Bilkent University

Department of Computer Engineering

CS 492: Senior Design Project

Fakenstein

Final Report

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

Supervisor: Dr. Selim Aksoy
Innovation Expert: Adnan Erdursun
Jury Members: Erhan Dolak and Tagmac Topal

Final Report May 6, 2022

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

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].

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.

2. Requirements Details

2.1 Functional Requirements

The user should be able to:

- use the application either on their Android device or directly from the website
- open the gallery on his/her device to upload pictures.
- choose faces to be replaced either from the application’s tagged faces or by tagging them manually.
- edit the features of the generated faces if the program does not classify the faces correctly.
- user should be able to randomly blend a face using I feel lucky button (on mobile)
- manually blur any blended face that they did not like.
- undo any blurs made.
- select any of the suggested faces (on the web application).
- save the new picture.
- restart the whole process.
- view the tutorial.

The system should be able to:

- open the gallery on the user's device.
- open pictures from the gallery.
- detect all the faces in the picture.
- label the detected faces as the main subject or background.
- enable the user to add or remove faces to be replaced with generated faces.
- classify the detected face’s gender, skin color, and age group.
- generate fake faces that are compatible with the detected features of the original faces.
- enable the user to blur any face.

- generate suggested fake face options according to given face specifications.
- replace original faces with artificial ones as seamlessly as possible.
- save the final image to the user's device.

2.2 Nonfunctional Requirements

2.2.1 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.

2.2.2 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.

2.2.3 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.

2.2.4 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.

2.2.5 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 and image formats(JPEG/PNG).

2.2.6 Accessibility

- The Android version of the application should be available for anyone with an Android device.
- The web version of the application should be available for anyone with access to the internet.

2.2.7 Portability

- 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.

2.2.8 Efficiency

- The application should be able to run smoothly in older and newer generations of Android smartphones with Android Lollipop 5.0 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).

2.2.9 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.

3. Final Architecture and Design Details

3.1 Overview

This section presents details about the 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.

