



Fayoum University



**Faculty of Computer
Science and Information**

Automated Intelligent Model for Students Concentration Measurement

By

**Osama Sayed Fathy
Kerellos Samy Labib
Ehdaa Osama Anwar
Toka Ashraf Mohamed
Norhan Mohamed Abbas
Marvy Magdy Ameen
Shahd Hesham Mohamed**

**CS Department
CS Department
IS Department
IS Department
IS Department
CS Department
CS Department**

Supervision

Dr. Shereen Ahmed

July | 2021

Abstract:

Increasing use of internet and improvements in the field of Information Technology has gradually increased the use of online meetings since last few years and more in a period of time when the whole world turned to work and study online due to COVID-19,. It has made communication easier, effective and efficient by using some web conferencing software like ezTalks Cloud Meeting, Skype, zoom and Google meet etc. Online meetings have provided a number of advantages especially to the business world as now they can communicate with their colleagues and associates situated at remote locations without moving out of their comfort zone to discuss business strategies and issues face-to-face. But where there are some advantages of online meetings there are some disadvantages also. Including difficulty for the teacher or manger (host) to determine whether meeting participants are focused or not. Many papers had discussed this option and provided some ideas such as using facial expression, mobile logs, and eye expression with Gaze Tracking which we think is the best one so we tackled this problem by Gaze Tracking and the use of EYEDIAP dataset which was designed to train and evaluate gaze estimation algorithms from RGB and RGB-D data, In the end, we achieved at an accuracy of 93%.

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deepest regards to our Diligent Supervisor **Dr. Shereen Ahmed** for her exemplary supervision and constant encouragement throughout the work of this Project. The blessing, help, and guidance given by her time to time shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express our gratitude to the Board of the Faculty of Computers and Information for their Support and efforts in providing us with all useful resources.

We express a Special thanks to the FCI stuff for the valuable information provided by them in their respective fields

Lastly, we thank Almighty, Our Parents and all the people who helped, supported and encouraged us to successfully finish the graduation project whether they were in the university or in the industry.

Table of Content

Content	Page
Abstract	2
Acknowledgment	3
Table of Content	4
List of Figures	5
List of Tables	6
Chapter 1: Introduction	7
1.1: Preamble	8
1.2: Project Motivation	8
1.3: Project Problem Statement	9
1.4: Project Aim and Objectives	9
1.5: Project Scope	9
1.6: Project Software and Hardware Requirements	10
1.7: Project Limitations	10
1.8: Project Expected Output	10
1.9: Project Schedule	11
Chapter 2: Literature Review	13
2.1: Introduction	14
2.2: Existing Systems	14
2.3: Overall Problems of Existing Systems	17
2.4: Overall Solution Approach	17
2.5: Summary	17
Chapter 3: System Analysis	19
3.1: Introduction	20
3.2: Data Flow Diagrams	21
3.3: UML Use Case Diagrams	24
3.4: UML Sequence and System Sequence Diagrams	30
3.5: Class Diagram	33
3.6: Requirements	33
3.7: Summary	35
Chapter 4: The Dataset	36
4.1: Introduction	37
4.2: Dataset Introduction	37
4.3: Setup of EYEDIAP Dataset	38
4.4: Data-Per-Session	39
4.5: Summary	41
Chapter 5: The Proposed System	42
5.1: Introduction	43
5.2: System Architecture	43

5.3: Frame Preprocessing	44
5.4: Building The Model	46
5.5: Focusing Classification	47
5.6: Summary	48
Chapter 6: System Implementation and Testing	49
6.1: Introduction	50
6.2: Backend Specification	50
6.3: System Testing and Validation	61
6.4: The Requirements Validation and Completeness	62
6.5: Summary	62
Chapter 7: Project Conclusion and Future Work	63
7.1: Project Conclusion	64
7.2: Future Work	64
References	65

[List of Figures](#)

Figure	Page
Figure 1.1: Project Schedule	11
Figure 1.2,3,4: Project Baseline	12
Figure 3.2.1: Context Diagram	21
Figure 3.2.2: DFD Level 0	22
Figure 3.2.3: DFD Level 1	23
Figure 3.2.4: DFD Level 2	23
Figure 3.3.1: Use Case Diagram	24
Figure 3.4.1: UML Sequence Diagram 1	30
Figure 3.4.2: UML Sequence Diagram 2	30
Figure 3.4.3: UML Sequence Diagram 3	31
Figure 3.4.4: UML Sequence Diagram 4	31
Figure 3.4.5: UML Sequence Diagram 5	32
Figure 3.4.6: UML Sequence Diagram 6	32
Figure 3.5.1: Class Diagram	33
Figure 4.3.1: Recording Setup for Dataset	38

Figure 4.4.1: Sample from EYEDIAP Dataset 1	39
Figure 4.4.2: Sample from EYEDIAP Dataset 2	39
Figure 4.4.3: Algorithm for Splitting the Video into Frames	40
Figure 5.2.1: System Architecture	43
Figure 5.3.2.2 : (a) Dlib face feature point positioning	45
Figure 5.3.2.2 : (a) face feature point positioning effect	45
Figure 6.2: Model's Code	50

List of Tables

Table	Page
Table 1: Log in Use Case	25
Table 2: Create Meeting Use Case	26
Table 3: Join Meeting Use Case	27
Table 4: Analyze Images Use Case	28
Table 5: Show Report Use Case	29
Table 6: Facial Landmarks Detectors Accuracy over Datasets	44
Table 7: Comparison Between Accuracy of Each Method	47

Chapter 1: Introduction

1.1- Preamble

The objective of the work outlined is detecting if the person is focusing or not, it can be useful for the educational field where the teacher can know if the student focusing during the lecture or not, or in the business field the manager can know the employee focusing time during work time in online work (Remotely work).

Deep Learning, Machine Learning, and image processing techniques rely on the approach of such a system. Our state-of-the-art Deep Learning approach is primarily aimed at achieving high precision of the focusing measure to provide all interested companies and organizations with the knowledge they need about staff or students.

We will talk in this chapter about the project from the user point of view, 1.2: motivation of the project, 1.3: the problem statement of the Project, 1.4: project aim and objectives, 1.5: project scope.

1.2- Project Motivation

The rationale for choosing this idea is the whole world was turned to online work and study due to COVID-19, which made a lot of confusion for all people such as teachers or professors who can't see student reactions or knowing how much they understand the lecture. After a long search, we have found an urgent need for such a system that uses the latest software and hardware advancements to detect if the person is focusing or not.

Solution's effect on society:

- Helping teachers or professors.
- Helping manager in companies.
- Make online communications more efficient.
- Gives useful reports.
- Increase people safety.
- Save precious time.

1.3- Project Problem Statement

After the emergence of the Coronavirus, the world turned to home quarantine to protect themselves from the risk of infection and used remote communication techniques to work and study, but this resulted in many problems due to such as not knowing if people focusing or not. Therefore, we want to solve this problem by adding a detection property, the ability to measure the concentration of people.

1.4- Project Aim and Objectives

The goals of that tool to accomplish within a timeline and with available resources are:

- Helping teachers or professors to be closer to students.
- Helping manager to monitoring their employees.
- Make online communications more efficient.
- Gives useful reports for business and organizations about time of focusing students or employees.
- Increase people safety by using online techniques instead of social affinity to avoid injury COVID-19.

1.5- Project Scope

The project will involve making model as an extension in desktop and mobile apps which are user-friendly and interactive such as zoom, Google meet, Skype, based on experience and user requirements. This Process would also include considering other related systems interactions such as camera or microphone that attached with pc or mobile phone, images and video recording during meeting that comes from mobile or PC.

Once the system is running during meeting, it would the capability to real-time process all persons that attend and detect if they are focusing depends on camera, microphone and mobile or computer logs, at the end of meeting it gives report about all attended persons.

In addition to this we will test the system for high accuracy classification of "people focusing measure" using various available datasets.

1.6: Project Software and Hardware Requirements

Our System needs special both software and hardware requirements

- Hardware Requirements:
 - Mobile or computer camera
 - A computer with high “GPU” for model training.
 - Microphone.
- Software Requirements:
 - Allow access to mobile logs for mobile.
 - Allow access to camera and microphone.

1.7: Project Limitations

Our Project Limitations are:

- ❖ Result Limitation
 - The accuracy of the measurement focusing system must be high.
- ❖ Time frame Limitation
 - The project must be finished by the end of work on 26 June.
- ❖ Resource limitation
 - Hardware Resources:
 - Mobile or computer camera
 - A computer with high “GPU” for model training.
 - Microphone.

1.8: Project Expected Output:

The expected output of the work outlined is to detect the attention of the participant, that can be beneficial for the educational field where the instructor can know the if the student focusing during the lesson, or the manager can know the employee focusing period during work in online work time in the business field (Remotely work).

1.9: Project Schedule:

➤ 1.9.1: Tasks

	Task Name	Duration	Start	Finish	Project Phase	Precedes	Resource Names
1	Graduation Project	.38 days?	Sun 7/12/20	Wed 7/14/21			labtob Dell core i5
2	Business Requirements	.13 days?	Sun 10/18/20	Wed 1/27/21			Google Colab
3	tools installation	1 day?	Sun 10/18/20	Mon 10/19/20			Toka,ehdaa,Norhan
4	fields Studing	60 days	Mon 10/19/20	Wed 1/27/21			Toka,Norhan,Ehdaa,Kerellos
5	System analysis	.13 days?	Thu 1/28/21	Mon 2/1/21			visual-paradigm,Lucidchart
6	DFD	1 day?	Thu 1/28/21	Thu 1/28/21			Norhan,Ehdaa
7	USE CASE	1 day?	Sun 1/31/21	Mon 2/1/21		6	Norhan,Ehdaa
8	USE CASE senario	1 day?	Mon 2/1/21	Mon 2/1/21			Norhan,Ehdaa
9	Design	2 days?	Tue 2/2/21	Wed 2/3/21			visual-paradigm
10	Sequence Digram	1 day?	Tue 2/2/21	Tue 2/2/21			Norhan,Ehdaa
11	Class digram	1 day?	Wed 2/3/21	Wed 2/3/21		10	Norhan,Ehdaa
12	Databse Model	1 day?	Thu 2/4/21	Thu 2/4/21			Lucidchart,visual-paradigm
13	Mapping	1 day?	Thu 2/4/21	Thu 2/4/21			Norhan,Ehdaa
14	Programming	.38 days?	Thu 2/11/21	Mon 5/17/21			Google Colab
15	Model Implementation	17.5 days?	Thu 2/11/21	Thu 3/11/21			Osama,Kerellos
16	Back End Programmii	.88 days?	Sun 3/14/21	Mon 5/17/21			Google Colab
17	Data Set collection	19.5 days?	Sun 3/14/21	Wed 4/14/21			Toka,Ehdaa
18	Video proceessing	7 days?	Thu 4/15/21	Mon 5/3/21		17	Toka,Ehdaa,Norhan,Marvy,shahd
19	System response	1 day?	Tue 5/4/21	Tue 5/4/21			Osama,Kerellos
20	Real Time Process	7 days?	Wed 5/5/21	Mon 5/17/21		19	Osama,Kerellos
21	Testing	.13 days?	Tue 5/18/21	Wed 6/9/21			Google Colab
22	Model Testing	7 days?	Tue 5/18/21	Thu 5/27/21			Osama,Kerellos
23	Back End Testing	7 days	Sun 5/30/21	Wed 6/9/21			Osama,Kerellos
24	Documentation	7 days	Thu 6/10/21	Tue 6/22/21			Toka,Ehdaa,Norhan,Kerellos,Osama
25	Meetings	3.88 days	Sun 7/12/20	Wed 7/14/21			
26	Exams	20 days	Thu 6/10/21	Wed 7/14/21			
27	final_Project disscussion	0 days	Sun 7/12/20	Sun 7/12/20			

Figure 1.1

➤ 1.9.2: Baseline

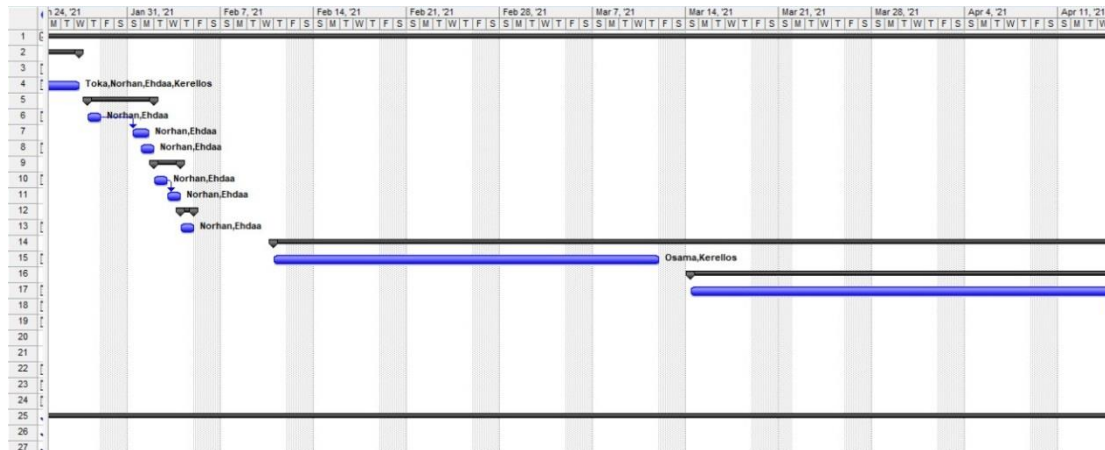
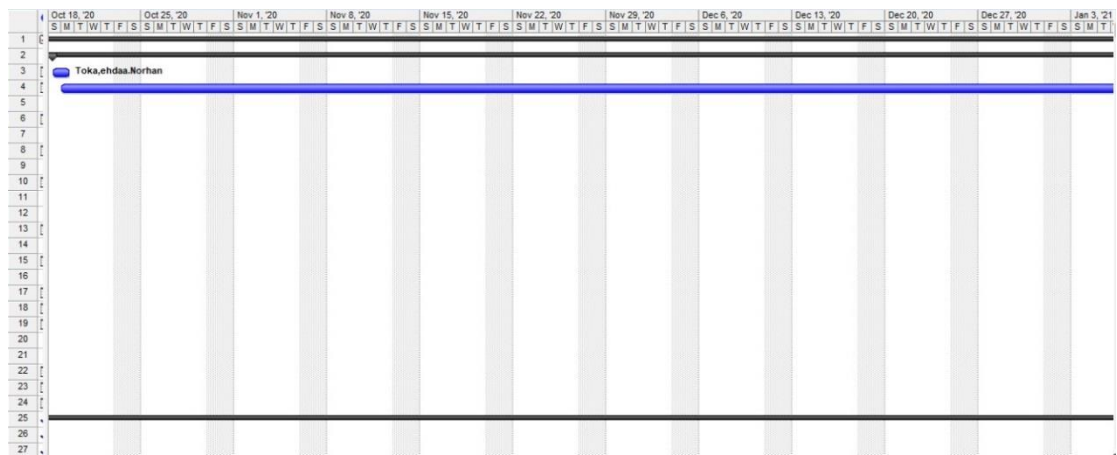


Figure 1.2



Chapter 2: Literature Review

2.1. Introduction

This chapter discusses other similar current systems and reflects on how the issue is solved by these systems, providing their solutions, these systems' overall challenges, and the overall solution method.

This chapter is organized into several sections 2.2: Existing Systems, 2.2.1: Facial Expression Recognition Systems.

2.2. Existing Systems

In the last year, we had seen what happened all over the world because of COVID-19 which forced all companies and organizations (like school and collages) to continue their work online using available online meeting applications; like zoom and google meet, but these applications lack some options that are needed to facilitate their work and nearly simulate their traditional offline work. These options include the ability to evaluate student's attention during online class and employee's attention during online working. Many papers had discussed this option and provided some ideas such as using facial expression, mobile logs, and eye expression with Gaze Tracking which we think is the best one so we conducted our system based on it.

