USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science

B.S. Computer Science

FINAL YEAR PROJECT REPORT

Batch-2019

MAKKHIMETER

By

Shaiz Akber	19B-125-CS
Muhammad Huzaifa	19B-139-CS
Abdul Wahab Butt	19B-088-CS
Mehad Bin Naveed	18B-035-CS
Imran Riaz Chohan	17B-043-CS

Supervised by

Dr. Muhammad Wasim

ST-13, Block 7, Gulshan-e-Iqbal, Abul Hasan Isphahani Road, Opposite Safari Park, P.O. Box 75300, Karachi, Pakistan. Phone: 34978274-5; 34994305; 34982476; http://www.uit.edu

Submission Performa

Name

(1) Shaiz Akber (19B-125-CS)

(2) Muhammad Huzaifa (19B-139-CS)

(3) Abdul Wahab Butt (19B-088-CS)

(4) Mehad Bin Naveed (18B-035-CS)

(5) Imran Riaz Chohan (17B-043-CS)

Address

(1) Flat Number: 32/3 Trust Gulshan-e-Noor Housing Society opposite Dow Ojha

(2) 5-E 9/38 Nazimabad behind InterBoard Office

(3) House No, R18, Block A, Sector 16B, Scheme 33, Gulzar e Hijri

(4) A-1212 Ghousia colony near Pib colony, Karachi East

(5) L-89 sector 31/A Allah Wala Town Korangi

MakkhiMeter

Dr. Muhammad Wasim

This report is submitted as required for the Project in accordance with the rules laid down by the Usman Institute of technology as part of the requirements for the award of the degree of Bachelor Computer Science. I declare that the work presented in this report is my own except where due reference or acknowledgement is given to the work of others.

Signatures of students Date

(1) 8-August-2023

(2) 8-August-2023

(3) 8-August-2023

(4) 8-August-2023

(5) 8-August-2023

Acknowledgments

We would like to express our sincere gratitude to everyone who actively supported us during project period. We would like to thank our supervisor, Dr. Muhammad Wasim and our external supervisor, Dr. Mushtaq Hussain, for the valuable efforts, extraordinary suggestions and valuable suggestions that have guided us in the development of this entire project and drafting of this report.

Finally, I would like to thank my teammates for showing their greatest enthusiasm and dedication and working hard to reach their goals.

Abstract

This website targets researchers from all over the world who are working on Drosophila Melanogaster. No application has yet been developed for researchers which provides them with all the facilities to work on this fruit fly. So, whenever a user lands on our website it got multiple options to opt for.

It could select the wing option and could extract the wing's features such as area, perimeter and segments. The count of the hairs on the wing using the user's predefined box could also be extracted in our application besides the decision whether the wing is mutant or normal is also deduced in our application.

Besides, we also had an eye module which also calculates the eye's features such as area and perimeter. The count of the lenses on an eye using the user's predefined box is also done in our application. Predicting the mutation using the colours of the eye is also part of our application.

Lastly from those extracted features, we provide some insights to our users like: ranges of hair count, range of area and perimeter, average values of the area and perimeter etc.

We aim to provide all these features on a single platform which would help the researchers to work on this fly because this fly is highly related to human beings so any testing done on this fly is moreover applicable to humans as well.

Table of Contents

Contents

DECLARATION..... ERROR! BOOKMARK NOT DEFINED.

ACKNOWLEDGMENTS **3**

ABSTRACT **4**

TABLE OF CONTENTS **5**

LIST OF FIGURES **8**

LIST OF SYMBOLS AND UNITS **9**

1. INTRODUCTION **11**

2. BACKGROUND AND LITERATURE REVIEW **15**

2.1 RELEVANT LITERATURE AND ALGORITHM 15
2.2 SIMILAR APPLICATIONS 19
2.3 ACTIVITY DIAGRAM 22
2.4 USE CASE DIAGRAM 24

3. AIM AND STATEMENT OF PROBLEM **25**

3.1 IN SCOPE 26
3.2 OUT SCOPE 26

4. HARDWARE, SOFTWARE ANALYSIS AND REQUIREMENTS **28**

4.1 HARDWARE 28
4.2 SOFTWARE 28

5. SOFTWARE DESIGN AND MODELING **30**

5.1 ARCHITECTURE 30
5.1.1 System Architecture *30*
5.2 UML DIAGRAMS 31
5.2.1 Introduction *31*
5.2.2 Class Diagram of MakkhiMeter *32*
5.2.2.1 Wing Dimensions *32*
5.2.3 Object Diagram *36*
5.2.4 Sequence Diagram *40*
5.3 ER DIAGRAM 44
5.4 HIGH-FIDELITY PROTOTYPE 45
5.4.1 Selection of Module *45*
5.4.2 Eye Dimensions *46*

FIGURE 5.4.2.2: UPLOADING WRONG IMAGE **47**

5.4.3 Eye Bristles *51*

6. ALGORITHM ANALYSIS AND COMPLEXITY **53**

6.1 WING DIMENSIONS 53
6.1.1 Preprocess *53*
6.1.2 Process *56*

6.1.3 Post-Process *60*
6.2 WING BRISTLES 62
6.2.1 Preprocess *62*
6.2.2 Process *64*

7. IMPLEMENTATION 66

7.1 IMPORTANT CODE/INTERESTING ASPECTS 66
7.1.1 Code for colouring the segments of the wing *66*
7.1.2 Wing Bristles Counter *69*
7.1.3 Eye Colours Detection *71*
7.2 COMPONENT DIAGRAM 75
7.3 OPERATIONAL DIAGRAM 76
7.4 DEPLOYMENT DIAGRAM 77
7.5 TECHNICAL INTERFACES 79
7.6 STATE TRANSITION DIAGRAM 82

8. TESTING 83

8.1 WHITEBOX TESTING 83
8.1.1 Wing Dimensions & Eye Dimensions *83*
8.1.2 Wing Bristles & Eye Ommatidium *84*
8.1.3 Wing Shape *85*
8.1.4 Eye Color *86*
8.2 BLACKBOX TESTING 88
8.2.1 Wing Dimension & Eye Dimensions *88*
8.2.2 Wing Bristles & Eye Ommatidium *90*
8.2.3 Wing Shape *92*
8.2.4 Eye Color *94*
8.2.5 Homepage *96*
8.2.6 Dashboard *98*
8.3 INTEGRATION TESTING 100
8.3.1 Bug List *102*

9. CONCLUSIONS 103

10. FUTURE WORK 104

11. ACHIEVEMENTS 106

12. APPENDICES 107

12.1 ANNEX A 107
12.2 ANNEX B 111
12.3 ANNEX C 113
12.4 ANNEX D 115
12.4.1 Architecture *115*
12.4.2 UML Diagrams *116*
12.5 ANNEX E 129
12.5.1 Deployment Architecture *129*
12.6 ANNEX F 131
12.7 ANNEX G 131

13. REFERENCES 132

List of Tables

Table 8.2.1: TestCases for Eye & Wing Dimension…….....89
Table 8.2.2: TestCases for Wing Bristles & Eye……..91
Table 8.2.3: TestCases for Wing Shape.……...93
Table 8.2.4: TestCases for Eye Color…….. 95
Table 8.2.5: Homepage…….97
Table 8.3: Integration Testing.…….100
Table 8.3.1.1: Missed out segment in Wing Dimension.……102
Table 8.3.1.2: Missed Calculations of Mutant Eye……..102

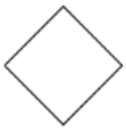
List of Figures

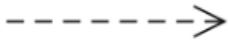
Figure 1.1: Normal & Defected wings ……....11
Figure 1.2: Eye Color & Eye Shape ……..…....12
Figure 1.3: System Diagram ……..…... .14
Figure 2.1: Preprocessing Techniques ……..……..15
Figure 2.2: Advance Preprocessing Techniques.……...…..16
Figure 2.3: Segmenting the wings feature ……..…....16
Figure 2.4: Wing Bristles ……... 17
Figure 2.5: Processing Ommatidium …….. 17
Figure 2.6: Working of [1] ……. 19
Figure 2.7: Working of [2] ……. 20
Figure 2.8: ImageJ working ……... 21
Figure 2.9: Activity Diagram ……. 23
Figure 2.10: Use Case Diagram ……. 24
Figure 5.1.1: Architecture of MakkhiMeter…… 30
Figure 5.2.2.3: Class Diagram of MakkhiMeter……. 35
Figure 5.2.3.1: Object Diagram……..... 36
Figure 5.2.4.1: Sequence Diagram…….. 40
Figure 5.3: ER Diagram…….. 44
Figure 5.4: High-Fidelity Prototype…… 45
Figure 6.1.1.1 Histogram Equalizing…….. 54
Figure 6.1.1.2: Median Filter…….. 55
Figure 6.1.2.1: ConnectedComponents……... 57
Figure 6.1.2.2: Mophological Close…… 57
Figure 6.1.2.4: WaterShed…….. 59
Figure 7.2: Component Diagram…… 75
Figure 7.3: Operational Diagram…… 76
Figure 7.4: Deployment Diagram…… 77
Figure 7.5: Technical Interface……79
Figure 7.6: State Transition Diagram…….. 82
Figure 8.1: FlowGraph…… 83
Figure 8.2.6: Dashboard…….. 98
Figure 12.2: Pre-Processing-techniques…….…….. 114
Figure 12.2: Wing Bristles……...115
Figure 12.4.1: System Archeitecture …….……...118
Figure 12.4 (1): Use Case Diagrams……..……...121
Figure 12.4 (2): Activity Diagram…….……122
Figure 12.4 (3): Wing Dimension Class Diagram…….……124
Figure 12.4 (4): Wing Bristles Class Diagram……..……125
Figure 12.4 (5): Wing Dimension Sequence Diagram……..……127
Figure 12.4 (6): Wing Bristles Sequence Diagram……...128
Figure 12.4 (7): Wing Dimension Object Diagram……...129
Figure 12.4 (8): Wing Bristles Object Diagram ……...130

List of symbols and Units

To assist the reader, we need to include a list of symbols and units. This should include Greek letters or other mathematical symbols, as well as the quantity and appropriate units they refer to.

Table 1: List of Symbols

	Start of the system
	Actor
	Interaction with other use cases
	Decision making
	Start Symbol
	Realization/ Implementation
	Association

	Dependency
	Inheritance
	Aggregation
	Composition
	Split the activity into multiple nodes
	End Point

1. Introduction

The Genetic analysis of humans is very important in the medical field. As we know that the human genome is highly related to the genome of chimpanzees, but the problem is that it is very difficult to store a lot of chimpanzees in the lab and if they are alive so it will take a lot of time to control them. But we have another specie, which its genome matches almost 75-80 per cent of the human genome which are known as Drosophila Melanogaster a fruit fly. The Drosophila wing has an excellent track record for genetic screening studies and is ideal for phenomic studies to uncover conserved biological processes relevant to human development and diseases. The Drosophila wing has successfully identified genes crucial for organ development and relevant to human health. The relationship between fruit flies and human genes is so close that often the sequences of newly discovered human genes, including disease genes, can be matched with equivalent genes in the fly.

The important point to ponder is which features of that fly are important which makes them very cohesive with the human genome. Firstly, its wing provides enough information regarding the **mutation or defect** of flies. This mutation or defect of the wings could either be from the testing of medicine, breeding etc and because of this, we see the change in the wing structure or the wing bristles. The dominant wing is shown in **Figure 1.1.**

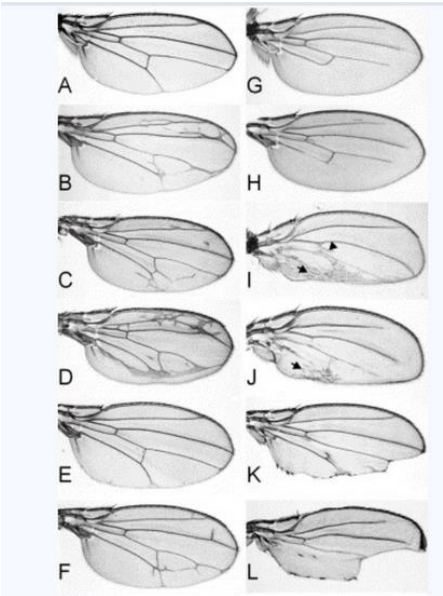


Figure 1.1: Normal and Defected wings

Secondly, the eye of the fly also plays a vital role when testing medicines, and breeding is done. This eye contains approximately 700-800 lenses in its single eye. If there is any mutation or defect in the eye so mainly the eye colour, shape and lens's structure change. The eye is shown in **figure 1.2.**

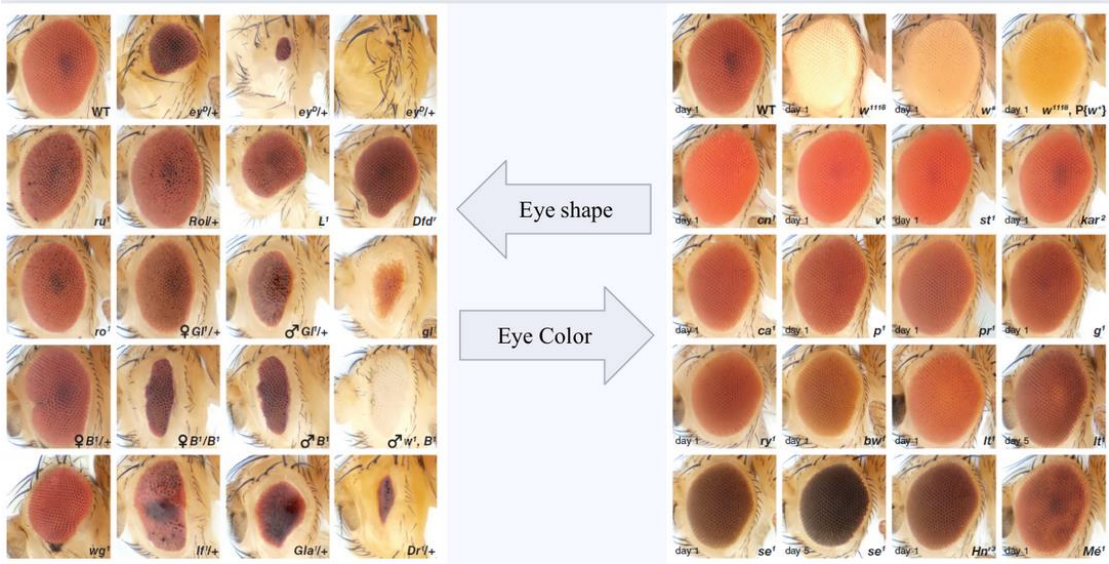


Figure 1.2: Eye colour and Eye shape

The project is based on Image Processing and Deep Learning followed by some analysis. In the Image Processing phase, we will preprocess the image and will apply some processing techniques to extract the different segments of the wing including the whole segment. Secondly, we will also find the number of hairs count in the wing in the user's predefined box provided. In the deep learning phase, we will predict the shape of the wing we will train our deep neural network which will extract some important features of defective and normal and will predict the shape of the wing.

This same technique as discussed above will apply to the eye as well with some extensive preprocessing of the eye images. The Deep learning model will be used to predict the 3 primary colours of the eye which mainly denote the mutation and normal eye.

After extracting all the features of the wing and eye we will be generating some insights from our data which includes the possible ranges of parameters, regression trend etc.

What we will be making is:

- Web Based Application to upload and show insights.
- Using an Image Processing Pipeline to preprocess the images and relevant processing algorithms will be applied to the wing and eye image.
- Using Deep Learning our model will be trained by using several different images of mutation and normal (mainly involving the structure and colour) of the eye and wing.
- The website will be hosted on the Cloud.
- Some useful insights will be provided based on the features extracted.

The detailed system diagram is shown in **Figure 1.3**.

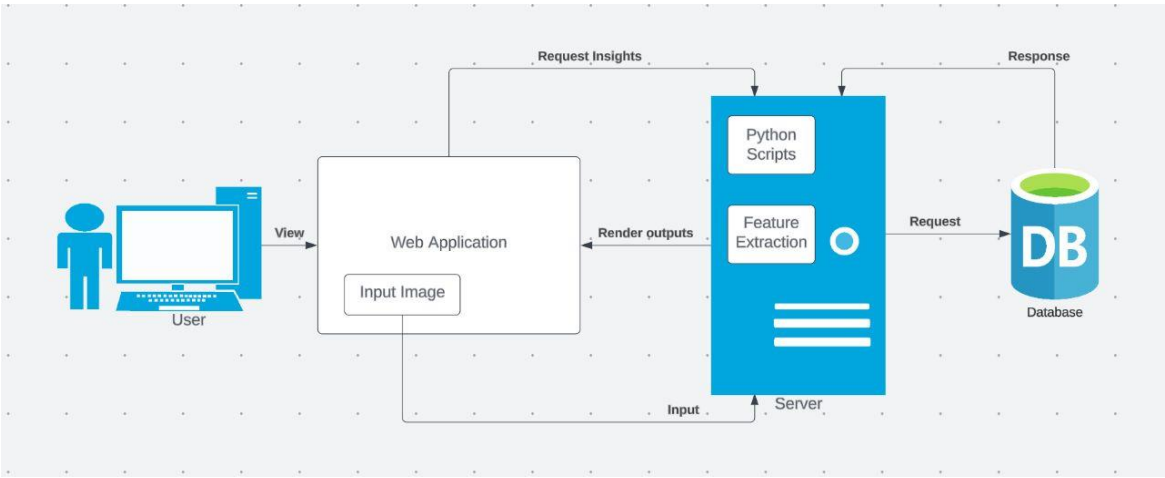


Figure 1.3: System Diagram

As shown in the system diagram the user could view the **web application**. Then the user got the option to either select the wing module, or eye module or to see the insights. After the decision is made to select either wing or eye the image is passed as **input** and the image is sent to the **web server** which contains the server-side scripts. The calculations which involve the **feature extraction** are done and the results are stored in the **database**. The result from the **database** is sent back to the **web server** which then renders the output back to our **web application**.

If the user wishes to see the insights, then the user's request is sent to the **web server** and then to the **database** which will respond to the **web server** by providing the data to the dashboard and the results are rendered on our **web application**.

2. Background and Literature Review

2.1 Relevant Literature and Algorithm

Research has been conducted on the Drosophila Melanogaster, and there have been numerous papers published regarding the feature extraction of the wing and eye. Some of the papers were highly correlated to our project's idea.

[1]. One of the approaches is to input the image and click the threshold, edge detection to get the edges of the wing and then if the wing is not complete you need to draw the edges of the complete wing. [1] requires more user work. As shown in **figure 2.1.**

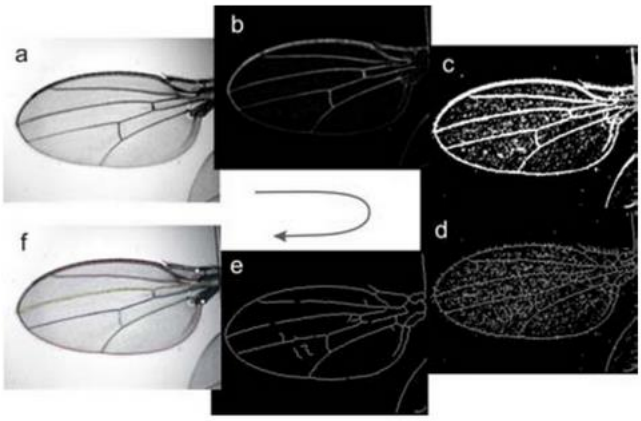


Figure 2.1: Preprocessing Techniques

[2]. Another methodology which has been discussed is to use the of contour and Distance Transformation for the preprocessing of the image. But the problem is that it also involves a lot of time and the user's manual work as shown in **figure 2.2.**

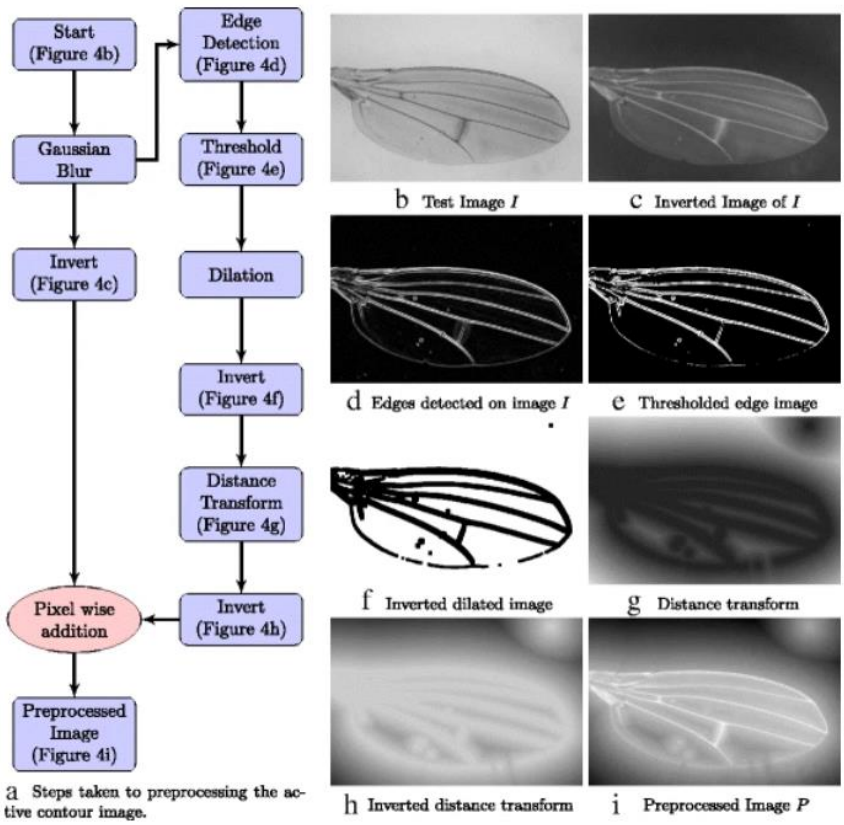


Figure 2.2: Advance Preprocessing Techniques

[3]. Another approach which has been discussed is to use the machine learning approach to extract the mask of the image and then process it to get the different segments of the wing. The approach requires a good amount of processing power and to train the model you need a lot of data to get good accuracy. The type of result of getting the features of the wing is shown in **figure 2.3**.

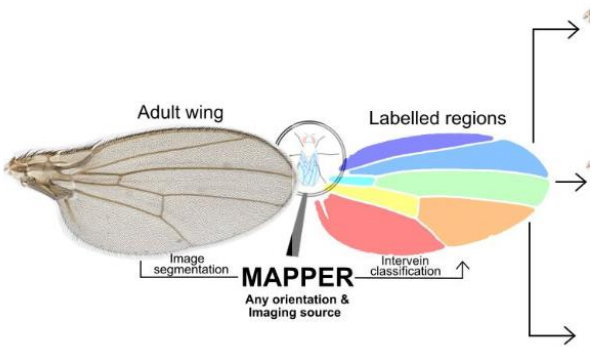
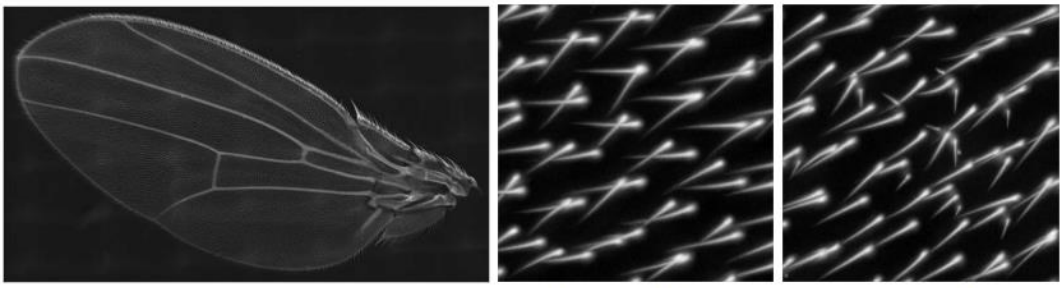


Figure 2.3: Segmenting the wing's features

[4]. One of the approaches to classify whether the wing is male or female has been discussed [4]. But nothing has been discussed to classify the mutant and normal wing.

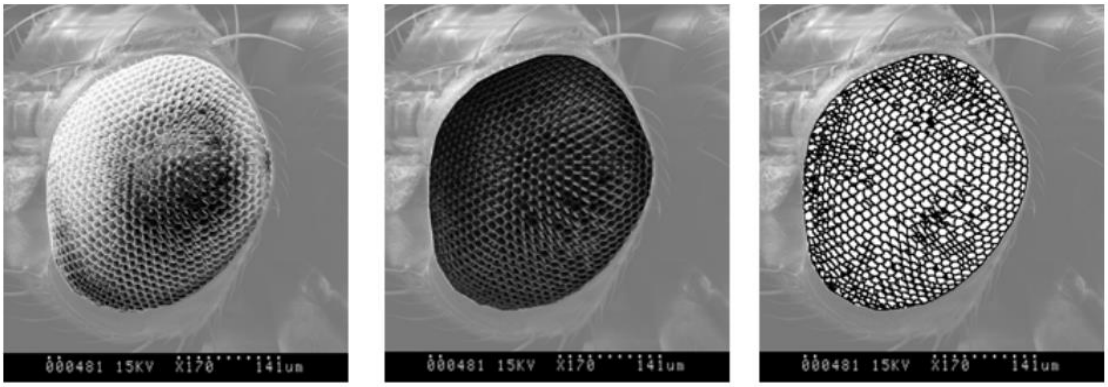
[5]. The technique to work on the bristles or hairs of the wing has been proposed to how to get the edges of that hair in the wing image. But no approach has been discussed to get the count of those hairs. The results are shown in **figure 2.4**.



(a) a wing (b) normal hairs (c) mutant hairs

Figure 2.4: Wing Bristles

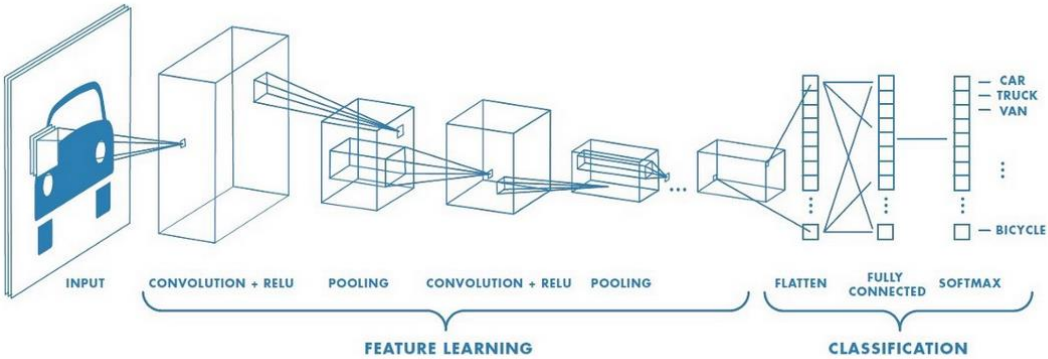
[6]. One of the methodologies discussed is to get the ommatidium count by automatically segmenting the eye. But the problem is the eye shape is curved so either you should have an expensive microscope to process the curved image, or you could just focus on the centre of the eye. Using the [6] approach won't work for the only centre-focused image as the count would be disrupted. The results have been shown in **figure 2.5**.



A: Contrast Enhancement B: Edge Detection C: Watershed Segmentation

Figure 2.5: Processing Ommatidium

Regardless, of the Image Processing, we would also be using Deep Learning techniques for classification purposes. We would be using Convolution Neural Network (CNN) for extracting complex features from our images. The structure of the CNN is as follows:



Feature Learning

The Convolutional layer will determine the output of neurons which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume [7]. The rectified linear unit (commonly shortened to ReLu) aims to apply an 'elementwise' activation function such as sigmoid to the output of the activation produced by the previous layer. Followed by the pooling layer will then simply perform down-sampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation [8].

Classification

The output of the Feature Learning will form itself into a fully connected neural network followed by the SoftMax activation function.

2.2 Similar Applications

As discussed in [1] a desktop application is available to extract the features from the wing, from the user perspective it is very difficult to use and requires a lot of user intervention. You need to input the image in the application and give the input of the number of points and select each point to get the structure of the wing and this work is all at the user end. The application is shown in **figure 2.6**.

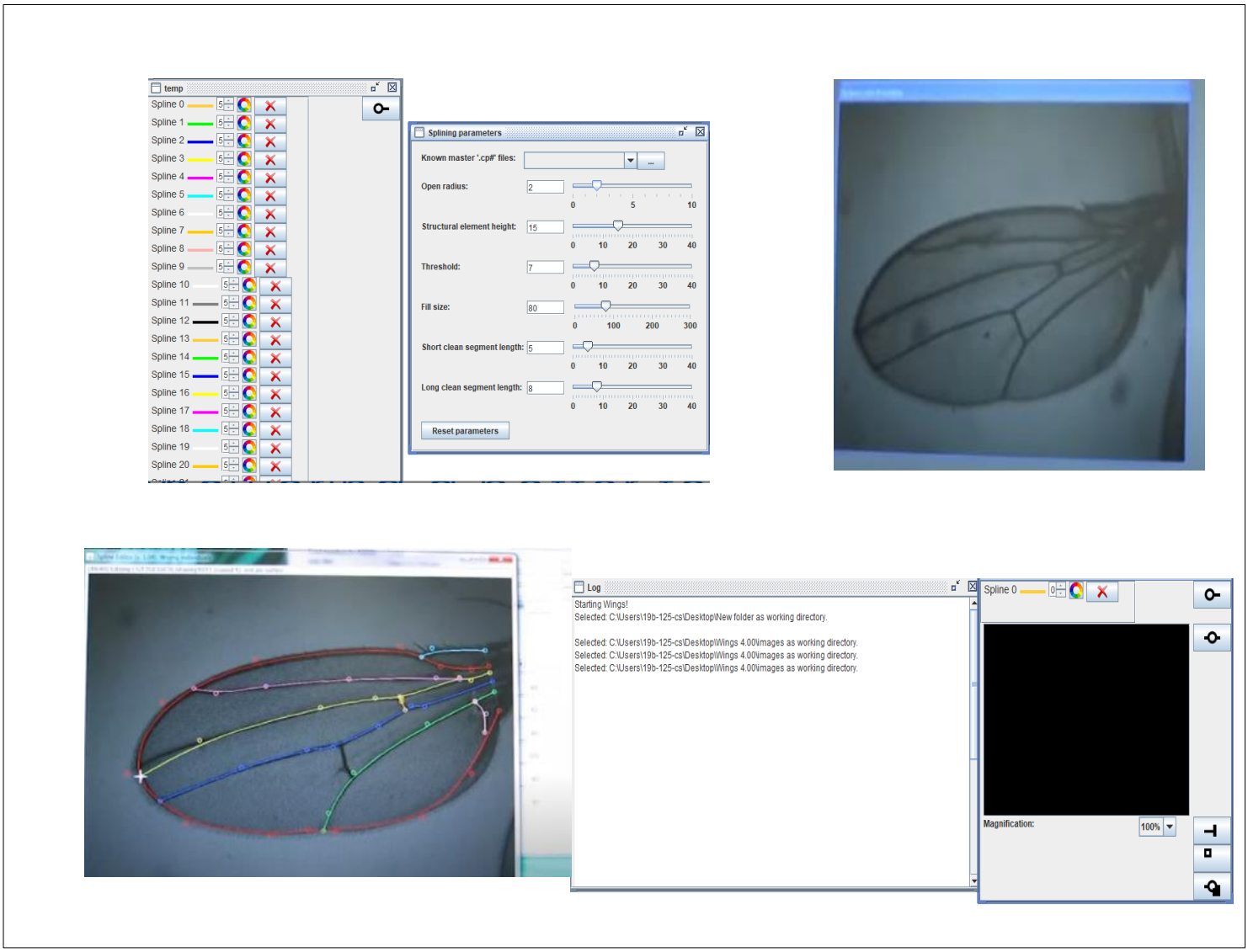


Figure 2.6 Working of [1]

The author [2] has described a similar technique but he had developed a web application which calculates the spline, distance, and intersection of points of the wing. Again, all the work is kept in the user's way. The work is described in **figure 2.7**.