3.2 Subsystem Decomposition

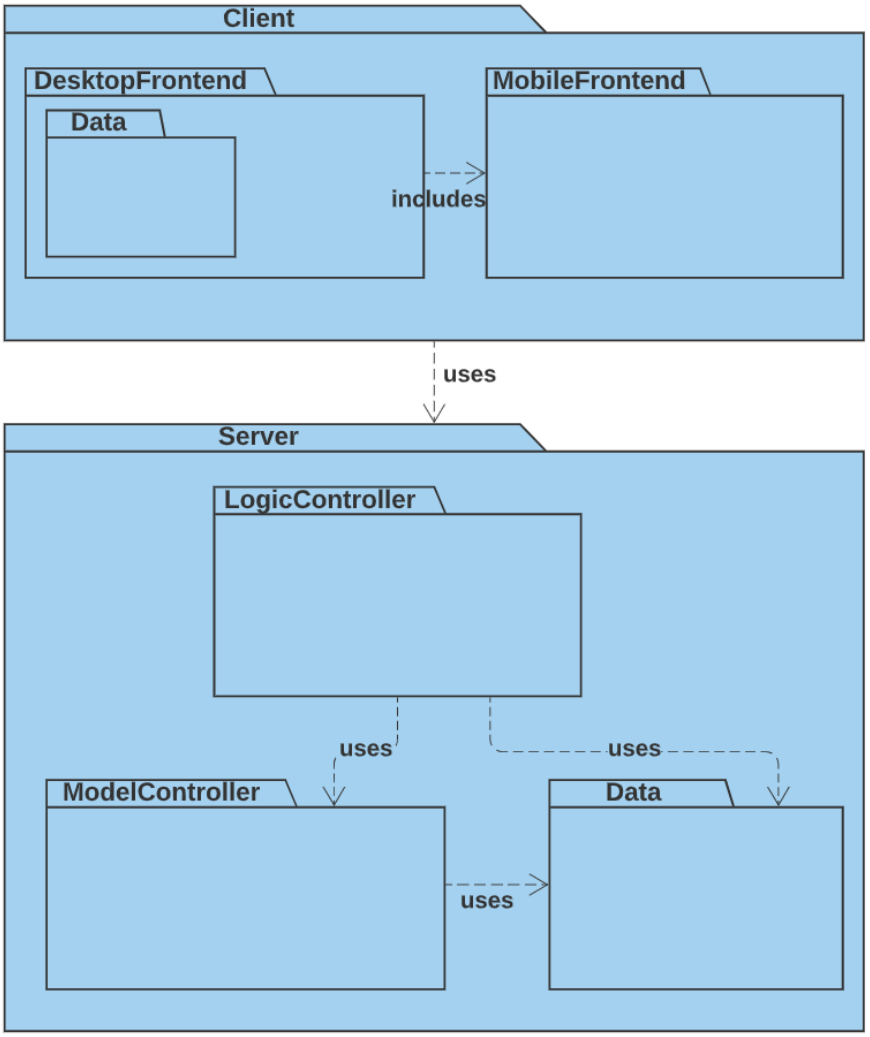

Figure 2: Overview of Subsystem Decomposition

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.

