
Bilkent University

Department of Computer Engineering

# Senior Design Project

Autoshop

# High Level Design Report

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

**Supervisor** : Fazlı Can
**Jury Members** : Cevdet Aykanat, Ercüment Çiçek
**Innovation Expert** : Duygu Gözde Tümer

**Website** : https://beastunited.github.io/

High Level Design Report
Dec 31, 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. 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 aim to provide an overview of the architecture and design of our system. First of all, the purpose of the system and the design goals of Autoshop are described. Then, the existing systems, their qualities, and architecture are specified. Afterwards, a detailed analysis of the structure of our project is presented. The subsystem decomposition, architectural plans of subsystems, and hardware/software mapping of these components are illustrated. Design decisions such as persistent data management,

access control and security, boundary conditions are reported in detail. Then, the functions of subsystem services and their interactions are outlined. Finally, the knowledge we have acquired using the learning strategies we had planned is discussed.

## 1.1 Purpose of the System

There are several purposes of our system, which can be stated under two separate headings, non-technical and technical. Since we are aiming to construct a software that is both innovative and technically high quality, it is inevitable that stating a single purpose will fall short explaining our objective.

First of all, the non-technical purpose of our system is to make the task of photoshopping easier and more accessible. Photoshopping is considered a time-consuming and arduous task, our system will automatize this process using state-of-the-art AI tools and technologies. The system will provide a user-friendly and self-explanatory interface that instructs its users on how to crop, drag, drop and erase parts of the images they want to edit, and that will be the only effort put in by the users. In the end, the need for professional photoshoppers will be replaced by the need to download *Autoshop*.

Second, the technical purpose of our system is to function in a scalable, error-prone, secure and efficient manner. The system aims to be scalable in order to sustain its performance and continue serving its users even if the number of users grows drastically. It aims to be error-prone to ensure that there does not exist any time interval that the application is out of service. Another objective of our system is to be secure to prevent any single one of our users from feeling distrusted and to keep their data safe. Lastly, the system aims to be efficient to provide a smooth user experience.

## 1.2 Design Goals

In this section, Autoshop's design goals will be presented and important user experience aspects will be discussed.

### 1.2.1 Extensibility

The system should:

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

### 1.2.2 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 [2].

### 1.2.3 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.

### 1.2.4 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.

### 1.2.5 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).

### 1.2.6 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.

## 1.3 Definitions, Acronyms, and Abbreviations

| | |
|---|---|
| ACM | Association for Computing Machinery. World's largest educational and scientific society, uniting computing educators, researchers and professionals to inspire dialogue, share resources and address the field's challenges. |
| AWS | Amazon Web Services. Reliable, scalable, and inexpensive cloud computing services provided by Amazon. Remote servers can be rented through this service [3]. |
| GDPR | General Data Privacy Regulation. A rule set concerning the protection of personnel data. |
| GUI | Graphical User Interface. An interface through which a user interacts with electronic devices such as computers, hand-held devices and other appliances. |
| HTTP | Hypertext Transfer Protocol. HTTP is the protocol used to transfer data over the web. It is part of the Internet protocol suite and defines commands and services used for transmitting webpage data. |
| SDK | Software Development Kit. An SDK is a collection of software used for developing applications for a specific device or operating system. |
| UI | User Interface. The user interface (UI) is the point of human-computer interaction and communication in a device. |
| 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 [4]. |

### 1.4 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 photographic 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.

## 2 Current Software Architecture

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 [5], 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 [6], albeit, these tools require advanced photoshop knowledge to perform addition and removal of objects. On top of that, Adobe products are relatively expensive [7] and not fully-functional on mobile platforms; which implies that these products have serious accessibility issues.

## 3 Proposed Software Architecture

Autoshop's architecture is composed of many subsystems. These subsystems are vital to the main system's success and must work in harmony with each other. In this section,

Autoshop's subsystems will be displayed. The interactions between the subsystems, the classes they consist of and their functions will be discussed.

## 3.1 Overview

Autoshop's system decomposition aims to present the structures of its systems and subsystems in great detail. Before getting into the diagrams, it is important to realize the importance of a well-defined subsystem structure for a project. The subsystem decomposition showcases how each of our chosen subsystems (explained in greater detail in the following sections) interacts with each other. From the diagrams, it is possible to see the responsibilities of each subsystem, which information it receives, sends or calculates, and how the transition between different subsystems takes place. To get these traits of the subsystems correct before starting the implementation translates to fewer errors during coding, and less revision required.

For these reasons, we spent a lot of time and effort in identifying the different subsystems that makeup Autoshop when combined. We focused our resources on this aspect for we know this is the one with the greatest return in the long run. After realizing the importance of a proper subsystem decomposition, we are ready to dive deeper into the subject.