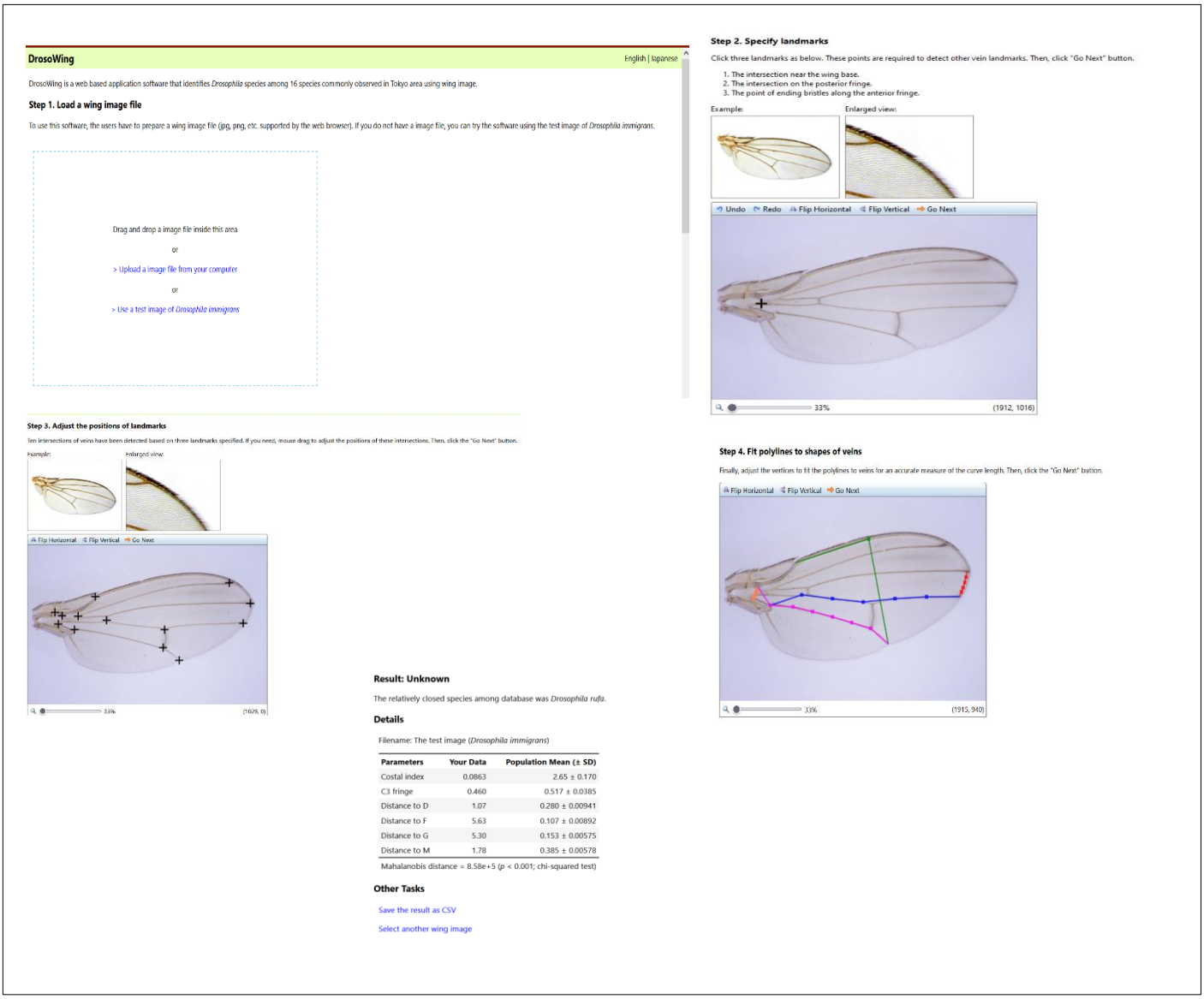


Figure 2.7 Working of [2]

ImageJ is one of the software which is mainly used to extract the features of the wing and eye. It is a desktop application and is just like the toolbox which provides the necessary processing tools such as thresholding. The working of ImageJ is shown in **figure 2.8**.

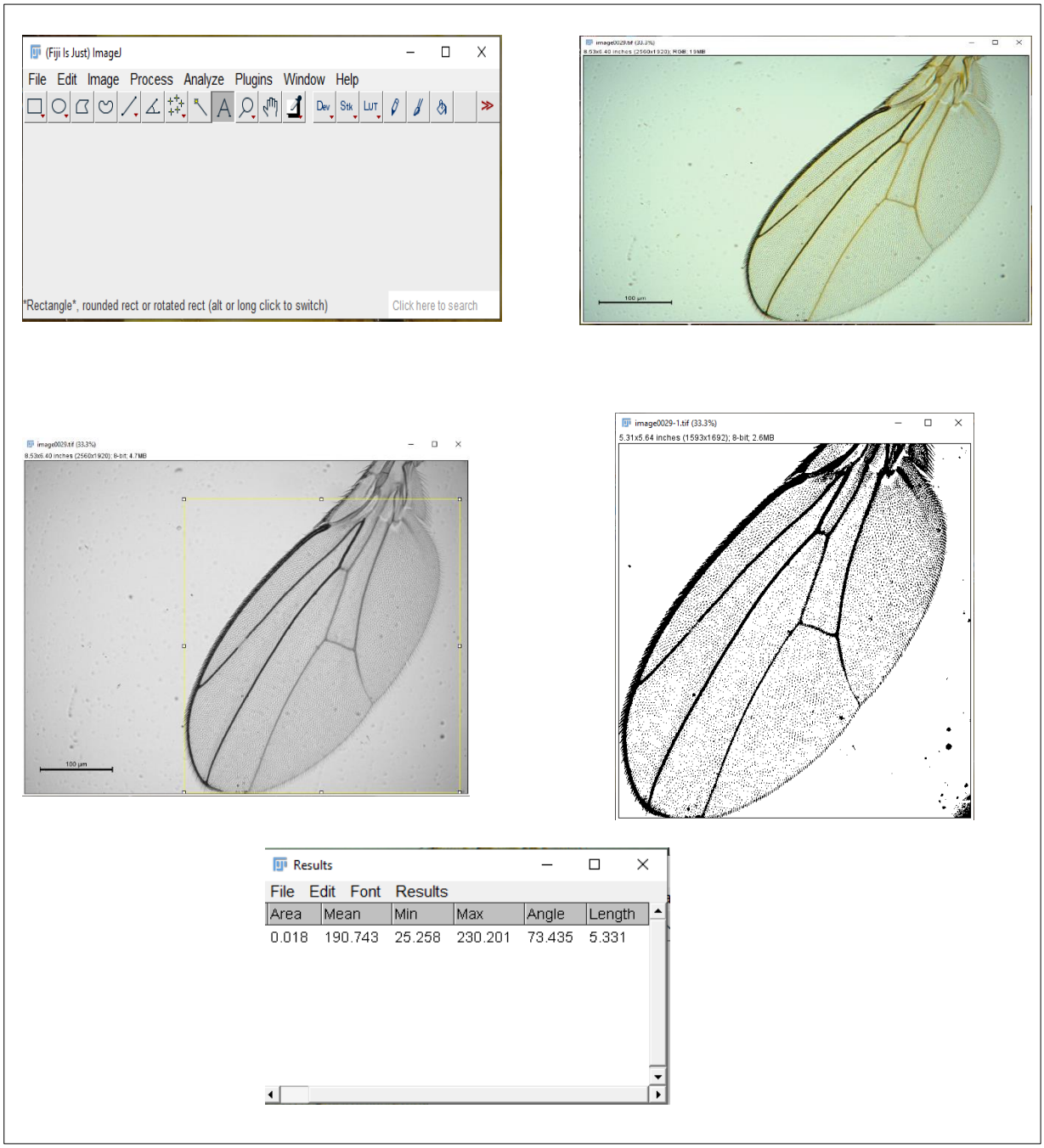


Figure 2.8 ImageJ working

2.3 Activity Diagram

In Figure 2.9 user has got 3 options either it could see the insights of the extracted features, work in the wing module or work in the eye module.

If the user selects the wing module so again it has got the option to either calculate the wing dimension, wing bristles and wing shape. It needs to select one of the options. If it selects the wing dimensions so the input is preprocessed, thresholding and dilation are done, then the segments of the wing are calculated, and results are shown in a data frame. If the wing bristles are selected so the user needs to select the area to get bristles and we would also be calculating whole wing bristles. Lastly, for wing shape the deep learning is loaded, preprocessed image is passed into it, features are extracted by the model and results are displayed.

The same flow is for the eye module with the changes in the preprocessing techniques and getting some extra features concerning the eye.

If the user selects insights, then the dashboard will be shown which will be displaying the relevant insights from the extracted features.

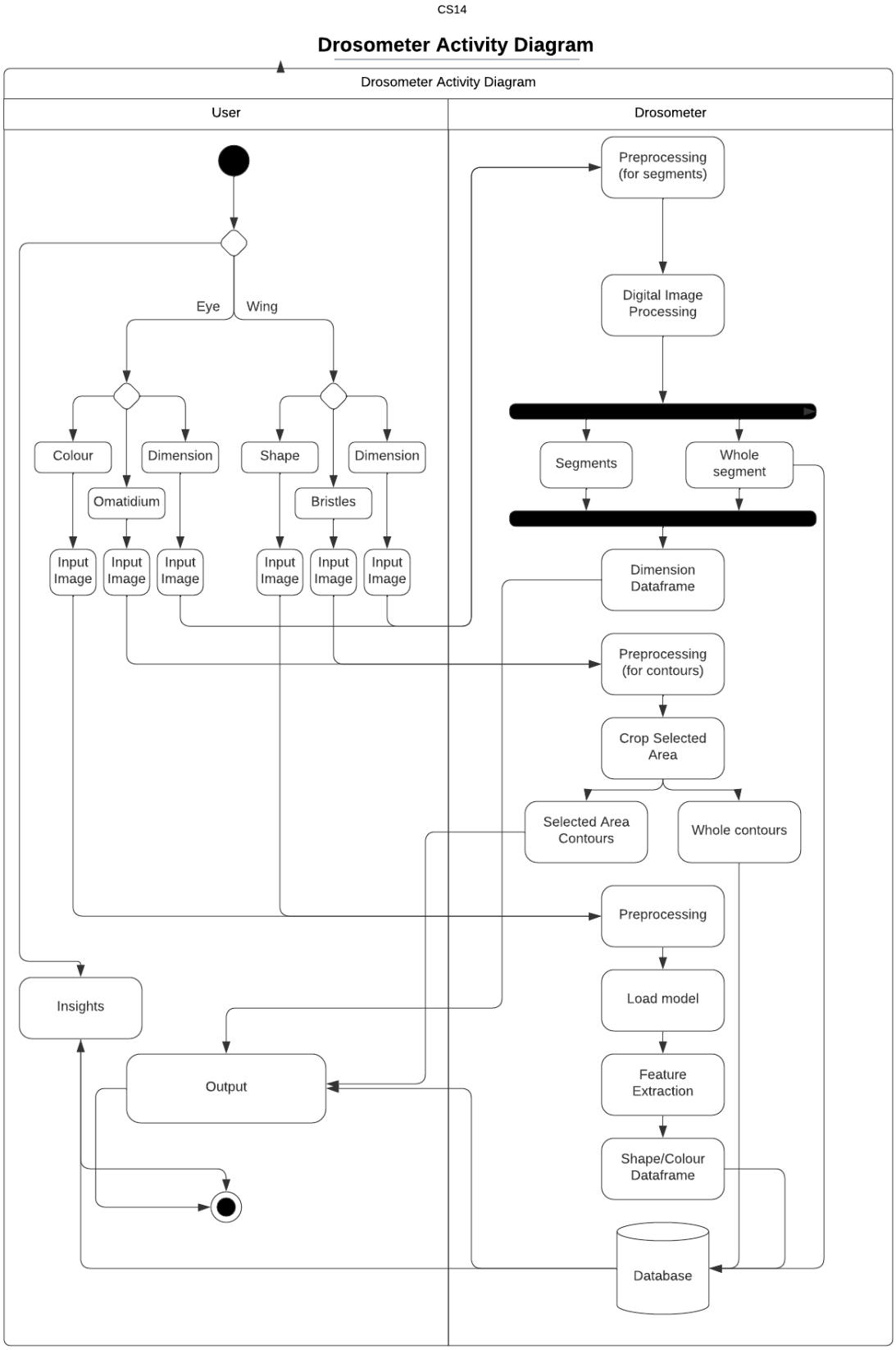


Figure 2.9: Activity Diagram

2.4 Use Case Diagram

In **figure 2.10** we have got 2 actors one is our user and second is system (MakkhiMeter). User could upload the image in either wing or eye module or could see the analysis/insights of extracted features. All the processing is done in the system which includes the wing shape and eye color, dimension of eye and wing, wing bristles and eye ommatidium.

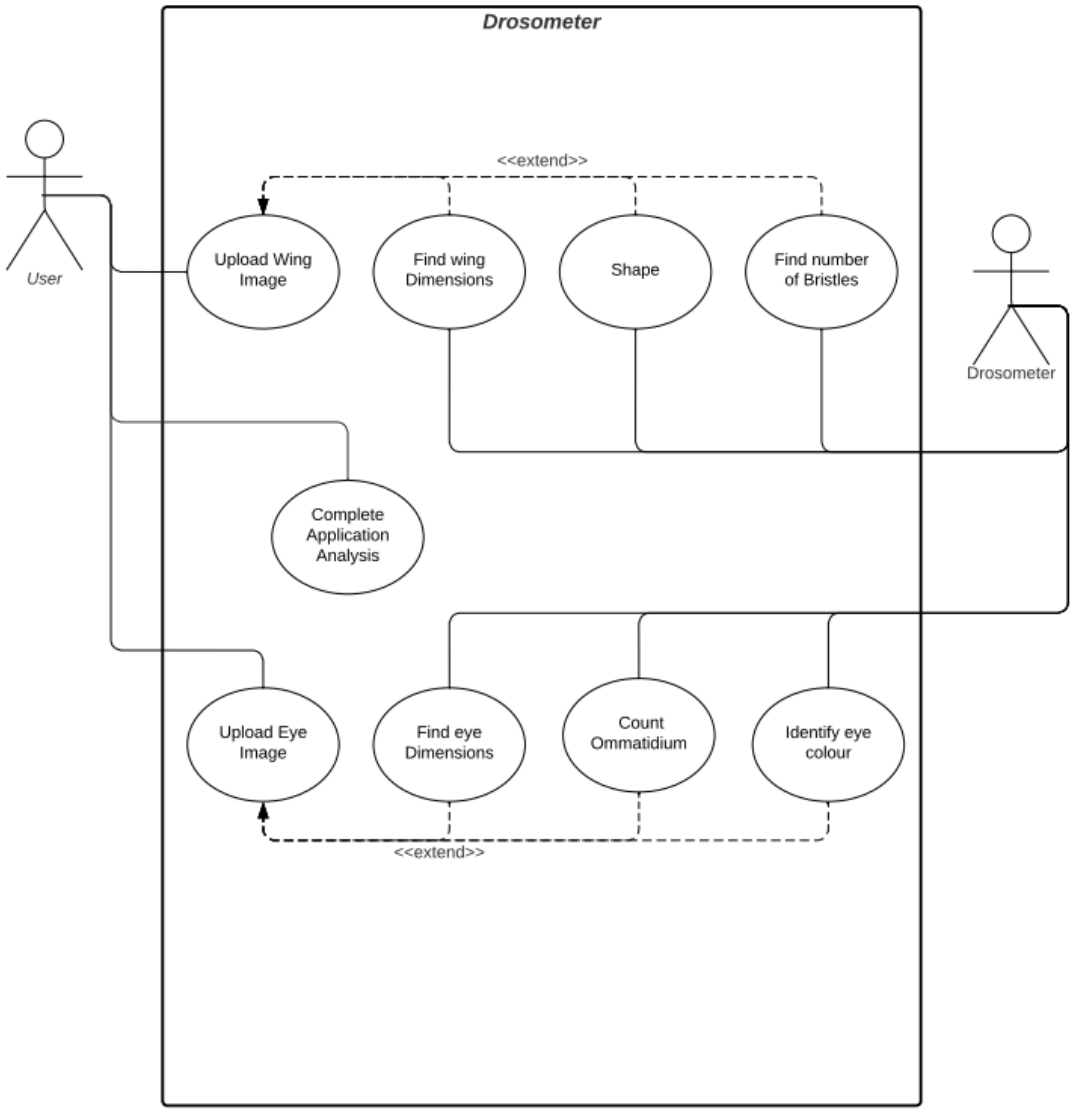


Figure 2.10: Use Case Diagram

3. Aim and Statement of Problem

Priori's work on Drosophila Melanogaster feature extractions was scribble based which requires a lot of user intervention and some of these features were nearly impossible to extract such as hairs count in the wing and the number of lenses, and its shape change in the eye. We aim to make this model almost user-intervention-free to extract the relevant features of the wing and eye. It is an approach that exploits the Image Processing techniques and Deep Learning Methodology which will then be evaluated on the multiple images of the wing and eye which includes any orientation of the wing and eye. This project aims to find the features of the Wings and Eyes.

1. **Wings**

   Firstly, in the wings module, we will be extracting the features such as **Areas and Perimeter** of the segments and the whole structure. Secondly, we aim to find the number of **Bristles** mainly known as hairs count in the wing. Thirdly, we would be predicting whether the given image **is Mutant or Normal**.

2. **Eye**

   In the eye module, our first aim would be to find the **Area and Perimeter** of the whole eye. Secondly, we will find the number of **Ommatidium** mainly known as the lens in an eye. Thirdly, we would be predicting the colour of the eye which mainly involves the **3 Major Colors** which define whether it's a mutation or not.

Then our application will generate useful **insights** based on the features extracted such as regression trend, area ranges of eye & wing, perimeter ranges of eye & wing etc.

The scope of this project includes and excludes the following items.

3.1 In scope

1. Web Based Application to upload and show insights.
2. Using an Image Processing Pipeline to preprocess the images and relevant processing algorithms will be applied to the wing and eye image.
3. Using Deep Learning our model will be trained by using several different images of mutation and normal (mainly involving the structure and colour) of the eye and wing.
4. The website will be hosted on Cloud.
5. Some useful insights will be provided based on the features extracted.

3.2 Out Scope

Certain things that our project does not cover are mentioned below:

Object too close to the border

When the Eye or wing is too close to the border of the image then there is no guarantee that the algorithm will return the correct features of the wing and eye.

Predicting the sub-mutation of the fly

The limitation of our application is that whenever it predicts that the given image is mutant, it does not highlight to which mutation type this belongs as there are 12 types of mutation in this fly. As shown in **Figure:1**

Internet Connectivity

The process of extraction & classification takes some amount of computation and to make it as accessible for everyone as possible we aim to deploy it on the cloud which means it requires a constant internet connection to function properly as well as intended.

The problem was that when users do these features extraction work manually, so it takes a lot of time to analyze the wing or eye draws line manually in each segment or whole

segment to get its approximate area or perimeter. Some features such as lens count or the bristle's work were nearly impossible to calculate.

So, our approach which works on the Image processing and Deep learning techniques to automate this work and saves a huge amount of time as the avg wing extraction takes 10 minutes for getting areas and perimeter so our system will get these in 2 minutes so we are saving the 80% time of our user. Lastly, our application will be providing insights as well which could lead to certain new findings.

4. Hardware, Software analysis and requirements

Below is the hardware and software that we have used in our project.

4.1 Hardware

For the computation the machine we wil use have the following specifications.

- CPU: Intel(R) Core(TM) i5-6300U CPU @ 2.60GHz
- GPU: Intel(R) HD Graphics 520
- RAM: 8 GB DDR4
- HDD: 256 SSD

4.2 Software

For the working of software in our project, we have a backend and frontend for our web application. Our front end includes the user interface of the web application.

While for the backend we have server-side scripting that responds to the request from the web application. Our main work which includes features extraction, classification are completely based on Image Processing and Deep learning, will take the input in form of an image from our front end then runs the python scripts and display the output/image back to our application.

For software work, we use the following technologies.

- **Django**

  Django is a high-level Python server-side web framework that allows the fast development of maintainable and protected websites, highly secure, scalable, portable etc. As it will be used as the backend of our web application.

- **Python**

We use python programming language for our image processing and deep learning work as it has all the functionalities, libraries and frameworks related to our application.

- **Image Processing**

We will be using certain libraries e.g: OpenCV, and SkImage to accomplish the task of features extraction of wing and eye.

- **Deep Learning**

It is a type of machine learning which learns to do predictions based on testing, training and validation based on labelled datasets.

- **JavaScript**

It is the leading language for the front-end design of our websites.

- **Bootstrap**

Bootstrap is a free, open-source front-end development framework for the creation of responsive websites.

5. Software Design and Modeling

This chapter provides the analysis of the system, various UML diagrams explaining briefly regarding the working of the system. This phase also contains the Class diagrams, ERD diagrams, Object diagrams and Sequence diagrams. We will discuss the overview of web-based application as this application will immensely help our users to analyze the fly in much less time. As before our application the analyzation process of the wing used to take hours and even some of the analyzation method were not even available to our users. But our application has solved this problem by providing the analyzation in few minutes and revealing the insghts from the features besides accomplishing those methodologies which were not available.

5.1 Architecture

5.1.1 System Architecture

Our application is using a three -tier Architecture which consist of device an application interface and database at back. The user will use web application to access the content of website. User could upload the image or either wing or eye and the extracted features will be displayed on the web along with the segmentation of the image where needed. Besides the dashboard of the wing and eye is also available on our website which will be displaying insightful features to our user.

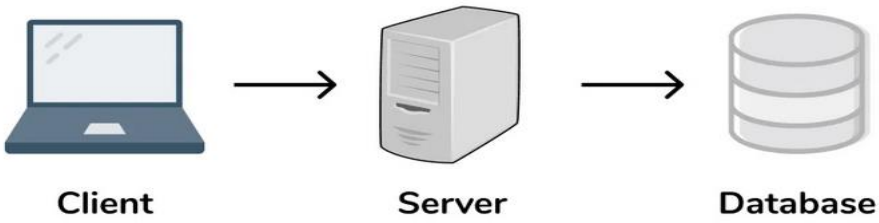


Figure 5.1.1: Architecture of MakkhiMeter

5.2 UML Diagrams

5.2.1 Introduction

Unified Modeling Language (UML) class diagram is the foremost broadly used graph sort for documentation of programming. As most computer program that are being made are still based on Object Oriented Programming (OOP) worldview, utilizing course graphs to record the software. This occurs as the basis of OOP is on the classes their relations between each other. The UML diagrams we include in this chapter are class diagram, object diagram.

5.2.2 Class Diagram of MakkhiMeter

5.2.2.1 Wing Dimensions

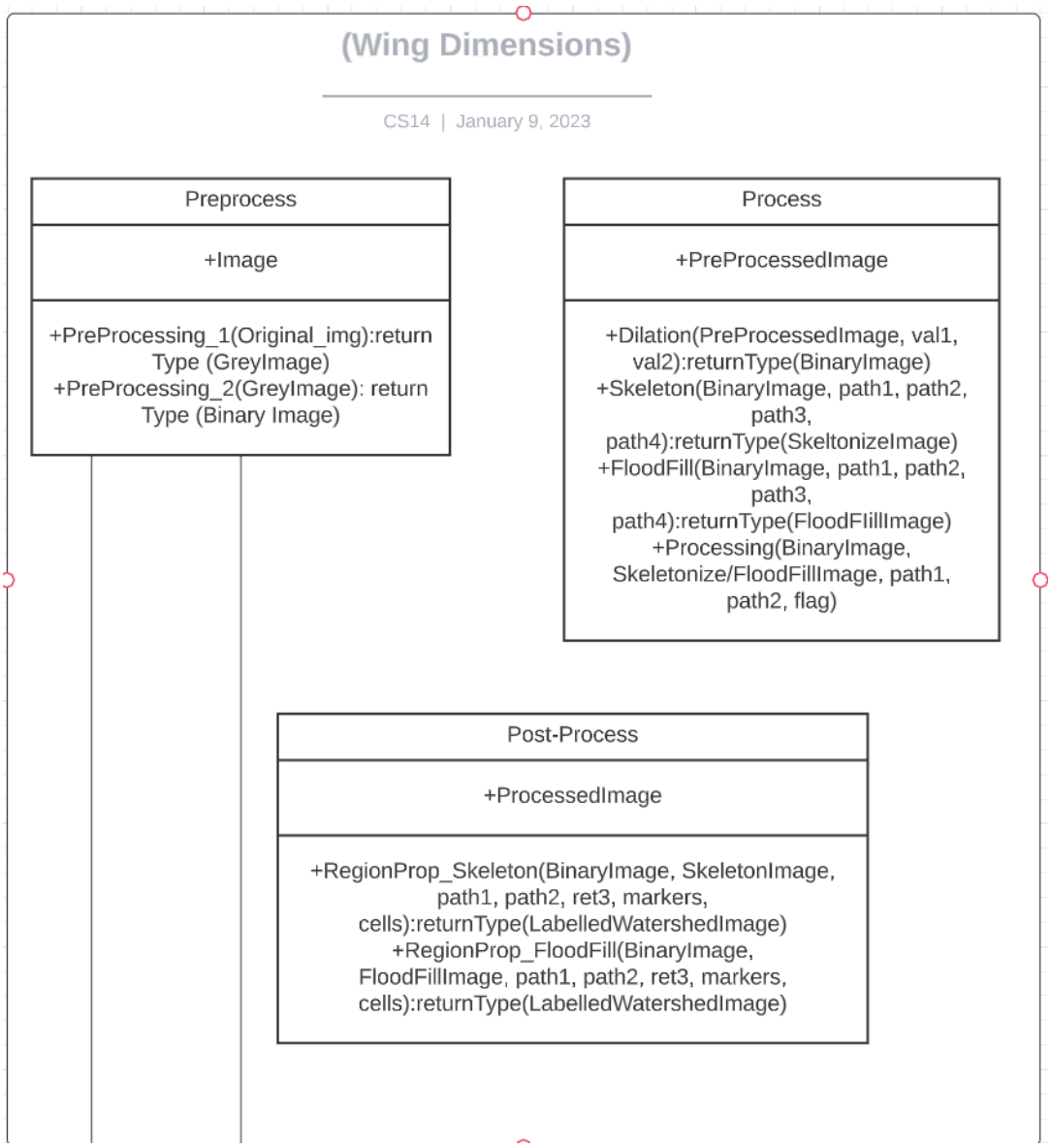


Figure 5.2.2.1: Wing Dimension Class Diagram

Description

In wing dimensions, we have three classes named Preprocess, Process & postprocessing. All the three classes are automated except one function “Dilation” that is of Processing class as it requires user’s interaction (if needed).

The original imaged is passed to the Preprocess class which have two functions, one performs some basic operations like to convert into greyscale and in the required format while the other converts the image into binary (black and white) image and passes it to the Process class.

The process class applies dilation with default values and displays it to the user. If changes are needed the user does that. In process class, we have two algorithms for finding the segments of the image. Skeleton and FloodFill, one is for the centered wing image while the other is for if the wing is touching the edges of the image frame respectively. The choice is on user.

The selected algorithm is called and it returns us the skeletonize or floodfill image, whatever the user has selected. This image is now passed to the last function of Processing that takes the algorithm’s returned image and color the seven basic segments of the wings.

This is colored image is now used as a parameter for our last class which is of Postprocessing which finds the areas and parameters of the colored segments of the image and the whole area and perimeter of the wing with help of a function called Regionprop.

5.2.2.2 Wing Bristles

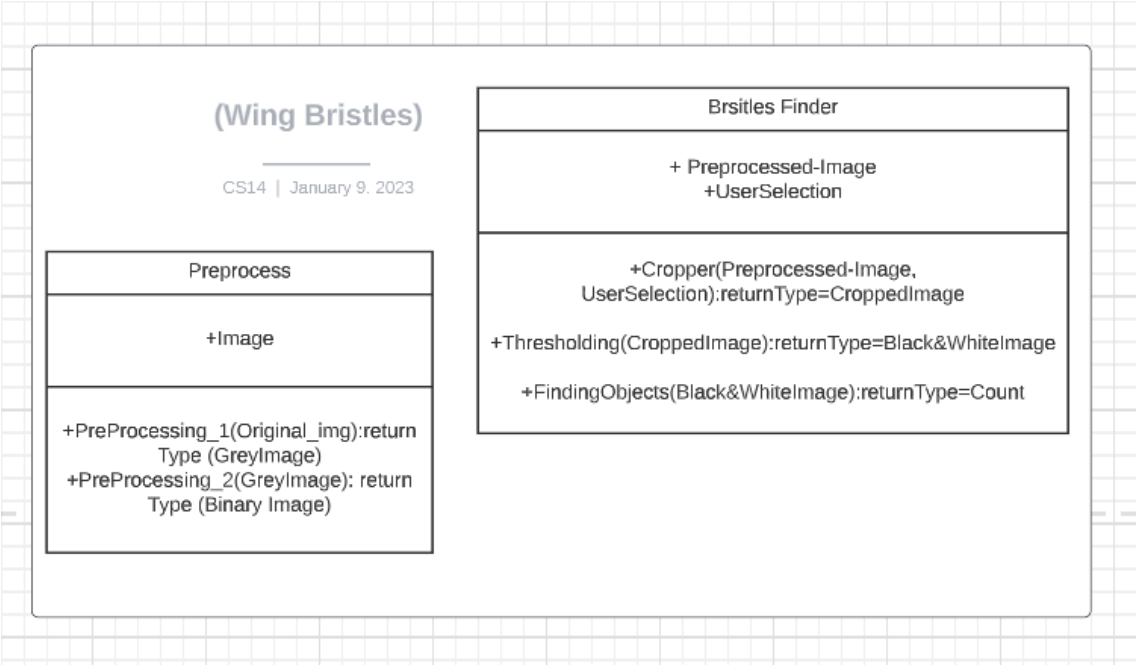


Figure 5.2.2.2: Wing Bristles Class Diagram

Description

In wing bristles we have two classes. One is of preprocess and the other is Bristles finder. The original imaged is passed to the Preprocess class which have two functions, one performs some basic operations like to convert into greyscale and in the required format while the other converts the image into binary (black and white) image and passes it to the Bristles Finder class.

The bristles finder class have three functions Cropper, Thresholding & finding objects. The cropper function holds some basic options for editing image like cropping, zooming, flipping and all. The thresholding function then applies some processing and passes the image to Finding objects functions which searches for the black dots in the cropped image and returns the count of them. In this way the count of the bristles on the selected area of the wing is achieved.

