Bilkent University

Department of Computer Engineering

SENIOR DESIGN PROJECT

Photonom

High Level Design Report

Group Members	Omer Faruk Bademez	21601216
	Idil Hanhan	21601289
	Beyza Kalkanli	21600944
	Berfin Kucuk	21502396
	Kerem Yilmaz	21601223
Supervisor	Asst. Prof. Hamdi Dibeklioğlu	
Jury Members	Prof. Ozcan Ozturk	
	Assoc. Prof. Selim Aksoy	
Innovation Expert	Mehmet Surav	

High Level Design Report
Dec 31, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction

Taking photos have turned into a daily practice, whether we are visiting a new country, celebrating a birthday or just having a normal day. In fact it has been estimated that more than 100 million photos and videos are uploaded to Instagram everyday [1]. Even though this is an impressive number, it does not even include the number of photos that people take for each upload. For some this number can be 20 and for other it might go up to 200 [2].
There are many reasons why one would have to take a lot of photos to get one shot they like. The light does not look right in one of them, someone walked behind them in another one, the building they want to be on the background did not fit the frame, they look too serious in that one and so on. Taking more photos can be a possible solution, but in most cases you might realise these problems when its too late. Going through your vacation photos to realise that picture you took in front of a historical church is missing the tip of the church is frustrating. Even if you do realise these problems when you have the chance to take more photos, some things cannot be changed. If it is raining and water droplets are in your lens, it is impossible to change the weather. And even if the problem is something you can change, why waste time taking tons of photos when it is possible for you to enhance or modify the ones you already have.
There are multiple tools that one can use to alter their photos. Advanced photoshop tools are often too complicated for occasional use. It is possible to find a tool that will remove a background item and then another one to alter your head position and/or facial expression and then find a filter in another app that reflects your style. In this case, it is not common to find all these features in one place and the users have to jump from one app to another to get what they want.
With Photonom, we are proposing a mobile application that will allow users to modify their photos interactively and in one place whilst conserving the true nature of the image itself. In the next section, Photonom will be described in more detail and constraints related to it will be explained. Then both the functional and nonfunctional requirements of Photonom will be explained. The references can be seen at the end of this report.

1.1 Purpose of the System

Photonom is an interactive photo editing tool. Photonom will include editing tools such as Object Removal, Image Stitching, Rain Removal, Style Transfer, Image Quality Evaluation, Face Expression and Head Position Modifier. These tools aim to allow the user to enhance

the details of their photos without sacrificing authenticity. Each of these tools will contain opportunities for the user to give their input, alter and change how the tool is being used. For example, the user will decide how much their facial expression will change, or during object removal, after the removal is done Photonom will ask it's user whether they are satisfied and if not will do the removal again in order to create a better result.

1.2 Design Goals

The main goal of Photonom is to successfully and interactively edit the photos given by the user with the tool that they have specified. The goal for each specific tool is to successfully edit the photo according to the specified purpose of the tool whilst conserving the true nature of the image.

1.2.1 Usability

Photonom is a mobile app and naturally will have a learning curve. One of our design goals is minimising the learning time as much as possible. Photonom includes many functionalities which potentially may detriment the usability of the application. In order to avoid this, the interface of Photonom should be as simple as possible and include only the necessary structures. Usability will be further supported with welcome tutorials which explain how the application is used.

1.2.2 Accessibility

Photonom should be accessible to everyone around the world. In order to ensure this, English will be the default language of the application and other languages will also be added later on according to demand.

1.2.3 Availability

Photonom aims to provide many functionalities to its users and will be created to have a modular easily-pluggable design. We also aim to ensure that any update to Photonom do not affect the availability of the application. In order to provide high availability but not spend too much monetary resources we will have two servers and a simple orchestration tool supported cluster with 3 containers. Here is a simple flow for our release mechanism:

1. When there will be a new release for any of the modules, the load will be balanced between two containers instead of three.

2. The separated cluster will be deployed with the new changes and will be made sure its alive and well.

3. The load will be shifted to the newly deployed cluster.

4. After all previous jobs are completed in the previous two machines they will be deployed with the updated changes and they will be health checked.

