



Middle East Technical University Northern Cyprus Campus
Computer Engineering Program

CNG492 Computer Engineering Design II

Iris Analyzer

2269421 Aref Khademi Najafabadi
2246452 Sameh Farid S. Algharabli
2269454 Evans Muthugumi Kimathi
2175263 Elbaraa Elsaadany

Supervised by
Asst. Prof. Dr. Meryem Erbilek

Final Report

Contents

1	Introduction	3
1.1	Motivation	3
1.2	Problem Formulation	3
1.3	Aims and Objectives	4
1.3.1	GUI objectives:	4
1.3.2	Modelling and Prediction objectives:	4
1.4	Stakeholders	6
2	Identification of Constraints	7
3	Literature Review	8
4	Research Method	11
5	Requirements Specification	13
5.1	Functional Requirements	13
5.2	Non-functional Requirements	15
5.3	Assumptions and Justifications	15
5.4	Structured Use-case Diagram	16
5.5	High Level Sequence Diagrams	17
5.6	Graphical User Interface	19
6	System Architecture and Models	22
6.1	Architecture Model	22
6.2	Process Model	23
6.3	Data Flow Models	25
6.4	Class Diagram	29
6.5	List of Data Structures and Data Types	29
6.6	List of Special Algorithms	30
7	Project Management	31
7.1	Software Estimations	31
7.1.1	Complexity Adjustment Table	32
7.1.2	Line of Code	33
7.1.3	Estimate the Efforts	33
7.2	Milestones, Main Project Tasks and Task Allocation	35
7.3	Gantt Chart	37
8	Software Implementation	38

9	Software Testing	40
9.1	Testing Strategy	40
9.2	Unit Testing	40
9.2.1	Preprocessing Unit	40
9.2.2	CNN Model Unit	42
9.2.3	Admin Unit (GUI)	47
9.2.4	User Unit (GUI)	49
9.3	System and Integration Testing	51
9.4	Performance and System Testing	53
9.5	Recovery Testing	54
9.6	Security Testing	55
10	Conclusion	56
10.1	Retrospective	56
10.2	Future Work	57
11	References	58

Chapter 1

Introduction

1.1 Motivation

In the recent years, bio-metric technology (mostly fingerprint and iris recognition) has grown, improved and is in large scale use in identification. Large companies are using iris recognition and fingerprint software to identify their employees. Not only in big companies but small devices such as phones, tablets and even personal computers are integrated with bio-metric devices such as fingerprint readers and iris recognition to login and even make payments, a good example is Samsung phones which use both fingerprint and iris sensors for login.

Iris recognition is not in much use as fingerprint, yet it is more secure: no 2 people have the same iris. Some people do not have fingerprints as its lost due to medical issues, as explained by Katherine Harmon[4] or working environment. So why don't people use iris recognition more?

This is the main motivation of our project, to show that iris recognition can give accurate results and can be implemented in an easy and simple way.

Iris recognition can also be used for gender prediction, which can complement identification in the case that identification of a person cannot be acquired. Imagine a case where the identity (name) of a person cannot be identified after a scan in the employer's/police database. How can you try to figure out the person? If the scan can predict the gender of the image, the employer's/police can have a place to start. They know that they are searching for a female or male.

The Iris analyser system shall take an image from user and return the identity if it exists in the database. It shall also predict the gender of the image.

1.2 Problem Formulation

Given the widespread use of classical texture descriptors (texture features that are extracted from the iris) for iris recognition, including the Gabor phase-quadrant features(for extracting these features) , it is instructive to take a step back and answer the following question: how do we know that these image processing feature descriptors proposed in the literature are actually the best representations for the iris? Furthermore, can we achieve better performance (compared to the Gabor-based procedure) by designing a better feature representation scheme that can perhaps attain the upper bound on iris recognition accuracy with low computational complexity?

Deep learning techniques are one of the best ways to solve the above problem as it increases the accuracy, by using improved feature extraction techniques, that is primarily data-driven.

Use of Convolution Neural Networks (CNN) models to train, classify and test images, gives better accuracy than previous image processing techniques. Images to be used in the CNN should be preprocessed: cropped, converted to numpy arrays and normalised (image pixels divided by 255 to make the pixels in a range of 0-1 for easier and faster processing).

1.3 Aims and Objectives

There are two main aims of this project:

1. User identification: The user shall select/scan an image and the identity CNN model (integrated into system) shall predict the user identity (e.g. User 001).
2. Gender prediction: The user shall upload an image and the gender CNN model (integrated in the system) shall predict the gender (either male or female).

The objectives of the project are as follows:

1.3.1 GUI objectives:

1. The user shall select/upload an image as input and choose output wanted either identification, gender or both.
2. Preprocessing, where the eye image will be segmented using image processing techniques. This step will partition the eye image into image objects (iris and pupil) and this will be used to locate the objects and their boundaries (diameter and radius of pupil and iris). This will help us work on the image in the next stages.
3. After cropping/segmentation, the image shall be converted to a numpy array (the CNN models use numpy arrays format). The image shall then be predicted by the CNN model depending on the output needed:
 - (a) Identification output selected: The image shall be predicted by the identification CNN model.
 - (b) Gender output selected: The image shall be predicted by the gender CNN model.
 - (c) Both Identification and Gender: predicted by both identification and gender CNN models.

1.3.2 Modelling and Prediction objectives:

1. **Image Acquisition:** Images shall be retrieved from the BioSecure Multimodal Database (BMDB) which we have available. The BMDB is a database containing 1600 eye images from 200 users: images include 8 eye images from each user, 4 from each eye (4 left and 4 right). The eye images are taken from an office space.

The database also includes:

- (a) File containing Iris and pupil radius dimensions that are useful for segmentation process.
- (b) User names (e.g. User1) and eye side (left or right)for each image in the database. This is useful for labelling in the identity process.
- (c) File containing gender information of each eye image (either male or female). This is used during gender analysis process.

2. **Preprocessing/Segmentation/cropping:** The images acquired from the database are cropped using the iris and pupil images. This is done to remove all unwanted parts of the eye like the eyelids, eyelashes and eyebrows.

The second part of preprocessing is to create training, validation and testing datasets that shall be used by the CNN models. They shall be divided into 50% training set, 25% validation set and 25% testing set.

- (a) Training set: The images in the training set shall be used to train the CNN models.
- (b) Validation set: The images in the validation set shall be used to check the accuracy of the training process. The hyper-parameters of the model shall be tuned to improve the validation accuracy.
- (c) Testing set: The images in the testing set shall be used to evaluate the model. If the accuracy is lower than expected, the training process is redone, with more images or with better hyper-parameters.

The last part of preprocessing is to convert the images in the datasets to numpy arrays and normalize them. The CNN models uses numpy arrays not images. The numpy arrays are finally normalised: converted to a range between 0-1 for easier and faster processing because the numbers are now smaller.

3. **Creating, training and validating CNN models:** Both the identification and gender CNN models are created. Each model shall contain different layers and different parameters to train and validate as shall be explained later.

After creation, the preprocessed images (training set) shall be trained and validated using the validation data-set. After each training and validation process, the hyper-parameters of the model shall be tuned to try and improve the training accuracy.

4. **Testing the CNN models:** Both the gender and identification CNN models shall be evaluated using the testing set data-set. The accuracy achieved shall determine if it shall be retrained or saved to be deployed in an application.

1.4 Stakeholders

1. System administrator : The people responsible for creating and deploying the iris analyser system.

Has the following functions:

- (a) Preprocessing the eye images.
- (b) Create the identification and gender CNN models.
- (c) Train, validate and test the CNN models.
- (d) Create the GUI for the system.
- (e) Testing the system as a whole.

2. Application users : The people that will use the iris analyser system to get prediction of both the identification and gender.

The users shall use the app as follows:

- (a) Upload an image as input.
- (b) Select output option wanted (either identification or gender or both)
- (c) Get output which is the prediction of the image uploaded.

Chapter 2

Identification of Constraints

We have ethical constraints. Eye images acquisition: Collecting people's images to use in the database was impossible because of ethical concerns. People are not comfortable with their data being used as they fear leakage of personal data or miss-use of their personal information.

Chapter 3

Literature Review

Iris analyser research has been done and discussed by many researchers. The process of retrieving data from the iris is difficult and has been improved over time, shifting from basic image processing techniques to complex deep learning techniques such as CNN. Below we discuss different ways to develop iris analyser system by researching different ways and choosing the best performing techniques.

In Tianming Zhao et al [7]. , they start by doing pre-processing on the iris image where they locate and segment the iris texture part from the original iris image through quality evaluation. The Region of Interest(ROI) is extracted, normalized and enhanced. In our model we are going to do pre-processing as discussed above.

They move on to convolution step where low level features are extracted from pixel intensities of the input images and form the primary features used for the primary capsule layer. Two strategies are used to construct the convolution part:

1. Applying transfer learning method: The entire network is divided into sub-networks with the vector part and a series of different networks is instantiated. This gets the outputs of different levels in the same network as the result of the convolution part to find the most suitable option of convolution structure for the entire network through experiments.
2. Building several several shallow convolution networks as the convolution part without pre-training.

The next step is the use of routing algorithm in the vector layer and is done in 2 stages:

1. Dynamic Routing in Capsule Network: provides non-linear mapping for two adjacent capsule layers. The dynamic routing algorithm uses the cosine similarity of two vectors to measure their agreement. However, this manner is not quite good at judging quite good agreement and very good agreement, which usually makes it difficult for the network to converge after training in the experiments.
2. Adjusted Dynamic Routing: Used to improve applicability of the capsule structure and make it easy for the network to converge when dealing with input images of large size.

The last step us the network architecture which consists of the convolution part, vector part, classification and reconstruction.

Finally, they simulate an environment with different illumination intensities, and train and test iris images with different pupil sizes (represent different light intensities) to show the recognition ability of the method when the environment is varied. Experiments show that a deep network with capsule architecture is feasible in iris recognition. In the test of the JluV3.1 iris dataset, InceptionV3-5blocksCDRDL achieves the highest accuracy of 99.37 percent. In the test of the JluV4 dataset, Iris-DenseCDRDL achieves the highest accuracy of 99.42 percent. In the test of the CASIA-V4 Lamp dataset, VGG16-4blocksCDRDL achieves the highest accuracy of 93.87 percent.

Kien Nguyen et al. [9], start by segmenting the eye image. Here, they localize the iris by extracting two circular contours pertaining to the inner and outer boundaries of the iris region. This is followed by normalization step which maps the iris to a region of fixed dimension. This helps to remove the effects caused by dilation and contraction of the pupil causing iris stretching.

The normalized iris image is fed into the CNN feature extraction module to extract features. They use 5 state-of-the-art and off-the-shelf CNNs (AlexNet, VGG, Google Inception, ResNet and DenseNet). We use ResNet and DenseNet as discussed above to extract the features.

Note that there are multiple layers in each CNN. Every layer models different levels of visual content in the image, with later layers encoding finer and more abstract information, and earlier layers retaining coarser information. One of the key reasons why CNNs work very well on computer vision tasks is that these deep networks with tens or hundreds of layers and millions of parameters are extremely good at capturing and encoding complex features of the images, leading to superior performance. To investigate the representation capability of each layer for the iris recognition task, we employ the output of each layer as a feature descriptor and report the corresponding recognition accuracy.

The extracted CNN feature is fed to the Support Vector Machine (SVM) classification module for classification. The multi-class SVM for N classes is implemented as a one-against-all strategy, which is equivalent to combining N binary SVM classifiers, with every classifier discriminating a single class against all other classes. The test sample is assigned to the class with the largest margin.

CNNs are effective in encoding discriminative features for iris recognition. However according to Kien et al., deep learning offers the following problems and open questions:

1. Computational complexity: very high during training phase due to millions of parameters used in the network.
2. Domain adaptation and fine tuning: entails freezing early layers and only re-training a few selected later layers to adapt the representation capability of the CNNs to iris images. Fine tuning is expected to learn and encode iris-specific, as opposed to generic image features.
3. Few-shot learning: limited number of training images can be partially solved with few-shot learning, which allows the network to perform well after seeing very few samples from each class.
4. Architecture evolution: evolve off-the-shelf CNNs in order to generate powerful networks that are more suited for iris recognition.

5. Other architectures: Architectures like Deep Belief Network(DBN), Stacked Auto-Encoder(SAE) and Recurrent Neural Network(RNN) have their own advantages and can be used to extract features for iris images.

Iris analysing can be used for gender prediction according to Sreya K C and Dr. Rini Jones S B[5]. They use a neural network with 20 neurons, 8000 epoch, and 0.01 learning rate. The preprocessing shall be done the same as the identification part. The only difference is in modelling part where we shall use 2 Conv2D layers, 2 MaxPool Layers, Relu layers and a fully connected layer. We shall use deep learning CNN model for our gender model.