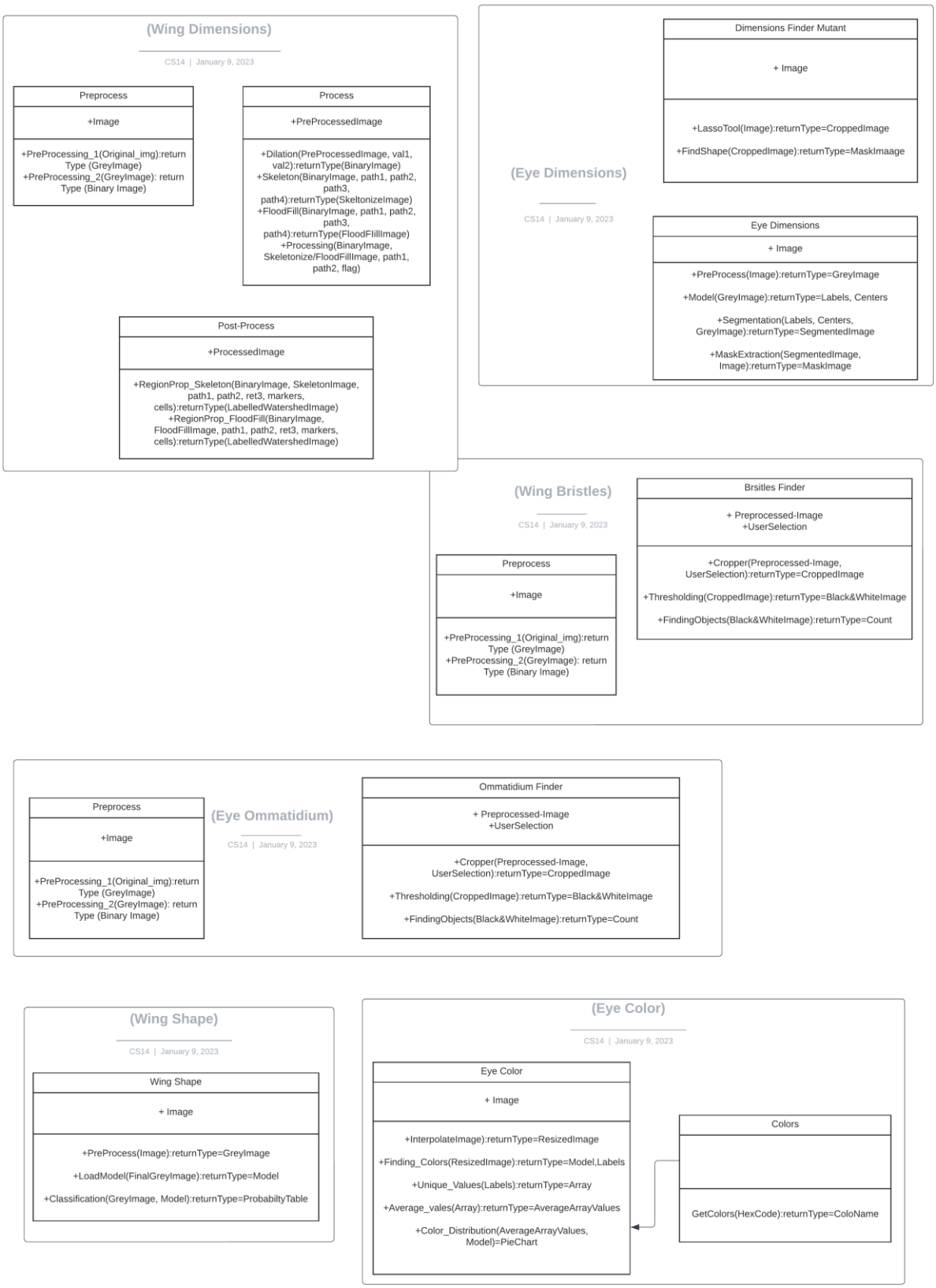


Figure 5.2.2.3: Class Diagram of MakkhiMeter

5.2.3 Object Diagram

5.2.3.1 Wing Dimensions

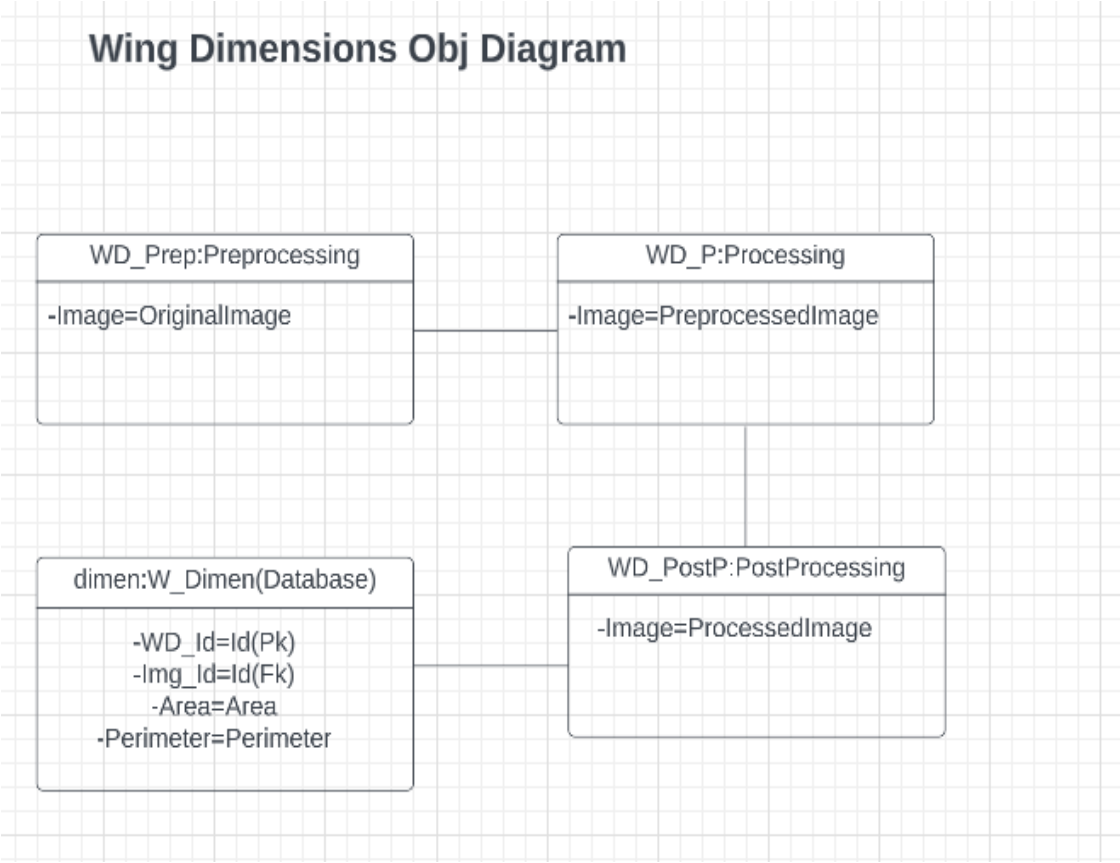


Figure 5.2.3.1: Wing Dimensions Object Diagram

Description

Object is an instance of a class in a particular moment in runtime that can have its own state and data values. Likewise a static UML object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time

The figure 5.2.3.1 depicts the flow of our objects which is completely based on our class diagram. In wing dimension module we have 4 objects 3 of which are Preprocessing, Processing, Postprocessing and last is of the database model.

Whenever the image is inputted the image is passed as an parameter to object name WD_Prep which is the object of Preprocessing class.

Secondly, the preprocessing object returns the preprocesssed image.

Thirdly, that preproceesed image object is passed to another object name WD_P which is the object of class Processing. WD_Prep will have that preprocessed image which will be passed in our WD_P which will perform processing on it.

Fourthly, that processed image object is passed into WD_Post object which is the object of Postprocessing class.

Lastly, the database class object name dimen will take the features extracted from the image along with the primary and foreign keys which are auto-incremented.

5.2.3.2 Wing Bristles

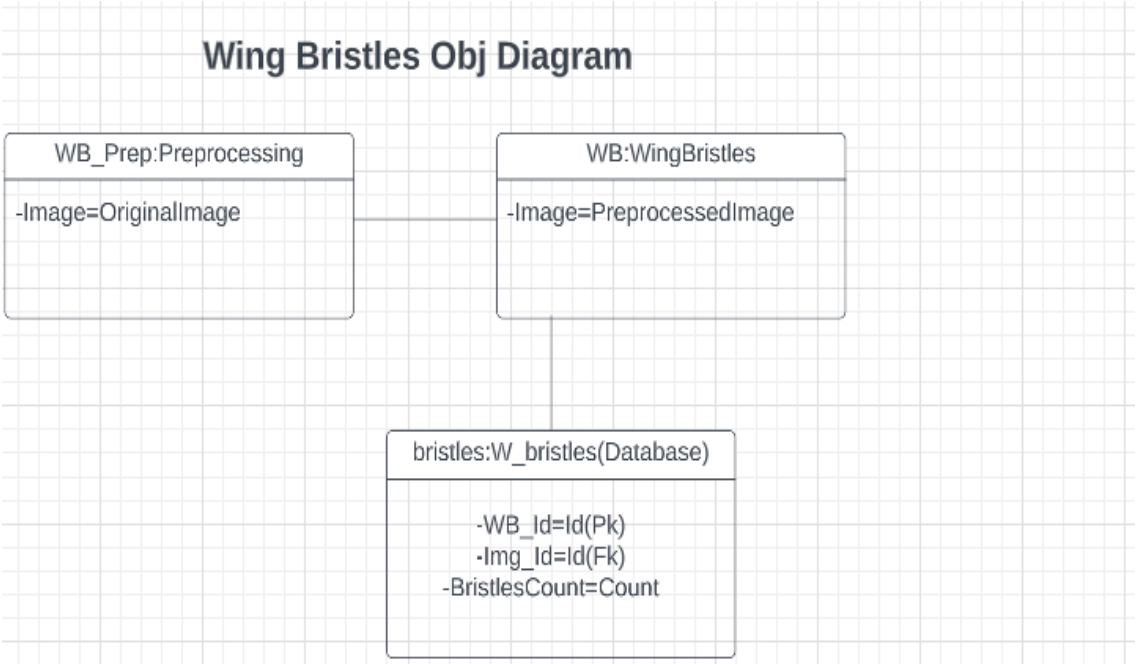


Figure 5.2.3.2: Wing Bristles Object Diagram

Description

Wing Brisltes contain 3 objects 2 of them are belongs to Preprocessing and wing bristles whereas 1 of them are of database.

Firstly, WD_Prep object will take the original image as an input and some preprocessing techniques will be applied to it.

Secondly, that preprocessed image in the object will be passed as parameter to second object name WB, were the brisltes extraction process will be carried out.

Lastly, the brisltes count extracted from the image will be stored in the database which will be passed in the database object name bristles which will take brisltes count and primary key and foreign keys which are at auto-increment.

5.2.4 Sequence Diagram

5.2.4.1 Wing Dimensions

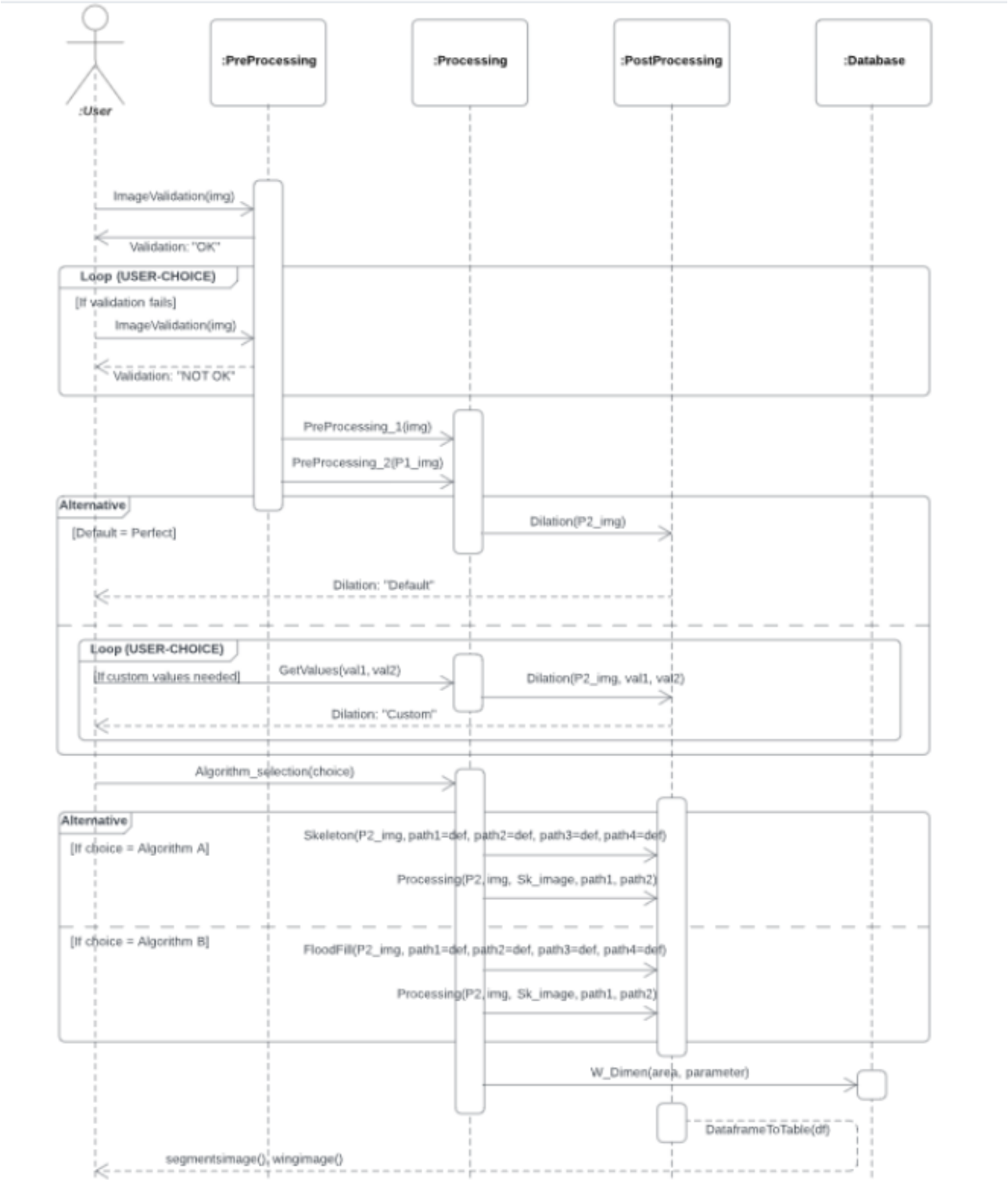


Figure 5.2.4.1: Wing Dimension Sequence Diagram

Description

A sequence diagram is used to identify a sequence of activities and response generated by a system on a particular operation. It shows how the operations are carried out in the system, which message are received and delivered. These are organized on the basis of time. These display the next activity carried out in a and which message is received in a sequential manner.

We have got 4 classes of Preprocessing, Processing, Postprocessing and Database.
The user will upload the image of wing and the first step will be to validate whether the uploaded image is of wing or not. If the image is of wing then the validation check is passes else a loop is run where the image will be returned to user and it wouldn't be able to enter.

If the image is passed from validation check then it is passed to preprocessing class where 2 preprocessing functions preprocessing 1 and preprocessing 2 are initiated. The output of those functions are passed to the processing class.

Now this is the phase where dilation need to be applied on wing so the system sets default value if the dilation looks correct to the user he proceedes to extract the feature else it sets the dilation values in which the dilation function is triggered along with the getvalues function which take our dilation values.

After this user selects the algorithm and it depends on the wing orientation, so if algorithm A is selected so the functions skeleton and processing are performed and the result are forwarded from processing lifeline to preprocessing lifeline.

If the algorithm B is selected then the functions floodfill and processing is performed on the image and the image is passes from processing lifeline to postprocessing lifeline.

Lastly results from postprocessing life are converted into Dataframe table and sended at user end and w_dimen function is instantiated to pass the features in to database.

5.2.4.2 Wing Bristles

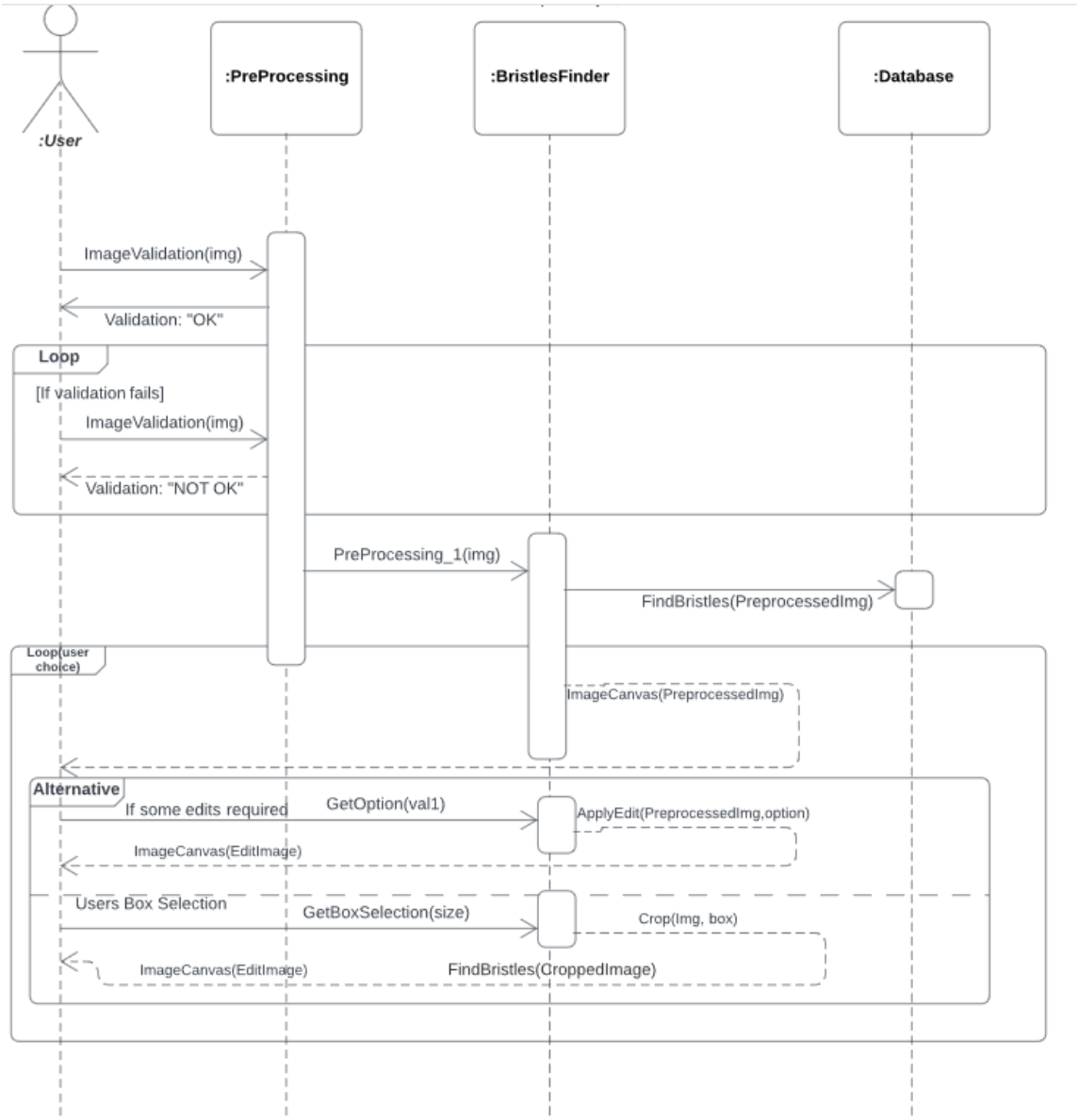


Figure 5.2.4.2: Wing Bristles Sequence Diagram

Description

We have got 3 classes of Preprocessing, BrislitesFinder and Database.
The user will upload the image of wing and the first step will be to validate whether the uploaded image is of wing or not. If the image is of wing then the validation check is passes else a loop is run where the image will be returned to user and it wouldn't be able to enter.

If the image is passed from validation check then it is passed to preprocessing class where 2 preprocessing functions preprocessing 1 and preprocessing 2 are initiated. The output of those functions are passed to the processing class.

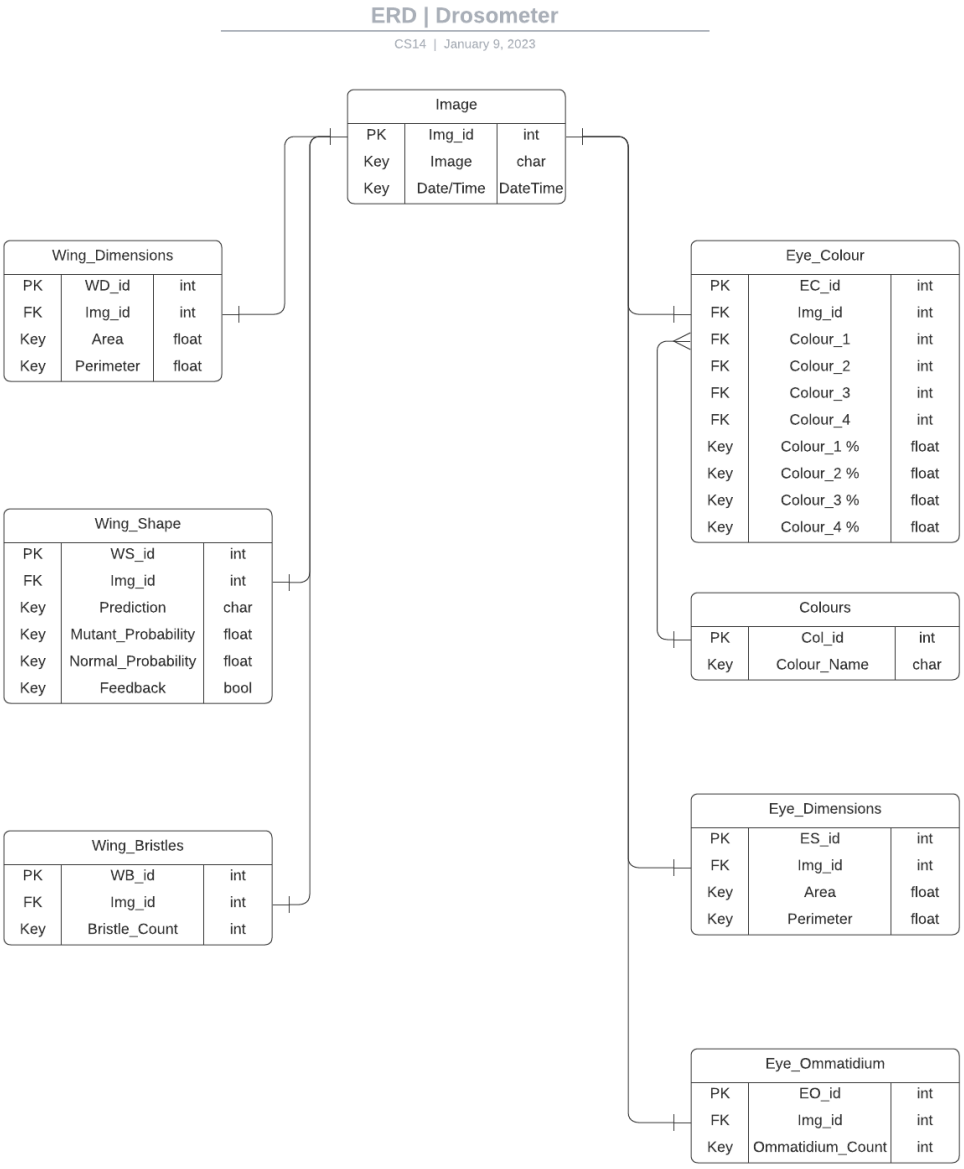
The the loop is run in the other process as the image from preprocessing class is forwarded to bristlesfinder where at first the whole preprocessed image is passed to findbristles function where the total bristles are calculated sended to database.

The image from bristlesfinder lifeline is revert back to the user, by using ImageCanvas function which displays the image on canvas at user end. We also have an option if user wants to edit the image such as zoom, tilting etc are also supported in canvas were the GetOption function is used and passed on to the FindBristles and then the ApplyEdit option is triggered.

The user select the box size for cropping the image where the GetBoxSelection is triggered and then the Crop function crops the image, FindBristles function is fired and provided with the cropped image which find the bristles and displays at user end.

These all function FIndBristles, GetOption, GetBoxSelection, Crop are part of BristlesFInder class.

5.3 ER Diagram



5.4 High-Fidelity Prototype

5.4.1 Selection of Module

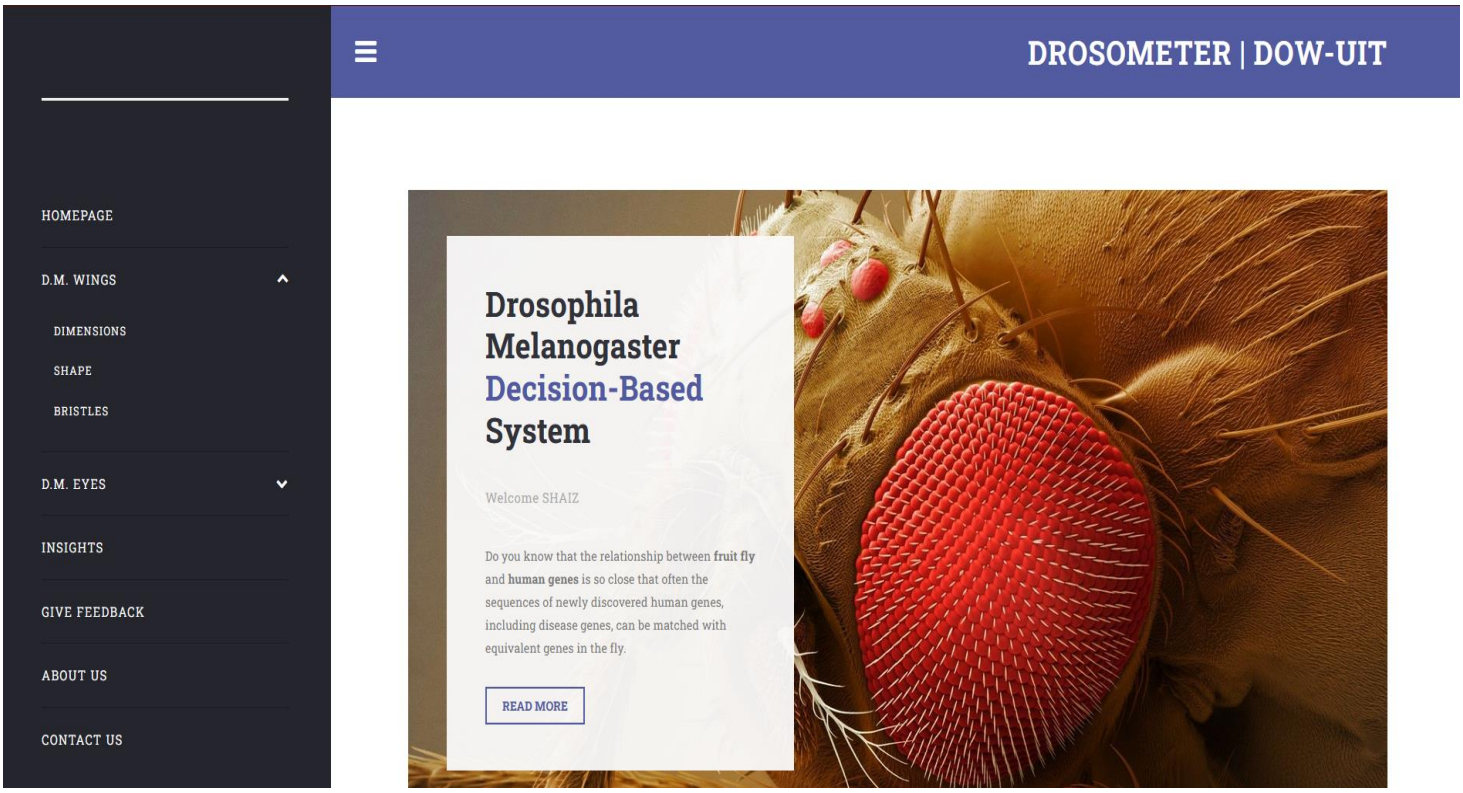


Figure 5.4.1: Landing page

Description

Fig 5.4.1 display the landing page of our website where user had the multiple options such as selecting which module to work on as in wing module we've got 3 sub-modules same for eye module, insights view etc.

5.4.2 Eye Dimensions

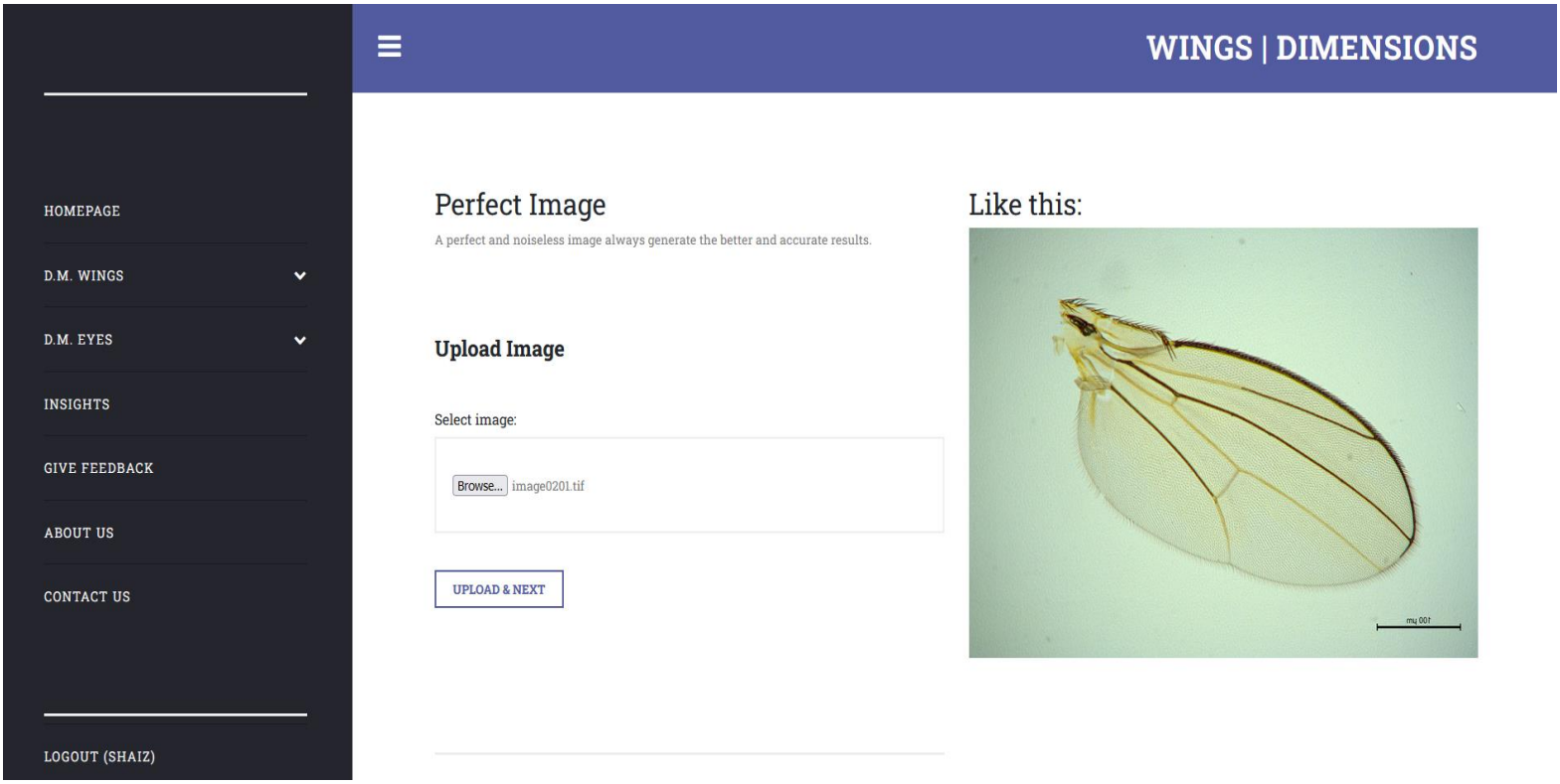


Figure 5.4.2.1: Uploading Image in Wing Dimensions

Description

In the wing dimension module we could upload the image as shown in the **fig 5.4.2** and the algorithm checks whether the image is of wing or not if its not it generates error as shown in **fig 5.4.3**.

