Bilkent University

Department of Computer Engineering

Senior Design Project

DeePaint : Smart Photo Editing on Your Pocket

Final Report

Project Group Members:

Yavuz Bakman - 21703309

Hande Sena Yılmaz - 21703465

Alperen Öziş - 21703804

Zübeyir Bodur - 21702382

Duygu Nur Yaldız - 21702333

Supervisor: Uğur Doğrusöz

Jury Members: Tağmaç Topal and Erhan Dolak

Table of Contents

Final Report

DeePaint: Smart Photo Editing on Your Pocket

1. Introduction

During the last decade, social media applications have become widely used and part of our daily lives due to the easy accessibility of mobile devices and the increasing quality of their cameras. All around the world, people take photos constantly and share them on social media. For instance, 1074 photos are uploaded on Instagram every second [1]. While sharing photos, people have an urge to get rid of the imperfections in the photo such as a stranger in the background of a selfie or a car in a beautiful scene.

However, applying these changes requires knowledge and experience in photoshop. A regular person can not achieve to remove an object from a photo in a realistic way. Nevertheless, this became a need for regular people on a daily basis. Therefore, applications that make these adjustments automatically have gained more and more demand. Those applications with a user-friendly interface, fast modification speed, and low error rate in modification (i.e. more realistic results) are preferred by the users.

DeePaint places itself just to this point. It is an easy-to-use, fast and fancy application that provides its users a variety of image manipulation tools. DeePaint gains its power by using state-of-the-art deep learning techniques. For example, we employed image segmentation methods in our application, which enables the users to select the object-to-remove with only one screen touch. Such fast and easy image manipulation methods make DeePaint one step ahead of its relative applications.

In this report, we first provide an overview of functional and non-functional requirements. Afterwards, we explain the final architecture and design details of the project. Then, we inspect development and testing details as well as maintenance plans. Finally we present other project elements and future work for the project.

2. Requirements Details

2.1. Functional Requirements

In this section, functional requirements of DeePaint are discussed in terms of both system functionality and user functionality.

System Functionality

- The system should require the user to sign in/up with an email and a password.
- The system should ask the user's permission to reach the gallery and use camera functionality.
- The system should display the images in the gallery.
- The system should receive an image to be edited as input from the user and display it.
- The system should segment the objects in the selected image and display them.
- The system should receive the position and size information of the areas that are empty in the image (i.e object is removed).
- The system should convert the given image to drawing.
- The system should change the style of an image to selected target style.
- The system should display the edited version of the image as well as the original one.
- The system should fill the empty areas in the image in a natural way.
- The system should be connected to a server.
- The system should send images to the server in order to process machine learning algorithms (e.g. segmentation, auto-fill).
- The system should receive processed images from the server.
- The system should receive segmentation information from the server.
- The system should provide an option to save a processed image and should save an image to the local gallery if the user decides.

- The system should provide an option for the user to reset the changes in the editing image.

User Functionality

- The user can allow the DeePaint to reach the user gallery and the camera.
- The user can select a photo from the local gallery or can take a new photo and upload it to the system to edit.
- The user can view the pre-segmented objects in the image.
- The user can draw the boundaries of the place-to-remove in the image.
- The user can remove any object in the image that is selected by either segmentation or drawing.
- The user can upload two images, one for styling target and one for target image. Then the user receives the target image in the style of the other image.
- The user can change the style of an image similar to the provided style types.
- The user can reset the changes in the image.
- The user can view the original version of the image as well as the edited version.
- The user can save the edited image to the local gallery.

2.2. **Non-Functional Requirements**

Performance

While the user is choosing what to do with his photograph, (What to remove from the scene etc.) our program should be smooth with its reactions, namely in 100ms. Once the user has inputted his desire to our app, we should return the edited photo back to the user in at most 20 seconds.

Privacy

Since users will upload their personal photographs to our app, we must ensure every user can only access photographs uploaded by themselves. We will render this possible via basic authorization.

Usability

Since one of the rationales of our project is to rescue our users from dealing with the difficulty of the traditional photoshop apps, our app should be easy to use. A new user should be able to adapt to it in 2 minutes.

Extendability

Given the highly increasing number of innovations in AI in recent years, our design patterns should allow us to easily add a new feature to our application or improve the existing ones in terms of accuracy, usability, etc.

Accuracy

Although there is not a universal metric to measure accuracy of our tasks numerically, our app should be able to output realistic images, to the extent of tricking a FakeorNot robot.

Security

Since the actual editing of the photo will be done in the cloud servers and not in the local workspace of the user, we cannot allow bots to use our application and keep our servers busy. Therefore we will use basic authentication before allowing users to utilize our app.

2.3. Pseudo Requirements

- DeePaint mobile application is written in Java programming language, and it is on the Android platform.
- Since the mobile application is in the Android platform, Android Studio IDE is used for development.
- For implementing deep learning features of the application, a server-side is required. Since those libraries, such as PyTorch, TensorFlow, Keras, Theano, are libraries that all can be used in Python, the server-side is implemented in Python.
- Object-oriented programming is used throughout the project.

- GitHub is used for version control and collaboration, as well as opening issues if necessary.
- The mobile application is free to use, and there are no in-app purchases. Hence, all of the users are able to use the features without making any payment.
- The application requires the user to have an internet connection since processing of the image will be carried out in the cloud.

3. Final Architecture and Design Details

3.1. Overview

In this section, we provide the overall architecture and design details of DeePaint. DeePaint adopts client-server architecture since it requires running machine learning applications in strong machines. The details of the proposed software architecture are explained in the following sections.

3.2. Subsystem Decomposition

The DeePaint application has two main components: client and server. The Client is the part of the application responsible for displaying images, receiving the editing prompts from the user and showing the final results, etc. The Server is responsible for applying the requested changes to target images by running machine learning algorithms and returning the edited images to the client.

