Bilkent University

Department of Computer Engineering

Senior Design Project

Autoshop

Low 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/

Low Level Design Report
Feb 17, 2020
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 right away 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.

This report aims to provide an overview of the low-level architecture and design of our system. First of all, the trade-offs of our design and engineering standards will be explained. What follows will be the documentation guidelines. After that, information about the packages and interfaces of Autoshop’s system will be presented. Finally, the class diagrams and a detailed look into each of Autoshop’s software components will conclude the report.

1.1 Object Design Trade-Offs

In software development, when choosing to enhance a certain feature of a program, usually another one has to be sacrificed. Almost all decisions with respect to design comes with certain trade-offs and implications. To create the optimal system for Autoshop we spent a lot of time identifying trade-offs during the design process. In the following sections, the trade-offs we have encountered will be presented.

1.1.1 Functionality vs Usability

Two of the most important aspects to consider about design trade-offs are Autohop's functionality and usability. Functionality refers to whether Autohop's functionalities are working as intended whereas the usability refers to the ease of use and intuitiveness of these presented functions. Even though our application offers excellent and practical functionalities, they will offer no value if the user cannot interact with them. For this reason, Autoshop's user interface will be as intuitive and easy to use as possible. Thus, it is possible to say that our design favors usability over functionality.

1.1.2 Compatibility vs Extensibility

First, we thought that our project will be an Android application but now we consider making Autoshop available both on Android and iOS platforms. Therefore, it is important that Autoshop is compatible with all these systems. For this reason, we decided to use Flutter for development. Flutter is an open-source UI software development kit developed by Google. It helps design and export apps that work in both Android and iOS while providing the ability to run native code using plug-ins. Thus, Flutter minimizes the drawbacks of cross-platform development, which are mainly caused by loosing the privileges of native development.

Extensibility is also important as Autoshop will have to evolve with updates in the future. There may even be a version of Autoshop that runs in a web browser. Although the ability to extend the project is crucial, we think creating an application compatible with multiple platforms is much more desired.

1.1.3 Space vs Time

Autoshop will necessarily use powerful servers to handle GPU heavy tasks such as computing neural network outputs and performing object segmentation. As the user wishes to edit high-resolution images the data stored and processed in the server-side gets larger. This will theoretically increase the response time of the application since the server will spend more time processing and transferring the data, so there will be an inevitable trade-off between space and time. To avoid this we plan to use lossless compression techniques that helps us reduce the image size while keeping the modifications visible. Our primary focus will be response time since it is the key for a good user experience.

1.1.4 Robustness vs Cost

Autoshop aims to provide a service that is reliable in terms of its outputs and is available with full functionality for the users to access the application whenever they desire. This leads the application to use better services rather than its minimal requirements, for example using a cloud maintained by Google is preferable than a small server that is maintained by the developers. Obviously, using better services is costly in terms of money. Even if robustness of Autoshop leads to more cost, it is a primary design goal that we will try to satisfy as much as possible.

1.2 Interface Documentation Guidelines

In this report, all the class names are named in the standard ‘ClassName’ format, where all of these names are singular. The variable and method names follow a similar rule as in ‘variableName’ and ‘methodName()’. In the class description hierarchy, the class names appear first, then the attributes of the class appear, and the hierarchy end with the methods of the class. In the following table, the detailed outline is presented as:

class Sample	
This is a sample class responsible for ...	
Attributes	
`private String name`	
`private int no`	
`public int IP`	
Methods	
`public String getName()`	Returns the name of this object.
`public void setIP(int IP)`	Sets the IP attribute of this object.
`public void setAll(String name, int no, int IP)`	Sets the entire set of attributes of this object, namely @name, @no, and @int.

1.3 Engineering Standards

For the descriptions of the class interfaces, diagrams, scenarios, use cases, subsystem compositions and hardware depictions, this report follows the UML guidelines [2]. UML is a commonly used way to generate these diagrams, easy to use and since it is the method taught at Bilkent University, we chose to utilize it in the following pages. For the citations, the report follows IEEE's standards [3].

1.4 Definitions, Acronyms, and Abbreviations

AWS	Amazon Web Services. Reliable, scalable, and inexpensive cloud computing services provided by Amazon. Remote servers can be rented through this service [4].
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.
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 [2].

2 Packages

Autoshop's Low Level system is composed of two main subsystems, which are client and server. First three parts are located inside the client-side. These parts are called View, Controller and Data. Last two parts represent the Server subsystem which are called Application Logic Tier and Data Tier. In our project, we used a novel approach to connect the client and the server. Client presents the system information to users. It also receives the interaction of users and sends them to server to keep the UI (user interface) running. On the other hand, the server-side is responsible for all non-local data processing and API usage.