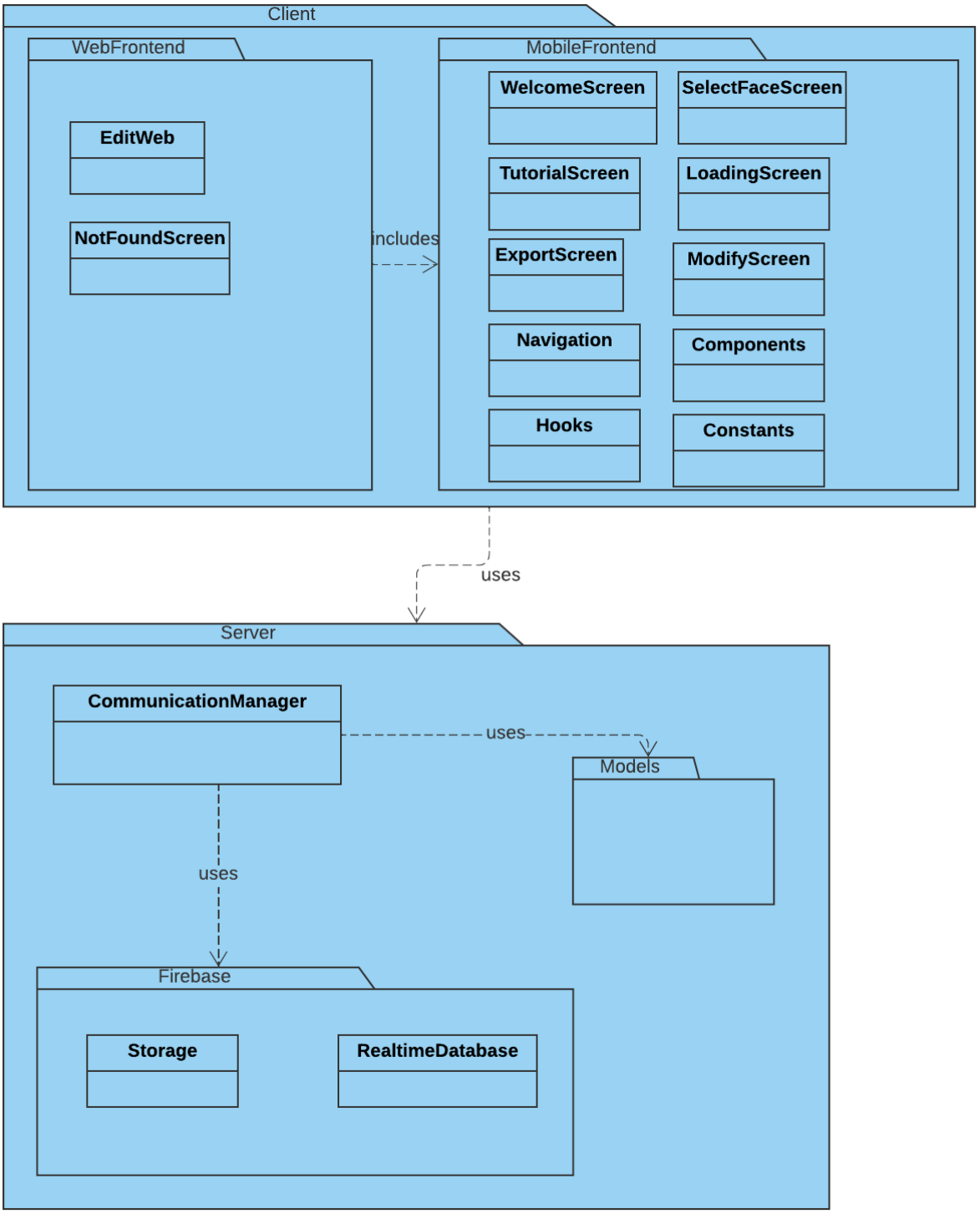

Figure 3: Detailed Subsystem Decomposition with Classes

Fakenstein will have the view layer which is divided into two subsystems: Web Frontend, which includes additional components, and Mobile Frontend. This layer will run on the client device which is either an Android device or a web browser in our case. Web 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 and Expo is used to be able to implement both subsystems simultaneously. 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 Web Frontend system has other components for its additional feature: Suggested Faces Selection 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 Machine Learning models inside the Models folder. Furthermore, the Communication Manager is used to create a bridge between the Firebase subsystem and the Client or the Models to increase efficiency. Server subsystem will generate the new image based on the requests and send the answer back to the Client.

In the Firebase subsystem inside the Server Tier, Fake Faces and hyperparameters and weights for deep learning models are kept in an online database. 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 storage class that holds these Fake Faces increases efficiency and robustness of the system.

The Model subsystem includes different ML models which will be used and initialized by the values in the Firebase Storage. After initialization, the Models also takes predictions from the required models and modify the image according to parameters and requests coming from the Communication Manager.

3.3 Hardware/Software Mapping

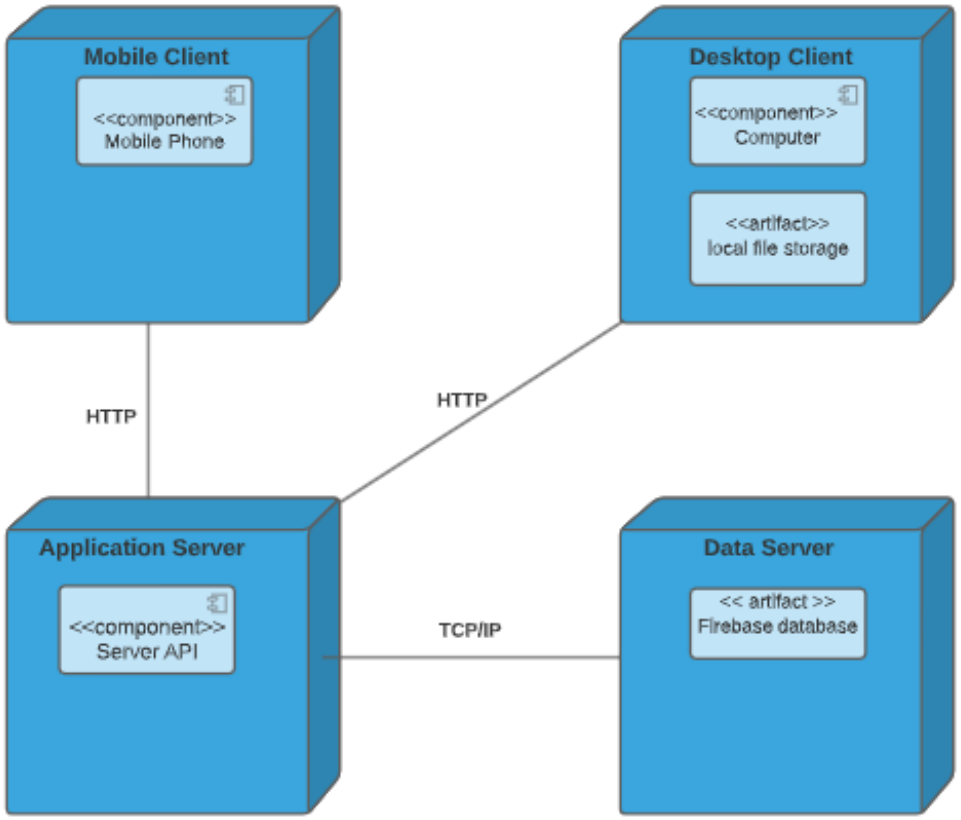

Figure 4: Deployment Diagram

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 web 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.