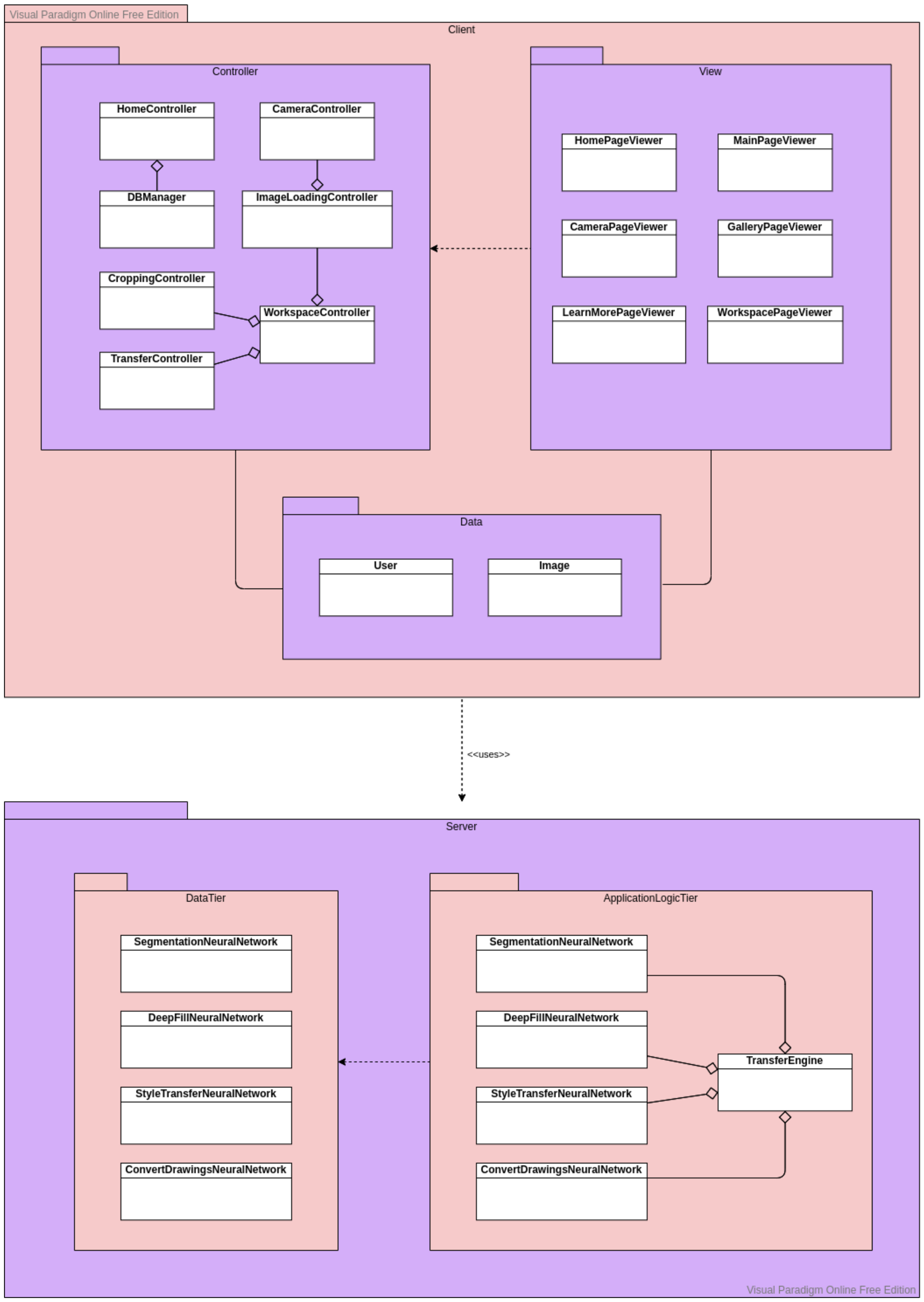


Figure 1: Subsystem Decomposition

3.3. Hardware/Software Mapping

DeePaint is a mobile application that runs on the Android operating system. Both the hardware and software are compatible with varying Android releases. However, since the computational power of a smartphone is not strong enough to perform machine learning tasks, necessary parts of the application, namely the Server Tier, are deployed on Google Colab.

The communication between these two systems will be made through HTTP requests. The robustness of Google Colab System will solve the computing problem mentioned above.

For example, the client-side, which is implemented in Java, will send an HTTP POST request to Google Colab each time the user uses a smart feature. Then, the information will be processed in the server, which is implemented in Python. Then, the server-side will send a response for this HTTP request, which will be basically the edited version of the image.

Below is the corresponding component diagram for this hardware/software mapping.

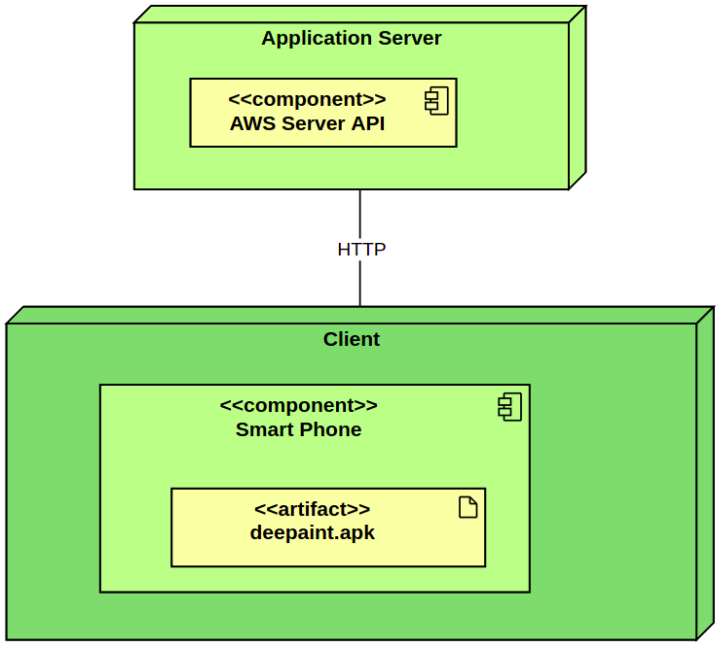


Figure 2: Hardware/Software Mapping

3.4. Persistent Data Management

In our system, we deploy client-server architecture and we care about the privacy of the users' data. Therefore, we do not store the users' image data on our server. We just send the necessary requests to the server which is differentially private and get the output which is also private.

We store the generated images on the local storage which is the users' phone. We store authentication information (i.e. emails, passwords) on an online database named Firebase.

3.5. Access Control and Security

Users can only use DeePaint by creating an account. Passwords of the users are encrypted so that they cannot be known by any other person. The images processed by the application are stored on their local device, instead of a database. However, in order for the users to use the application to its fullest, they need to permit the application to access the following;

- Storage
- Camera
- Gallery

3.6. Global Software Control

We'll go through how the whole system is managed on a global basis in this part. We go through how requests are made and how subsystems communicate with one another. Finally, several concurrency difficulties are addressed.

There are server and client side and totally 3 phases in our system which are request, processing and response phases.

In the request phase, the user selects the image from its local storage and opens the edit page. In the edit page, the user decides the operation. After the decision, the user clicks the process button and sends the operation type, the image and how the operation is done.

In the processing phase, the server gets the necessary input from the client and utilizes the neural networks to process the input. There might be multiple requests from different users. In this case, the server has a priority algorithm to select which operation is done first.

In the response phase, the output is sent to the client from the server side. This processing phase and response phase should be fast because of the usability concerns. We want to bring concurrency to this phase, as well as the request phase, by finding an appropriate port configuration that minimizes the client-side delay.

3.7. Boundary Conditions

Boundary conditions for DeePaint can be categorized under three conditions which are initialization, termination and failure.

Initialization

Users of DeePaint need to download our app from the google play store or find an apk version. They need to sign up and then give our app access to their gallery and/or their camera if they want to retrieve images from there. Internet connection will be needed to use DeePaint since it connects to cloud servers for different tasks.

Termination

When the DeePaint gets terminated, any running process on the device and on the server will also be terminated. If there are any unsaved changes, they will be lost. Memory that is saved by DeePaint will be released after the termination.

Failure

There are several failure cases a user can experience while using DeePaint. If there is a failure while taking a photo via camera, the user will be redirected to the main page where he can either try with the camera again or choose another photo from his gallery.

If there is a failure while sending a request to the server due to no or weak internet connection, the user will be notified that there is a problem with the internet connection and will be redirected to the edit page without losing information about the photo he sent to the server.