2.1 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.

2.1.1 Controller

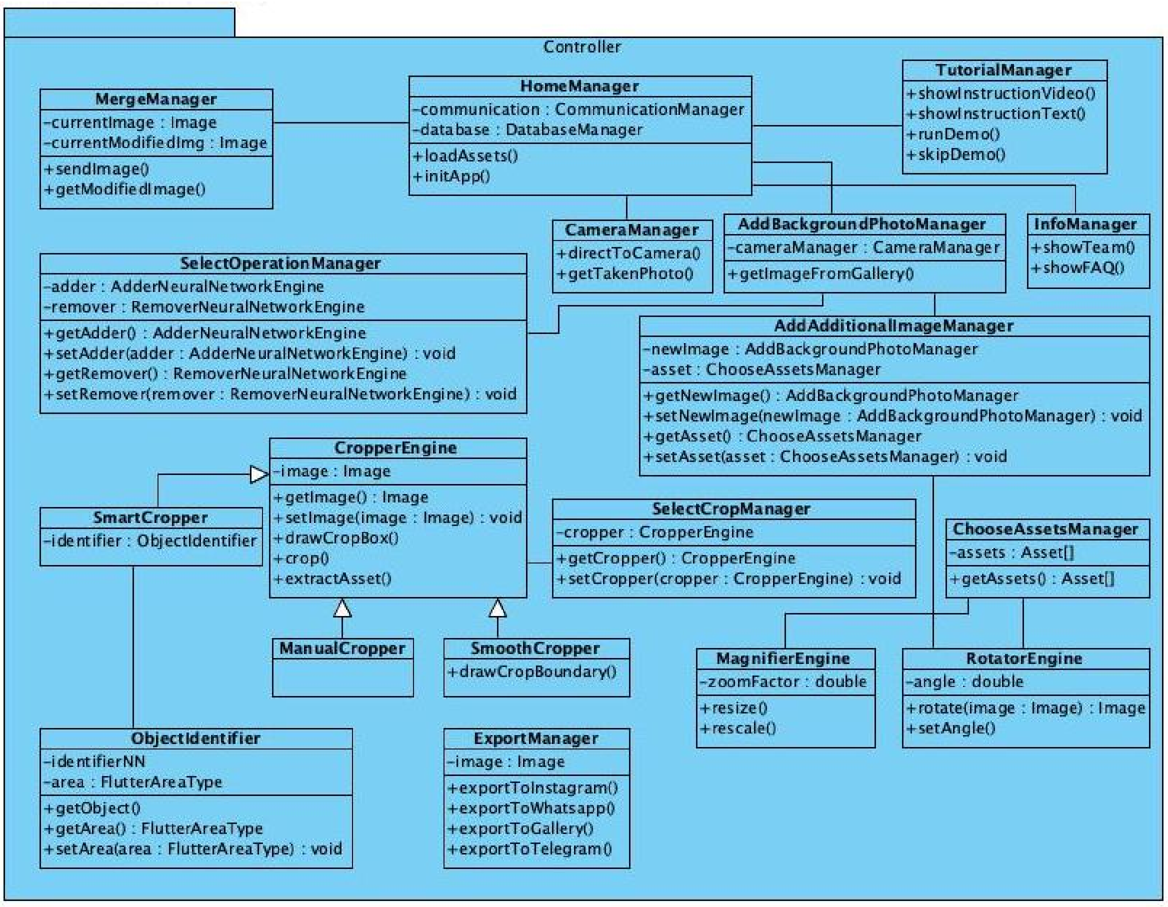
Controller
MergeManager
-currentImage : Image
-currentModifiedImg : Image
+sendImage()
+getModifiedImage()
HomeManager
-communication : CommunicationManager
-database : DatabaseManager
+loadAssets()
+initApp()
TutorialManager
+showInstructionVideo()
+showInstructionText()
+runDemo()
+skipDemo()
CameraManager
+directToCamera()
+getTakenPhoto()
AddBackgroundPhotoManager
-cameraManager : CameraManager
+getImageFromGallery()
InfoManager
+showTeam()
+showFAQ()
SelectOperationManager
-adder : AdderNeuralNetworkEngine
-remover : RemoverNeuralNetworkEngine
+getAdder() : AdderNeuralNetworkEngine
+setAdder(adder : AdderNeuralNetworkEngine) : void
+getRemover() : RemoverNeuralNetworkEngine
+setRemover(remover : RemoverNeuralNetworkEngine) : void
AddAdditionalImageManager
-newImage : AddBackgroundPhotoManager
-asset : ChooseAssetsManager
+getNewImage() : AddBackgroundPhotoManager
+setNewImage(newImage : AddBackgroundPhotoManager) : void
+getAsset() : ChooseAssetsManager
+setAsset(asset : ChooseAssetsManager) : void
CropperEngine
-image : Image
+getImage() : Image
+setImage(image : Image) : void
+drawCropBox()
+crop()
+extractAsset()
SmartCropper
-identifier : ObjectIdentifier
SelectCropManager
-cropper : CropperEngine
+getCropper() : CropperEngine
+setCropper(cropper : CropperEngine) : void
ChooseAssetsManager
-assets : Asset[]
+getAssets() : Asset[]
ManualCropper
SmoothCropper
+drawCropBoundary()
MagnifierEngine
-zoomFactor : double
+resize()
+rescale()
RotatorEngine
-angle : double
+rotate(image : Image) : Image
+setAngle()
ObjectIdentifier
-identifierNN
-area : FlutterAreaType
+getObject()
+getArea() : FlutterAreaType
+setArea(area : FlutterAreaType) : void
ExportManager
-image : Image
+exportToInstagram()
+exportToWhatsapp()
+exportToGallery()
+exportToTelegram()

