REAL TIME DROWSINESS DETECTION SYSTEM

A Report Submitted

in partial fulfilment of the requirements

for the degree of

Master of Computer Application

By

Anurag	**(20719)**
Himanshu Gupta	**(20729)**
Shashank Rai	**(20747)**
Vineet Singh	**(20760)**

Under the Supervision of

Mr. Vijay Kumar Tiwari

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KAMLA NEHRU INSTITUTE OF TECHNOLOPGY, SULTANPUR – 228118(U.P.)

(An autonomous Government Engineering Institute under 2f & 12B of UGC Act)

Affiliated to

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW (U.P.), INDIA

(Formerly Uttar Pradesh Technical University, Lucknow (U.P.), INDIA)

DECLARATION

We hereby declare that the work presented in this report entitled "**Real Time Drowsiness Detection System**", was carried out by me. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, We shall be fully responsible and answerable.

Name :
Roll. No. :
Branch :

(Candidate Signature)

CERTIFICATE

Certified that Anurag (20719), Himanshu Gupta (20729), Shashank Rai (20747) and Vineet Singh (20760) has carried out the project work presented in this report entitled "**Real Time Drowsiness Detection System**" for the award of **Master of Computer Application** from Kamla Nehru Institute of Technology, Sultanpur affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow under my supervision. The report embodies results of original work, and studies are carried out by the students themselves and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature
Mr. Vijay Kumar Tiwari
(Asst. Professor)
(KNIT, Sultanpur)

Date :

ABSTRACT

Driver fatigue is one of the major causes of accidents in the world. Detecting the drowsiness of the driver is one of the surest ways of measuring driver fatigue.

A Real Time Drowsiness Detection System has been developed, using a non- intrusive machine vision based concepts. The system uses a small monochrome security camera that points directly towards the driver's face and monitors the driver's eyes in order to detect fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver. This report describes how to find the eyes, and also how to determine if the eyes are open or closed.

The system deals with using information obtained for the binary version of the image to find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by computing the horizontal averages in the area. Taking into account the knowledge that eye regions in the face present great intensity changes, the eyes are located by finding the significant intensity changes in the face. Once the eyes are located, measuring the distances between the intensity changes in the eye area determine whether the eyes are open or closed. A large distance corresponds to eye closure. If the eyes are found closed for 5 consecutive frames, the system draws the conclusion that the driver is falling asleep and issues a warning signal. The system is also able to detect when the eyes cannot be found, and works under reasonable lighting conditions.

Keywords : OpenCV, Drowsiness, Camera, Frames, Eyes, Face, Alarm.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1
INTRODUCTION

This chapter presents the overview of the problem definition selected along with the description of existing such systems which served as motivation behind the selection and the proposed system.

1.1 PROBLEM DEFINITION

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves a sequence of images of a face, 7and the observation of eye movements and blink patterns. The analysis of face images is a popular research area with applications such as face recognition, virtual tools, and human identification security systems. This project is focused on the localization of the eyes, which involves looking at the entire image of the face, and determining the position of the eyes by a self-developed image-processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes are opened or closed, and detect fatigue.

1.2 EXISTING SYSTEM

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

1.3 PROPOSED SYSTEM

The proposed system will try to accomplish the following tasks:

- Driver drowsiness detection is a car safety technology which helps to save the life of the driver by preventing accidents when the driver is getting drowsy.
- The main objective is to first design a system to detect driver's drowsiness by continuously monitoring retina of the eye.

- The system works in spite of driver wearing spectacles and in various lighting conditions.
- To alert the driver on the detection of drowsiness by using buzzer or alarm.

1.4 SCOPE OF THE PROJECT

Currently, the system can detect only slightly tilted face of the driver. So, we are looking for an approach to easily detect tilted faces so that the application can be practically used.

This system only looks at the number of consecutive frames where the eyes are closed. At that point it may be too late to issue the warning. By studying eye movement patterns, it is possible to find a method to generate the warning sooner.

Using 3D images is another possibility in finding the eyes. The eyes are the deepest part of a 3D image, and this maybe a more robust way of localizing the eyes.

Adaptive binarization is an addition that can help make the system more robust. This may also eliminate the need for the noise removal function, cutting down the computations needed to find the eyes. This will also allow adaptability to changes in ambient light.

CHAPTER 2
LITERATURE OVERVIEW

This chapter describes the research that was undertaken to learn about the different aspects of Drowsiness Detection System.