If the request is sent to the server but there is a problem in the server and cant process the image at that moment, the user will be notified about the problem in the server and asked to try again later.

If there is no response from the server after passing a specified timeout duration, the user will be notified about the problem and asked to try again later. He can also send a report to the developers about the problem.

If there is not enough free space in the device’s disk for the saving of an edited image or a figure, the user will be notified and asked to open up space in the device.

4. Development/Implementation Details

4.1. Object Design Trade-offs

4.1.1. Usability vs Functionality

The main purpose of DeePaint is to ease the photoshop process and thus usability is maybe the most important design goal we have. However this doesn’t stop us from offering complex features. We are still able to manage such complex things like filling the background of the removed object but we are doing it in a way that minimum user effort is involved thanks to the automatization coming with deep learning techniques.

4.1.2. Compatibility vs Extensibility

Computer vision is an area where almost every day there is a new advancement. So there are many aspects of our project where we can improve our current features or add new ones. Therefore extensibility is something we have to achieve if we want to keep our app up-to-date and compete in the market. Compatibility is also an important factor to reach a higher number of users. Though we will be releasing only on Android for now, we can easily go live on other operating systems thanks to java being an environment friendly programming language. We just need to change OS dependent features while keeping the underlying structure the same to export our project, say iOS. Using java for the app and Google Colab servers for the backend will help us to easily extend new features while keeping main functionalities compatible with different operating systems.

4.1.3. Space vs Time

We will be using big models which would be a burden to store locally. It also wouldn't be fast to execute since they require high computation power which most of the mobile devices lack. Keeping our models in cloud servers which offer us strong GPUs will enable us to both avoid the redundant data storage and achieve faster computations. Though there will be a time overhead sending and receiving requests to/from servers, it is an overhead we can tolerate because the benefits will outdo the disadvantages.

4.2. **Packages and Class Interfaces**

4.2.1 client.view

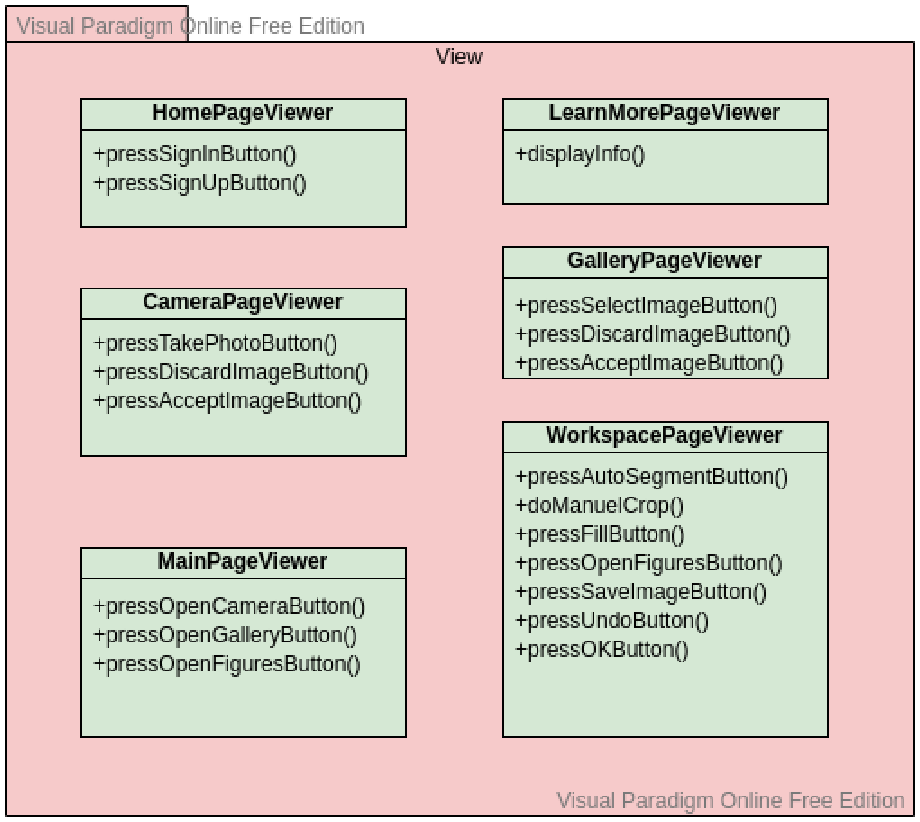

Visual Paradigm Online Free Edition
View
HomePageViewer
+pressSignInButton()
+pressSignUpButton()
LearnMorePageViewer
+displayInfo()
GalleryPageViewer
+pressSelectImageButton()
+pressDiscardImageButton()
+pressAcceptImageButton()
CameraPageViewer
+pressTakePhotoButton()
+pressDiscardImageButton()
+pressAcceptImageButton()
WorkspacePageViewer
+pressAutoSegmentButton()
+doManuelCrop()
+pressFillButton()
+pressOpenFiguresButton()
+pressSaveImageButton()
+pressUndoButton()
+pressOKButton()
MainPageViewer
+pressOpenCameraButton()
+pressOpenGalleryButton()
+pressOpenFiguresButton()
Visual Paradigm Online Free Edition

Figure 3: Class diagram for client.view package

HomePageViewer

HomePageViewer is responsible for displaying the welcome page when the application is started and allowing users to sign in/signup for the app.

Methods:

- protected void pressSignInButton(): Directs app to sign in page.
- protected void pressSignUpButton(): Directs app to sign up page.

MainPageViewer

MainPageViewer is responsible for showing the initial actions a user can take, i.e. opening the camera, displaying saved figures, or exporting an image from the gallery.

Methods:

- protected void pressOpenCameraButton(): Directs app to Camera Page and opens the camera for taking photos.
- protected void pressOpenGalleryButton(): Directs app to Gallery Page and lists the pictures already exists in the users' gallery.

CameraPageViewer

CameraPageViewer is responsible for displaying the camera and allowing the user to take photos, which they will be editing in the workspace view.

Methods:

- protected void pressTakePhotoButton(): Takes the photo and directs it to the state where users decide to accept or discard the image and take a new one.
- protected void pressDiscardImageButton(): Discards the taken image and directs to the photo-taking page.
- protected void pressAcceptImageButton(): Accepts the taken image and directs to the main editing page.

GalleryPageViewer

GalleryPageViewer is responsible for showing the initial actions a user can take, i.e. opening the camera, displaying saved figures, or exporting an image from the gallery.

Methods:

- protected void pressSelectImageButton(): Selects the already existing image from the gallery of the user and directs it to the state where users decide to accept or discard the image and choose a new one.
- protected void pressDiscardImageButton(): Discards the selected image and directs to the gallery page to choose a new one.
- protected void pressAcceptImageButton(): Accepts the selected image and directs to the main editing page.

LearnMorePageViewer

LearnMorePageViewer includes the information about the app and displays the buttons, “Contact Us”, “FAQ” and “Team”. The “Contact Us” button will redirect the user to the DeePaint website. The FAQ button will redirect the FAQ page of the DeePaint application. Finally, the Team button will display a modal that shows the names of developers who contributed to this project.