Figure 1: 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 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.

InfoManager:

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.

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 Telegram, 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.

SmartCropper

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

ObjectIdentifier

This class is responsible for identifying objects which would be used to extract objects when smart cropper is used.

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.

2.1.2 View

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

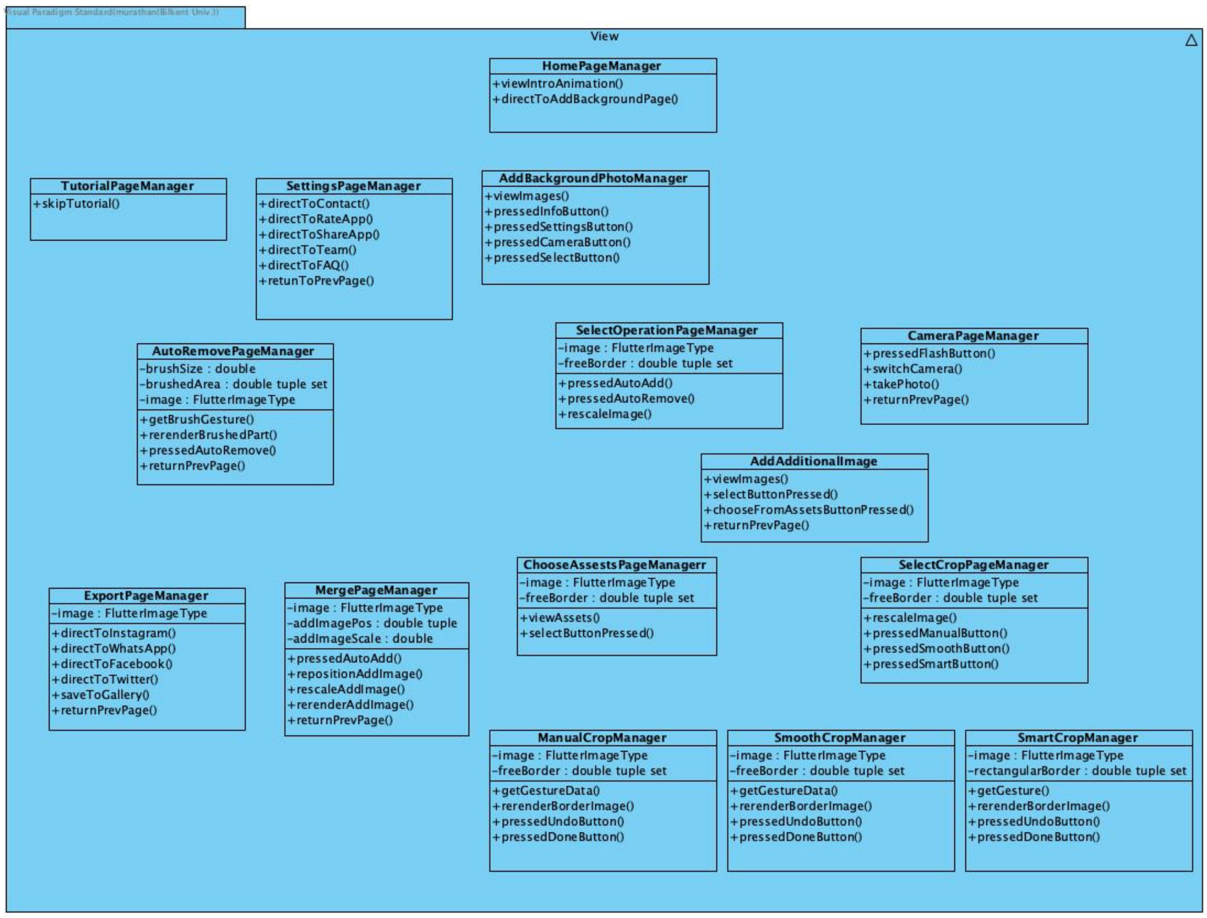

Figure 2: Subsystem Decomposition: View

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.