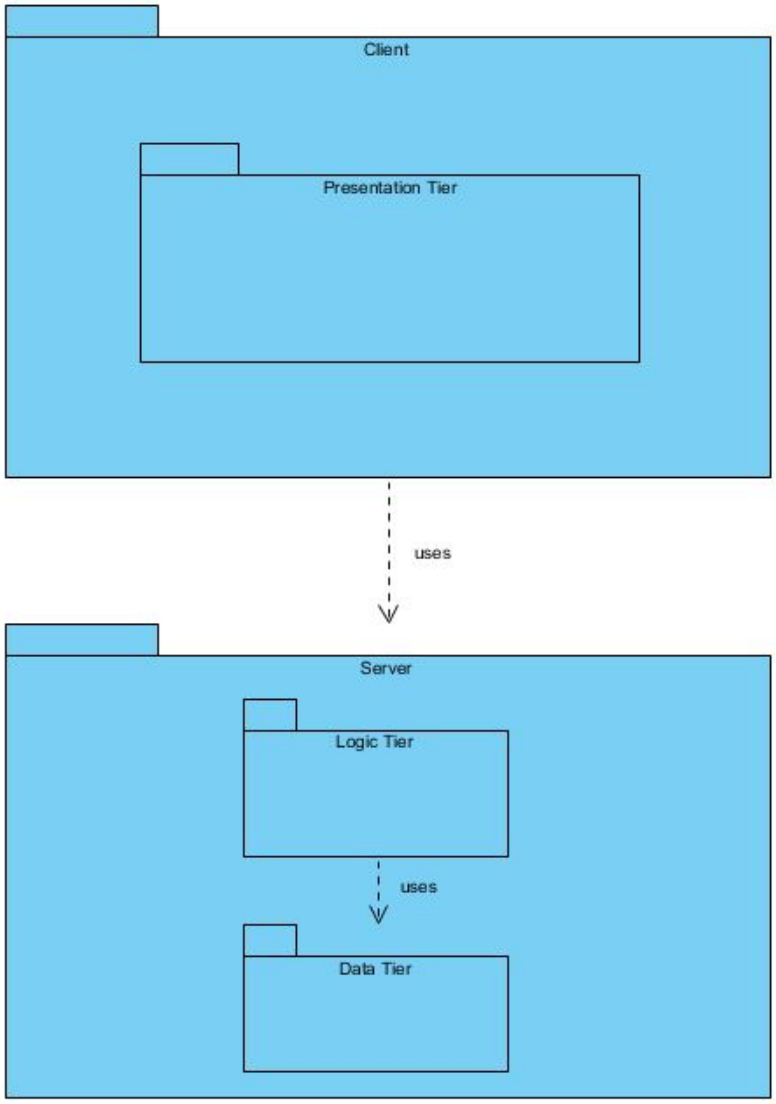


Figure 1: Subsystem Decomposition

## 3.2 Subsystem Decomposition

*AutoShop* has a subsystem decomposition of client-server architecture as shown in Figure 2 . This architecture model is chosen for *AutoShop* as the application basically needs the client to get inputs and present the output while doing most of the image processing on the server-side. Cohesion is satisfied with keeping presentation and input services in client subsystem and Image Modification Models for Auto Removal and Auto Addition functionalities the application in the Server subsystem.

*AutoShop* has a Presentation tier having 3 subsystems of Controller, Data and View. This tier will run on the client device which is a mobile phone in our case. The client device will be able to crop the image with the *Auto Crop*, *Smart Crop* and *Smooth Crop* options basic models that will run on its own processor. The application will do the

cropping and scaling tasks on its own as the communication with the server will cause overhead in time and cost in terms of money. The subsystem is composed of classes responsible for managing these operations. The Controller subsystem will be connected to the Data subsystem where the communication will occur to save the cropped and scaled images in the disk.

The View subsystem of the Presentation tier will control the navigation and composition of the pages. This subsystem will run on the client device which is mobile phone of the user. This subsystem will have Android GUI implementations that will have practical page designs and functionalities. Every page will have a class to operate itself. The View subsystem is connected to the Data subsystem as the data will be retrieved to be shown from Assets and created images.

Client Tier is using Server Tier to apply Neural Network Model to both *Auto Add* and *Auto Removal* functionalities of the *AutoShop*. This communication will be done after Client merges the asset with re-scaled and resized images. Server subsystem will generate the new image based on the image, asset and relative position. These data will be saved in the Server Database only if the user agrees on that. If the user does agree to deploy her photos, assets and final products they will be saved in *AutoShop* Databases. For the arrangement of this data *AutoShop* has Data Tier in Server subsystem. This subsystem has classes to efficiently receive the data, upload and receive it to the database.

Server subsystem has Application Logic Tier in addition to the Data Tier. Application Logic Tier has 4 essential classes. *Communication Manager* class handles the communication between Client and Server subsystems. *AdderNeuralNetworkEngine* is the class that gets the required inputs and runs the Neural Netowork Model on cloud computing results. The resulting image is returned to the user. *RemoverNeuralNetworkEngine* is similar to *AdderNeuralNetworkEngine* runs the model for *Auto Removal* functionality. *DatabaseManager* handles the communication between the Application Logic Tier with Data Tier of Server subsystem.