We employ the CNN method as discussed by Kien above, and lastly check the percentage accuracy the model gets, if the accuracy is not good enough(below 90 percent), we add more layers to the CNN and train the model more until we achieve the goal.

Chapter 4

Research Method

The project will be done in three parts: image preprocessing part, CNN modelling part and prediction part.

Image preprocessing techniques shall be used to segment the eye images by removing the unwanted parts like eyelids, eyelashes and eyebrows. The iris and pupil radius dimensions shall be used to crop/segment the image to leave only the iris and pupil parts.

The image processing part will be achieved using the following methods:

1. Images shall be acquired from the BioSecure Multimodal Database(BMDB).
2. Segmentation: The eye images shall be cropped using the iris and pupil radius dimensions.
3. Creating datasets: Three datasets shall be created: 50% training set, 25% validation set and 25% testing set. The segmented images shall be appended to the datasets according to the ratio mentioned above.
4. Conversion to numpy arrays and normalising: The images shall be converted to numpy arrays to be used in the CNN models. The numpy arrays shall be normalised (by dividing each pixel by 255 to get a value between 0-1) for faster and easier processing in the CNN models.

The second part is the CNN modelling. Two CNN models shall be created: identification CNN and gender CNN and afterwards shall be trained and validated. They shall be done as discussed below:

1. **CNN model Creation:** The CNN (Convolution Neural Network) models were created using python programming language, using Keras library. They are made up of layers which contain the number of filters and the filter sizes. The difference between the 2 models is the number of layers and the number of filters and filter sizes in each layer.

The layers used include:

- (a) Input Layer: takes the input data
- (b) Conv2D Layer: This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

- (c) MaxPooling Layer: Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window for each channel of the input.
 - (d) Dropout Layer: Applies Dropout to the input. Drop some the layers from the previous layer.
 - (e) BatchNormalization Layer: Layer that normalizes its inputs. Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.
 - (f) Flatten Layer: Flattens the input. Makes it into a 1D object.
 - (g) Dense Layer(Output Layer): The output layer (classifies the output).
2. **CNN model training and Validation:** The CNN models shall be trained and validated using the training and validation sets. The validation accuracy shall determine whether the training process needs to be fine tuned using the hyper-parameters.
 3. **CNN model evaluation/testing:** The models shall be evaluated using the testing set. The accuracy value attained shall be compared with the expected accuracy and if similar or greater it shall be saved for future use in prediction. If lower than expected accuracy it retraining shall be redone.

Chapter 5

Requirements Specification

5.1 Functional Requirements

1. The system shall use an authentication for the system administrator.
 - (a) The system shall display an authentication message to the administrator.
 - (b) The system shall give the administrator access to the database if the pin is correct.
 - (c) The system display an error message if the pin entered is correct.
 - (d) The system shall display a add confirmation message to the system administrator.
2. The system administrator shall be able to add eye images in the database.
 - (a) The system shall authenticate the system administrator.
 - (b) The system shall send the added images to the database.
 - (c) The system shall display an added image successfully message to the system administrator.
3. The system administrator shall be able to create a model to be used in deep learning section.
 - (a) The system shall authenticate the system administrator.
 - (b) The system administrator shall divide the eye images in the database into 2 parts: 50 percent training set and 50 percent testing set.
 - (c) The system shall save the 2 sets in the database.
 - (d) The system shall use the training set for modelling and the testing set for prediction using deep leaning algorithms.
4. The user shall be able to upload eye image as inputs.
 - (a) The user shall upload eye image as input.
 - (b) The system shall display on screen the image chosen by user.

- (c) The system shall send a copy of the chosen eye inputs to the next stage (pre-processing stage).
5. The user shall be able to choose the expected output (identification and gender output).
- (a) The system shall read checkbox input of either or both identification and gender.
 - (b) The system shall display a confirmation message for the selected output.
 - (c) The system shall use the chosen input to process the data after the output confirmation from the user.

5.2 Non-functional Requirements

1. The system shall be available to users 24hrs a day so that users can access it at anytime in the day.
 - (a) The system shall check every 2hrs (system troubleshoot) that all parts are functioning properly.
 - (b) The system shall send a notification to the system administrator about any errors that occurred.
2. The system shall work on Windows, Linux and Mac operating system to work on all devices and be accessible to all users.
 - (a) The system shall run on all devices with Windows 7 and above, all Linux and Mac devices.
 - (b) The system shall display an error message if the device is incompatible.
3. The system interface input response time should not be more than 1 second to users for fast interaction process.
 - (a) The system shall detect keyed and clicked inputs at a response time of 50ms.

5.3 Assumptions and Justifications

1. The eye images used are not captured in real time but are taken from a database. These is due to the following reasons:
 - (a) It needs special cameras and scanners.
 - (b) The eye is sensitive to flashes and they might be damaged.
 - (c) Ethic and privacy concerns, because users may not be sure of their safety of their images.

5.4 Structured Use-case Diagram

The use case diagram shows how the relationship between the user and different use cases. The user can insert an image for the prediction but he has to set an output option first. Prediction model is then called to process the user input. The system administrator is entitled to manage the system. The system administrator can modify the processing model of the system later on. However, the user will be notified by the new modification(named "Display notification") through a message on the screen.

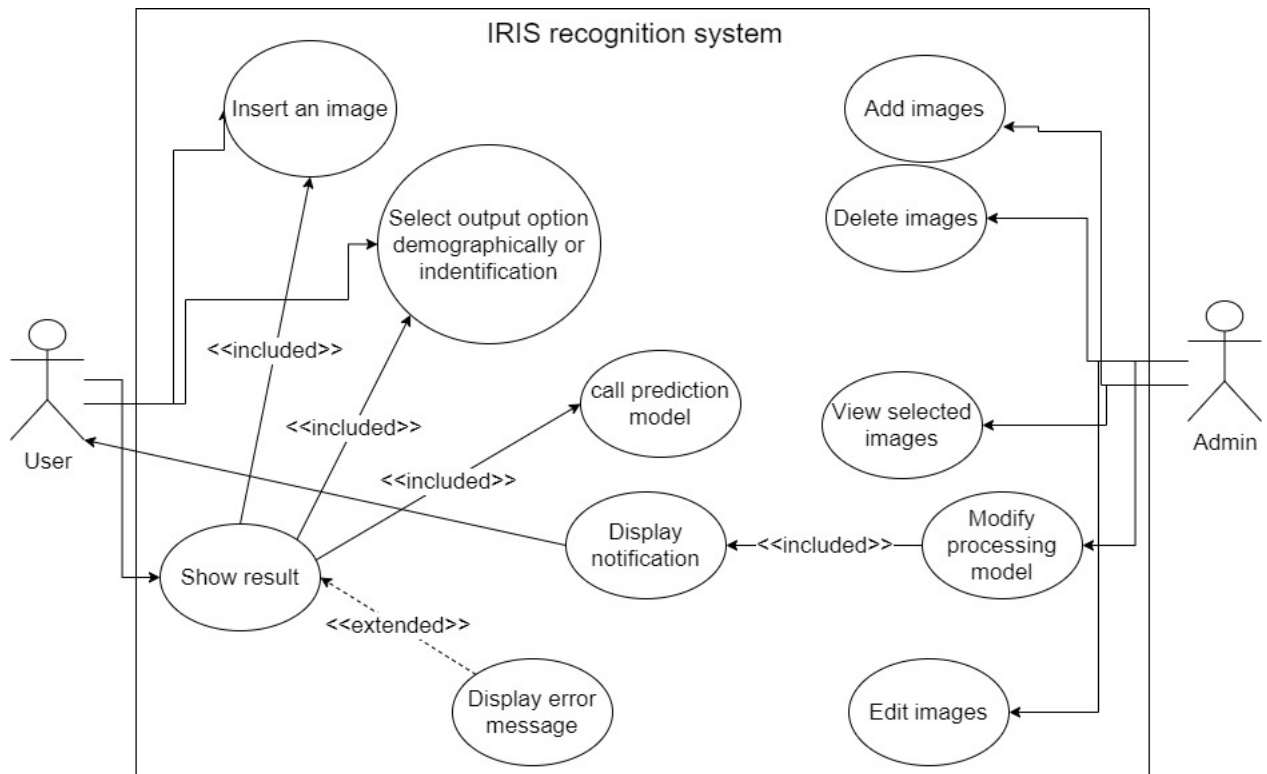


Figure 5.1: Structured Use Case Diagram for Iris Analyzer System

5.5 High Level Sequence Diagrams

The system shall require authentication from the system administrator and give access to the system and Database. The System administrator shall choose training and testing sets from the images in the database. The sets will be used as a model to train the user inputs using deep learning algorithms and test them to get the output data.

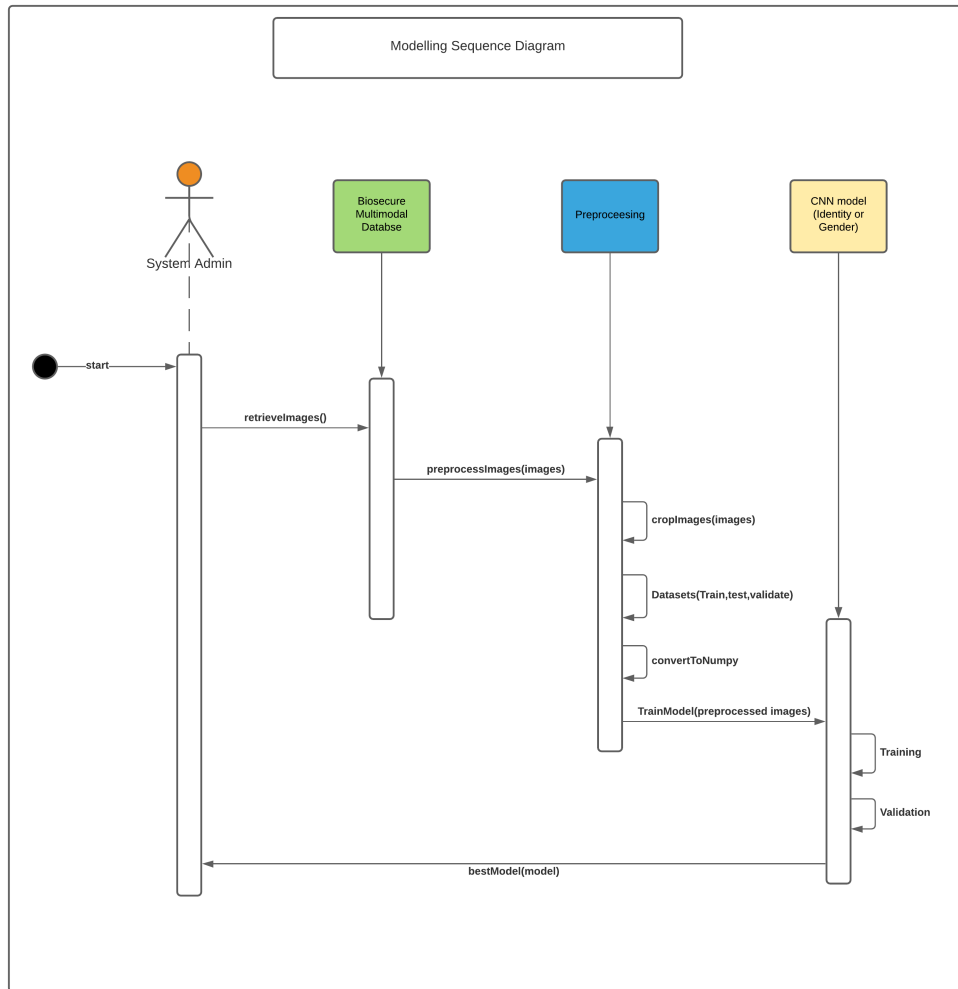


Figure 5.2: Modelling Image Use Case Sequence Diagram

The user shall choose an image and output option. The image shall be segmented first then passed through the Convolution Neural Network system for feature extraction. The features extracted shall be trained, validated and tested. The system through the test set shall give the output.