Part I: Software Competitors Systems in Facial Expression Recognition

1- (Development of system for human Emotional analysis using facial recognition). (2020) [3]

Emotion or facial recognition system is based on processing the movement of facial muscles and divides face into parts then analyses each part separately. Accuracy for 4 emotions (happy, angry, surprised, disgust) is 80%. Accuracy of Classification of the six basic emotions by utilization of rule-based system is 88%.

The system based on image processing by using machine learning library when extraction of the muscle movement by utilization of "optical flow" and the classification using "K-nearest neighbor". It used Neural network with the help of Keras Model, Training set contains Images, & Test set is a live video to build the model with using algorithms which have been used in LBPH: Local binary pattern histogram which is a part of open CV and K-nearest neighbor. The Extraction of the form and the movements of the mouse, eye and eyebrow was done by using "parametric models". It makes too much processing to get human behavior.

2- Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks (2020) [2]

The system can be easily adapted for various consumer applications such as face detection-based home automation, device control, attendance system, intruder detection etc. It proposed to design and evaluation of a real-time face recognition system. Maximum recognition accuracy of 98.75% and 98.00% is obtained from the proposed system on using “AT&T” and real-time inputs respectively.

This system is close to what we need, it using “Convolutional Neural Networks”. The performance of proposed system and CNN architecture is evaluated by tuning various parameters of CNN to enhance the recognition accuracy of the system designed. The initial evaluation of the proposed design is carried out using standard “AT&T dataset” and the same is later extended towards the design of a real time system but it works on face recognition not human behavior.

3- Human Behavior Understanding in Big Multimedia Data Using CNN based Facial Recognition Expression (2019) [6]

This System dealt with human behavior analysis in a novel manner by analyzing facial expression of individuals in a famous TV-series. to achieve precise predictions of the behavior of an individual. Accuracy of the System: the average accuracy of the proposed facial recognition can be calculated as 89.85%.

This System is based on just Image Processing techniques using machine learning library, it using “Viola-Jones algorithm” to detected the faces and then tracked each one using “KLT” tracker. Then it utilized HOG features with “SVM” to classify faces of different actors. Next, it presented an efficient “CNN” particularly trained over “KDEF dataset” with extensive level of data augmentation. which is good but will be bad for real-time processing because person may move fast and model can’t detect his focus.

Part II: Software Competitors Systems in Gaze Tracking (Eye Tracking)

1- Recurrent CNN for 3D Gaze Estimation using Appearance and Shape Cues (2018) [9]

This system tackles the problem of person- and head pose independent 3D gaze estimation from remote cameras.

It is used a “Multi-modal recurrent convolutional neural network (CNN).” it proposes to combine face, eyes region, and face landmarks as individual streams in a CNN to estimate gaze in still images. Then, it exploits the dynamic nature of gaze by feeding the learned features of all the frames in a sequence to a many-to-one recurrent module that predicts the 3D gaze vector of the last frame. Its multi-modal static solution is evaluated on a wide range of head poses and gaze directions, achieving a significant improvement of “14.6%” over the state of the art on “EYEDIAP” dataset, further improved by 4% when the temporal modality is included, it showed that adding geometry features to appearance-based methods has regularizing effect on the accuracy.

Adding sequential information further benefits the final performance compared to static-only input, 3D gaze direction, unrestricted gaze target, Full face, Eye region, Facial landmarks Sequential information, but dataset collected and evaluated exclusively under controlled laboratory conditions.

2- It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation (2017) [12]

This system works on study of full-face, that leveraged information from the full face. They demonstrated that, compared to current eye-only and multi-region methods, their method is more robust to facial appearance variation caused by extreme head pose and gaze directions as well as illumination.

Appearance-based gaze estimation and proposed spatial weights CNN for full-face appearance-based 2D and 3D gaze estimation. Their method achieved an accuracy of 4.8° and 6.0° for person-independent 3D gaze estimation on the challenging in-the-wild MPIIGaze and EYEDIAP datasets, respectively – a significant improvement of 14.3% and 27.7% over the state of the art.

3- Eye Tracking for Everyone (2016) [14]

This work introduced an end-to-end eye tracking solution targeting mobile devices. First, they introduced GazeCapture, the first large-scale mobile eye tracking dataset. Then, using GazeCapture they trained iTracker, Through careful evaluation, they show that iTracker is capable of robustly predicting gaze, achieving an error as low as 1.04cm and 1.69cm on mobile phones and tablets respectively.

They report the performance of applying various state-of-the-art approaches (TabletGaze, TurkerGaze and MPIIGaze) and other baseline methods for comparison. They propose two simple baseline methods: (1) center prediction (i.e., always predicting the center of the screen regardless of the data) and (2) applying support vector regression (SVR) to image features extracted using AlexNet pre-trained on ImageNet. Interestingly, they find that the AlexNet + SVR approach outperforms all existing state-of-the-art approaches despite the features being trained for a completely different task. Importantly, they find that the features from iTracker significantly outperform all existing approaches to achieve an error of 2.58cm demonstrating the generalization ability of their features.

4- Appearance-Based Gaze Estimation in the Wild (2015) [15]

This system works on appearance-based gaze estimation in the unconstrained daily-life setting. It built a novel in-the-wild gaze dataset through a long-term data collection using laptops, which shows significantly larger variations in eye appearance than existing datasets. this work and its dataset provide a critical insight on addressing grand challenges in daily-life gaze interaction.

Throughout the comprehensive benchmarking of image-based monocular gaze estimation methods. Its CNN-based estimation model significantly outperforms state-of-the-art methods in the most challenging person- and pose-independent training scenario, in addition to its CNN-based method, they evaluate the following baseline methods using the same facial landmark detection, head pose estimation, and input features. “Random Forests (RF), k-Nearest Neighbours (kNN), Adaptive Linear Regression (ALR), Support Vector Regression (SVR), Shape-Based Approach (Eye Tab)”.

its CNN-based approach shows the best accuracy on both datasets (13.9 degrees on MPIIGaze, 10.5 degrees on Eyediap), with a significant performance gain (10% on MPIIGaze, 12% on Eyediap, paired Wilcoxon test, $p < 0.05$) over the state-of-the-art RF method. However, performance on MPIIGaze is generally worse than on the Eyediap dataset, which indicates the fundamental difficulty of the in-the-wild setting.

2.3: Overall Problems of Existing Systems

These are not a good method to achieve our goal (measure person focus) because of many problems such as:

- These systems depend on eye tracking without considering head position which decreases their accuracy.
- The systems haven't set a threshold to detect if the person is focusing or not.
- Dataset collected and evaluated exclusively under different conditions which decrease their accuracy.

2.4: Overall Solution Approach

Our System is a real time system using software and hardware developments for processing based on Deep Learning techniques, using laptops or mobiles cameras real-time videos as input to our system we can segment those videos to measure person focus.

2.5: Summary

In this chapter, we analyzed the current systems and the rivals who have the same thinking as us, presented the overall issues of these systems, and presented Our Solution Approach.

- ❖ Traditional Existing System
 - Using current and traditional online solutions such as zoom and google meet to make their meetings.
- ❖ Software Competitors Systems in Facial Expression Recognition
 - Development of system for human Emotional analysis using facial recognition (2020).
 - It makes too much processing to get human behavior.

- Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks (2020).
 - The system works on face recognition not human behavior.
- Human Behavior Understanding in Big Multimedia Data Using CNN based Facial Recognition Expression (2019)
 - This System is based on just Image Processing techniques such as SVM and CNN.
- ❖ Software Competitors Systems in Gaze Tracking (Eye Tracking)
 - Recurrent CNN for 3D Gaze Estimation using Appearance and Shape Cues (2018)
 - Dataset collected and evaluated exclusively under controlled laboratory conditions.
 - It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation (2017)
 - This system it makes too much processing to get face tracking.
 - Eye Tracking for Everyone (2016)
 - This system not specified in one thing it work in general to tracking eye that cause to low accuracy.
 - Appearance-Based Gaze Estimation in the Wild (2015)
 - This System is based on just Image Processing techniques such as SVM and CNN.
- ❖ Overall Problems of Existing Systems
- ❖ Overall Solution Approach

Our Proposed System:

- The System is using a detector to detect the facial landmarks for each frame in the video
- Make a red rectangle over the face and both the two eyes
- Crop the two eyes from the face
- Concatenate them together into one image (Focused or Unfocused image)
- The concatenated images enters the model
- The Model determines whether the image is Focused or Unfocused according to the eyes position and whether the student looks at the monitor directly or not.

Chapter 3: System Analysis

3.1: Introduction

This chapter reviews System USE-CASE Scenario and USE-CASE Diagram which is a list of actions or evented steps typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system to achieve a goal, Sequence Diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur, System Sequence Diagram which is a sequence diagram that shows, for a particular scenario of a use-case, the events that external actors generate, their order, and possible inter-system events, Class Diagram which is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects, Context Diagram which presents the sub-systems of our system and its data flow processing, and system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

It is organized into 3.2: System Architecture, 3.3: Data Flow Diagrams, 3.4: UML Use case Diagram, 3.5: UML Sequence and System Sequence Diagrams, 3.6: UML Class Diagram, 3.7 Requirements, 3.8: Summary.