2.1 OVERVIEW

Industrialized countries, drowsiness has been estimated to be involved 1in0 20% to 23% of all crashes. Increase in such fatal crashes has urged many automobile companies now-a-days to build their own software that will detect whether a driver is feeling sleepy and fatigue well in advance. We have thus build a system that will detect when drivers are becoming drowsy and promise to be a valuable aid in preventing accidents.

The system deals with using information obtained using Haar-based classifier to find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by looking in the area of the cropped face to determine whether the eyes are open or closed. If the eyes are found closed for specific no. of frames, the system draws the conclusion that the driver is falling asleep.

2.2 EXISTING SYSTEM

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

Driver Drowsiness Detection System and Techniques: According to the studies it has been observed that when the drivers continuously drive without taking a break they tend to run a high risk of becoming drowsy. Study shows that accidents occur due to sleepy drivers in need of a rest, which means that road accidents occurs more due to drowsiness rather than drink-driving. Attention assist can warn of inattentiveness and drowsiness in an extended speed range and notify drivers of their current state of fatigue and the driving time since the last break, offers adjustable sensitivity and, if a warning is emitted, indicates nearby service areas in the COMAND navigation system.

Implementation of the Driver Drowsiness Detection System: This paper is about making cars more intelligent and interactive which may notify or resist user under unacceptable conditions, they may provide critical information of real time situations to rescue or police or owner himself. Driver fatigue resulting from sleep disorders is an important factor in the increasing number of accidents on today's roads. In this paper, we describe

a real-time safety prototype that controls the vehicle speed under driver fatigue. To advance a system to detect fatigue symptoms in drivers and control the speed of vehicle to avoid accidents is the purpose of such a mode. In this paper, we propose a driver drowsiness detection system in which sensor like eye blink sensor are used for detecting drowsiness of driver .If the driver is found to have sleep, buzzer will start buzzing and then turns the vehicle ignition off.

Driver Drowsiness Detection System: One of the major cause of traffic accident is Driver‘s drowsiness. It is a serious highway safety problem. If drivers could be warned before they became too drowsy to drive safely, some of these crashes could be prevented. In order to reliably detect the drowsiness, it depends on the presentation of timely warnings of drowsiness. To date, the effectiveness of drowsiness detection methods has been limited by their failure to consider individual differences. Based on the type of data used, drowsiness detection can be conveniently separated into the two categories of intrusive and non-intrusive methods. During the survey, non-intrusive methods detect drowsiness by measuring driving behavior and sometimes eye features, through which camera based detection system is the best method and so are useful for real world driving situations. This paper presents the review of existed drowsiness detection techniques that will be used in this system like Circular Hough Transform, FCM, and Lab Color Space etc.

Drowsiness Detection System Using MATLAB: As the survey done, driver fatigue is the major reason why half (50 %) of road accidents takes place. It is an interesting challenge in today’s date to detect drowsiness in order prevent accidents. Various experiments have been done earlier with regard to the drowsiness detection of driver. In the past few years, many countries became curious to pay high attention towards driver’s safety problems. Researchers have been making various efforts to invent techniques for the detection of drowsy driver such as monitoring of road and physiological techniques which requires the contact of electrode with our body such as chest, face making it an implantable method.

Detecting Driver Drowsiness Based on Sensors: Researchers have attempted to determine driver drowsiness using the following measures: (1) vehicle-based measures; (2) behavioral measures and (3) physiological measures. A detailed review on these measures will provide insight on the present systems, issues associated with them and the enhancements that need to be done to make a robust system. This paper reviews the three measures as to the sensors used and discuss the advantages and limitations of each. The various ways through which drowsiness has been experimentally manipulated is also discussed. It is concluded that by designing a hybrid drowsiness detection system that combines non-intrusive physiological measures with other measures one would accurately determine the drowsiness level of a driver. A number of road accidents might then be avoided if an alert is sent to a driver that is deemed drowsy.

CHAPTER 3

SOFTWARE PROJECT MANAGEMENT PLAN (SPMP)

This chapter describes in detail the various aspects which were considered essential for project management like budget, risks, schedule etc. and their effects on the project. The project timeline and milestones are represented using a Gantt Chart. The roles and responsibilities of all the team members are described.

3.1 INTRODUCTION

This introduction provides background information for the rest of the document. It briefly describes the project, the client deliverables, the project milestones, and expected document changes.