5. The load will be separated between all clusters again.

1.2.4 Scalability

The application is aimed to provide its service globally. Therefore it should scale up as usage increases. We aim to design a dockerized server which will be easily deployed [3]. This way the system can scale itself up in case of high loads automatically by the use of cloud services such as AWS, GCP [4] [5].

1.2.5 Portability

We are aiming Photonom to be a minimum viable product (MVP) and we want to ensure our application can be used with various operating systems such as iOS, Android, or Windows Phone. Therefore, we will be using Flutter by Google which is a cross-platform framework that would ensure Photonom is easily portable.

1.2.6 Performance

Each action user wants to perform should not take longer than 3 seconds. Transition between screens and response to user functions should not take longer than a second.

1.2.7 Security

The personal data of the users will persist on the server during the operations. We will ensure our server is secure for the sake of our user's privacy. In case there is an intrusion, all operations will be cancelled and the corresponding container will be quarantined.

1.2.8 Backup

With the users permission, different versions of their photograph could be stored in their preferred cloud storage service such as Google Photos at different stages of the performed operations.

1.2.9 Maintainability

The system is going to have a modular design, hence it should not be highly coupled. Moreover, there will be a defined code standard along with additional automated checks regarding the new code changes. There will be a reviewer to each pull request to increase the maintainability of the code in the future changes.

1.3 Definitions, acronyms, and abbreviations

- Object Removal: Tool of Photonom which can be used to remove an object from a photo.
- Image Stitching: Tool of Photonom which can be used to stitch multiple photos together to make one photo.
- Face Expression Modifier: Tool of Photonom which can be used to modify the face expression of a recognised face in the photo.
- Image Quality Evaluation: Tool of Photonom that evaluates selected photo(s) according to both aesthetics and technical details.
- Style Transfer: Tool of Photonom that modifies a photo to match the style of a provided style photo.
- Rain Removal: Tool of Photonom that removes the effects of rain from a photo.
- Head Position Modifier: Tool of Photonom which can be used to modify the head position of a recognised face in the photo.
- Server: The system which processes the images sent to it according to specified tool.
- OpenFace: Python and Torch implementation of face recognition with deep neural networks. It is used for face detection, face tracking and extraction of action units. [6]
- PyTorch: An open source machine learning library.
- GAN: Generative Adversarial Network. These networks create new instances based on the training data.[7]

1.4 Overview

Photonom is a mobile application that provides its users different tools to edit their photos interactively. The tools of Photonom aim to enhance the user's photo whilst ensuring the true nature of the photo does not change. The users will be able to upload their photos to Photonom or take photos using the application. The tools provide the following functionalities; object removal, rain removal, theme transfer, image stitching, changing face expression, image quality evaluation and head pose modifier. Whilst the user is editing their photo, each change they make will be saved and the user will be able to revert their actions and see the entire history of their editing and go back to any state they want. After their editing is done they will be able to share their final product.

1.4.1 Photo Selection

In order to use the functionalities of Photonom, the user first has to select the photo(s) they want to edit or evaluate. There are multiple ways for a user to make this selection. The first one is the case where the user uploads a photo they already have in their device's gallery to Photonom. The second case is where the user uses Photonom to take photos. Additionally the user will also be able to continue from a photo they previously uploaded to Photonom. The choice for photo selection will be made by the user when they first open the application.

1.4.2 Editing Tools

Photonom provides many editing tools to it's users and in this section these tools will be introduced. They will then be explained in detail in section 2.2.

1.4.2.1 Remove Objects

User will use Remove Objects tool when they want to remove an object from their photos. After the user specifies the objects they want to remove and this operation is performed, the user will be able to modify the end result.

1.4.2.2 Rain Removal

The purpose of Rain Removal is to offer a tool that allows the users to remove the effect of rain from their photos such as the rain droplets on the camera lens.

