Accident Prevention

SQA Assignment - 14 November 2019

SRS and SDD Document

SQA Plan

White Box Testing

Black Box Testing

Submitted by

Mohd Shoaib Rayeen

Kachuipam Rungsung

Software Requirement Specification

Overview: A real time system is build to detect if A Person is yawning or sleeping and alert accordingly.

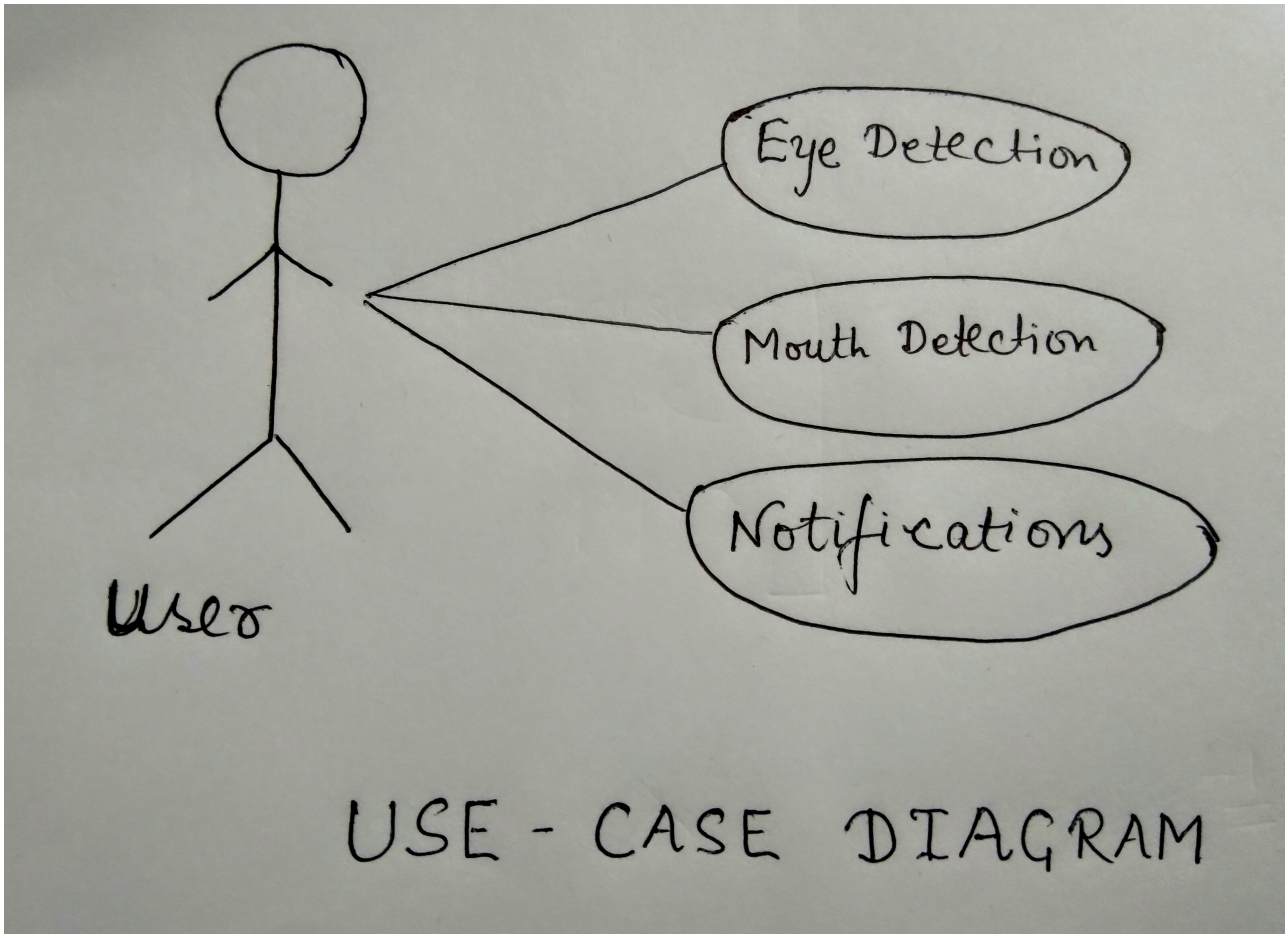
Scope: In India, there are a lot of accidents taking place every day due to mishaps. As the population grows the number of cars and accidents are directly proportional. This example program shows how to find frontal human faces in an image and estimate their pose. The pose takes the form of 68 landmarks. These are points on the face such as the corners of the mouth, along the eyebrows, on the eyes, and so forth. So we would detect If their eyes have been closed for a certain amount of time, we will assume that they are starting to doze off and play an alarm to wake them up and grab their attention and if the person yawns it would prompt another alert alarm for this to make him/her caution.