3.2 PROJECT OVERVIEW

The system will capture the video i.e system will continuously capture image frames from the webcam. After capturing the frames, they are pre-processed before the application of algorithms. After pre-processing, face and eyes of the driver detected. There is already a template which is stored in the database and if the system detects the drowsiness of the driver for more than specified period of time then the firing alarm will alert the driver.

3.2.1 PROJECT DELIVERABLES

Preliminary Project Plan ..... 10 Feb 2022

Synopsis ..... 24 Feb 2022

Software Project Management Plan ..... 01 March 2022

Software Requirement Specification ..... 15 March 2022

Data Collection ..... 20 March 2022

Data Preparation ..... 30 March 2022

Model Training ..... 05 April 2022

Face Detection Module ..... 10 April 2022

Eyes Detection Module ..... 15 April 2022

Software Design Document ..... 20 April 2022

Software Test Document ..... 22 April 2022

Testing Module ..... 05 May 2022

Demo ..... 10 May 2022

3.2.2 Evolution of This Document

This document will be updated as the project progresses. Updates shou1d be expected in the following sections:

References - updated as necessary

Definitions, acronyms, and abbreviations - updated as necessary.

Organizational Structure will be updated as the team leaders are assigned for each phase.

Technical Process - this section will be revised appropriately as the requirements and design decisions become clearer.

Schedule – as the project progresses, the schedule will be updated accordingly.

Table 3.1 Document Revision History

Revision	Date	Updated By	Update Comments
1.0	04th March	Anurag, Shashank Rai, Vineet Singh	Additional functionalities added.
2.0	10th April	Himanshu Gupta, Shashank Rai, Anurag, Vineet Singh	Final Project Plan.

References

1. Driver Drowsiness Detection Using Haar Classifier and Template Matching (IJARET – April 2015)
2. Study of the Eye-Tracking Methods Based Video (IEEE - July 2011)

3.3 Project Organization

3.3.1 Process Model

The process used for this project will be a Feature Driven Development Model such that each prototype allows us to update the project plan and other deliverables for missing areas or correctness.

3.3.2 Organizational Structure

Team Members –

- Anurag
- Himanshu Gupta
- Shashank Rai
- Vineet Singh

3.3.3 Organizational boundaries and interfaces

Team leader during each phase will be responsible for coordinating team meetings, updates, communications, and team deliverables

3.3.4 Project responsibilities

Entire project team is responsible for the successful delivery of the product. Team member assignments per deliverable according to expertise.

- Preliminary Project Plan
- Synopsis
- Software Project Management Plan
- Software Requirement
- Data Collection
- Data Preparation
- Model Training
- Face Detection Module
- Eyes Detection Module
- Software Design Document
- Software Test Document
- Testing Module
- Demo

3.4 Managerial Process

3.4.1 Management objectives and priorities

The objective of the project is to develop a Driver Drowsiness Detection System that functions as per overview defined above.

3.4.2 **Assumptions, dependencies, and constraint**

The project assumptions, dependencies and constraints are as follows

- Team of 4
- Equipment/Hardware and software availability

3.4.3 Risk management

- Application Crash risk
- Anaylsis risk

3.4.4 Monitoring and controlling mechanisms

- Weekly project status meetings
- Shared document repository (Google Drive/Email services)
- Project tracking
- Tracking utilizing baselines

