Bilkent University

Department of Computer Engineering

CS 491: Senior Design Project

Fakenstein

High Level Design Report

Group Members:
Yusuf Ardahan Doğru
Atakan Dönmez
Öykü Irmak Hatipoğlu
Elif Kurtay
Cansu Moran

Website: Fakenstein

Supervisor: Dr. Selim Aksoy
Innovation Expert: Adnan Erdursun
Jury Members: Dr. Shervin Arashloo and Dr. Hamdi Dibeklioğlu

High Level Design Report December 24 , 2021

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

1.1. Purpose of the System

The concerns about the violation of privacy have become more and more prominent in human lives with the ever growing internet. Every day, people are photographed without their consent and appear in the photographs that are uploaded online. On average, an American is caught on camera 75 times a day without being aware of it [1]. This poses a security threat for people on a daily basis. Examples of people trying to protect other people's privacy on social media is increasing everyday, especially with celebrities covering their children's faces with emojis to protect them from media publicity. Moreover, in social events when a picture is going to be taken, there are always a few people who do not want to appear in the photograph. Therefore, before taking a photograph, permission must be sought from all participants. With more widespread use of the internet from day to day, personal data protection laws are becoming stricter. When taking pictures, people have to be more sensitive about the privacy of the data holders. For example, from Fig. 1, it can be seen that Google Street View blurs the faces of people appearing in photographs in order to protect their privacy [2]. Apart from blurring the images, another method to protect the privacy rights of the individuals that appear in the photograph is removing them using external tools such as Adobe Photoshop which people should buy a subscription to use [3], and Magic Eraser which is actually not available for anyone who does not own a Google Pixel 6 and thus have a limited user base [4].

Figure 1: People with blurred faces in the Google Street View of Bilkent.

Additionally, removing people and filling the remaining space with background is a task that requires advanced knowledge in Adobe Photoshop and would take a significant amount of time and effort. A more user-friendly alternative to remove unwanted people from the photograph is an image manipulation tool named Inpaint designed for the purpose of image restoration [5]. However, this tool requires the user to carefully paint each object to be removed. If the painting is not precise enough or if the photograph has a complicated background, then the Inpaint tool outputs unrealistic photographs. The Inpaint tool does not solely exist to remove humans but it is a general purpose removing algorithm. A complicated background is considered to be a background that does not have consistent lines or has different patterns in close areas.

Our proposed phone application, Fakenstein, aims to realistically replace the faces of unknown people with artificially generated faces in order to protect their anonymity. Unlike Inpaint or Adobe Photoshop, Fakenstein intends to keep user interaction as little as possible while outputting a realistic looking photograph. Today, the style-based GAN algorithms can produce extremely realistic artificially generated faces that do not exist. The examples of such faces can be examined from the website where each time the page is refreshed, the website displays a generated face [6]. For the task of replacing faces in a picture, the guidance of the infamous “deep fakes” can be used. Mostly originated from the GitHub repository “DeepFaceLab”, deep fakes are used to replace faces in videos to generate realistic fake videos. For example, it is possible to replace Iron Man’s face with Tom Cruise’s and produce a video that looks like Tom Cruise is the actual actor playing Iron Man. Using a similar method we aim to obtain seamlessly replaced faces. Additionally, we are trying to build an application that will be available for everyone unlike Magic Eraser which is only available for Google Pixel 6 owners and Adobe Photoshop which can only be used through an expensive subscription.

This report will start with the description of the current systems. Then, we will illustrate our proposed system, Fakenstein. Our system will be described thoroughly by specifying the functional, non-functional, and pseudo requirements. We will further explain our system by providing system models such as scenarios, use case models, object and class models, dynamic models, a navigational path, and screen mockups. We will further analyse our proposed system in terms of engineering design, alternatives and risks, project plan, teamwork, ethical and professional issues, and the learning strategies we use.

1.2. Design Goals

Usability

- The application should have a simple and self-explanatory interface.
- The language used in the application should be clear and easy to understand.
- The application should enable the user to upload and get their image.

User Friendliness

- The application should not be hard for the user to understand and use.
- The application should include a tutorial section to help users adjust to the application.

- The application should be optimized in order not to bore users with long loading and processing time.
- The user should not be required to know about Deep Learning algorithms that will be used in the application.

Maintainability

- Any open source or third party software used will be required to have long term support for future maintenance.
- The application should have periodic version updates to use the latest versions of the external modules and third party applications.

Extensibility

- The application should be extensible so that in the future it can be used as a plug-in or be integrated by other applications.
- The application should be open to any upgrades considering there could be new functional/non-functional requirements or user interface changes.

Reliability

- The application will be tested for various cases in terms of features.
- The application should not crash for reasons other than operating system based version differences. The reason for the crash must be logged and the user should be informed about it.
- The application should work for differing sizes, brightness, image types, and image formats such as BMP, JPEG, PNG, WebP, and HEIF.

Accessibility

- The Android version of the application should be made available on the Google Play Store to reach a vast amount of users.
- The desktop version of the application should be available for download on the website of the application.

Portability

- The desktop application will be able to run independently from the operating system via Java Virtual Machine.
- The Android application will be able to run independently from the brand or model of the mobile phone. The only constraint should be the version of Android installed.

Efficiency

- The application should be able to run smoothly in older and newer generations of Android smartphones with Android Marshmallow 6.0 or higher installed.
- The picture upload should not take more than approximately 10 seconds (subject to change).
- The application should provide the user with the output image under approximately 20 seconds (subject to change).

Scalability

- The application is made for daily use, so available platforms and users may change as there are new developments.
- The service we are providing will not be affected by user/customer number as the application will work locally once downloaded.

1.3. Definitions, Acronyms, and Abbreviations

BMP	Bitmap Image File
JPEG	Joint Photographic Expert Group Image
PNG	Portable Network Graphics
WebP	Web Picture Format
HEIF	High Efficiency Image File Format
JVM	Java Virtual Machine
IDE	Integrated Development Environment
ML	Machine Learning
DL	Deep Learning
SDK	Software Development Kit
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
API	Application Programming Interface

1.4. Overview

Fakenstein aims to produce realistic photos by replacing people's faces in the background with artificially generated faces in order to preserve people's privacy. Fakenstein will be developed as a cross-platform application and will be available as an Android and desktop application. Our application can be used by uploading a photograph from the gallery. Our algorithm will perform a face detection using Deep Learning models and find all the identifiable faces in the image and tag faces on the background of the photograph. Our main goal is to make the application as automated as possible. However, if the face detection algorithm can't detect some faces in the picture, the user will be able to manually blur the faces. After the faces are selected, artificially generated faces will replace real human faces using the deepfake method to perform realistic relocation. In order to preserve the nature of the original photograph, the genders, skin colors, and the ages of people will be taken into consideration. This will be possible by performing

classification on the detected faces in terms of their sex (man, woman), skin color (fair or dark skin), and age group (children, adult, elderly). After that, the corresponding generated faces will be used according to the classification results. Again, if the face generated does not seem to be suitable, the user will have the capability to change the artificial face with another one by specifying its qualities. Additionally, in order to increase the realisticity, the orientation of the face will be determined by using a head pose estimation algorithm and utilizing an appropriately generated face.

Our main target audience is anyone who uses social media and publishes photographs online. From event photography to news and blog posts, instead of trying to blur, cover, or remove people from photographs, the publishers can easily use Fakenstein to obtain realistic looking photographs that do not violate the privacy of individuals.