1.4.2.3 Theme Transfer

The Theme Transfer will be used when the user has a sample theme that they want to apply

to their photo. Once this tool is selected and the user provides a photo that represents the theme they want to adapt, the photo they selected for editing will be changed to fit this theme.

1.4.2.4 Image Stitching

User will use Image Stitching tool when their photo includes missing sections. The user will provide the photos to stitch and they will be placed by the system. The user will be able to determine the limits of the photo by cropping the stitched images. Then they will be given a final photo where the missing parts are completed.

1.4.2.5 Face Expression Modifier

The purpose of Face Expression Modifier is to allow the user to make slight changes to their facial expression according to their desire.The user will be able to choose the face they want to change and slightly change their facial expression by, for example, making them smile more.

1.4.2.6 Image Quality Evaluation

Using Photonom the user will also be able to evaluate the quality of a selection of photos. When the users select multiple photos, these photos will be evaluated and each will be given a score representing their quality. One of these photos can be chosen for editing.

1.4.2.7 Head Pose Modifier

This is a tool that will only be implemented if time permits. The purpose of Head Pose Modifier is to offer a tool that allows users to slightly change the position of their heads.

1.4.3 Undo and Redo

Whilst the user is editing their photo, it is possible that they will make a move that they regret and want to undo. Photonom will include an Undo option that can be used when the user wants to undo the most recent modification they made. Photonom will also include a Redo option which will allow the user to reapply the most recent modification they made.

1.4.4 History

In Photonom, the user will be able to see the entire history of their editing process. In here the user will see all previous states of the photo with the quality score of that state and will be able to go back and continue from any point in history.

1.4.5 Saving & Sharing the Edited Photo

The user will be able to save the state of their photo so that they can continue editing later on. When they finish their editing, they will be able to save their photo to their device's gallery and/or share their final product.

2 Proposed Software Architecture

2.1 Overview

This section explains the structure of Photonom in detail. In the following subsection, the subsystems, how we manage data, accessibility, control flow and boundaries will be explained in order.

2.2 Subsystem Decomposition

Photonom is created with the Client-Server architecture. In this architecture the system is divided into two parts. Client subsystem will be a mobile application which will provide the interactions between the user and the program. This part will send the photos and necessary information to the server. Server subsystem will get what Client sent, and apply the models to the photos received.

Client subsystem has two parts, presentation and controller. Presentation is where the subsystem provides the user interface. There will be four main view and these will also be separated when tools are considered. These views are explained in section three. Controller part of the client subsystem handles the control of the logic behind the application. The users can use camera, open gallery, save and share photos thanks to controller. The client subsystem will be implemented with Flutter which is an open-source UI software development kit used to develop applications for Android, iOS and etc.

Server subsystem handles the image operations that the application provides. It has Modules subsubsystem which includes all the operators for Photonom's editing tools. After it gets the photo(s) from the client subsystem, it uses the module that will is selected by the user on the client side. After it apply the operation it returns the result to the client side to display the result to the user. Server subsystem is implemented with Python using Django. Also, the modules inside this subsystem are trained with PyTorch or TensorFlow.

These subsystems are explained in section three in detail.

2.3 Hardware/Software Mapping

We are implementing the server modules in Python. The communication between the client and server is provided by REST API. HTTP request are made after the actions of users on their mobile phones. The client can be run on Android and iOS operating systems. The interaction between the server and the device can be seen in the deployment diagram below.