Approach: The system is developed using OpenCV and Shape Predictor Model. The image will be detected by the OpenCV interface and the eyes & the mouth would be detected by the Shape Predictor Model. After detection, the

model predicts that if the person is sleeping , yawning or both by comparing with a threshold value, then the model sends notification to the person according to the situation.

Intended Audience: The system will be useful for everyone who uses a vehicle.

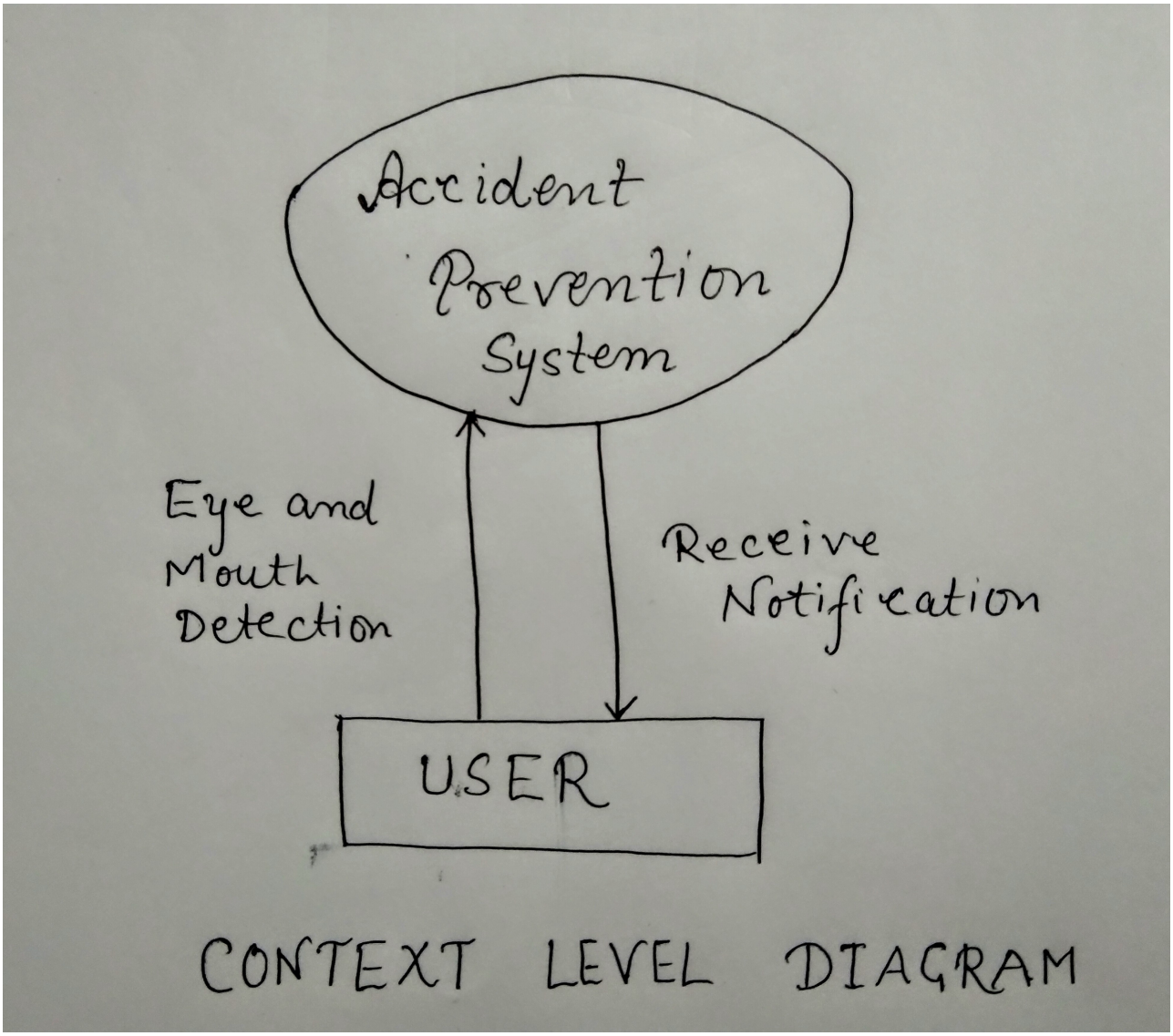
Use Case Overview: As mentioned, the system would be useful for everyone who uses a vehicle. Let's say that it's a user.