2. Current Software Architecture

In order to protect the privacy of people appearing in pictures without consent, there are certain options available. One of the most common tools for photo manipulation is Adobe Photoshop [3]. With Adobe Photoshop, users can remove people from pictures or manipulate their faces to make them indistinguishable to protect their privacies. While Adobe Photoshop offers a wide selection of tools to achieve such goals, it is an expensive software which requires professional skills, effort and time to perform such tasks. Another alternative is the Inpaint software, which is aimed to make removal of objects from pictures easier [5]. While Inpaint is a free and a more easy-to-use alternative to Adobe Photoshop, the image quality in Inpaint doesn't reach the quality of a professionally edited photo in Adobe Photoshop. Since Inpaint is an automated software, it doesn't perform well in backgrounds that are complex. A newly introduced alternative is the Magic Eraser tool that is integrated in the latest Google Pixel phone, Google Pixel 6, released on 25th of October 2021 [4], [7]. Magic Eraser tool in Google Pixel 6 allows users to remove people from pictures automatically without the need of any professional skills. While this tool does not have much review available, since it is fairly new, the first impressions are that the Magic Eraser is quite successful in removing people from pictures while not disrupting the background [4]. However, no matter the quality of the tool, it is only available to a very limited audience, since only the owners of Google Pixel 6 can achieve it. Moreover, in instances where removing people from the picture is not the goal, both Magic Eraser and Inpaint are not applicable tools. For example, imagine an event holder that wants to share a picture taken of the attendees during the conference, but can't share it because there are attendees that don't give consent to their pictures being shared. In that case, removing the attendees from the picture would not be sensible, since the aim of the picture is to demonstrate the participation in the conference.

Considering these tools and their limitations, our proposed application offers a novel approach for protecting the privacy of people in the pictures. Our system doesn't require professional skills since it is automated, aims to create higher quality images than other automated approaches by replacing the faces of people with fake generated faces instead of trying to remove them from the picture, offers a lite version on mobile phones to reach a wider range of people and an extended version for desktop for more professional uses such as newspapers, blogs, etc. and offers a solution for protecting people's privacy even in cases where removal is not the goal.

Since our goal is to replace the faces of people in pictures with fake faces that don't exist, it is also required to mention current fake face generation tools. Replacing people's faces with other faces is already being done with the emerging Deepfake technologies. While visual artists are performing such replacements as a profession, websites such as ThisPersonDoesNotExist.com [6] or Generated.Photos [8] display computer generated fake faces for entertainment purposes. While there is advanced research for generating fake faces, there is no system available that can perform fake face replacement in a given picture in an automated way. In that sense, our proposed system aims to create models inspired by research in fake face generation and advance this technology while combining it with automated picture editing.

3. Proposed Software Architecture

3.1. Overview

This section presents details about the proposed software architecture. First, the subsystem decomposition will be introduced which will include a detailed description of which subsystems were chosen for our software and how each subsystem interacts with each other. Then, hardware/software mapping of the project will be given, followed by details about how persistent data management will be achieved. Access Control and Security and Global Software Control sections will lay out certain control mechanisms crucial for the implementation of the application. Finally, boundary conditions of the software will be given which will include initialization, termination and failure conditions of the program.

3.2. Subsystem Decomposition

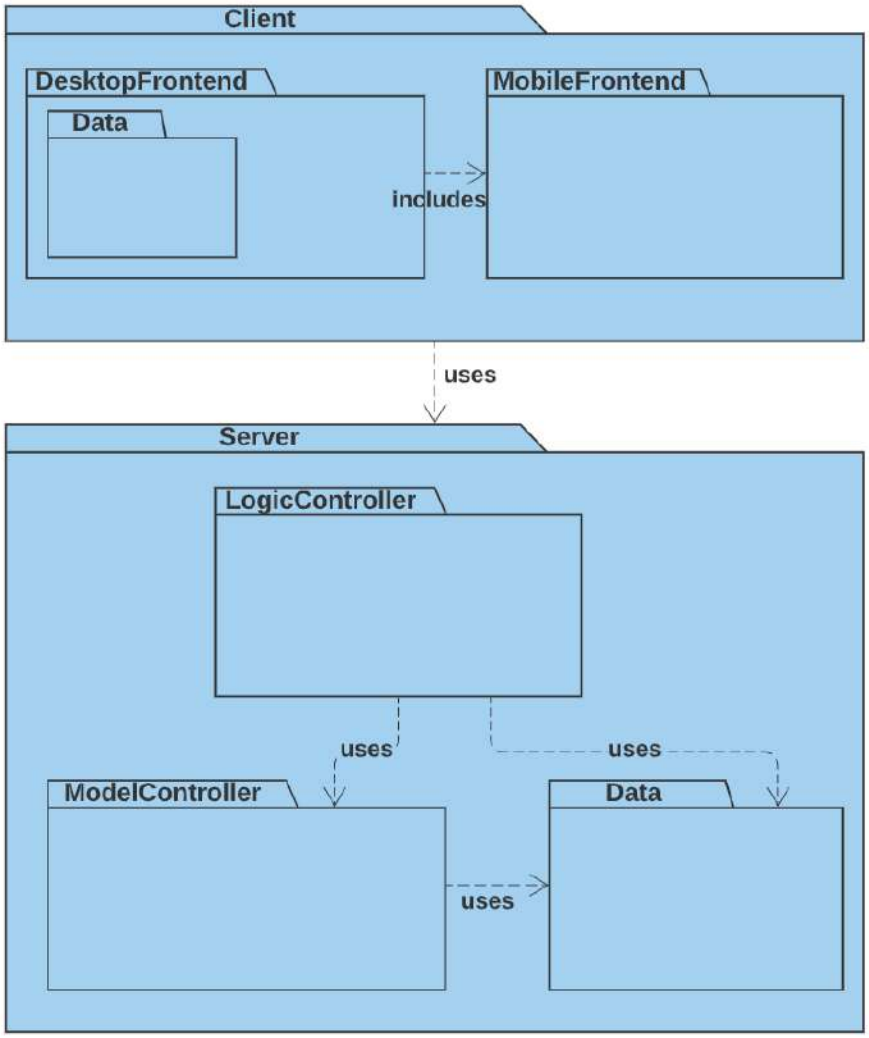

Figure 2: Overview of Subsystem Decomposition

Client-server architecture will be used in the subsystem decomposition of Fakenstein as shown in Figure 2. The reason behind choosing this architecture model is simple. Since image processing needs to be done on a server, to achieve a lightweight application for the client, and client only needs to give input and request for an output, client-server architecture fits Fakenstein naturally. Cohesion is satisfied with keeping view and input services in client subsystem and image modification models, database, and other functionalities of the application in the Server subsystem.