4. Development/Implementation Details

4.1 Deep Learning Models

4.1.1 Face Detection

For face detection purposes, we made use of the MTCNN library already available [11]. We used MTCNN due to its high accuracy and lightweight computation. We used the detector directly to detect boundary boxes of faces in the input images. Hence, no additional training was performed.

4.1.2 Face Classification

In order to classify faces in images according to their age, gender and skin color, we trained ResNet classifiers [12] on CelebA dataset [13]. However, in the CelebA dataset, we observed data imbalances causing our model to have a bias towards one label. Therefore, we balanced our dataset to have an equal number of both positive and negative labels. Training was done according to the labels in the CelebA dataset: for gender CelebA had the label “male”, for skin color it had the label “pale” and for age group it had the label “old”. One problem we encountered was with skin color, as the samples with negative “pale” label were not necessarily “dark”, but rather a combination of “not pale” skin colors. Therefore, after training, we realized that our classifier was having a hard time identifying the difference between pale and not pale samples. To eliminate this problem, we retrained our classifier on UTKFace dataset [14], where the images are labeled according to different races from white

to black and even Asian. For simplicity, we limited our labels as white for negative and black for positive labels. After these changes we observed that we were able to classify light and dark skin colors successfully.

4.1.3 Face Generation

We first trained a Conditional GAN model to generate artificial faces according to given attributes (age, skin color, gender). However, this approach was unsuccessful as training the GAN model required high computation time and high GPU power that was unavailable to us. The model was required to be trained for approximately 100 epochs to get a decent result, which we couldn't achieve due to computational constraints. Therefore, we decided to use a StyleGAN [15] model to generate artificial faces, independent of the attributes. StyleGAN is a well-known and well-researched GAN model, known to produce high-quality artificial faces. After creating artificial faces with the StyleGAN model, we used our Resnet classifiers(age, skin color, gender) to classify the generated samples and populated our database with the generated samples and their corresponding labels accordingly.

4.1.4 Face Blending

In order to blend two faces -the artificially generated one and the original one- we first used a pre-trained shape predictor model from dlib [16] to detect the 68 landmarks of both faces. After detecting these landmarks, the faces were aligned using the landmark points, and the combined image was color corrected to achieve a more realistic appearance using OpenCV library [17].

4.2 Backend and Frontend

4.2.1 Backend

We didn't deploy our server, instead we used a local server which can be reached by any device in the network. We didn't deploy our server to available services due to cost issues. With AWS, the Free Tier offers an EC2 server with 1GB of memory, which is not enough to deploy our deep learning models as they require pre-trained models to be loaded to the server as well as several libraries to be installed, which exceed the free storage limit.

We implemented the backend of our application using Python and Flask framework. Our backend receives HTTP requests from the frontend and runs the deep learning models accordingly and responds with the output image/data. We created four different app routes for the purpose of communicating with the frontend. These routes are:

- Detect route
- Replace route
- Blend route
- Blur route
- Suggested faces route

- Suggested swap route
- I feel lucky route

Detect route: Detect route takes the image from the HTTP request and runs face detection and foreground/background classification algorithms to produce an array of face boundary boxes, each labeled with their respective foreground/background detections and returns this information back to the frontend.

Replace route: Replace route takes the image from the HTTP request and an array of faces to be replaced(only boundary boxes) and runs age, gender and skin color classification algorithm on those faces. After classifying each face, it retrieves a suitable fake image from the online Firebase database and replaces each face with the suitable fake faces using the blending algorithm. The finalized image is encoded as a Base64 string to be sent back to the frontend.