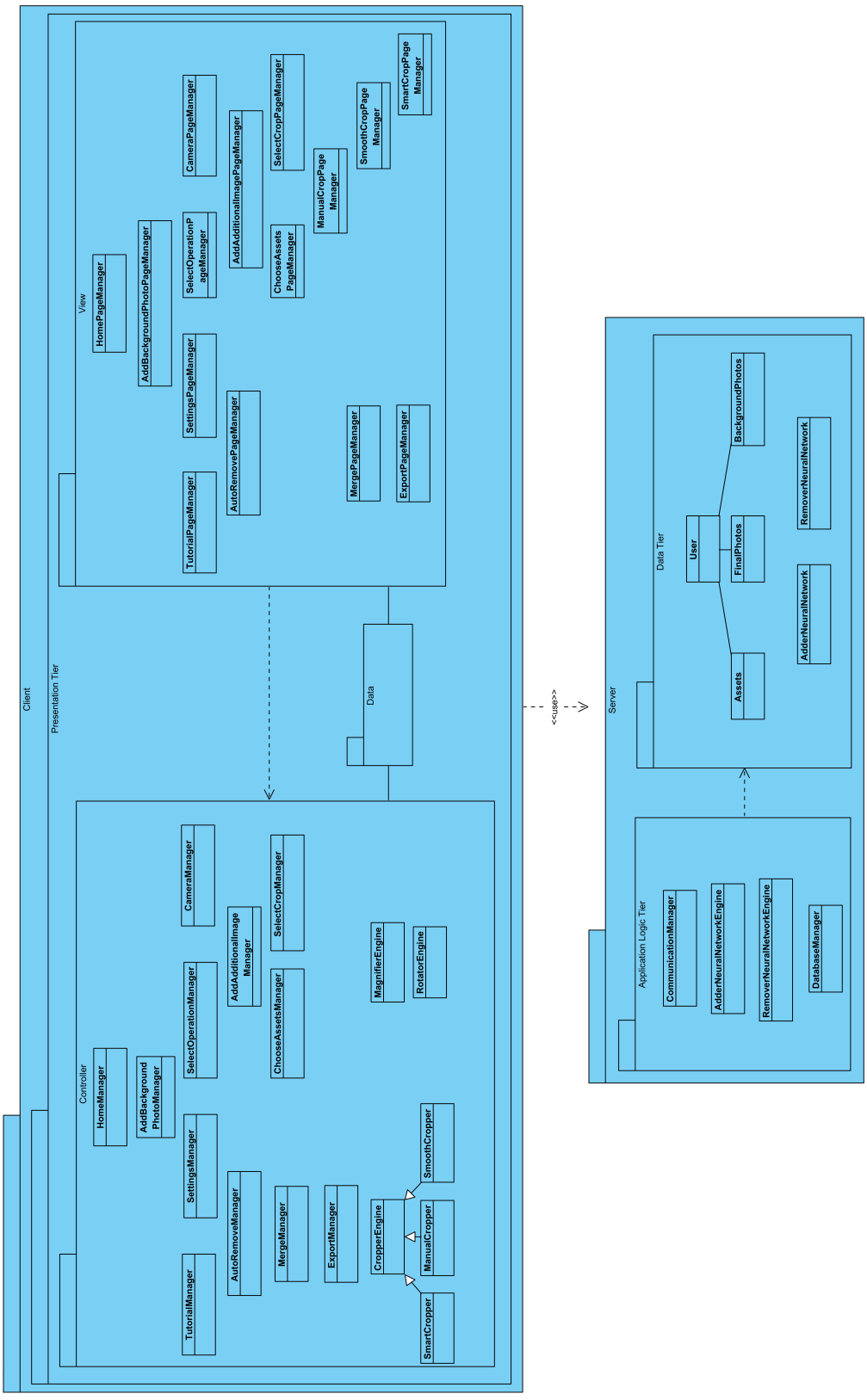


Figure 2: Detailed Subsystem Decomposition

## 3.3 Hardware/Software Mapping

Autoshop is an app that runs on Android smartphones. So, the software that runs within the client-side is compatible with various Android releases. The hardware side of the client-side is the various smartphones and tablets all around the world that run on Android. Autoshop will use the resources of the mobile phone such as camera, connectivity hardware, memory and so on. We will depend on the phone's memory since we are going to save the final version of the photos after a photo is processed by our system that we deployed on the AWS Cloud system.

Our backend which is server code and API integration will be implemented in Python programming language and Android application development will be done with Java and Android SDK. HTTP requests/responses will be used to be able to communicate with the server. We will use a relational database for storage. The backend/server-side of Autoshop is going to run on Amazon Web Services Linux servers. We choose Linux because it is stable and it is quite easy to compile and run programs on Linux. It is also more lightweight compared to other alternatives. Also, as mentioned before, we will benefit from NVIDIA libraries [2].

There will also be a database to keep essential information (explained in section 3.4 with more details). This database will have connections with the server and thus, provide and retrieve the information about the user and his/her images. In this deployment diagram below depicts the relationship between run-time components and nodes.