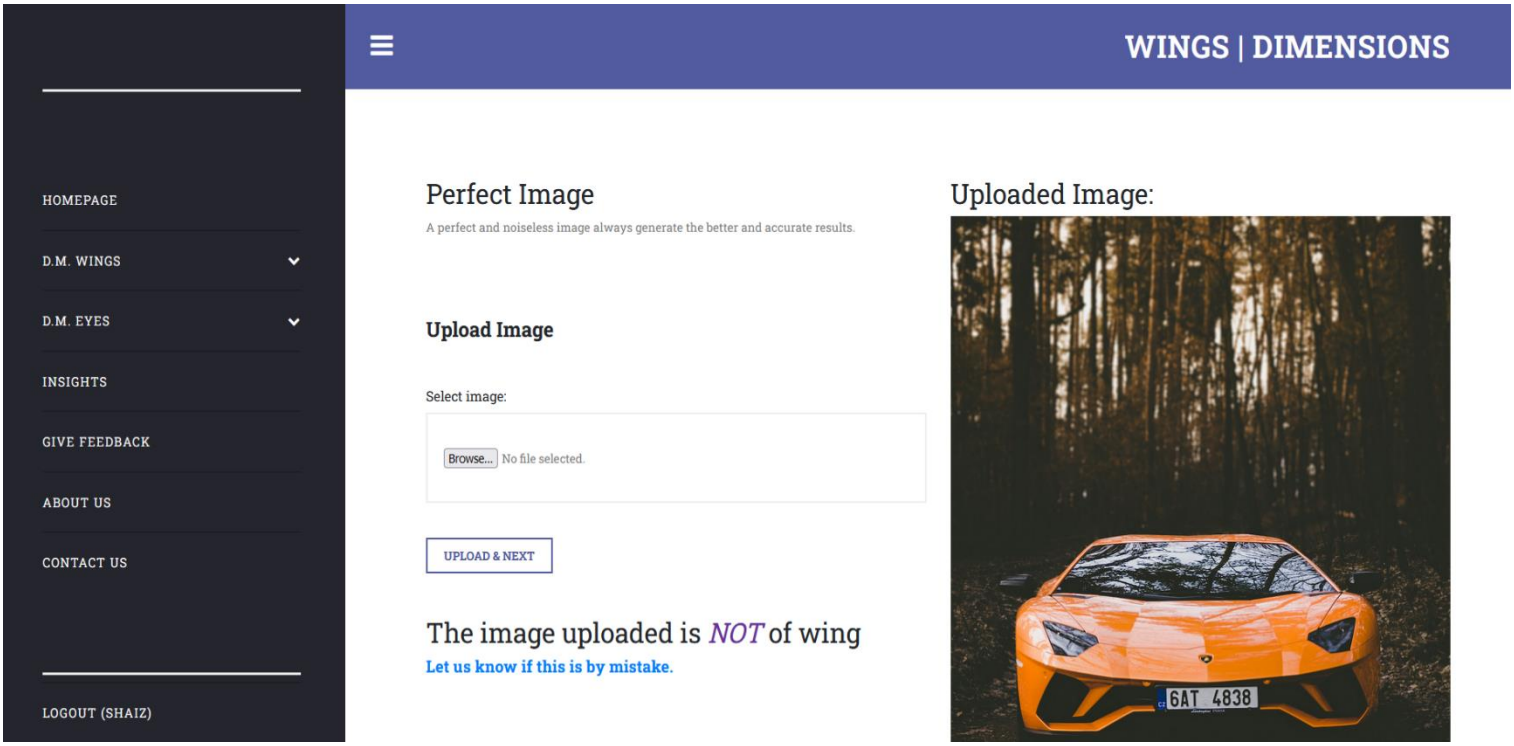


Figure 5.4.2.2: Uploading Wrong Image

Description

In this phase if anyone try to upload wrong image just like of car or something else the system MakkhiMeter will prompt error and wont let the image to enter in system.

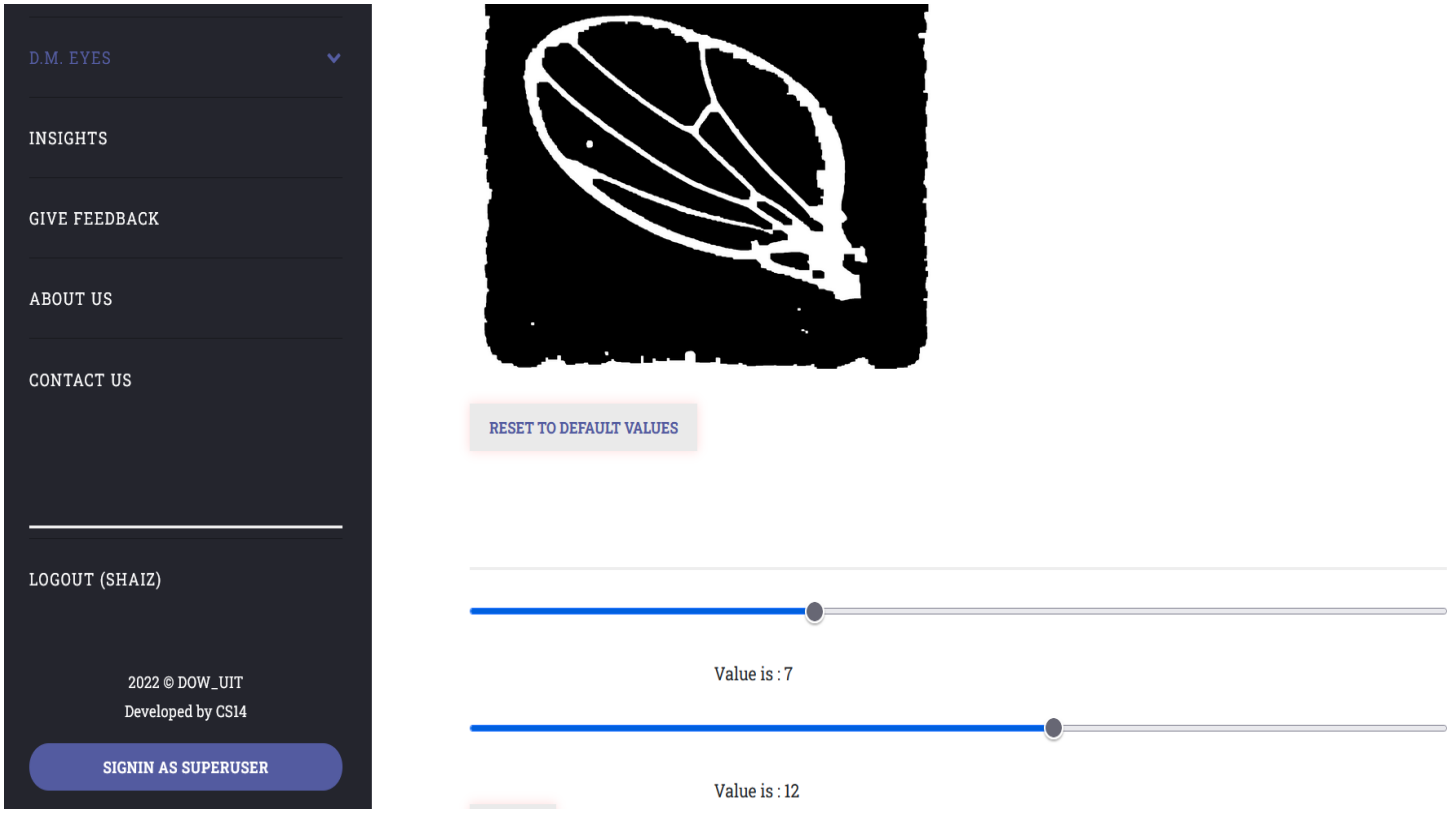
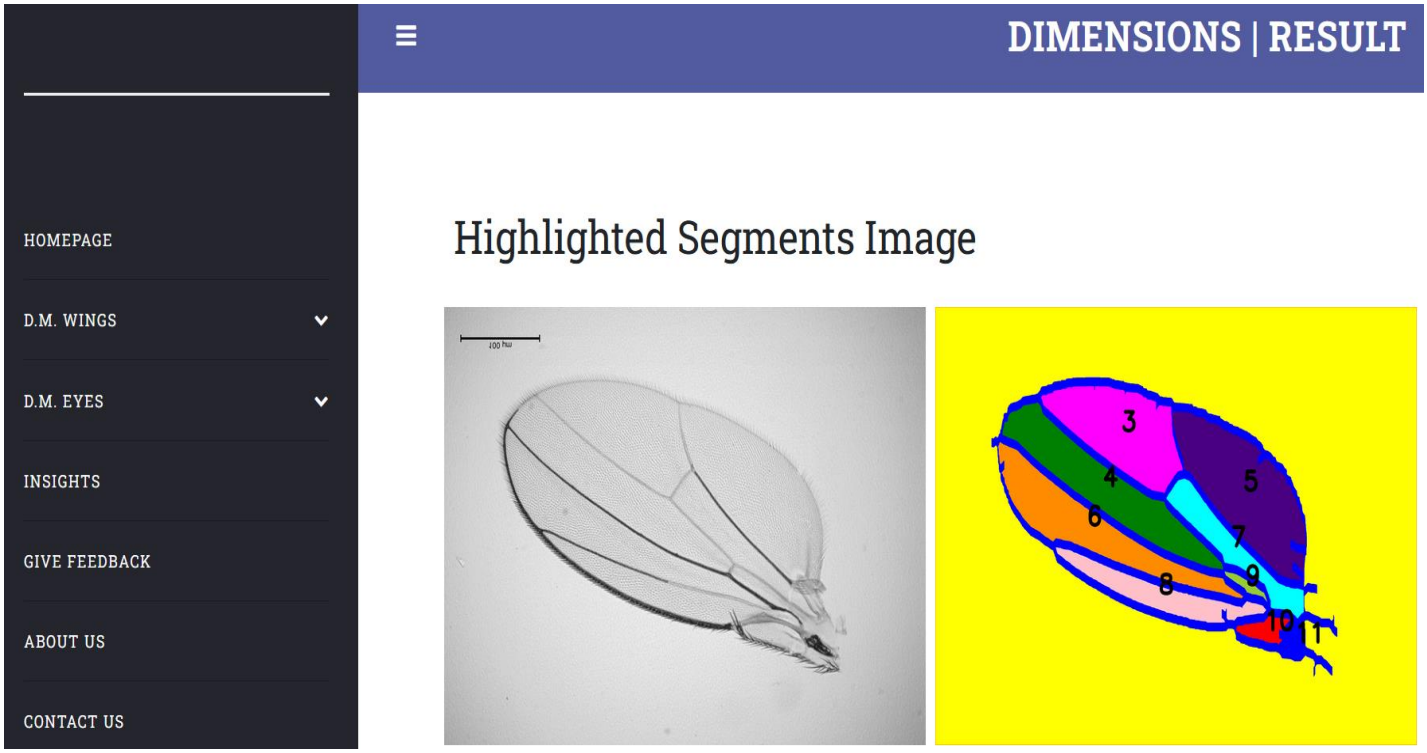


Figure 5.4.2.3: Extracting the binary image and setting dilation values

Description

If the uploaded image is correct then the image will be passed to the preprocessing and processing stage where the system display the binary image at user end, as the wing is dynamic in nature so we need the user input for the dilation values so here the bar of the values are provided and also the default values of dilation have also been applied to reduce the user effort.



Label	Area In Mm²	Perimeter In Mm
1	87494.6731234867	5818.7578795108
2	847851.573849879	3674.4454045013
3	48075.3026634383	518.1618324853
4	48738.7409200969	732.8234605825
5	60026.6343825666	674.281381324
6	44604.1162227603	808.9237929106
7	20871.4285714286	509.460393886
8	21909.4430992736	648.6002533373
9	1744.3099273608	140.826214309
10	4625.6658595642	178.2947775061
11	2010.6537530266	88.2263135698

Figure 5.4.2.4: Result of Segmentation of wing

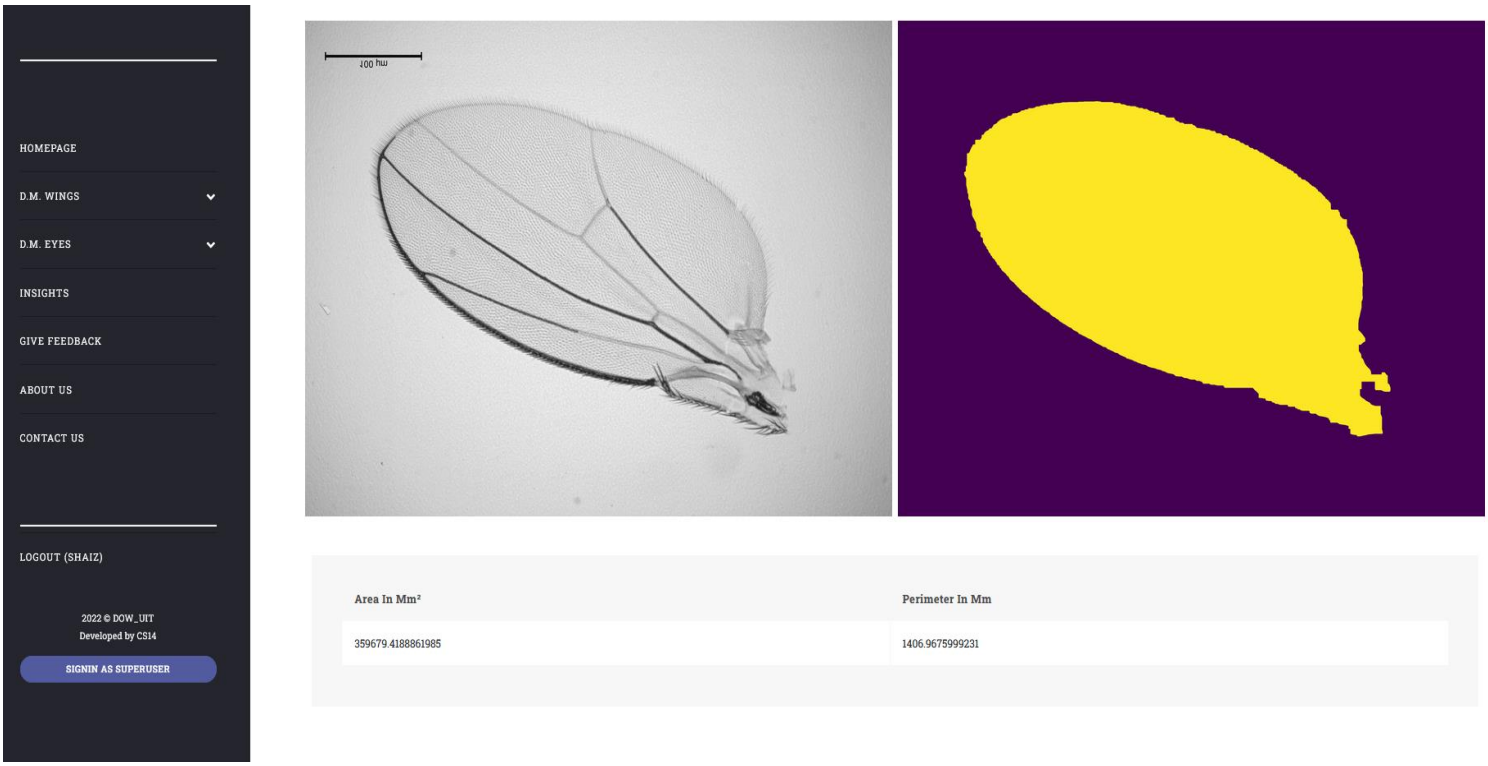


Figure 5.4.2.4: Complete Wing Segmentation

Description

In the **fig 5.4.2.3** and **5.4.2.4** is the result of the segmented wing where the wing 7 segments and complete segment is calculated and returned. Along with it the Area and Perimeter of the wing is also returned which are in mm^2 and mm**.**

5.4.3 Eye Bristles

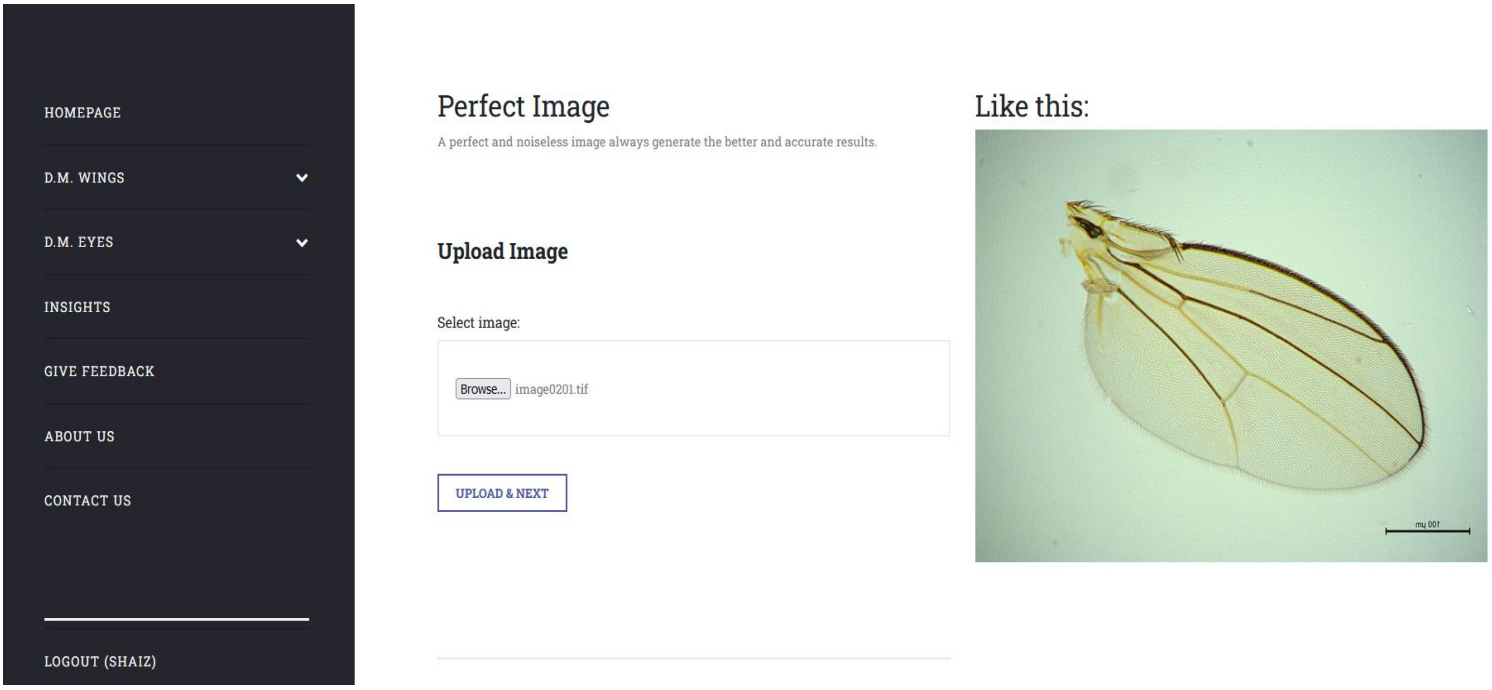


Figure 5.4.3.1: Uploading Image in Wing Bristles

Description

In the wing bristles module we could upload the image as shown in the **fig 5.4.3** and the algorithm checks whether the image is of wing or not if its not it generates error as shown in **fig 5.4.3** of wing dimensions.

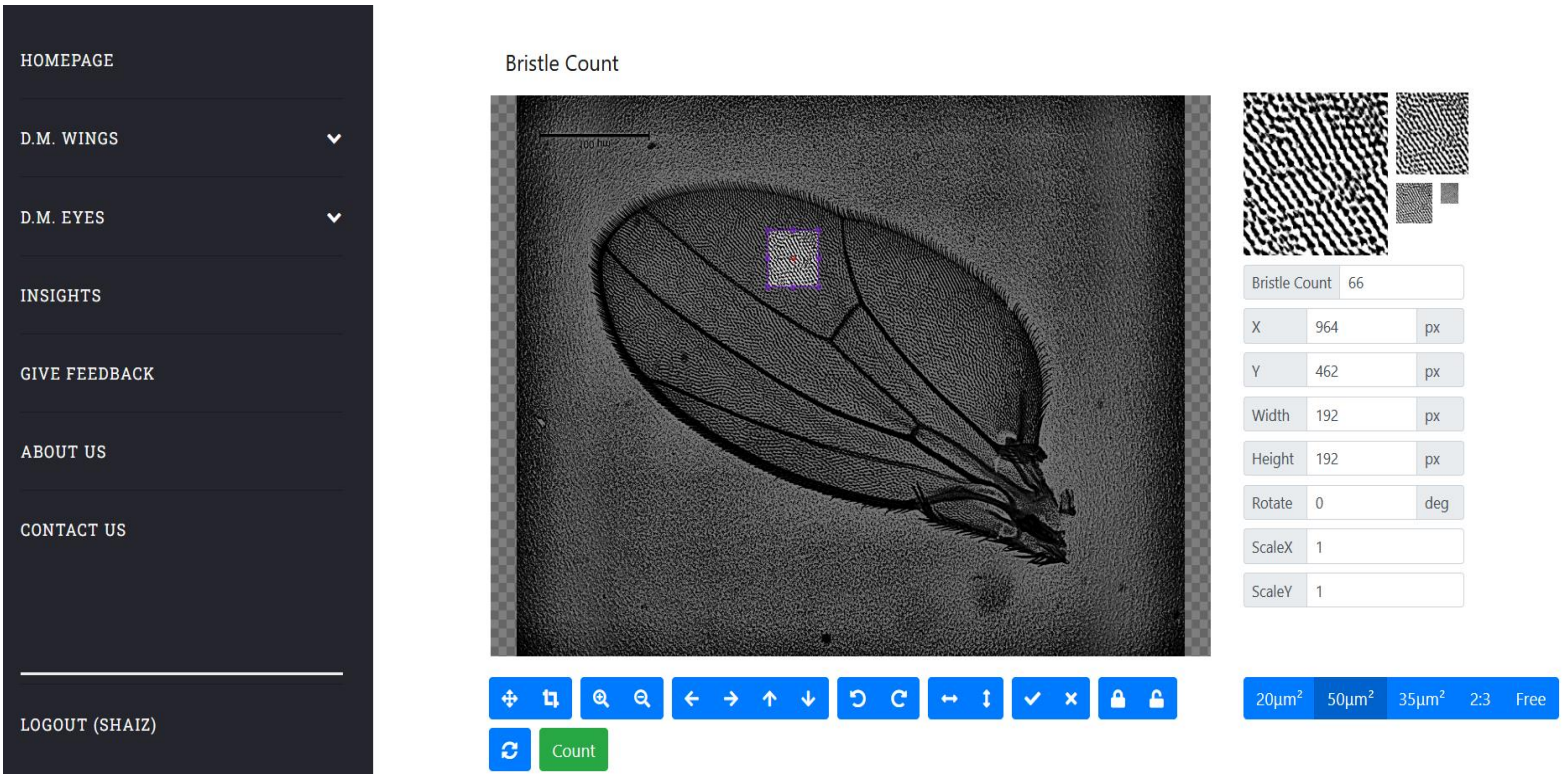


Figure 5.4.3.2: Bristles Extraction

Description

The input image is preprocessed and passed to canvas which displays the image. The its upon the user whether he want to do some edits in the image such as zoom, flip etc. Then user could select the crop box size which are provided of 20, 50 and 35 micrometer, also the free size is available. The box is dragged on the image canvas just showed in **fig 5.4.3.1** and count button is clicked and then the bristles count is shown in the bristle count box which are right now 66.

6. Algorithm analysis and complexity

This chapter's focuses on the algorithms that are used within our project, their significance, along with its pseudocode. It also contains the comparison of our selected algorithm with similar other algorithms. Each algorithm shows its best, average, and worst values in the context of time and space complexity.

6.1 Wing Dimensions

This module focuses on the extraction of the segments and complete segments of the wing. User only need to input the image and at the backend we have divided the our algorithm in to 3 parts:

1. Preprocess
2. Process
3. Postprocess

Our algorithm is more capable of handling the image when is too attached to the border and close to the border although there is certain tradeoff in complexity [1][2].

6.1.1 Preprocess

The uploaded image is first preprocessed where we equalize the histogram of the image by adding some white and black pixels in it. We also applied some filter such as median, savoula to defocus the bristles from the wing image.

Algorithm	complexity
1. Convert image into ubyte	O (MNC)
2. Convert image into gray	O (MN)
3. Histogram Equalizing	O (MN)
4. Applying Savoula Thresholding	O (MN)
5. Applying Median Filter	O (K)

Here M and N is the number of pixels, c is the channel of the image and k represents kernel size.

Step-1

The image is converted into ubyte which basically, means that we are converting image into unsigned byte. The reason is as we are using 2 libraries opencv and scikit-image so scikit used the ubyte image.

Step-2

The image is then converted into the gray channel which makes the computations fast and grayscale simplifies the algorithm.

Step-3

The image is taken in the environment where some dark pixels enters into the image and in the center of the image there are more white pixels so in order to work with these image we need to equalize the histogram by adding some black and white pixels. Just shown in fig 6.1.1.1

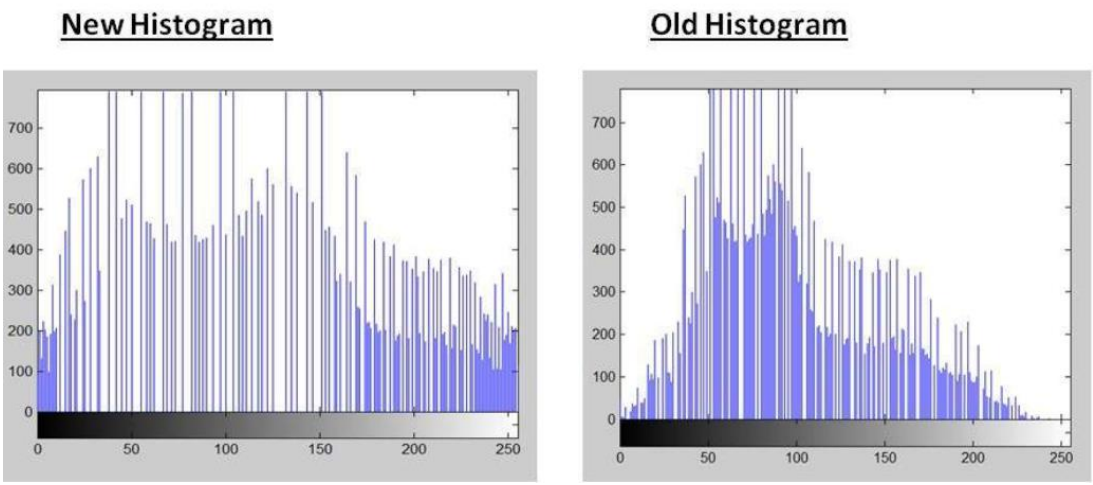


Figure 6.1.1.1 Histogram Equalizing

The main idea is to define a local function for each pixel (x_0, y_0) based on the neighboring pixels within a small window centered at this pixel: $I_0(x_0, y_0) = f(I(x, y))$.

The main idea of histogram equalizing methods is to re-assign the intensity values of pixels to make the intensity distribution curve uniform to utmost extent [9].

Step-4

The aim of using this algorithm is to defocus the background bristles in an image so that when using the thresholding the bristles won't come into the play. The method that performs preprocessing of local contents, the method is soft decision method [11].

Step-5

Each pixel will need to be added and removed to 2r+1 histograms over the course of processing the whole image, which causes the O(r) complexity. Intuitively, we can guess that we will need to accumulate each pixel at most a constant number of times [10]. The aim of this algorithm is to remove noises from image as some images were containing some noise in it due to the black pixels. As shown in fig 6.1.1.2.

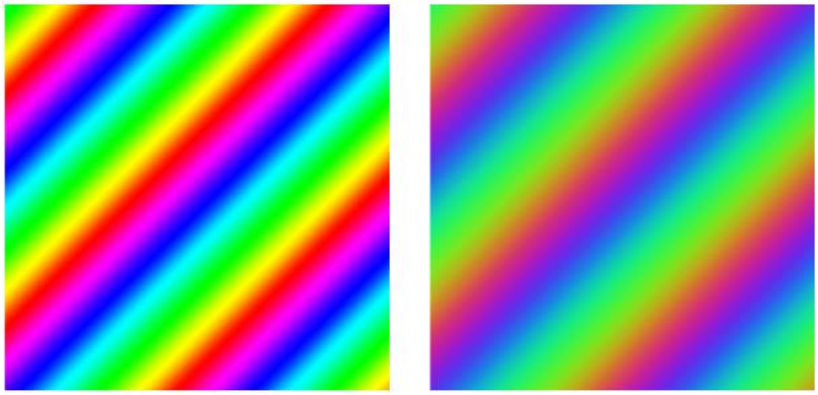


Figure 6.1.1.2: Median Filter

Conclusion

Overall we got C and K keeping these constants we have the dominating factor of MN where M and N are the number of pixels in an image. So overall time complexity of preprocessing image is an O(MN).

6.1.2 Process

The preprocessed image is then sended to the processing stage where the segments of the wing are extracted. Some of the algorithms are connectcomponents, watershed etc. This phase comprises of the 2 algorithms, one algorithm works when the image is centered where the second works when the image is either attached or too close to the border.

Algorithm	**complexity**
1. Thresholding	O (log(MN)
2. Dilation	O (5*MNK) => 5 O (MNK)
3. Erosion	O (4*MNK) => 4 O (MNK)
4. ConnectedComponnectStats	O (MNlogMN)
5. Filtering out unwanted areas	O (n)
6. Morphological (Close)	O (nLn(W))
7. Clear Border	O (MN)
8. Skeleton	O (MN)
9. Dilation	O (2*MNK)=> 2 O (MNK)
10. Watershed	O (N^2log(N^2))

Here M and N is the number of pixels, c is the channel of the image and k represents kernel size.

<u>Step-1</u>

Thresholding is applied on the preprocessed image where it is converted into black and white image, at every iterations pixels value are reduced hence the complexity goes into the log.

<u>Step-2</u>

In the Dilation process the wing image is dilated in order to complete the broken segment, here the 5 is a constant as the iteration is 5 so this processes is repeated 5 times on the kernel k.

Step-3

In the Erosion process the wing image is eroded in order minize the greater increase in the border of the wing, here the 4 is a constant as the iteration is 5 so this processes is repeated 5 times on the kernel k.

Step-4

Connected component relies on that of a pixel's neighbourhood: if this includes 4-neighbours C is said to be 4-connected. In the first scan a temporary label is assigned to each pixel in F based on the values of its neighbours already visited by the scan[12] as shown in fig 6.1.2.1.

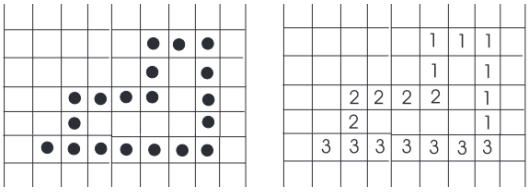


Figure 6.1.2.1: ConnectedComponents

Step-5

Some noise detected in the areas are filtered out using area as decision parameter.

Step-6

The closing operation dilates an image and then erodes the dilated image. Morphological closing is very useful method for filling small holes in an image while maintaining the shape and size of large holes and objects in the image shown in fig 6.1.2.2.

Figure 6.1.2.2: Mophological Close

Step-7

After this since there was some darkness in the corners of image so some noise has been part of it so the remaining noise which still persists is removed using clear border which operates in MN.

Step-8

Since the resulting image borders have got thickin and need to be reduced in order to improve the accuracy hence we used skeleton, its an iterative approch divided into sequential parallel process by removing the contour pixels iteratively till reach one pixel width[13] as shown in fig 6.1.2.3.

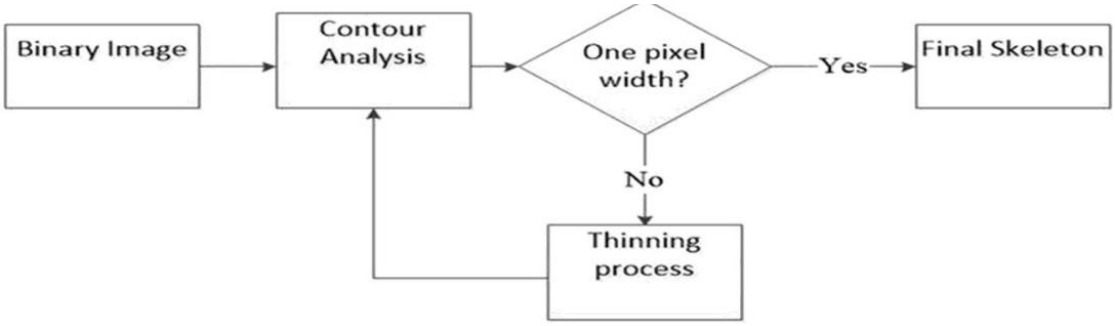


Figure 6.1.2.3: Flow of skeleton algorithm

Step-9

The dilation is applied to the skeleton image on the iteration of 2 and fixed kernel size.