3.2: Data Flow Diagrams

3.2.1: Context Diagram

Context Diagram

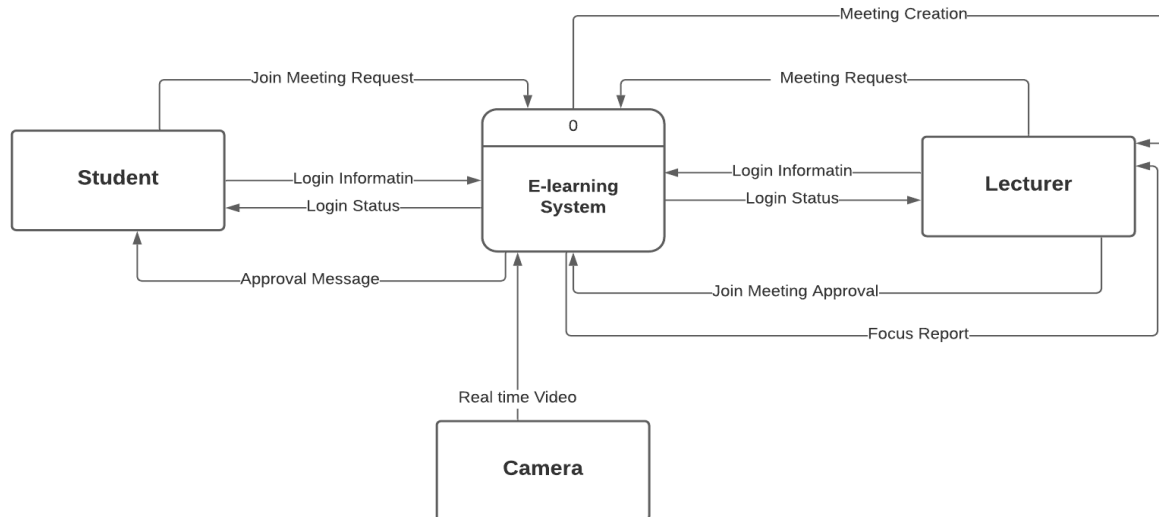


Figure 3.2.1

3.2.2: DFD Level 0

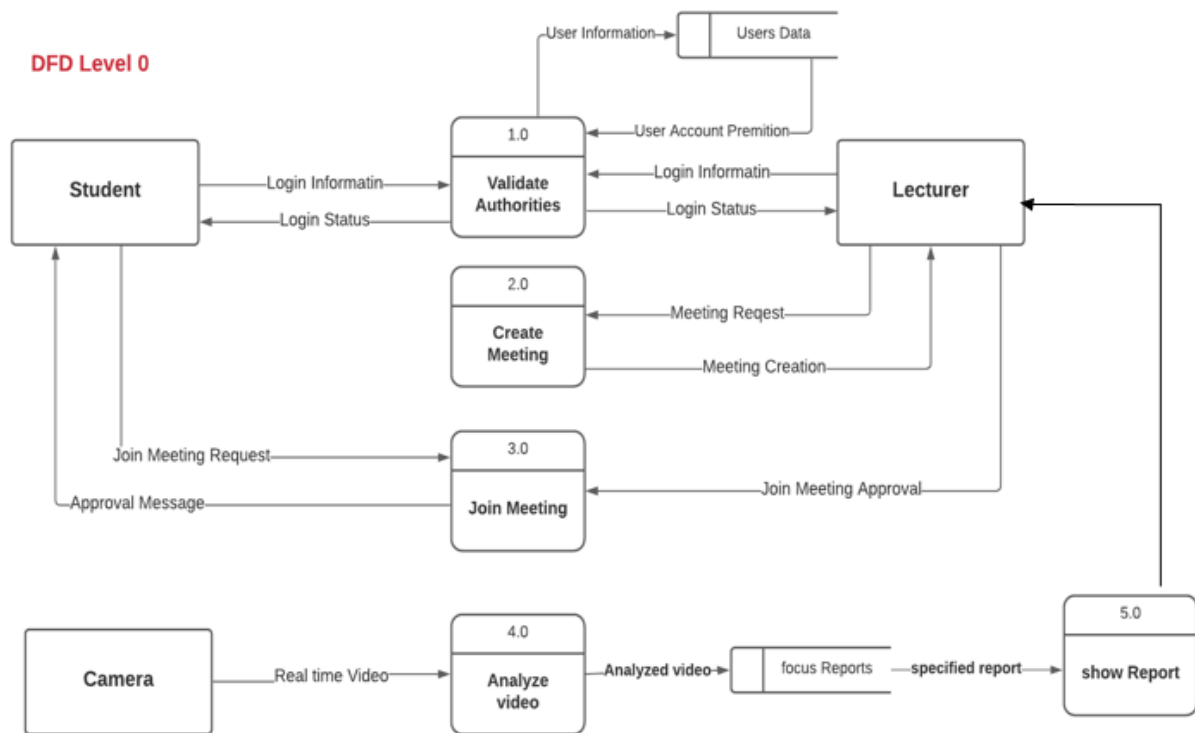


Figure 1.2.2

3.2.3: DFD Level 1

DFD level 1

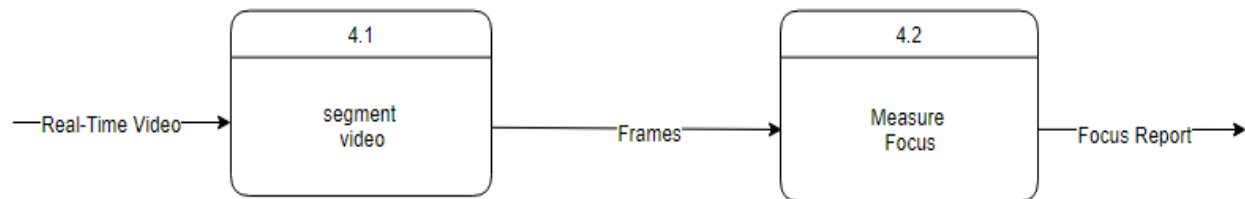


Figure 3.2.3

3.2.4: DFD Level 2

DFD level 2

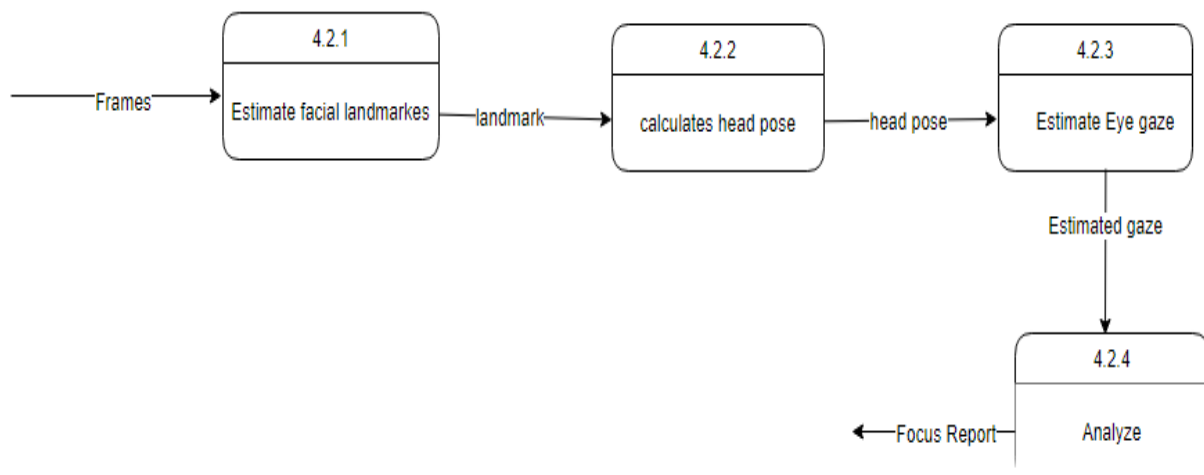


Figure 3.2.4

3.3: UML Use case Diagram

3.3.1: Use case Diagram

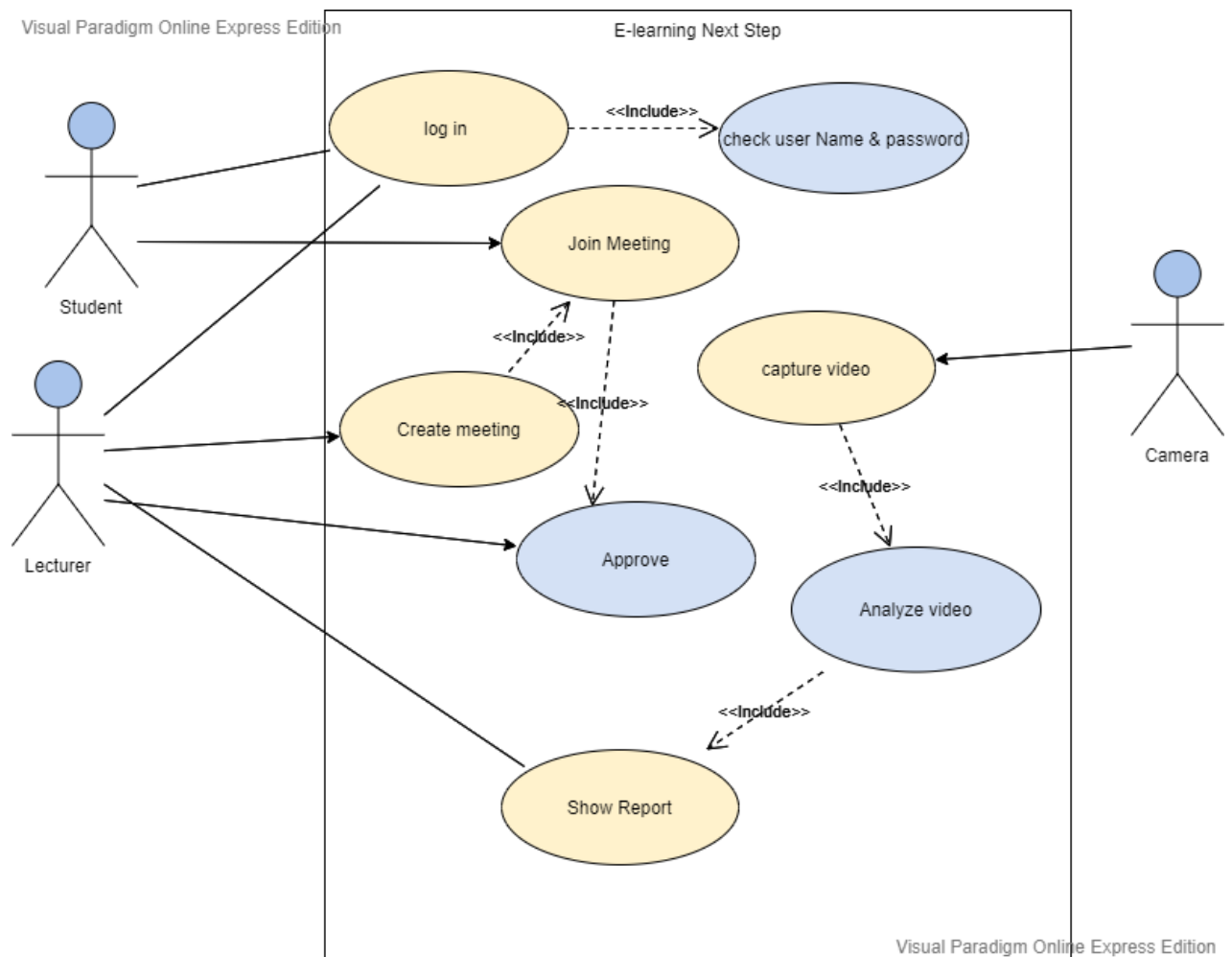


Figure 3.3.1

3.3.2: Use case Scenario:

1-Use Case Name: Login

Actor	Student & lecturer	
Description	This use case describes the process of Student, lecturer Logins into the System, The Student, and lecturer will receive a Notification that he was Successfully Logged or not.	
Typical Course of Events	Actor Action	System Response
	Step 1: This use case is initiated when Student & lecturer enter their Information to login into the system.	Step 2: The System would Check in the DB For the Student & lecturer Data. Step 3: the System would send a Confirmation message to the Student & lecturer.
Alternate Course of Events	If there is a missing or incorrect information, call this use case to revise data. If the Data was not enrolled in the DB the System would send Incorrect message.	
Pre-condition	Student & lecturer have to be successfully registered.	
Post condition	Student & lecturer have been successfully logged	

2-Use Case Name: Create Meeting

Actor	Lecturer	
Description	This use case describes the process of lecturer create meeting on the System, lecturer will receive a Notification that the meeting was Successfully created or not.	
Typical Course of Events	Actor Action	System Response
	Step 1: This use case is initiated when lecturer request to create meeting on the system.	Step 2: The System would create The meeting Step 3: the System would send notification that the meeting was Successfully created
Alternate Course of Events	If there is a problem in internet connection, call this use case to send a notification	
Pre-condition	Lecturer has to be successfully registered.	
Post condition	Meeting has been successfully created	

3-Use Case Name: join Meeting & Approve

Actor	student & lecturer	
Description	This use case describes the process of student join meeting on the System, lecturer will receive a Notification that there is a student in the waiting room, lecturer approve ,then student receive the approval message and join the meeting	
Typical Course of Events	Actor Action	System Response
	Step 1: This use case is initiated when student request to join meeting on the system. Step 3: , lecturer approve student's request	Step 2: The System would send alert to lecturer that there is a student in the waiting room Step 4: the System would send approval message to student
Alternate Course of Events	If there is a problem in internet connection, call this use case to send a notification	
Pre-condition	Lecturer & Student has to be successfully registered.	
Post condition	student has been successfully join to meeting	

4-Use Case Name: Analyze images

Actor		
Description	This use case describes the process of Manipulation the images taken from Camera to measure focus.	
Typical Course of Events	Actor Action	System Response
	Step 1: The System is manipulating the video taken from camera.	Step 2: The system divides the video into frames. Step 3: The System is manipulating the frames to measure focus Step 4: the system sends the report to lecturer to make a decision.
Pre-condition	Camera has to send a real-time video.	
Post condition	The System measure focus successfully, and send the report to lecturer, then lecturer can make a decision.	

5-Use Case Name: Show Report

Actor	Lecturer	
Description	This use case describes the process of lecturer getting the Reports from the Database.	
Typical Course of Events	Actor Action	System Response
	Step 1: This use case initiated when a lecturer orders to get a report. Step 4: The lecturer show the report.	Step 2: The System gets the Required reports from the Database. Step 3: The System sends the reports to the lecturer
Alternate Course of Events	If there is a missing or incorrect information, call this use case to revise data.	
Pre-condition	The lecturer's account has to be validated.	
Post condition	The Reports has been sent Successfully.	