2.1.3 Data

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

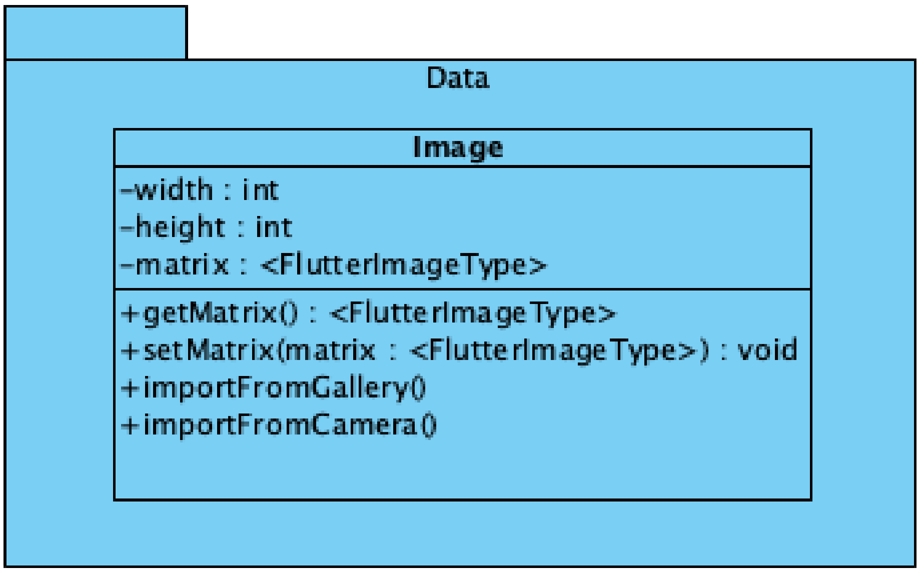
Data
Image
-width : int
-height : int
-matrix : <FlutterImageType>
+getMatrix() : <FlutterImageType>
+setMatrix(matrix : <FlutterImageType>) : void
+importFromGallery()
+importFromCamera()

Figure 3: Subsystem Decomposition: Data

Image:

This class is responsible for keeping images in storage of the device and utilizing other classes which take images and do operations on this input image that they take.

2.2 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.

2.2.1 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.

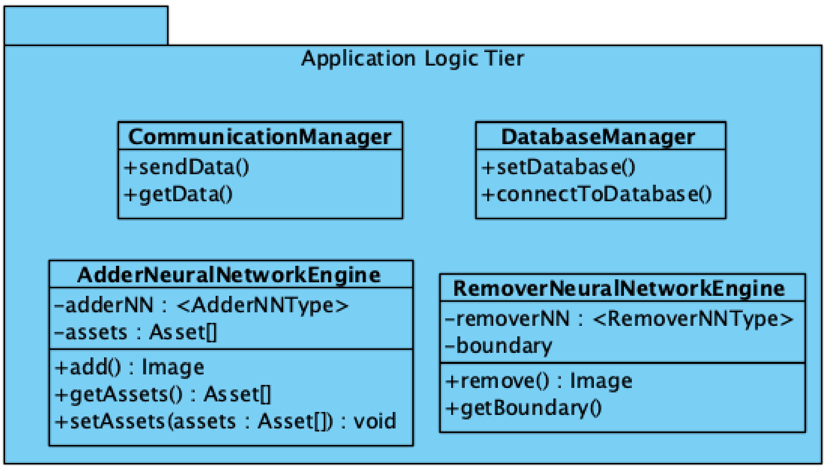

Figure 4: Subsystem Decomposition: Logic

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.