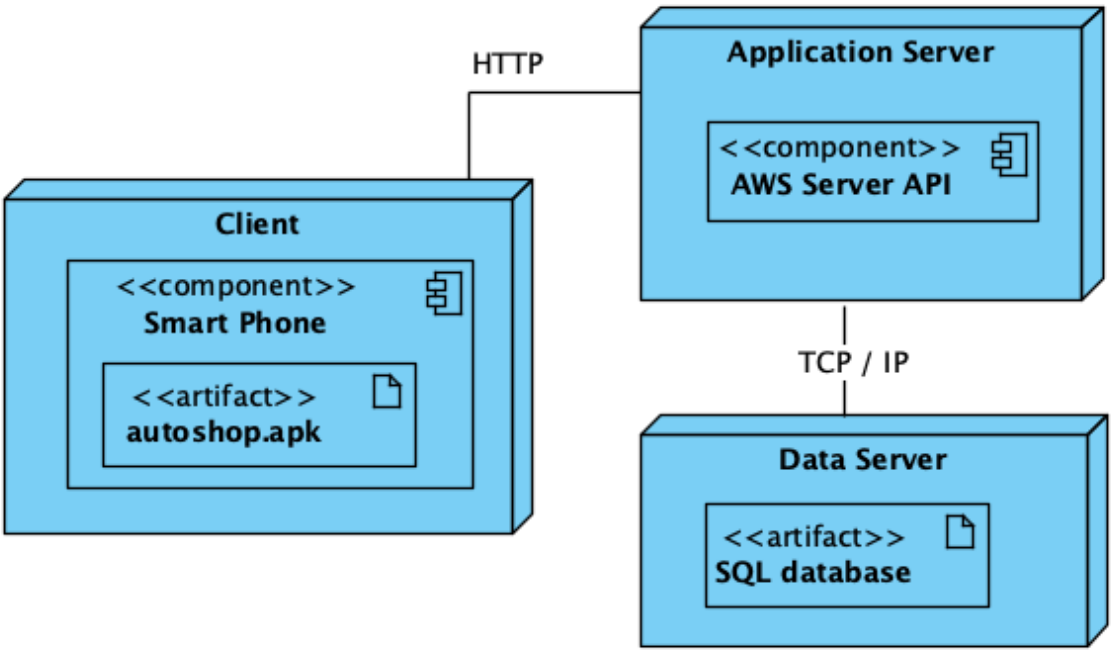


Figure 3: Hardware/Software Mapping

## 3.4 Persistent Data Management

Persistent data represents a bottleneck in the system on many different fronts: most functionality in the system is concerned with creating or manipulating persistent data. For this reason, access to the data should be fast and reliable. If retrieving data is slow, the whole system will be slow. If data corruption is likely, complete system failure is likely.

Almost all the unique features of our application are targeted to each individuals' needs. For example smart crop, smooth crop, automated image styling, etc. Thus for each user, our application will tailor a specific service. Therefore we will need to take this into consideration when designing our database. In our project, we are going to use a relational database to store persistent object information.

All the data will be saved in the Server Database only if a user agrees on that. Furthermore, if a user permits us we would also keep information that is specific to an individual user such as: user id, assets, final image that is edited by the user. We will make sure that our data does not get lost. This will be done via a backup database.

## 3.5 Access Control and Security

In a multi-user system, different actors must have different access rights over the data and functionality. Thus, access control in such applications is essential to ensure that only the users having the necessary permissions have control over the restricted parts of the system. Access control is particularly important in our setting, since *Autoshop* gathers data that belongs to separate individuals. In other words, the photos our users wish to edit may be private to them so the application should be accountable for protecting their data.

We applied the Group Based Access Control scheme when deciding who should access what. There do not exist many groups in our application but these groups together with the participants are given below:

- Models = {Auto-Add Model, Auto-remove model, Smart-crop model, etc.}
- Administrators = {Developers, System Maintainers}
- Users = {Anyone who enrolls by downloading the application}

Table 1 provides an Access Control matrix that specifies which of these entities have which access rights over which objects.

In Table 1, r denotes read permission, w denotes write permission and x denotes execute permission. Note that a specific user, say John, does not have any access right over another user's data, say Emilie's. Hence the permissions shown for the "Users" group is only valid for a specific user and his/her own data.

| | Assets | Input Images | Output Images |
|---|---|---|---|
| Models | rw- | rw- | r– |
| Administrators | - - - | - - - | - - - |
| Users | rw- | rw- | rw- |

Table 1: Access Control Matrix

Data privacy is a significant societal issue that we should certainly safeguard. *Autoshop* is designed to conform to the General Data Privacy Regulation (GDPR) [8]. Hence, as additional access control constraints, 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.

## 3.6 Global Software Control

In this section, we describe how the overall system is controlled on a global scale. We give a discussion on how requests are initiated and how subsystems synchronize. Finally, we address certain concurrency issues.

We consider two fundamental subsystems in this section, namely *Client* and *Server*, with the intent of providing a better abstraction. In brief, the software operates as requests to the server(s) are received from the client(s), these requests are then processed and the action taken in response is sent back to the client(s). This procedure can be divided into three main phases:

1. **Request Phase**: This phase begins with the initiation of the requests on the client-side. A request is initiated once the user is done preparing a photo and presses the buttons associated with the Auto-Add or Auto-Remove features. This user-action triggers a request which sends the photo that the user has prepared to the server as an input to be processed by the generative models. A naturally arising problem here is that the server(s) may be overwhelmed by a flood of requests, this flood may be a result of an unexpected increase in the number of users or a potential Denial of Service (DoS) attack. As a precaution for such scenarios, we will implement rate-limiting in our API, which will keep the client-side waiting until the server(s) are available.
2. **Processing Phase**: After a request arrives at the respective server port, a `Python`

script should run and apply the Auto-Add or the Auto-Remove model to the specific input. This is the phase that delays the response the most. Hence, it requires serious work to be parallelized. Due to the cost limitations, we will not be able to accommodate a large number of servers. However, in case the number of users increases significantly a budget must be allocated to acquire new servers which will in turn increase the number of cores so that various inputs can be processed by separate model instances at the exact same time. In such a case a *Load Balancing Algorithm* should be implemented to fairly distribute the work among different units, this also requires incorporating a *Load Balancer* hardware to the server-side.

3. **Response Phase**: In the response phase, we basically send the model outputs back to the clients from the ports where the respective requests have been received. To introduce concurrency to this phase and also the request phase we aim to find an optimal port configuration such that the delay experienced in the client-side is minimized.

## 3.7 Boundary Conditions

*AutoShop* will have three boundary conditions. These include: Starting the application, terminating the application and facing failure in the application. These conditions are discussed in detail in the following subsections below.