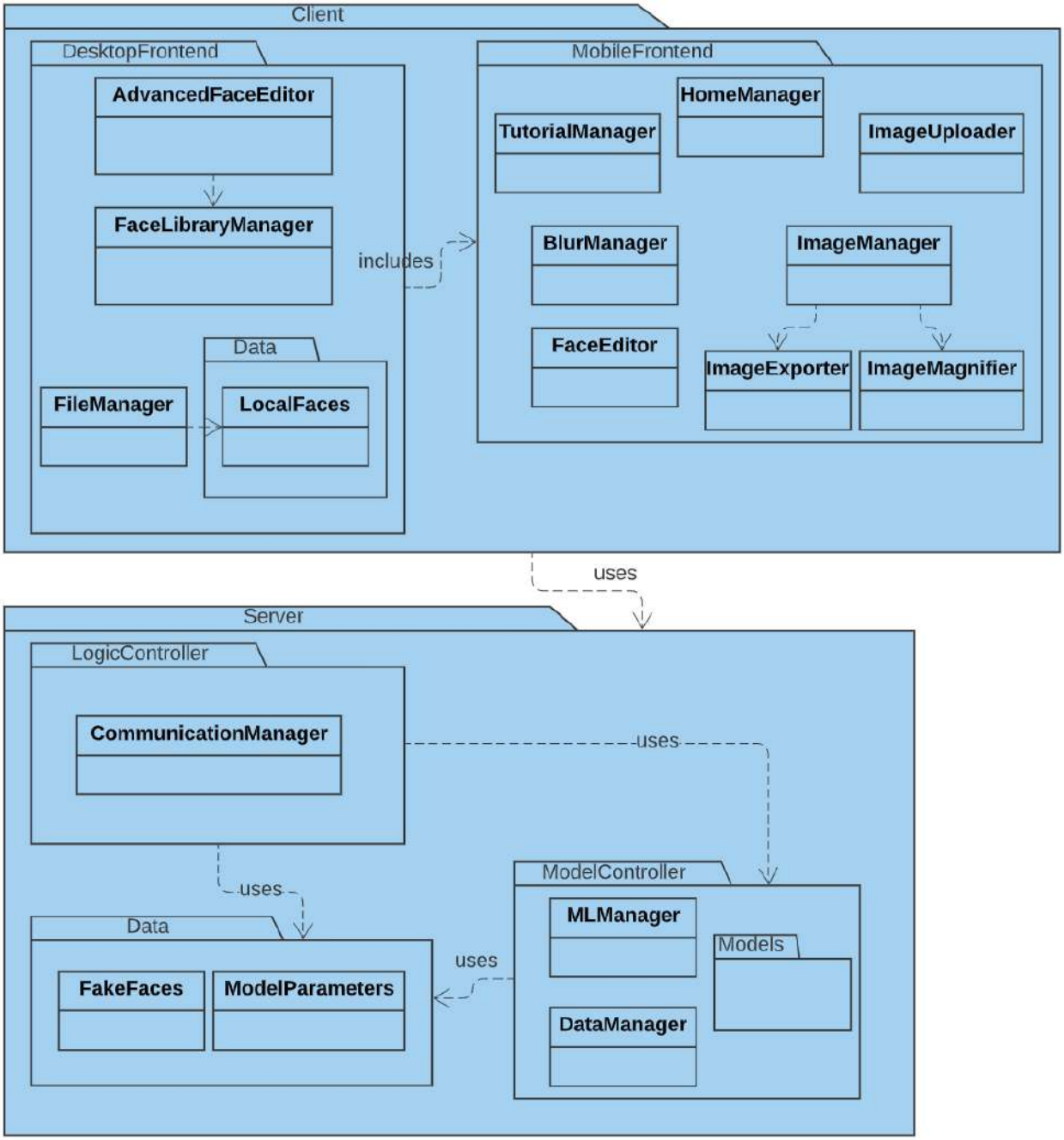

Figure 3: Detailed Subsystem Decomposition with Classes

Fakenstein will have the view layer which is divided into two subsystems: Desktop Frontend, which includes another Data subsystem, and Mobile Frontend. This layer will run on the client device which is a mobile phone and a desktop application in our case. Desktop Frontend extends and modifies the classes in Mobile Frontend according to its domain. In order to prevent repetition this relation is shown with the “includes” relationship between the subsystems. A frontend implementation with React Native is planned to be used for mobile subsystems, and both subsystems will be implemented in Android Studio IDE. In React Native, the controller and the view scripts are in the same class. Therefore, there is no need for a separate controller layer only for the view. The mobile clients will be able to navigate in the application, upload an image and manipulate the image by blurring or changing faces in the application with the classes given inside the Mobile Frontend subsystem. On top of the available Mobile Frontend features, the Desktop Frontend system has two more classes for its additional feature: Face Library. In addition, the Desktop Frontend subsystem has a File Manager and a small local Data subsystem to store and manage its local face library. The Desktop application will do the file management tasks on its own as the communication with the server will cause overhead in time and additional costs. Furthermore, in order not to violate any user privacy rights, saved faces will not be kept in the server layer, in an online database. This way the faces used by users will not be known.

Client Tier is using Server Tier to apply the ML models to face detection, background faces classification, face information (age, gender, skin color, face orientation) classification, fake face generation and face blending functionalities of Fakenstein. This communication will be done with the requests of the Client which are handled with the Communication Manager in the Logic Controller subsystem. Communication Manager transforms the request coming from the Client and sends the required values and data to the MLManager in the Model Controller subsystem. Furthermore, the Communication Manager might need to create a bridge between the Data subsystem and the Client or the MLManager to increase efficiency. Server subsystem will generate the new image based on the requests and send the answer back to the Client.

In the Data subsystem inside the Server Tier, Fake Faces and Model Parameters are kept in an online database, Firebase. Model Parameters include hyperparameters and weights for deep learning models. Fake faces are pre-generated face images which can be used instead of generating a new face in case an existing face matches the requested classifications. The Fake faces database increases efficiency and robustness of the system.

The ModelController has a subsystem for different ML models which will be used and an MLManager which initializes them with parameters from the Data subsystem taken by the Data Manager class. After initialization, MLManager also takes predictions from the required models and modifies the image according to parameters and requests coming from the Communication Manager.