Step-10

The watershed algorithm comprises of several steps such as distance transform of the image, medianblur, morphology, 2d-filters etc. The objective of watershed segmentation algorithm is to find all of the watershed lines (the highest gray level). The best way to explain watershed segmentation is the Immersion Approach: imagine that a hole is drilled in each minimum of the surface, and we flood water into different catchment basins from the holes[14]. If the water of different catchmen tbasins is likely to merge due to further immersion, a dam is built to prevent the merging. This flooding process will eventually reach a stage when only the top of dam (the watershed lines) is visible above the water line.

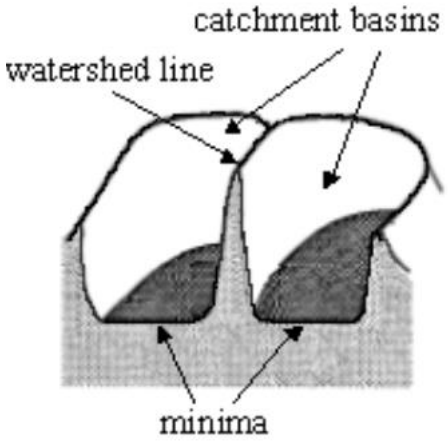


Figure 6.1.2.4: Watershed

<u>Conclusion</u>

For the algorithm A if we keep the kernel sizes and iterations contanst then the most dominating factor would be the O (MN) because the rate of change this shows is higher than the rate of change of log and ln.

Hence Algorithm A has O (MN).

6.1.3 Post-Process

The segmented image is forwarded to the the functions where labels are applied to each segments and the segments areas, perimeter are calculated and displayed.

	Algorithm A	**complexity**
1.	FindContours	O (MN)
2.	DrawContours	O (MN)
3.	ContourArea	O (N)
4.	RegionProps Table	O (MN)

<u>Step-1</u>

The segmented image is passed to the findcontours function where it finds the largest contour which is the complete wing.

<u>Step-2</u>

The contour founded is then drawn on the image.

<u>Step-3</u>

The drawn contour area is retrieved from the contour area.

<u>Step-4</u>

The segmented regions are passed to the regionprop to get the features of segmented regions.

Overall Conclusion

1. **Preprocess**

The preprocessing algorithm take O (MN) times to execute. The space complexity is O (MN).

2. **Process**

If Algorithm A and B is selected then the time complexity is O (MNlog(MN) + MN). The only thing which differentiates them is more computation is needed when operating on Algorithm B because more labels are involved in the image. The space complexity is O (MN).

3. **Post-Process**

The dominating factor is of the O (MN). The space complexity is O (MN).

Overall

Taking the values to infinity the complexity O (MNlog(MN) + MN) exceeds hence the overall space complexity of wing dimension is:

O (MN).

Time Complexity

- **Best-Case**

  In the best case scenario we will get all number of segments correctly and the features will also be retrieved correctly.

- **Average-Case**

  In the average case we get some number of labels which were not originated or we also get the dust as the labels or the labels of labels.

- **Worst-Case**

  For the worst case scenario we donot get labels and and the wing gets cleaned and the thing which only originates is the dust particles.

6.2 Wing Bristles

This module focuses on the extraction of the bristles from the wing. User only need to input the image and select the box size to crop the image, user also had the options to edit the image but these are the part of front-end and at the backend we have divided the our algorithm in to 2 parts:

1. Preprocess
2. Process

There were no similar algorithms available but the edge detection algorithm have been implemented[5] make this more complex and it extracts all bristles again and again which is slow process.

6.2.1 Preprocess

The uploaded image is first preprocessed where we equalize the histogram of the image by adding some white and black pixels in it. We also applied some filter such as median, savoula to defocus the bristles from the wing image.

Algorithm	**complexity**
1. Convert image into ubyte	O (MNC)
2. Convert image into gray	O (MN)
3. Histogram Equalizing	O (MN)
4. Sharpening Filter	O (MN)

Here M and N is the number of pixels, c is the channel of the image and k represents kernel size.

<u>Step-1</u>

The image is converted into ubyte which basically, means that we are converting image into unsigned byte. The reason is as we are using 2 libraries opencv and scikit-image so scikit used the ubyte image.

<u>Step-2</u>

The image is then converted into the gray channel which makes the computations fast and grayscale simplifies the algorithm.

Step-3

The image is taken in the environment where some dark pixels enters into the image and in the center of the image there are more white pixels so in order to work with these image we need to equalize the histogram by adding some black and white pixels.

Step-4

For the sharpening of the image we have used the unsharp masking where

sharpened image = original image – blurred image

we subtract a blurred version of an image produces sharpening output image.

Conclusion

The overall time complexity is O (MN). The space complexity is O (MN).

6.2.2 Process

Algorithm	complexity
1. Thresholding	O (log(MN)
2. FindContours	O (MN)

Step-1

Thresholding is applied on the preprocessed image where it is converted into black and white image, at every iterations pixels value are reduced hence the complexity goes into the log.

Step-2

The segmented image is passed to the findcontours function where it finds the largest contour which is the complete wing.

Conclusion

The overall time complexity is O (log(MN). The space complexity is O (MN).

Overall Conclusion

1. **Preprocess**

Preprocess works in the complexity of O (MN).

2. **Process**

Processing works in the O (log(MN)).

Overall the complexity is of O (log(MN) and space complexity is O (MN).

Time Complexity

- **Best-Case**

  In the best case scenario we will get all number of bristles correctly.

- **Average-Case**

  In the average case the if box is dragged on the union of the image and outside of the image then some dust particles might play some role in the bristle count.

- **Worst-Case**

  For the worst case scenario the box is dragged only on the dust section which is outside the wing.

7. Implementation

Give code details (not a complete listing, but description of key parts). Discuss the most important/interesting aspects. It probably won't be possible to discuss everything- give a rationale for what you do. Code shall not be more than 3-5 pages. Use appropriate code writing standards, draw operational diagram, component diagram and deployment diagram. Show only two to three technical interfaces that represents the core project functionality with explanation (You may use POC interfaces). Draw state transition diagram of project interface (input, output, and processes).

7.1 Important code/Interesting aspects

7.1.1 Code for colouring the segments of the wing

The code displayed above demonstrates the primary processing involved in segmenting the wing. Prior to this, some basic operations were performed to enhance the quality of the image as required.

```
class WD_Procesing:
    def __init__(self, preprocess_img=None, e_img=None, proc=None):
        self.preprocess_img = preprocess_img
        self.e_img = e_img
        self.proc = proc

    def Dilation(self, val1=7, val2=12):
        _, thresh = cv2.threshold(self.preprocess_img, 80, 255,
cv2.THRESH_BINARY_INV)

        kernel = np.ones((int(val1), int(val2)), np.uint8)
        d_im = cv2.dilate(thresh, kernel, iterations=5)
        self.e_img = cv2.erode(d_im, kernel, iterations=4)
        return self.e_img
```

The dilation function in this code takes kernel values from a user-end slider to complete the vein structure.

```
    def Skelatonize(self, path1, path2, outimg, outimg2):
        _, thresh2 = cv2.threshold(self.e_img, 75, 255,
cv2.THRESH_BINARY)
        nlabels, labels, stats, centroids =
cv2.connectedComponentsWithStats(thresh2, None, None, None, 8,
cv2.CV_32S)
```

```
        # get CC_STAT_AREA component as stats[label, COLUMN]
        areas = stats[1:, cv2.CC_STAT_AREA]

        result = np.zeros((labels.shape), np.uint8)

        for i in range(0, nlabels - 1):
            if areas[i] >= 11000:  # keep
                result[labels == i + 1] = 255
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (25, 25))
        morph = cv2.morphologyEx(result, cv2.MORPH_CLOSE, kernel)
        mask = 0 - morph
        edge_touching_removed = clear_border(mask)

        skeleton = skeletonize(edge_touching_removed)
        plt.imsave(path1, skeleton, cmap='gray')
        skeleton = cv2.imread(path1, 0)
        kernel = np.ones((20, 2), np.uint8)
        dil = cv2.dilate(skeleton, kernel, iterations=2)

        plt.imsave(path2, dil, cmap='gray')

        img = cv2.imread(path2)
        self.proc = img
        result = self.Processing(edge_touching_removed, outimg,
outimg2)
outimg2)
        return result[0], result[1]

    def Processing(self, edge_touching_removed, outimg, outimg2,
flag=False):
        cells = self.proc[:, :, 0]  # Blue channel. Image equivalent
to grey image.

        adj_img = np.zeros(cells.shape, cells.dtype)

        # Defining alpha and beta:
        alpha = 1.9  # Contrast Control [1.0-3.0]
        beta = 0  # Brightness Control [0-100]

        # Scaling and converting the image contrast and brightness
        adj_img = cv2.convertScaleAbs(cells, alpha=alpha, beta=beta)

        # Blur the image for better edge detection
        img_blur = cv2.medianBlur(adj_img, 3)
        # sharpness apply on images
        kernel = np.array([[0, -1, 0],
                           [-1, 5, -1],
                           [0, -1, 0]])
        image_sharp = cv2.filter2D(src=img_blur, ddepth=-1,
kernel=kernel)


        ret, thresh = cv2.threshold(image_sharp, 90, 255,
```

```
cv2.THRESH_BINARY_INV)


        kernel = np.ones((3, 3), np.uint8)
        opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel,
iterations=1)

        opening = clear_border(opening)


        sure_bg = cv2.dilate(opening, kernel, iterations=3)


        dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2,
0)



        ret2, sure_fg = cv2.threshold(dist_transform, 0.001 *
dist_transform.max(), 255, 90)

        sure_fg = np.uint8(sure_fg)
        # cv2.imwrite('opening.tif',sure_bg)

        unknown = cv2.subtract(sure_bg, sure_fg)

        ret3, markers = cv2.connectedComponents(thresh)

        markers = markers + 1

        markers = cv2.watershed(self.proc, markers)

        self.proc[markers == -1] = [0, 255, 255]

        if flag:
            result =
WD_PostProcessing.RegionProp_FloodFill(self.proc,
edge_touching_removed, outimg, outimg2, ret3,
markers,cells)
            df = result[0]
            data = result[1]
            return df, data
        else:
            result =
WD_PostProcessing.RegionProp_Skeleton(self.proc,
edge_touching_removed, outimg, outimg2, ret3,
markers,cells)
            df = result[0]
            data = result[1]
            return df, data
```

The Skelatonize function takes in an image, performs thresholding and morphological operations to extract the skeleton of the sample, and then passes the resulting image to the Processing function for further processing. The Processing function performs additional thresholding and morphological operations and makes a dataframe of labels and their pixel values with the help of RegionProp.

7.1.2 Wing Bristles Counter

To find the number of bristles, we've used Cropper.js library. The Cropper.js library provides an image cropper functionality for web applications.

The code creates a Cropper object with options that determine the behavior and appearance of the image cropper. It then attaches event listeners to the object that handle events such as 'ready,' 'cropstart,' 'cropmove,' 'cropend,' 'crop,' and 'zoom.' It also sets up buttons and options for the image cropper, such as toggles for aspect ratio, preview, and guides.

7.1.2.1 Number of bristles on a user-defined size box

The essential chuck is extracted from a lengthy codebase and is presented in a section where a user has selected and clicked on the "count" button, resulting in a cropped canvas of the image in the box. Using computer vision libraries, we apply a threshold to the cropped image and detect contours.

```
      case 'getCroppedCanvas':
        if (result) {
          // Bootstrap's Modal
          $('#getCroppedCanvasModal').modal().find('.modal-
body').html(result);

          if (!download.disabled) {
            download.download = uploadedImageName;
            download.href = result.toDataURL(uploadedImageType);
          }
          var href=download.href ;
          var a=href.split(';');
          var image = new Image();
          image.src = href;
          image.onload = function(){
            console.log(image);
```

```
            let src = cv.imread(image);
            cv.cvtColor(src, src, cv.COLOR_RGB2GRAY, 0);
            cv.threshold(src, src, 100, 255, cv.THRESH_OTSU);
            let contours = new cv.MatVector();
            let hierarchy = new cv.Mat();
            cv.findContours(src, contours, hierarchy, cv.RETR_LIST,
cv.CHAIN_APPROX_SIMPLE);
            var xcnts=[];
            for (let i = 0; i < contours.size(); i++) {

              const cnt = contours.get(i)
              let area = cv.contourArea(cnt, false)

              xcnts.push(area)
            }
            var x=xcnts.length;
            console.log(x)
            document.getElementById("Bristle").value=x
            };
        }

        break;

      case 'destroy':
        cropper = null;

        if (uploadedImageURL) {
          URL.revokeObjectURL(uploadedImageURL);
          uploadedImageURL = '';
          image.src = originalImageURL;
        }

        break;
    }

    if (typeof result === 'object' && result !== cropper && input) {
      try {
        input.value = JSON.stringify(result);
      } catch (e) {
        console.log(e.message);
      }
    }
  }
};
```

7.1.2.2 Overall Bristles on the wing

This method processes an image of a wing by first converting it to grayscale and applying a threshold to create a binary image. The contours in this binary image are then identified using the OpenCV library's "findContours" function.

```
class WB_Processing:
    def __init__(self, prep=None):
        self.prep = prep

    def overallbristles(self):
        self.prep = cv2.imread(self.prep, 0)
        th, threshed = cv2.threshold(self.prep, 100, 255,
cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)


        cnts = cv2.findContours(threshed, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)[-2]


        xcnts = []
        for cnt in cnts:
            if cv2.contourArea(cnt):
                xcnts.append(cnt)

        return len(xcnts)
```

The contours are then filtered based on their area and only those that meet the specified criteria are stored in a list. Finally, the number of elements in this list is returned as the total number of bristles present on the wing.

7.1.3 Eye Colours Detection

The eye color detection system consists of two classes: "Color" and "Eye_Col". The "Color" class includes a comprehensive list of all possible eye colors, with around 150-200 color options along with their corresponding hex values and names. Additionally, this class contains functions such as "rgb_to_hex" and "names_to_hex" for color conversions.

```
class eye_col:
    def __init__(self):
        self.cc = color()
        pass

    def get_image(self, image_path):
        image = cv2.imread(image_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        return image

    def interpolating_image(self, image):
        modified_image = cv2.resize(image, (600, 400),
interpolation=cv2.INTER_AREA)
        modified_image =
modified_image.reshape(modified_image.shape[0] *
modified_image.shape[1], 3)
        return modified_image

    def KModel(self, image):
        # we sett the max value of 255 in clusters
        clf = KMeans(n_clusters=4)
        labels = clf.fit_predict(image)
        return labels, clf

    def RGB2HEX(self, color):
        return "#{:02x}{:02x}{:02x}".format(int(color[0]),
int(color[1]), int(color[2]))

    def hex2name(self, c):
        h_color = '#{:02x}{:02x}{:02x}'.format(int(c[0]), int(c[1]),
int(c[2]))
        try:
            nm = self.cc.hex_to_name(h_color, spec=u'css3')
        except ValueError as v_error:
            rms_lst = []
            for img_clr, img_hex in
self.cc.CSS3_NAMES_TO_HEX.items():
                cur_clr = self.cc.hex_to_rgb(img_hex)
                rmse = np.sqrt(mean_squared_error(c, cur_clr))
                rms_lst.append(rmse)

            closest_color = rms_lst.index(min(rms_lst))

            nm =
list(self.cc.CSS3_NAMES_TO_HEX.items())[closest_color][0]
        return nm
```

The "Eye_Col" class utilizes k-means clustering with four clusters to analyze an input image of an eye after basic preprocessing steps such as reshaping and resizing. The clustering algorithm groups pixels in the image with similar colors into one of the four clusters.

```
    def final(self, img):
        image1 = self.get_image(img)
        image = self.interpolating_image(image1)
        model = []
        values = []
        new = {}

        for i in range(10):
            labels, clf = self.KModel(image)
            if i == 0:
                model.append(clf)

            values.append(labels)

        zero = 0
        one = 0
        two = 0
        three = 0

        for i in range(10):
            zero += np.count_nonzero(values[i] == 0)
            one += np.count_nonzero(values[i] == 1)
            two += np.count_nonzero(values[i] == 2)
            three += np.count_nonzero(values[i] == 3)

        new[0] = int(zero / 4)
        new[1] = int(one / 4)
        new[2] = int(two / 4)
        new[3] = int(three / 4)
        return new, model

    def run(self, img):
        out = self.final(img)
        colors = {}
        counts = out[0]
        clf = out[1][0]
        center_colors = clf.cluster_centers_
        ordered_colors = [center_colors[i] for i in counts.keys()]

        hex_colors = [self.RGB2HEX(ordered_colors[i]) for i in
counts.keys()]
        rgb_colors = [ordered_colors[i] for i in counts.keys()]
        lbl_color = [self.hex2name(ordered_colors[i]) for i in
counts.keys()]
        lst = list(counts.values())


        return lbl_color, lst, hex_colors
```

The last two functions performs K-means clustering 10 times on the input image and returns the average count of each cluster and then calculates the center colors, ordered colors, and label colors for each cluster, and returns them.

7.2 Component Diagram

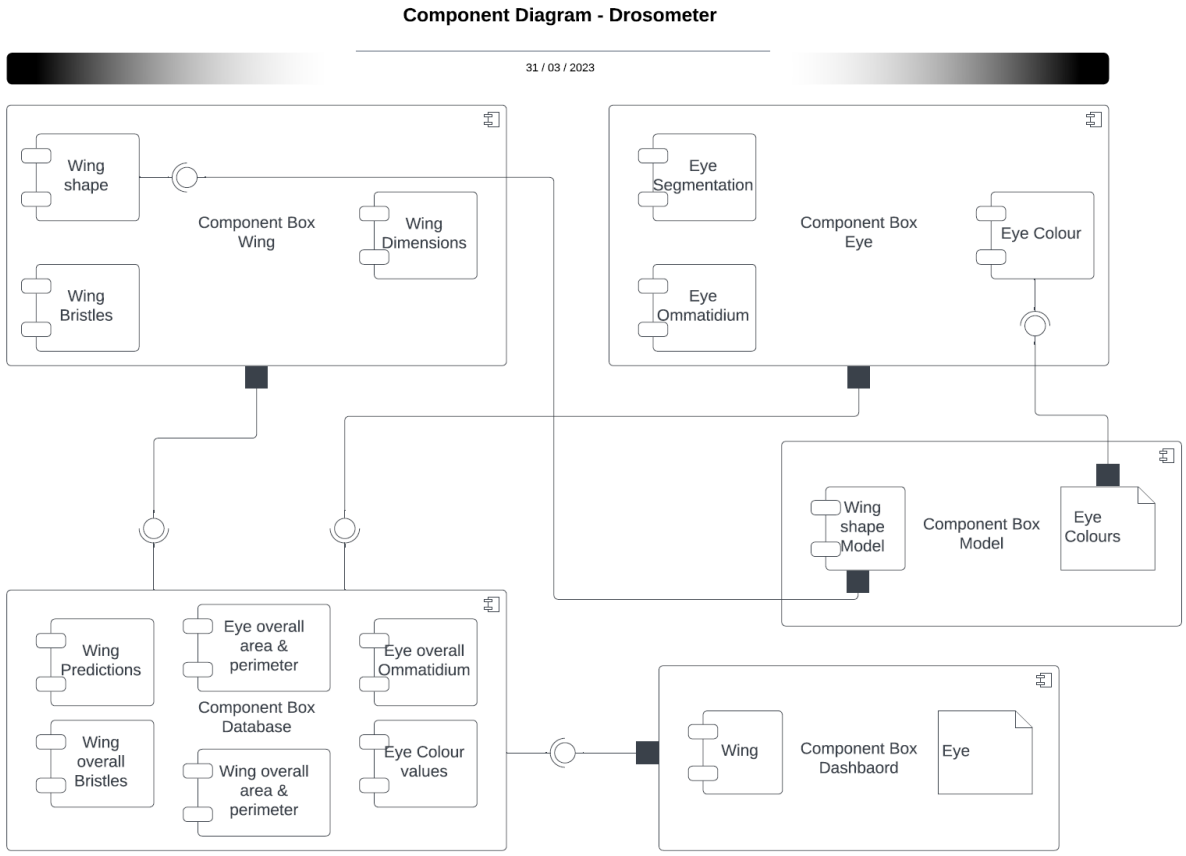


Figure 7.2 - Component Diagram of MakkhiMeter

In **Figure 7.2**, there is a box consisting of four distinct components. The process for the two main components, namely "wing" and "eye", is identical in that an uploaded image is analyzed, and the resulting data is saved in a database for display on the dashboard.

7.3 Operational Diagram

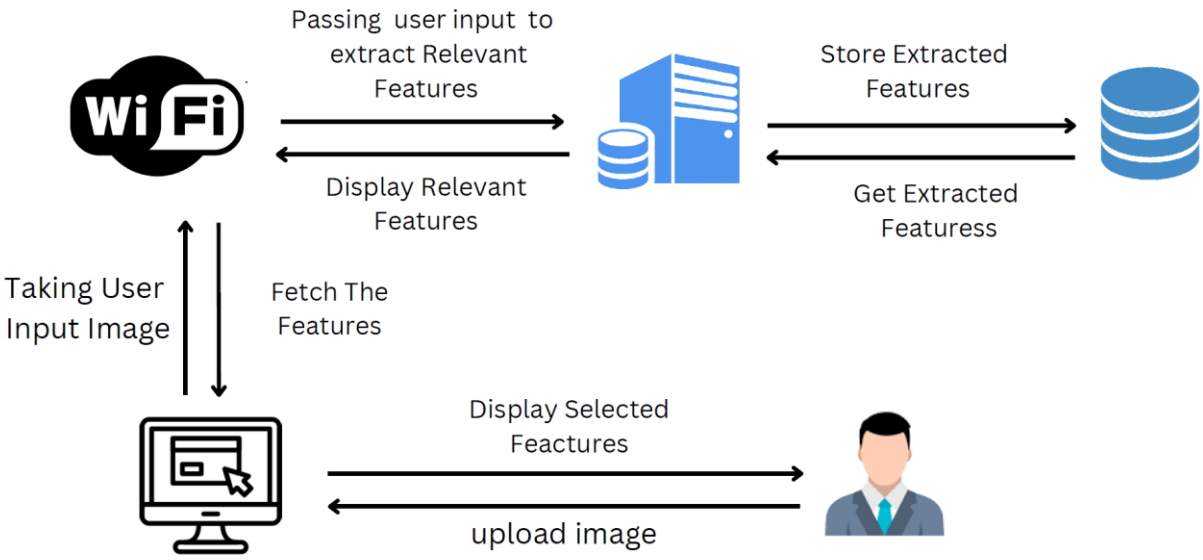


Figure 7.3 - Operational Diagram of MakkhiMeter

The operational diagram of the system, as depicted in **Figure 7.3**, illustrates the connectivity of the various components and outlines the sequence of operations to be performed. The diagram highlights the following steps: a user uploads an image on the web application, the application processes the image, and if the results do not already exist in the database, the data is stored in the database. Finally, the results are displayed on the web application for the user to view.

7.4 Deployment Diagram

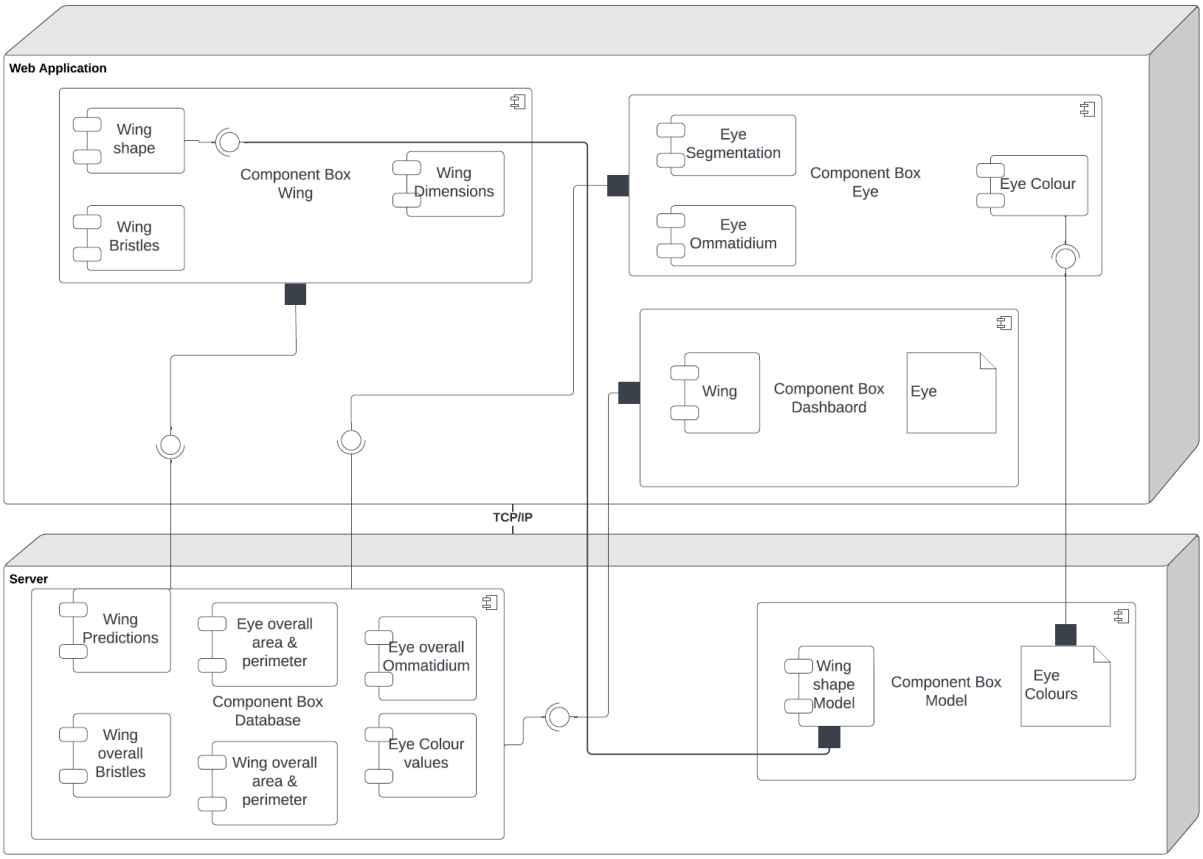


Figure 7.4 - Deployment Diagram of MakkhiMeter

Figure 7.4 depicts a deployment diagram comprising two blocks. The upper block represents our web application, which includes all of its modules and a dashboard for insights. The Web Application backend is developed using Python programming language and its Web Framework of Django & for the Front-End we have used the Html, Css, Javascript & Bootstrap.

The lower block represents the server where all the scripts are executed, and the database. The Application will be deployed on the DUHS Web Server using Microsoft Server 2019. For the Database we have used the SQLITE Database the reason behind is the amount of the data which will be stored in our database which will be less complex and less in size and since we are storing some numericals data hence we donot need to deploy the large databases to handle.

Additionally, there is another component box on the server that contains a deep learning model for wing shape and a file that assists in identifying colors in the eye.

These all will form a 3-tier architecture application and user will be able to access this application on DUHS server.
For the configuration on DUHS server so at first we have to setup the environment on the Microsoft Server 2019. Install the required libraries and adjust the application settings so that it could allow the traffic to be flown to our application.

After setting the libraries we will deploy the database on it and the required files of the web application project on server.

Then we will get the ip address to access the application and that ip will be masked with the domain name of the DUHS.

7.5 Technical Interfaces

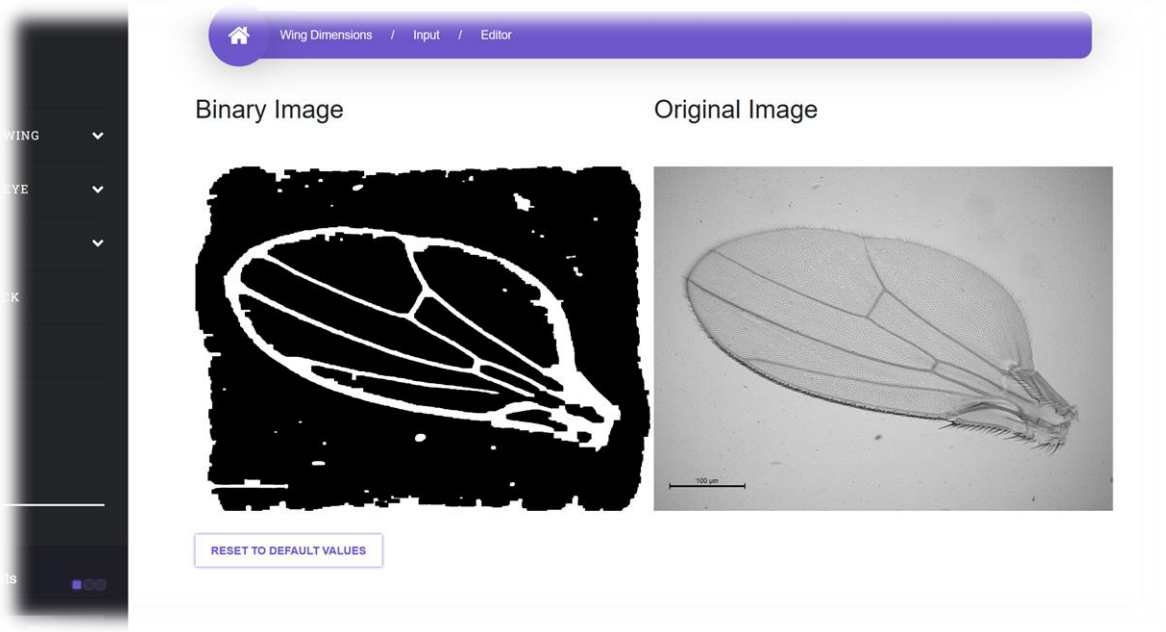


Figure 7.5. 1 - Wing Dimensions (User interaction)

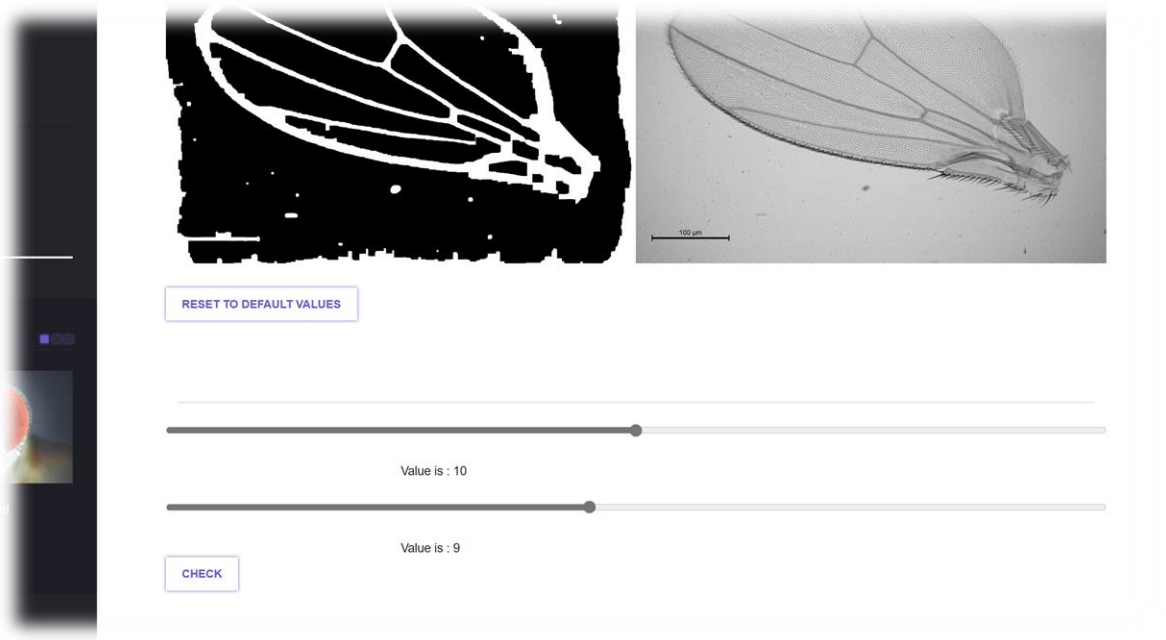


Figure 7.5. 2 - Wing Dimensions (User interaction - Slider)

The **figures 7.5.1** and **7.5.2** demonstrate the sole user interaction within the overall project, where the user uploads an image, adjust the binary image and then the results are displayed. To enhance the vien structure of the wing, we utilized two sliders to adjust the dilation process. This was necessary because certain veins were very faint and thin, almost indistinguishable from the intensity of bristles. The two sliders have a range of 2-20 and allow for horizontal and vertical completion of the vein.