### Initialization

The user must have the application on their phone in order to use it which can be downloaded via Google Play Store. For the processing of the photo connection with the server must be established. Thus, internet connection is needed to initialize the application. If any asset or final product saved in previous runs, they will be called from local disk to be available to the user.

### Termination

When the user terminates the app, any running process in the server created by the user's run will be terminated. The final products generated by the app will be saved in the phone's local disk. Any unsaved photo or assets and any undone operations as cutting will not be saved for further runs.

### Failure

Failure can occur in five cases. First case can be seen while taking a new photo. The built-in camera application closes itself if the charge level of the phone drops below a certain level. The "certain level" mentioned above can change from phone to phone. In

such a case, a user is informed that the camera has been closed due to a low battery and the device should be connected to a power outlet to be charged so that the camera can be opened and a user can continue to take a new photo.

Second case of the failure is seen when the internet connection is dropped while uploading a photo. In such a case, the application waits for an internet connection to continue photo uploading job. Furthermore, user is notified about this situation and application shows a message "Network connection is lost, Please get into a network to continue uploading the photo." to the user.

Third case of the failure is seen if one of the modules stops working due to a bug in code or a problem in the server. In such a case, we inform the user that one module cannot be used at that time and the user will be notified whenever the module is ready.

Fourth case of failure is not having available space in phones disk. In this case user will not be able to save files. The user will be informed of this error directly. After this error prompt, if the user creates space while keeping *AutoShop* running in the background, the user will be able to save the product.

Fifth case of failure happens when the method for exporting to any social media platforms is not working that may be caused by various reasons like APIs becoming out-dated, social media applications crushing etc. These errors will be handled without interrupting the flow of *AutoShop* and will be prompted to the user.

# 4 Subsystem Services

In this section, we will describe the major components, i.e. subsystems, of our system. Our main two subsystems are client and server sides of the software which is a traditional method for implementing and governing mobile applications.

## 4.1 Client

Client has a Presentation tier which consists of 3 subsystems respectively Controller subsystem, View subsystem and Data subsystem. Controller subsystem will be responsible for the connection between client and server when data will be sent controller collects it and when responses are taken for requests, Controller collects it. View subsystem will be responsible for interface operations. Displaying pages or taken data on the screen will be done by the view subsystem. Data subsystem handles local storage for the output photos of users.

### 4.1.1 Controller Subsystem

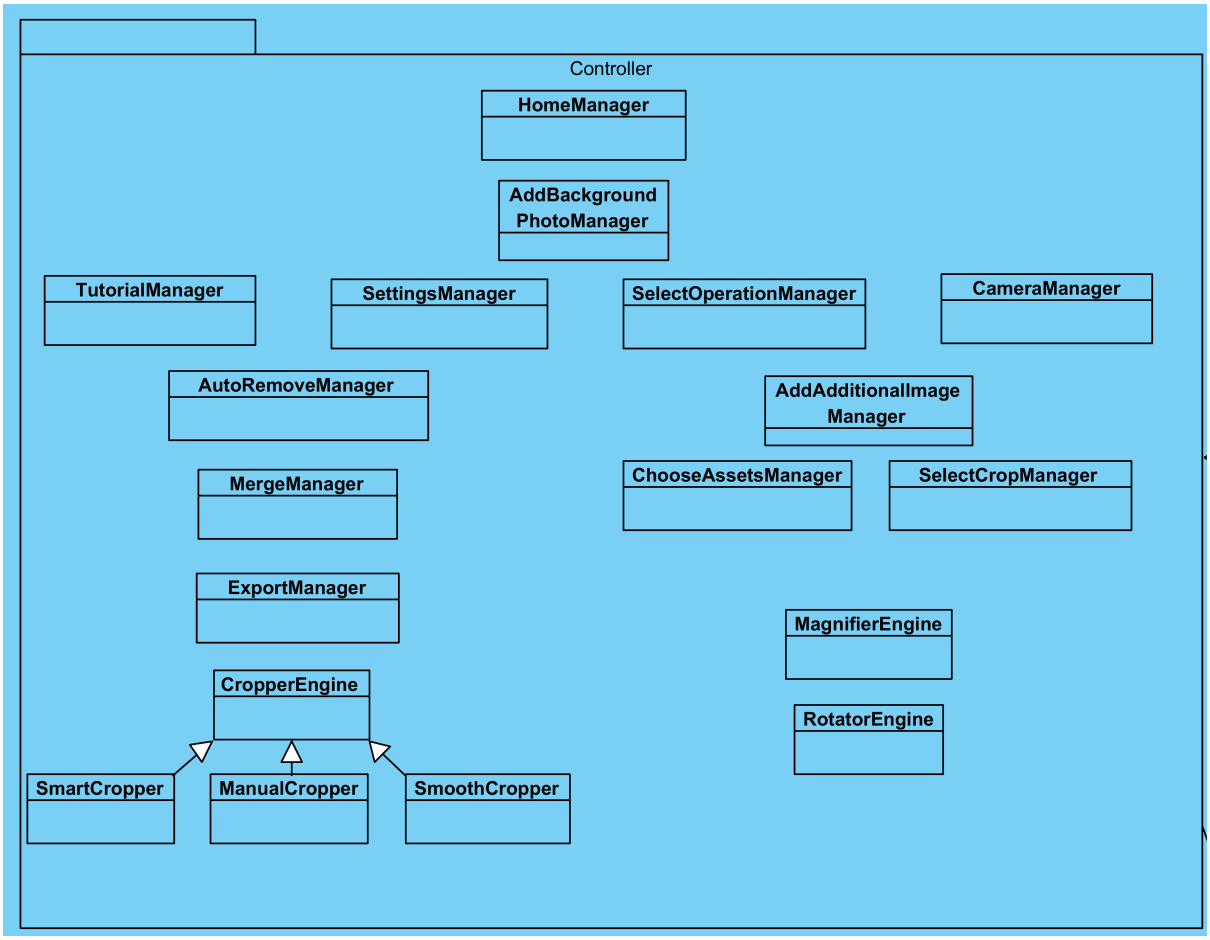


Figure 4: Subsystem Decomposition: Controller

Controller subsystem is responsible for handling the events that are received from the UI components that are mentioned in the View.