3.3. Hardware/Software Mapping

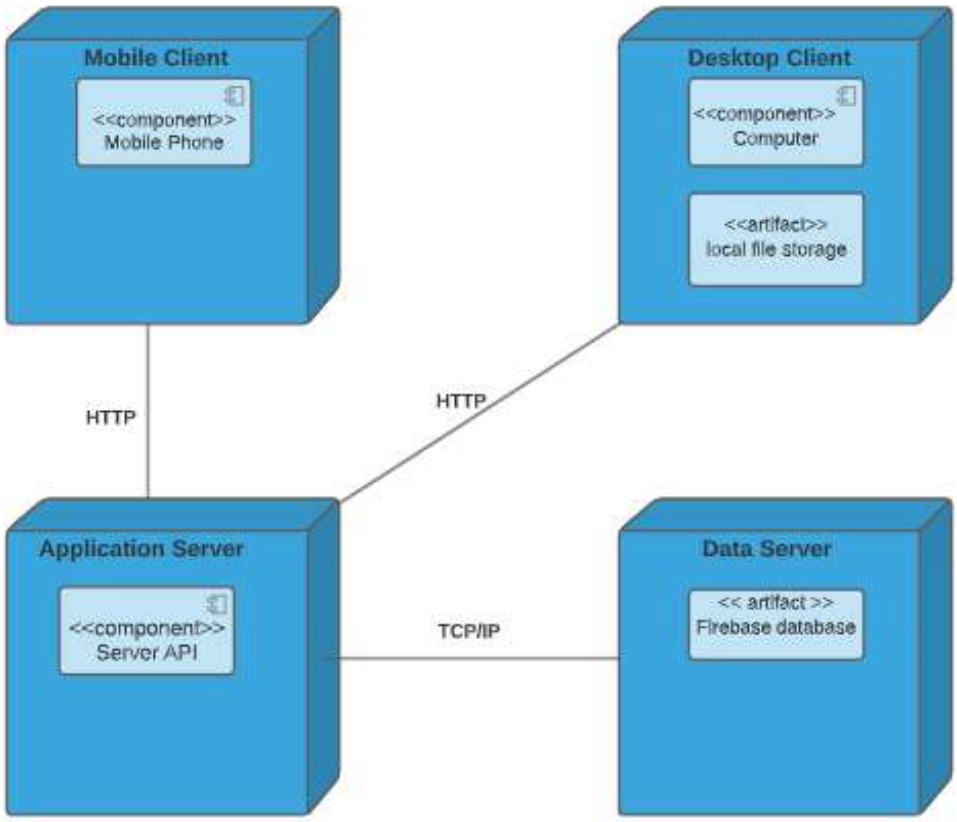

Figure 4: Deployment Diagram

Fakenstein is an application designed to run on Android devices and Windows desktop. Hence, the client-side software is compatible with Android versions. For desktop, JVM is used to abstract the hardware and make the software compatible with all desktops with the required JVM version installed. In terms of hardware, the client-side consists of the mobile phone and computer hardwares. Fakenstein will make use of the certain components of these hardwares such as memory and internet connection modules. Both desktop and Android client-side makes use of local file storage to access initial images and save the generated images. However, the desktop client also uses extra storage for the face library.

The backend of Fakenstein consists of the server code and the API integration. The implementation of the Model Controller subsystem (details in section 3.2) will be done in Python, which will be used to implement the DL algorithms through the Pytorch framework. The Android and desktop application development will be done with Java for backend and JavaScript for the React frontend. Android application development will also make use of Android SDK. Android Studio IDE will be used to create these applications in both domains. HTTP requests/responses will be used to make the connection between the client and the application server. The

backend/server-side of Fakenstein will run independently of the client's operating system via JVM.

The data server of the application will essentially be the database which will be used to keep persistent data (details in section 3.4). Firebase will be used for the database and the database will have a connection with the application server. The connection serves as a means to provide the application server with the persistent data mentioned such as the fake faces and model parameters. The deployment diagram in Figure 4 depicts the relations between the components, artifacts and nodes which make up Fakenstein.

3.4. Persistent Data Management

Persistent data is vital for effective and usage of the application. In Fakenstein, the data that needs to be managed persistently is composed of the artificially generated faces that will be used to replace the faces in the input photos and the weights and hyperparameter values of the DL models. Additionally, for the desktop version, the user will be able to save certain fake faces to their face library, which will be another type of data that needs to be managed persistently.

Both the Android and the desktop version of the application will use the online Firebase database and its own servers to store certain pre-generated fake faces, to shorten the runtime of DL models. Additionally, the desktop version will make use of the local file storage system for user specific fake face libraries.

3.5. Access Control and Security

One of the main goals of this application is to help preserve privacy of life and personal information safety thus, security is of utmost importance. The application requires the user to provide images to process, therefore the users will be prompted to give access to their media before doing so. Furthermore, only the images that the user manually selects will be fed into the application to limit unwanted access to the user's gallery. Accessing the image and saving the output will be done locally. Although processing the image will take place in the application server, after the image processing is complete, all user data(including the input and output images) will be deleted.

Access to the internet will be required in two situations: to process the images in the application server and to use one of the fake faces previously generated by the ML model and saved in the online database with the aim of reducing runtime to edit an image. It should be noted that the fake faces that are saved in the online database are independent from the user pictures or selections. These faces are generated by the developers to reduce the need of face generation process in cases where a fitting face is already saved in the database. Hence, only the admin can directly access and modify the images saved in the online database. It will be ensured that users' interaction with the faces generated by the ML model is only through the request of the app.

The application also makes detections and inferences on the input image on the characteristics of the “original” faces (the faces of the people in the background of the image that is to be removed from the end product). Such inferences will be deleted from the application server and cache with the termination of the app in order to ensure information safety.

The source code of the application will be available to anyone to try and test for malevolent activities if the above guarantees are not satisfactory or convincing. However, the online database where certain fake faces are stored for the sake of improving the runtime of the application will not be publicly available.

The users will need to accept the prompted terms and conditions before having access to the app. Both the terms and conditions and in-app activities will be designed in accordance with both “The General Data Protection Regulation” [9] and the “Kişisel Verilerin Korunması Kanunu” [10].

3.6. Global Software Control

The application follows a rather linear sequence of tasks in an order because model layers work by utilizing outputs of other models as input. Therefore, each task will wait until the previous one is done. The sequence of tasks start with the image being uploaded. After the user has uploaded an image, the frontend will be halted until the server-side finishes face detection and background faces classification steps. The control will be passed back to the client-side as the user makes any changes or selections they want. After that the frontend will be halted again while the server-side finishes face classification, generation and combination steps. The user will then be provided with the output and will be allowed to make any further changes or finish the process.

The last server-side step takes the longest so preemptive tactics such as creating a library of generated faces on the server side will help make the app more responsive. A possible problem that can occur is giving back the client side of the application control before the server side processing is done. Therefore when we want to halt the frontend the user will be met with a loading spinner and any input on the app will be disabled during loading. Unfortunately, the processes that await each other cannot be optimized to utilize concurrency by performing each of the operations to single inputs since inputs of the models depend on the output of a previous model. Therefore, the client side and server side will have to await each other’s signal before starting to process so that we ensure all the computations are done before moving on the other steps.

3.7. Boundary Conditions

The app will have three boundary conditions: Initialization, termination and failure of the application. More details about each boundary condition can be found below:

Initialization: The user must have access to the application, which entails the user downloading the app from either Google Play Store on Android devices or from the application website on desktop. The application should be connected to the internet to be able to connect to the server and process the images.

Termination: A user can terminate the application by closing the application from their device (Android or desktop). When the app is terminated, any running process will be terminated without saving and any photo that has been generated but not saved on the local device will be discarded. Any data generated during the processing of the image will be deleted from the application's cache as well as the server with the termination of the app.

Failure: A program failure can happen in certain cases. These cases are listed below:

- Program failure can occur if a gallery photo access is denied. In that case, an error message prompting the user to give gallery access to the application will be displayed. The process will continue once the access is granted.
- Failure can occur due to server connection loss caused by server failure or internet connection loss. If the connection with the server is lost, the user will be notified about the problem and the ongoing process will be terminated. If the connection is restored, the user can restart the process, as the terminated process and any data generated during the process will not be saved.
- Limited device storage can also cause a failure while trying to save the finalized image to the local disk. In such a case, if the user opens up enough free space on their local disk without terminating the application, when they return back to the running application, they can try saving the image again. If the app is terminated during this process, the finalized image will not be saved.
- Any other unexpected failures (loading the image, problem booting the program, etc.) will be reported to the user with an easy to understand error message.