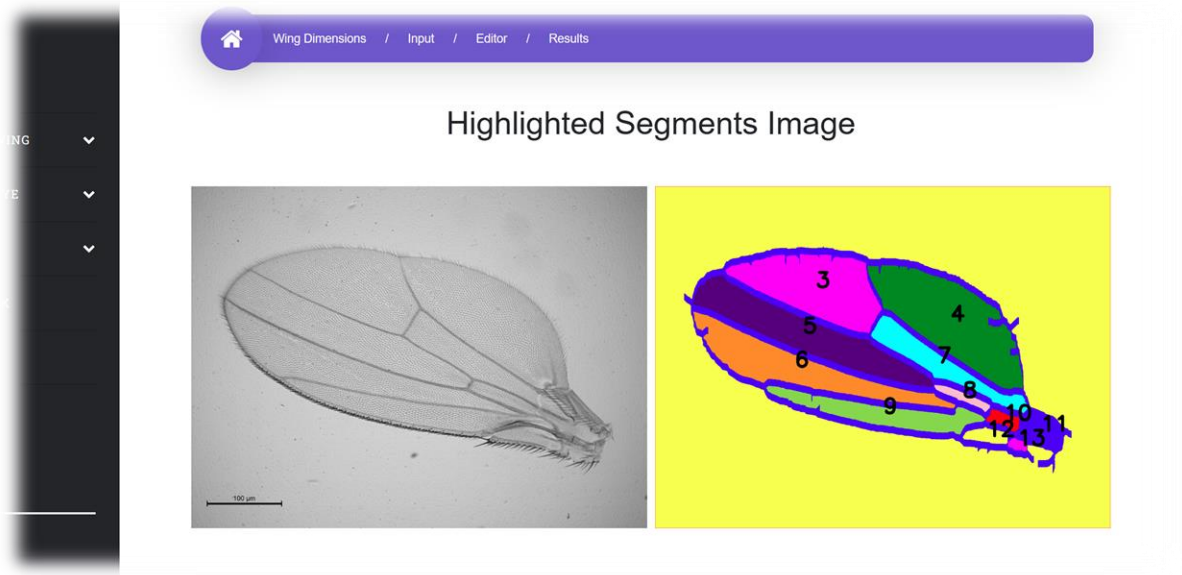


Figure 7.5. 3 - Seven Segments of the wing

In **figure 7.5.3**, the basic structure of the wing has been divided into seven segments and each region has been labeled and coloured accordingly. The corresponding area and perimeter of each region is then tabulated.

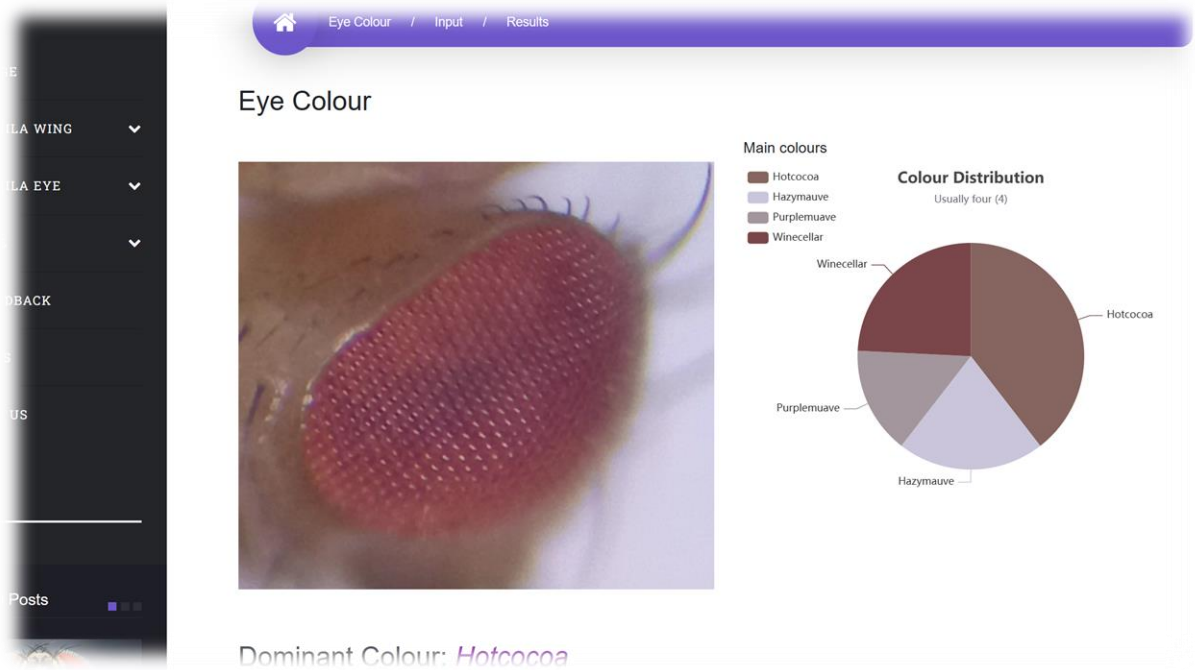


Figure 7.5. 4 - Eye Colour Detection

In **Figure 7.5.4**, the four most prominent colours in the eye are displayed. This is a crucial aspect of the project, as it involves identifying the number of colors present in the eye and generating a pie chart that illustrates the percentages of those four colours.

Additionally, hex values, pixel values, and other relevant data is shown in tabulated form.

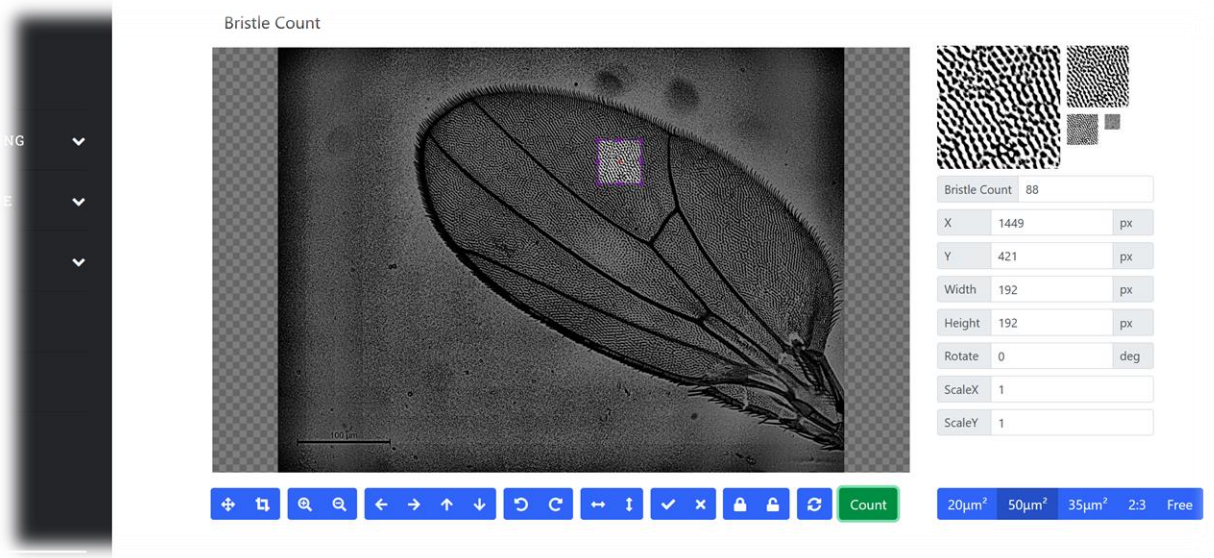


Figure 7.5. 5 - Bristles Counter

In **Figure 7.5.5**, the bristles counter screen is displayed which shows an enhanced and increased contrast version of the uploaded image. This helps in making the bristles on the wing visible and clear. The screen has several basic operations such as zooming, rotating, and resetting. Additionally, there are five options for different box sizes located on the bottom left-hand side of the screen, which can be moved all over the image as per requirement.

By clicking the green count button, the count of bristles in the selected box is displayed on the left-hand side, along with the x and y position of the box.

This feature provides an easy and efficient way to count the number of bristles in a specific area of the wing.

7.6 State transition Diagram

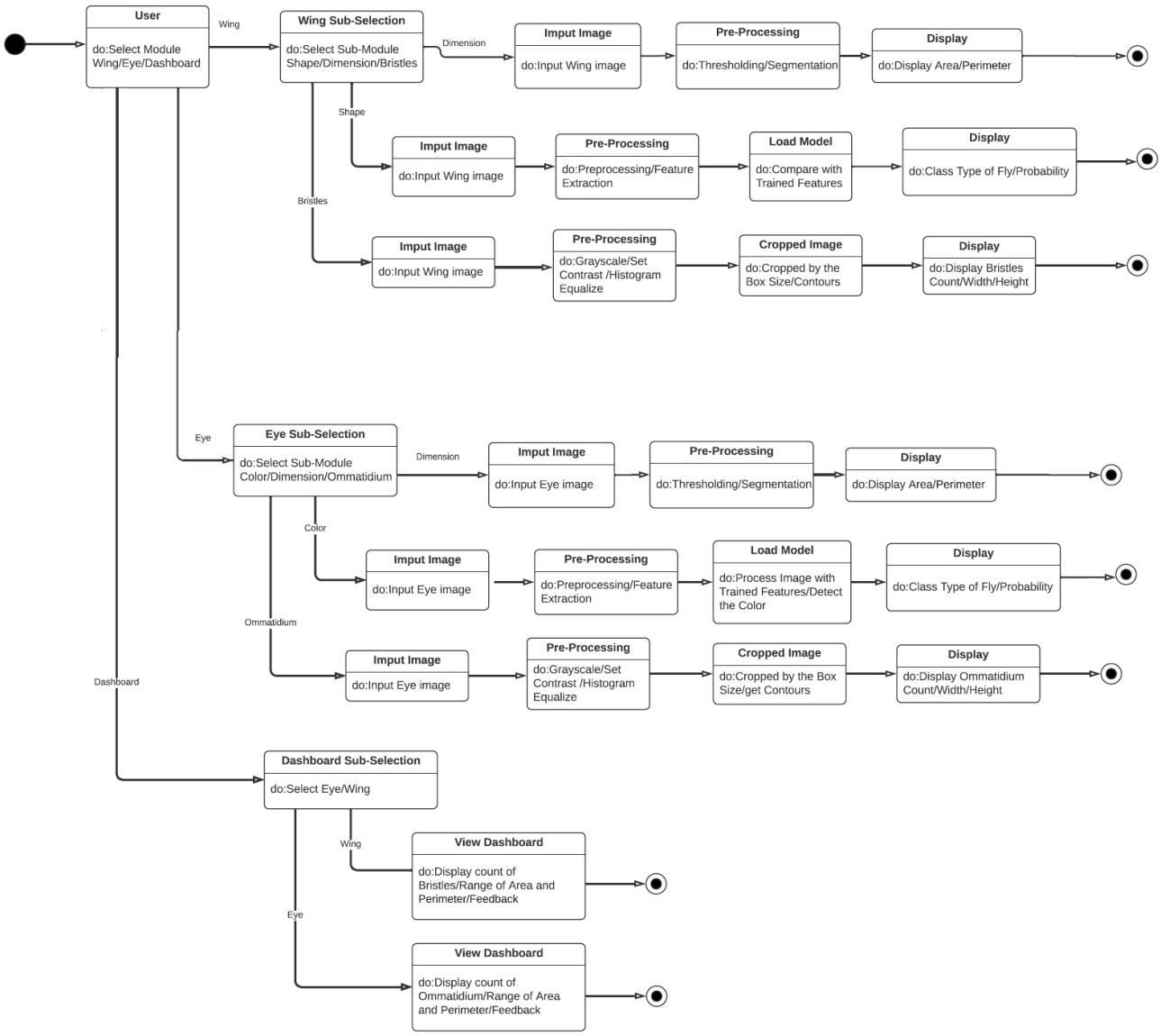


Figure 7.6 – State transition diagram of MakkhiMeter

Figure 7.6 displays the state diagram of MakkhiMeter, which illustrates the various states of the "eye" and "wings" components. Each state performs specific operations, including preprocessing, processing, and post-processing. Additionally, there is a state for the dashboard, which allows the user to view insights for both the "eye" and "wings" components.

8. Testing

In the previous chapter we covered the algorithms used, their complexities and their comparisons with other algorithms. We also covered a descriptive introduction about the system, described the UML and Interactions diagrams and the coding architecture of the system. This chapter covers all tests performed in the system and the most important code in the system. The tests include Blackbox, Whitebox, and integration testing.

8.1 WhiteBox Testing

8.1.1 Wing Dimensions & Eye Dimensions

8.1.1.1 FlowGraph

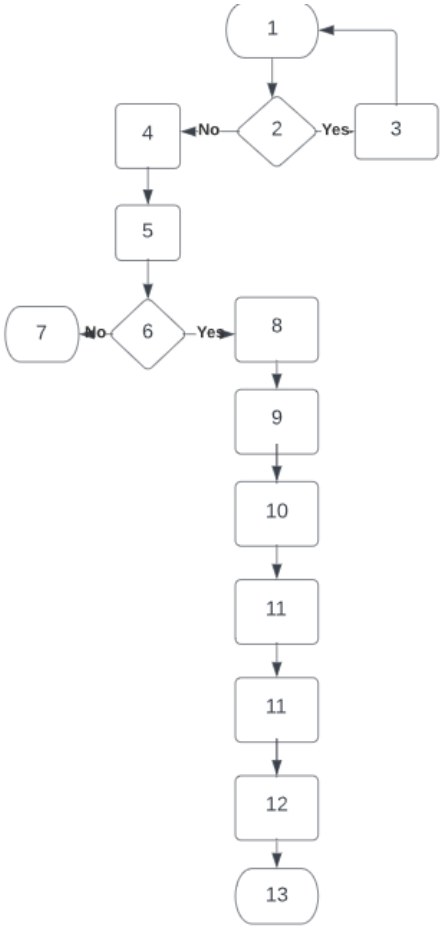


Figure 8.1.1.1: FlowGraph of Wing Dimensions & Eye Dimensions

8.1.1.2 Complexity

V(n)=2+1=3

Total Test case 3.

8.1.2 Wing Bristles & Eye Ommatidium

8.1.2.1 FlowGraph

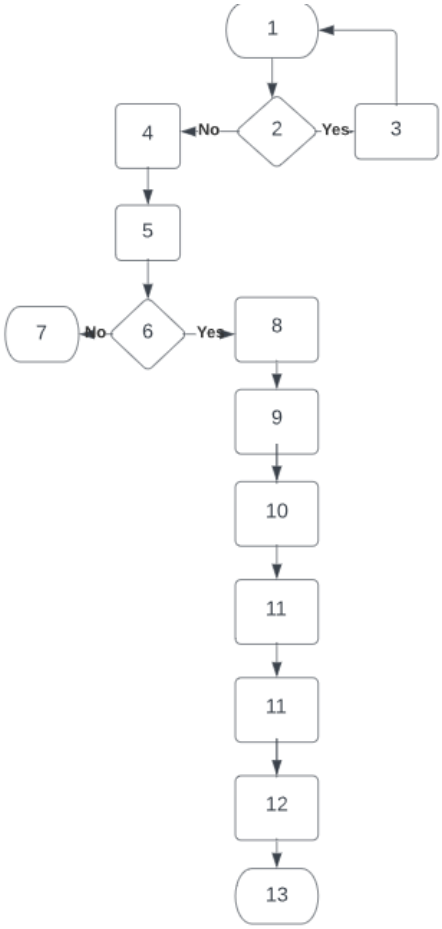


Figure 8.1.2.1: FlowGraph of Wing Bristles & Eye Ommatidium

8.1.2.2 Complexity

V(n)=2+1=3

Total Test case 3.

8.1.3 Wing Shape

8.1.3.1 FlowGraph

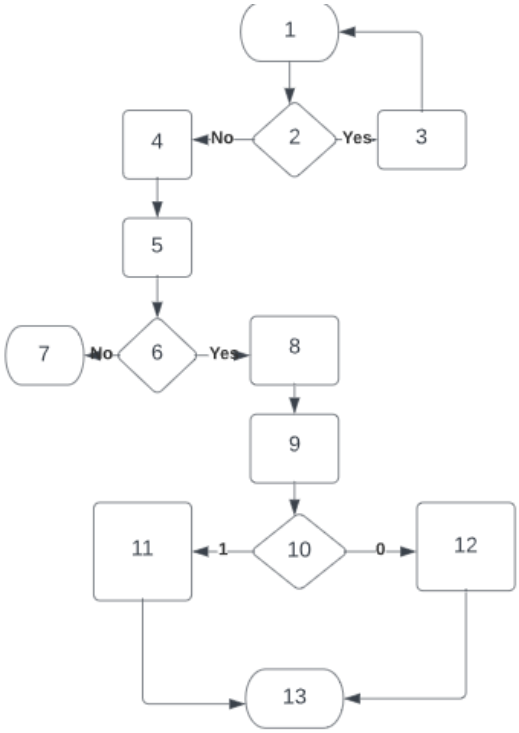


Figure 8.1.3: FlowGraph of Wing Shape

8.1.3.2 Complexity

V(n)=3+1=4

Total Test case 4.

8.1.4 Eye Color

8.1.4.1 FlowGraph

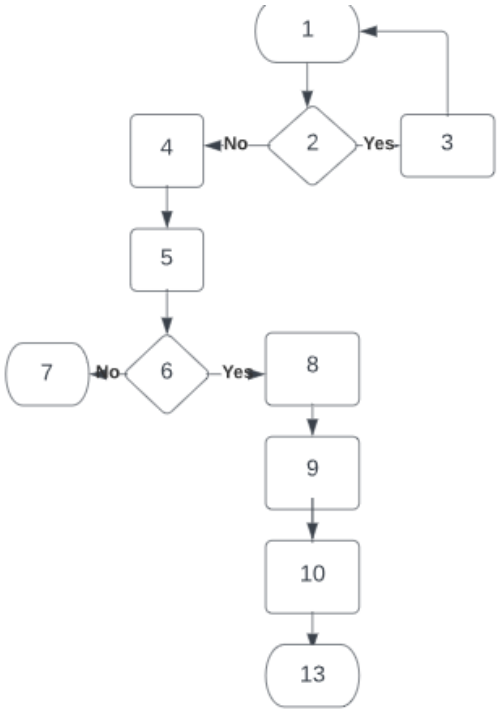


Figure 8.1.4: FlowGraph of Eye Color

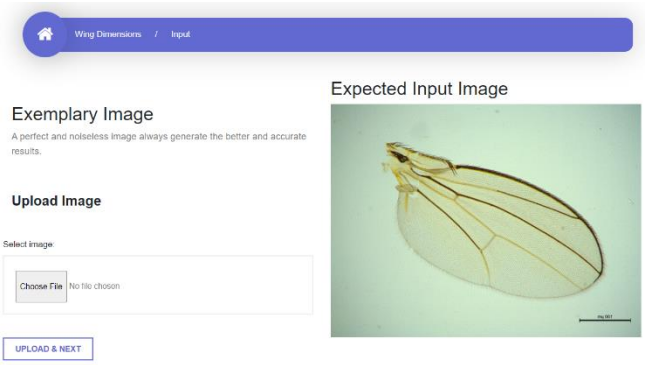
8.1.4.2 Complexity

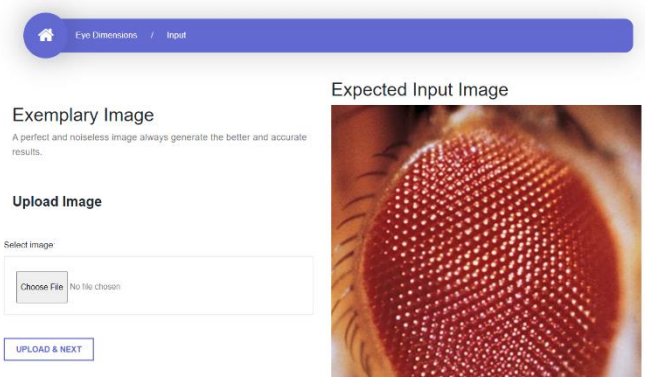
V(n)=2+1=3

Total Test case 3.

8.2 BlackBox Testing

8.2.1 Wing Dimension & Eye Dimensions





8.2.1.1 CheckList

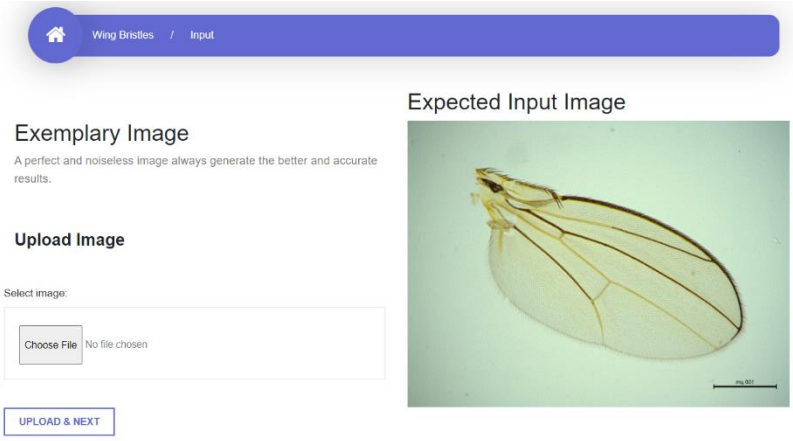
- Check whether the Input Image button is clickable and allows us to upload an image.
- Check whether the Upload & Next button directs us to new page.
- Check whether the system rejects the incorrect image.
- Check whether system gives correct units at output.

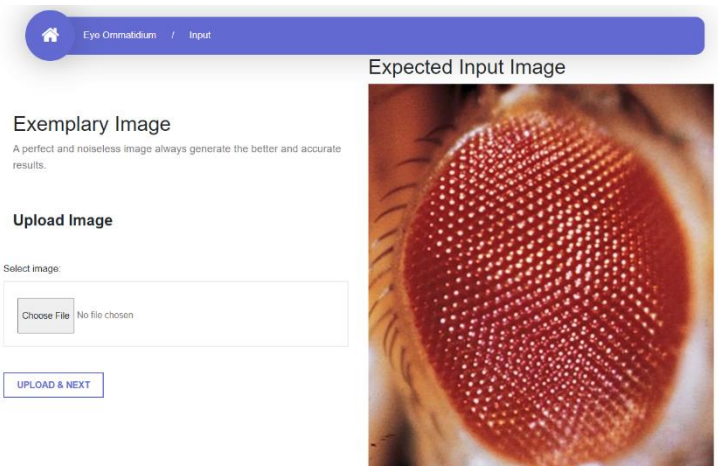
8.2.1.2 Test Cases

Test Cases	Input	Output	Status
Validates that system allows to input image	Image in tif format	Directs to other Page	Valid
Validates that system donot allows to input incorrect image	Input Image of car	Rejects and prompts that image is not correct	Valid
Validates that system returns output in mm & mm^2	Wing or eye Image	Returns Area and perimeter in mm & mm^2	Valid
Check if buttons are properly aligned and place accordingly in their respective places	Buttons are placed at correct places and readable	User easily see them and use them to perform certain action or task	Valid

Table 8.2.1: Test cases for Wing & Eye Dimensions

8.2.2 Wing Bristles & Eye Ommatidium





8.2.2.1 CheckList

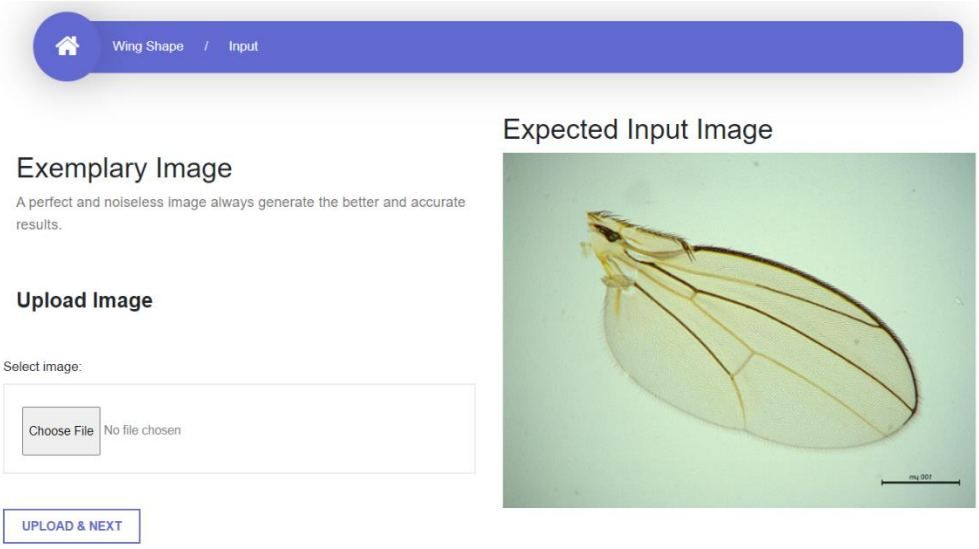
- Check whether the Input Image button is clickable and allows us to upload an image.
- Check whether the Upload & Next button directs us to new page.
- Check whether the system rejects the incorrect image.
- Check whether system gives correct number of bristles and ommatidium.

8.2.2.2 Test Cases

Test Cases	Input	Output	Status
Validates that system allows to input image	Image in tif/png format	Directs to other Page	Valid
Validates that system donot allows to input incorrect image	Input Image of car	Rejects and prompts that image is not correct	Valid
Validates that system returns correct number of bristles & Ommatidium	Wing or eye Image	Returns the count of bristles and ommatidium	Valid
Check if buttons are properly aligned and place accordingly in their respective places	Buttons are placed at correct places and readable	User easily see them and use them to perform certain action or task	Valid

Table 8.2.2: Test cases for Wing Bristles & Eye Ommatidium

8.2.3 Wing Shape



8.2.3.1 CheckList

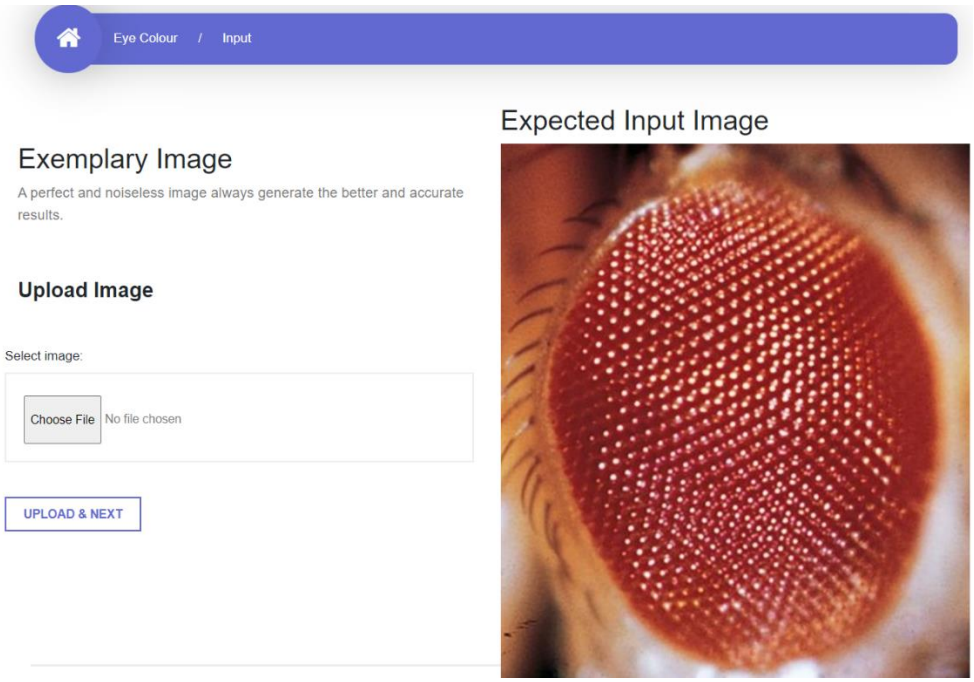
- Check whether the Input Image button is clickable and allows us to upload an image.
- Check whether the Upload & Next button directs us to new page.
- Check whether the system rejects the incorrect image.
- Check whether system gives correct output based on the image provided whether it is Normal or Mutant.

8.2.3.2 Test Cases

Test Cases	Input	Output	Status
Validates that system allows to input image	Image in tif format	Directs to other Page	Valid
Validates that system donot allows to input incorrect image	Input Image of car	Rejects and prompts that image is not correct	Valid
Validates that system returns correct class	Wing or eye Image	Returns class either normal or mutant	Valid
Check if buttons are properly aligned and place accordingly in their respective places	Buttons are placed at correct places and readable	User easily see them and use them to perform certain action or task	Valid

Table 8.2.3: Test cases for Wing Shape

8.2.4 Eye Color



8.2.4.1 CheckList

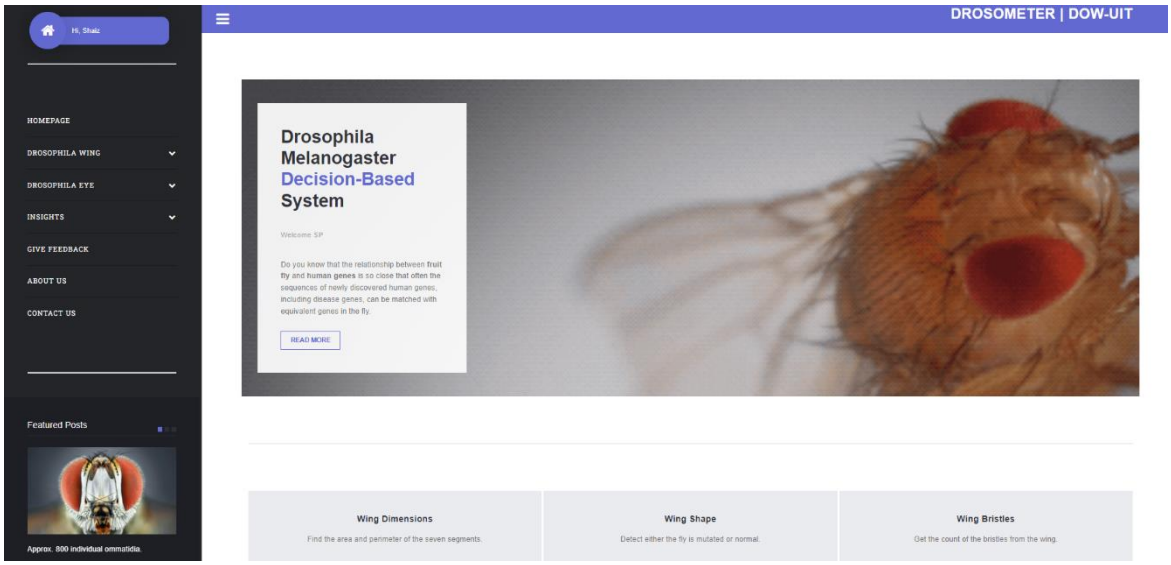
- Check whether the Input Image button is clickable and allows us to upload an image.
- Check whether the Upload & Next button directs us to new page.
- Check whether the system rejects the incorrect image.
- Check whether system gives output of 4 color along with percentages.