3.4.5 Project Timeline

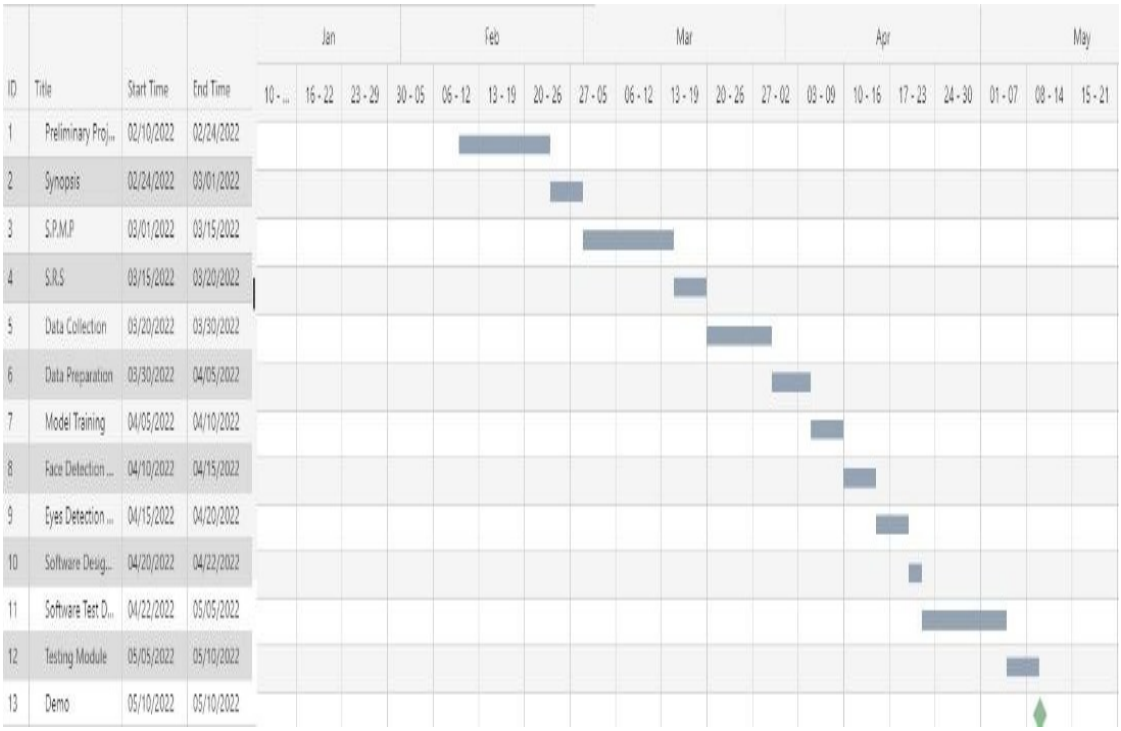


Fig. 3.1 Gantt Chart

3.5 Technical Process

3.5.1 Methods, tools, and techniques

The project will be implemented utilizing prototype model methodology, and tools such as Computer/Laptop, Webcam, Visual Studio, Open CV will be utilized.

3.5.2 Software documentation

Documentation such as Software Requirement Specification, SDD, Test Plan.

3.5.3 Project support functions

All project support documents will be completed in applicable phases.

CHAPTER 4
SOFTWARE REQUIREMENT SPECIFICATION (SRS)

The Software Requirements Specification (SRS) will provide a detailed description of the requirements for the Driver Drowsiness Detection System. This SRS will allow for a complete understanding of what is to be expected of the system to be constructed. The clear understanding of the system and its functionality will allow for the correct software to be developed for the end user and will be used for the development of the future stages of the project.

4.1 INTRODUCTION

The Software Requirements Specification (SRS) will provide a detailed description of the requirements for Driver Drowsiness Detection System. This SRS will allow for a complete understanding of what is to be expected of the system to be constructed. The clear understanding of the system and its functionality will allow for the correct software to be developed for the end user and will be used for the development of the future stages of the project. It will provide the foundation for the project. From the recorded specifications, the Driver Drowsiness Detection System can be designed, constructed, and finally tested.

4.1.1 Document Scope

A Driver Drowsiness Detection System has been developed, using a non-intrusive machine vision based concepts. The system uses a small monochrome security camera that points directly towards the driver's face and monitors the driver's eyes in order to detect fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver. This report describes how to find the eyes, and also how to determine if the eyes are open or closed. The algorithm developed is unique to any currently published papers, which was a primary objective of the project. The system deals with using information obtained for the binary version of the image to find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by computing the horizontal averages in the area. Taking into account the knowledge that eye regions in the face present great intensity changes, the eyes are located by finding the significant intensity changes in the face. Once the eyes are located, measuring the distances between the intensity changes in the eye area determine whether the eyes are open or closed. A large distance corresponds to eye closure. If the eyes are found closed for 5 consecutive frames, the system draws the conclusion that the driver is falling asleep and

issues a warning signal. The system is also able to detect when the eyes cannot be found, and works under reasonable lighting conditions.

4.1.2 Overview

The SRS is organized into two main sections. The first is The Overall Description and the second is the Specific Requirements. The Overall Description will describe the requirements of the Driver Drowsiness Detection System from a general high level perspective. The Specific Requirements section will describe in detail the requirements of the system.

4.2 Overall Description

4.2.1 Product Perspective

Driver Fatigue is a significant factor in a large number of vehicle accidents. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves a sequence of images of a face, and the observation of eye movements and blink patterns.

The analysis of face images is a popular research area with application such as face recognition, virtual tools, and human identification security systems. This project is focused on the localization of the eyes, which involves looking at the entire image of the face, and determining the position of the eyes by a self-developed image-processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes are opened or closed, and detect fatigue.