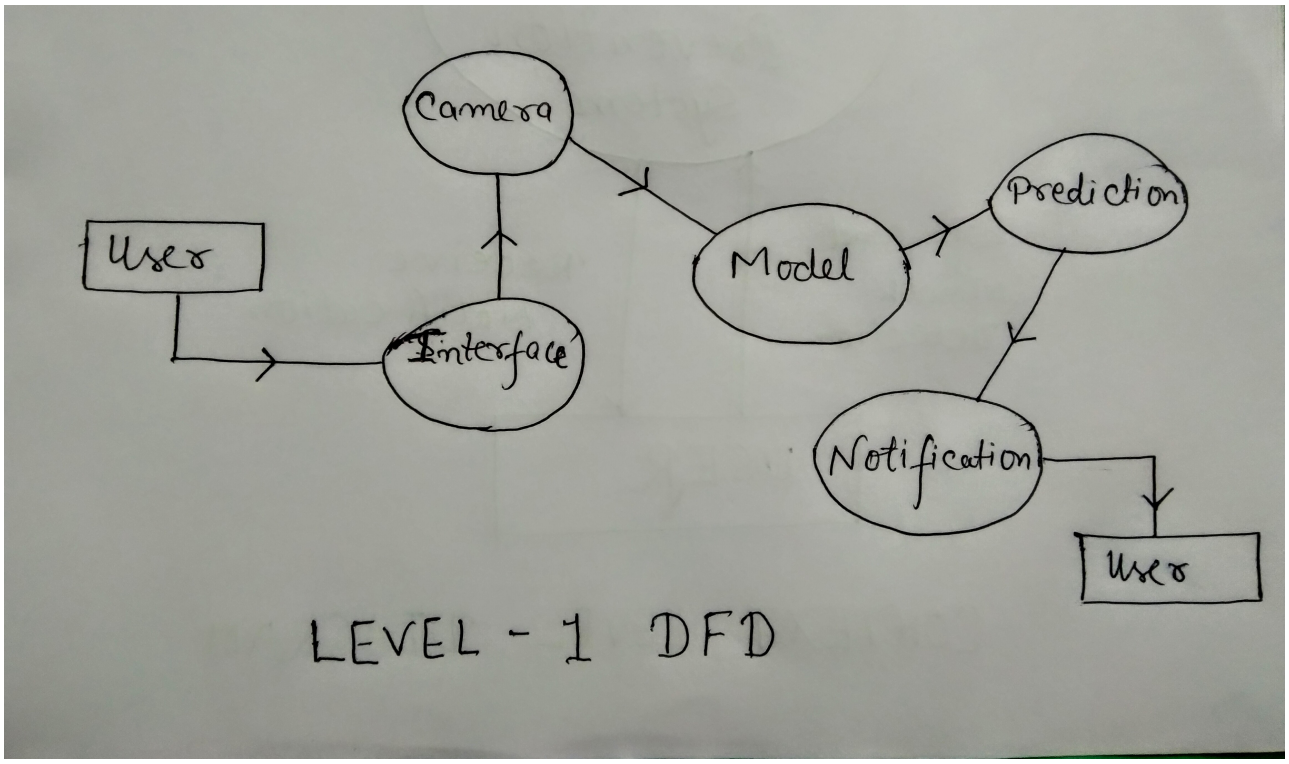
1. **Eye Detection:** The system would detect user's eyes.
2. **Mouth Detection:** The system would detect user's mouth.
3. **Notifications:** The system would sends the notification to the user if it predicts that the user is sleeping, yawning or doing both.