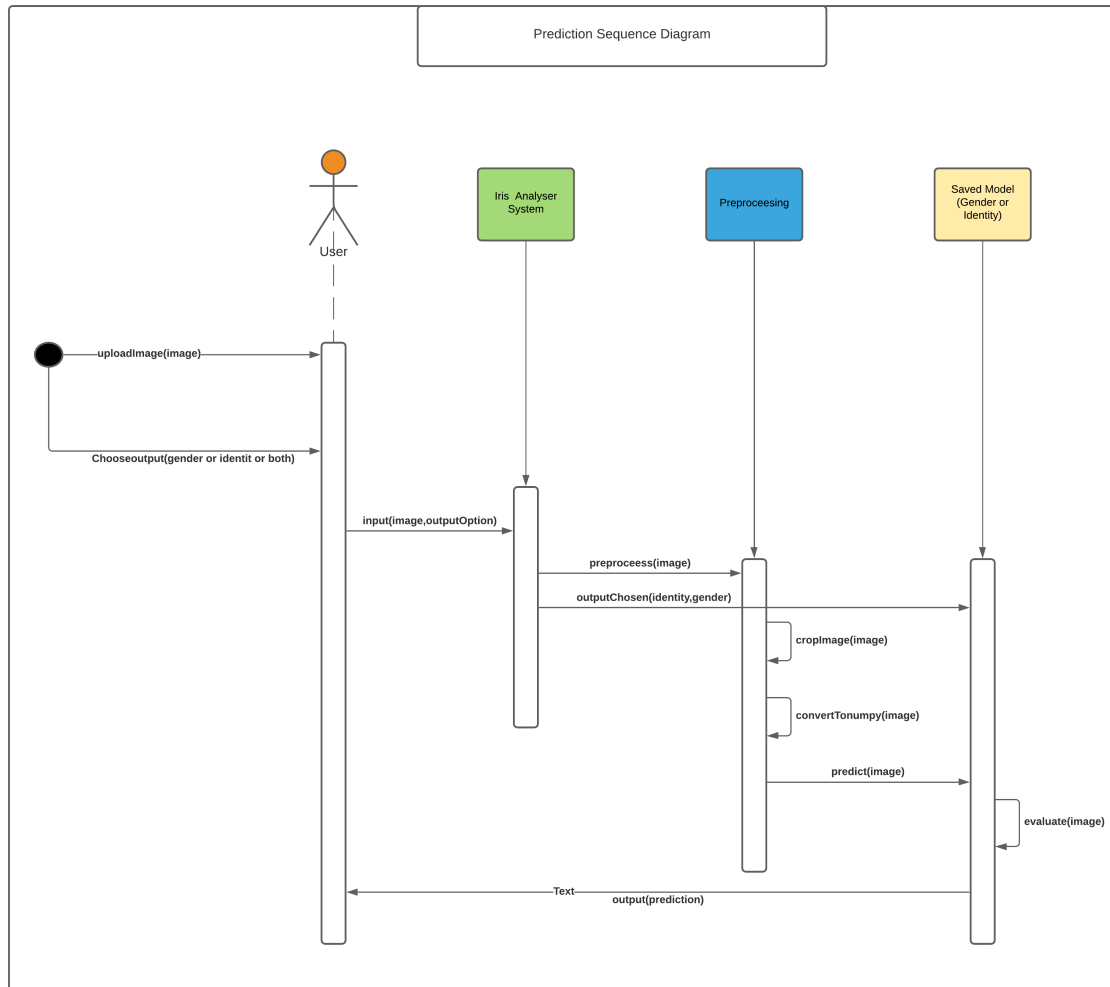


Figure 5.3: Prediction Image Use Case Sequence Diagram

5.6 Graphical User Interface

Below is an early iteration of how the graphical user interface of the Iris Analyser application will look like. The interface is in early stages of design and is subject to many changes during the next semester as the focus of this semester was on coding and modelling rather than the GUI itself.

Main Page GUI:

The user will be given an option either to open the system administrator section or User section.

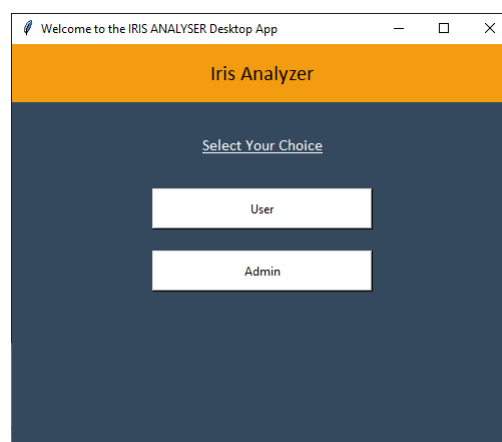


Figure 5.4: Graphical User Interface Main page view

User GUI:

The User Interface works as follows; After the appearance of the interface below, the user must first click on "Choose eye image" in order to choose an eye image of their choice from the Iris Analyzer eye image database. If the user presses any other option before doing so an appropriate error message will be prompted. The user shall choose either the identification or demographic output or both. After having chosen an eye image, output and confirming, the user must press the "Get Output" button and will be immediately presented with the identification and demographic result in the "Identification output" and "Demographic output" boxes respectively.

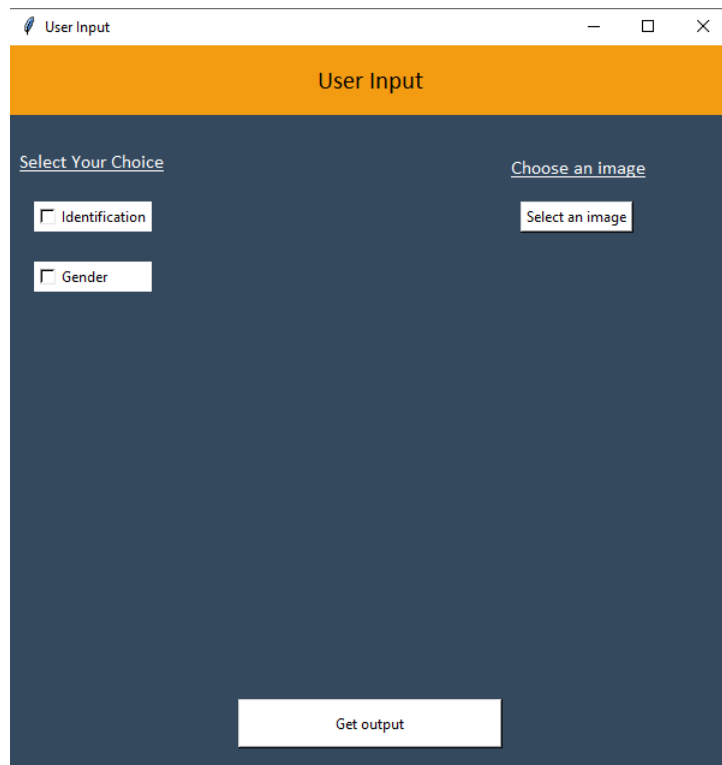


Figure 5.5: Graphical User Interface User view (Input)

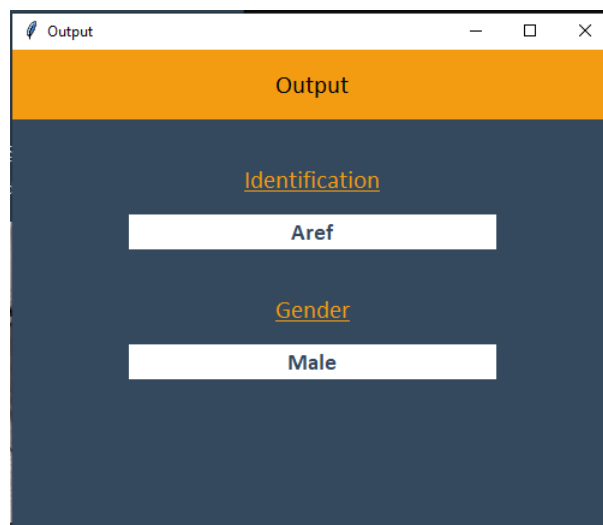


Figure 5.6: Graphical User Interface User view (Output)

System Administrator GUI:

The system administrator interface works as follows; After the appearance of the interface below, the system admin shall be prompted to enter a 4-digit pin. The system shall move on to the next interface if pin is correct and if its wrong display an error message. In the next interface the system admin can add and delete images from the database.

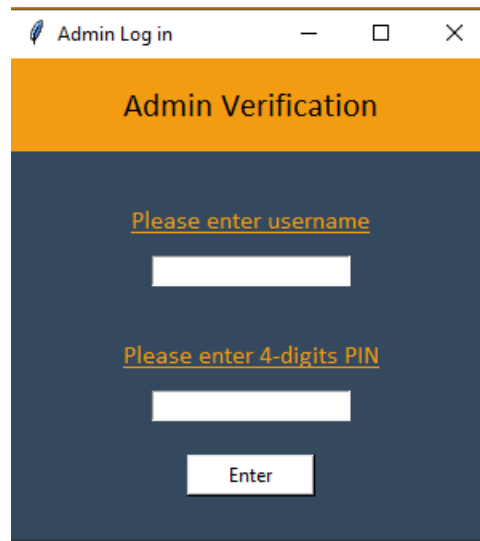


Figure 5.7: Graphical User Interface Authentication view

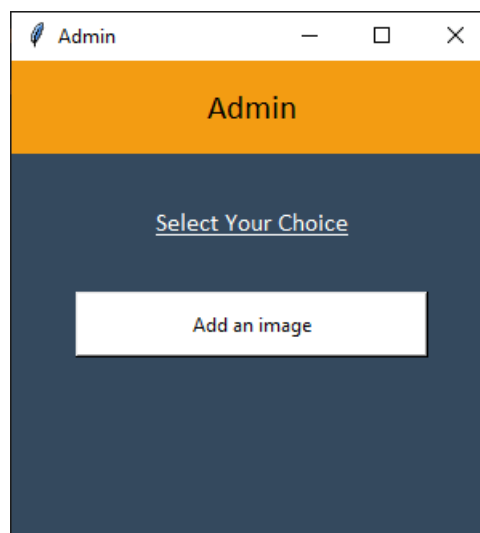


Figure 5.8: Graphical User Interface User database add image view

Chapter 6

System Architecture and Models

6.1 Architecture Model

This is a pipe and filter architecture diagram since the flow of the diagram is very straight forward. We have a data-set in which some parts of it get segmented, to build the image processing model. After that, the user can insert an image or set of images. These pictures will be first filtered and segmented by image processing model, then passed to the prediction model which is developed by deep learning for the final result.

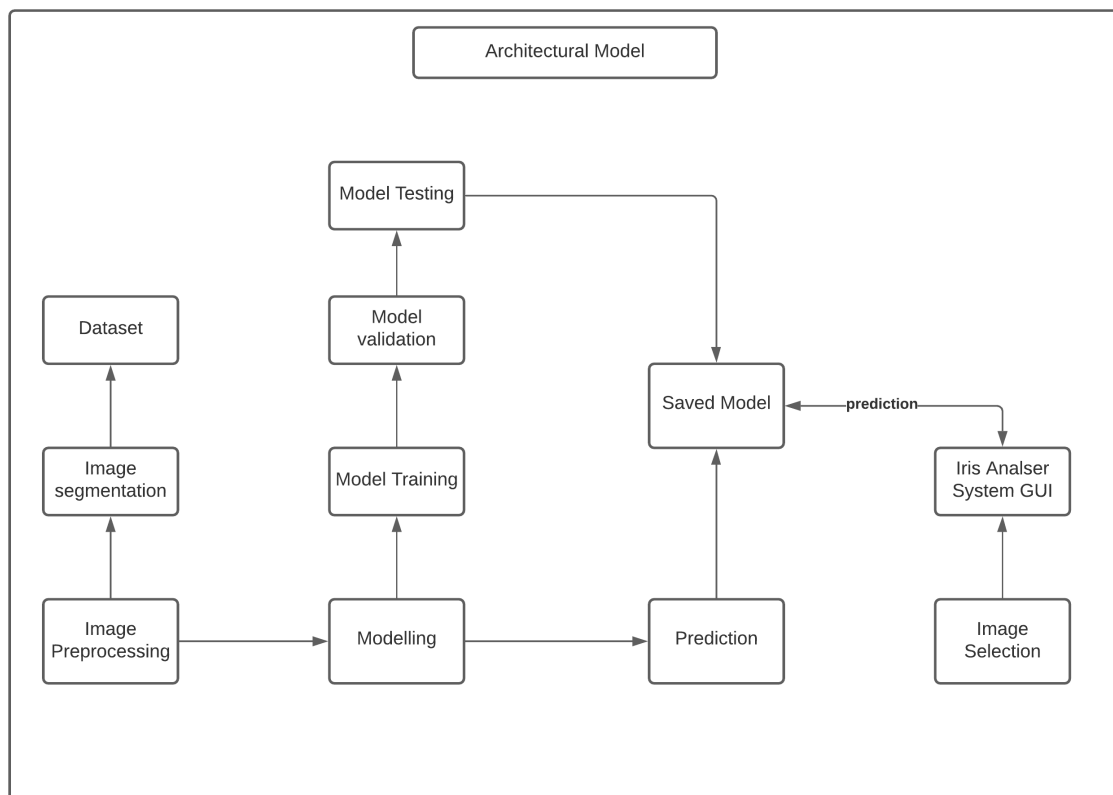


Figure 6.1: Architectural Model for the Iris Analyzer System

6.2 Process Model

Figure 6.2 shows the procedure to get the output from the inputted data. The user shall choose an image and the expected output wanted. The image shall be segmented and prediction done using deep learning methods to display the output.

