[4] "Adobe pricing photoshop." `https://www.adobe.com/tr/creativecloud/plans.html`. [Accessed: 3- Nov- 2019].

[5] "Functional requirements vs non functional requirements: Key differences." url=https://www.guru99.com/functional-vs-non-functional-requirements.html. [Accessed: 3- Nov- 2019].

[6] Nvidia, "Nvidia/fastphotostyle." `https://github.com/NVIDIA/FastPhotoStyle`, Feb 2019. [Accessed: 3- Nov- 2019].

[7] "Test and deploy." `https://travis-ci.org/`. [Accessed: 3- Nov- 2019].

[8] "Engineering metrics to improve continuous delivery practices: Velocity." `https://codeclimate.com/`. [Accessed: 3- Nov- 2019].

[9] "Download android studio and sdk tools: Android developers." `https://developer.android.com/studio/`. [Accessed: 3- Nov- 2019].

[10] "Grafana: The open observability platform." `https://grafana.com/`. [Accessed: 3- Nov- 2019].

[11] M. Rouse and M. Rouse, "What is use case?." `https://searchsoftwarequality.techtarget.com/definition/use-case`. [Accessed: 3- Nov- 2019].

[12] "Uml - class diagram." `https://www.tutorialspoint.com/uml/uml_class_diagram.htm`. [Accessed: 3- Nov- 2019].

[13] "Sequence diagram." `https://en.wikipedia.org/wiki/Sequence_diagram`, Oct 2019. [Accessed: 3- Nov- 2019].

[14] "Uml - activity diagrams." `https://www.tutorialspoint.com/uml/uml_activity_diagram.htm`. [Accessed: 3- Nov- 2019].

[15] "Social media and psychology." https://www.allpsychologyschools.com/psychology/social-media-psychology/. [Accessed: 6- Nov- 2019].

[16] "General data privacy regulation." https://eugdpr.org/. [Accessed: 6- Nov- 2019].

[17] "Google play terms of service." https://play.google.com/about/play-terms/index.html. [Accessed: 6- Nov- 2019].

[18] "Bus factor." https://en.wikipedia.org/wiki/Bus_factor. [Accessed: 3- Nov- 2019].

[19] "Graphics market share report." https://www.datanyze.com/market-share/graphics. [Accessed: 3- Nov- 2019].

[20] "Adobe's value." https://producthabits.com/adobe-95-billion-saas-company/. [Accessed: 3- Nov- 2019].

[21] "Data center electricity consumption." https://www.iea.org/tcep/buildings/datacentres/. [Accessed: 3- Nov- 2019].

[22] "Danger in deep fakes." https://www.cnbc.com/2019/10/14/what-is-deepfake-and-how-it-might-be-dangerous.html. [Accessed: 3- Nov- 2019].

[23] "Gpu." https://en.wikipedia.org/wiki/Graphics_processing_unit. [Accessed: 8- Nov- 2019].

[24] "Neural networks." https://skymind.ai/wiki/neural-network. [Accessed: 8- Nov- 2019].

[25] "Generative adversarial netorks." https://developers.google.com/machine-learning/gan. [Accessed: 8- Nov- 2019].

[26] "Amazon web services." https://aws.amazon.com/. [Accessed: 8- Nov- 2019].

[27] "Agile development." https://www.cprime.com/resources/what-is-agile-what-is-scrum/. [Accessed: 8- Nov- 2019].

[28] "What is sprint?." https://www.cprime.com/resources/what-is-agile-what-is-scrum/. [Accessed: 8- Nov- 2019].

[29] "Git version control." https://git-scm.com/. [Accessed: 8- Nov- 2019].

[30] "Unified modelling language." https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/. [Accessed: 10- Nov- 2019].