Blend route: Blend route is used when the user changes certain classifications of the face attributes(age, gender, skin color) manually and wants to re-blend the faces accordingly. Since face attributes are retrieved from the user, this route receives the new attributes, associated boundary boxes and the image from the HTTP request. After receiving this information, it retrieves a suitable fake image from the online Firebase database and replaces each face with the suitable fake faces using the blending algorithm. The finalized image is encoded as a Base64 string to be sent back to the frontend.

Blur route: Blur route is used when the user wants to blur a detected face. It receives the boundary box of the face to be blurred in addition to the image from the HTTP requests. Then it runs the blur algorithm to blur the face and returns the finalized image as a Base64 string.

Suggested faces route: Suggested faces route is used when the user wants to view suggested artificial faces that can be used to be replaced with the original face. After receiving the age, gender and skin color information of the original face from the HTTP request, it retrieves three random artificial faces from the database that are fit with the given facial attributes. These faces are sent back as Base64 encoded strings.

Suggested swap route: This route is taken when the user selects one of the suggested artificial faces to replace with the original face. The selected face in addition to the original face are sent through the HTTP request. The received faces are blended and sent back as a Base64 string back to the frontend.

I feel lucky route: I feel lucky route is used when the user clicks the “I feel lucky” button. The purpose of this functionality is to provide a fun face blending alternative. Only the original face is received from the HTTP request. After receiving the face, a completely random artificial face is selected from the database and blended with the original face. The resulting image is sent back as a Base64 string back to the frontend.

4.2.2 Frontend

We developed our frontend using the Expo framework to build React Native applications for Android, iOS, and web. [18] Expo is chosen as it is optimized for creating fast and beautiful UI's practically for different platforms. Expo provides its users with useful tools such as React Hooks which increases reusability, readability, and testability that improves the quality of the product while decreasing the time for implementation. Expo apps are written in TypeScript language which has optimized features for UI developing such as async-await functions enabling event-driven design.

In the implementation stage of Fakenstein, we initially planned on developing a desktop app and an Android app. However, during the implementation of the Expo project, we realized the advantages of Expo where we could be working on an Android application parallel with a web application. Hence, after deliberation we decided that providing a web interface to our application would facilitate development and increase accessibility to users when compared to a desktop application.

Therefore, the two 2 members of the team were selected to be responsible for Expo frontend development where either one is working on native capabilities of Android and web. After the general navigation screens were implemented, the team figured out native API's for accessing users' libraries to retrieve and write files into for images. This stage required knowledge on representation of image types of different platforms and accessibility permissions. In the final stage the communication with the server side was implemented using Expo's HTTP 'fetch' calls and the Fakenstein app became fully functional.

During the implementation, the team used github to coordinate their works by merging and branching. For the sake of consistency in code variable naming style etc. was decided on and the commit was reviewed by another team member before merging. We could not implement test automation due to time constraints since testing is a later stage in waterfall application development. The main method for testing Fakenstein while in the implementation stage was the Expo Go app and the web. With the help of the hot-reload feature of Expo Go, it was possible to integrate the changes on the code to a real device instantly and without the need for re-bundling without connecting the device to the computer, thus making the testing fast and efficient.

5. Testing Details

The main testing activity in the development was doing manual testing through using the application. While doing so, we tried to find and eliminate bugs, crashes, and edge cases while also trying to improve the user experience and optimize the code until the application was up to the standards we set before starting.

5.1 Non-functional Testing

There were certain technical criteria that Fakenstein should achieve such as performance. Therefore, we wanted to make sure that the application did indeed achieve the said criteria.

We simulated cases to understand the performance of Fakenstein with respect to non-functional requirements, and if the performance was lacking, we optimized such non-functional requirements.

5.2 UI Usability Testing

While the application may be straightforward and easy to use for users such as the developers of the app, or any person who has had experience with similar applications, we also wanted to see the perspective of an outsider. We requested some of our friends and family to give Fakenstein a try and get their feedback on it. We deliberately chose friends and family that were not much involved with engineering. This way we were able to get user feedback and adapt our UI with respect to the reported complaints.