4.2.2 Product Functions

The system will perform the following functions:

- System will capture the video i.e. the system will continuously capture image frames from the webcam.
- After capturing the frames, the captured frames are pre-processed before the application of the algorithms.
- After pre-processing, face and eyes of the user are detected.
- There is already a template in the database which is compared with face and eyes of the driver and if the system detects the drowsiness for more than specified period of time then the firing alarm will alert the user.

4.2.3 Operating Environment

The system has two main modules, one for the detection of the face of driver and other for the detection of the eyes of driver. In general, the system should work in any operating environment.

4.2.4 Design and Implementation Constraints

The most important aspect of implementing a machine vision system is the image acquisition. Any deficiencies in the acquired images can cause problems with image analysis and interpretation. Ambient light is required for detection. Examples of such problems are a lack of detail due to insufficient contrast or poor positioning of the camera: this can cause the objects to be unrecognizable, so the purpose of vision cannot be fulfilled.

4.3 External Interface Requirements

4.3.1 User Interfaces

First of all, the system will capture the video of the driver i.e the system will capture the image frames continuously. After that, the system will detect the face and eyes of the driver. Template is stored in the database and the face and eyes are compared with this template. If the system detect the drowsiness for more than set period of time then a firing alarm will alert the driver.

4.3.2 Hardware Interfaces

1. Minimum Specifications

   RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit

   Processor: 1 gigahertz (GHz) or faster processor.

2. Computer with a webcam OR a laptop.

4.3.3 Software Interfaces

1. Visual Studio
2. Open CV

4.4 System Requirements

4.4.1 Functional Requirements

- ***Calculating Real Time Video***

  The system should capture frames at 10 frames per second.

- ***Processing the frames***

  1. Captured frames will be pre-processed i.e. converted into grey scale.
  2. This system will be responsible for finding facial and eye regions.

- ***Capturing no. of Frames in which eyes are closed***

  System determines if the driver is drowsy based upon whether these frames exceed the threshold value we set,

- ***Generating Results/Output (Alarm)***

  Based on output, the system will decide whether the driver is drowsy and generate an alarm.

4.4.2 Non-Functional Requirements

- ***Security***

  System will start on ignition and hence the driver will have no control of the system other than turning the alarm off.

- ***Extensibility***

  New modules such as haptic feedback can be added in future versions.

- ***Usability***

  System is easy to learn and very efficient to use.

- ***Cost Effective***

  System is cost-effective.

CHAPTER 5
SOFTWARE DESIGN DOCUMENT (SDD)

The Software Design Document (SDD) will provide overall description for the architecture for the Driver Drowsiness Detection System. This SDD will allow for a complete understanding of the design of the system design. It includes requirements traceability matrix and description of components.

5.1 INTRODUCTION

5.1.1 Design Overview

This document is a high-level overview defining our testing strategy for the Driver Drowsiness Detection System. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. We will utilize testing criteria under the white box, black box, and system-testing paradigm. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process we will be applying the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation.

5.1.2 Requirements Traceability Matrix

The following is a table depicting which components are expected to satisfy which requirements.

Table 5.1 Traceability Matrix

Components Requirements	User	Input(Camera)	System	Output
Face Detection			X	X
Face Detection with Left Motion			X	X

Face Detection with Right Motion			X	X
Eye Detection within the face			X	X
Firing Alarm	X	X	X	