8.2.4.2 Test Cases

Test Cases	Input	Output	Status
Validates that system allows to input image	Image in tif/png format	Directs to other Page	Valid
Validates that system donot allows to input incorrect image	Input Image of car	Rejects and prompts that image is not correct	Valid
Validates that system returns 4 colors	Eye Image in png format	Returns 4 colors along with percentage	Valid
Check if buttons are properly aligned and place accordingly in their respective places	Buttons are placed at correct places and readable	User easily see them and use them to perform certain action or task	Valid

Table 8.2.4: Test cases for Eye Color

8.2.5 Homepage



8.2.5.1 CheckList

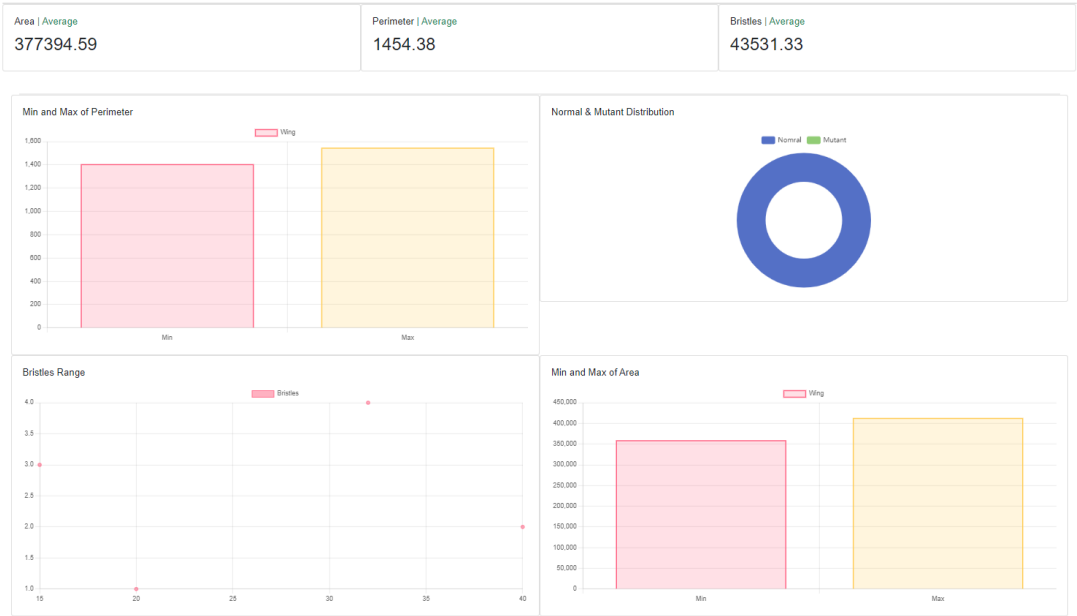
- Check whether all the modules are accessible.
- Check for the About us and Contact us page.
- Check for Dow-Uit logo.
- Application name should be visible.

8.2.5.2 Test Cases

Test Cases	Input	Output	Status
Validates that modules are accessbile	Select any module	Directs to the intro page of module	Valid
Validates that About us and Contact us page availability	Select any of the page	Displays the page.	Valid
Validates Dow-Uit Logo	Assert for logo in right side	Logo displayed	Valid
Validated Application name	Assert for Application name	Name displayed	Valid
Check if buttons are properly aligned and place accordingly in their respective places	Buttons are placed at correct places and readable	User easily see them and use them to perform certain action or task	Valid

Table 8.2.5: HomePage

8.2.6 Dashboard



8.2.6.1 CheckList

- Check whether all the graphs are visible.
- Check whether all the aggregate values are visible.
- Checks that the feature which we extracted are shown.

8.2.6.2 Test Cases

Test Cases	Input	Output	Status
Validates that graphs are visible	Select dashboard	Directs to the into dashboard which displays 4 graphs	Valid
Validates that Aggregate values	Select the dashboard or when dashboard is empty input the image and check the aggregate value	Displays the values	Valid
Validates Feature values	Select the dashboard or when dashboard is empty input the image and check the values are shown in graphs	Values displayed	Valid

Table 8.2.5: HomePage

8.3 Integration Testing

Integration testing is a type of testing where software modules are integrated logically and are tested as a group. Software projects mostly contain multiple modules that are collectively combined together in components and are coded by different programmers. The purpose for integration testing is to identify the defects in the interaction between software modules when they are integrated. We will follow the big bang approach as it is convenient for small systems as in this approach all the components are integrated simultaneously, and testing is done as a whole system. The following are some of the test cases which shows how the system is integrated at once

Test Cases	**Description**	**Expected Output**	**Status**
Validate the link between Wing Dimensions and Dashboard	Upload wing image extract features and features should be showed in dashboard	Display visualization	Pass
Validate the link between Wing Bristles and Dashboard	Upload wing image extract brisltes and it should be showed in dashboard	Display visualization	Pass
Validate the link between Wing Shape and Dashboard	Upload wing image get the class and it should be showed in dashboard	Display visualization	Pass
Validate the link between Eye Dimensions and Dashboard	Upload eye image extract features and it should be showed in dashboard	Display visualization	Pass

Validate the link between Eye Ommatidium and Dashboard	Upload eye image extract features and features should be showed in dashboard	Display visualization	Pass
Validate the link between Eye color and Dashboard	Upload eye image extract color and it should be showed in dashboard	Display visualization	Pass
Validate that each module is accessible from homepage and we extract results from that module	Upload the image and extract results	Output displayed	Pass

Table 8.3: Integration Testing

8.3.1 Bug List

Following are the bugs that are present in the applications:

1.2.1 Missed out Segment in Wing Dimension

Bug	**Input**	**Faulty Output**	**Expected output**
For some wing data incorrect segment was extracted	Input Wing image	Some segment misses out	Segment should not miss out

8.4.2 Miscalculated on Mutant Eye

Bug	**Input**	**Faulty Output**	**Expected output**
For some mutant eye image incorrect dimensions returned	Input Mutant eye image	Miscalculates dimensions	Dimensions should return correctly for mutants.

9. Conclusions

Drosophila shares many genes and biological processes with humans, making it a valuable tool for studying human genetics. For example, many genes that regulate basic cellular processes, such as cell cycle control and DNA repair, are highly conserved between Drosophila and humans. In addition, many diseases that affect humans, such as cancer and neurodegenerative disorders, can be modeled in Drosophila. The genetic tools and techniques developed for studying Drosophila have also been instrumental in advancing our understanding of human genetics. For example, the discovery of RNA interference (RNAi) in Drosophila led to the development of this technique for silencing genes in many other organisms, including humans.

In this project we identified a critical need in Drosophila research for a proper tool to analyze the characteristics of Drosophila wings and Drosophila eye, including accurate measurements of wing area and perimeter, and precise counts of wing bristles. Currently, there is no existing tool or method available that allows researchers to effectively perform these measurements, which can significantly impede their research progress. Additionally, there is a need for a tool that can classify different types of Drosophila wings based on their morphological characteristics. By addressing this need, the project has developed an innovative solution that can greatly improve the efficiency and accuracy of Drosophila research. The tool provides automated and precise measurements of wing area and perimeter, as well as bristle counts, and can classify different types of Drosophila wings based on their morphological features. This breakthrough can help researchers gain a better understanding of the genetic and developmental mechanisms underlying the morphology of Drosophila wings, which can have far-reaching implications for genetics, evolution, and ecology research.

In addition to the critical need for measuring and analyzing Drosophila wing characteristics, the project also identified a similar need for analyzing Drosophila eye morphology. Specifically, there is a need for a proper tool to accurately measure the area and perimeter of Drosophila eyes, as well as to count the number of ommatidia present in the eye.

10. Future work

In the meantime, we are working to develop a user-friendly software application that will enable researchers to easily count the number of bristles on drosophila wings, as well as classify the wing type, and calculate the wing segment's area and perimeter. Additionally, we are working to count the number of ommatidia in the drosophila eye, as well as identify the number of colors in drosophila eye and calculate the area and perimeter of drosophila eye shape. Our application will be designed to work with digital images of drosophila and will utilize image processing algorithms to automatically segment, classify and count the bristles/ommatidia of drosophila.

Initially, we will focus on developing algorithms for segment, classify and counting bristles on common strains of drosophila. As we continue to develop the application, we plan to incorporate image data from a wide range of drosophila strains to increase the accuracy of our algorithms.

Our application will be accessible to everyone and will be free of cost as we aim to make it a valuable tool for the research community. In the future, we plan to expand the application's capabilities to include more detailed analysis of drosophila morphology, such as tracking the movement of individual flies and cell lineage of the thoracic muscles of Drosophila.

As our project progresses, we plan to conduct experiments to test the accuracy of our algorithms. In future we explore new methods for improving their performance. We also plan to collaborate with researchers in the field to gather feedback on our application and identify areas for improvement.

In future counting the number of thoracic bristles on a Drosophila fly can provide valuable information for genetic and environmental studies. One traditional method of counting thoracic bristles is by manually examining the dorsal surface of the thorax under a microscope and counting each bristle one by one. However, this method can be time-consuming and prone to human error. For future work in this area, there are several potential avenues for improving the accuracy and efficiency of bristle counting. One approach could be to incorporate multi-scale and multi-resolution analysis to account

for the variability in bristle sizes and locations. Additionally, more extensive training datasets could be created to improve the performance of machine learning algorithms. Furthermore, new techniques for feature extraction and classification could be explored to increase the accuracy of the algorithms.

In future we will also accommodate that our model is also highlight the mutation area in the drosophila wing. Here are a few additional approaches that we consider in future.

- **Texture analysis:**
  Texture analysis is a method for characterizing the spatial arrangement of pixels in an image. By analyzing the texture of the wing surface, we can identify any variations in the texture caused by mutations or abnormalities. We can use different texture analysis techniques such as local binary patterns (LBP), gray-level co-occurrence matrix (GLCM), or Gabor filters to highlight these variations.

- **Color analysis:**
  Color analysis is a method for characterizing the colors in an image. By analyzing the color of the wing, we can identify any variations in color caused by mutations or abnormalities. We can use different color analysis techniques such as color histogram, color moments, or color coherence vectors to highlight these variations.

In the future, it is likely that machine learning and deep learning approaches will be applied to segment the wing area. This will involve using advanced algorithms to analyze data in order to identify different parts of the wing and their respective functions.

11. Achievements

We are the group of 5 people have worked in Dow Fly's Lab for 5 months. In this time span we have learned many things such as using the Compound Mircroscope, Delivering the tasks on daily basis, 2-D Microscope etc.. During this tenure some of our group members also got the exposure to work with the foreign university projects as well.

We also attended the Drosophila Mini Symposium in Dow University. In this symposium we learned the basics of this fly, its importance and how this fly originates. We also got to know that how this fly generation works, about its mutation and its importance and how this mutation effects the hairs and wing shape, sizes, color etc.

This project will be applied for funding in the month of December-January. This funding will be applied through the Dow university, and this will be foreign based funding.

12. Appendices

12.1 Annex A

Introduction

Drosophila melanogaster, commonly known as the fruit fly, has a genome that shares approximately 75-80% similarity with the human genome. he fruits fly has proven to be an invaluable model organism in genetic analysis, particularly in the study of conserved biological processes relevant to human development and diseases. The wings of Drosophila melanogaster, in particular, have been extensively utilized in genetic screening studies and are an ideal site for phenomics analysis due to their sensitivity to mutation or defects caused by various factors such as the testing of medications or breeding. The observation of changes in wing structure or wing bristles can provide valuable insight into the genetic makeup of these insects and how it may be related to the genetic makeup of humans. In this Project we were working on two modules

1. Wing Module
2. Wing Dimensions
3. Wing Bristle
4. Wing Shape
5. Eye Module
6. Eye Color
7. Eye Ommatidium
8. Eye Dimensions

<u>Wing Module</u>

Wing Dimension

Studying the area and perimeter of the wings of Drosophila melanogaster can reveal genetic mutations and differences, providing insight into the genetic basis of traits and their conservation in other organisms, including humans.

Wing Bristle

Counting the wing bristles of Drosophila melanogaster can reveal specific genetic mutations or defects and provide valuable information in understanding the genetics of these insects and their relationship to human genetics. The number and arrangement of wing bristles are genetically determined and can be used as a marker to identify changes in the normal pattern of bristle arrangement caused by genetic mutations or other factors.

Wing Shape

Classifying the wing shape of Drosophila melanogaster can reveal specific genetic mutations or defects and provide valuable information in understanding the genetics of these insects and their relationship to human genetics. The shape of the wings is genetically determined and can be used to identify changes in the normal shape caused by genetic mutations or other factors. Additionally, wing shape classification can be useful in identifying and classifying different species or strains of Drosophila melanogaster.

Eye Module

Eye Color

The eye color of Drosophila Melanogaster is determined by genetic and environmental factors, with pigmentation controlling the typical red color. Eye color variations are caused by mutations in multiple genes affecting pigmentation, and can also be influenced by chemicals or light exposure. Studying the genetics of eye color variation in Drosophila Melanogaster has provided insights into the molecular mechanisms of pigmentation and has applications in understanding pigmentation genetics in other organisms.

Eye Ommatidium

Counting the ommatidia in the eyes of Drosophila melanogaster can reveal specific genetic mutations or defects and provide valuable information in understanding the genetics of these insects and their relationship to human genetics. The number and arrangement of ommatidia are genetically determined and can be used to identify

changes caused by genetic mutations or other factors. Additionally, counting ommatidia can be useful in studying visual perception and behavior in these insects.

Eye Dimensions

The compound eye of the fruit fly, Drosophila melanogaster, is composed of many ommatidia, each containing light-sensitive photoreceptors that detect and convert light into neural signals sent to the brain. The roughly hexagonal arrangement of ommatidia allows for wide field of view, detection of movement and fine details, and detection of polarisation of light. The fruit fly eye is useful for studying genetics, development, and evolution due to the individual development of each ommatidium, and is a well-understood system for understanding the neural basis of visual processing in the brain.

About Organization

Usman Institute of Technology University (UITU)

Usman Institute of Technology and University is a non- for-profit institute serving the nation since 1994. Hence the institute offers undergraduate studies in Computer science and Engineering. Usman Institute Technology and University also offers Scholarships to the students at the Institute. Its undergraduate programs are recognized by Higher Education Commission (HEC) and are accredited by Pakistan Engineering Council (PEC) and National Computing Education Accreditation Council ((NCEAC).

Dow University of Health Sciences (DUHS)

Dow University of Health Sciences (DUHS) is a premier institute offering MBBS and BDS undergraduate education, with a strong emphasis in the medical, biomedical sciences, public health, allied health sciences and business management programs. The University has one main campus, and another sub-campus located in the urban metropolitan city of Karachi, Sindh. The University_has a school of Postgraduate studies, which offers postgraduate and_doctoral programs in almost all academic disciplines relating to the biomedical and health sciences, including clinical diplomas in 13 specialties, Masters/MS /MPhil and Ph.D. programs in 32 different specialties, such as Master of Advanced Physiotherapy, Masters in Nursing, Master of Business Administration, with 18 specialties of the biomedical and basic sciences offered at the MPhil level.

Organizations societies for Students

The UITCS ACM Student Chapter is a group of students who are interested in promoting computing and information technology. In March 2006, the group changed its name from UITCS to UITCS ACM Student Chapter.

UTECH is a software competition held at the UIT Campus in Pakistan. It is one of the largest IT events in the country, featuring contests and exhibitions of software from various regions.

12.2 Annex B

Background and Literature Review

Relevant Literature and Algorithm

The Drosophila Melanogaster fruit fly is one of the most popular insects in the world, but little is known about how to analyze and extract specific characteristics of the fly's wing.

[1] One method for analyzing the wing of Drosophila Melanogaster is to start by inputting an image and using edge detection techniques to identify the edges of the wing but requires a lot of manual input from the user and even takes a lot of time. **in Figure 12.2**.

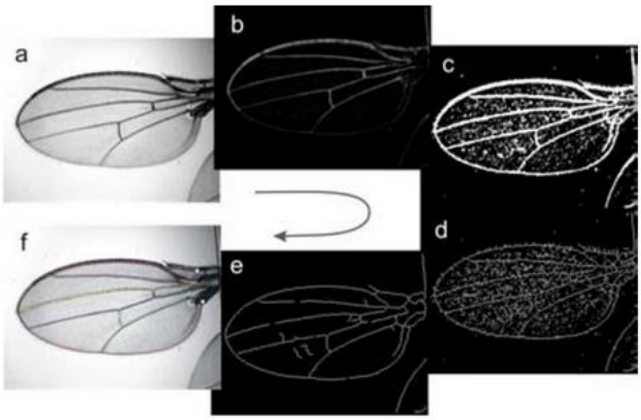
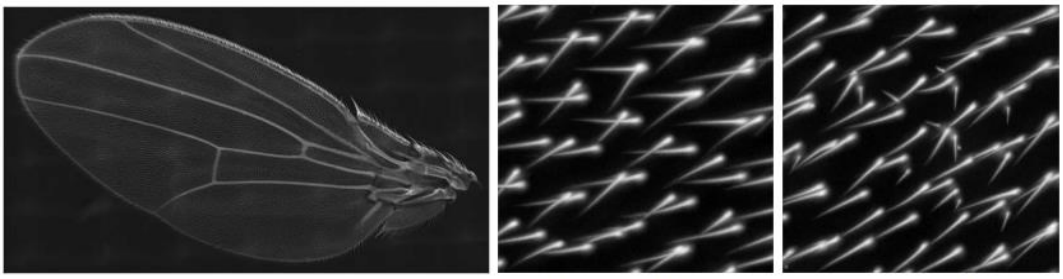


Figure 12.2: Preprocessing Techniques

[2] Another approach that has been proposed is the use of contour and distance transformation techniques for preprocessing the image. However, this methodology also has the drawback of being time-consuming and requiring significant manual effort from the user.

[3] An alternate methodology that has been proposed, is using machine learning to extract the mask of the image, and then further process it to segment the wing into different parts. The output of this approach is expected to provide detailed information about the features of the wing.

[4] A method for analyzing the bristles or hairs on the wing of Drosophila Melanogaster has been proposed, focusing on how to identify the edges of these structures in wing images. However, it has been noted that no approach has been presented on how to accurately count the number of these hairs. The results are shown in **Figure (4) 13.2.**



(a) a wing (b) normal hairs (c) mutant hairs

Figure (4) 12.2: Wing Bristles

12.3 Annex C

Hardware, Software Analysis and Requirements

Software Requirements

The project involves creating a web application that combines frontend and backend technologies. The frontend is responsible for the user interface while the backend responds to requests from the web application. The main focus is on extracting features and classifying images using image processing and deep learning techniques. The process involves running the input image through a series of python scripts and displaying the output on the web application. A variety of software technologies are used to accomplish this task.

1. **Django**

Django is a high-level Python server-side web framework that allows the fast development of maintainable and protected websites, highly secure, scalable, portable etc. As it will be used as the backend of our web application.

2. **Python**

We use python programming language for our image processing and deep learning work as it has all the functionalities, libraries and frameworks related to our application.

3. **Image Processing**

We will be using certain libraries e.g: OpenCV, and SkImage to accomplish the task of features extraction of wing and eye.

4. **Deep Learning**

It is a type of machine learning which learns to do predictions based on testing, training and validation based on labelled datasets.

5. **JavaScript**

It is the leading language for the front-end design of our websites.

6. **Bootstrap**

Bootstrap is a free, open-source front-end development framework for the creation of responsive websites.

Hardware Analysis

For the computation the machine we will use have the following specifications.

CPU: Intel(R) Core (TM) i5-6300U CPU @ 2.60GHz

GPU: Intel(R) HD Graphics 520

RAM: 8 GB DDR4

HDD: 256 SSD

12.4 Annex D

Software Design and Modeling

In this Phase project architecture is presented (explain which architecture is used), overall design diagrams (complete object diagram, complete class diagram, database diagram, etc.) to be shown. (As such diagrams are usually huge in nature, it is appropriate to print in on A3 or bigger sheets, and then fold it to A4 size). use few behavioral diagrams (sequence diagram, timing diagram, activity diagram, state transition diagram, or composite diagram) only for core technical functionality of the project against use cases. Also show high fidelity prototypes (input, output, inquiry, debugging, customization, wrapper) on (Image, Audio, video, text, tags, AR, VR) against each use case.

12.4.1 Architecture

System Architecture

Our application is using a three -tier Architecture which consist of device an application interface and database at back. The user will use web application to access the content of website. User could upload the image or either wing or eye and the extracted features will be displayed on the web along with the segmentation of the image where needed. Besides the dashboard of the wing and eye is also available on our website which will be displaying insightful features to our user.

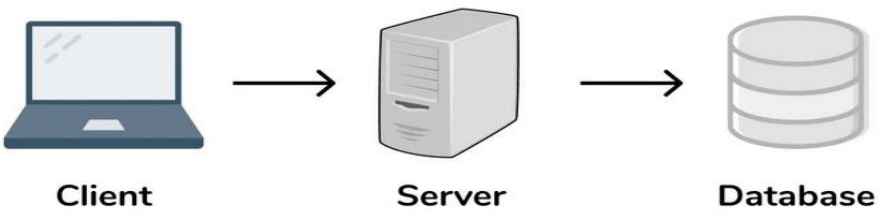


Figure 12.4.1: System Architecture

12.4.2 UML Diagrams

Introduction

Unified Modeling Language (UML) class diagram is the foremost broadly used graph sort for documentation of programming. As most computer program that are being made are still based on Object Oriented Programming (OOP) worldview, utilizing course graphs to record the software. This occurs as the basis of OOP is on the classes their relations between each other. The UML diagrams we include in this chapter are class diagram, object diagram as shown in **Figure 13.4.2.**

UML Diagrams	**Maps of Problem Space (Business Analyst)**	**Model of Solution Space (Designer)**	**Model of Architectural Space (Architect)**
Use case Diagram	Very Important	Not so Important	Not Important
Activity Diagram	Very Important	Not so Important	Not Important
Class Diagram	Average	Very Important	Not so Important
Sequence Diagram	Important	Very Important	Not Important
Object Diagram	Not Important	Very Important	Average

Figure 12.4.2:Mapping of UML Diagrams

12.4.2.1 Use Case Diagram

Use case diagrams are useful for identifying and modeling the requirements of a system, and can be used to communicate these requirements to developers, stakeholders, and other relevant parties. They are also useful for identifying potential problems and areas of improvement within a system.

Main Components of Use Case Diagram

Actors: These are the entities that interact with the system. They can be users, external systems, or any other entity that performs a specific role within the system.

Use cases: These are the actions that the actors can perform within the system. Each use case represents a specific functionality or task that the system can perform.

System boundary: This represents the boundaries of the system, and defines what is inside and outside the system.

Association: This represents the relationships between the actors and the use cases. It shows which actors can perform which use cases.

We have got 2 actors one is our user and second is system (MakkhiMeter). User could upload the image in either wing or eye module or could see the analysis/insights of extracted features. All the processing is done in the system which includes the wing shape and eye color, dimension of eye and wing, wing bristles and eye ommatidium.

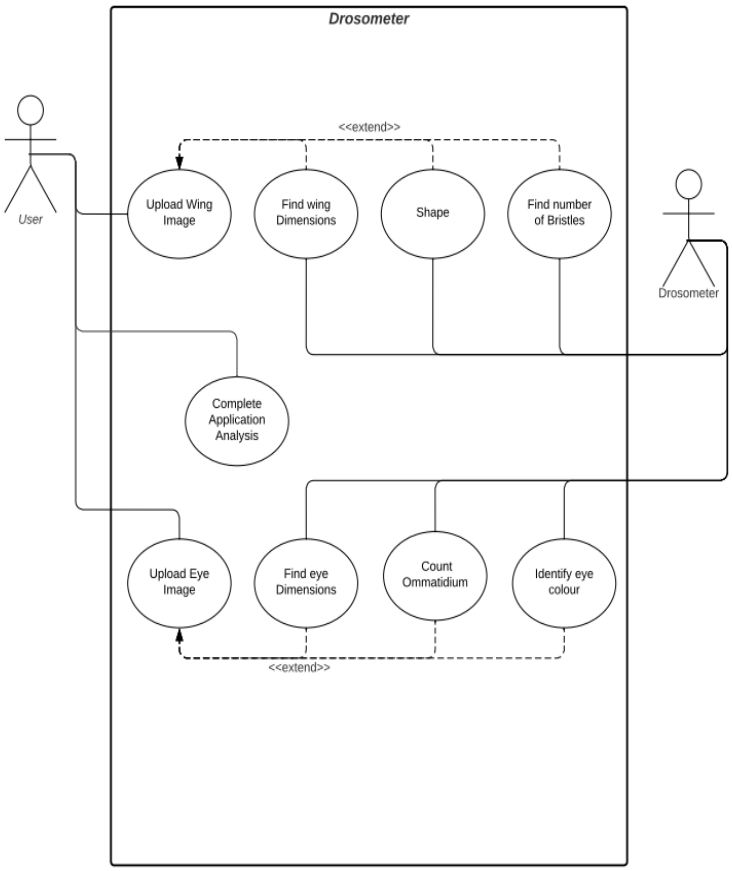


Figure (1) 12.4: Use Case Diagram

12.4.2.2 Activity Diagram

User has got 3 options either it could see the insights of the extracted features, work in the wing module or work in the eye module.

If the user selects the wing module so again it has got the option to either calculate the wing dimension, wing bristles and wing shape. It needs to select one of the options. If it selects the wing dimensions so the input is preprocessed, thresholding and dilation are done, then the segments of the wing are calculated, and results are shown in a data frame. If the wing bristles are selected so the user needs to select the area to get bristles and we would also be calculating whole wing bristles. Lastly, for wing shape the deep learning is loaded, preprocessed image is passed into it, features are extracted by the model and results are displayed.

The same flow is for the eye module with the changes in the preprocessing techniques and getting some extra features concerning the eye.

If the user selects insights, then the dashboard will be shown which will be displaying the relevant insights from the extracted features.

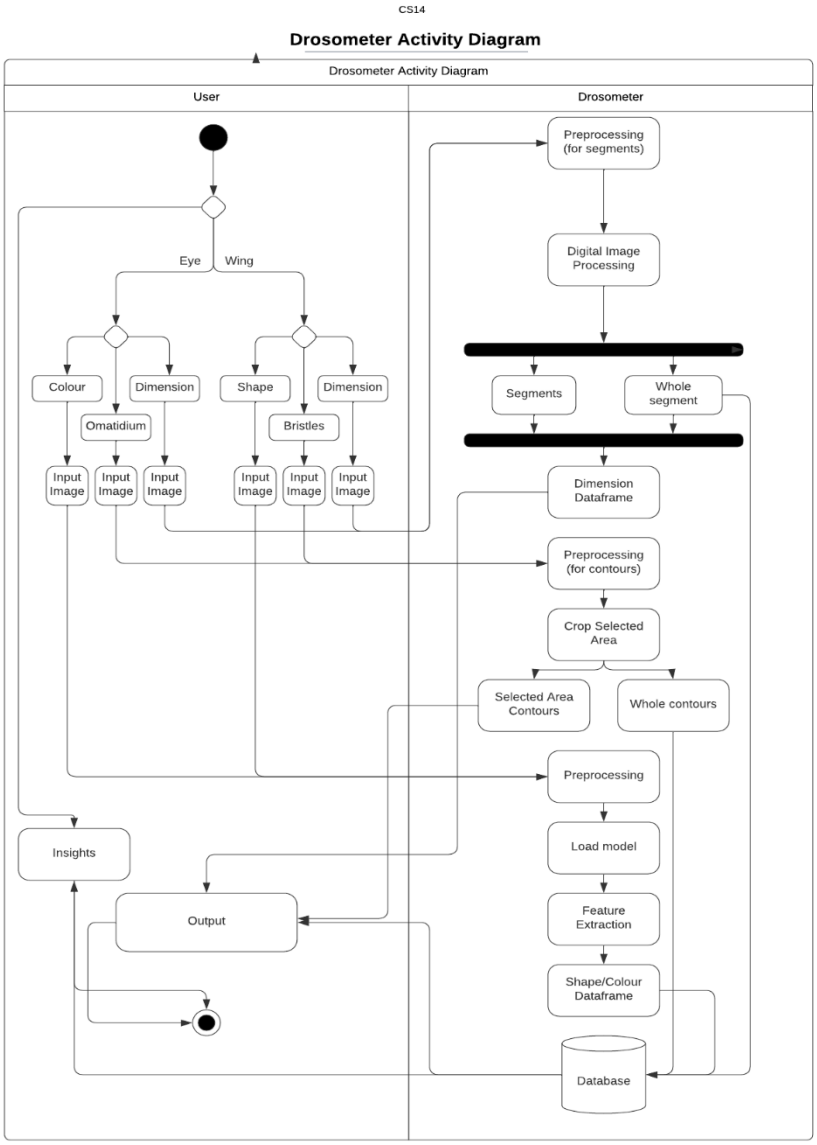


Figure (2) 12.4: Activity Diagram

12.4.2.3 Class Diagram

A class diagram is a type of visual representation used in the field of object-oriented software engineering. It is used to depict the structure of a system by showing the relationships between classes, their attributes, and the operations they can perform.

Main components of a class diagram

Classes: These are the basic building blocks of an object-oriented system. A class represents a group of objects that share the same characteristics and behavior.
Attributes: These are the properties or characteristics of a class. Each attribute has a name and a data type, and represents a piece of information that an object of the class can store.
Operations: These are the actions that an object of a class can perform. Each operation has a name and a list of parameters, and represents a specific task that the object can perform.
Relationships: These are the connections between classes, and show how they are related to each other. There are several types of relationships that can be represented in a class diagram, including inheritance, association, aggregation, and composition.
Class diagrams are useful for understanding the structure of a system and the relationships between its various components. They can be used to communicate the design of a system to developers, stakeholders, and other relevant parties, and can also be used to identify potential problems and areas of improvement in the design.

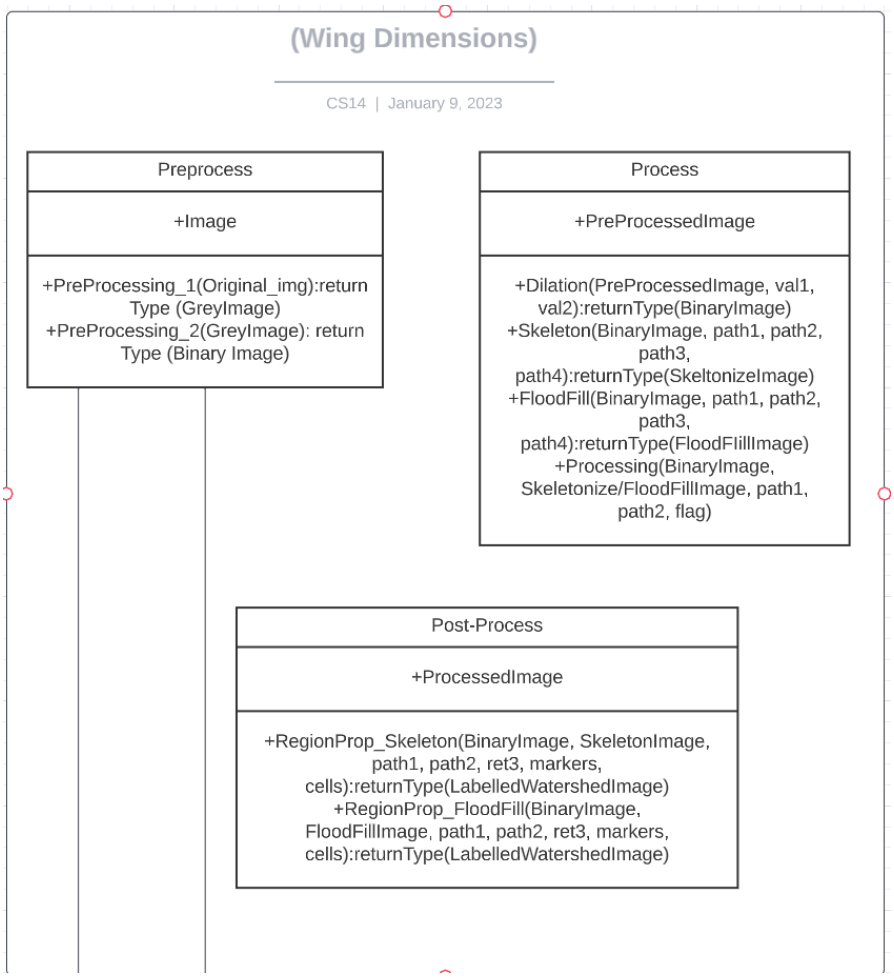


Figure (3) 12.4: Wing Dimension Class Diagram

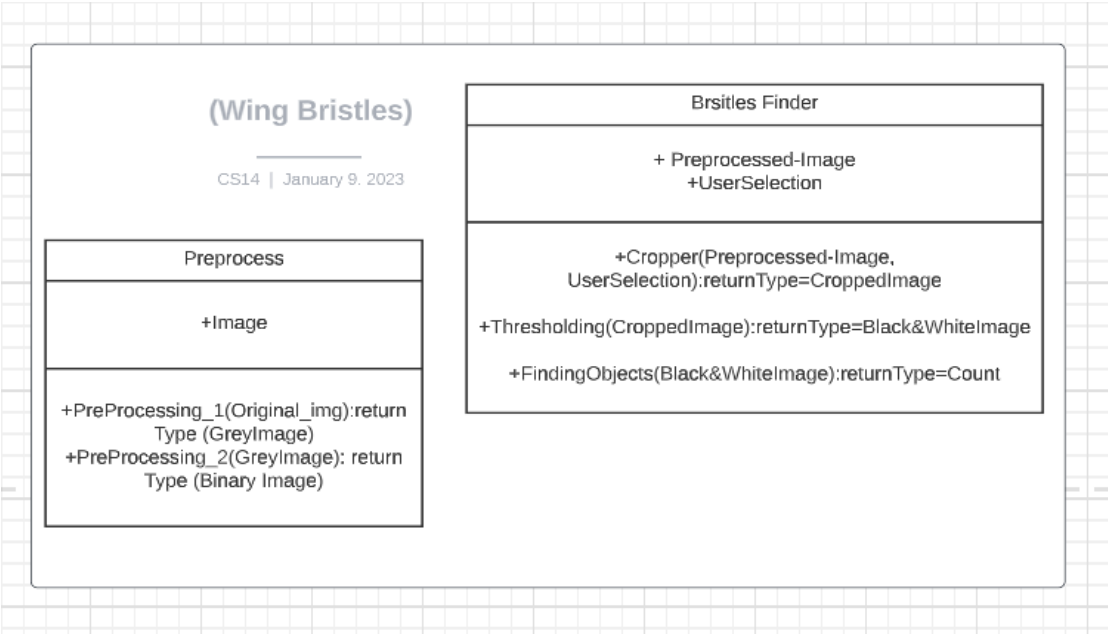


Figure (4) 12.4: Wing Bristle Class Diagram

12.4.2.4 Sequence Diagram

A sequence diagram is a type of visual representation used in the field of software engineering to depict the interactions between objects in a system. It shows the order in which these interactions take place, and the messages that are exchanged between the objects.