3.4: UML Sequence and System Sequence Diagrams

3.4.1: UML Sequence Diagram

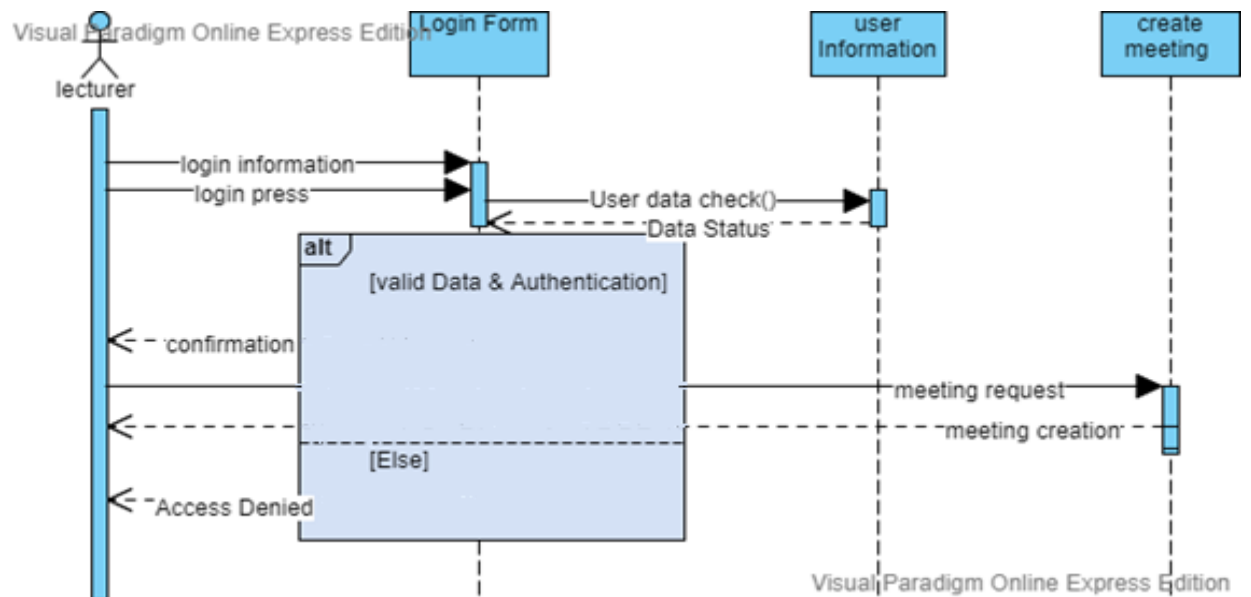


Figure 3.4.1

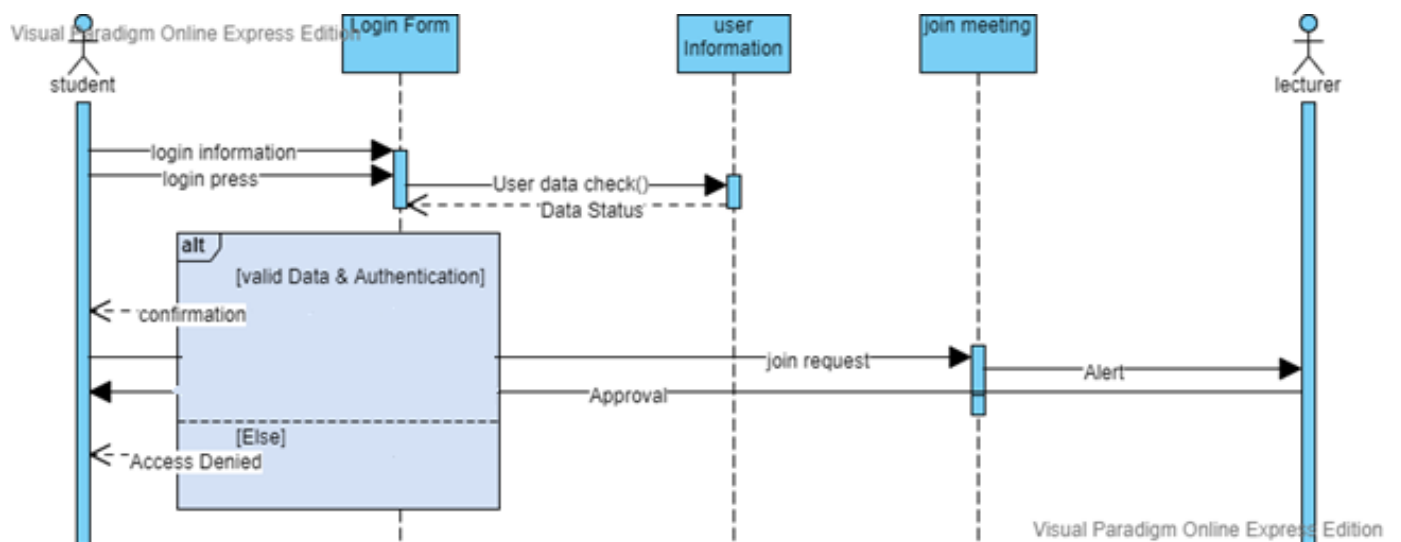


Figure 3.4.2

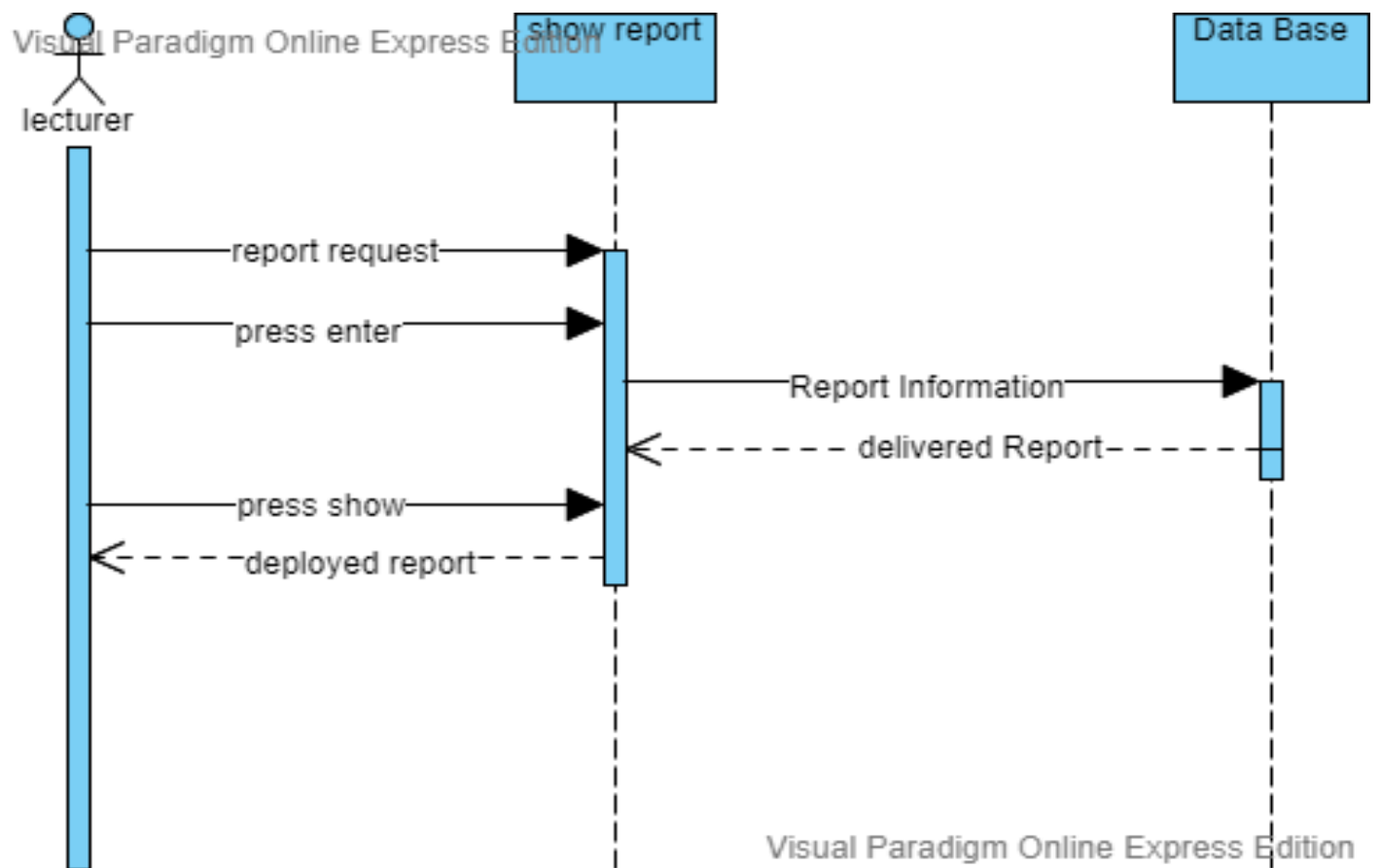


Figure 3.4.3

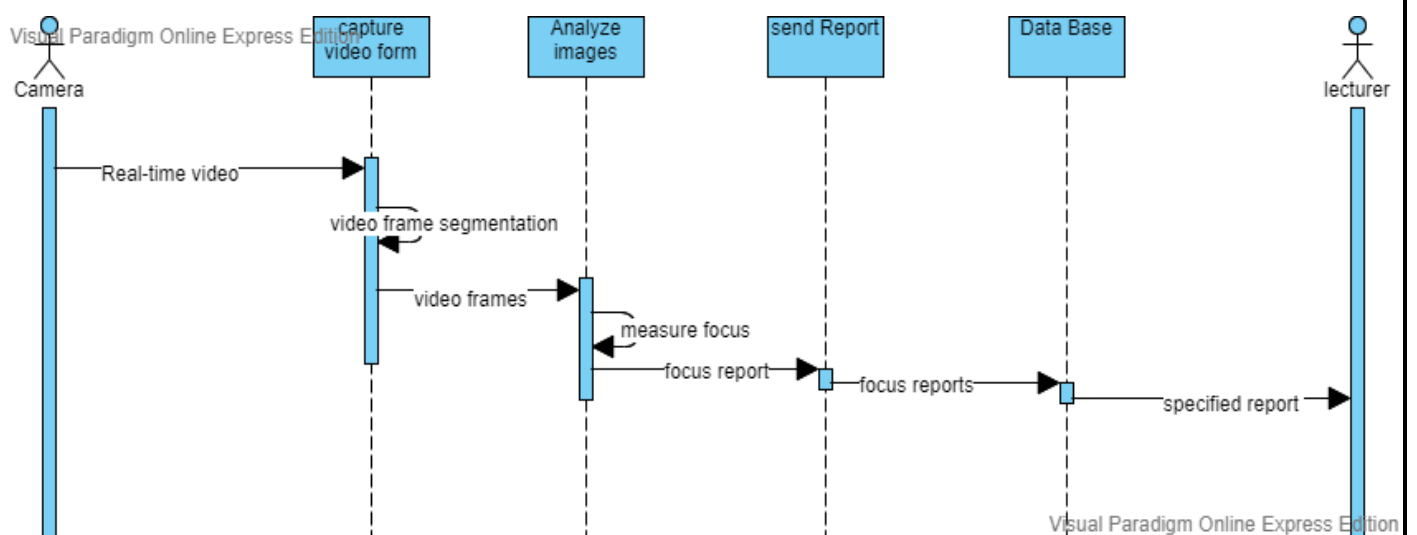


Figure 3.4.4

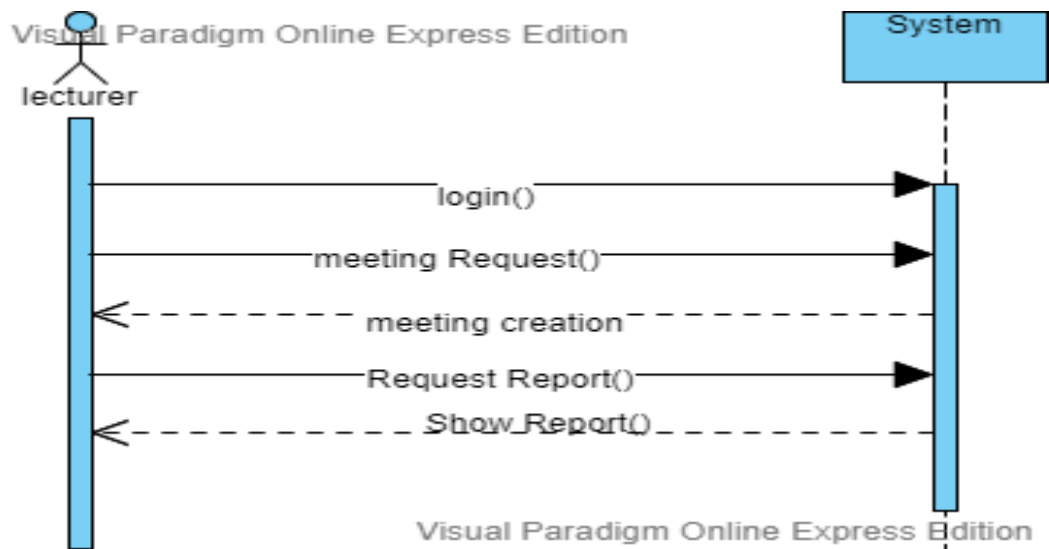


Figure 3.4.5

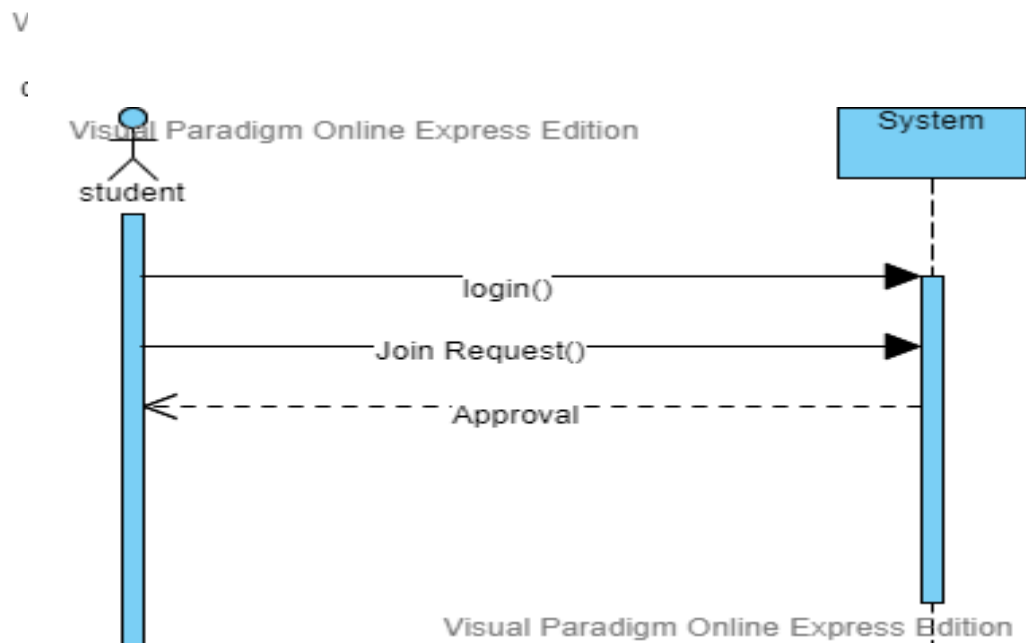


Figure 3.4.6

3.5: Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.

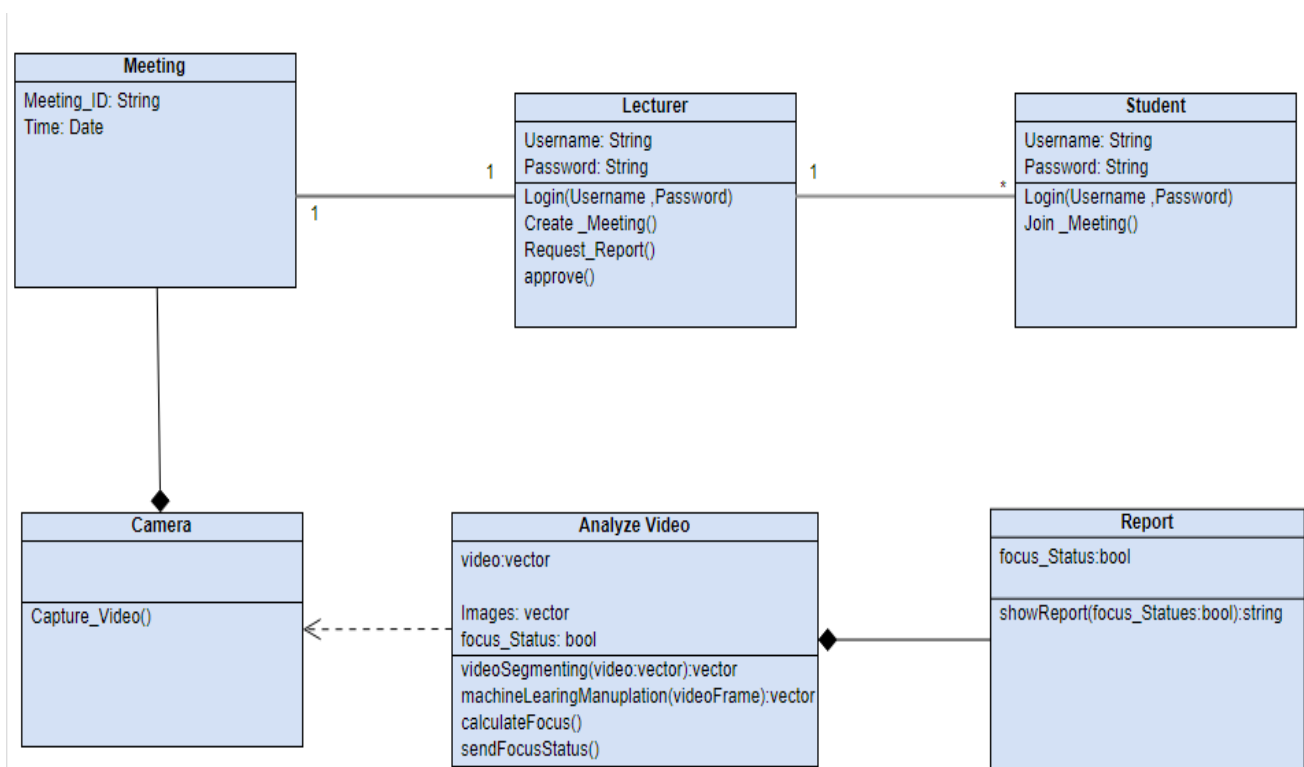


Figure 3.5.1

3.6: Requirements:

3.6.1: Functional Requirements

Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements.

- Make online communications more efficient.
- The system shall monitor meetings.
- Measuring person focus during the lecture, or in the business (Remotely work).

- Gives useful reports for business and organizations about time of focusing students or employees.
- The project will involve making desktop and mobile apps user-friendly and interactive.
- The system must have an unbiased dataset.
- The system must be correctly able to load the face classifier model.
- Load face image as input.
- Frame Preprocessing
- Detecting the frame facial landmarks
- Perform image processing.
- The system must be able to detect face on human faces & eyes on every frame in a live video.
- The results must be viewed by showing the probability along with the output of 'Focus' or 'Unfocused'.

3.6.2: Non-Functional Requirements

One of the best ways to gather requirements is to get all of the stakeholders together for a guided brainstorming session. Remember that in many cases the upper-level stakeholders are not the users. Include user representatives on the team, who are one of the best sources for non-functional requirements.

- **Usability:** The system should be usable.
- **Reliability:** meeting should be accurately monitored, especially when estimated result of focusing.
- **Accessibility:** the manager or teacher can know the percentage of employee focusing time during work time in online work (Remotely work).
- **Flexibility:** The system should be flexible enough to accommodate evolving data - e.g., the sets of concerned participants.
- **Performance:** The amount of participants, fast send report with result of focusing of participants.
- **Accuracy:** Accurate result for each participant in the meeting about his concentration/focusing during the lecture.
- **Security:** Only host (teacher & manager) has the authority to view the focus reports of the participants.

3.7: Summary

In this Chapter we have discussed System Architecture and System Design which includes System Data Flow Diagrams, UML Use case Diagram, UML Sequence and System Sequence Diagrams, UML Class Diagram It's organized in the following manner:

- ❖ 3.2:Data Flow Diagram
 - Context Diagram
 - DFD Diagram
- ❖ 3.3:UML Use Case Diagram
 - Use Case Diagram
 - Use Case scenario
- ❖ 3.4: UML Sequence and System Sequence Diagrams
 - UML Sequence Diagram
 - UML System Sequence Diagram
- ❖ 3.5: UML Class Diagram.
- ❖ 3.6: Requirements.
 - Functional Requirements.
 - Non- Functional Requirements.

Chapter 4: The Dataset

4.1: Introduction

This chapter reviews the Dataset information and Set-up of the dataset.

The chapter is organized into 4.2: Dataset Introduction, 4.3: Set-up of EYEDIAP dataset, 4.4: Data-Per-Session, 4.5: Summary.

4.2: Dataset Introduction

The dataset we had used in our model is **EYEDIAP** dataset which was designed to train and evaluate gaze estimation algorithms from RGB and RGB-D data. It contains a diversity of participants, head poses, gaze targets and sensing conditions.

The **EYEDIAP** dataset is a dataset for gaze estimation from remote RGB, and RGB-D (standard vision and depth), cameras. The recording methodology was designed by systematically including, and isolating, most of the variables which affect the remote gaze estimation algorithms:

- Head pose variations.
- Person variation.
- Changes in ambient and sensing condition.
- Types of target: screen or 3D object.