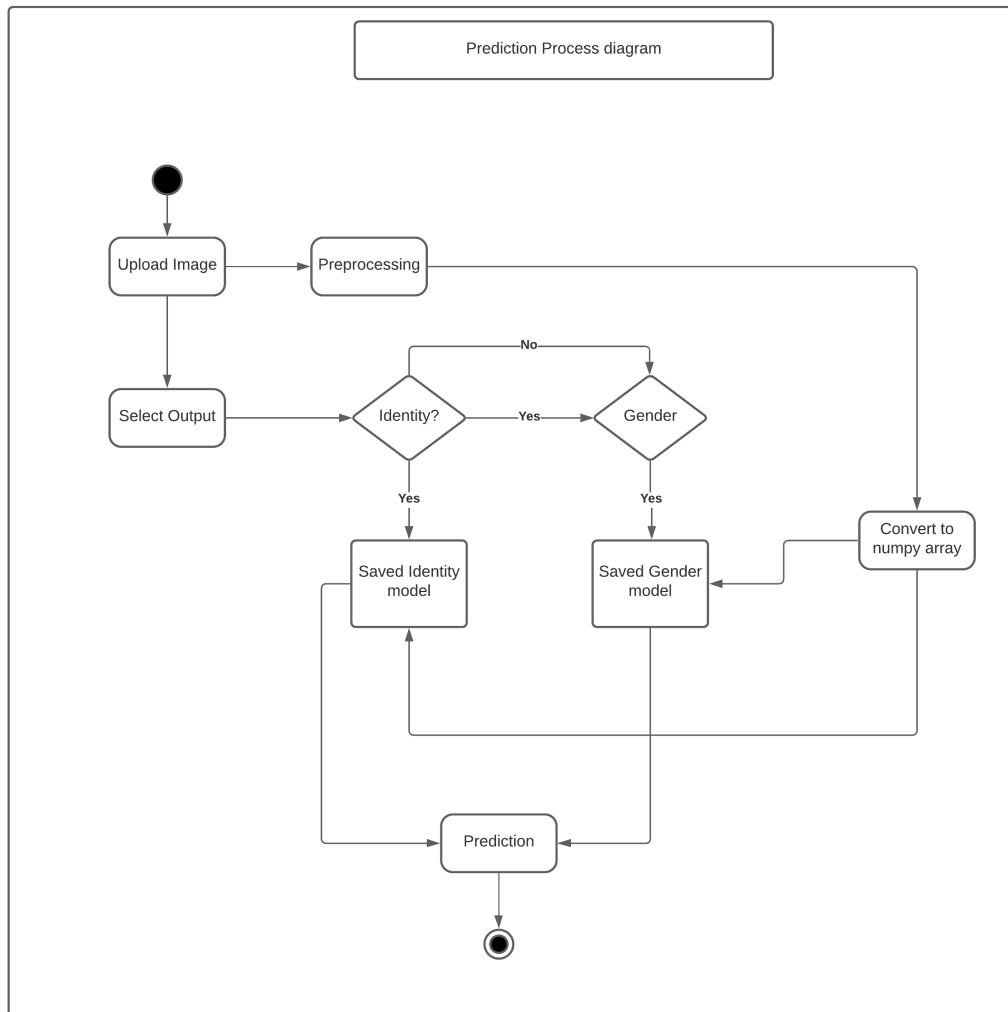


Figure 6.2: Process Model

An image is taken through the first step which is segmentation process. The segmented image is then passed through the Convolution Neural Network system which extract the features. The features are then trained using the training set and validated. This validation process helps give information that may assist us with adjusting our hyper-parameters. The trained data is then tested and if the accuracy threshold is above 85percent we accept the output otherwise it is trained again to improve the accuracy.

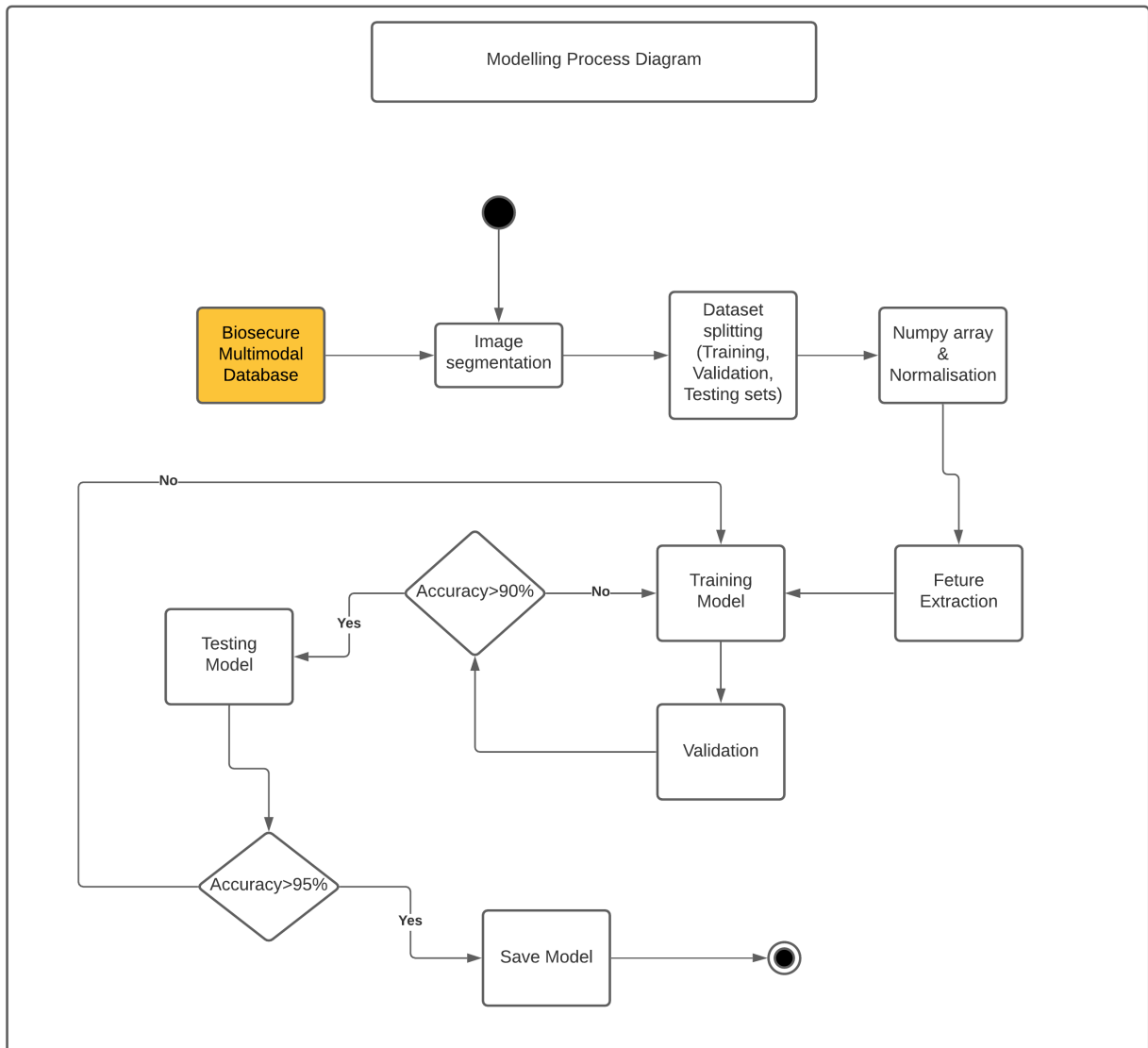


Figure 6.3: Modelling Process Model for the Iris Analyzer System

6.3 Data Flow Models

Figure 6.4 represents the data flow model which shows the abstract form of each process and how it is decomposed further into the deeper level.

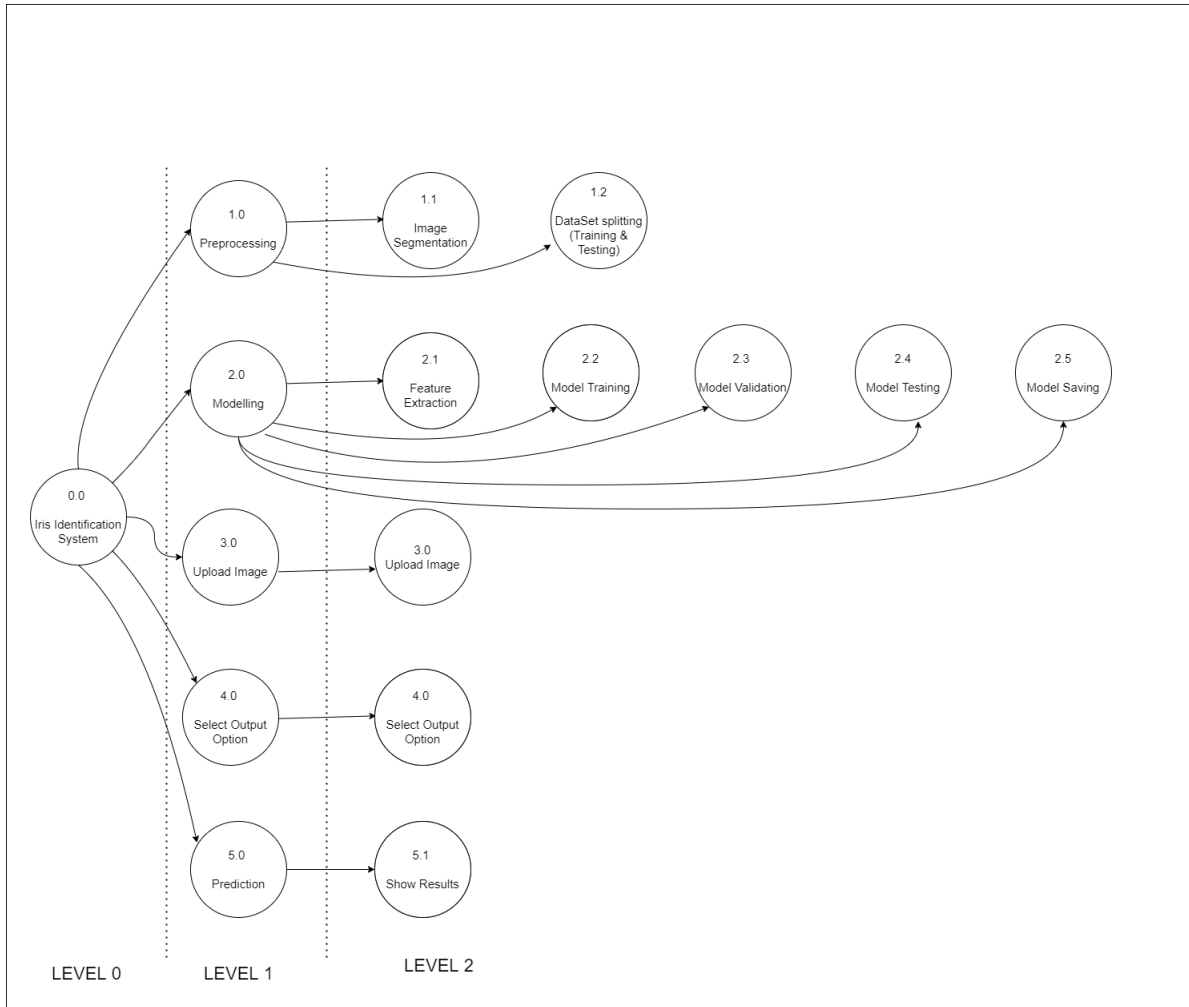


Figure 6.4: Data Flow Process Decomposition (Level 0,1 and 2)

Level 0 of the data flow diagram (context diagram) shows the flow of data between the main system and the external systems and actors which are the Users, System Administrator and Database in figure 6.5.

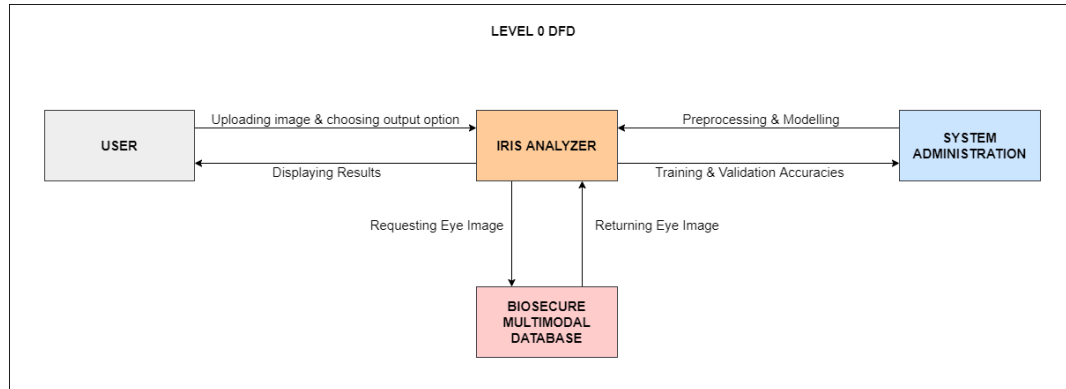


Figure 6.5: Level 0 Data Flow Diagram

The first level of the data flow model shows the base format of the processes without revealing the inner sub-levels. It also shows their relationship with the external system.