Methods:

- protected void displayInfo(): Displays all the information included in LearnMorePage by directing to “Contact Us”, “FAQ” or “Team” based on the preference of the user.

WorkspacePageViewer

WorkspacePageViewer is the main page where the users will interact with the photo and edit them. It includes GUI items for displaying the saved figures, segmenting the image, manually editing the image, saving the edited images, and more for the manipulation of the image.

Methods:

- protected void pressAutoSegmentButton(): Starts auto segmentation on the image and displays the segmented figures afterward.
- protected void doManuelCrop(): Allows users to manually crop the figures by hand instead of auto segmentation, then display the segmented figure.
- protected void pressFillButton(): Fills the removed areas smoothly after figures are removed by using the deep fill technique.

- protected void pressConvertDrawingButton(): Converts the current photo in the style of a hand drawing.
- protected void pressTransferStyleButton(): Transfers the style of the one image to the other one.
- protected void pressSaveImageButton(): Saves the edited image on the main editing page to users' gallery.
- protected void pressUndoButton(): Undos the changes that have been made on the edited image one by one in each press.
- protected void pressOKButton(): Approves the changes made so far for the image editing and manipulation.

4.2.2 client.controller

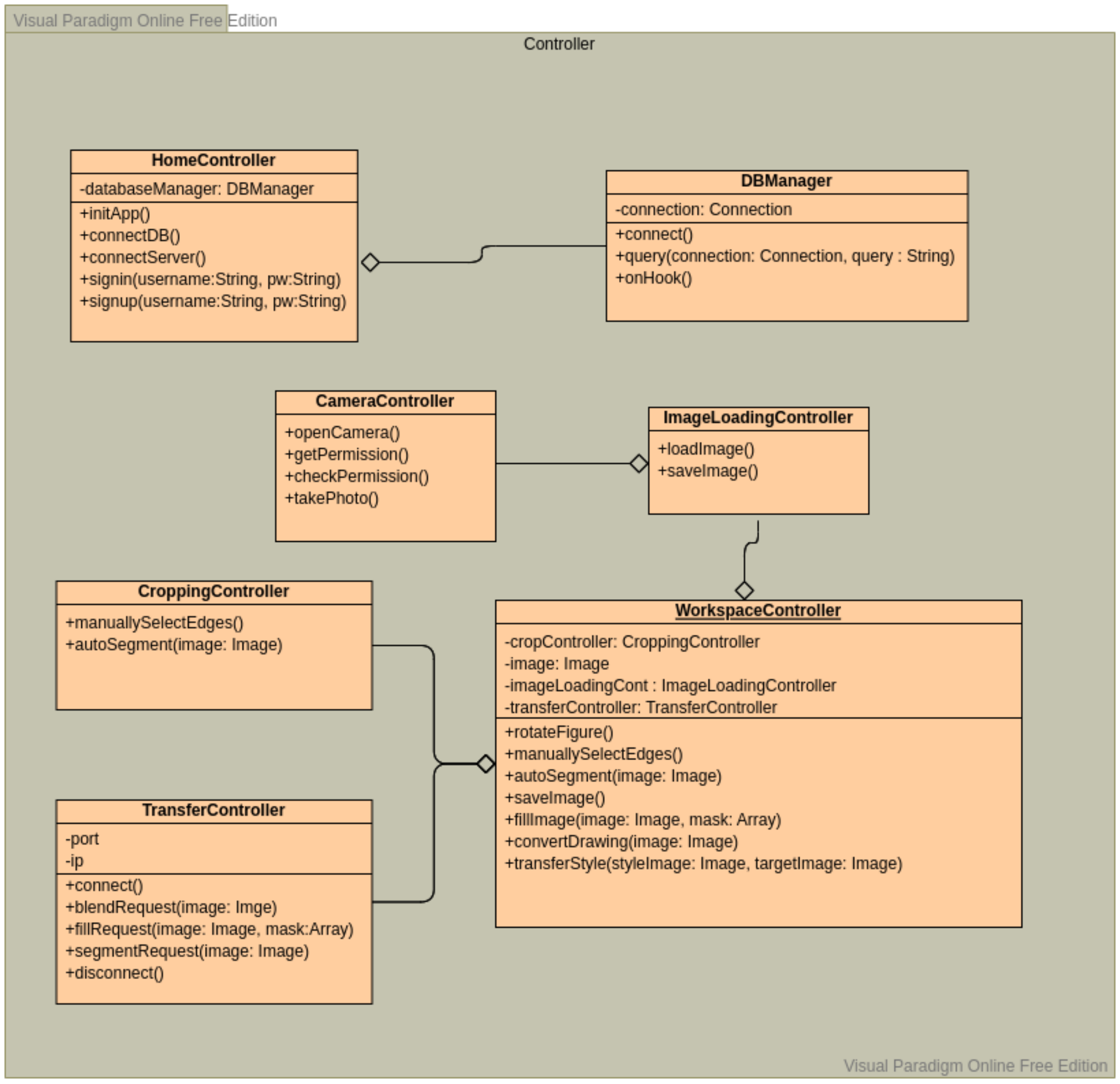

Visual Paradigm Online Free Edition
Controller
HomeController
-databaseManager: DBManager
+initApp()
+connectDB()
+connectServer()
+signin(username:String, pw:String)
+signup(username:String, pw:String)
DBManager
-connection: Connection
+connect()
+query(connection: Connection, query : String)
+onHook()
CameraController
+openCamera()
+getPermission()
+checkPermission()
+takePhoto()
ImageLoadingController
+loadImage()
+saveImage()
CroppingController
+manuallySelectEdges()
+autoSegment(image: Image)
WorkspaceController
-cropController: CroppingController
-image: Image
-imageLoadingCont : ImageLoadingController
-transferController: TransferController
+rotateFigure()
+manuallySelectEdges()
+autoSegment(image: Image)
+saveImage()
+fillImage(image: Image, mask: Array)
+convertDrawing(image: Image)
+transferStyle(styleImage: Image, targetImage: Image)
TransferController
-port
-ip
+connect()
+blendRequest(image: Imge)
+fillRequest(image: Image, mask:Array)
+segmentRequest(image: Image)
+disconnect()
Visual Paradigm Online Free Edition

Figure 4: Class diagram for client.controller package

TransferController, DBManager, and WorkspaceController are singleton classes, and WorkspaceController is the façade class.

WorkspaceController

This class manages all of the operations related to the workspace the user is interacting with, such as image editing, selecting a figure, saving the image into the gallery, etc.

This class is also singleton, as façade classes usually have only one instance at any given time. This is also true for the DeePaint.controller.WorkspaceController as well, there will be only one workspace.

<u>Attributes:</u>

- private CroppingController cropController: The cropping controller the workspace will use. It will provide operations for retrieving manually selected edges and retrieving the figure.
- private Image image: The image the workspace is currently working on.
- private ImageLoadingController imageLoadingCont: The image loading controller will manage operations related to the gallery, such as saving or loading an image locally.
- private static TransferController = null: The transfer controller will send the necessary requests to AWS for segmentation, blending, and filling (inpainting). It is a singleton class since connection classes are usually singletons.