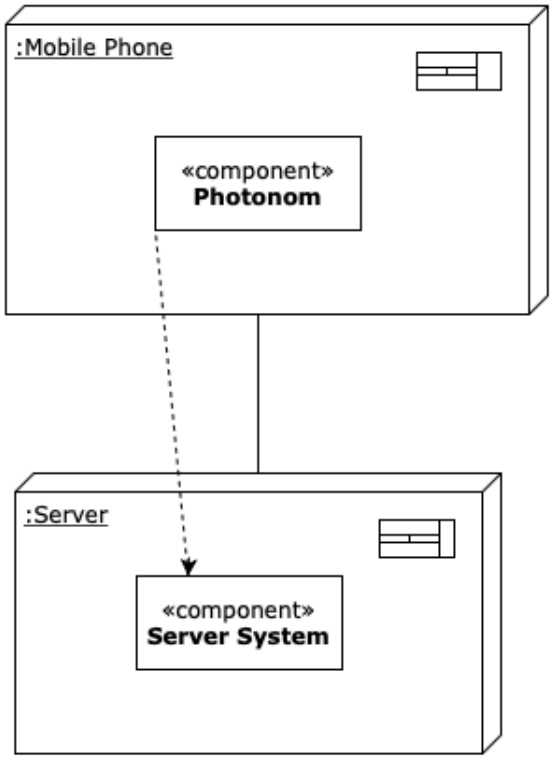

Figure 1: Deployment Diagram

Hardware systems that are used (mobile phones) need to have a camera if user wants to take photos on our application. Other than that, the phone needs to have a speaker for the sound effects of the application.

2.4 Persistent Data Management

Since Photonom is an interactive photo editing tool, the only data that we keep is photos of the users. We are keeping these data locally, because we believe that it is the most reliable

way. Hence those that users edit will be photos from their own photos, we consider saving photos on their own device appropriate. Users will also be able to save photos to their gallery.

2.5 Access Control and Security

The only photos that users can access is their own photos. Therefore, their permission will be necessary to access their gallery or camera. To keep their photos safe, all of the edited photos will be kept locally, on the mobile phone of the user.
Also, we will keep the models and datasets which are sensitive data. Server will be unapproachable to any user that will use the application. Therefore, any code that provides the access to server will be unreachable.

2.6 Global Software Control

The control flow mechanism of Photonom is event-driven control. The flow of the application is dependant on the actions done by the user which are interpreted as events by the application, so the software control resides in the user. If the actions done requires the server to generate an output, the application sends the necessary information and photo to the server, servers run the necessary model and send the output photo back to the application.

2.7 Boundary Conditions

2.7.1 Start-Up

After downloading, the user can start the application by clicking the application icon on the mobile phone homepage.

2.7.2 Shutdown

The user can exit the application by using the exit method of their mobile phones.

2.7.3 Errors

We will try to catch as many exceptions as possible, by doing tests during and after the implementation and updating our code accordingly. In case any exception is missed and the application crashes, we will ensure that the user can continue the editing from the point that application crashed, with exact tools used. We believe that such practice will make our application robust and more usable. Each time user edits the photo using any of the tools,

this feature will be provided. We will also show all the states of the photo to the user in case, s/he wants to go back to that state and continue to edit from there.

3 Subsystem Services

Photonom is formed by the interaction of two systems: the client and the server.

3.1 Client

The client can be used through devices that has Android or IOS operating system. The client is the user interface layer of Photonom, where the user interact with his/her photos and edit them. The client contains the presentation and controller subsystems. The presentation subsystem contains user interface elements for the user to interact with. The controller subsystem is responsible for creating appropriate requests depending on the events initiated by the views and then communicating with the server to send the requests.