Software Design Descriptions

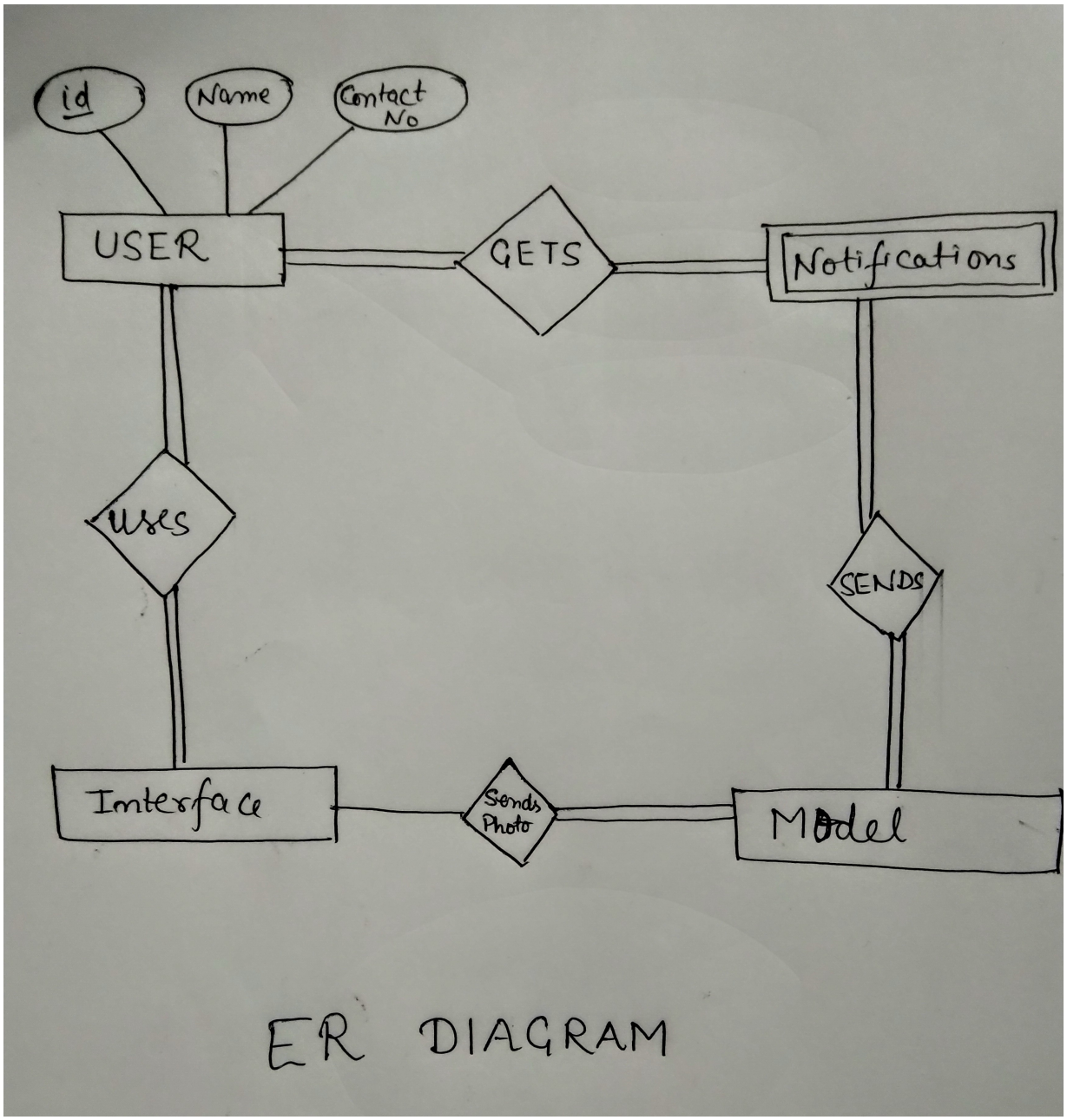
Context Level Diagram(Level 0 DFD): A context diagram is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and the user.