<u>Methods:</u>

- public void manuallySelectedEdges(): Encapsulated version of the `CroppingController.manuallySelectedEdges()` method.
- public void autoSegment(Image image): Sends the AWS request for segmenting the image in the workspace, using the workspace's `TransferController`. After receiving the response, processes the `ArrayList<Image> binaryEdges`, so that at the end, those edges are placed at the top of the image in the View (not in the actual image in the controller, as we want to select those segmented lines and do operations with them, such as filling, saving the figure, etc.).
- public void saveImage(Image image): This method also encapsulates `ImageLoadingController .saveImage()`.

- public void trasferStyle(Image sourceImage, Image destImage) : Sends the request to the server, so that the source image's style is transferred to the destination image.
- public void fillImage(Image image, byte[] mask): Sends the request to the server, so that the image in the workspace is inpainted by the given mask, which will be either provided by the user or retrieved from a segmentation.
- public void convertDrawing(Image image): Sends the request to the server, so that the image in the workspace is converted to hand drawing style.

TransferController

This class will manage the connection between the AWS and the application. It will be also responsible for sending the HTTP requests to the server and receiving the HTTP response.

Attributes:

- private String port: The port number of the connection.
- private String IP: The IP address of the connection.

Methods:

- public void connect(): Establishes the connection between the AWS and application, using the port number and IP address of this singleton instance.
- public void disconnect(): Removes the connection between the AWS and the application.
- public Image convertDrawingRequest(Image dest): Sends a convert to drawing request to the server to process the image
- public Image transferStyleRequest(Image dest, Image src): Sends a transfer style request to the server.
- public Image fillRequest(Image dest, byte[] mask): Sends an image filling request to the server to execute.
- public Image segmentRequest(Image dest, byte[] mask): Sends a segmentation request to AWS to execute. Returns an image that represents the segmented regions.

CroppingController

This class is responsible for manual cropping operations.

Methods:

- public void manuallySelectEdges() : Opens the palette in the View so that the user can manually select their own edges.
- public void autoSegment(Image image) : Automatically segments the image

ImageLoadingController

This class is responsible for file operations on client.data.Image class, which is a not-so-straightforward implementation of WhatsApp Stickers in Java.

<u>Methods:</u>

- public Image loadImage(String imageUri): Returns a selected figure from the gallery, where the selected figure is specified by `figureUri`.
- public void saveImage(Image img): Save a given figure f, to the constant path DeePaint app will use: `“~/DeePaint/Workspace/Images/image_file_name.image_extension”`.

CameraController

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

<u>Methods:</u>

- public void openCamera(): Initiates the camera of the device of the user based on the permission to let the user take a picture.
- public void getPermission(): Gets the users’ permission for asking the device to initiate the camera for taking pictures.
- public void checkPermission(): Checks whether the permission that the user gives is stable before connecting to the camera of the device.
- public void takePhoto(): If the camera is reached by the application, this allows users to take pictures of their preference to further use it for editing.

HomeController

This class will manage the user log-in and sign-up operations and will use the DBManager to access the database of users. The DBManager is a singleton that HomeController will use.

Attributes:

- private static DBManager databaseManager = null: This is the singleton instance is responsible for establishing a connection between the database and the mobile device, using the JDBC driver.

Methods:

- public void initApp(): Communicates with the view classes so that the application is initialized.
- public void connectDB(): Establishes the connection between database and application using a hardcoded connection string, which consists of a URL, DB user, and DB password.
- public void connectServer(): Establishes the connection between the AWS and application (?)
- public void signIn(String username, String pw): Tells the DBManager to execute such a query so that the user with the given information exists, and credentials are correct. After that, the user will have their own session as the application is open.
- public void signUp(String username, String pw): Tells the DBManager to execute such a query so that the user with the given information does not exist, and a row in the User table is created. After that, the user will be asked to sign in, so this method will also redirect the view in such a way.

DBManager

This singleton class is responsible for establishing a connection and then storing it, using the JDBC driver for MySQL. It can also execute any SELECT query, and then retrieve the results.

Attributes:

- private Connection connection: The connection established will be stored in this attribute.

Methods:

- public ResultSet query(Connection c, String query): Executes any MySQL query using the connection provided. If there is a result, it is returned. If there is any MySQL exception, they are also handled.

- public ArrayList<T> processResultSet(ResultSet rs): A ArrayList<T> processResultSet() method will be implemented if necessary for SELECT queries, that is if the database gets larger.

4.2.3. client.data

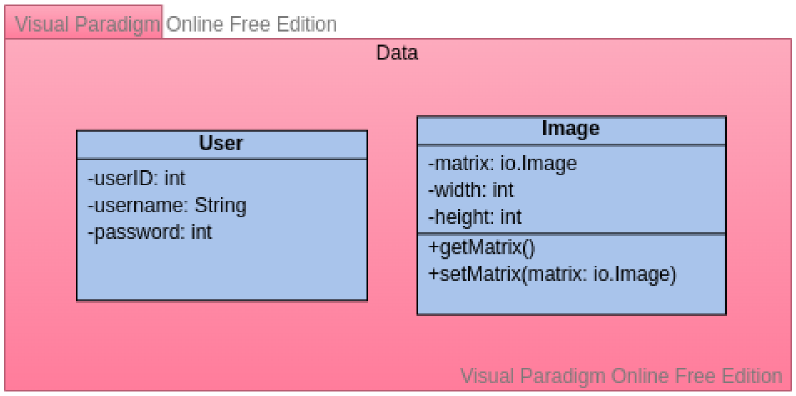


Figure 5: Class diagram for client.data package

User

This class represents the User table of the application’s database as entities.

<u>Attributes:</u>

- private int userID: Stores the id of the user.
- private String username: Stores the name of the user.
- private int password: Stores the password of the user.

Image

This class represents the edited images as an image object for local storage.