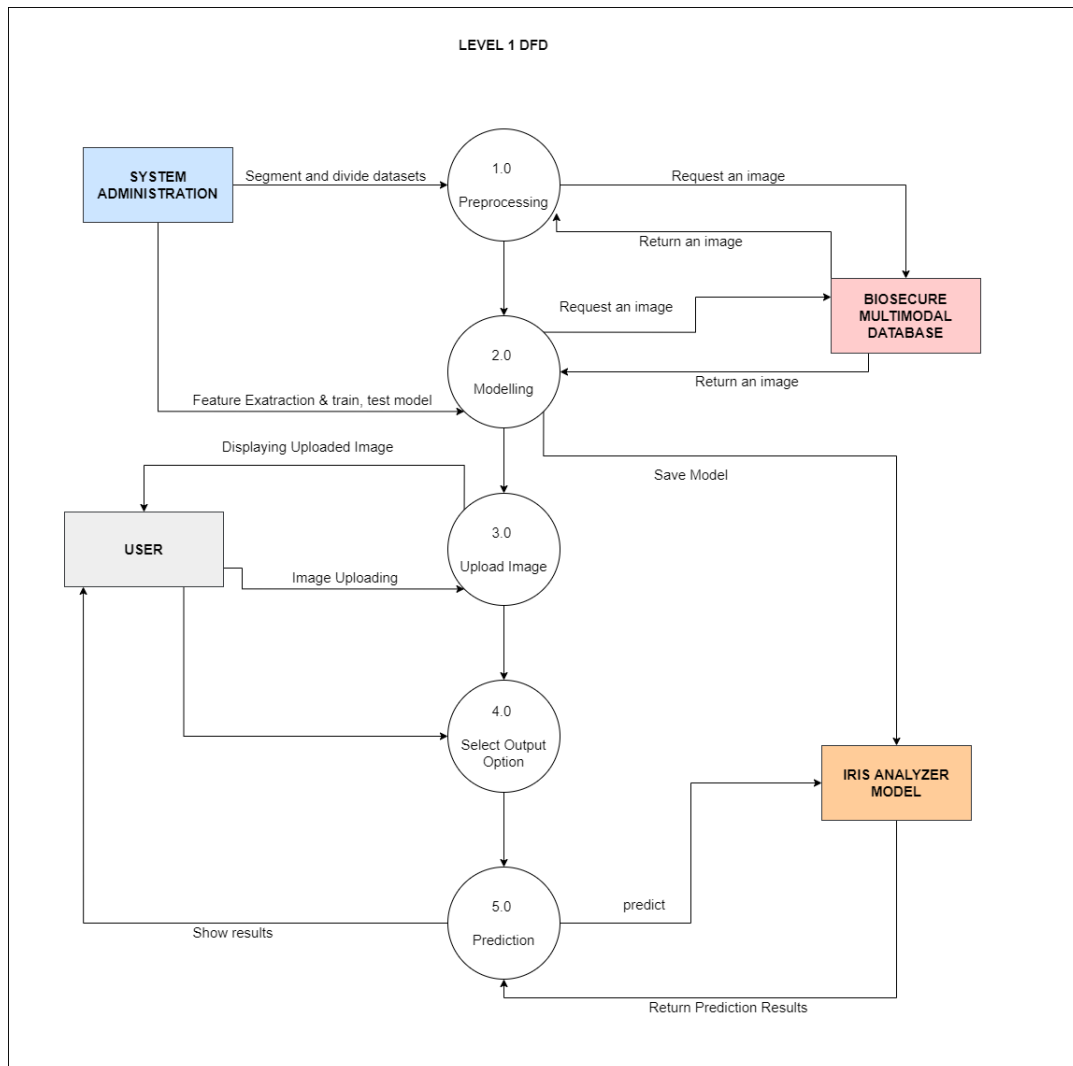


Figure 6.6: Level 1 Data Flow Diagram

Figure 6.7 is the inner level of the data flow diagram and it shows the inner workings of the processes in detail, step by step. As well as their relationship with the external system.

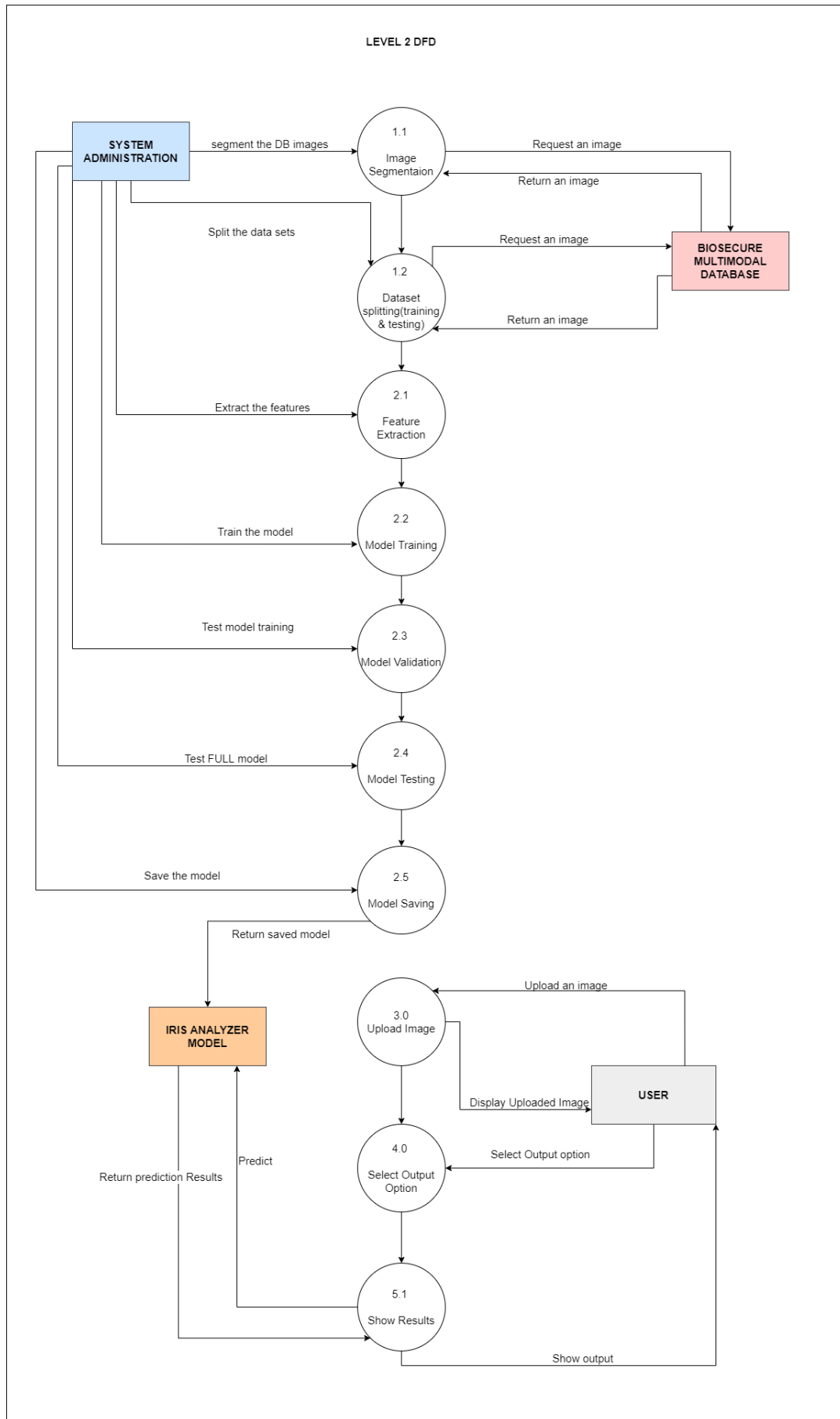


Figure 6.7: Level 2 Data Flow Diagram

6.4 Class Diagram

The class diagram is a representation of the system and its classes as well as its variables and methods. The relationship between each class or object is also shown.

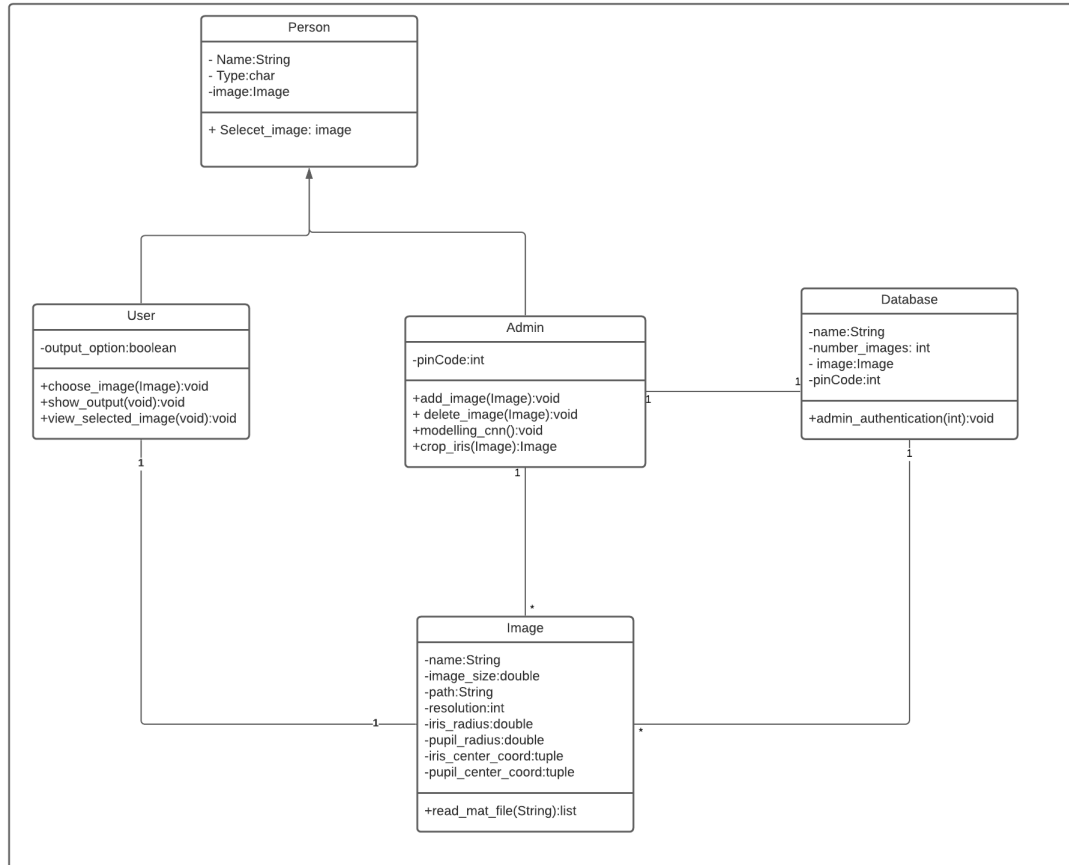


Figure 6.8: Class Diagram

6.5 List of Data Structures and Data Types

The Data Structures used are:

1. Dictionaries used to store users and their respective images.
2. Lists, e.g. training and testing lists, values of the keys in the dictionary.

The Data types used are:

1. Strings, as the images names and paths, and as keys in the dictionary.
2. Doubles, as the iris/pupil radius and center coordinates.

6.6 List of Special Algorithms

The algorithms used are:

1. Cropping algorithm to extract the iris segment from the image. The algorithm takes the iris and pupil radius dimension and uses them to extract only those parts.
2. Identification CNN model: The algorithm uses Keras library. It is made up of an input layer to take the input image, 3 Conv2D layers, 3 MaxPooling layers, 3 BatchNormalization layers, 1 fully connected layer and an output layer(Dense layer). The model is then trained using the fit function and tested using evaluation function.
3. Gender CNN model: The algorithm uses Keras library. It is made up of an input layer to take the input image, 2 Conv2D layers, 2 MaxPooling layers, 2 BatchNormalization layers, 1 fully connected layer and an output layer(Dense layer). The model is then trained using the fit function and tested using evaluation function.

Chapter 7

Project Management

The project management aspect of the Iris Analyzer for this half of the project was rather straightforward as most of the foundation was set in the previous half of the project and the team focused mainly on completing the tasks set previously such as designing the Graphical User Interface , making CNN model updates and more.

For the software estimation part we relied mostly on decisions made as a group when laying the core points and purposes of the project agreed upon with the advice of our supervisor. The milestones and main project tasks were the continuation of the previously completed milestones with emphasis on the GUI and integration of tasks in addition to tests done.

7.1 Software Estimations

- **Inputs :** User image insertion-L, log-in form-M, Add images form(admin)-H,prediction model H Prediction mode-L, Use mode(Admin/user)-L
- **Outputs:** Showing the selected image-H, predicted results-H
- **Inquiries:** User image insertion-L, admin image insertion-H, admin modeling process modification-H, admin image modification-H, Password authentication-M.
- **Logical internal files:** IRIS Images dataset-M, Coordinates file-M

Figure 7.1 shows the Estimation inputs based on their weight and complexity level.