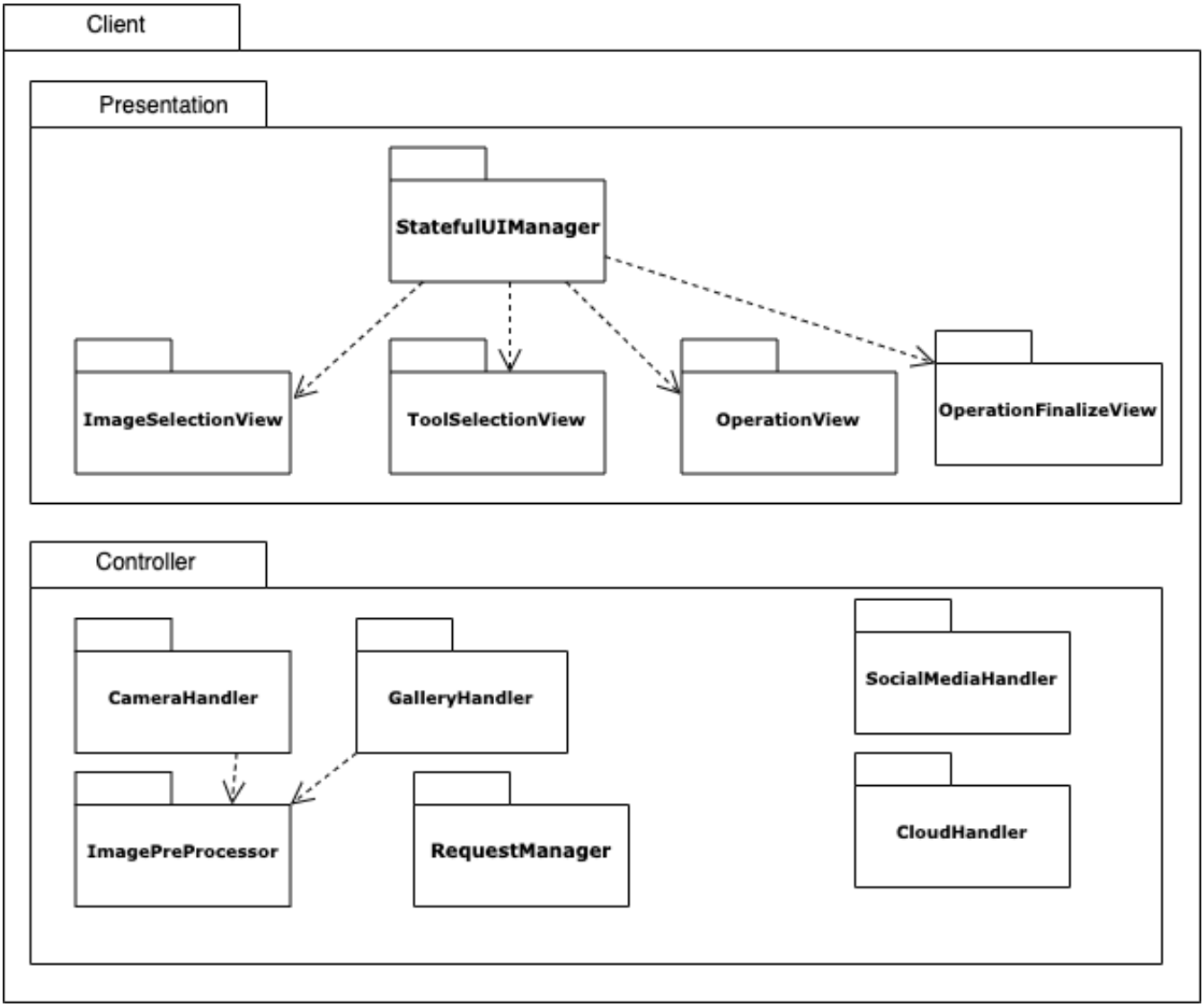

Figure 2: The hierarchy of modules inside the client system

3.1.1 Presentation

The presentation layer contains views and their event firing mechanisms.

StatefulUIManager: A global manager to mediate other view classes.

ImageSelectionView: The view class for the image selection screen. Image or images are selected before asking for request creation. Uses CameraHandler or GalleryHandler to select image(s).

ToolSelectionView: The view class for the tool selection screen. Uses RequestManager to make tool selection requests.

OperationView: The view class for the editing screen. It allows interactions with the image for editing. It can launch the camera or the gallery to select more images which will be used in the editing process.

OperationFinalizeView: The view class for the share screen. Uses SocialMediaHandler and CloudHandler to finalize editing.

3.1.2 Controller

The logic of the client executes operations like image pre-processing and server communication, depending on the events created by the views on presentation layer.

CameraHandler: The handler responsible for configuring the camera in IOS and Android apps.

GalleryHandler: The handler responsible for selecting the image in IOS and Android apps from the local gallery.

ImagePreProcessor: Responsible for handling image structure before sending to the server such as resizing the image according to the selected operation.

SocialMediaHandler: Handles share of edited images on social media such as Instagram, Whatsapp, Facebook ,etc.