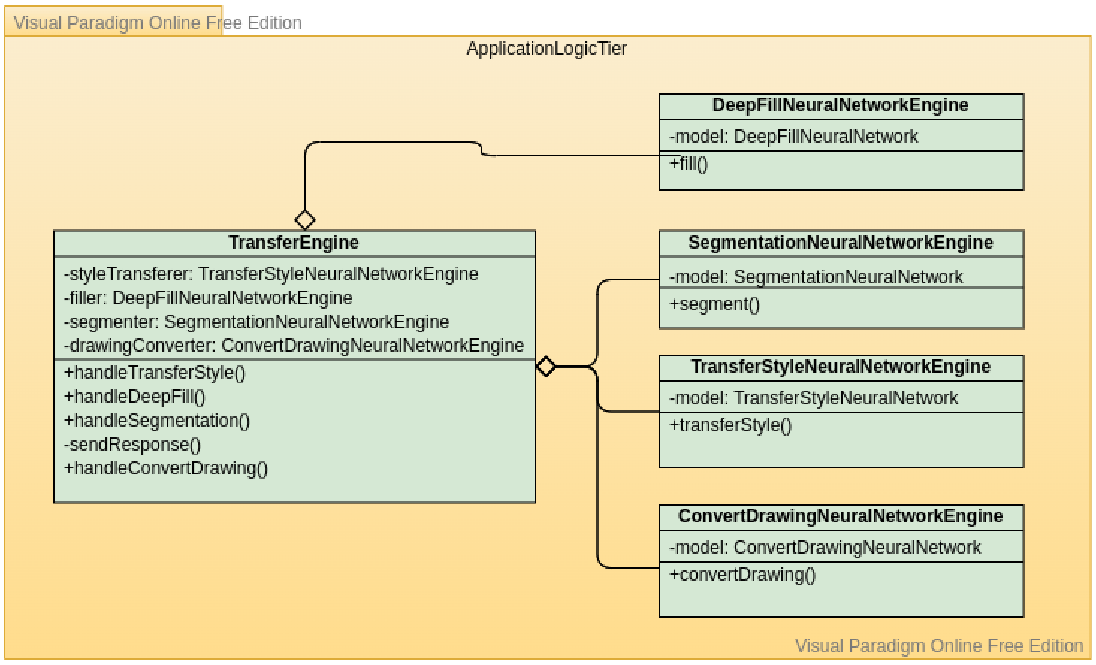
<u>Attributes:</u>

- private int width: Stores the width of the image.
- private int height: Stores the height of the image.
- private io.Image matrix: Stores the matrix representation of the image pixel by pixel.

Methods:

- public io.Image getMatrix(): Returns the matrix representation of the edited image.
- public void setMatrix(io.Image matrix): Sets the matrix representation of the edited image.

4.2.4. server.applicationLogic



Visual Paradigm Online Free Edition
ApplicationLogicTier
DeepFillNeuralNetworkEngine
-model: DeepFillNeuralNetwork
+fill()
TransferEngine
-styleTransferer: TransferStyleNeuralNetworkEngine
-filler: DeepFillNeuralNetworkEngine
-segmenter: SegmentationNeuralNetworkEngine
-drawingConverter: ConvertDrawingNeuralNetworkEngine
+handleTransferStyle()
+handleDeepFill()
+handleSegmentation()
-sendResponse()
+handleConvertDrawing()
SegmentationNeuralNetworkEngine
-model: SegmentationNeuralNetwork
+segment()
TransferStyleNeuralNetworkEngine
-model: TransferStyleNeuralNetwork
+transferStyle()
ConvertDrawingNeuralNetworkEngine
-model: ConvertDrawingNeuralNetwork
+convertDrawing()
Visual Paradigm Online Free Edition

Figure 6: Class diagram for server.applicationLogic package

SegmentationNeuralNetworkEngine

This class is responsible for the Segmentation task.

<u>Attributes:</u>

- private model SegmentationNeuralNetwork: This attribute stores the segmentation neural network parameters.

<u>Methods:</u>

- public ArrayList<int> segment(Image i): This method makes the segmentation.

DeepFillNeuralNetworkEngine

This class is responsible for the Deep Fill task.

<u>Attributes:</u>

- private model DeepFillNeuralNetwork: This attribute stores the deep fill neural network parameters.

<u>Methods:</u>

- public Image fill(Image i, ArrayList<int> mask): This method makes the neural network works and fill the selected area

ConvertDrawingNeuralNetworkEngine

This class is responsible for the converting to drawing style task..
<u>Attributes:</u>

- private model ConvertDrawingNeuralNetwork: This attribute stores the neural network parameters.

<u>Methods:</u>

- public Image converDrawing(Image i): This method makes the neural network work and applies the converting to drawing task with the given parameters.

TransferStyleNeuralNetworkEngine

This class is responsible for the converting to drawing style task..

Attributes:

- private model TransferStyleNeuralNetwork: This attribute stores the neural network parameters.

Methods:

- public Image transferStyle(Image i, Image d): This method makes the neural network work and applies the transfer styling to the destination image.

TransferEngine

This class gets the request from the client and make the appropriate models work to handle the request and send the response to the client

Attributes:

- private blender ConvertDrawingNeuralNetworkEngine: This attribute provides the methods for the convert drawing task.
- private blender TransferStyleNeuralNetworkEngine: This attribute provides the methods for the transfer styling task.
- private filler DeepFillNeuralNetworkEngine: This attribute provides the methods for the deep filling task.
- private filler SegmentationNeuralNetworkEngine: This attribute provides the methods for the segmentation task.

Methods:

- public Image handleConvertDrawing(Image i): This method handles the convert drawing task
- public Image handleTransferStyle(Image i, Image d): This method handles the transferring the style task
- public Image handleDeepFill(Image i, ArrayList<int> mask): This method handles the deep filling task.
- Public ArrayList<int> mask handleSegmentation (Image i): This method handles the segmentation task
- public void sendResponse(Image i, Figure f, ArrayList<int> mask): This method sends the output of neural networks into the client

4.2.5 server.data

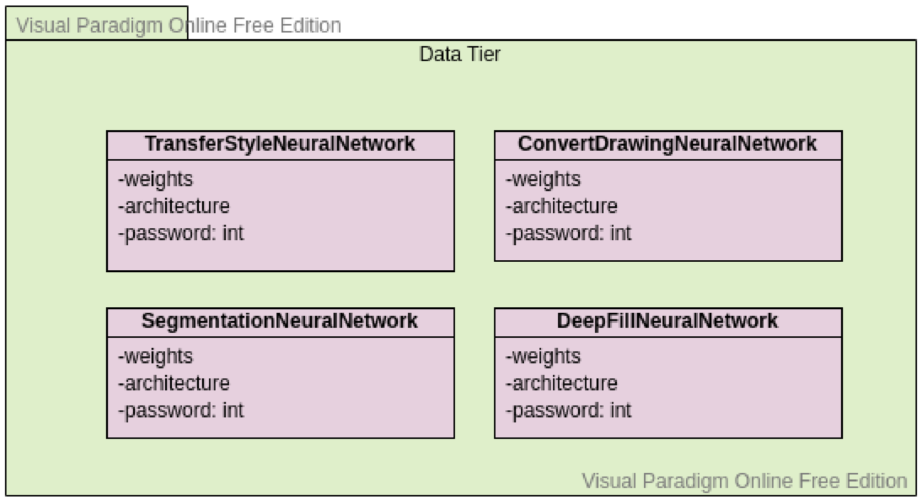


Figure 7: Class diagram for server.data package

SegmentationNeuralNetwork

This is the NN model responsible for the segmentation task.

Attributes:

- Private Weights: This represents the learned parameters by the model to do the task.
- Private Architecture: This represents the underlying architecture of the segmentation NN.
- Private Password: This is the password needed to access the models.

DeepFillNeuralNetwork

This is the NN model responsible for the deepfilling task.

Attributes:

- Private Weights: This represents the learned parameters by the model to do the task.
- Private Architecture: This represents the underlying architecture of the deepfill NN.
- Private Password: This is the password needed to access the models.

TransferStyleNeuralNetwork

This is the NN model responsible for the transferring style task.

Attributes:

- Private Weights: This represents the learned parameters by the model to do the task.
- Private Architecture: This represents the underlying architecture of the transferring style NN.
- Private Password: This is the password needed to access the models.

ConvertDrawingNeuralNetwork

This is the NN model responsible for the conversion to the drawing style task.