	Simple	Average	Complex
Number of User Inputs	2	1	3
Number of User Outputs			2
Number of User Inquiries	1	1	3
Number of Files		1	1
Number of External Interfaces			

Figure 7.1: Estimation Input

7.1.1 Complexity Adjustment Table

		0	1	2	3	4	5
1.	Does the system require reliable backup and recovery?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
2.	Are data communications required?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3.	Are there distributed processing functions?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4.	Is performance critical?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
5.	Will the system run in an existing, heavily utilized operational environment?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
6.	Does the system require on-line data entry?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7.	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

		0	1	2	3	4	5
8.	Are the master files updated on-line?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
9.	Are the inputs, outputs, files, or inquiries complex?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
10.	Is the internal processing complex?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
11.	Is the code designed to be reusable?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
12.	Are conversion and installation included in the design?	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
13.	Is the system designed for multiple installations in different organizations?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
14.	Is the application designed to facilitate change and ease of use by the user?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 7.2: Estimation Complexity Adjustment Table

Result	
Project Function Points	89.24

7.1.2 Line of Code

To calculate the number of code lines we chose Python as the language to use and had a unit size of 29 which we multiplied by ATFP. This is the average unit size for both.

Python 100%

$$\text{Loc} = 29 * 89.24 = 2587.96$$

7.1.3 Estimate the Efforts

Constructive Cost Model COCOM0 Basic

Development mode: Organic.

$$a=2.4, b=1.05, c=0.38$$

$$\begin{aligned} \text{Number of thousand delivered source instructions} = \\ (KDSI) = ATFP * Python_{Language_unit_size} / 1000 = 2587.96 / 1000 = 2.58796 \end{aligned}$$

$$\begin{aligned} \text{Effort in staff months} = \text{effort in man-months} = \\ MM = a * KDSI * b = 2.4 * ((2.58796)^{1.05}) = 6.51 \end{aligned}$$

$$\begin{aligned} \text{The development time} = \\ TDEV = 2.5 * MM^c = 2.5 * (6.51^{0.38}) = 5.09 = 6months \end{aligned}$$

Jones's first -order effort estimation methods

We are building a shrink-wrap kind of software in an average organization.

$$\text{Rough schedule estimation} = ATFP^{exp} = 90^{0.042} = 6.61 = 7months.$$

Schedule Rule of Thumb

Efficient Schedule (Shrink-Wrap)

Schedule(Months)= 5.9 6 months

Effort(Man-month) = 8

Table 8-9. Efficient Schedules

	Systems Products		Business Products		Shrink-Wrap Products	
System Size (lines of code)	Schedule (months)	Effort (man- months)	Schedule (months)	Effort (man- months)	Schedule (months)	Effort (man- months)
10,000	8	24	4.9	5	5.9	8
15,000	10	38	5.8	8	7	12
20,000	11	54	7	11	8	18
25,000	12	70	7	14	9	23
30,000	13	97	8	20	9	32
35,000	14	120	8	24	10	39
40,000	15	140	9	30	10	49
45,000	16	170	9	34	11	57
50,000	16	190	10	40	11	67
60,000	18	240	10	49	12	83
70,000	19	290	11	61	13	100
80,000	20	345	12	71	14	120
90,000	21	400	12	82	15	140
100,000	22	450	13	93	15	160
120,000	23	560	14	115	16	195
140,000	25	670	15	140	17	235
160,000	26	709	15	160	18	280
180,000	28	910	16	190	19	320
200,000	29	1,300	17	210	20	360
250,000	32	1,300	19	280	22	470
300,000	34	1,650	20	345	24	590

Figure 7.3: Estimation Efficient Schedules

7.2 Milestones, Main Project Tasks and Task Allocation

Having reached the end of the semester as well as the final project deadline, the team had completed the tasks set for the previous semester as well as the planned tasks and milestones set throughout the current semester and phase of the project. The project was worked on in order to make it more extendable as well as making the model more accurate and dynamic in order to be used by different organization which was achieved successfully.

Tables 7.1 7.2 below show all the tasks and milestones set and completed at the end of the previous academic semester (January 2021) along with the duration of each task and the team members responsible for the task respectively.

Table 7.1: Tasks for the previous semester

Task #	Task	Duration (weeks)	Dependencies	Responsible By
T1	Research (Image Proc Techniques) + Literature Review	2		All
T2	Research (Deep learning Basics) + Literature Review	2		All
T3	Write a Project Report (1)	2	T1,T2 (M1)	All
T4	Research (Deep Learning Algorithms)	2		All
T5	Research (Image Segmentation Techniques)	2		Aref, Sameh
T6	Research (Convolutional Neural Networks)	2		Baraa, Evans
T7	Segmentation of Iris region from Eye Images	1	T5 (M2)	Aref, Sameh
T8	Crop and Prepare Segmented Images	1	T7	Baraa, Evans
T9	Divide Images into Training and Test set	1	T8	Baraa, Evans
T10	Write a Project Report (2)	2	T4,T5,T6 (M2)	All
T11	Convolutional Neural Network (Code)	1	T6	Baraa, Evans
T12	Train Deep Learning Model	1	T7,T8,T11(M3)	All
T13	Gather Outcome (Accuracy) Information	1	T12	Baraa, Evans
T14	Analyze Deep Learning Model Results	1	T13	Aref , Sameh
T15	Compare Deep Learning Outcome vs Traditional Methods	1	T13 (M5)	Aref, Sameh
T16	Write a Project Report (3/Final)	2		All

Table 7.2: Milestones for the previous semester

Milestone #	Milestone
M1	Project Fundamentals Understood
M2	Sufficient Fundamental Research for Deep Learning Model
M3	Images are Segmented and Ready (Finalized Format)
M4	Deep Learning Model Complete
M5	Results (Output) Analysis Ready

As for Table 7.3, the tasks and milestones set and completed this semester (July 2021) are shown along with the duration, dependencies and responsible members.

Table 7.3: Tasks for the semester

Test Task #	Test Task	Duration (Weeks)	Facilities/Tools (For Testing Tasks)	Responsible By
T1	Design GUI User Page	2	-	Aref, Elbaraa
T2	Design GUI Admin Page	2	-	Aref, Elbaraa
T3	Improve CNN Identity Model Accuracy	2	-	Evans, Sameh
T4	Improve CNN Gender Model Accuracy	2	-	Evans, Sameh
T5	Add CNN Model Retrain Option	1	-	Evans, Aref
T6	Integrate GUI with CNN Model	2	-	Elbaraa, Sameh
T7	Preprocessing Unit Test	1	Manual Python Functions	Evans , Sameh
T8	CNN Model Unit Test	1	Manual Python Functions	Evans, Sameh
T9	GUI User Unit Test	1	Manual Functions	Aref, Elbaraa
T10	GUI Admin Unit Test	2	Unittest Module	Aref, Elbaraa
T11	Model and GUI Integration Test	2	Manual Functions	Team
T12	Finalizing Touches and Changes	1	-	Team

Table 7.4 shows the milestones that were worked towards during the current academic semester. The milestones represent any significant goal or activity in the project development cycle.

Table 7.4: Milestones for the semester

Milestone #	Milestone
M1	GUI Components completed and functional
M2	CNN Models at desired performance level
M3	CNN Model tested components are functional
M4	GUI Unit tested components are functional
M5	Integration of all parts successful and functional
M6	Project Finalized and works as intended

7.3 Gantt Chart

Figure 9.2 shows the visual representation of tasks where scheduled to be completed through the active Spring semester of 2021. The tasks and milestones from tables 7.3 and 7.4 are shown in the Gantt chart format with approximate dates and duration it was worked on. Since we have reached the end of the semester that tasks are at full completion as intended and have been successfully fulfilled. The tasks are divided by color based on the group of task it falls within which we have neatly divided into 4 different groups (Purple = GUI related , Light Purple = CNN Improvements, Green = Integration, Turquoise = Testing , Yellow = Finalization).

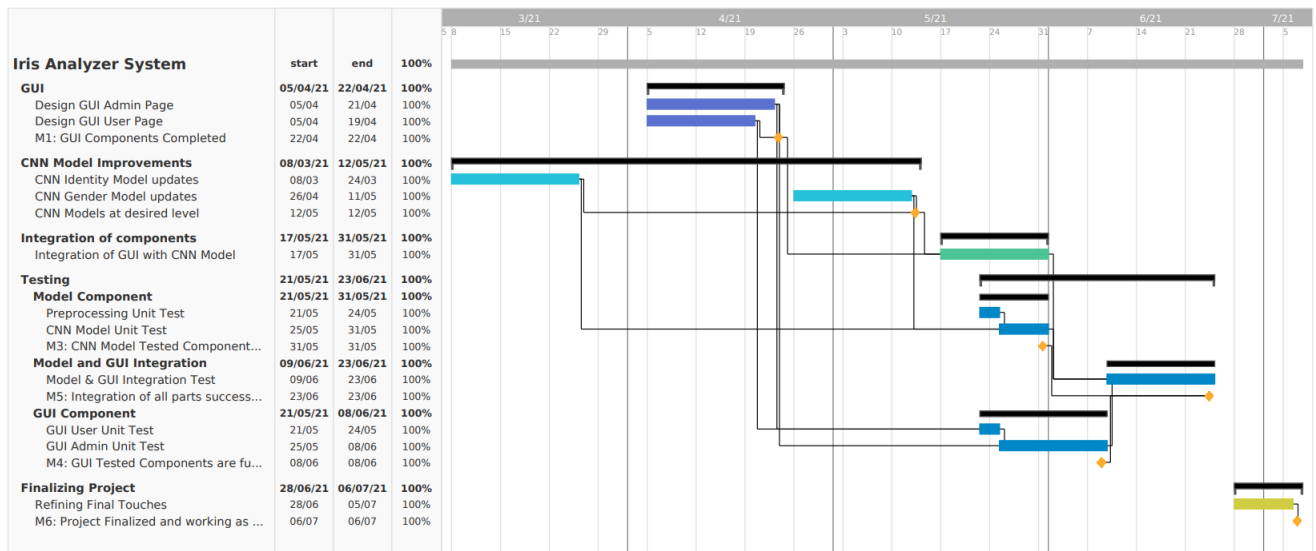


Figure 7.4: Gantt Chart

Chapter 8

Software Implementation

1. **Programming Languages Used:** Python which is an interpreted, object-oriented, high-level programming language.

2. **Libraries and frameworks:** Used to open the file that contains original images.

(a) **Preprocessing:**

- i. **scipy:** It's a library used for scientific and technical computing. We used it to load the matlab files that contains the details of the IRIS and PUPIL dimensions.
- ii. **os:** which is a module in Python provides functions for interacting with the operating system. We used it to open the files that contains original images.
- iii. **cv2:** Which is a library of Python that is designed to solve computer vision problems. We used it to load the images and it's also used in doing the cropping of the images.

(b) **GUI:**

- i. **tkinter:** Which is a is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit. We used it to create a simple GUI, 'fielddialog' is also used to show the select image box to the user.
- ii. **PIL:** Which is a library for Python that adds support for opening and saving many different image file formats. We used it to read the image uploaded by the user.
- iii. **Keras:** Which is a library that provides artificial neural networks. We used it to load the saved model.

(c) **CNN:**

- i. **Keras:** Used to create the CNN models and the layers.
- ii. **matplotlib:** Which is a library that provides provides an object-oriented API for embedding plots into applications. We used it to plot graphs for testing and training
- iii. **numpy:** Which is a library that adds support for large, multi-dimensional arrays and matrices. We used it to convert images to numpy arrays and reshape the images.
- iv. **os:** Used to open the images folder and read them.

v. **PIL**: Used to load and open the images.

Chapter 9

Software Testing

9.1 Testing Strategy

Testing strategy is a plan for defining the overall approach for testing a specific project, where we begin with "testing-in-the-small" and move on to "testing-in-the-large". The Iris Analyzer is a conventional software, where testing the module is our initial focus.

Our project is separated into front-end and back-end. The front-end is a simple GUI that handles user's interactions. While the back-end is where the data processing takes place.

For the back-end of our project, we have two main parts, initial/pre processing and modelling. Initial processing is the preparation of the data to make it ready for the modelling part. We will first do unit testing for the pre processing part as well as modelling. After doing the unit testing, we will proceed with integration testing where we will test the interactions of these parts together. To do the back-end testing.

For the front-end of the project, we will do a series of tests to make sure that the GUI is user friendly and interacting efficiently with the database. We will test the components in our GUI such as buttons and check if they are working correctly when clicking for example. We are planning to test the GUI manually.

9.2 Unit Testing

As the Iris Analyzer is fundamentally a research-based project, the team decided to move forward with a non-tabular format for the unit testing section ahead as there are significantly less tests conducted compared to a project that is not of this type.