4.3: Set-up of EYEDIAP dataset

The set-up is composed of the following elements:

- Kinect: this consumer device provides standard (RGB) and Depth video streams at VGA resolution (640x480) and 30fps.
- HD camera: the Kinect was designed with a large field of view imposing less restriction on user mobility but this is problematic for eye tracking based on VGA resolution. Therefore, we also recorded the scene with a full HD camera (1920x1080) at 25fps. The provided videos are synchronized with the Kinect data at 30fps.
- LEDs: 5 LEDs visible to both cameras were used to synchronize the RGB-D and HD streams.
- Flat screen: we used a 24" screen to display a visual target.
- Small ball: we used a 4cm diameter ball as a visual target with a double purpose: to serve as a visual target in a 3D environment and be discriminative in both RGB and depth data such that its 3D position could be precisely tracked

This set-up was used for the recording of 94 sessions, each with different characteristics

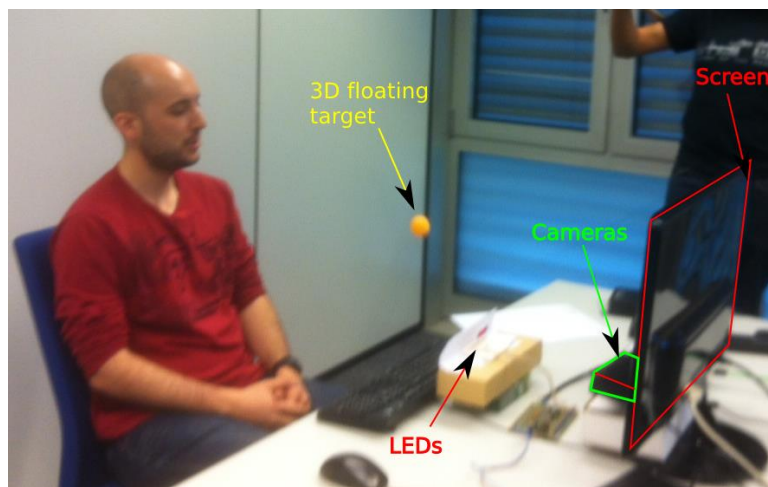


Figure 4.3.1: Recording Setup for the dataset

4.4: Data Per Session

At each session folder, the following files can be found:

- rgb_vga.mov: the RGB Kinect video. Encoded using MPEG-4.
- depth.mov: the Depth video. Encoded using ZLIB.
- rgb_hd.mov: the RGB HD video. Encoded using MPEG-4 (The HD video is not available for a few sessions).
- head_pose.txt: the frame-by-frame head pose parameters.
- eye_tracking.txt: the frame-by-frame 2D and 3D eyes position.
- ball_tracking.txt: the frame-by-frame 2D and 3D position of the ball target (if relevant).
- screen_coordinates.txt: the frame-by-frame 2D and 3D screen coordinates (if relevant).
- rgb_vga_calibration.txt: the calibration parameters for the RGB Kinect camera.
- depth_calibration.txt: the calibration parameters for the Depth camera.
- rgb_hd_calibration.txt: the calibration parameters for the RGB HD camera.

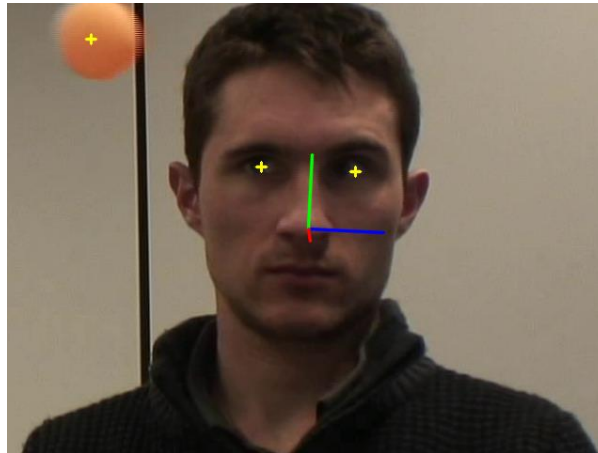


Figure 4.4.1: 1_a_ft_rgb_hd_443



Figure 4.4.2: 1_a_ft_depth_443

Each recording session is of a combination of the following parameters:

- **Participants.** We have recorded 16 people: 12 male and 4 female.
- **Recording conditions.** For participant 14, 15 and 16, some sessions were recorded twice, in two different conditions (denoted A or B): different day, illumination and distance to the camera.
- **Visual Target.** It is the object which the participant was requested to gaze at. To be representative of different applications, we included the following cases: Discrete screen target (DS), where a small circle was uniformly drawn every 1.1 seconds on random locations in the computer screen; Continuous screen target (CS), in which the circle was programmed to move along a random trajectory for 2s, to obtain examples with smoother gaze movement; 3D floating target (FT): a ball with a 4cm diameter hanging from a thin thread attached to a stick that was moved within a 3D region between the camera and the participant. In contrast to the screen target, the participant was at a larger distance (1.2m instead of 80-90cm) from the camera to allow sufficient space for the target to move.
- **Head pose.** To evaluate methods in terms of robustness to head pose, we asked participants to keep gazing at the visual target while (i) keeping an approximately static head pose facing towards the screen (Static case, S); or (ii) performing head movements (translation and rotation) to introduce head pose variations (Mobile case, M).

Each session is denoted by the string "P-C-T-H" which refers to the participant P=(1-16), the recording conditions C=(A or B), the target T=(DS, CS or FT) and the head pose H=(S or M) respectively

Our Model is working on images so we needed to divide the dataset's videos into frames, so we used the code in figure () to get these frames, the code take frame each millisecond because the pose of the gaze change each millisecond.

```
2 import cv2
3 import numpy as np
4 import os
5
6 # Playing video from file:
7 cap = cv2.VideoCapture('set1.mp4')
8
9 try:
10     if not os.path.exists('data'):
11         os.makedirs('data')
12 except OSError:
13     print ('Error: Creating directory of data')
14
15 currentFrame = 0
16 while(True):
17     # Capture frame-by-frame
18     ret, frame = cap.read()
19
20     # Saves image of the current frame in jpg file
21     name = './data/frame' + str(currentFrame) + '.jpg'
22     print ('Creating...' + name)
23     cv2.imwrite(name, frame)
24
25     # To stop duplicate images
26     currentFrame += 1
27
28 # When everything done, release the capture
29 cap.release()
30 cv2.destroyAllWindows()
```

Each video divided into around 4000 frame. Then these frames classified into two categories (Focused – Not Focused)

Figure 4.4.3: Algorithm for splitting the video into many frames

4.5: Summary

In This chapter we have reviewed the Dataset information and Set-up of the dataset.

The chapter is organized in the following manner:

- 4.2: Dataset Introduction
- 4.3: Set-up of EYEDIAP dataset
- 4.4: Data-Per-Session

Chapter 5: The Proposed System

5.1: Introduction

This chapter reviews the architecture of the proposed system, Frame Preprocessing, Building the Model and Focusing Classification.

The chapter is organized into 5.2: System Architecture, 5.3: Frame Preprocessing, 5.4: Building the model, 5.5: Focusing Classification, 5.6: Summary.

5.2: System Architecture

The Following diagram explains in details the major steps of our system

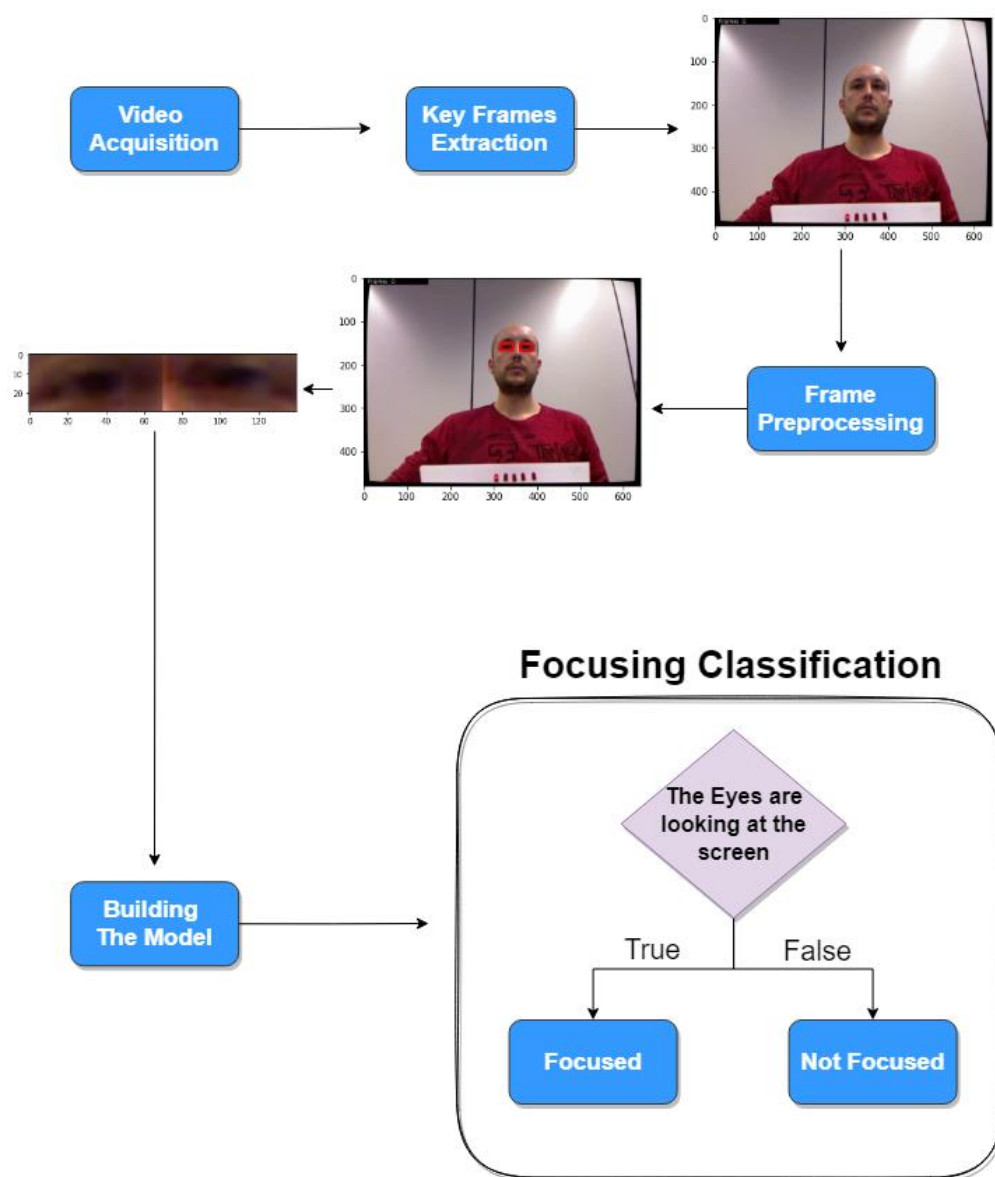


Figure 5.2.1: The System Architecture

Figure 5.2.1 shows the block diagram of the proposed system. As can be seen, first, the student's face is captured by the camera and converted into many frames. In the second step, each frame in the video is preprocessed and in the final third step, the frames enter into the model to detect whether the student is focused or not.

5.3: Frame Preprocessing

5.3.1: Frame Preprocessing Definition

Frame pre-processing is the term for operations on frames, these operations do not increase frame information content but they decrease it.

The aim of pre-processing is an improvement of the frame data that suppresses undesired distortions or enhances some frame features relevant for further processing and analysis task.

In the frame preprocessing we are using the facial recognition technique which is known as a computer technique that uses artificial intelligence (AI) to recognize and identify human faces in digital photographs. Face detection technology can be used in a variety of industries to enable real-time surveillance and tracking of people, including security, biometrics, law enforcement, entertainment, and personal safety.

5.3.2: Detecting the frame facial landmarks

5.3.2.1: Different facial landmarks detectors

Figure 5.3.2.1 summarize the results obtained from applying each of these face detectors to the datasets which are **BioID**, **MUCT**, **HELEN**, **300W** and **Menpo**.

With some images in the HELEN, 300W and Menpo datasets containing multiple faces. Due to the lack of bounding boxes or landmarks for the additional faces, the results count the additional faces in the image which are not the ground truth as false positives.

Detector	BioID (N= 1521)		MUCT (N=3755)		HELEN# (N= 2330)		300W# (N= 600)		Menpo (Front)# (N=6679)	
	% Acc	False P	% Acc	False P	% Acc	False P	% Acc	False P	% Acc	False P
OpenCV 1	96.45	184	97.95	616	89.06	3548	75.33	6203	73.24	2775
OpenCV 2	96.12	28	98.24	77	84.64	447	71.17	3574	67.35	387
OpenCV 3	96.45	52	98.62	135	85.88	821	72.00	3966	69.35	670
OpenCV 4	N/A	N/A	N/A	N/A	27.21	274	24.83	1597	22.74	349
Dlib	99.34	1	99.89	6	96.82	351	88.17	2488	87.50	339

*Some images in this dataset contain more faces than just the ground truth. These additional faces may be recorded as a false positive result

Figure 5.3.2.1: Table of the facial landmarks detectors accuracy over many datasets

The results of **Figure 5.3.2.1** show that all of the frontal face detectors performed best when applied to the BioID and MUCT datasets, as these images were taken in highly controlled environments with consistent lighting, pose, and expression. In contrast, the detectors performed poorly on images from uncontrolled environments, dropping to 67.35% accuracy in the frontal Menpo set.

It was also observed that the **Dlib detector** has the best accuracy in all examples and with fewer false positives so we will use it.

5.3.2.2: The Dlib Facial landmarks Detector

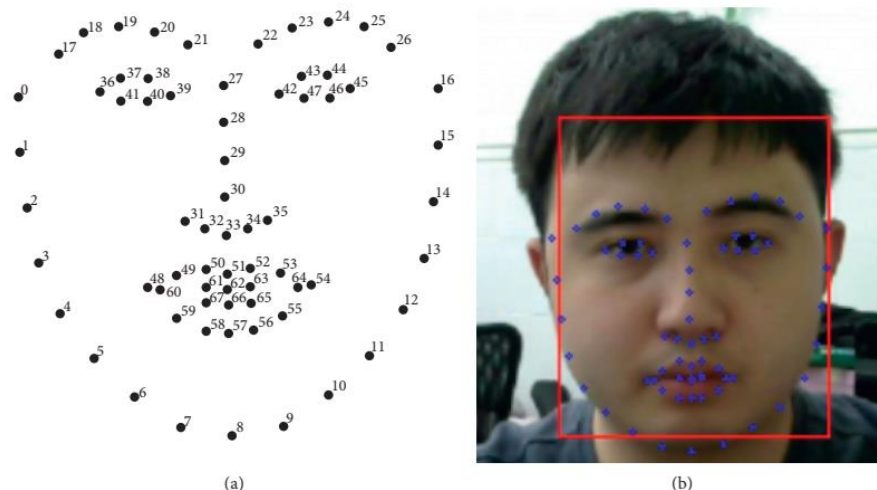
Dlib is principally a C++ library. However, you can use a number of its tools from python applications. It extracts the fine-grained features of a student's face (as is shown in **Figure 5.3.2.2(a)**).

Dlib library contains 68 face key points; testing principle is applying cascading shape regression to check all the key points of the face component. Face detection process is as follows:

Firstly, the feature of the input image is extracted, including the features of the face contour, eyebrows, eyes, nose, and mouth contours.

Secondly, the extracted features are mapped to the face feature points through a trained detector; at this point, an initial shape of the key points of the human face component is generated from its original image.

Thirdly, gradient boosting is used to iteratively adjust the initial shape until it matches with the real shape.



***Figure 5.3.2.2: Student's face feature point acquisition based on Dlib.
(a) Dlib face feature point positioning. (b) Face feature point positioning effect.***

- We Use `dlib.get_frontal_face_detector()` to return the default face detector.
- We use `dlib.shape_predictor` class ; the object from class is a tool that takes in an image region containing some object and outputs a set of point locations that define the pose of the object. The classic example of this is human face pose prediction, where you take an image of a human face as input and are expected to identify the locations of important facial landmarks such as the corners of the mouth and eyes, tip of the nose, and so forth.
- In our model we use `dlib.get_frontal_face_detector()` to detect the face then create an object from `dlib.shape_predictor` class and send the eyes coordinates to segment the eyes.
- After the segmentation of the two eyes and making a red rectangle around them, the model generates two images for both left and right eyes but the two generated images are not with the same size so the next step is resizing the images then concatenate them to generate a one image for the two eyes.

5.4: Building the model

- **Tensor Flow definition:** Tensor Flow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Steps:

- 1) Append all focused images to `Focused_Data` array and all unfocused images to `UnFocused_Data` array.
- 2) Label each image with 1 to focused and the same thing for unfocused with append 0 in `Labels_Data` array.
- 3) Collect all focused and non-focused images in one array `Training_Data` and list it with `Labels_Data` array.
- 4) Train the model but the results were not good. It generates an accuracy with **52%** for learning so we using another learning model.