Level 1 DFD: A level 1 DFD notates each of the main sub-processes that together form the complete system.



Entity Relationship Diagram: An entity relationship diagram (ERD) is a graphical representation that depicts relationships among the user, interface, model and notification events of the system.



Software Quality Assurance Plan

Purpose: The purpose of this document is to build a real time system that detects if A Person is yawning or sleeping and notify accordingly.

Management: This part has 2 different sub-parts:

1. Organisation and Responsibilities: Accident Prevention System Project was managed by two members and they have organised and performed all tasks by themselves.

2. Tasks: There are multiple tasks that are needed to be performed:

(i) Preparation of An SRS
(ii) Review of the SRS
(iii) Preparation of An SDD
(iv) Review of the SDD
(v) Preparation of An SQA Plan
(vi) Preparation of Software Development Process
(vii) Preparation of Software Testing Plan
(viii) Production of the source code

Documentation: The documentation of the project contains following units:

(i) SRS: The detail description of SRS is mentioned in the first part of this document.

(ii) SDD: The detail description of SDD is mentioned in the second part of this document.

(iii) Review: The review of the project has been approved by the convenor of Department of Computer Science, IIT BHU.

Standards: IEEE Standards were followed during the design of SQA Plan.

Reviews and Audits: The review was done formally as well as informally.

(i) Informal Review: The members discussed with other teams about the idea, if it's feasible to design the idea, if the requirements fit with the real time environments, which programming languages to be used, if the dependancies won't affect much environment, etc

(ii) Formal Review: The product was reviewed by the AWS Solution Architects, Amazon on 22nd August '19.

They reviewed the required documents and got to see the working demo of the system.

Tools and Techniques: The project uses the following strategy for selection of the tool on the project:

(i) The requirements are specified based on real time environment and tools used in the industry.

(ii) The programming language is selected on based on less execution time and easy to understand criteria.

(iii) The model is selected on the basis of 68 landmarks of the face criteria so that we can use the required landmarks to calculate EAR and MAR.

(iv) The testing tools are selected on the basis of the programming language and the modules are tested in the real time environment.

Code Control: The required libraries are used to implement the code and are selected on the basic of execution time and the environment. The user won't be able to access the code.

Media Control: The system does not store the images in the database so no external storage is needed.

Records Collection and Maintenance: The record is maintained by the team and if there is bugs in the system, they'd fix it by theirselves.

Testing Strategy: Testing would be mainly focused on increasing the accuracy and detecting inconsistency between implementation and the requirements.

To achieve the goal, The four main testing levels are followed:

(i) Unit Testing: Each Module is tested separately and the module is working fine. As the system works in the real time environment, The face landmarks are passed to the modules and it works accordingly. For Example, In Eye Aspect Ratio Module, only eyes landmarks are passed as the argument so that it could compute the aspect ratio.

(ii) Integration Testing: In Integration Testing, we integrate multiple small modules together and test on that. Top Down Integration was performed to test the system. The integration testing was performed by the team and the demo of the system was working after integration and unit testing. For Example, all the aspect ratio modules were integrated to main module to

compute the aspect ratio of the real time image based on the facial landmarks.

(iii) Acceptance Testing: Acceptance Testing is used to evaluate the functionality of the system. It verifies that the system meets to its requirements. As the system was designed on the four major test cases and it's working on all of that so the system meets to its requirements. For Example, the system detects all the 4 test cases and notifies accordingly.

(iv) Regression Testing: This is used to catch new bugs into the code. As every time the code changes, the regression testing should be performed. As the whole system is working on each test case and there is no new bugs found during the testing, so the model does not find any issue.

Testing Level	No of Bugs(%)	Follow UP
Unit Testing	0.00	Each module is tested separately and they are working fine.
Integration Testing	0.00	Modules are integrated and no bug is detected.
Acceptance Testing	0.00	As per demo, It works with all the test cases. It meets to the requirements.
Regression Testing	0.00	The system was testing more than 10 times and it works.