9.2.1 Preprocessing Unit

The tests below cover the range of items presented in the test plan relating to the preprocessing section of the Model side of the project. The preprocessing takes care of the data-set and its corresponding format. The preprocessing tests are done in 3 areas with their respective testing criteria covered below.

1. **Unit name:** Reading Images from Database Test
2. **Participants:** Evans, Sameh

3. **Pass/Fail criteria:**

- (a) Pass - All images read
- (b) Fail - Some images not read

4. **Methodology:** When reading images a counter is used to check number of images read and also a print function of images that are not read is displayed on screen. The count value is compared to total number of images.

5. **Test case:** All images are read from database

6. **Actual Results:** All images read from database. Counter number is equal to number of images

7. **Results:** Passed

1. **Unit name:** Cropping/Segmentation Test

2. **Participants:** Evans, Sameh

3. **Pass/Fail criteria:**

- (a) Pass - All images segmented
- (b) Fail - Some images not segmented

4. **Methodology:** A counter to check the number of images segmented successfully and compares to the total number of images.

5. **Test case:** All images from database are segmented.

6. **Actual Results:** All images in the database were segmented. Counter number is equal to number of total images.

7. **Results:** Passed

1. **Unit name:** Writing Cropped Images to Folder Test

2. **Participants:** Evans, Sameh

3. **Pass/Fail criteria:**

- (a) Pass- All images are stored in new cropped folder
- (b) Fail- Some images not saved in cropped folder

4. **Methodology:** Use a counter to keep record of number of stored segmented images in the cropped folder and compare it to the total number of segmented images.

5. **Test case:** All cropped images stored in a cropped folder.
6. **Actual Results:** All cropped images were stored in a cropped folder. Counter number was equal to the number of segmented images.
7. **Results:** Passed

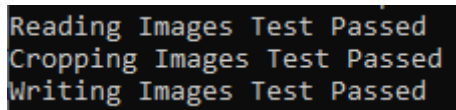
```
import os

def readingImagesTest(number):

    if number==1600:
        print("Reading Images Test Passed")
    else:
        print("Reading Images Test Failed")
    -----#
def croppingImagesTest(number):
    if number==1600:
        print("Cropping Images Test Passed")
    else:
        print("Cropping Images Test Failed")
    -----#
def writingImagesTest(number,folder):
    count=0
    for filename in os.listdir(folder):
        count+=1

    if number==count:
        print("Writing Images Test Passed")
    else:
        print("Writing Images Test Failed")
    -----#
```

Figure 9.1: Preprocessing Test Function



```
Reading Images Test Passed
Cropping Images Test Passed
Writing Images Test Passed
```

Figure 9.2: Testing passed

9.2.2 CNN Model Unit

The tests below cover the core of the Iris Analyzer system which is the CNN model and ensures that the functionality of the model is retained by checking the output accuracy of the model and ensuring it meets the minimum passing standard which was chosen mathematically based on multiple research papers provided in the references.

1. **Unit name:** Validation Testing (Identification)
2. **Participants:** Evans, Sameh
3. **Pass/Fail criteria:**
 - (a) Pass - An accuracy of above 95% is achieved
 - (b) Fail - An accuracy below 95%
4. **Methodology:** Validation is done on the training part using built-in validation methods which return an accuracy result and we determine the training process using that as mentioned in the criteria above.
5. **Test case:** Validation accuracy is above 80%
6. **Actual Results:** Validation accuracy was 80%
7. **Results:** Passed

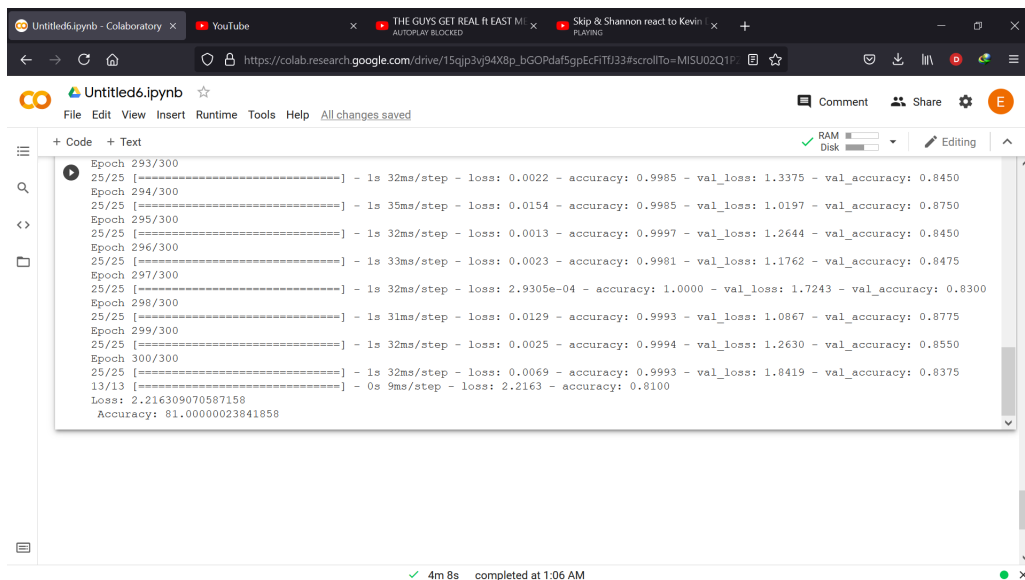
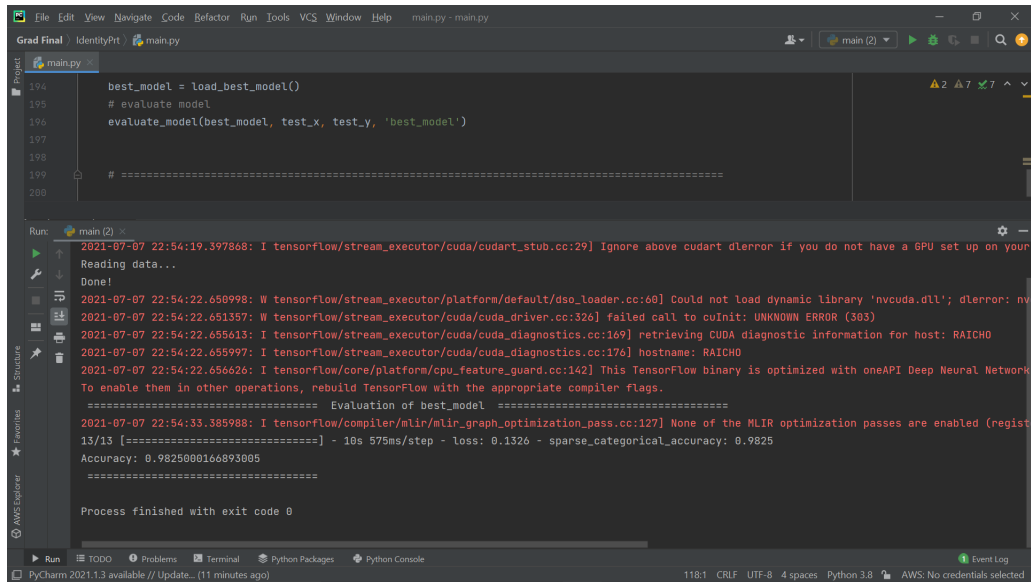


Figure 9.3: Validation Accuracy

1. **Unit name:** Predicted Results Testing (Identification)
2. **Participants:** Evans, Sameh
3. **Pass/Fail criteria:**
 - (a) Pass - An accuracy of above 85% is achieved
 - (b) Fail - An accuracy below 85%
4. **Methodology:** This is done using built-in testing methods that takes the testing set and gets their classification and returns a result from that. We check the accuracy result to determine the testing process as mentioned above in the criteria part.

5. **Test case:** Testing accuracy should be above 85%
6. **Actual Results:** Testing accuracy was 87%
7. **Results:** Passed



The screenshot shows the PyCharm IDE interface. The top pane displays a Python script named `main.py` with the following code:

```
194 best_model = load_best_model()
195 # evaluate model
196 evaluate_model(best_model, test_x, test_y, 'best_model')
197
198
199 # =====
200
```

The bottom pane shows the Run console output for the `main (2)` process. The output includes several TensorFlow warnings and a final accuracy report:

```
2021-07-07 22:54:19.397868: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your
Reading data...
Done!
2021-07-07 22:54:22.650998: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda
2021-07-07 22:54:22.651357: W tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed call to cuInit: UNKNOWN ERROR (303)
2021-07-07 22:54:22.655613: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: RAICHO
2021-07-07 22:54:22.655997: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: RAICHO
2021-07-07 22:54:22.656626: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
===== Evaluation of best_model =====
2021-07-07 22:54:33.385988: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:127] None of the MLIR optimization passes are enabled (regist
13/13 [=====] - 10s 575ms/step - loss: 0.1326 - sparse_categorical_accuracy: 0.9825
Accuracy: 0.9825000166893005
=====
Process finished with exit code 0
```

The status bar at the bottom indicates the environment is Python 3.8 on an AWS SageMaker instance.

Figure 9.4: Identity Accuracy

1. **Unit name:** Validation Testing (Gender)
2. **Participants:** Evans, Sameh
3. **Pass/Fail criteria:**
 - (a) Pass - An accuracy of above 95% is achieved
 - (b) Fail - An accuracy below 95%
4. **Methodology:** Validation is done on the training part using built-in validation methods which return an accuracy result and we determine the training process using that as mentioned in the criteria above.
5. **Test case:** Validation accuracy is above 95%
6. **Actual Results:** Validation accuracy was 95%
7. **Results:** Passed

1. **Unit name:** Predicted Results Testing (Gender)
2. **Participants:** Evans, Sameh
3. **Pass/Fail criteria:**
 - (a) Pass - An accuracy of above 80% is achieved
 - (b) Fail - An accuracy below 80%
4. **Methodology:** This is done using built-in testing methods that takes the testing set and gets their classification and returns a result from that. We check the accuracy result to determine the testing process as mentioned above in the criteria part.
5. **Test case:** Testing accuracy should be above 80%
6. **Actual Results:** Testing accuracy was 82%
7. **Results:** Passed

The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script with the following code:

```
194 best_model = load_best_model()
195 # evaluate model
196 evaluate_model(best_model, test_x, test_y, 'best_model')
197
198 # =====
199
200
201
202 if __name__ == '__main__':
203     main()
204
```

The Run tool window at the bottom shows the execution output for 'main (2)'. The output includes several log messages from TensorFlow and a final summary of the evaluation results:

```
2021-07-07 22:54:22.655997: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: RAICHO
2021-07-07 22:54:22.656626: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
===== Evaluation of best_model =====
2021-07-07 22:54:33.385988: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:127] None of the MLIR optimization passes are enabled (regist
13/13 [=====] - 10s 575ms/step - loss: 0.1326 - sparse_categorical_accuracy: 0.9825
Accuracy: 0.9825000166893005
=====
Process finished with exit code 0
```

The status bar at the bottom indicates the PyCharm version (2021.1.3), the current file encoding (UTF-8), and the Python interpreter (Python 3.8).

Figure 9.5: Identity Accuracy

9.2.3 Admin Unit (GUI)

The GUI testing is divided into 2 sections with the first being related to the Administrator's control and access within the Iris Analyzer system. For the GUI the Unittest Python library was made use of when conducting the Admin tests.

1. **Unit name:** Login Test (Admin)
2. **Participants:** Elbaraa, Aref
3. **Pass/Fail criteria:**
 - (a) Pass - Login is successful when correct username and password are applied for all admins.
Login is unsuccessful if wrong username or password is applied.
 - (b) Fail - Login is successful when wrong username or password is applied.
Login fails when correct username and is applied.
4. **Methodology:** we create a function that has access to our login details file. We then try different login details wrong and right and check for errors mentioned in fail part above.
5. **Test case:** All correct login details work and wrong ones fail
6. **Actual Results:** All correct logins work. Wrong logins do not work.
7. **Results:** Passed

1. **Unit name:** Update Images Test (Admin)
2. **Participants:** Elbaraa, Aref
3. **Pass/Fail criteria:**
 - (a) Pass - Admin can access the database with images.
Admin can add and delete images from the database.
 - (b) Fail -Admin cannot access the image database.
Admin cannot add or remove images from the database.
4. **Methodology:** Checks are done by adding images to the database and making sure it works smoothly. We run tests like adding images and checking if they are added to the database.
Same process is done for deletion.
5. **Test case:** Admin can access all images, add and delete images from database.
6. **Actual Results:** Admin accesses all images in database and can add and delete any image.