- **Keras definition:** Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

Steps:

- 1) Read all images from files using filename.split(' ')
- 2) Append it's category for 1 to focused and 0 for unfocused in categories array based on file name.
- 3) Making the layers for the model
- 4) Data partition for training, testing and validation sets and starting training the model, the accuracy of this model is very good it generates accuracy with **93%**.

Method	Accuracy (%)
Development of system for human Emotional analysis using facial recognition [3]	88
Human Behavior Understanding in Big Multimedia Data Using CNN based Facial Recognition Expression [6]	89.85
Fatigue Driving Detection Algorithm Based on Facial Motion Information [1]	94.32
Estimation of students' attention in the classroom from Kinect facial and body features [13]	75.3
Tensor Flow Model	52
Keras Model	93

Figure 5.4.1: Comparison between the accuracy of each method

5.5: Focusing Classification

This is the final step in the system architecture is Focusing Classification and in this step the model determines if the student is focused or not according to its eyes position whether he looks directly at the monitor in front of him or looks far away from the monitor and not focused in the lecture.

5.6: Summary

In this chapter we discussed the architecture of the proposed system, Frame Preprocessing, Building the Model and Focusing Classification.

- System Architecture
- Frame Preprocessing
 - Frame Preprocessing Definition
 - Detecting the frame facial landmarks
 - Different facial landmarks detectors
 - The Dlib Facial landmarks Detector
- Building the model
- Focusing Classification

CHAPTER 6: System Implementation and Testing

6.1: Introduction

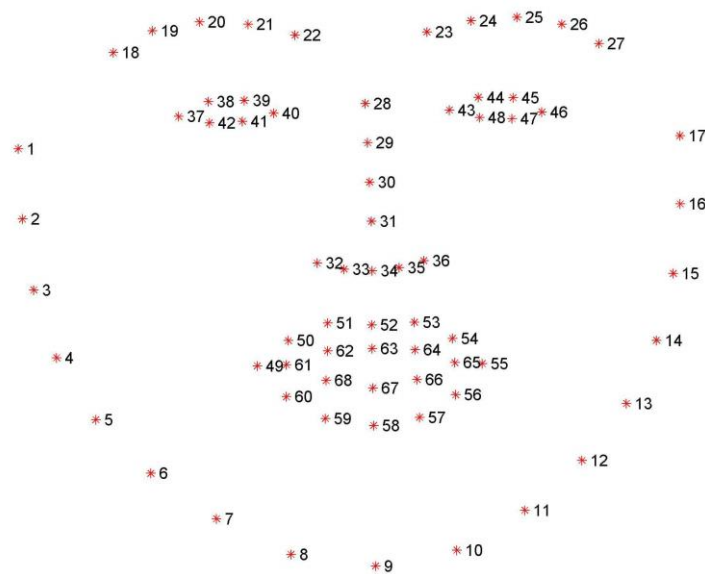
This chapter reviews the System Implementation Phases concerning the User Manual, Backend implementation and System Testing and Validation.

6.2 Backend Specification

The Backend of our system is constructed in Python Programming Language in Two Mainly Phase:

1- Face detection and eyes extraction model:

- Using Dlib library to extract facial land marks then extract eyes using points from “face_landmarks.dat”.



```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
In [9]: Fpath = "/content/drive/MyDrive/Colab_Notebooks/Dataset/LastV/Focused/"
UnFpath = "/content/drive/MyDrive/Colab_Notebooks/Dataset/LastV/UnFocused/"
for i in os.listdir(Fpath):
    if c == 1500:
        break
    image = cv2.imread(Fpath + i)
    frame = image
    img = frame.copy()
    if (get_landmarks(img).size == 2):
        print("continue Focused")
        continue
    eyes = GetEyes(img)
    reye = cv2.resize(eyes.crop_right_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    leye = cv2.resize(eyes.crop_left_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    new_array = cv2.hconcat([reye, leye])
    Focused_Data.append(new_array)
    Labels_Data.append(1)
c = 0
for i in os.listdir(UnFpath):
    if c == 1500:
        break
    image = cv2.imread(UnFpath + i)
    frame = image
    img = frame.copy()
    if (get_landmarks(img).size == 2):
        print("continue UnFocused")
        continue
    eyes = GetEyes(img)
    reye = cv2.resize(eyes.crop_right_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    leye = cv2.resize(eyes.crop_left_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    new_array = cv2.hconcat([reye, leye])
    UnFocused_Data.append(new_array)
    Labels_Data.append(0)
c += 1

continue UnFocused

Writing images to training folder

In [42]: j=1
for i in Focused_Data:
    status = cv2.imwrite('/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/TrainingSet/Focused {}.jpg'.format(j), i)
    j += 1

j=1
for i in UnFocused_Data:
    status = cv2.imwrite('/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/TrainingSet/UnFocused {}.jpg'.format(j), i)
    j += 1

```

```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
In [1]: import numpy as np
import cv2
import matplotlib.pyplot as plt
from IPython import display
import sys
import os
import dlib
import glob
from skimage import io
from skimage.draw import circle
from skimage.io import imread_collection
from random import shuffle
from tqdm import tqdm
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
import matplotlib.image as mpimg

In [2]: from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

In [ ]: Focused_dir = '/content/drive/MyDrive/Colab_Notebooks/Dataset/LastV/Focused/*.jpg'

UnFocused_dir = '/content/drive/MyDrive/Colab_Notebooks/Dataset/LastV/UnFocused/*.jpg'

Focused_imgs = imread_collection(Focused_dir)
UnFocused_imgs = imread_collection(UnFocused_dir)
# print(type(img_col))

In [3]: Training_Data = []
Focused_Data = []
UnFocused_Data = []
Labels_Data = []
reye = []
leye = []
Focused = '/content/drive/MyDrive/Colab_Notebooks/Dataset/Focused'
UnFocused = '/content/drive/MyDrive/Colab_Notebooks/Dataset/UnFocused'

In [36]: c = 0
c2 = 0

Taking 1500 Focused and 1500 UnFocused images from data set

```

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

Taking 1500 Focused and 1500 UnFocused images from data set to training set and 500 Focused and 500 UnFocused to testing set

```

In [37]: testF_array = []
testUF_array = []
Fpath = "/content/drive/MyDrive/Colab_Notebooks/Dataset/LastV/Focused/"
UnFpath = "/content/drive/MyDrive/Colab_Notebooks/Dataset/LastV/UnFocused/"
for i in os.listdir(Fpath):
    print(c2, c)
    c+=1
    c2+=1
    if(c2 == 2000):
        break
    image = cv2.imread(Fpath + i)
    frame = image
    img = frame.copy()
    if(get_landmarks(img).size == 2):
        print("continue Focused")
        continue
    eyes = GetEyes(img)
    reye = cv2.resize(eyes.crop_right_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    leye = cv2.resize(eyes.crop_left_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    new_array = cv2.hconcat([reye, leye])
    if c > 1500:
        testF_array.append(new_array)
        continue
    Focused_Data.append(new_array)
    Labels_Data.append(1)

c = 0
c2 = 0
for i in os.listdir(UnFpath):
    c+=1
    c2+=1
    print(c2, c)
    if(c2 == 2000):
        break
    image = cv2.imread(UnFpath + i)
    frame = image
    img = frame.copy()
    if(get_landmarks(img).size == 2):
        print("continue UnFocused")
        continue
    eyes = GetEyes(img)
    reye = cv2.resize(eyes.crop_right_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    leye = cv2.resize(eyes.crop_left_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
    new_array = cv2.hconcat([reye, leye])
    if c > 1500:
        testUF_array.append(new_array)
        continue
    UnFocused_Data.append(new_array)

```

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

Writing test images to test folder

```

In [43]: j=1
for i in testF_array:
    status = cv2.imwrite('/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/Test_imgs/Focused.{}.jpg'.format(j),i)
    j+=1

j=1
for i in testUF_array:
    status = cv2.imwrite('/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/Test_imgs/UnFocused.{}.jpg'.format(j),i)
    j+=1

```

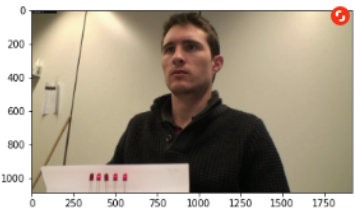
sample of Images

```

In [ ]: frame = im_col[0]
plt.imshow(frame)

```

Out[37]: <matplotlib.image.AxesImage at 0x7f802dd36690>



Model to detect face and class to extract eyes

```

In [7]: # dlib face Landmarks detection
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('/content/drive/MyDrive/Colab_Notebooks/shape_predictor_68_face_landmarks.dat')

class TooManyFaces(Exception):
    pass

class NoFaces(Exception):
    pass

# extract Land marks

```

```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
Model to detect face and class to extract eyes

In [7]: # dlib face Landmarks detection
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('/content/drive/MyDrive/Colab_Notebooks/shape_predictor_68_face_landmarks.dat')

class TooManyFaces(Exception):
    pass

class NoFaces(Exception):
    pass

# extract Land marks
def get_landmarks(im):
    rects = detector(im, 1)

    if len(rects) > 1:
        #raise TooManyFaces
        return np.matrix([0,0])
    if len(rects) == 0:
        #raise NoFaces
        return np.matrix([0,0])

    return np.matrix([[p.x, p.y] for p in predictor(im, rects[0]).parts()])

class GetEyes():
    def __init__(self, img):
        self.img = img.copy()
        landmarks = self.get_landmarks(frame)
        self.left_eye_lms = landmarks[42:47]
        self.right_eye_lms = landmarks[36:41]
        self.left_eye_rect = self.get_rect(self.left_eye_lms)
        self.right_eye_rect = self.get_rect(self.right_eye_lms)

    @staticmethod
    def get_landmarks(im):
        # extract Land marks representing the eyes
        rects = detector(im, 1)

        if len(rects) > 1:
            raise TooManyFaces
        if len(rects) == 0:
            raise NoFaces

        return np.matrix([[p.x, p.y] for p in predictor(im, rects[0]).parts()])

    @classmethod
    def get_rect(self, landmarks):
        # get rect from eye Landmarks

```

```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
# get rect from eye Landmarks
border = 5
l_ul_x = min(landmarks[:,0])
l_ul_y = min(landmarks[:,1])
l_lr_x = max(landmarks[:,0])
l_lr_y = max(landmarks[:,1])
pt1 = (np.sum(l_ul_x)-border, np.sum(l_ul_y)-border)
pt2 = (np.sum(l_lr_x)+border, np.sum(l_lr_y)+border)
return [pt1, pt2]

def draw(self):
    # draw rect
    eimg = self.img
    eimg = cv2.rectangle(img, self.right_eye_rect[0], self.right_eye_rect[1], (255, 0, 0), thickness=3)
    eimg = cv2.rectangle(img, self.left_eye_rect[0], self.left_eye_rect[1], (255, 0, 0), thickness=3)
    plt.imshow(eimg)
    return eimg

def crop_right_eye(self):
    # crop out right eye
    return self.img[self.right_eye_rect[0][1]:self.right_eye_rect[1][1], self.right_eye_rect[0][0]:self.right_eye_rect[1][0]]

def crop_left_eye(self):
    # crop out left eye
    return self.img[self.left_eye_rect[0][1]:self.left_eye_rect[1][1], self.left_eye_rect[0][0]:self.left_eye_rect[1][0]]

```

The sample after extracting eyes

```

In [ ]: img = frame.copy()
eyes = GetEyes(img)
plt.imshow(eyes.draw())

Out[8]: <matplotlib.image.AxesImage at 0x7f7f29b56ad0>

```



```

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
In [ ]: # frame= cv2.imread(os.path.join(Focused, item), cv2.IMREAD_GRAYSCALE)
# img = frame.copy()
# eyes = GetEyes(img)

# a = eyes.crop_left_eye()
# b= eyes.crop_right_eye()

# fig = plt.figure()

# ax1 = fig.add_subplot(2,2,1)
# ax1.imshow(a)
# ax2 = fig.add_subplot(2,2,2)
# ax2.imshow(b)

# na = cv2.resize(a, dsize=(70, 30), interpolation=cv2.INTER_CUBIC)
# nb = cv2.resize(b, dsize=(70, 30), interpolation=cv2.INTER_CUBIC)

# ax1 = fig.add_subplot(2,2,3)
# ax1.imshow(na)

# bb = (na + nb)

# ax2 = fig.add_subplot(2,2,4)
# print('aaaaaaaa')
# ax2.imshow(bb)

frame= im_col[17]
plt.imshow(frame)

img = frame.copy()
eyes = GetEyes(img)

reye = (cv2.resize(eyes.crop_right_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC))
leye = (cv2.resize(eyes.crop_left_eye(), dsize=(70, 30), interpolation=cv2.INTER_CUBIC))

plt.imshow(newImg)

Out[28]: <matplotlib.image.AxesImage at 0x7f802e824a90>
0
10
20
0 20 40 60 80 100 120

```

```

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
In [8]: def preprocess(img):
# preprocess eye images
if len(img.shape)==3:
img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
#img = cv2.GaussianBlur(img,(3, 3),0);
img = cv2.medianBlur(img, 5)
windowClose = np.ones((5,5),np.uint8)
windowOpen = np.ones((3,3),np.uint8)
windowErode = np.ones((3,3),np.uint8)
img = cv2.morphologyEx(img, cv2.MORPH_CLOSE, windowClose)
#img = cv2.morphologyEx(img, cv2.MORPH_ERODE, windowErode)
img = cv2.morphologyEx(img, cv2.MORPH_OPEN, windowOpen)
plt.imshow(img)
return img

def detect_pupil(img):
output = img.copy()
if len(img.shape)==3:
# use red channel because of human complexion
img = preprocess(img[:, :,0])
# detect circles in the image
circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 1, int(img.shape[1]/3),200,100,8,8)

# ensure at least some circles were found
if circles is not None:
# convert the (x, y) coordinates and radius of the circles to integers
circles = np.round(circles[0, :]).astype("int")
# Loop over the (x, y) coordinates and radius of the circles
for (x, y, r) in circles:
# draw the circle in the output image, then draw a rectangle
# corresponding to the center of the circle
cv2.circle(output, (x, y), r, (255, 255, 255), 1)
cv2.rectangle(output, (x - 2, y - 2), (x + 2, y + 2), (255, 0, 0), 1)
return circles, output

In [ ]: eye_img_r = eyes.crop_right_eye()
eye_img_l = eyes.crop_left_eye()
proc_r = preprocess(eye_img_r)
proc_l = preprocess(eye_img_l)

# display right and left eye image
fig = plt.figure()
ax1 = fig.add_subplot(2,2,1)
ax1.imshow(eye_img_r)
ax2 = fig.add_subplot(2,2,2)
ax2.imshow(eye_img_l)
ax1 = fig.add_subplot(2,2,3)
ax1.imshow(proc_r)
ax2 = fig.add_subplot(2,2,4)
ax2.imshow(proc_l)

```

```
In [ ]: eye_img_r = eyes.crop_right_eye()
eye_img_l = eyes.crop_left_eye()
proc_r = preprocess(eye_img_r)
proc_l = preprocess(eye_img_l)

# display right and left eye image
fig = plt.figure()
ax1 = fig.add_subplot(2,2,1)
ax1.imshow(eye_img_r)
ax2 = fig.add_subplot(2,2,2)
ax2.imshow(eye_img_l)
ax1 = fig.add_subplot(2,2,3)
ax1.imshow(proc_r)
ax2 = fig.add_subplot(2,2,4)
ax2.imshow(proc_l)
```