5.2 SYSTEM ARCHITECTURAL DESIGN

5.2.1 Chosen System Architecture

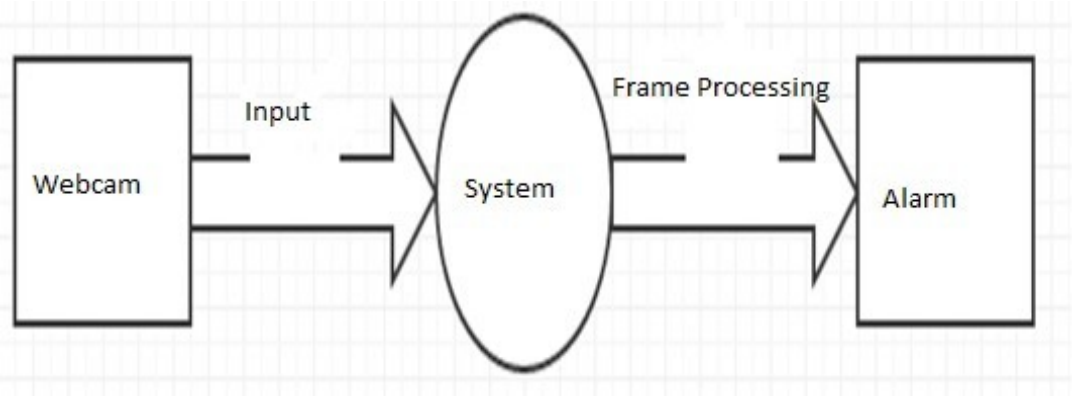


Figure 5.1 DFD Level 0

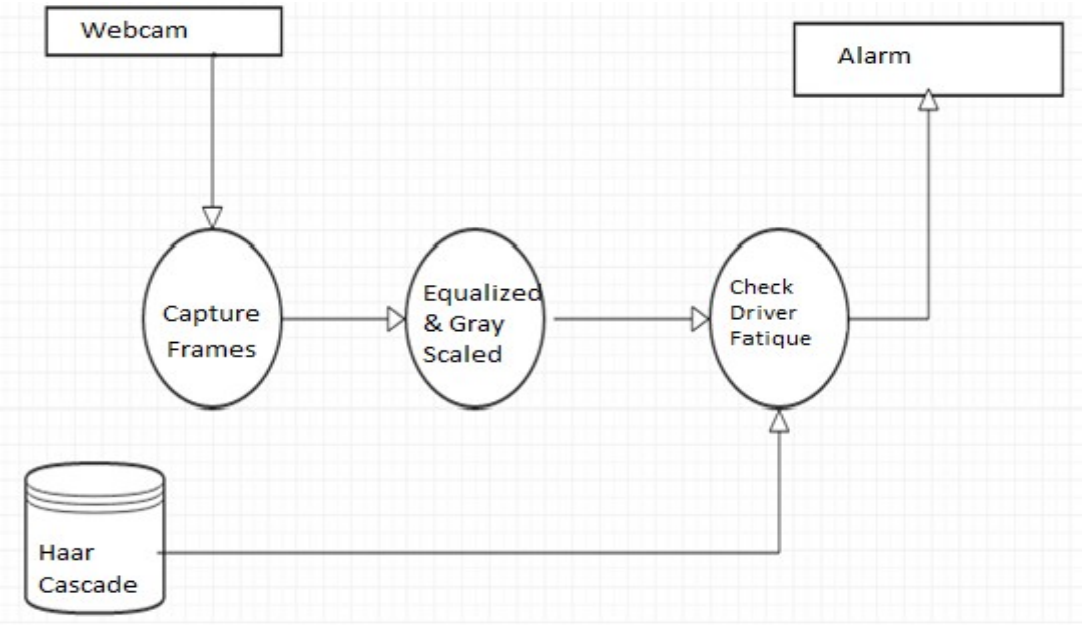


Figure 5.2 DFD Level 1

5.2.2 System Interface Description

Hardware Interfaces

The physical characteristic consists of computer system with a webcam along with a speaker for alarm.

Hardware Requirements:

CPU: 32 bit or 64 bit processor.

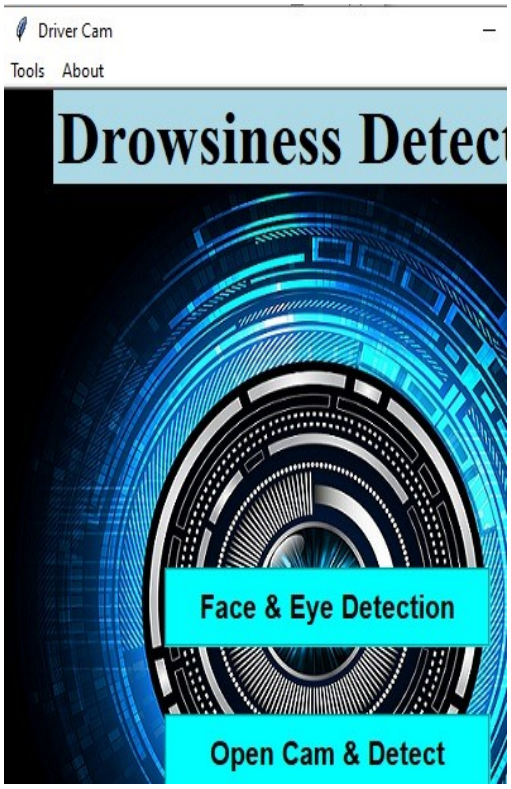
RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit Processor: 1 gigahertz (GHz) or faster processor.