Attributes:

- Private Weights: This represents the learned parameters by the model to do the task.
- Private Architecture: This represents the underlying architecture of converting to drawing NN.
- Private Password: This is the password needed to access the models.

5. Testing Details

5.1. Functional and non-Functional Testing

We needed to make sure that every point in the functional and non-functional requirements were met in the DeePaint application to ensure that our project delivers its users the promised features. This testing method enabled us to find the correct UX for DeePaint users.

5.2. Continuous Integration

As we had a continuous integration development pipeline, we were able to define our mistakes very early enabling us to handle them before there were any big and irreversible consequences of these mistakes. We tested different components of our application as separate as from each other and followed a build up method. We did not continue implementation of that component until we assured that it was working as intended.

5.3. Testing the UI

User interface is an important aspect of our project, hence we paid lots of attention while developing this side to make sure it was working in the correct way. To make sure that our UI is working as intended, we used our application at different times and tried to locate any possible errors, or any weak aspects. This method enabled us to detect and solve UI mistakes and missing parts in a fast and easy manner.

5.4. Server Performance and Response

As DeePaint is an application centered around image manipulations tasks, we need to send and receive images to and from the server. Consequently, delivery time is an important aspect of the project. Long image retrieval makes the users unhappy. Hence, we sent and received back different images with different sizes and resolutions to and from the server in order to ensure delivery time is smaller than the desired upper limit. Also, we need to process images using neural networks on the server side. Some of the neural networks lasted around 1 minute to process desired tasks. Therefore, we moved to the other networks that do the same task in a shorter amount of time. This part of the project was very crucial to us since it is directly related to the user experience.

6. Maintenance Plan and Details

6.1. Server Maintenance and Optimization

Since our application makes use of state-of-the-art neural network models for different kinds of image manipulations tasks, it is highly probable that new and better alternative neural network architectures that do the same task as ours will be released in the future. Therefore, adapting to those kinds of changes is crucial for our application in order to keep up with current technology levels.

On the other hand, in terms of hardware requirements, currently we use Google Colab to run our machine learning algorithms. However, in the future as the number of users increases, we will require much more computational power. Therefore, we might need to switch to a stronger GPU machine or a completely new server provider company. In order to determine this, we will need to regularly monitor server performance in terms of response time and ability to cope with multiple simultaneous requests.

6.2. Database Maintenance

We use Firebase to store our user login information (i.e emails and passwords). Therefore, as the number of users increases, maintaining this database will be much more important. We will need to do backup regularly, query optimization, and efficient indexing.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

Public Health

Our project is a mobile application so people are required to use their mobile phones and spend some time applying changes in the image. Using mobile phones excessively causes some physical problems such as neck and eye problems [2]. This is not special to our application but as the program increases the amount of time spent on mobile phones, we should consider this factor. Therefore, we should design a smooth user interface such that people can make photoshop easily and do not have to spend excessive time. However, the most important health issue we care about is psychological effects. According to the latest research, social media causes psychological problems such as anxiety and depression [3]. People compare themselves with other people and use photo-shop applications to make themselves more beautiful or handsome. They try to make their photos better to get more likes. Our program is designed for making photographs more desired. However, we do not want to connect our application to social media apps. For instance, after editing the photograph, there is no option such as ‘post it on Instagram’. We want people to change their photographs because only they want. Therefore, we do not encourage users to post their edited photos on social media.

Public Safety

As our application has the capability of generating realistic fake images, a malevolent person can abuse it. To prevent it, we want to add a mark to our generated images saying that it is generated by DeePaint. Therefore, people can easily understand whether the image is fake or not. Furthermore, as the program takes image input from the user and sends it to a server to process it, we have to care about the security of the data because it is the online environment. Therefore, we use the servers of Google Colab which is a well-known company that cares about the privacy and security of the users’ data.

Public Welfare

Our program is free to download. Therefore, there is no effect on public welfare.

Global Factors

Our project is designed for each person around the world. Therefore, the language of the app should be English. Also, in the future, we may put additional languages as options.

Cultural Factors

Our program has no correlation with cultural factors.

Social Factors

People generally use photo-shop applications for social media. However, we do not want to increase the amount of time spent on social media. Therefore, we do not connect our application with any social media app.

	Effect Level	Effect
Public Health	8	Physical and psychological problems
Public Safety	5	Data privacy and Fake News
Public Welfare	0	-
Global Factors	4	The program’s language should be English
Cultural factors	0	-
Social Factors	3	Increase the amount of time spent on social media

Figure 8: Evaluation criteria for factors in engineering design

7.2. Ethics and Professional Responsibilities

In our application data privacy is very important. Therefore, we follow the data privacy code [4]. Another important factor in our program is the biases in the neural networks. Neural networks can carry implicit biases against some ethnicities, cultures etc. However, there are also well-studied methods to reveal biases and prevent biases. We will ensure that our neural networks used in photo-shop do not carry such biases. The libraries, APIs will be used according to their license agreements. Furthermore, we will follow the ACM code of ethics [5].

7.3. Judgements and Impacts to Various Contexts

Judgment Description:	DeePaint must watch for data privacy for its users. Since the users upload their private information (i.e) photos to edit them, we must ensure our users that nobody can access their

	photos.	
	Impact Level	**Impact Description**
Impact in Global Context	10/10	Privacy would allow DeePaint to provide trustworthy services to worldwide users.
Impact in Economic Context	7/10	Since privacy is very important for us, we need to work with trustworthy server providers, which have a certain cost. We needed to consider different options to choose the best fitting privacy / cost ratio efficient service provider.
Impact in Environmental Context	0/10	Does not have an environmental impact
Impact in Societal Context	10/10	Prioritized privacy will increase user trust in DeePaint, which is an important aspect.

Figure 9: Privacy Judgement

Judgment Description:	After editing the photograph, we do not offer an option such as ‘post it on Instagram’	
	Impact Level	**Impact Description**
Impact in Global Context	2/10	Since we do not encourage users to post their edited photos on social media, this impact might become global.
Impact in Economic Context	0/10	Does not have an economic impact
Impact in Environmental Context	0/10	Does not have an environmental impact
Impact in Societal Context	10/10	People compare themselves with other people and use photo-shop applications to make themselves more beautiful or handsome. We want people to change their photographs because only they want. Therefore, we do not encourage users to post their edited photos on social media.

Figure 10: Social media share option judgement

7.4. **Teamwork Details**

7.4.1 Contributing and functioning effectively on the team

When collaboration doesn't feel organic, it can seem incredibly tiresome. When practiced effectively, however, as a team we are aware that the importance of teamwork in software development, or any type of business for that matter, is paramount. Thus, we have divided the workload as effectively as possible to drive a functioning project plan.

As our application depends on neural networks and the working principle is stochastic, we first want to be sure that current neural network technology will satisfy our expectations. Therefore, we divide our team into 2 parts: people mostly dealing with neural networks and people mostly working on the application development side. However, this separation is not strict, and each member is welcome to participate in both parts.