Out[10]: <matplotlib.image.AxesImage at 0x7f727e0c3d0>



2- Using Keras model to

```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
```

```
In [1]: import numpy as np
import cv2
import matplotlib.pyplot as plt
from IPython import display
import sys
import os
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalization
import keras.utils
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.model_selection import train_test_split
import random
import dlib
import glob
from skimage import io
from skimage.draw import circle
from skimage.io import imread_collection
from tqdm import tqdm
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
import matplotlib.image as mpimg

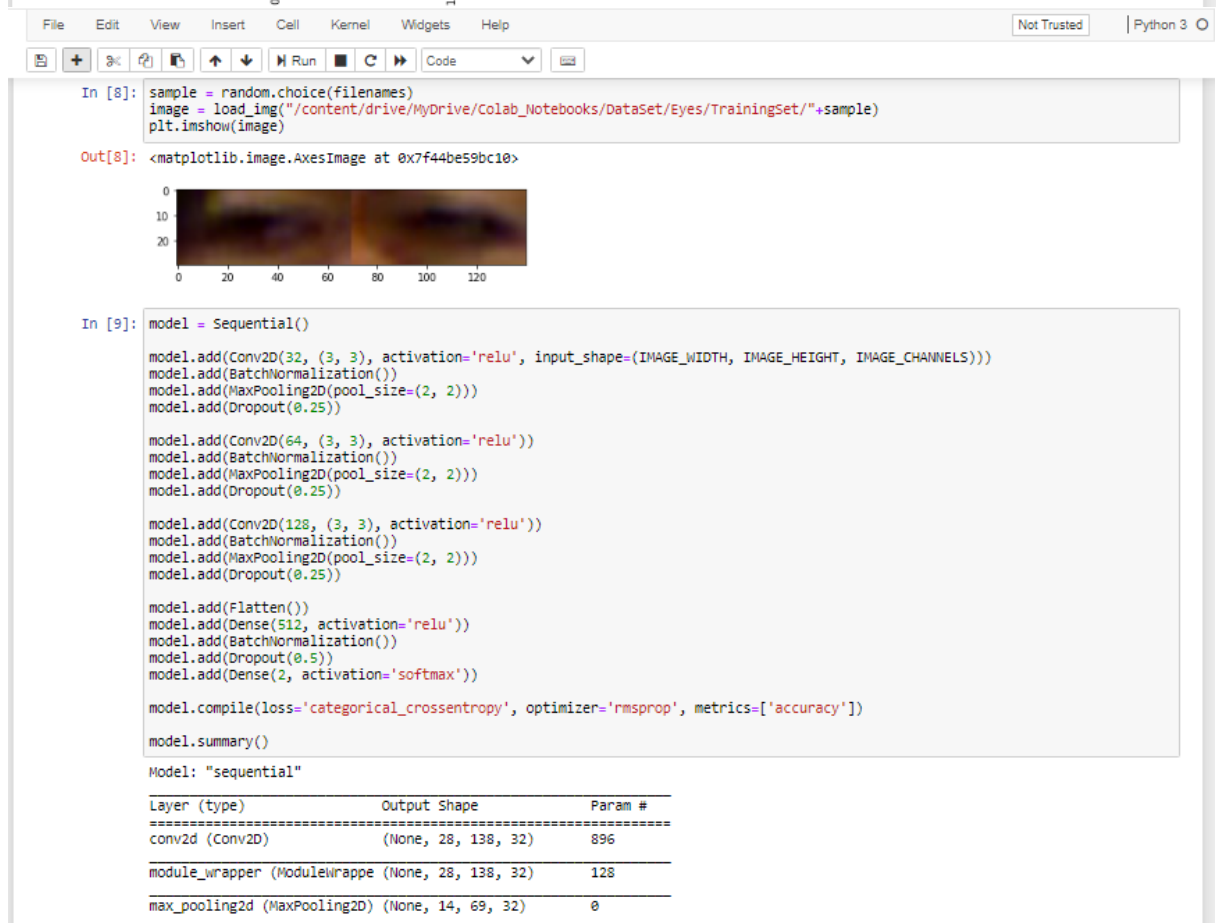
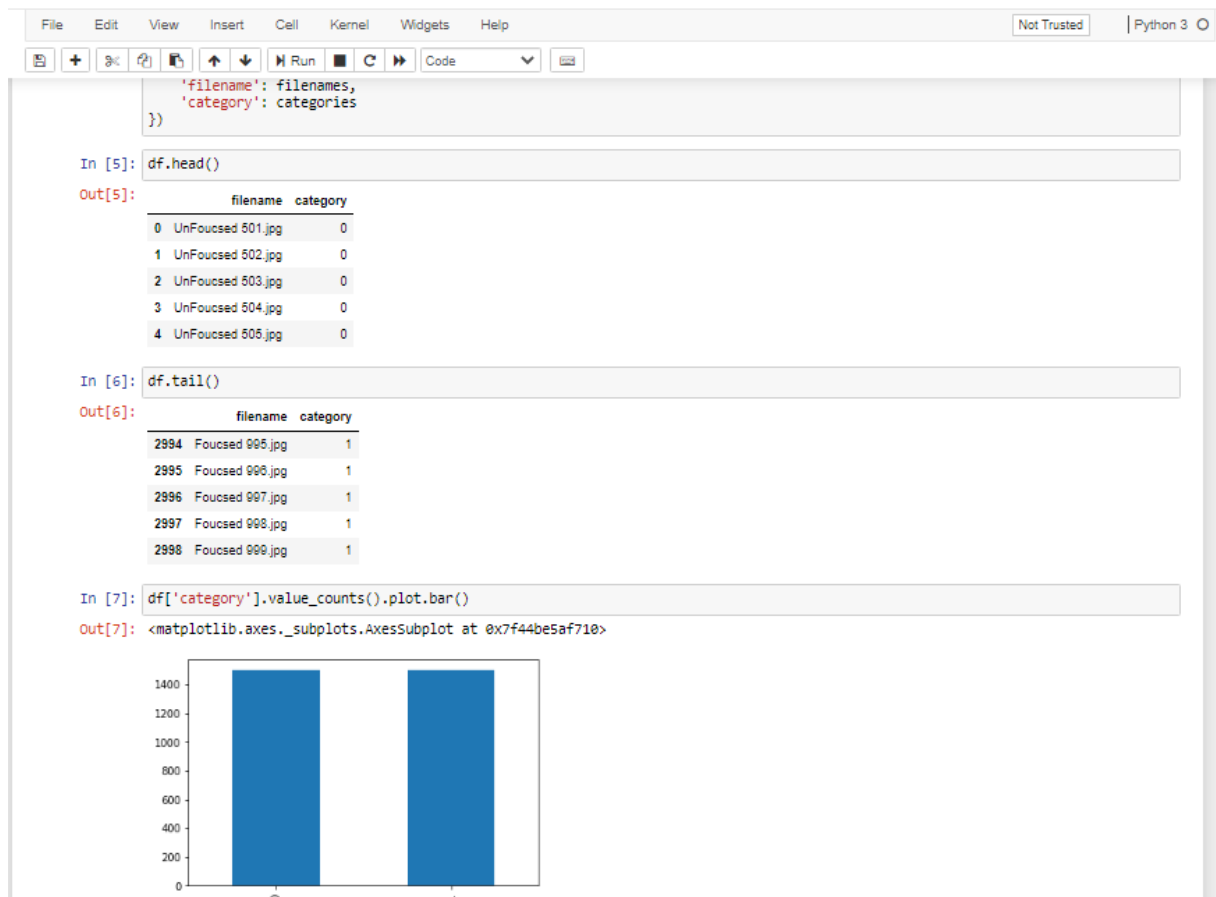
In [2]: from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive

In [3]: FAST_RUN = False
IMAGE_WIDTH=30
IMAGE_HEIGHT=40
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3

In [4]: filenames = os.listdir("/content/drive/MyDrive/Colab_Notebooks/DataSet/Eyes/TrainingSet")
categories = []
for filename in filenames:
    category = filename.split(' ')[0]
    if category == 'Focused':
        categories.append(1)
    else:
        categories.append(0)

df = pd.DataFrame({
    'filename': filenames,
    'category': categories
})
```

classifying images to 0 for unfocused and 1 for focused



File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

+
-
↶
↷
↺
↻
⏮
⏪
⏩
⏭
Run
Code

conv2d_1 (Conv2D)	(None, 12, 67, 64)	18496
module_wrapper_1 (ModuleWrap	(None, 12, 67, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 6, 33, 64)	0
dropout_1 (Dropout)	(None, 6, 33, 64)	0
conv2d_2 (Conv2D)	(None, 4, 31, 128)	73856
module_wrapper_2 (ModuleWrap	(None, 4, 31, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 2, 15, 128)	0
dropout_2 (Dropout)	(None, 2, 15, 128)	0
flatten (Flatten)	(None, 3840)	0
dense (Dense)	(None, 512)	1966592
module_wrapper_3 (ModuleWrap	(None, 512)	2048
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026

=====
Total params: 2,063,810
Trainable params: 2,062,338
Non-trainable params: 1,472
=====

In [10]: `earlystop = EarlyStopping(patience=10)`

In [11]: `learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
patience=2,
verbose=1,
factor=0.5,
min_lr=0.00001)`

In [12]: `callbacks = [earlystop, learning_rate_reduction]`

In [13]: `df["category"] = df["category"].replace({0: 'UnFocused', 1: 'Focused'})`

In [14]: `train_df, validate_df = train_test_split(df, test_size=0.20, random_state=42)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)`

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

+
-
↶
↷
↺
↻
⏮
⏪
⏩
⏭
Run
Code

In [12]: `callbacks = [earlystop, learning_rate_reduction]`

In [13]: `df["category"] = df["category"].replace({0: 'UnFocused', 1: 'Focused'})`

In [14]: `train_df, validate_df = train_test_split(df, test_size=0.20, random_state=42)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)`

In [15]: `train_df['category'].value_counts().plot.bar()`

Out[15]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f44bdeea790>`

category	count
Focused	1150
UnFocused	1100

In [16]: `validate_df['category'].value_counts().plot.bar()`

Out[16]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f44ba566210>`

category	count
Focused	320
UnFocused	280

57 | Page

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

In [17]:

```
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
batch_size=15
```

In [18]:

```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    "/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/TrainingSet/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

Found 2399 validated image filenames belonging to 2 classes.

In [19]:

```
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    "/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/TrainingSet/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

Found 600 validated image filenames belonging to 2 classes.

In [20]:

```
example_df = train_df.sample(n=1).reset_index(drop=True)
example_generator = train_datagen.flow_from_dataframe(
    example_df,
    "/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/TrainingSet/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
)
```

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

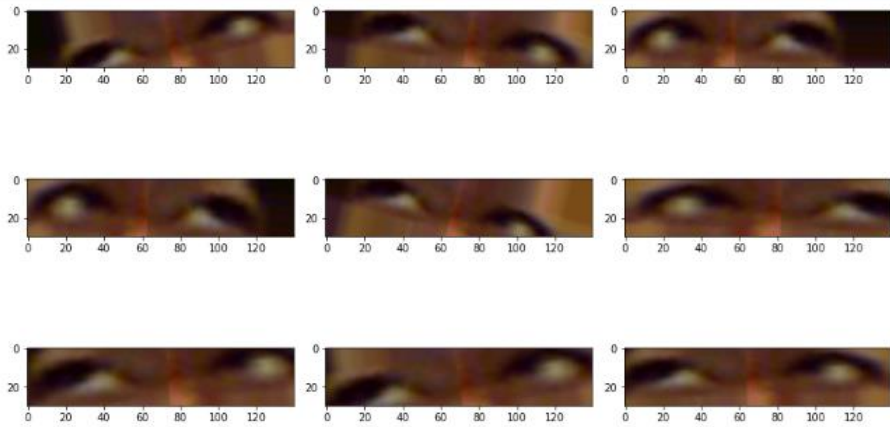
In [20]:

```
example_df = train_df.sample(n=1).reset_index(drop=True)
example_generator = train_datagen.flow_from_dataframe(
    example_df,
    "/content/drive/MyDrive/Colab_Notebooks/Dataset/Eyes/TrainingSet/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical'
)
```

Found 1 validated image filenames belonging to 1 classes.

In [21]:

```
plt.figure(figsize=(12, 12))
for i in range(0, 15):
    plt.subplot(5, 3, i+1)
    for X_batch, Y_batch in example_generator:
        image = X_batch[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()
```



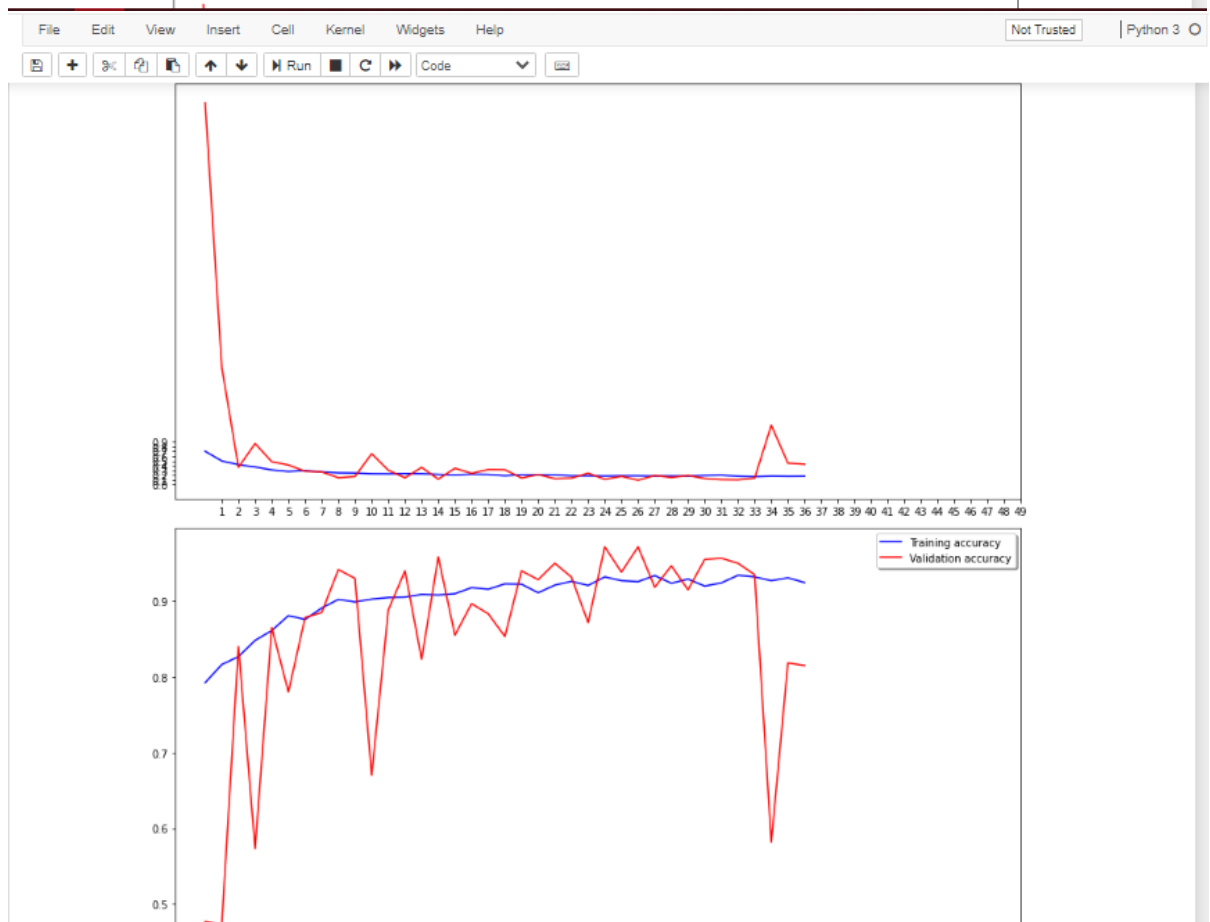
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
In [22]: epochs=3 if FAST_RUN else 50
history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size,
    callbacks=callbacks
)
loss, accuracy, val_loss, val_accuracy, lr
Epoch 34/50
159/159 [=====] - 30s 187ms/step - loss: 0.1665 - accuracy: 0.9320 - val_loss: 0.1359 - val_accuracy: 0.9350
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss, accuracy, val_loss, val_accuracy, lr
Epoch 35/50
159/159 [=====] - 30s 186ms/step - loss: 0.1784 - accuracy: 0.9270 - val_loss: 1.2562 - val_accuracy: 0.5817
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss, accuracy, val_loss, val_accuracy, lr
Epoch 36/50
159/159 [=====] - 30s 186ms/step - loss: 0.1739 - accuracy: 0.9308 - val_loss: 0.4521 - val_accuracy: 0.8183
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss, accuracy, val_loss, val_accuracy, lr
Epoch 37/50
159/159 [=====] - 30s 180ms/step - loss: 0.1759 - accuracy: 0.9245 - val_loss: 0.4292 - val_accuracy: 0.8150
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss, accuracy, val_loss, val_accuracy, lr

In [23]: model.save_weights("model.h5")

In [24]: fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12))
ax1.plot(history.history['loss'], color='b', label="Training loss")
ax1.plot(history.history['val_loss'], color='r', label="validation loss")
ax1.set_xticks(np.arange(1, epochs, 1))
ax1.set_yticks(np.arange(0, 1, 0.1))
ax2.plot(history.history['accuracy'], color='b', label="Training accuracy")
ax2.plot(history.history['val_accuracy'], color='r', label="Validation accuracy")
ax2.set_xticks(np.arange(1, epochs, 1))
legend = plt.legend(loc='best', shadow=True)
plt.tight_layout()
plt.show()

```



```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
In [25]: test_filenames = os.listdir("/content/drive/MyDrive/Colab_Notebooks/DataSet/Eyes/Test_imgs")
test_df = pd.DataFrame({
    'filename': test_filenames
})
nb_samples = test_df.shape[0]

In [26]: test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(
    test_df,
    "/content/drive/MyDrive/Colab_Notebooks/DataSet/Eyes/Test_imgs",
    x_col='filename',
    y_col=None,
    class_mode=None,
    target_size=IMAGE_SIZE,
    batch_size=batch_size,
    shuffle=False
)

Found 998 validated image filenames.