5.3 Debug Tools

For debugging we relied on built-in tools of Visual Studio Code. The debugger allowed us to easily identify our mistakes and unexpected scenarios where our logic failed.

5.4 User Experience

It is important to mention that all of the developers that contributed to the project have had experience with desktop and web applications that relied on machine learning models. So, it was not difficult to adapt a user-side perspective to our application as we literally used the tool ourselves as we developed it. Previous experience in the field helped us understand which parts were adequate and which parts could be improved.

6. Maintenance Plan and Details

In Fakenstein, we devised different plans considering the maintenance of the components that make up the application. These plans include frontend, backend, and ML model maintenance.

6.1 Maintenance Plan of the Backend

Since we didn’t deploy our application to any cloud service provider, our current maintenance plan of the server only includes the maintenance of the local server. However, once we deploy it to a cloud service provider(such as Amazon Web Services(AWS)), we will need to consider how to keep the server running with the minimum cost and maximum efficiency. To achieve this, we will look at the memory and CPU usage metrics. This is also applicable to our local server as for the time that server is running on the computer, these metrics are crucial to understand how to maintain the services provided. To maintain the Firebase Database, we will not only make sure that this third party online database will continue to provide service for our application but we will regularly populate our dataset with new images to increase the diversity of face options we provide to the user, while keeping an eye on the disk usage. We also plan on regularly checking whether the packages used in the backend side are deprecated or updated and update our backend accordingly.

6.2 Maintenance Plan of the Frontend

In order to maintain our Expo frontend, we will be required to follow new permission regulations of changing platforms and the verification of functionality of our application. Currently our first priority towards maintenance would be to implement test automation according to our requirements. After confirming verification through model and integration testing, we would move to host our backend server in the cloud and then apply for deployment to Google Play Store.

We already have our application registered for deployment to Android Play Store through the Expo framework. However, we could not move forward with deploying due to the lack of accessible and free cloud servers large(in terms of memory) and fast(in terms of CPU/GPU) enough to deploy our Machine Learning models.

6.3 Maintenance Plan of the Machine Learning Model

We plan on regularly training our models on larger, more diverse datasets with different hyper-parameter values to see if we can obtain a model that performs better than the ones we have trained so far. Moreover, we plan to keep up with the latest articles released regarding our subject, to see if any of those new models or approaches can be applicable to our problem and adapt our models accordingly.

7. Other Project Elements

7.1.Consideration of Various Factors in Engineering Design

There are various factors to consider for Fakenstein in the relevance of engineering design. Firstly, the collaboration of the development team in which managing and engineering tasks were distributed considering each group member’s strengths and previous experiences. Communication among the group was kept fresh with weekly meetings, whether online or face-to-face, in which each group member would report their progress on their given part. Each time an issue came up, it was raised in our Whatsapp group chat, and then discussed with the people relevant to the issue through a Zoom meeting. Github Issues were also used to track our progress and problems encountered throughout CS491/CS492. This allowed us to be transparent in Fakenstein management, allowing us to design a robust system. With expert opinions from Erhan Dolak and Tağmaç Topal, we had opportunities to discuss what we wanted Fakenstein to be like and how we would go about implementing what we wanted.

Data privacy of the Fakenstein is another topic we considered in our engineering design. After all, the application deals with photos, which are private to the user. Therefore the users had private information in the app, which was made inaccessible by other users, solving the security problem.

Additionally, progression in the field of machine learning was another factor considered in our engineering design. The core part of Fakenstein is that it uses the deep learning model

to replace faces with generated ones. It is expected that the model will evolve through time and usage of the application due to improvements in models that are used as components in Fakenstein. The system in Fakenstein is built in such a way that those changes are welcome and can easily be accommodated into the system.

Lastly, Fakenstein also has public safety, cultural, social and economic factors which will be discussed in the following subsections.

7.1.1 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.

7.1.2 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 [7, 8].

7.1.3 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 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.

7.1.4 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 application will provide a free alternative as opposed to Adobe Photoshop which is 21 dollars a month [9].

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 application will be free of cost for the time being.