7. **Results:** Passed

1. **Unit name:** Update Model Test (Admin)
2. **Participants:** Elbaraa, Aref
3. **Pass/Fail criteria:**
 - (a) Pass - Admin can access details of the CNN models. Admin can also add new CNN models in the future.
 - (b) Fail - Admin does not have access to the CNN models.
4. **Methodology:** Checks are done in the admin GUI, where we make sure that admin can interact with the CNN models smoothly. We run checks like opening up CNN models from admin GUI.
5. **Test case:** Admin can access the CNN models.
6. **Actual Results:** Admin accesses the CNN models.
7. **Results:** Passed

9.2.4 User Unit (GUI)

The GUI testing is divided into 2 sections with the second being related to the User's access and usage of the Iris Analyzer system through the user interface. For this part, the module went through manual testing which verified the user's ability to successfully choose an image as well as performance response tests as we would like the system to display the output in a fixed amount of time.

1. **Unit name:** Choosing Images Test (User)
2. **Participants:** Elbaraa, Aref
3. **Pass/Fail criteria:**
 - (a) Pass - User can choose an image from a folder of images.
Image chosen is shown on screen to user.
 - (b) Fail - User cannot choose any image from the folder of images.
Chosen image is not displayed on screen.
After choosing image, wrong image is displayed to user.
4. **Methodology:** We make a function that takes an image from a folder and returns the image name as output. If its correct its a pass. We check also that user can choose all images and check if they are displayed on screen.
5. **Test case:** User can choose an image from the displayed ones.
6. **Actual Results:** User can choose an image from the presented folder. Image chosen is displayed on screen.
7. **Results:** Passed

1. **Unit name:** Output Response Test (User)
2. **Participants:** Elbaraa, Aref
3. **Pass/Fail criteria:**
 - (a) Pass - Output for the given input image is correct for both identification and gender.
Identification output works correctly: returns name of user (the image belongs to).
Gender output works correctly, returns either male or female.
 - (b) Fail - Either identification or gender output is incorrect for the input image.
 - (c) Format of output is wrong as explained above.
4. **Methodology:** Select input image and check the output given for identification and gender.

We use a function to make sure the output format/results displayed for both the gender and identification is correct as explained above.

5. **Test case:** User gets correct output for the chosen image for both identification and gender.
6. **Actual Results:** User gets correct output for the chosen images for both identification and gender.
7. **Results:** Passed

9.3 System and Integration Testing

As the integration of both the CNN Model as well as the GUI is crucial for the Iris Analyzer to work as a whole, the team has implemented tests that cover the fluidity of the of the modules working together which falls under the category of the preprocessing of the data-set together with the Iris Analyzer CNN model , the database of images with the GUI and finally the combination of all parts together.

Integration testing was done in the following parts:

1. Preprocessing and Model Integration

- (a) **Participants:** Evans, Sameh
- (b) **Pass/Fail criteria:**
 - i. Pass - Preprocessing module and CNN model module work together. All images are cropped then are trained and tested with the CNN models without any error.
 - ii. Fail - Both modules (preprocessing and CNN models) do not work together. Errors with getting preprocessed images working with models.
- (c) **Methodology:** Code is run from preprocessing to CNN model testing to make sure they work perfectly together. Check for errors that occur between the two modules.

During the coding process, we used GitHub branches and testing was done to check if the models integrated with the preprocessing part.
- (d) **Test case:** All aspects of preprocessing integrate together with CNN model module.
- (e) **Actual Results:** Correct output attained after running both modules together. Models accuracy from the segmented images retrieved from preprocessing part is correct.
- (f) **Results:** Passed

2. Database and GUI

- (a) **Participants:** Elbaraa, Aref
- (b) **Pass/Fail criteria:**
 - i. Pass - Images from the database can be accessed in the user interface, both for User and Admin parts.

Admin can update parts of the database, add and delete images in the database.
 - ii. Fail - Images in the database cannot be accessed by the GUI neither, the user or admin parts.

Admin cannot update images in the database.
- (c) **Methodology:** Testing is done during User and Admin GUI testing parts above.

- (d) **Test case:** All aspects of database integrate together with GUI for admin and user.
- (e) **Actual Results:** Admin can access the database and update the images, while users can see the displayed input images they can choose to get output.
- (f) **Results:** Passed

3. GUI and CNN models

- (a) **Participants:** Elbaraa, Aref
- (b) **Pass/Fail criteria:**
 - i. Pass - User can get outputs from the image he selects (model is working)
Admin can access models from the GUI.
 - ii. Fail - No output or wrong output after program runs.
Admin cannot access models from the GUI.
- (c) **Methodology:** Test done in user and admin GUI testing parts above.
- (d) **Test case:** GUI For both user and admin integrate well with the CNN models module.
- (e) **Actual Results:** The admin can access the CNN models while the users get correct output from the system when they run the program.
- (f) **Results:** Passed

4. Individual modules that have multiple units - For parts like preprocessing and GUI that are made up of multiple units, their testing was done in their respective parts above.

Integration testing on them was done after all parts were joined together.

- (a) **Pass/Fail criteria:**
 - i. Pass - All interfaces/units making up the large modules integrate together perfectly.
 - ii. Fail - some of the interfaces or units that make up the modules do not work together.
- (b) **Test case:** All modules interfaces after integration work smoothly.
- (c) **Actual Results:** As seen in their respective parts in the sections above, all interfaces/units that make up a module work together perfectly.
- (d) **Results:** Passed

9.4 Performance and System Testing

The performance and stress testing is done to make sure that the system/ program runs optimally at all times. The performance testing shall be done in both the GUI and CNN model parts to make sure they run at the minimum best time (3 seconds).

On the other hand, stress testing is not implemented because our program shall always have one user at max, running the program at each time. Also, a single image shall be selected at each program run. Hence, the system shall never be at stress at any given time. Testing results are recorded below:

1. **Participants:** Elbaraa, Aref
2. **Pass/Fail criteria:**
 - (a) Pass - On given input, the output is displayed after 15 seconds maximum.
User and admin interactions in GUI like getting images to select and clicking on check boxes and buttons is instant (milliseconds response)
 - (b) Fail - Output takes more than 15 seconds to get displayed.
User and admin GUI interactions take a lot of time, more than 1 second to respond or load.
3. **Methodology:** Check response times of the following GUI parts:
 - (a) Buttons - response time less than 1 second
 - (b) Check boxes - response time less than 1 second
 - (c) User image selection - response time less than 1 second.
 - (d) User output - output displayed after less than 15 seconds
 - (e) Admin image database access - response time less than 2 seconds
 - (f) Admin CNN models access - response time less than 2 seconds
4. **Test case:** All response times are fast according to the speeds specified.
5. **Actual Results:** All interface response times speeds are as specified above.
6. **Results:** Passed

9.5 Recovery Testing

Recovery testing ensures that the Iris Analyzer system is able to return to the first stage/ or to a previous part of the program in the case of an unprecedented break or crash in the system. The purpose of this test section is to ensure the stability of the user interface integration but more importantly the program as a whole.

It also makes sure that the system shall continuously run even after crashes or breaking.

1. **Participants:** Evans, Sameh

2. **Pass/Fail criteria:**

(a) Pass - The system returns to the previous part that was accessed before the crash.

The system shall not hang and stop running after crash.

(b) Fail - The system stops or freezes after crashing.

3. **Methodology:** We did it in different parts of the program:

(a) Admin login - if the system crashes during login it takes the admin back to the login page.

(b) User GUI part(choosing input image and output wanted): returns the user to main user page.

4. **Test case:** The system goes back to the previous or original state after crash.

5. **Actual Results:** For all areas specified above, after crash the system reverted to the previous state.

6. **Results:** Passed

9.6 Security Testing

For security testing, since the system is working with a data-set that includes high-quality private data type images, it is important to consider attempts of outsider data theft. In order to combat this, the Iris Analyzer has a standard authentication system for the login details of the admins which is stored independently from the system. The images and other data are only in read only mode to the users so cannot be manipulated. Furthermore, to ensure there is no unauthorized access to the data from the User-side page, thorough anti-loophole data leakage checks are done.

1. **Participants:** Evans, Sameh

2. **Pass/Fail criteria:**

(a) Pass - All private files and folders are safe and cannot be accessed by unauthorised users.

No leakage in any part of the system that shall cause unauthorised access or can lead to data being changed.

(b) Fail - Private data can be accessed by unauthorised users.

Leakage that can cause access by unauthorised users or change of data.

3. **Methodology:** All private files like admin login details are stored locally but not in the system(code) so cannot be accessed by unauthorised users.

Database and other data are read only so cannot be manipulated or changed unless b the admin.

Testing was done by trying to access parts of the system by checking for leakages. Also, we gave access to a few people to try and break into the system to check for loop holes.

4. **Test case:** All parts of the system specified above are secure and no loop-holes or leakages.

5. **Actual Results:** All areas of the system specified for security cannot be accessed by unauthorised users. No leakages or loop-holes found.

6. **Results:** Passed

Chapter 10

Conclusion

The process of iris analyzing is not something new in the technological world we are living in, however, the implementation of CNN algorithms and embedding them within an Iris Analyzer software is a world that has much room for research and improvements. Our Iris Analyzer software was initially intended to be used by target scientists and researchers who work in fields related to image processing and machine learning however after further progress and work on the software and with the insight of the Computer Engineering Department faculty we came to the conclusion to provide for the possibility of the software to be applicable for institutions such as police departments wanting to work with the analysis of Iris data using their own extensive database.

10.1 Retrospective

Building upon the foundation we laid from the first semester of the project, we continued working on the Iris Analyzer immediately this semester. The team began by first analyzing the prior problems and errors that was brought to our attention from the faculty of the Computer Engineering department. The first concern was regarding the project itself and the direction in which it was heading as well as the target audience the software could be used by, this forced our team to reconsider several design patterns as well as introducing the ability for the user of the software to be to implement their own database and retrain the model as they desire. Moving on-wards with the project the team struggled with an obstacle at the start which was with the introduction of GitHub into the scheme of plans, however, we were able to learn more about GitHub and it's features which helped greatly in the flow of the project. During the middle phase of the project and semester which could categorically fall under the second sprint, the team made the most significant and apparent progress here by starting and designing the Graphical User Interface of the Iris Analyzer. This phase came with some obstacles such as the agreement and implementation of certain design concepts and ideas, mostly related with the output panel for the results shown to the user on the GUI User page.

One of the largest obstacles was the security concern when the designing aspect of the CNN model in the GUI was being worked on. The team overlooked the idea of showing and giving direct access to the source code for the admin of the Iris Analyzer software as we had changed the software to accommodate for usage by 3rd party owners; instead, the team worked to change this implementation and allow for the use of other databases while being able to retrain the model and keeping the source code safe.

Throughout the semester, we were able to continuously optimize and improve the

CNN model for both Identification and Gender which is a continuous process and were able to repeatedly increase accuracy results compared to last semester and the start of this semester. In Summary:

What went wrong throughout the semester:

- Sprints were not well defined in terms of duration and were more abstract in nature and feel
- Testing period took longer than expected and interfered with some report and decision making choices
- Testing and tests were not logged efficiently at the start and were not completed on the expected, calculated time
- Gave direct access to source code, big security concern

What was improved throughout the semester:

- More detailed note-taking during meetings and for Trello Board
- Ideas and faculty insight from previous project presentation and demo was taken into consideration and integrated into project
- Work folders were organized more efficiently using Google Drive and GitHub
- Each member acted responsibly in terms of keeping up with critical events and the project as a whole

What can still be improved (in the future if applicable:

- Further optimization of CNN model (optional)
- Underlying understanding and implementation of various tests and testing phase lengths

10.2 Future Work

Since the project has mostly come to a completion state, future work on the project is limited but not impossible or over. There is always room for improvement and updates as the project can always be improved in terms of achieving even greater accuracy and more visually pleasing graphical user interface.

An aspect that could be further improved can also be the security of the overall software; since we are planning for it to be accessible to specific parties who would like to use it with their own database, it should ensure heavy security as databases are most commonly a private entity.

Chapter 11

References

1. M. Fairhurst, M. Erbilek and M. C. D. C. Abreu, "Exploring Gender Prediction from Iris Biometrics", International Conference of the Biometrics Special Interest Group (BIOSIG),
2. R. P. Wildes, "Iris recognition: an emerging biometric technology," in Proceedings of the IEEE, vol. 85, no. 9, pp. 1348-1363, Sept. 1997, doi: 10.1109/5.628669.
3. Alaa S. Al-Waisy, Rami Qahwaji, Stanley Ipson, Shumoos Al-Fahdawi, Tarek A.M., Nagem "A multi-biometric iris recognition system based on a deep learning approach". Pattern Anal Applic(2018) 21:783-802. <https://doi.org/10.1007/s10044-017-0656-1>
4. "Can You Lose Your Fingerprints?", Katherine Harmon on May 29, 2009 <https://www.scientificamerican.com/article/lose-your-fingerprints/>
5. Sreya K C , Dr. Rini Jones S B, "Gender Prediction from Iris Recognition using Artificial Neural Network (ANN)". DOI : <http://dx.doi.org/10.17577/IJERTV9IS070394>