CloudHandler: Handles share of edited images on the cloud selected by the user such as Google Drive and DropBox.

RequestManager: Responsible for creating all the requests depending on the events called by the presenation layer and communicates with the server. Waits for the server response and notifies views of the result.

3.2 Server

All the processing happens in the server. An image with the required operations on that image arrive as a request to the server. Then, the server processes the image with the given modules and creates an output. The output is then sent back to the client as a response. The server consists RequestHandler, ImageOperator and the Modules layer. The Modules layer consists of 7 image operations such as image stitching and sytle transfer, etc.

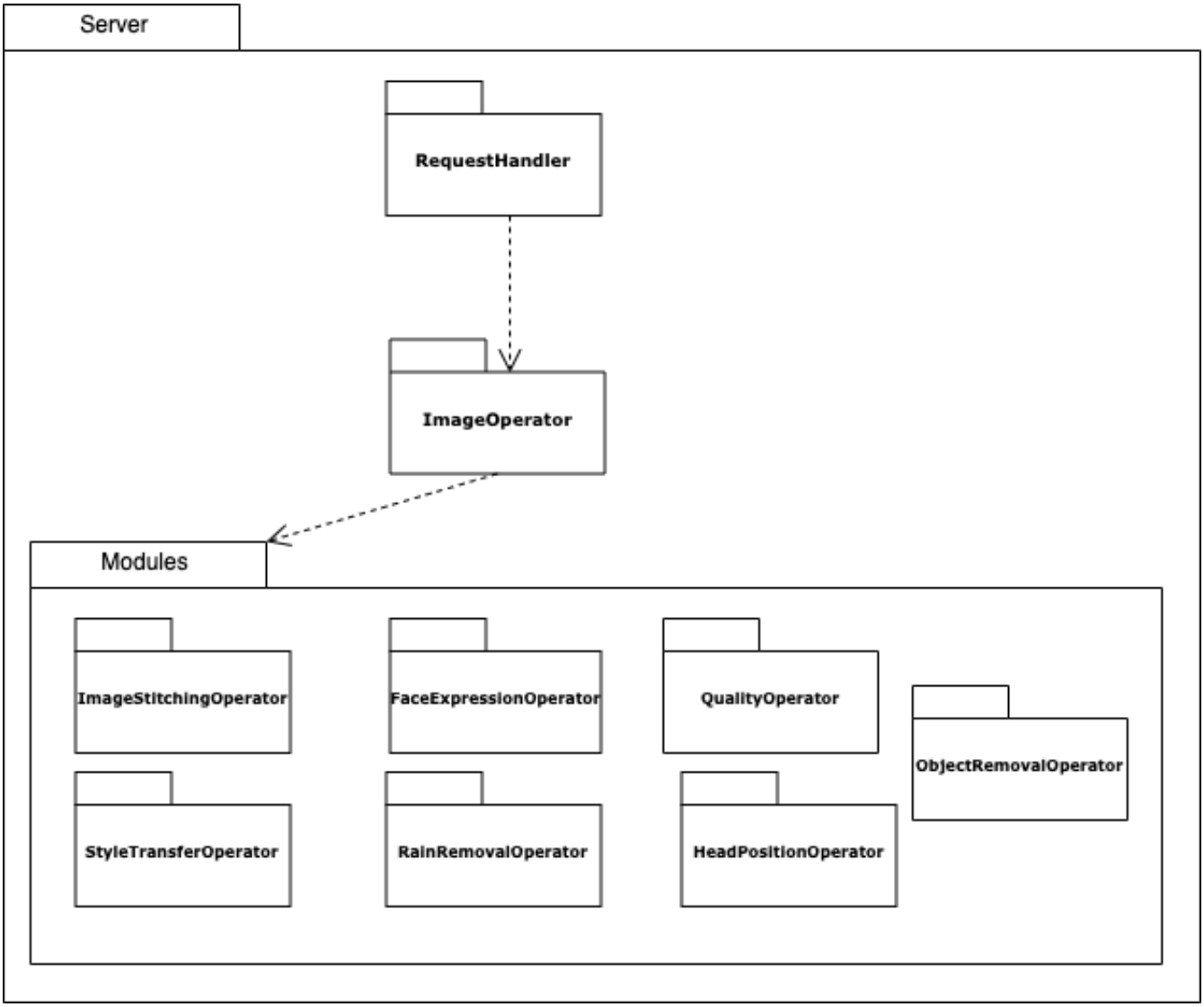

Figure 3: The hierarchy of modules inside the server system

RequestHandler: Handles client request by turning them into tasks and runs them on ImageOperator. Creates responses and sends them back to the client.

3.2.1 Modules

Module package contains the module used to edit the image(s).

ImageStitchingOperator: Stitches selected images into an image.

FaceExpressionOperator: Changes the expression of the selected face in the image.

QualityOperator: Evaluates the quality of the selected images.

StyleTransferOperator: Transfers the style of the secondly selected image to the first image and returns a new image.

RainRemovalOperator: Removes rain drops from the image and returns the output image.

HeadPositionOperator: Modifies the head position of the selected face in the image.

ObjectRemovalOperator: Removes the selected object from the image and fills the background accordingly.

4 New Knowledge Acquired and Learning Strategies Used

During the implementation of the project, we followed the strategies we discussed before. Since we have both research and development work packages, the strategies are designed accordingly and given below.

For each research component of the project, we started with a literature review to have an insight into the problem. Especially, the articles with open-source codes were beneficial for us during the learning process. For the implementation of some of the functionalities, we utilized the articles we perused before. We did not use the open-source projects only for the implementation of the neural networks directly related to the functionalities but also for the pre-processing parts. We learned how to use the OpenFace library to make our data compatible with the models we need. We acquainted ourselves with Generative Adversarial Networks, GAN, to implement the face modification tools. For the implementation of the