7.2.Ethics and Professional Responsibilities

Fakenstein is deeply involved with data privacy which brings about its ethical and professional responsibilities.Our background research has revealed that some of the previous Deep Learning implementations that also deal with identifying faces and some of its properties have suffered from biases due to different amounts of data for each class [10]. Therefore, we are responsible to make sure that our dataset is not tweaked or skewed for particular characteristics to be able to train our models properly with equal amounts of data for classes (gender, race, age as well as pose properties). Moreover, we will be utilizing datasets, libraries and implementation techniques from outer sources so we must make sure that any help we get from outside sources are properly cited and that we do not violate any licensing for them. Furthermore, the program will be accessing the user's gallery and modifying images therefore we have to request appropriate permissions and make sure to only make changes in the user's local device and not store their information on an outside storage. Another professional responsibility we have is to make sure the application covers the users' needs even in cases where the Deep Learning models are insufficient. Thus, we will be adding a manual blurring option for the cases where the user wants to hide parts of the image that are not faces or faces that are too distorted in the image that it does not register as one in the identification model. A Terms and Conditions agreement will have to be

accepted in order to use this application. In the agreement, the users will have to accept not to use this application for manipulating photos for unethical purposes.

7.3.Judgements and Impacts to Various Contexts

7.4 Teamwork Details

7.4.1) 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	Cansu Moran	All members
WP3	Face Identification	Öykü Hatipoğlu	Cansu Moran, Ardahan Doğru
WP5	Face Modification	Cansu Moran	Elif Kurtay, Atakan Dönmez
WP6	Expo GUI	Elif Kurtay	Atakan Dönmez
WP7	Database Integration	Atakan Dönmez	Öykü Hatipoğlu, Cansu Moran
WP8	Testing	Elif Kurtay	All Members

Gantt Chart for planned Work Package start and end dates:

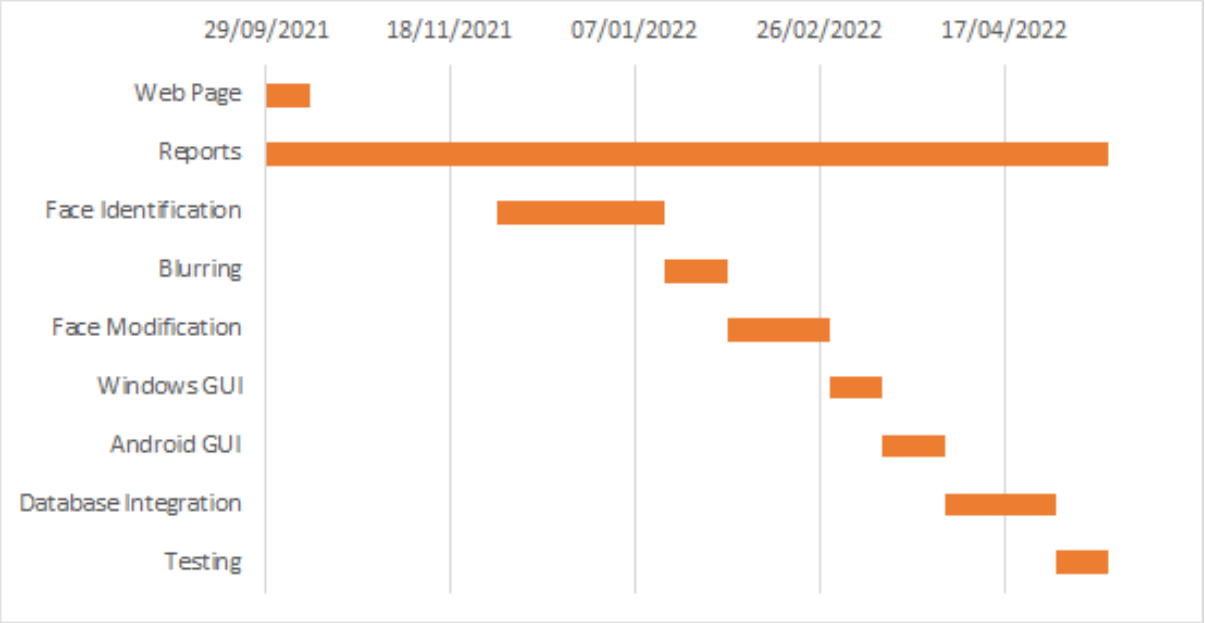

Figure 12: The Gantt Chart.