4. Subsystem Services

4.1. Client Subsystem

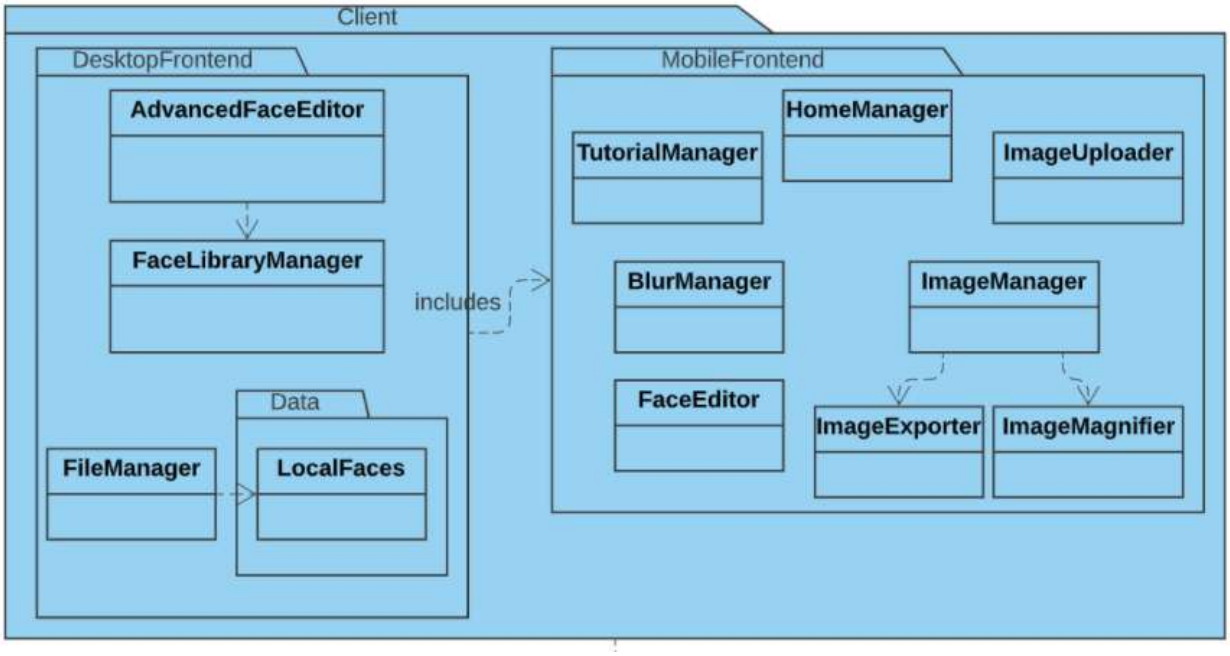

Figure 5: Client System Diagram

4.1.1. Mobile Frontend

This package provides the user with a GUI to use the application's features.

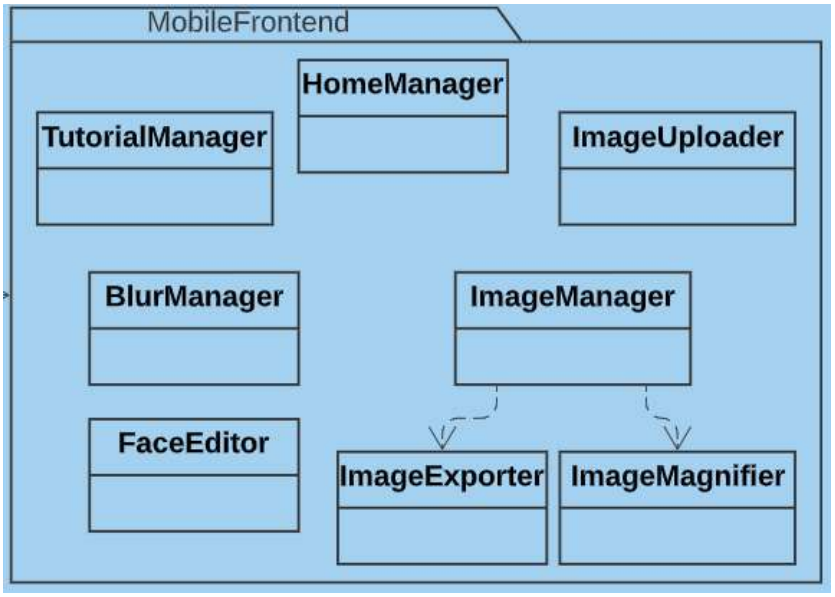

Figure 6: Mobile Frontend Diagram

TutorialManager

This view is responsible for displaying tutorials.

HomeManager

This view is responsible for displaying the initial home page where there is the possibility of advancing to the tutorial or uploading an image.

BlurManager

This view is responsible for blurring the areas selected by the user on an image.

FaceEditor

This view is responsible for displaying the foreground and background images, and allowing the user to manually select the faces which they wish to be changed.

ImageUploader

This view is responsible for displaying the user their gallery to choose an image to upload.

ImageManager

This view allows the user to make changes on the image by extending respective classes for Image Exporter and Image Magnifier. In addition, the modifications on the image will be selected, requested, and received in this view.

ImageExporter

This class allows the user to export the resulting image by saving the image to their gallery.

ImageMagnifier

This class allows the user to magnify the image for easing selection of individual faces or blurring areas.

4.1.2. Desktop Frontend

This package extends the Mobile Frontend and provides the user with additional capabilities such as the face library.

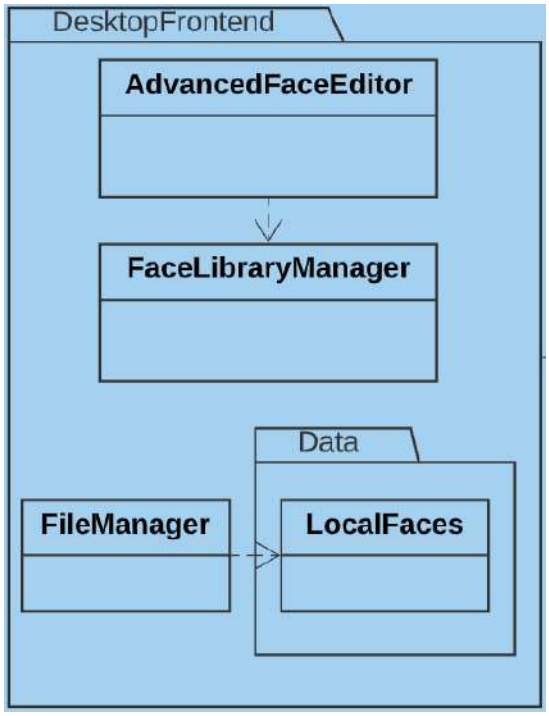

Figure 7: Desktop Front End Diagram

AdvancedFaceEditor

This view allows the desktop users more advanced editing options on top of the mobile ones such as displaying different generated face options for replacement.

FaceLibraryManager

This class handles the face library, which allows the users to save/delete faces to/from the face library, display the faces in the library and choose a face from the library for replacement.

FileManager

This class handles the communication between the FaceLibraryManager and the local database, LocalFaces, to add, get, modify, or delete an entry in LocalFaces.

LocalFaces

A local database which contains the generated faces saved by the user with their names.