**HomeManager:**

HomeManager is responsible for loading the application on at the initial startup. At this stage, the application will establish a connection with the server.

**AddBackgroundPhotoManager:**

This class is responsible for retrieving the user gallery and allowing the user to select one of them.

**TutorialManager:**

This class is responsible for giving tutorials on how to use the application.

**SettingsManager:**

This class is responsible for allowing the user to change the settings of the application.

**SelectOperationManager:**

This class is responsible for allowing the user to choose between adding or removing images.

**CameraManager:**

This class is responsible for connecting to the camera and letting the user take a picture.

**AutoRemoveManager:**

This class is responsible for handling the scan input received from the user and applying the remove operations. The user will also be able to change the brush size using this class.

**AddAdditionalImageManager:**

This class is responsible for allowing the user to add additional images over the background image. If the user selects an image from the gallery shown, this class will forward that image for cropping. If the user chooses to select from assets, this class will change the page to a different view.

**MergeManager**

This class is responsible for sending the selected images to the server and receiving the modified image.

**ChooseAssetsManager**

This class is responsible for fetching the stored assets of the user.

**SelectCropManager**

This class is responsible for changing the page to the desired crop operator selected by the user.

**ExportManager**

This class is responsible for exporting finalized images to social media such as Tele-

gram, Whatsapp, Instagram, Facebook etc.

**CropperEngine**

This class is the parent class for all crop managers that are described below. It will implement generic functionality that will be used in each crop class.

**SmoothCropManager**

This class is responsible for letting the user crop the image using gestures. It will receive gesture information from the view and try to estimate the area to be cropped.

**SmartCropper**

This class will automatically crop the focused object in the image using a neural network architecture.

**ManualCropper**

This class is responsible for letting the user crop the image using gestures. It will receive gesture information from the view and draw rectangles representing the area to be cropped.

**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.

**MagnifierEngine**

This class is responsible for scaling images and assets up and down.

**RotatorEngine**

This class is responsible for rotating images and assets to the desired angle.

### 4.1.2 View Subsystem

The view subsystem will be responsible for managing the user interface operations.