2.2.2 Data Tier

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

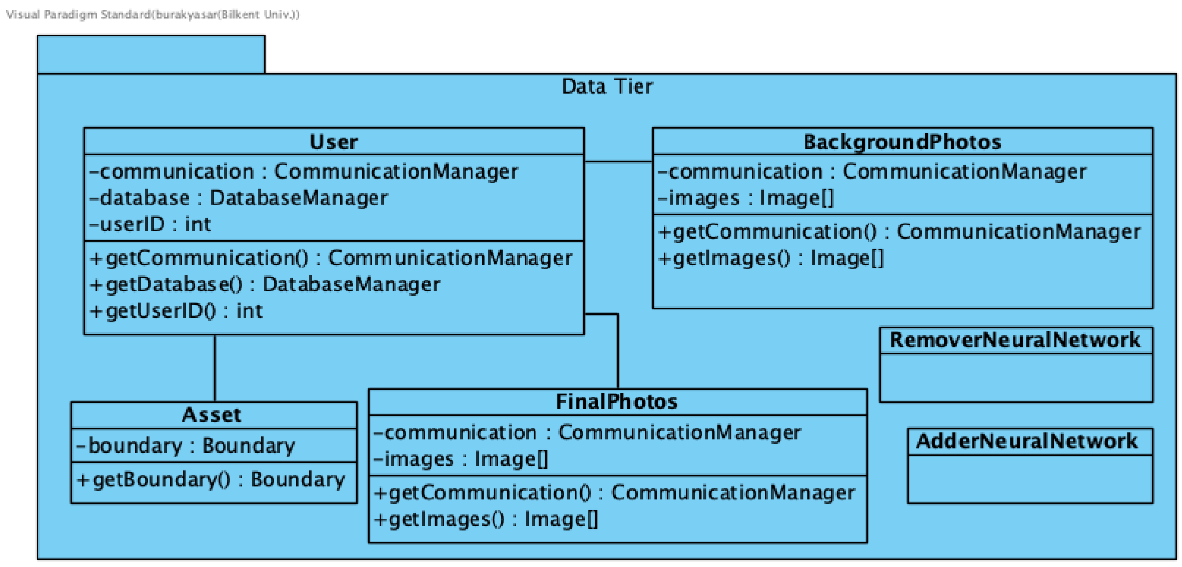
Visual Paradigm Standard(burakyasar(Bilkent Univ.))
Data Tier
User
-communication : CommunicationManager
-database : DatabaseManager
-userID : int
+getCommunication() : CommunicationManager
+getDatabase() : DatabaseManager
+getUserID() : int
BackgroundPhotos
-communication : CommunicationManager
-images : Image[]
+getCommunication() : CommunicationManager
+getImages() : Image[]
RemoverNeuralNetwork
AdderNeuralNetwork
Asset
-boundary : Boundary
+getBoundary() : Boundary
FinalPhotos
-communication : CommunicationManager
-images : Image[]
+getCommunication() : CommunicationManager
+getImages() : Image[]

Figure 5: 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.

3 Class Interfaces

In this section, signatures, properties and methods of the classes will be provided. In addition, their specific duties will be indicated in detail.

3.1 Client

In this section the subsystems are dedicated for the client part namely mobile application. Names of the functions and classes that are listed may alter along the development life cycle of the project.

3.1.1 Controller

class 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.	
Attributes	
`private CommunicationManager communication`	
`private DatabaseManager database`	
Methods	
`public Asset[] loadAssets()`	Loads assets from device storage
`public void initApp()`	Boots up the application

class AddBackgroundPhotoManager	
This class is responsible for retrieving the user gallery and allowing the user to select one of them.	
Attributes	
`private CameraManager cameraManager`	
Methods	
`public Image getImageFromGallery()`	Returns the image from the gallery.

class TutorialManager	
This class is responsible for giving tutorials on how to use the application.	
Methods	
`public void showInstructionVideo()`	Shows a tutorial video about Autoshop.
`public void showInstructionText()`	Navigates to the text which explains the functionalities of Autoshop
`public void runDemo()`	Starts the demo.
`public void skipDemo()`	Provides user to skip the demo.

class InfoManager	
This class is responsible for allowing the user to change the settings of the application.	
Methods	
`public void showTeam()`	Shows the developer team.
`public void showFAQ()`	Shows frequently asked questions.

class SelectOperationManager	
This class is responsible for allowing the user to choose between adding or removing images.	
Attributes	
`private AdderNeuralNetworkEngine adder`	
`private RemoverNeuralNetworkEngine remover`	
Methods	
`public AdderNeuralNetworkEngine getAdder()`	Returns adder object to have the picture added automatically.
`public RemoverNeuralNetworkEngine getRemover()`	Returns remover object to have the selected object removed automatically.

class CameraManager	
This class is responsible for connecting to the camera and letting the user take a picture.	
Methods	
`public void directToCamera()`	Provides access to camera.
`public Image getTakenPhoto()`	Returns the image which is taken by camera.