Main components of a sequence diagram

Objects: These are the entities that participate in the interactions. They can be objects within the system, or external entities that interact with the system.

Lifelines: These are the vertical lines that represent the lifespan of an object. They show the point at which an object is created, and the point at which it is destroyed.

Messages: These are the interactions between objects, shown as arrows connecting the objects. Each message has a sender and a receiver, and represents a request or a response between the objects.

Activation bars: These are the horizontal lines that show the duration of an operation. They show the point at which an object begins to perform an operation, and the point at which the operation is completed.

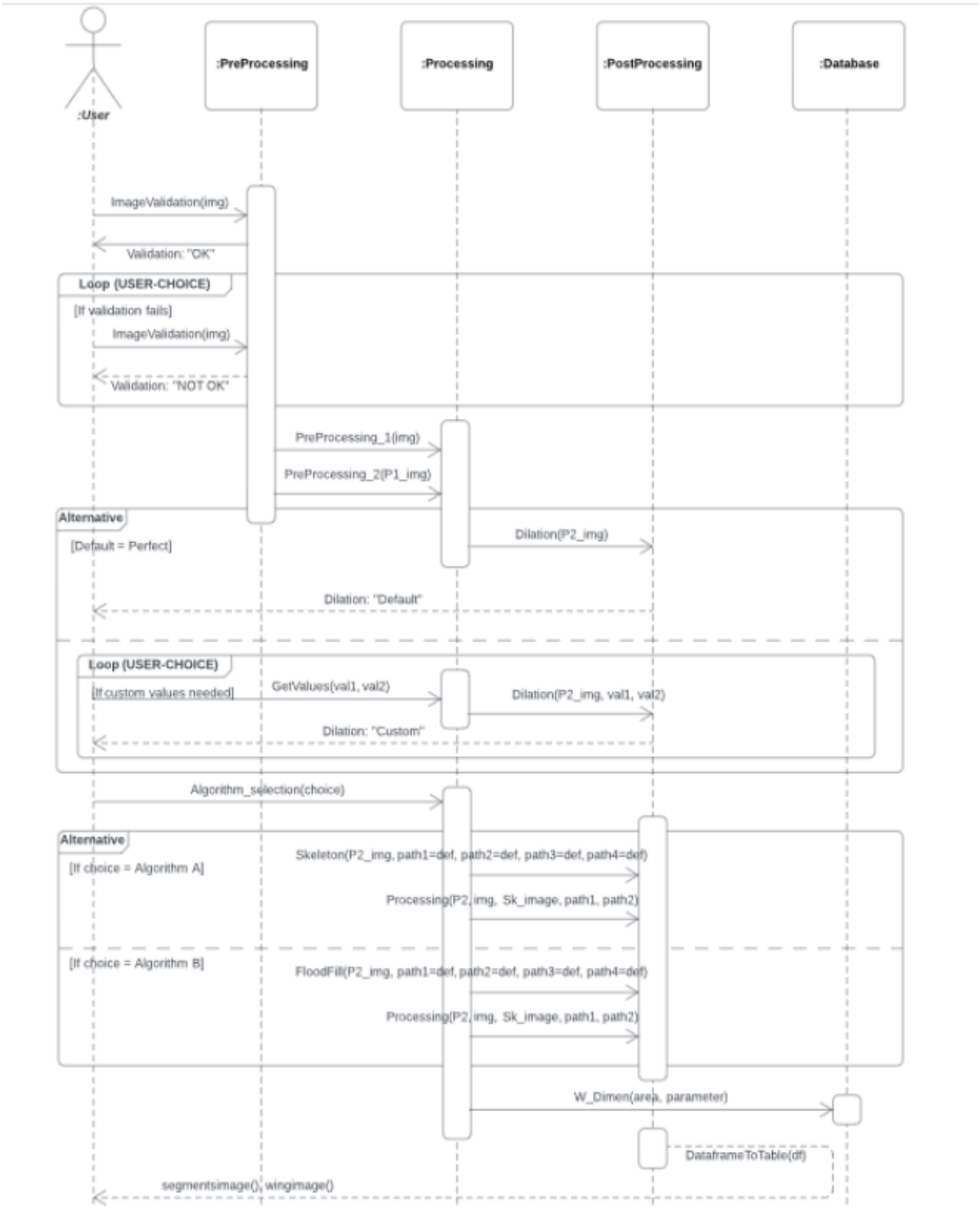


Figure (5) 12.4: Wing Dimension Sequence Diagram

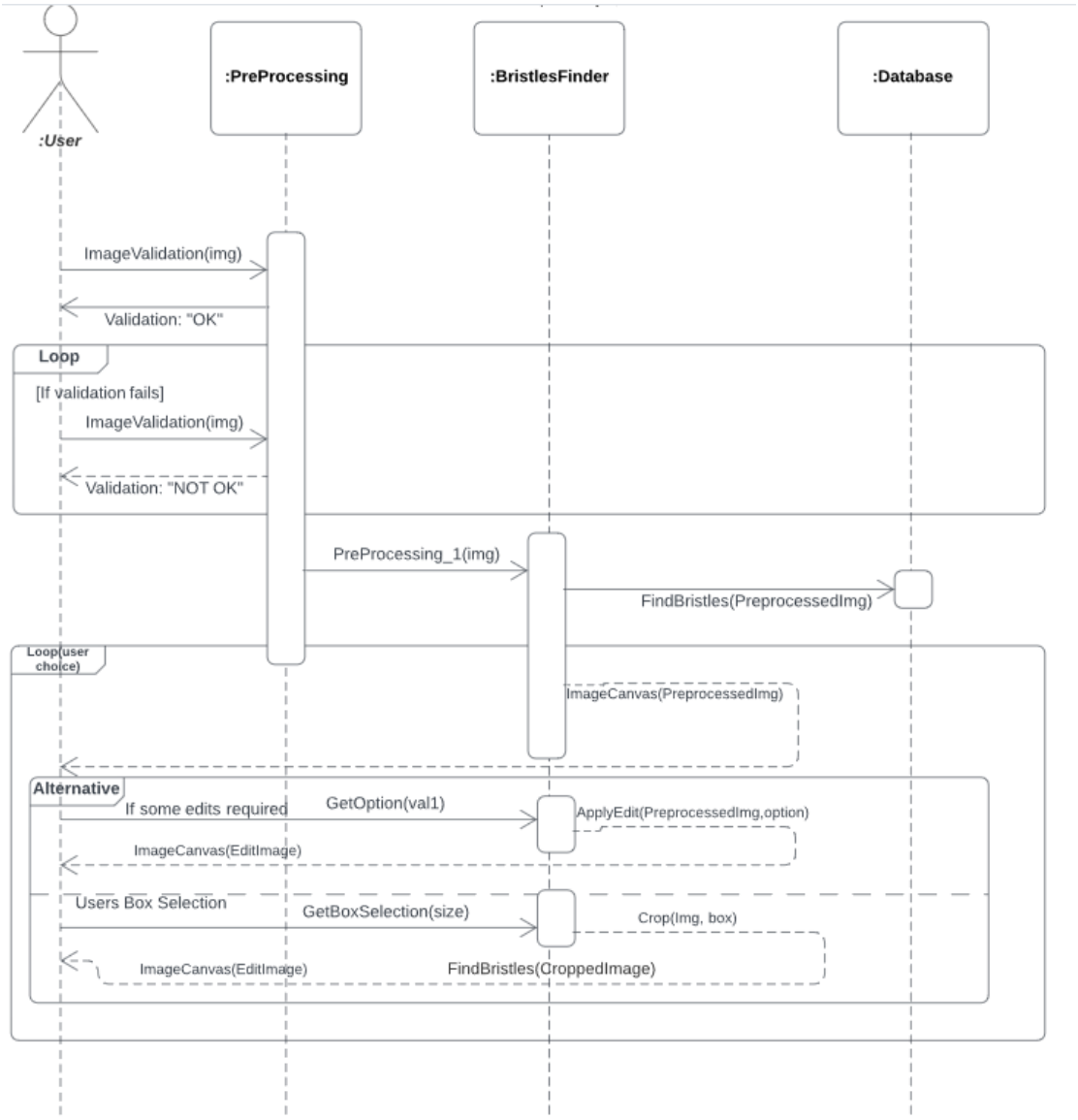


Figure (6) 12.4: Wing Bristle Sequence Diagram

12.4.2.5 Object Diagram

An object diagram is a type of visual representation used in the field of object-oriented software engineering to depict the instances of classifiers in a system, and the relationships between these instances. It is similar to a class diagram, but shows the actual objects in a system rather than the class definitions.

Main components of an object diagram

Objects: These are the instances of classifiers in a system. Each object has a name and a type, and represents a specific entity in the system.

Attributes: These are the properties or characteristics of an object. Each attribute has a name and a value, and represents a piece of information that the object stores.

Relationships: These are the connections between objects, and show how they are related to each other. There are several types of relationships that can be represented in an object diagram, including inheritance, association, aggregation, and composition.

Object diagrams are useful for understanding the runtime structure of a system, and can be used to identify potential problems and areas of improvement in the design. They are also useful for communicating the design of a system to developers, stakeholders, and other relevant parties.

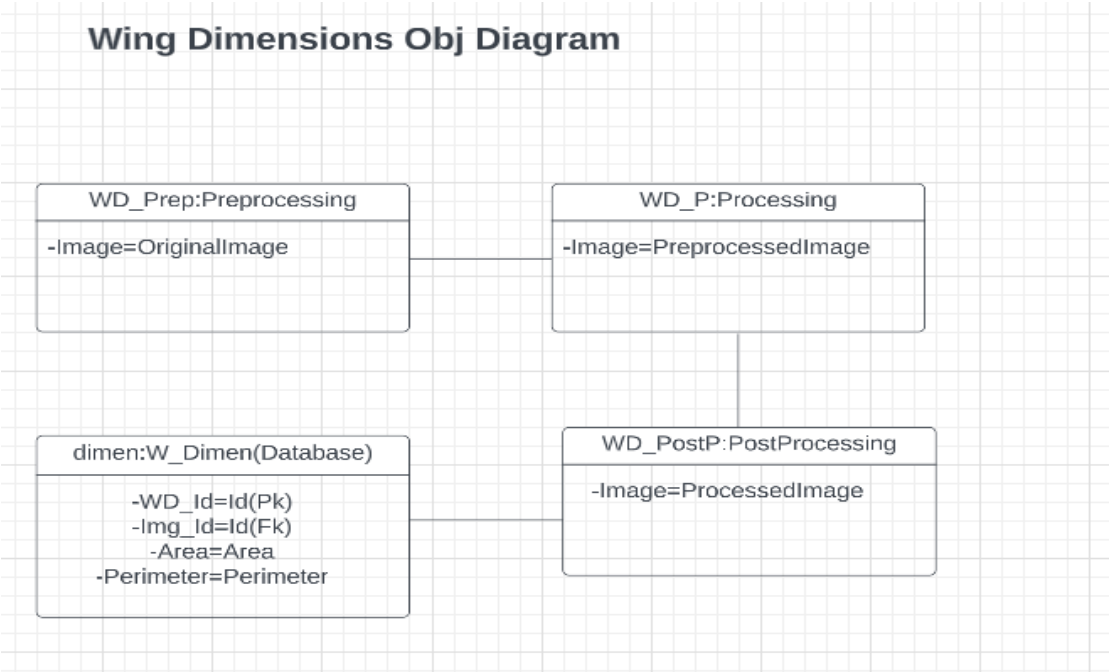


Figure (7) 12.4: Wing Dimension Object Diagram

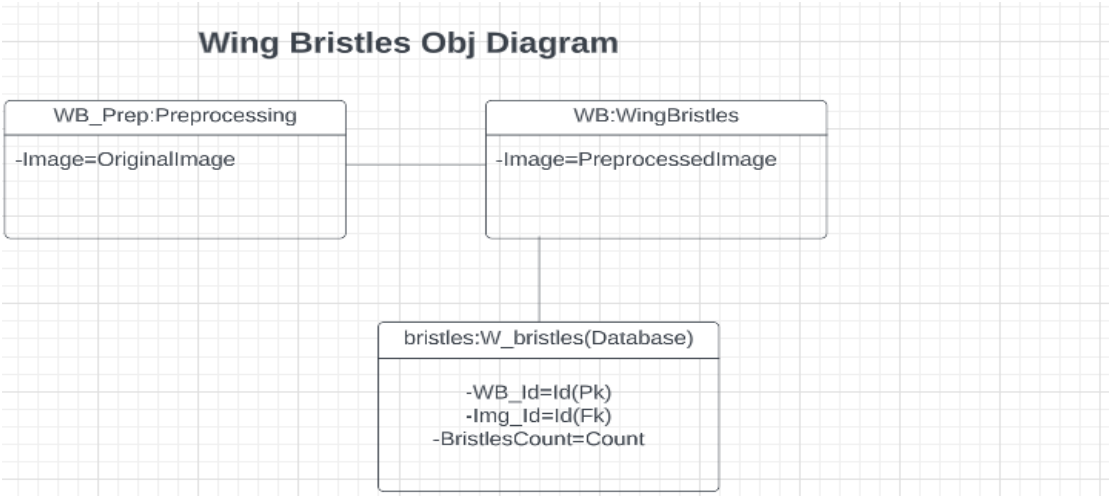


Figure (8) 12.4: Wing Bristle Object Diagram

12.5 Annex E

12.5.1 Deployment Architecture

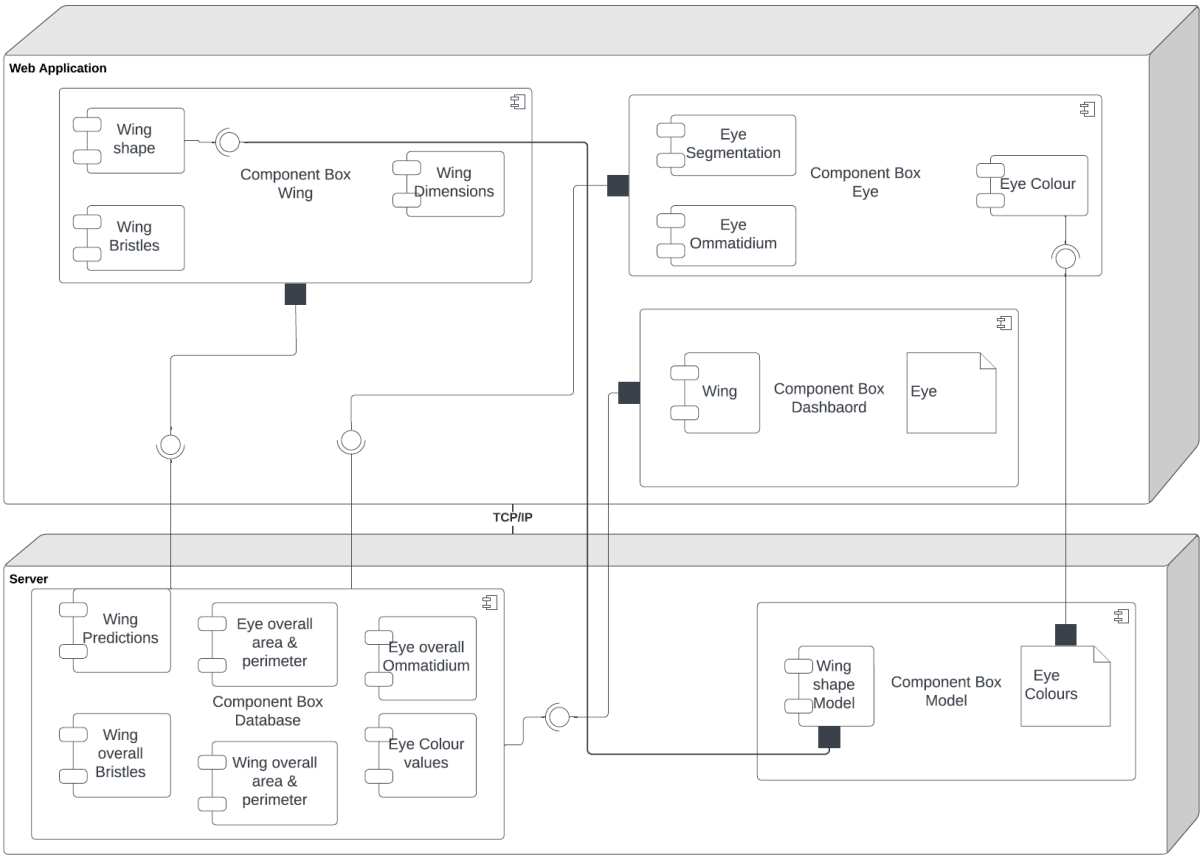


Figure 12.5.1 - Deployment Diagram of MakkhiMeter

Figure 12.5.1 depicts a deployment diagram comprising two blocks. The upper block represents our web application, which includes all of its modules and a dashboard for insights. The Web Application backend is developed using Python programming language and its Web Framework of Django & for the Front-End we have used the Html, Css, Javascript & Bootstrap.

The lower block represents the server where all the scripts are executed, and the database. The Application will be deployed on the DUHS Web Server using Microsoft Server 2019. For the Database we have used the SQLITE Database the reason behind is the amount of the data which will be stored in our database which will be less complex and less in size and since we are storing some numericals data hence we donot need to deploy the large databases to handle.

Additionally, there is another component box on the server that contains a deep learning model for wing shape and a file that assists in identifying colors in the eye.

These all will form a 3-tier architecture application and user will be able to access this application on DUHS server.
For the configuration on DUHS server so at first we have to setup the environment on the Microsoft Server 2019. Install the required libraries and adjust the application settings so that it could allow the traffic to be flown to our application.

After setting the libraries we will deploy the database on it and the required files of the web application project on server.

Then we will get the ip address to access the application and that ip will be masked with the domain name of the DUHS.

12.6 Annex F

Algorithm and Complexity

Please refer to the chapter 6 of the report and chapter 13 of references.

12.7 Annex G

Team Work

Collaboration is essential in any professional setting, as it not only benefits the team as a whole, but also provides opportunities for personal growth and career advancement. Our group effectively divided tasks and worked tirelessly to meet our deadline. The key traits of an effective team include teamwork, dedication, shared roles, and clear communication. By completing the project, team members were able to improve their skills and identify areas for growth, allowing them to apply these lessons to other aspects of their work and personal lives.

Research Scope

Participating in the project was a valuable educational opportunity that allowed us to enhance our research abilities. By examining various research papers, we were able to acquire a more comprehensive understanding of the subject matter and gather useful information that assisted us in making progress on the project. Additionally, we utilized multiple technology-related websites and YouTube, Google as resources for learning about the latest technological innovations relevant to the project. This helped us stay current on the newest methods and advancements in our field and improve our abilities in those areas. Overall, the project gave us the chance to apply and improve our research and analytical skills and become more proficient in our field of work.

13. References

S.No	Reference
[1]	Houle, David, et al. "Automated measurement of Drosophila wings." *BMC evolutionary biology* 3.1 (2003): 1-13.
[2]	Loh, Sheng Yang Michael, et al. "Semi-automated quantitative Drosophila wings measurements." *BMC Bioinformatics* 18.1 (2017): 1-14.
[3]	Kumar, Nilay, et al. "MAPPER: A new image analysis pipeline unmasks differential regulation of Drosophila wing features." *bioRxiv* (2020).
[4]	Leonardo, Matheus Macedo, et al. "Deep feature-based classifiers for fruit fly identification (Diptera: Tephritidae)." *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. IEEE, 2018.
[5]	Kwak, Jihoon, et al. "Hair Separation and Detection in Mutant Drosophila Hair Images." *Proceedings of the Korean Society for Industrial Applied Mathematics Conference* 5.1 (2010): 153-155.
[6]	Woodman, Peter N., Amy M. Todd, and Brian E. Staveley. "Eyer: automated counting of ommatidia using image processing techniques." *Drosophila Information Service* 94 (2011): 142.
[7]	Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." *2017 international conference on engineering and technology (ICET)*. Ieee, 2017.
[8]	O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." *arXiv preprint arXiv:1511.08458* (2015).
[9]	Cheng, Heng-Da, and X. J. Shi. "A simple and effective histogram equalization approach to image enhancement." *Digital signal processing* 14.2 (2004): 158-170.
[10]	Perreault, Simon, and Patrick Hébert. "Median filtering in constant time." *IEEE transactions on image processing* 16.9 (2007): 2389-2394.
[11]	Sauvola, Jaakko, et al. "Adaptive document binarization." *Proceedings of the Fourth International Conference on Document Analysis and Recognition*. Vol. 1. IEEE, 1997.
[12]	Di Stefano, Luigi, and Andrea Bulgarelli. "A simple and efficient connected components labeling algorithm." *Proceedings 10th international conference on image analysis and processing*. IEEE, 1999.
[13]	Abu-Ain, Waleed, et al. "Skeletonization algorithm for binary images." *Procedia Technology* 11 (2013): 704-709.
[14]	Chen, Qing, Xiaoli Yang, and Emil M. Petriu. "Watershed segmentation for binary images with different distance transforms." *Proceedings of the 3rd IEEE international workshop on haptic, audio and visual environments and their applications*. Vol. 2. 2004.

Plagiarism Report

Turnitin Originality Report

Processed on: 28-Jul-2023 01:32 PKT
ID: 2137700824
Word Count: 15755
Submitted: 1

CS14 Project Report By Shaiz Akber

Similarity Index	Similarity by Source	
17%	Internet Sources:	14%
	Publications:	4%
	Student Papers:	13%

5% match (student papers from 09-Jan-2023)
Submitted to Higher Education Commission Pakistan on 2023-01-09

1% match (Internet from 02-Nov-2022)
https://www.duhs.edu.pk/new/about-us/

1% match (Internet from 29-Nov-2011)
http://www.discover.uottawa.ca/~qchen/my_papers/have04_published_version.pdf

1% match (Internet from 29-Oct-2021)
https://www.slideshare.net/BhumikaPatel80/offline-character-recognition-for-handwritten-gujarati-text

< 1% match (student papers from 11-Jan-2023)
Submitted to Higher Education Commission Pakistan on 2023-01-11

< 1% match (student papers from 08-Apr-2013)
Submitted to Higher Education Commission Pakistan on 2013-04-08

< 1% match (student papers from 06-Aug-2018)
Submitted to Higher Education Commission Pakistan on 2018-08-06

< 1% match (Internet from 19-Jun-2023)
https://WWW.coursehero.com/file/161037801/DIT-0305-OBJECT-ORIENTED-ANALYSIS-AND-DESIGN-ASSIGNMENT-ONEdocx/

< 1% match (Internet from 28-Feb-2022)
https://www.coursehero.com/file/128713173/cell-nuclei-watershedpy/

< 1% match (Internet from 14-Aug-2022)
https://forum.opencv.org/t/segmentation-of-white-object-on-white-background/3886

< 1% match (Internet from 09-Apr-2023)
https://forum.opencv.org/t/opencv-js-guidance-on-most-appropriate-use-of-cv-findcontours/2061

< 1% match (student papers from 20-Dec-2022)
Submitted to Tresham College of Further and Higher Education on 2022-12-20

< 1% match (Internet from 20-Mar-2022)
https://pastebin.com/6yN5v7GU

< 1% match (Internet from 03-Aug-2022)
https://pastebin.com/R7Ug2SPK

< 1% match (Internet from 23-Feb-2010)
http://facweb.cs.depaul.edu/research/vc/seminar/Paper/Cheng_2004_Digital-Signal-Processing.pdf

< 1% match (Internet from 25-May-2023)
https://www.biorxiv.org/content/10.1101/2020.12.16.422888v1.full

< 1% match (Internet from 16-Mar-2020)
http://docplayer.net/32774576-Median-filtering-in-constant-time.html

< 1% match (Internet from 20-Nov-2020)
http://docplayer.net/7615812-Opencv-python-tutorials-documentation.html

< 1% match (student papers from 04-Nov-2019)
Submitted to Nelson Mandela Metropolitan University on 2019-11-04

< 1% match (Internet from 12-May-2022)
https://www.uj5u.com/qukuanlian/452764.html

< 1% match (Internet from 08-Jul-2023)

< 1% match (student papers from 08-May-2022)
Submitted to University of Nottingham on 2022-05-08

< 1% match (student papers from 01-Jul-2023)
Submitted to Southern New Hampshire University - Continuing Education on 2023-07-01

< 1% match (Internet from 01-Oct-2022)
https://web.archive.org/web/20170829024039if_/http:/marcin.szyderca.com/00797615.pdf

< 1% match (Internet from 02-Oct-2022)
https://web.archive.org/web/20180721161826if_/http:/www.journalijar.com/uploads/756_IJAR-18168.pdf

< 1% match (Internet from 26-Sep-2022)
https://thesai.org/Downloads/Volume12No5/Paper_41-Traffic_Engineering_in_Software_defined_Networks.pdf

< 1% match (Internet from 02-Mar-2023)
https://www.igi-global.com/viewtitle.aspx?TitleId=45468&isxn=9781615209118

< 1% match (student papers from 03-May-2019)
Submitted to University of Sydney on 2019-05-03

< 1% match (Internet from 16-Jan-2023)
https://vdocuments.mx/fyp-report-format-template.html

< 1% match (Internet from 07-Oct-2022)
https://vdocuments.mx/final-assig.html

< 1% match (student papers from 03-Aug-2022)
Submitted to Asia Pacific University College of Technology and Innovation (UCTI) on 2022-08-03

< 1% match (student papers from 23-Dec-2022)
Submitted to Asia Pacific University College of Technology and Innovation (UCTI) on 2022-12-23

< 1% match (A. Bulgarelli. "A simple and efficient connected components labeling algorithm", Proceedings 10th International Conference on Image Analysis and Processing ICIAP-99, 1999)
A. Bulgarelli. "A simple and efficient connected components labeling algorithm", Proceedings 10th International Conference on Image Analysis and Processing ICIAP-99, 1999

< 1% match (student papers from 02-Sep-2022)
Submitted to University College London on 2022-09-02

< 1% match (Internet from 29-Mar-2023)
https://community.intel.com/t5/OpenCL-for-CPU/How-to-install-OpenCL-CPU-runtime-18-1-without-removing-Intel/m-p/1172531

< 1% match (student papers from 21-Jan-2023)
Submitted to Cavite State University on 2023-01-21

< 1% match (Internet from 10-Sep-2020)
https://myfacesocks.jp/collections/boxers/products/custom-boxer-shorts-daruma

< 1% match (Ying-Hong Wang. "Image indexing and similarity retrieval based on a new spatial relation model", Proceedings 21st International Conference on Distributed Computing Systems Workshops, 2001)
Ying-Hong Wang. "Image indexing and similarity retrieval based on a new spatial relation model", Proceedings 21st International Conference on Distributed Computing Systems Workshops, 2001

< 1% match ()
Carbó Chicón, Albert. "Reducció del bor en l'aigua d'una dessaladora mitjançant extracció amb líquids iònics", Universitat Politècnica de Catalunya, 2012

< 1% match (student papers from 04-Nov-2011)
Submitted to University of Technology, Sydney on 2011-11-04

< 1% match (Internet from 13-Dec-2022)
https://nepis.epa.gov/Adobe/PDF/2000ZORL.pdf

< 1% match (Internet from 21-Dec-2022)
https://espace.etsmtl.ca/id/eprint/3125/1/ATAKY_Steve_Tsham_Mpinda.pdf

< 1% match (Internet from 06-Jan-2023)
https://www.turing.com/kb/how-to-use-python-for-server-side-language

< 1% match (student papers from 28-Mar-2019)
Submitted to City University of Hong Kong on 2019-03-28

< 1% match (student papers from 03-May-2023)

< 1% match (Internet from 26-Oct-2021)
https://drum.lib.umd.edu/bitstream/handle/1903/14625/Rueger_umd_0117E_14419.pdf;jsessionid=3AB61157EC33559870D19A82D8175E70?sequence%3D1=

< 1% match (Internet from 26-Dec-2022)
https://fdocuments.us/document/cs619-final-project-final-viva-preparation-khan-03334444071-page-1-of-49-cs619.html

< 1% match (Internet from 08-Dec-2022)
https://ipfs.io/ipfs/bafykbzacebcpvtdqdk2ppn5gwu53xb4b5yiexqn67fozapncgho3lyrhbaaro?filename=%5BLecture+Notes+in+Computer+Science+%E2%84%965857%5D+Daniel+Bernhardt%2C+Peter+Robinson+%28auth.%29%2C+Halimah+Visual+Informatics%3A+Bridging+Research+and+Practice%3A+First+International+Visual+Informatics+Conference%2C+IVIC+2009+Kuala+Lumpu13%2C+2009+Proceedings+%282009%2C+Springer%29+%5B10.1007%2F978-3-642-05036-7%5D.pdf

< 1% match (Internet from 22-Jul-2020)
https://stackoverflow.com/questions/48154642/how-to-count-number-of-dots-in-an-image-using-python-and-opencv/48155164

< 1% match (Internet from 10-Nov-2022)
https://ultraexotics.shop/which-of-the-following-does-the-object-below-represents.html

< 1% match (Internet from 28-Apr-2023)
https://www.geeksforgeeks.org/image-enhancement-techniques-using-opencv-python/?id=938107&type=article

< 1% match (Internet from 18-Nov-2021)
https://www.scitepress.org/Papers/2020/103770/pdf/index.html

< 1% match (Internet from 27-Aug-2022)
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/

< 1% match ("RFID Security and Privacy", Springer Science and Business Media LLC, 2008)
"RFID Security and Privacy", Springer Science and Business Media LLC, 2008

< 1% match ()
Nilay Kumar, Francisco J. Huizar, Keity J. Farfán-Pira, Pavel A. Brodskiy et al. "MAPPER: An Open-Source, High-Dimensional Image Analysis Pipeline Unmasks Differential Regulation of Wing Features", Frontiers in Genetics

< 1% match (Internet from 01-Nov-2022)
https://repositorio.usp.br/item/002931800