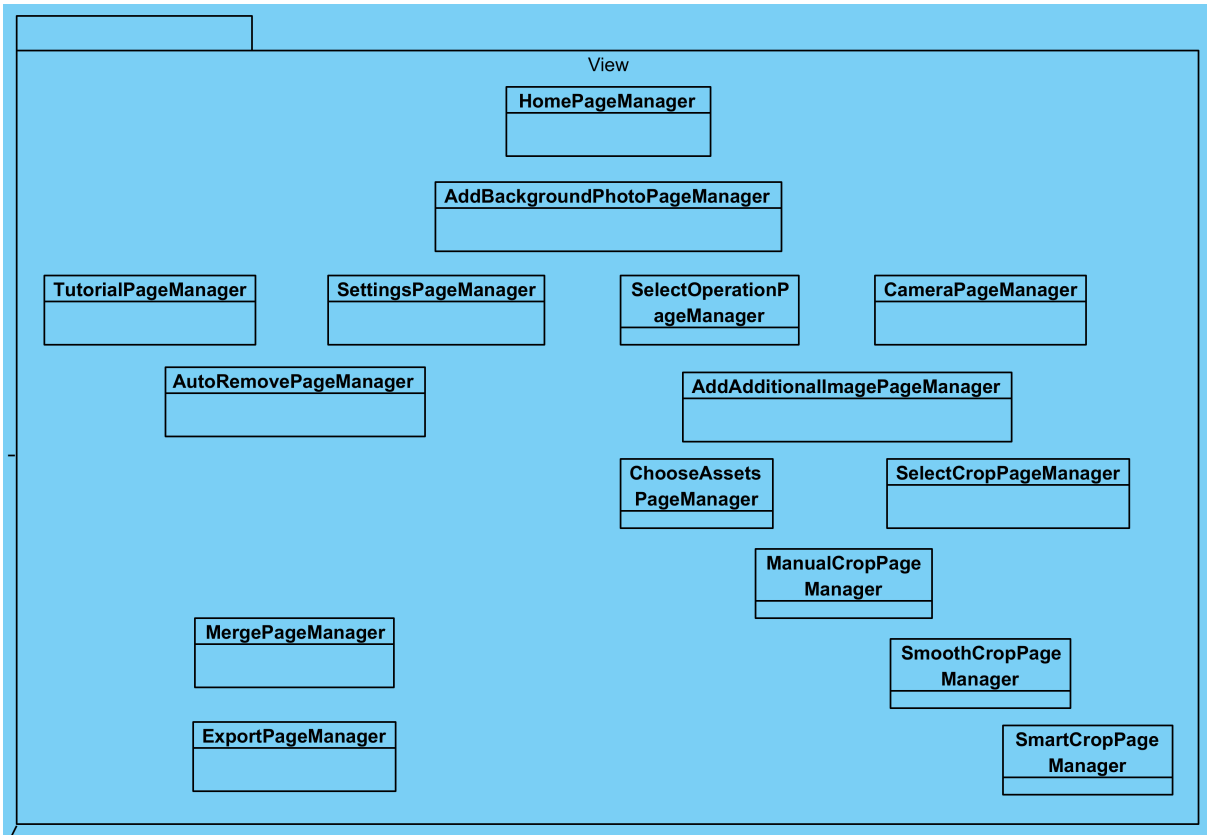


Figure 5: Subsystem Decomposition: View

View subsystem will consist of following parts:

**HomePageManager:**

This view is responsible for displaying an initial home page.

**AddBackgroundPhotoPageManager:**

This view is responsible for displaying the Add background Photo page where user chooses the background photo.

**TutorialPageManager:**

This view is responsible for displaying Tutorials.

**SettingsPageManager:**

This view is responsible for displaying the settings page.

**SelectOperationPageManager:**

This view is responsible for displaying an option to choose either add or remove

operation.

**CameraPageManager:**

This view is responsible for displaying camera for users to take a photo.

**AutoRemovePageManager:**

This view is responsible for displaying remove page after selecting remove operation.

**AddAdditionalImagePageManager:**

This view is responsible for displaying the additional images that the user can add over the background image.

**ChooseAssetsPageManager:**

This view is responsible for displaying the user's assets for selection.

**SelectCropPageManager:**

This view is responsible for displaying the cropping options to the user.

**ManualCropPageManager:**

This view is responsible for displaying the image to be cropped and receiving user gestures. After the gesture is received, a rectangle marking the area to be cropped is displayed.

**SmoothCropPageManager:**

This view is responsible for displaying the image to be cropped and receiving user gestures. After the gesture is received, a polygon marking the area to be cropped is displayed.

**SmartCropPageManager:**

This view is responsible for displaying the image to be cropped and receiving user gestures. After the gesture is received, a rectangle marking the area to be cropped is displayed, similar to ManualCropPageManager.

**ExportPageManager:**

This view is responsible for displaying export options.

**MergePageManager:**

This view is responsible for displaying the final images that are ready to be merged.

### 4.1.3 Data Subsystem

This subsystem stands for managing local storage in which final edited pictures of the users will be stored.


Figure 6: Subsystem Decomposition: Data

## 4.2 Server

Server has two layers. Logic Tier and Data Tier. Logic Tier is where all user interaction is handled. Logic Tier interacts with the client in a request/response manner. Every time a user wants to access a service of Autoshop, Logic Tier will handle the request and generate the appropriate ate response for the user. Data Tier includes a Database Management Subsystem. Basically, this database is where all the persistent objects are stored.

### 4.2.1 Logic Tier

This layer is responsible for all major operations of the system. The part of the server will communicate with many different APIs to service for instance Autoshop's auto-add and auto-remove operations. When the client sends a request, then the request will be parsed and transmitted to the appropriate service.

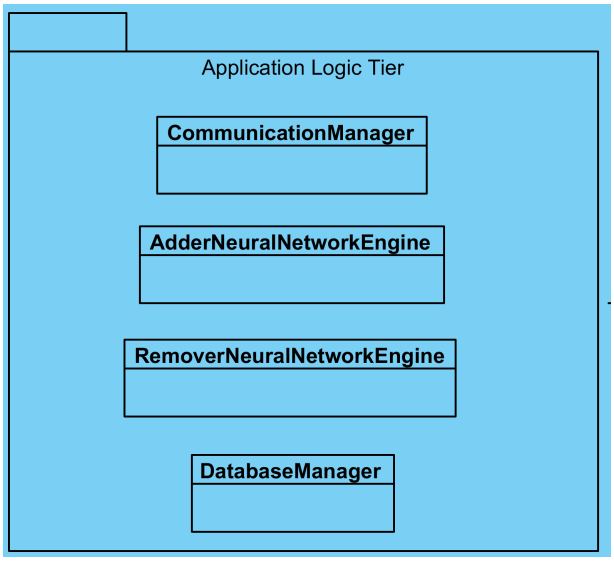


Figure 7: Subsystem Decomposition: Logic

**CommunicationManager**

This Manager is responsible for creating and controlling communication between client and user. It will arrange the http ports and allow the data exchange between client and server.

**AdderNeuralNetworkEngine**

A specific **Model** that contains a neural network trained for context-aware object addition, in other words automated image styling.

**RemoverNeuralNetworkEngine**

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.

**DatabaseManager**

This class is responsible for managing user data. It communicates with the Data Tier to save the processed images. The client will send a request to this class, and the class will call the necessary functions to give the appropriate response to the client.

### 4.2.2 Data Tier

Data Tier manages interactions with the database. It will communicate with the Logic Tier to service the requested data.