In [27]: predict = model.predict_generator(test_generator, steps=np.ceil(nb_samples/batch_size))

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:2001: UserWarning: `Model.predict_generator`
is deprecated and will be removed in a future version. Please use `Model.predict`, which supports generators.
warnings.warn("`Model.predict_generator` is deprecated and ")

In [28]: test_df['category'] = np.argmax(predict, axis=-1)

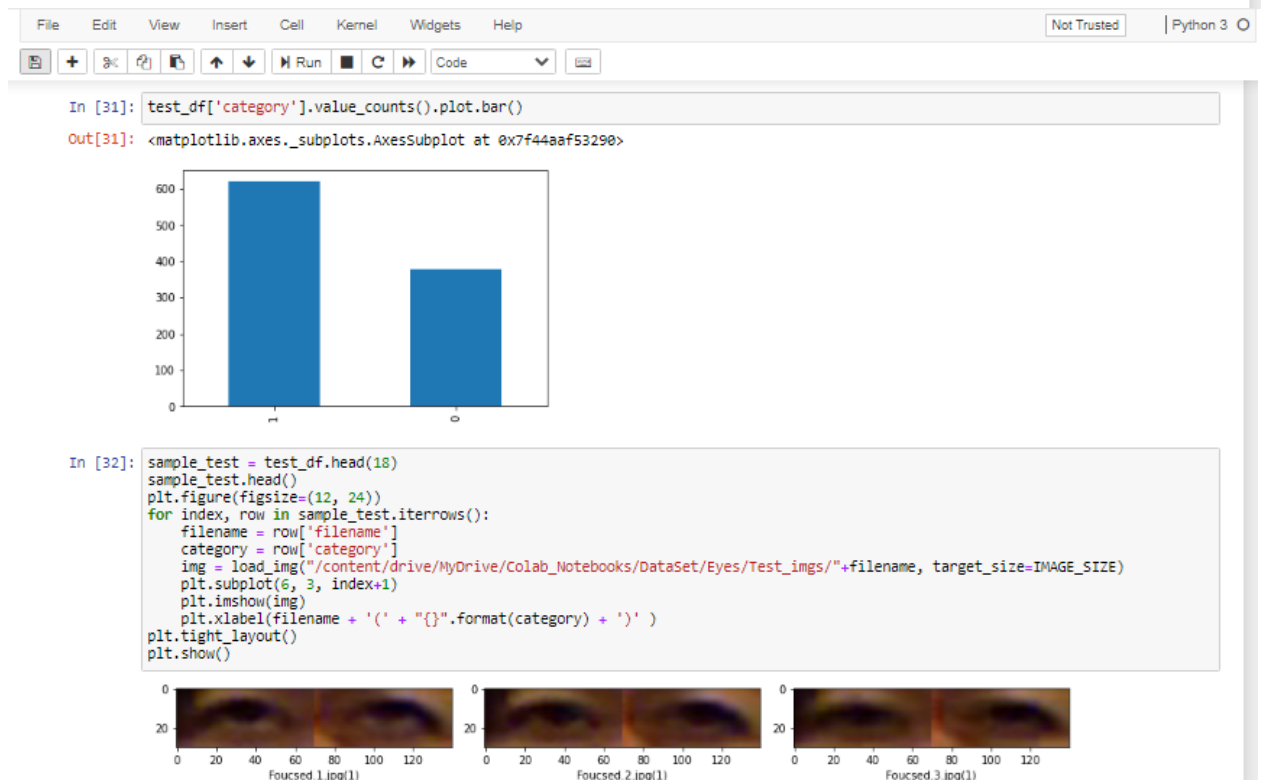
In [29]: label_map = dict((v,k) for k,v in train_generator.class_indices.items())
test_df['category'] = test_df['category'].replace(label_map)

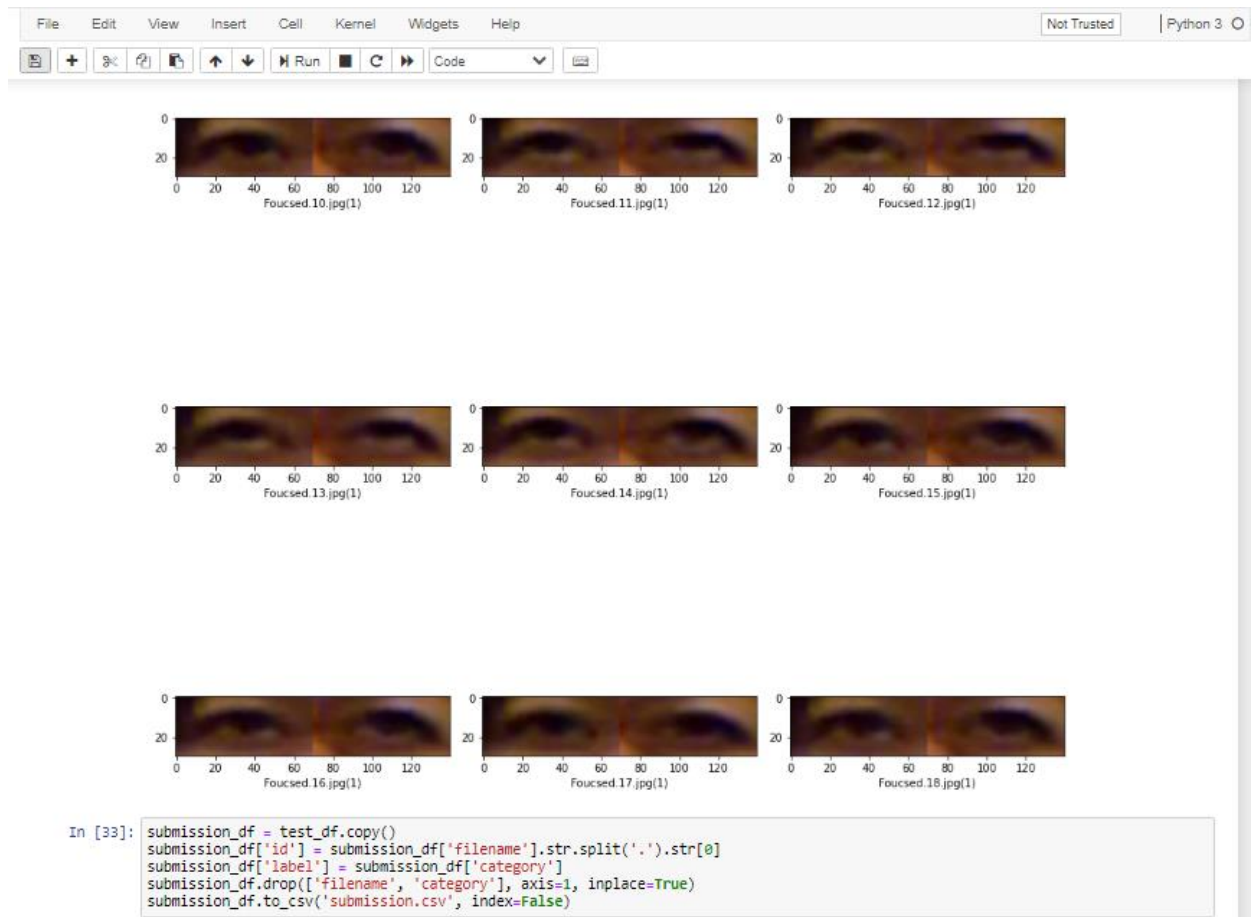
In [30]: test_df['category'] = test_df['category'].replace({'Foucsed': 1, 'UnFoucsed': 0})

In [31]: test_df['category'].value_counts().plot.bar()

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f44aaf53290>

```





6.3 System Testing and Validation

- **Visibility of System status**
The system should always give user information about status of tracked person, through give them a feedback if this person was focused or not during the online meeting.
- **Match between system and the real world**
The System is making a real-time in order to alert the user, manager or professor that some student or employee is not focused in particular time, that make manger or professor that can take action in this time.
- **User Control and Freedom**
The model can be an extension for any online meeting application with suitable GUI which user can deal with it easy.
- **Flexibility and Efficiency of use**
The System is considered to be simple for unexperienced users' experience.
- **Consistency and Standards**

Different words and actions are taken into consideration for facilitating the model experience.

- **Error Prevention**
Model content error prevention in case of model can't detect the face it gives an exception that he doesn't find a face or if detecting more than one face it gives an exception that he found more than one face.
- **Help users recognize, diagnose, and recover from errors**
Error messages are expressed in plain language and it suggests a solution.

6.4 The Requirements Validation and Completeness

Hardware Requirements

In order to have a real-time processing a GPU has to be provided, the face detection and eyes extraction model are trained and tested using google colab which is a drive platform which provide a high-performance hardware using to training and testing machine learning models also camera has to be provided to capture images to send it to classification model.

Software Requirements

Make sure you have a good connection to internet if you will work in cloud platform and it's provided a whole python library that you need.

6.5 Summary

In this chapter we have discussed the System implementation components which is Backend with Python programming language and System Testing and Validation.

- 6.1: Introduction
- 6.2: Backend Specification
- 6.3: System Testing & Validation
- 6.4 The Requirements Validation and Completeness

CHAPTER 7: PROJECT CONCLUSION AND FUTURE WORK

7.1 Conclusion:

Since the last few years, and especially during a period when the entire world switched to work and study online due to COVID-19, the increasing usage of the internet and developments in the field of information technology has gradually boosted the use of online meetings. Using web conferencing services such as ezTalks Cloud Meeting, Skype, Zoom, and Google meet has made communication easier, more effective, and efficient. Online meetings have offered a various benefits, particularly to the business world and education, since they now allow employees to speak with colleagues and associates in faraway regions without having to leave their office to discuss corporate strategy and challenges. However, while online meetings have some benefits, they also have some limitations. Can be difficult for the teacher or manager (host) to tell whether or not the meeting participants are focusing. Many papers had discussed this option and provided some ideas, such as using facial expression, mobile logs, and eye expression with Gaze Tracking, which we thought was the best choice, so we solved this problem using Gaze Tracking and the EYEDIAP dataset, which was designed to train and test assess techniques for gaze estimation from RGB and RGB-D data, In the end, we were able to reach a 93 percent accuracy rate.

7.2 Future Work:

We aim to:

- Measure Focusing Percentage; classify it into (High – Medium - Low).
- Measure Focusing according to head pose in addition to eye tracking.
- Measure Focusing using Mobile Log to detect if the student was Distracted.
- Improve the model so that if a student's eyes are focused in one place for a long time, it is classified as not focus

References

- [1] You F, Gong Y, Tu H, Liang J, Wang H. A fatigue driving detection algorithm based on facial motion information entropy. *Journal of advanced transportation*. 2020 Jun 15;2020.
- [2] Pranav KB, Manikandan J. Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks. *Procedia Computer Science*. 2020 Jan 1;171:1651-9.
- [3] Raj C. Development of System for Human Emotional Analysis Using Facial Recognition Technique. *Development*. 2020;29(10S):7782-90.
- [4] Khan MQ, Lee S. Gaze and eye tracking: techniques and applications in ADAS. *Sensors*. 2019 Jan;19(24):5540.
- [5] Zhang L, Morgan M, Bhattacharya I, Foley M, Braasch J, Riedl C, Foucault Welles B, Radke RJ. Improved visual focus of attention estimation and prosodic features for analyzing group interactions. In *2019 International Conference on Multimodal Interaction 2019 Oct 14* (pp. 385-394).
- [6] Sajjad M, Zahir S, Ullah A, Akhtar Z, Muhammad K. Human behavior understanding in big multimedia data using CNN based facial expression recognition. *Mobile networks and applications*. 2019 Sep 9:1-1.
- [7] Baltrusaitis T, Zadeh A, Lim YC, Morency LP. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018) 2018 May 15* (pp. 59-66). IEEE.
- [8] Sharma P, Esengönül M, Khanal SR, Khanal TT, Filipe V, Reis MJ. Student concentration evaluation index in an e-learning context using facial emotion analysis. In *International Conference on Technology and Innovation in Learning, Teaching and Education 2018 Jun 20* (pp. 529-538). Springer, Cham.
- [9] Palmero C, Selva J, Bagheri MA, Escalera S. Recurrent cnn for 3d gaze estimation using appearance and shape cues. *arXiv preprint arXiv:1805.03064*. 2018 May 8.
- [10] Johnston B, de Chazal P. A review of image-based automatic facial landmark identification techniques. *EURASIP Journal on Image and Video Processing*. 2018 Dec;2018(1):1-23.
- [11] Massé B, Ba S, Horaud R. Tracking gaze and visual focus of attention of people involved in social interaction. *IEEE transactions on pattern analysis and machine intelligence*. 2017 Dec 13;40(11):2711-24.
- [12] Zhang X, Sugano Y, Fritz M, Bulling A. It's written all over your face: Full-face appearance-based gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops 2017* (pp. 51-60).

- [13] Zaletelj J. Estimation of students' attention in the classroom from kinect features. InProceedings of the 10th International Symposium on Image and Signal Processing and Analysis 2017 Sep 18 (pp. 220-224). IEEE.
- [14] Krafska K, Khosla A, Kellnhofer P, Kannan H, Bhandarkar S, Matusik W, Torralba A. Eye tracking for everyone. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 2176-2184).
- [15] Zhang X, Sugano Y, Fritz M, Bulling A. Appearance-based gaze estimation in the wild. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 4511-4520).
- [16] Funes Mora KA, Monay F, Odobez JM. Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. InProceedings of the Symposium on Eye Tracking Research and Applications 2014 Mar 26 (pp. 255-258).