models, some of us acquired and some of us improved the skill of using different frameworks, specifically PyTorch, where the open-source projects were useful to get hands-on experience in the first place. In addition to using deep learning models to implement the functionalities, we also used the knowledge we acquired in class. By that way, we had a chance of applying our knowledge to the problems we encounter during the implementation of the application. We used the feature matching methods we learned in the Image Analysis course for the image stitching property.

Since we don't have any prior experience related to app development with Flutter, we needed to check the existing tutorials first. We tried to keep the time required for the tutorials as short as possible considering the fact that the best way to learn is making practice. Instead of watching tutorials, we mostly check the codes and tried to understand the working principles of this framework. After grasping the logic of the framework, we started to develop our application. Since the base of the application is taking photographs with a camera or choosing them from the gallery, our first strategy in the development package was implementing these functionalities then continuing with the integration of the other packages.

We decided to make the research and development packages work in parallel to avoid the risk of having problems while combining packages in the end. After the first step of the development package, we were able to integrate the research components we managed to implement into our application. To make sure that the process is being done as it is supposed to be, we started with the dummy operations. After this step, we started to add the functionalities one by one. Since this part requires the use of the server, we also created an environment that includes the needed libraries to run the codes with the server.

5 Glossary

Photonom: A word derived from the merging of two words: photoshop and autonom.

References

[1] "Instagram by the numbers (2019): Stats, demographics fun facts," 2019. [Online]. Available: https://www.omnicoreagency.com/instagram-statistics/

[2] J. Brucculieri, "4 instagrammers show us how many photos they took before nailing 'the shot'," 2018. [Online]. Available: https://www.huffpost.com/entry/instagramphoto-camerarolls_n_5ac4ed48e4b063ce2e58131f

[3] 2019. [Online]. Available: https://www.docker.com/

[4] 2019. [Online]. Available: https://aws.amazon.com/

[5] 2019. [Online]. Available: https://cloud.google.com/

[6] "Openface," 2019. [Online]. Available: https://cmusatyalab.github.io/openface/

[7] "Introduction | generative adversarial networks," 2019. [Online]. Available: https://developers.google.com/machine-learning/gan