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.	
Attributes	
`private AddBackgroundPhotoManager backPhoto`	
`private ChooseAssetsManager chooseAssets`	
`private CameraManager camera`	
Methods	
`getters for attributes`	Returns the objects of corresponding class.

class MergeManager	
This class is responsible for sending the selected images to the server and receiving the modified image.	
Attributes	
`private Image currentImage`	
`private Image modifiedImage`	
Methods	
`public Image getModifiedImage()`	Returns modified image.
`public void sendImage()`	Send the image to the server.

class ChooseAssetsManager	
This class is responsible for fetching the stored assets of the user.	
Attributes	
`private Asset[] assets`	
Methods	
`public Asset[] getAssets()`	Returns the assets.

class SelectCropManager	
This class is responsible for changing the page to the desired crop operator selected by the user.	
Attributes	
`private CropperEngine cropperEngine`	
Methods	
`public void setCropperEngine()`	Sets the cropper type with polymorphism cropper type may be smooth, smart or manual cropper.

class ExportManager	
This class is responsible for exporting finalized images to social media such as Telegram, Whatsapp, Instagram, Facebook etc.	
Attributes	
`private Image image`	
Methods	
`public void exportToInstagram()`	Exports the image to Instagram.
`public void exportToWhatsapp()`	Exports the image to Whatsapp.
`public void exportToTelegram()`	Exports the image to Telegram.
`public void exportToFacebook()`	Exports the image to Facebook.
`public void exportToTwitter()`	Exports the image to Twitter.
`public void exportToGallery()`	Exports the image to gallery.

class 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.	
Attributes	
`private Image image`	
Methods	
`public Image getImage()`	Gets the image object.
`public void drawCropBox()`	Draws the crop box.
`public void crop()`	Crops the selected area.
`public void extractAsset()`	Extracts the selected area from the image.

class SmartCropper
This class will automatically crop the focused object in the image using a neural network architecture.
Attributes
`private ObjectIdentifier identifier`

class ObjectIdentifier	
This class is responsible for identifying objects which would be used to extract objects when smart cropper is used.	
Attributes	
`private <NeuralNetType> identifierNN`	
`private <FlutterAreaType> area` `public int IP`	
Methods	
`public Asset getObject()`	Returns the identified object.
`public <FlutterAreaType> getArea()`	Returns the area (library specific type).

class 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. No methods and attributes, directly inherits from CropperEngine.

class 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.	
Methods	
`public void drawCropBoundary()`	Provides an aided crop box selection after drawing.

class MagnifierEngine	
This class is responsible for scaling images and assets up and down.	
Attributes	
`private double zoomFactor`	
Methods	
`public void resize(Image image)`	Resizes the image.
`public void rescale(Image image)`	Rescales the image.

class RotatorEngine	
This class is responsible for rotating images and assets to the desired angle.	
Attributes	
`private double angle`	
Methods	
`public void setAngle(Image image)`	Sets the angle of the rotation.
`public void rotate(Image image)`	Rotates the image

3.1.2 View

class HomePageManager	
HomePageManager is responsible for screening the *AutoShop* logo while the Control package of the app initiates the server connections.	
Methods	
`public void viewInto()`	Plays the animation of *Beast United* without user input.
`public void directToAddBackgroundPage()`	Directs app to the next page which is AddBackground Page.

class AddBackgroundPhotoManager	
AddBackgroundPhotoManager is responsible for show the the images from gallery and a user friendly UI to choose them or direct user to Camera page. *AutoRemove*'d.	
Methods	
`public void viewImages()`	Renders image on the page with in a grid-style design.
`public void pressedInfoButton()`	Directs App to Info page.
`public void pressedSettingsButton()`	Directs App to Settings page.
`public void pressedCameraButton()`	Directs App to Camera page
`public void pressedSelectButton()`	Makes photos able to be selected.

class SelectOperationPageManager	
SelectOperationPageManager is responsible for providing the user friendly UI to let user tdecide on which operation to do.	
Attributes	
`private FlutterImage image`	
`private double tuple set freeBorder`	
Methods	
`public void pressedAutoAdd()`	Directs App to AutoAdd Page.
`public void pressedAutoRemove()`	Directs App to AutoRemove Page
`public void rescaleImage()`	With the data gathered by user it rescales the image.