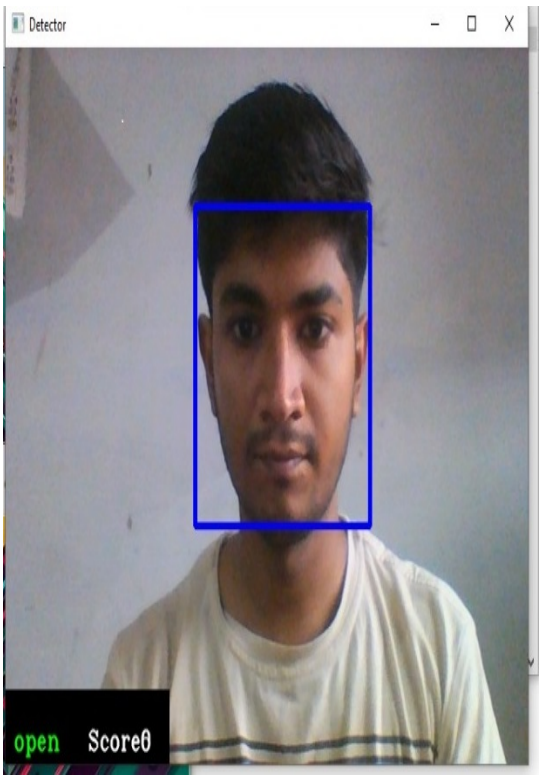
Software Interfaces

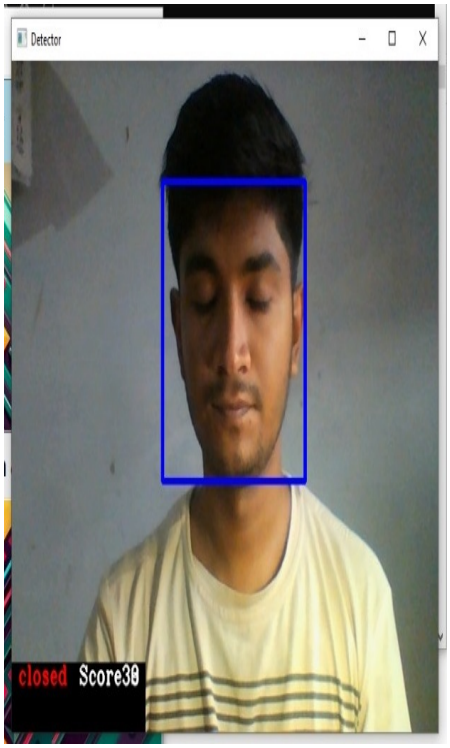
The system will consist various interfaces between the above mentioned software components. These interfaces will be built using standard tools.

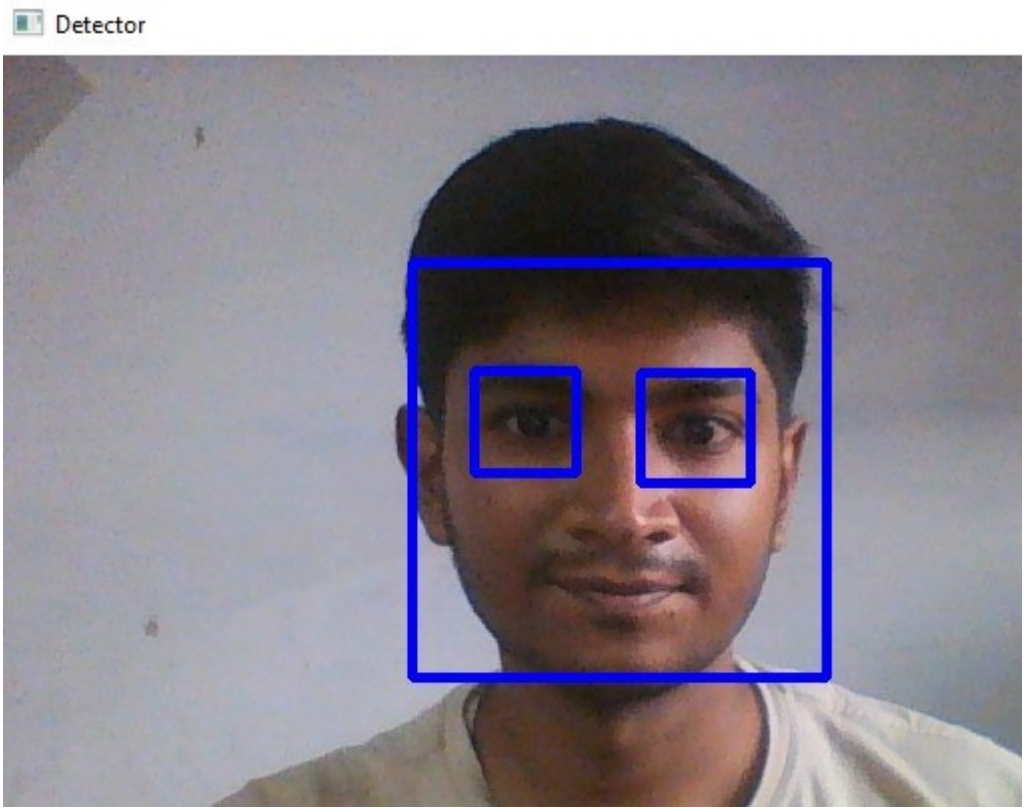
Visual Studio, Open CV.