Work Packages:

Work Package 1: Web Page	
Start Date: 29/09/2021	**End Date:** 11/10/2021
Leader: Ardahan Doğru	**Members Involved:** Atakan Dönmez, Öykü Hatipoğlu
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: Cansu Moran	**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: ExpoGUI	
Start Date: 01/03/2022	**End Date:** 15/03/2022
Leader: Elif Kurtay	**Members Involved:** Atakan Dönmez
Objectives: Design and implement a web and Android app for the functionalities	
Tasks **Task 6.1:** Design the UI **Task 6.2**: Implement the GUI **Task 6.3:** Combine the frontend with the backend **Task 6.4:** Implement platform specıfic requirements	
Deliverables	

Deliverable 6.1: Web and Android app

Work Package 7: 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: Elif Kurtay	**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.	

7.4.2) 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.

7.4.3) 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.

7.4.4) Meeting objectives

In order to meet deadlines and requirements for our project, we devised the work packaging method. Every member was given a job according to their strengths and previous work experience. Therefore we used the divide-and-conquer strategy to meet deadlines and requirements. In the case a problem occurred, the team leader would take charge and decide what to do for that certain work package. This way all requirements and deadlines were achieved smoothly.

7.5 New Knowledge Acquired and Applied

While the team had prior experience with the components of our project such as application development and deep learning, it was inadequate to build up Fakenstein right away. Therefore, there was a need to learn and acquire new knowledge to build up the application. Some topics that we focused on were:

- Expo
- Flask
- React/React Native
- GAN

8. Conclusion and Future Work

In conclusion, we were able to achieve the goals that we set for ourselves at the beginning of the academic year. Frankenstein is up and working, it is able to take image type inputs from users and it is able to blur or replace the faces in the image, with corresponding inputs from the user. It is free and open-source, anyone can access the code, download it, and use it via Github. We are content with how far we have come as well as what Fakenstein came to be. While there are still areas to work on, in general, we all improved our communication, application development, and managing skills.

In the future, naturally, we want to further improve Fakenstein. While the development wasn't rushed, it wasn't easy-going either. With more time and experience, we believe that we may optimize certain aspects and make the app even more polished. Another future goal is to make the application marketable and therefore profitable. Therefore we want to make Fakenstein as a product deliverable in the market. Thus, the next step for us will be to modify the app according to the feedback we are going to receive from our professors and colleagues. After that, we aim to commercialize Fakenstein so that we may distribute and collect greater feedback which will improve the application further.

9. Glossary

JPEG	Joint Photographic Expert Group Image
PNG	Portable Network Graphics
AWS	Amazon Web Services
JVM	Java Virtual Machine
IDE	Integrated Development Environment
ML	Machine Learning
DL	Deep Learning
SDK	Software Development Kit
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
GAN	Generative Adversarial Network

10. 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] 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].

[8] 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].

[9] “Adobe Creative Cloud Plans, pricing, and membership.” [Online]. Available: https://www.adobe.com/creativecloud/plans.html. [Accessed: 15-Nov-2021].

[10] A. Palmer, “Study finds popular face ID systems may have racial bias,” *Daily Mail Online*, 12-Feb-2018. [Online]. Available: https://www.dailymail.co.uk/sciencetech/article-5382979/Study-finds-popular-face-ID-systems-racial-bias.html. [Accessed: 15-Nov-2021].

[11] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] Z. Zhang, Y. Song, and H. Qi, “Age progression/regression by conditional adversarial autoencoder,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative Adversarial Networks,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] “Dlib C++ library,” *dlib C++ Library*. [Online]. Available: http://dlib.net/. [Accessed: 06-May-2022].

[17]*OpenCV*, 03-May-2022. [Online]. Available: https://opencv.org/. [Accessed: 06-May-2022].

[18] Expo Documentation, 03-May-2022. [Online]. Available: https://docs.expo.dev/. [Accessed: 06-May-2022].