4.2. Server Subsystem

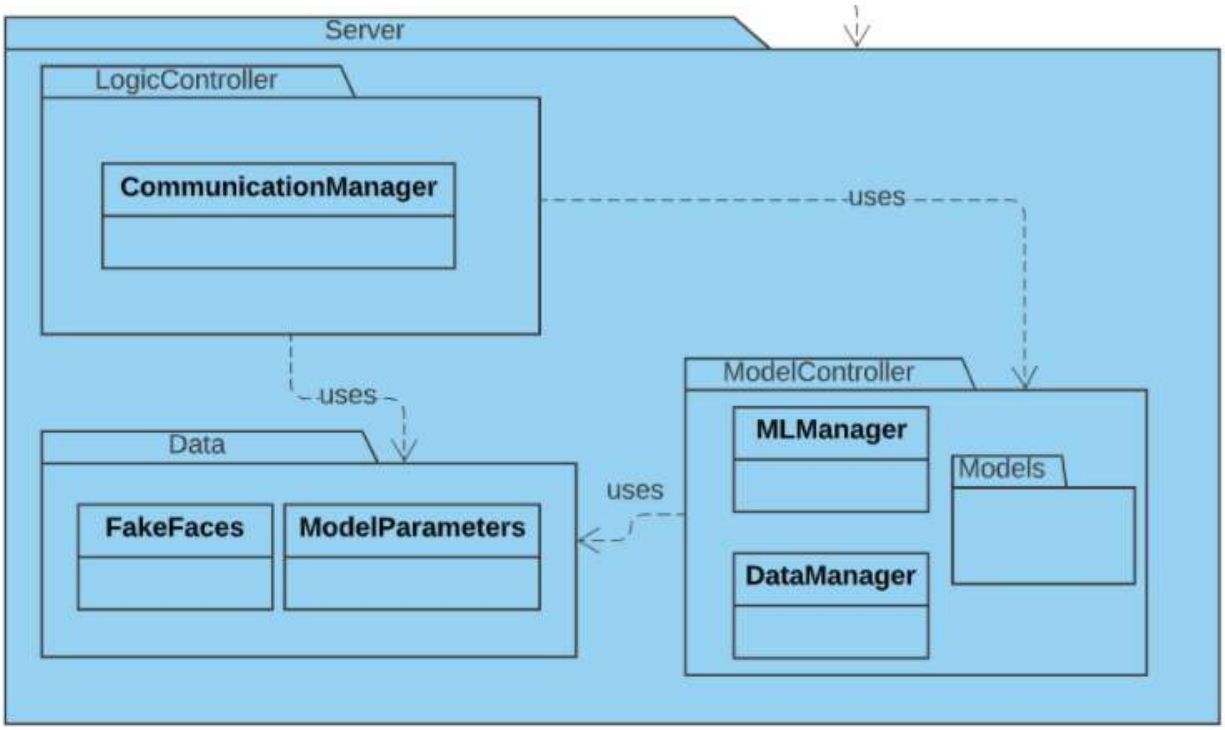

Figure 8: Server System Diagram

4.2.1. Logic Controller

This subsystem controls the requests and responses between the server and the client.

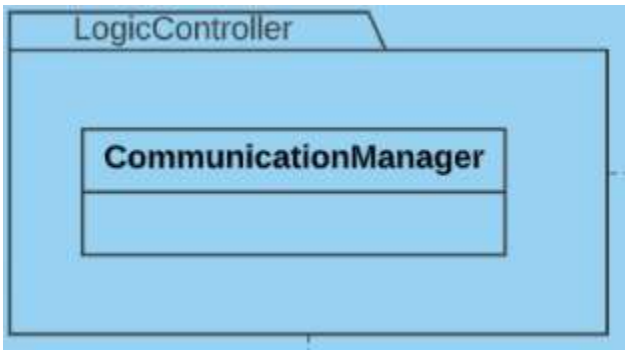

Figure 9: Logic Controller Diagram

CommunicationManager

This class handles the requests and responses between the Client and Server subsystems. This class also formats the resulting image or other outputs coming from the components to adapt to the requirements of each side.

4.2.2. Data

This package holds the required data (model weights) to process and generate images and is responsible for passing them onto the ML models. It also contains the pre-generated Fake Faces (from the online database) which are used to eliminate the fake face generation time in the applications.

Figure 10: Data Package Diagram

FakeFaces

The artificially generated faces to be used as a replacement for the people that appear in the background of an image. These faces are stored in the online database to reduce the runtime of the application.

ModelParameters

Parameters used in the ML models. These parameters can be weights of pretrained models in addition to hyperparameter values for any model that needs to be re-trained.

4.2.3. Model Controller

This package holds the different ML models to be used in the application and is responsible for their operation.

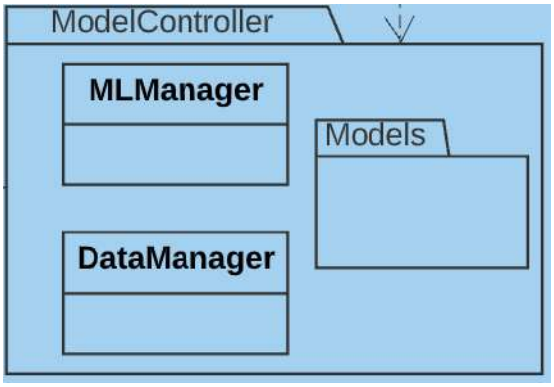

Figure 11: Model Controller Diagram

MLManager

This class initializes the ML models and manages the input and output of the models.

DataManager

This class handles the communication between the Data subsystem and the MLManager so that the ML model can use the pre-decided parameters and pre-generated fake faces can be used instead of new fake face generation if required information for a face has a match in the database.

5. Consideration of Various Factors in Engineering Design

5.1. Public Health Factors

Fakenstein's contains no features that might affect or correlate to public health in any shape or form.

5.2. Public Safety Factors

Fakenstein will need access to the user's media and also permissions to modify such media. Collection and safety of the data are major issues therefore Fakenstein will not store any of the user's media to an external server and database. Thus, personal information safety will not be infringed and be protected by performing the operations on the user's local device.

5.3. Global Factors

Fakenstein's contains no features that might have a global impact.

5.4. Cultural Factors

There are many cultures that hold every bit of their privacy as sacred due to malpractices in their history. Fakenstein can help restore that privacy as appearing in the photographs that others post online is a possible way to determine where someone was at a particular time. There are also cultures where drawings or photography of individuals are frowned upon or even straight up forbidden. Taking a picture that includes other individuals in the background can trespass on such cultural values without the individuals' knowledge [11, 12].

5.5. Social Factors

Protection of privacy is getting harder by the day in our society as social media grows bigger and bigger. Providing a way for people to remove the background people in their photographs allows people to reclaim their daily privacy. Also, there is a recent trend of people (especially celebrities) removing or censoring the face of their children on social media is growing more and more popular. This comes from the fact that introducing parts of people's personal lives takes their ability to participate in the society as a regular individual and invites strangers into intimate moments of their lives. Fakenstein is an application that can help reinstate such social barriers that permit control over individuality.

5.6. Environmental Factors

Fakenstein's contains no features that might affect or correlate to environmental factors other than the amount of electricity used by your phone or computer while the application is running.

5.7. Economic Factors

Posting photos of people without their permission infringes on their rights of privacy (namely Turkish Personal Data Protection Law No. 6698 [1] and European “Data Protection and online privacy” Law [2]) and therefore allows for possible lawsuits. Through the use of Fakenstein, such personal privacy infringements can be thwarted to avoid penalty fines. Furthermore, the application eliminates the need to use costly photoshopping programs, therefore saving money to application’s users on that front. The basic application for the mobile phone will be free so that everyone can use it. The professional system for the desktop will require a paid subscription for 1 dollar a month which is insignificant next to Adobe Photoshop which is 21 dollars a month [13].

Table 1: The table of design factors, their effect levels and their effects in summary.

Factor	Effect Level	Effect
Public Health Factors	0	Fakenstein’s contains no features that might affect or correlate to public health in any shape or form.
Public Safety Factors	10	Fakenstein will not store any user data that has been accessed or modified
Global Factors	0	Fakenstein’s contains no features that might have a global impact
Cultural Factors	4	Fakenstein allows removal of figures of individuals which can be utilized to fit certain cultural niches.
Social Factors	8	Fakenstein bolsters protection of privacy in the society
Environmental Factors	1	Fakenstein’s contains no features that might affect or correlate to environmental factors other than the amount of electricity used by your phone or computer while the application is running.
Economic Factors	5	The mobile application will be free and the desktop application will require a subscription that costs 1 dollars a month.