We have divided the entire work of the project into small work packages. According to this, we have 12 packages: Analysis Report, High-Level Design Report, Implementation of Neural Networks for segmentation and Image Generation Tasks, Development of User Interface and Backend in Android, Connecting Android Application with Neural Networks for Object Removing Task Only, Demo in Desktop, Low-Level Design Report, Add Other Functionalities into Neural Networks, Add Other Functionalities into Back-end and Front-end, Connect server-application, Mobile Phone Demo, Final Report. So far, Analysis Report, High-Level Design Report, Implementation of Neural Networks, some other Functionalities of Back-end and Front-end, and Connecting server-application for Demo in Desktop have been completed.

For reports all members are involved and contribute to different sections. For Demo in Desktop, Duygu, Yavuz, and Alperen contribute to the implementation of Neural Networks for segmentation and Image Generation. Zübeyir and Hande contribute to it in the application development side with the user interface, server connection, and more.

7.4.2 Helping creating a collaborative and inclusive environment

Productivity, effective communication, exchange of ideas, and enhanced performance are the key qualities of a functioning project team. Moreover, this helps create unity within the team through inclusiveness and collaboration.

To achieve this, we have communicated on a weekly basis using synchronous environments. We have also communicated persistently in asynchronous environments.

The synchronous environments used are the following;

- Online meetings, through Zoom. All members opened their cameras and were on time.
- Face-to-face meetings. Especially in the implementation phase, we have arranged face-to-face meetings to communicate and collaborate better and make sure an inclusive environment is achieved.
- Short phone calls when necessary.

The asynchronous environments used are the following;

- Messaging through WhatsApp.
- Communicating through GoogleDocs comments to share ideas while writing project documents.

7.4.3 Taking lead role and sharing leadership on the team

Since we have divided the entire workload into small work packages as mentioned above, each package is led by different members of the group.

Zübeyir Bodur: Taking the lead role in Analysis Report, and Add Other Functionalities into Neural Networks.

Hande Sena Yılmaz: Taking the lead role in the Development of User Interface and Backend in Android.

Alperen Öziş: Taking the lead role in Connecting Android Application with the server for Neural Networks, Add Other Functionalities into Neural Networks, and Mobile Phone Demo.

Yavuz Bakman: Taking the lead role in the Implementation of Neural Networks for segmentation, deep-fill, and other smart features.

Duygu Nur Yaldız: Taking the lead role in High-Level Design Report, Demo in Desktop, Low-Level Design Report, support for neural network models, and Final Report.

Each member of the group integrates with the members involved in developing each work package which is led by mentioned group members and every member is welcome to share the work at any point to develop the best functioning products.

7.4.4 Meeting objectives

WP#	**WP Title**	**Leader**	**Members Involved**
WP1	Analysis Report	Zübeyir Bodur	All members
WP2	High Level Design Report	Duygu Nur Yaldiz	All members
WP3	Implementation of Neural Networks	Yavuz Faruk Bakman	Alperen Ozis, Duygu Nur Yaldiz
WP4	Development of User Interface and Backend in Android	Hande Sena Yilmaz	Zubeyir Bodur, Duygu Nur Yaldiz
WP5	Connecting Android Application with Neural Networks/Server	Alperen Ozis	Yavuz Faruk Bakman
WP6	Demo in Desktop	Duygu Nur Yaldiz	All members
WP7	Low Level Design Report	Zubeyir Bodur	All members
WP8	Add Other Functionalities into Neural Networks	Yavuz Faruk Bakman	Alperen Ozis
WP9	Add Other Functionalities into	Hande Sena	Zubeyir Bodur

	Back-end and Front-end	Yilmaz	
WP10	Mobile Phone Demo	Alperen Ozis	All members
WP11	Final Report	Duygu Nur Yaldiz	All members
WP12	Poster	Hande Sena Yilmaz	All members

Above is given the final work packages of our project. Each work package is done successfully by the assigned group members. Hence, now we have our application DeePaint providing its users easy-to-use and fast image manipulation features.

7.5. **New Knowledge Acquired and Applied**

Our initial knowledge to develop DeePaint was not sufficient enough. For example, even though some of us had previous experience on Android Studio and mobile application development, we needed to learn much more features while developing our project. We took help from Youtube videos, Udemy courses and Github tutorials to enhance our knowledge on the application side.

Also, although some of our team members were experienced on machine learning algorithms, we did not have too much experience in generative and segmentation neural networks. Therefore, we read lots of papers regarding our aimed features and inspected their Github repositories.

1. Instance Segmentation: For this task, we employ the algorithm proposed in Masked-attention Mask Transformer for Universal Image Segmentation. The key component is masked attention which enables the model to extract localized features. The model outperforms other methods by a significant margin.
2. Image Inpainting: For this task, we used Resolution-robust Large Mask Inpainting with Fourier Convolutions paper. The special property of this architecture is the success in high resolution images which are very common in mobile photos.
3. Convert Drawings: For this task, we used the paper named as Learning to generate line drawings that convey geometry and semantics. This recent work enables our app to convert a photo into a sketched version.

4. Style Transfer: The most of the work for this task requires iterative algorithms which consumes too much time. However, our app should work fast and smoothly. Therefore, we adopt the algorithm provided by tensorflow which is quite fast.

Besides the literature review, we did not know how to upload a neural network into a server and communicate with mobile applications. We considered various server providers such as AWS, Microsoft Azure, and Google Colab. In the end, we decided to continue with Google Colab.

8. Conclusion and Future Work

Overall, DeePaint became a working application for Android mobile devices. It is capable of removing a desired region from an image (this region is defined either by drawing by hand or selecting an object that is automatically segmented by the application), making a photo as if it is hand drawn, and last but not final transferring the style of an image to another one. DeePaint provides its users with these functionalities as easily as possible which was one of our main goals initially.

However, of course, DeePaint can be enhanced by adding brand new features, such as image blending, text to image manipulation etc. Thanks to our project design, adding new features to the application will not be challenging. Moreover, we are able to easily change the current neural networks with new ones if state-of-the-art changes in time. Lastly, as a future work, we can consider adapting the application to the iOS platform to reach out more users.

9. Glossary

Neural Network: Algorithms that reflect the behavior of the human brain, allowing programs to recognize patterns [6].

10. References

[1] “Instagram by the Numbers: Stats, Demographics & Fun Facts,” *Omnicore Agency*, Jan 3, 2021 [Online]. Available: https://www.omnicoreagency.com/instagram-statistics/ [Accessed Oct 10, 2021].

[2] “Signs and Symptoms of Cell Phone Addiction,” *PsychGuised*, [Online]. Available: https://www.psychguides.com/behavioral-disorders/cell-phone-addiction/signs-and-symptoms/. [Accessed Nov. 13, 2021].

[3] “Social Media and Mental Health,” *Help Guide*, [Online]. Available: https://www.helpguide.org/articles/mental-health/social-media-and-mental-health.htm. [Accessed Nov. 13, 2021].

[4] “Data Privacy,” *Code of Conduct*, [Online]. Available: https://code-of-conduct.roche.com/en/data-privacy.html. [Accessed Nov. 13, 2021].

[5] “ACM Code of Ethics and Professional Conduct,” *ACM*, [Online]. Available: https://www.acm.org/code-of-ethics . [Accessed Nov. 13, 2021].

[6] “Neural Networks,” *Intel*, [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks . [Accessed Nov. 15, 2021].