Quality Assurance Process Measures: Measurement of the SQA process provides an evaluation criteria to increase the quality of the software.

Measurement	Goal
Defect Find Rate	There is no defect found during the testing of the system. If there'd be any defect in the future, it'd be very less.
Defect Fix Rate	The fixing rate would be very high as it works on the face landmarks predictor.
Environment Effect	As the system is highly dependant on the camera and it'd need light. If the system gets good image, the model would give the desired output. So There's chance to be noise in this case

Estimation: As it is small size project, the shape predictor model is used, Function Points Analysis is used for the cost estimation.

Number of External Input = 1

Number of External Output = 1

Number of External Inquiry = 1

Number of Internal Logical File = 1

Number of External Interface File = 6

Function Point Analysis

Functional Unit	Count	Complexity	Total	Functional Unit Totals
External Input	1	High = 6	6	6
External Output	1	High = 7	7	7

Functional Unit	Count	Complexity	Total	Functional Unit Totals
External inquiry	1	High = 6	6	6
ILF	1	High = 15	15	15
EIF	2	Average = 7	14	
	4	High = 10	40	54
			UFP	88

CAF Calculation

Factors	Value
Backup and Recovery	0
Data Communication	5
Distributed Processing	3
Performance Critical	2
Existing Operating Environment	5
Online data entry	0
Input Transaction Over Multiple Screens	0
Master File Updated Online	0
Information Domain Values Complex	5
Internal Processing Complex	5
Code Designed to Reusable	5
Multiple Installations	5
Application Design for Change	5
Total	40

$$CAF = 0.65 + 0.01 * 40$$

$$= 1.05$$

$$FP = UFP * CAF$$

FP = 88* 1.05

FP = 92.4

Assuming that the average productivity for the system is 7 FP/PM and labour rate is 30000 per month.

Cost per FP = 30000/7

= 4285.71 Rs.

Estimated Cost of Complete Project = 92.4 * 4285.71

= 396000 Rs.

Effort = 92.4/7 = 13.2

The estimated cost is 396000 and the effort would 14 person per month.

ACRONYMS

CV	Computer Vision
MAR	Mouth Aspect Ratio
EAR	Eyes Aspect Ratio
FP	Function Points
CAF	Complexity Adjustment Factor
UFP	Unadjusted Function Point
PM	Person Month
SRS	Software Requirement Specification
SDD	Software Design Document
SQA	Software Quality Assurance
DFD	Data Flow Diagram
ER	Entity Relationship

White Box Testing

White Box Testing is used to design test cases of the the project. Here, Basic Path Testing has been used to derive the test cases. Basic Path Testing uses Flow Graph Notation to represent the control flow.

The module has been developed in the Python Interpreter.

Here's the screenshots of the main module.

```
10  def helper():
11
12      # Eyes and mouth threshold value
13      eyeThresh = 0.25
14      mouthThresh = 0.60
15
16      # frame to check
17      frame_check_eye = 5
18      frame_check_mouth = 5
19
20      # Initializing the Face Detector object
21      detect = dlib.get_frontal_face_detector()
22
23      # Loading the trained model
24      predict = dlib.shape_predictor("../model/shape_predictor_68_face_landmarks.dat")
25
26      # Getting the eyes and mouth index
27      (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
28      (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
29      (mStart, mEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["mouth"]
30
31      # Initializing the Video capturing object
32      cap=cv2.VideoCapture(0)
33
```

```
    # Initializing the flags for eyes and mouth
    flag_eye=0
    flag_mouth=0

    # Calculating the Euclidean distance between facial landmark p
    while True:
        ret, frame=cap.read()
        frame = imutils.resize(frame, height = 800, width=1000)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        subjects = detect(gray, 0)
        for subject in subjects:
            shape = predict(gray, subject)
            shape = face_utils.shape_to_np(shape)
            leftEye = shape[lStart:lEnd]
            rightEye = shape[rStart:rEnd]
            mouth = shape[mStart:mEnd]
            leftEAR = eye_aspect_ratio(leftEye)
            rightEAR = eye_aspect_ratio(rightEye)
            ear = (leftEAR + rightEAR) / 2.0
            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            mar = mouth_aspect_ratio(mouth)
            mouthHull = cv2.convexHull(mouth)