6. Teamwork Details

6.1. Contributing and Functioning Effectively on the Team

Table 3: The table of work packages, the title leaders and members involved in these work packages.

WP	Work Package	Title Leader	Members Involved
WP1	Web Page	Ardahan Doğru	Atakan Dönmez, Öykü Hatipoğlu
WP2	Reports	Elif Kurtay	All members
WP3	Face Identification	Öykü Hatipoğlu	Cansu Moran, Ardahan Doğru
WP4	Blurring	Atakan Dönmez	Cansu Moran, Elif Kurtay
WP5	Face Modification	Cansu Moran	Elif Kurtay, Atakan Dönmez
WP6	Desktop (Windows) GUI	Elif Kurtay	Öykü Hatipoğlu, Ardahan Doğru
WP7	Mobile (Android) GUI	Öykü Hatipoğlu	Elif Kurtay, Ardahan Doğru
WP8	Database Integration	Atakan Dönmez	Öykü Hatipoğlu, Cansu Moran
WP9	Testing	Cansu Moran	All Members

Gantt Chart for planned Work Package start and end dates:

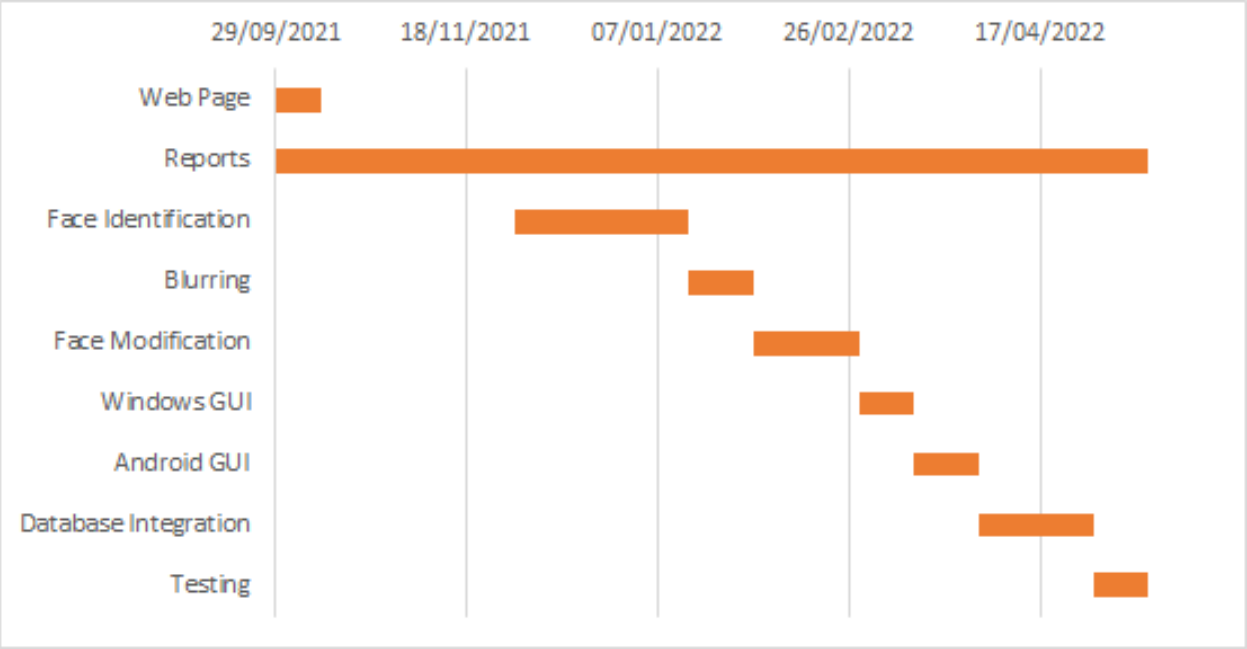

Figure 12: The Gantt Chart.

Work Package 1: Web Page	
Start Date: 29/09/2021	**End Date:** 11/10/2021
Leader: Ardahan Doğru	**Members Involved:**
Objectives: Create a web page for the project that introduces the team and the project. The web page should also serve as a way to access all reports for the project.	
Tasks **Task 1.1:** Write a short description about the project and create an about page where it is displayed. **Task 1.2**: Create a members page with introductory information for the members of the group **Task 1.3:** Create a page where the reports will be uploaded by the group and always accessible to those that have the link for the web page	
Deliverables **Deliverable 1.1:** About Page **Deliverable 1.2:** Members Page: **Deliverable 1.3:** Reports Page:	

Work Package 2: Reports	
Start Date: 29/09/2021	**End Date:** 15/05/2022
Leader: Elif Kurtay	**Members Involved:** All Members
Objectives: Documenting the steps needed for the project and preparing reports accordingly. The reports also help clear up the details of the project and set up a course of actions to follow.	
Tasks **Task 2.1:** Decide the overall specifications of the project and write a report about it. **Task 2.2**: Analyse the systems that are to be implemented in the project and elements of the strategies to implement them then write a report about it. **Task 2.3:** Design the low-level structure of the project and write a report about it. **Task 2.4:** Design the high-level structure of the project and write a report about it. **Task 2.5:** Finalise the details of the project and write a report about it	
Deliverables **Deliverable 2.1:** Project Specifications Report **Deliverable 2.2:** Analysis Report **Deliverable 2.3:** High-Level Design Report **Deliverable 2.4:** Low-Level Design Report **Deliverable 2.5:** Final Report	

Work Package 3: Face Identification	
Start Date: 01/12/2021	**End Date:** 15/01/2022
Leader: Öykü Hatipoğlu	**Members Involved:** Cansu Moran, Ardahan Doğru
Objectives: Implement an algorithm that identifies faces, classes them as foreground and background and marks the area of the face with a bounding box.	
Tasks **Task 3.1:** Find suitable data to train a model that identifies faces. **Task 3.2:** Implement a model that identifies faces. **Task 3.3**: Implement a model that classes the identified faces based on whether they are on the background or foreground. **Task 3.4:** Mark the area of each face with a bounding box that shows which class it is a part of by the color of the bounding box.	
Deliverables **Deliverable 3.1:** Face identification algorithm **Deliverable 3.2:** Algorithm to show which faces are on the background	

Work Package 4: Blurring	
Start Date: 15/01/2022	**End Date:** 01/02/2022
Leader: Atakan Dönmez	**Members Involved:** Cansu Moran, Elif Kurtay
Objectives: Implementing the feature to allow the users to select parts of the image to blur	
Tasks **Task 4.1:** Implement selecting a circular boundary in the image **Task 4.2**: Implement blurring for a circular area	
Deliverables **Deliverable 4.1:** Blurring functionality	

Work Package 5: Face Modification	
Start Date: 01/02/2022	**End Date:** 01/03/2022
Leader: Cansu Moran	**Members Involved:** Elif Kurtay, Atakan Dönmez
Objectives: Implementing the face modification feature where the user replaces the identified face with that of an artificially generated one.	
Tasks **Task 5.1:** Implement a model that classifies the faces according to their properties such as position, age, gender pose, etc. **Task 5.2**: Implement a model that generates a face according to the given properties such as position, age, gender pose, etc. **Task 5.3:** Implement placing the generated faces on the place of the selected faces. **Task 5.4:** Implement blending for the new faces to not look out of order. **Task 5.5:** Combine the algorithms from the previous tasks so that the outputs feed into one another's inputs.	
Deliverables **Deliverable 5.1:** Face Classification **Deliverable 5.2:** Face Generation **Deliverable 5.3:** Face Combination	