class CameraPageManager	
CameraPageManager is responsible for providing the user friendly UI take a photo.	
Methods	
`public void pressedFlashButton()`	Turns on/off the flash light.
`public void switchCamera()`	Switches the camera between rear and front.
`public void takePhoto()`	Takes a photo.
`public void returnPrevPage()`	Directs App to previous page

class AddAdditionalImage	
AddAdditionalImage is responsible for providing the user friendly UI to add Additional Image for AutoAdd functionality of *AutoShop*.	
Methods	
`public void viewImages()`	Shows images to user from Gallery.
`public void selectButtonPressed()`	Enables photos to be selected.
`public void chooseFromAssetsButtonPressed()`	Directs App to ChooseFromAssets Page.
`public void returnPrevPage()`	Directs App to previous page

class SelectCropPageManager	
SelectCropPageManager is responsible for providing the user friendly UI to let user choose the crop type she will perform.	
Attributes	
`private FlutterImage image`	
`private double tuple set freeBorder`	
Methods	
`public void rescaleImage()`	Rescales images with the gesture data from user gestures.
`public void pressedManualButton()`	Directs App Manual Crop Page.
`public void pressedSmoothButton()`	Directs App to Smooth Crop Page.
`public void pressedSmartButton()`	Directs App to Smart Crop Page.

class ChooseAssetsPageManager	
ChooseAssetsPageManager is responsible for providing the user friendly UI represents the pre-available assets and let user choose among them.	
Attributes	
`private FlutterImage image`	
`private double tuple set freeBorder`	
Methods	
`public void viewAssets()`	Renders Asssets in a grid-style design.
`public void selectButtonPressed()`	Asset is chosen by user.

class SmartCropManager	
SmartCropManager is responsible for providing the user friendly UI to user set the rectangular border for smart cropper.	
Attributes	
`private FlutterImage image`	
`private double tuple set rectangularBorder`	
Methods	
`public void getGesture()`	Get gestures from user to adjust the rectangular border.
`public void rerenderBorderImage()`	Rerenders the image after border is changed.
`public void pressedUndoButton()`	Undos users last move.
`public void pressedDoneButton()`	Directs App to apply the smart crop.

class SmoothCropManager	
SmoothCropManager is responsible for providing the user friendly UI to set the border for Smooth Crop.	
Attributes	
`private FlutterImage mage`	
`private double tuple freeBorder`	
Methods	
`public void getGestureData()`	Get gesture input from the user.
`public void rerenderBorderImage()`	According to latest user gestures rerenders the border.
`public void pressedUndoButton()`	Undos the latest change by the user on the border.
`public void pressedDoneButton()`	Finishes this stage of app and directs user to next page.

class ManualCropManager	
ManualCropManager is responsible for providing the user friendly UI to manually crop the additional image.	
Attributes	
`private FlutterImage image`	
`private double tuple set rectangularBorder`	
Methods	
`public void getGestureData()`	Gets user gesture data on touch screen.
`public void rerenderBorderImage()`	Rerenders border image after new change.
`public void pressedUndoButton()`	Undos users last change.
`public void pressedDoneButton()`	Finishes manual crop stage and directs to next stage.

class SettingsPageManager	
SettingsPageManager is responsible for providing the user friendly UI to show user settings options.	
Attributes	
`private FlutterImage mage`	
`private double brushSize`	
`private FlutterArea brushedArea`	
Methods	
`public void directToContact()`	Directs App to Contact page.
`public void directToRateApp()`	Directs App to Rate App functionality.
`public void directToShareApp()`	Directs App to Share App functionality.
`public void directToTeam()`	Directs App to show team members.
`public void directToFAQ()`	Directs App to FAQ page.
`public void returnToPrevPage()`	Directs App to previous page.

class TutorialPageManager	
TutorialPageManager is responsible for providing the user friendly UI to show user basic tutorial about how to use the app.	
Methods	
`public void skipTutorial()`	Skips the tutorial.

class AutoRemovePageManager	
AutoRemovePageManager is responsible for providing the user friendly UI to user to brush the region that is wanted to be *AutoRemove*'d.	
Attributes	
`private FlutterImage mage`	
`private double brushSize`	
`private FlutterArea brushedArea`	
Methods	
`public void getBrushGesture()`	Gets user data from touchscreen and adjusts brushed area.
`public void rerenderBrushedPart()`	Rerenders brushed area after it is changed.
`public void pressedAutoRemove()`	After pressing AutoRemove app is directed to apply AutoRemove
`public void returnPrevPage()`	Directs App to previous page

class MergePageManager	
MergePageManager is responsible for providing the user friendly UI to re-position and re-scale the Additional Image before AutoAdd functionality runs.	
Attributes	
`private Image : image`	
`private double tuple : addImagePos`	
`private double : addImageScale`	
Methods	
`public void pressedAutoAdd()`	Finishes this stage and directs app to apply AutoAdd functionality.
`public void repositionAddImage()`	Repositions additional image regarding the gestures of the user on the touch screen.
`public void rescaleAddImage()`	Rescales additional image according to the hand gestures of the user.
`public void rerenderAddImage()`	Rerenders additional image.
`public void returnPrevPage()`	Directs App to previous page

class ExportPageManager	
ExportPageManager is responsible for providing the user with user-friendly UI that she can command to export the *AutoShop*'d image to various Social Media platforms or save it to the local image gallery.	
Attributes	
`private Image image`	
Methods	
`public void directToInstagram()`	Directs App to run functions to share image on Instagram
`public void directToWhatsApp()`	Directs App to run functions to share image on Instagram
`public void directToTwitter()`	Directs App to run functions to share image on Instagram
`public void directToFacebook()`	Directs App to run functions to share image on Instagram
`public void saveToGallery()`	Directs App to run functions to save image to gallery
`public void returnPrevPage()`	Directs App to previous page