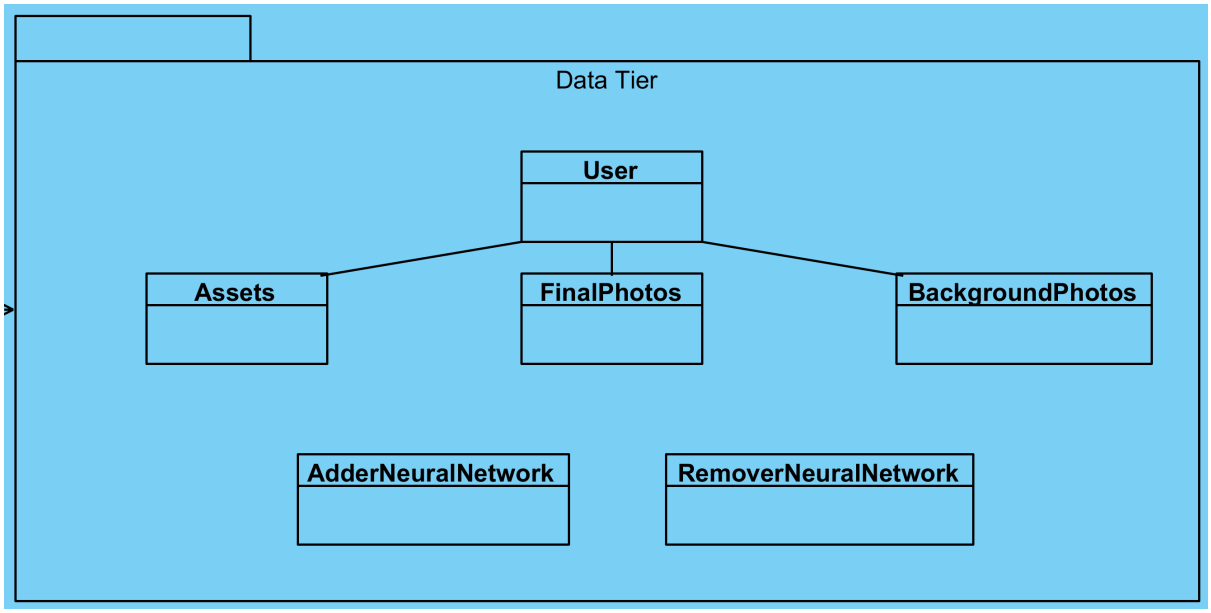


Figure 8: Subsystem Decomposition: Data

**User:** A user with her id is stored within the database to query her essential data such as final processed photo, assets if she permits us.

**Assets:** A specific type of **Image** with a well-defined boundary that is not necessarily rectangular. This class is used to model the additional images.

**FinalPhotos:** These photos will correspond to the finalized and photoshopped photos received from the users that give us permissions.

**BackgroundPhotos:** These photos will correspond to the background and additional photos received from the users that give us permissions.

**AdderNeuralNetwork:** The essential data that represents the adder neural network will be stored here. This data consists of the weights, biases, number of layers, neurons, activation function details, optimizer parameters and other hyper-parameters that require tuning.

**RemoverNeuralNetwork:** The essential data that represents the remover neural network, which are described above, will be stored here.

# 5 New Knowledge Acquired and Learning Strategies Used

| **Knowledge Acquired** | **Learning Strategy Used** |
|---|---|
| Details about NVIDIA's `FastPhotoStyle` and `ImageInpainting` Libraries: How the libraries operate, how they should be deployed, whether to deploy on mobile or dedicate a server etc. | Hands-on Experience: Libraries were tested on `Jupyter` environment, an interview is conducted with an ex-NVIDIA-employee on how to efficiently deploy the models. |
| Android Development Basics: Android Studio, Intents, Activities, etc. | Online Learning: Specified topics are learned from official online tutorials. |
| Synchronization and Concurrency in Client-Server Architectures: How to reduce delay in Client-Server architectures, Load Balancing, I/O Ports, Binding and Listening Ports in Python etc. | Literature Review and Online Learning: Reviewed related sections of Object Oriented Software Engineering using UML, Patterns and Java by Bruegge and Dutoit, studied from online sources. |
| Access Control and User Rights: Access Control Lists, Capabilities, Group Based Access Control | Literature Review: Reviewed related sections of Computer security by Dieter Gollmann. |
| System Decomposition: Decomposing a system into subsystems, designing how subsystems should interact , etc. | Literature Review and Online Learning: Reviewed related sections of Object Oriented Software Engineering using UML, Patterns and Java by Bruegge and Dutoit. |
| Soft Skills in Software Engineering: Teamwork, Work Ethics, Punctuality, etc. | Hands-on Experience: Conducted team meetings on a regular basis, distributed workload fairly and checked each others' work. |
| Presentation and Communication Skills | Hands-on Experience: Presented reviews of two ACM papers to our supervisor. |
| Relational Database Model | Literature Review: Reviwed the introductory sections of Database System Concepts by Silberschatz, Korth, and Sudarshan |

Table 2: Knowledge Acquired and Learning Strategies Used

# 6 Glossary

| | |
|---|---|
| `Component` | Self-contained entities that provide services to other components or actors. A Web server, for example, is a component that provides services to Web browsers. A Web browser such as Safari is a component that provides services to a user [9]. |
| `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 [2]. |
| `ImageInpainting` | An NVIDIA library available in `Python`. The library can remove parts of the image and replace the removed parts with realistic artificial parts [5]. |
| `Jupyter` | Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages. |
| Neural Networks | A set of algorithms, modeled loosely after the human brain, that is designed to recognize patterns [10]. |
| `Node` | Physical device or an execution environment in which components are executed [9]. |

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

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

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

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

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

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

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

[9] B. Bruegge and A. H. Dutoit, "Object-oriented software engineering using uml, patterns, and java," 2009.

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