Work Package 6: Windows GUI	
Start Date: 01/03/2022	**End Date:** 15/03/2022
Leader: Elif Kurtay	**Members Involved:** Öykü Hatipoğlu, Ardahan Doğru
Objectives: Design and implement a desktop app for the functionalities	
Tasks **Task 6.1:** Design the UI for a desktop app **Task 6.2**: Implement the GUI for the desktop app **Task 6.3:** Combine the frontend with the backend **Task 6.4:** Implement temporary placeholders for the features that are not implemented yet.	
Deliverables **Deliverable 6.1:** Desktop app	

Work Package 7: Android GUI	
Start Date: 15/03/2022	**End Date:** 01/04/2022
Leader: Öykü Hatipoğlu	**Members Involved:** Elif Kurtay, Ardahan Doğru
Objectives: Design and implement a mobile Android app for the functionalities	
Tasks **Task 6.1:** Design the UI for a mobile Android app **Task 6.2**: Implement the GUI for the mobile Android app **Task 6.3:** Combine the frontend with the backend **Task 6.4:** Implement temporary placeholders for the features that are not implemented yet.	
Deliverables **Deliverable 7.1:** Mobile Android app	

Work Package 8: Database Integration	
Start Date: 01/04/20212	**End Date:** 01/05/2022
Leader: Atakan Dönmez	**Members Involved:** Öykü Hatipoğlu, Cansu Moran
Objectives: Create an online database that stores prior generated faces to reduce the waiting times for the users.	
Tasks **Task 8.1:** Generate a large set of faces. **Task 8.2**: Create an online database. **Task 8.3:** Combine the database with the desktop and mobile applications. **Task 8.4:** Use generated faces in the classification process for optimization.	
Deliverables **Deliverable 8.1:** Database with prior generated faces **Deliverable 8.2:** Optimized classification process	

Work Package 9: Testing	
Start Date: 01/05/2022	**End Date:** 15/05/2022
Leader: Cansu Moran	**Members Involved:** All Members
Objectives: Implement tests that inspect model outputs, connections between different models, UI functionalities and application response to invalid/unexpected input.	
Tasks **Task 9.1:** Write tests to inspect model output accuracy **Task 9.2**: Write tests to inspect how models utilize other models' output as input. **Task 9.3:** Write test for UI functions **Task 9.4:** Write time/efficiency tests **Task 9.5:** Write tests to inspect invalid/unexpected input	
Deliverables **Deliverable 9.1:** Testing results **Deliverable 9.2:** Tools for testing future features.	

6.2. Helping Creating a Collaborative and Inclusive Environment

In order to ensure proper teamwork our team uses synchronous and asynchronous communication tools. Weekly Zoom calls and face to face meetings are held where job distribution and progress tracking is done. Decisions made during the meetings are noted. Then through WhatsApp and emails all members are reminded of said decisions (or informed if they were unable to attend the meeting) and also sent a calendar invitation and a link to the next meeting. After the meetings the members are split up and paired to accustom the difficulty of tasks at hand. Google Docs is used as a collaborative tool for simultaneous documentation and report preparation. GitHub will be utilized to implement parts of the project concurrently after which merges and conflict resolutions will be solved in the weekly meetings. The project requires multiple new technologies for the members to learn such as web-development, Deep Learning, Android Studio, etc. therefore ensuring each member is able to focus on specific subjects rather than having to tend to every task is crucial for a good end-product and the aforementioned methods allow such an environment.

6.3. Taking Lead Role and Sharing Leadership on the Team

In order to ensure a collaborative work environment where everyone can equally participate as a leader and a team member, work has been divided into work packages. For every work package, a different person has been assigned as a leader. The leader of each work package is responsible for sharing the work related to the work package equally among other team members and following the progress on deliverables related to the package. The leader is expected to organize the team to meet the deadline of the given work package as necessary and each team member is expected to do the work assigned to them by the work package leader. If a team member is having trouble finishing their assignments, the leader has the role of following up on the work and making sure the work allocated to that member is finished. When deciding on which member should be assigned to which work package, a Zoom meeting was held where everyone picked whichever work package they were most comfortable with. The overall division of the work into work packages was also done in the same manner, prior to leader assignments.

7. References

[1] R. Eveleth, “How Many Photographs of you are out there in the world?” ,” *The Atlantic*, 03-Nov-2015. [Online]. Available: https://www.theatlantic.com/technology/archive/2015/11/how-many-photographs-of-you-are-out-there-in-the-world/413389/. [Accessed: 15-Nov-2021].

[2] “Contributed street view imagery policy,” Google. [Online]. Available: https://www.google.com/intl/en_uk/streetview/policy/. [Accessed: 09-Oct-2021].

[3] “Photoshop free trial | official adobe photoshop.” [Online]. Available: https://www.adobe.com/products/photoshop/free-trial-download.html. [Accessed: 9-Oct-2021].

[4] E. Then, “Pixel 6 magic eraser removes uninvited people from photos,” *SlashGear*, 20-Oct-2021. [Online]. Available: https://www.slashgear.com/pixel-6-magic-eraser-removes-uninvited-people-from-photos-19695941/. [Accessed: 15-Nov-2021].

[5] “Remove people from photo: The easy way,” *Inpaint*. [Online]. Available: https://theinpaint.com/tutorials/pc/how-to-remove-unwanted-people-from-photo. [Accessed: 09-Oct-2021].

[6] ”This person does not exist,” *This Person Does Not Exist*. [Online]. Available: https://thispersondoesnotexist.com/. [Accessed: 09-Oct-2021].

[7] D. Lumb, “Google Pixel 6 review,” *TechRadar*, 05-Nov-2021. [Online]. Available: https://www.techradar.com/reviews/google-pixel-6. [Accessed: 14-Nov-2021].

[8] “Unique, worry-free model photos,” *Generated Photos*, 2021. [Online]. Available: https://generated.photos/#. [Accessed: 14-Nov-2021].

[9] “General Data Protection Regulation (GDPR) – Official Legal Text.” https://gdpr-info.eu/ [Accessed: 22-Dec-2021].

[10] “Mevzuat Bilgi Sistemi.” https://www.mevzuat.gov.tr/mevzuat?MevzuatNo=6698&MevzuatTur=1&MevzuatTertip=5 [Accessed: 22-Dec-2021].

[11] E. by F. Jerga and F. Jerga, “REACT JS & Firebase Complete Course (incl.. chat application),” *Udemy*. [Online]. Available:

https://www.udemy.com/course/react-js-firebase-complete-course-incl-chat-application/. [Accessed: 15-Nov-2021].

[12] R. Farkas, “Who cares about privacy? surprising facts from around the Globe,” *Argonaut*, 2015. [Online]. Available: https://www.argonautonline.com/blog/attitudes-to-privacy-surprising-global-facts/. [Accessed: 15-Nov-2021].

[13] S. Karolius, “Perception of privacy – different cultures and different approaches,” *Behavioral Development Economics*, 10-Jul-2017. [Online]. Available: https://behavioraldevelopmentblog.wordpress.com/2017/07/10/perception-of-privacy-different-cultures-and-different-approaches/. [Accessed: 15-Nov-2021].