```

```
            # Drawing the overlay on the face
            cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [mouth], -1, (255, 0, 0), 1)
            cv2.putText(frame, "Eye Aspect Ratio: {}".format(ear), (5, 50),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,255), 2)
            cv2.putText(frame, "Mouth Aspect Ratio: {}".format(mar), (5, 80),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,255), 2)

            # Comparing threshold value of Mouth Aspect Ratio (MAR)
            if mar > mouthThresh:
                flag_mouth += 1
                if flag_mouth >= frame_check_mouth:
                    cv2.putText(frame, "****************** SUBJECT IS YAWNING ****
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255), 2)
            else:
                flag_mouth = 0

```

```
            # Comparing threshold value of Eye Aspect Ratio (EAR)
            if ear < eyeThresh:
                flag_eye += 1
                if flag_eye >= frame_check_eye:
                    cv2.putText(frame, "****************** SUBJECT IS SLEEPING
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            else:
                flag_eye = 0

        # Plotting the frame
        cv2.imshow("Frame", frame)

        # Waiting for exit key
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break

    # Destroying all windows
    cv2.destroyAllWindows()
    cap.stop()
```

The flow graph has been designed on the basic of the code.

Total Number of Edges in the Flow Graph, E = 18

Total Number of Nodes in the Flow Graph, N = 11

Total Number of Predicate Nodes, P = 8

Cyclomatic Complexity for the Flow Graph

$V(G) = E - N + 2$

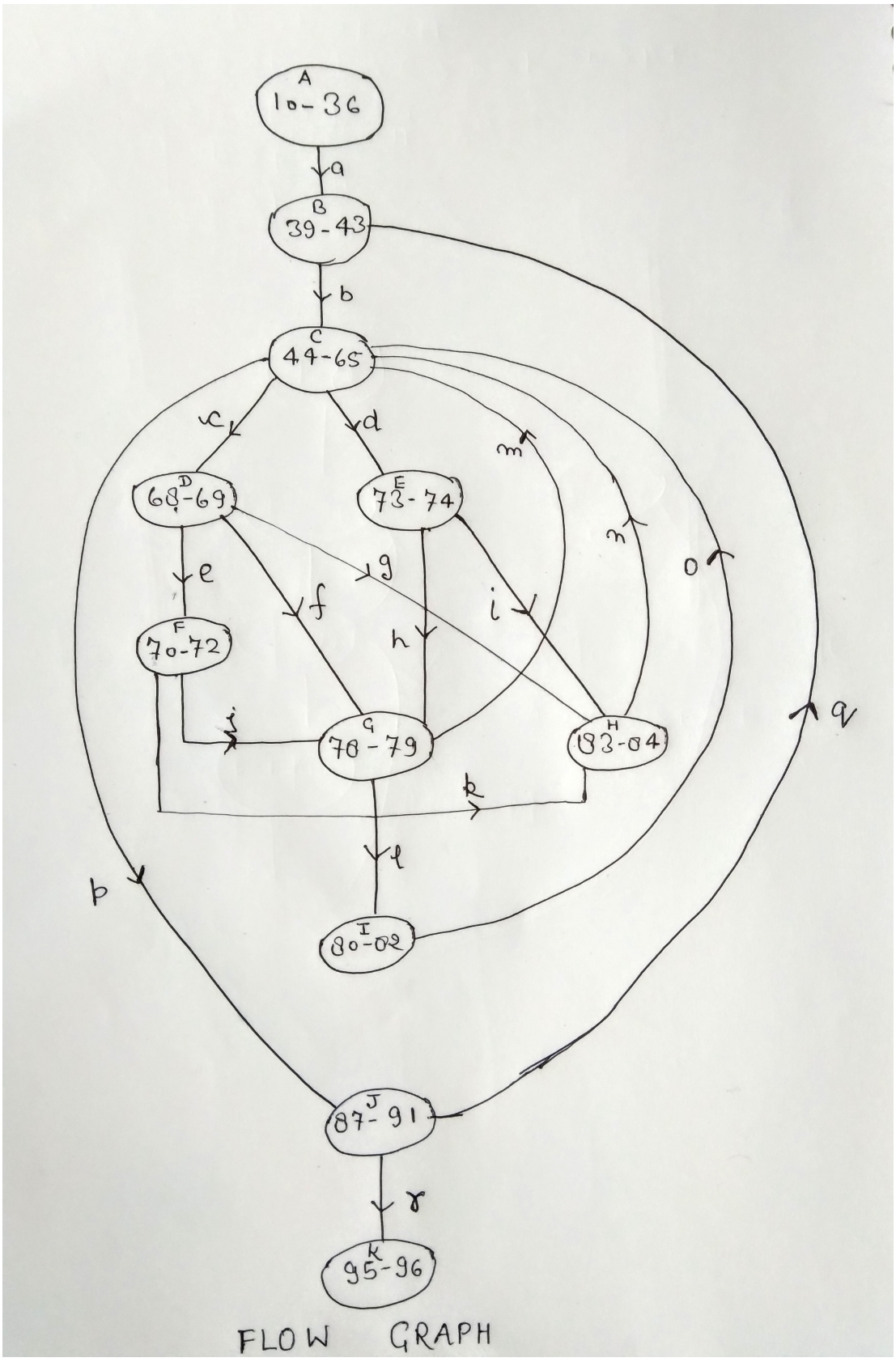
$= 18 - 11 + 2$

$= 9$

$V(G) = P + 1$

$= 8 + 1 = 9$

Here's the Flow Graph to the module.



FLOW GRAPH

Here's the Predicate Nodes : B, C, D, E, F, G, H, I

Graph Matrix												
	A	B	C	D	E	F	G	H	I	J	K	
A		1										1 - 1 = 0