3.1.3 Data

class Image	
This class is responsible for keeping images in storage of the device and utilizing other classes which take images and do operations on this input image that they take.	
Attributes	
`private int width`	
`private int height`	
`private <FlutterImgType> matrix`	
Methods	
`public <FlutterImgType> getMatrix()`	Gets the matrix of the image which stores pixel by pixel.
`public void importFromGallery()`	Imports the image from the gallery.
`public void importFromCamera()`	Imports the image from the camera.

3.2 Server

In this section the subsystems are dedicated for the server part. Names of the functions and classes that are listed may alter along the development life cycle of the project.

3.2.1 Logic Tier

class 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.	
Methods	
`public Image getImage()`	Returns the image from the server.
`public void sendImage(Image image)`	Sends image to the server.

class 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.	
Methods	
`public void connectToDatabase()`	Connects database when the application starts.
`public void setDatabase()`	Sets the database from the server.

class AdderNeuralNetworkEngine	
A specific **Model** that contains a neural network trained for context-aware object addition, in other words automated image styling.	
Attributes	
`private <AdderNNType> adderNN`	
`private Asset[] assets`	
Methods	
`public Image add()`	Modify the image with add operation then returns the modified image.
`public void setAssets(Asset[] assets)`	Sets the assets.
`public Asset[] getAssets()`	Gets the assets.

class 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.	
Attributes	
`private <RemoverNNType> removerNN`	
`private Boundary boundary`	
Methods	
`public Image remove()`	Modify the image with remove operation then returns the modified image.
`public Boundary getBoundary()`	Returns boundary of this specific type of Image.

3.2.2 Data Tier

class 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.	
Attributes	
`private DatabaseManager database`	
`private CommunicationManager communication`	
`private int userID`	
Methods	
`public int getUserID()`	Returns the id of the user.
`public DatabaseManager getDBManager()`	Returns DatabaseManager object to get user's data in the server.
`public CommunicationManager getManager()`	Returns the CommunicationManager object.

class 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 and to store the cropped images like stickers in Instagram or Telegram.	
Attributes	
`private Boundary boundary`	
Methods	
`public Boundary getBoundary()`	Returns boundary of this specific type of Image.

class FinalPhotos	
These photos will correspond to the finalized and photoshopped photos received from the users that give us permissions.	
Attributes	
`private CommunicationManager communication`	
`private Image[] modifiedImages`	
Methods	
`public Image[] getModifiedImages()`	Returns the final modified images of the users.
`public CommunicationManager getManager()`	Returns the CommunicationManager object.

class BackgroundPhotos	
These photos will correspond to the background and additional photos received from the users that give us permissions.	
Attributes	
`private CommunicationManager communication`	
`private Image[] images`	
Methods	
`public Image[] getImages()`	Returns the background images of the users.
`public CommunicationManager getManager()`	Returns the CommunicationManager object.

class 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.
Attributes
Types differ according to different libraries, most of them implemented in Python therefore no specific types.
`weights`
`biases`
`neurons`
`optimizerParameters`
`numberOfLayers`

class RemoverNeuralNetwork
The essential data that represents the remover neural network, as described in AdderNeuralNetwork, will be stored here.
Attributes
Types differ according to different libraries, most of them implemented in Python therefore no specific types.
`weights`
`biases`
`neurons`
`optimizerParameters`
`numberOfLayers`

4 Glossary

`Activity`	In android an activity is a entry point for a user's interaction with the application.
`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 [5].
`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 [7].
`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 [8].
`Node`	Physical device or an execution environment in which components are executed [5].

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] “Unified modelling language.” `https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/`. [Accessed: 11- Feb- 2020].

[3] “Ieee reference guide.” `https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf`. [Accessed: 11- Feb- 2020].

[4] “Amazon web services.” `https://aws.amazon.com/`. [Accessed: 8- Nov- 2019].

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

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

[7] “Remove unwanted objects.” `https://online.theinpaint.com/`. [Accessed: 3- Nov- 2019].

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