B			1									1 - 1 = 0
C				1	1					1		3 - 1 = 2
D						1	1	1				3 - 1 = 2
E							1	1				2 - 1 = 1
F							1	1				2 - 1 = 1
G			1						1			2 - 1 = 1
H			1									1 - 1 = 0
I			1									1 - 1 = 0
J		1									1	2 - 1 = 1
K												
												8 + 1 = 9

Number of Connections would be 9. The Cyclomatic complexity would be 9.

Number of Independent Paths would be 9.

Path 1: A -> B -> C -> D -> F -> G -> I -> C -> J -> K

Path 2: A -> B -> C -> D -> F -> G -> C -> J -> K

Path 3: A -> B -> C -> D -> G -> C -> J -> K

Path 4: A -> B -> C -> D -> G -> I -> C -> J -> K

Path 5: A -> B -> C -> D -> H -> C -> J -> K

Path 6: A -> B -> C -> D -> F -> H -> C -> J -> K

Path 7: A -> B -> C -> E -> G -> I -> C -> J -> K

Path 8: A -> B -> C -> E -> G -> C -> J -> K

Path 9: A -> B -> C -> E -> H -> C -> J -> K

Black Box Testing

Black Box Testing is used to focus on the functional requirements of the software. Here, Decision Table Testing is used to show how the system works in the real time environment.

CONDITIONS	Case 1	Case 2	Case 3	Case 4
MAR	F	T	F	T
EAR	F	F	T	T
NOTIFICATIONS	X	N	N	N

T - When the calculated aspect ratio is greater than the threshold value.

F - When the calculated aspect ratio is less than or equal to the threshold value or Face is not detected.

N - Notification is Displayed.

X - Nothing is displayed or Face is not detected.

Cases According to the Decision Table:

1. When Either no face is detected or the calculated aspect ratio is less than or equal to the threshold value, Nothing would be displayed.

2. When Mouth Aspect Ratio is greater than Mouth Threshold Value, Notification for ‘Yawning’ would be displayed.

3. When Eyes Aspect Ratio is greater than Eyes Threshold Value, Notification for ‘Sleeping’ would be displayed.

4. When Eyes Aspect Ratio and Mouth Aspect Ratio are greater than Eyes Threshold Value and Mouth Threshold Value respectively, Notifications for ‘Sleeping and Yawning’ would be displayed.