5.3 USER INTERFACE DESIGN






Detector
closed Score30


Detector

Driver Drowsiness Detection Implementation

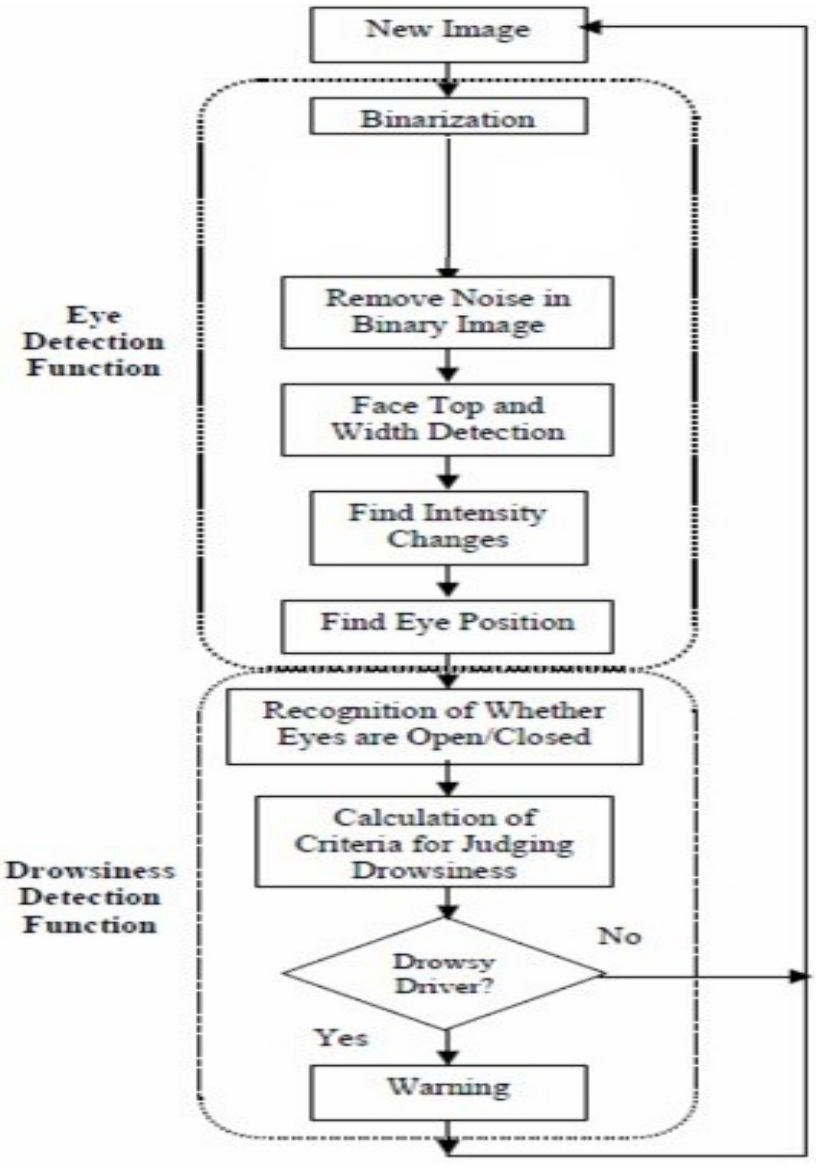


Figure 5.3 Driver Drowsiness Detection Implementation

5.4 UML Diagrams

5.4.1 Use Case Diagram

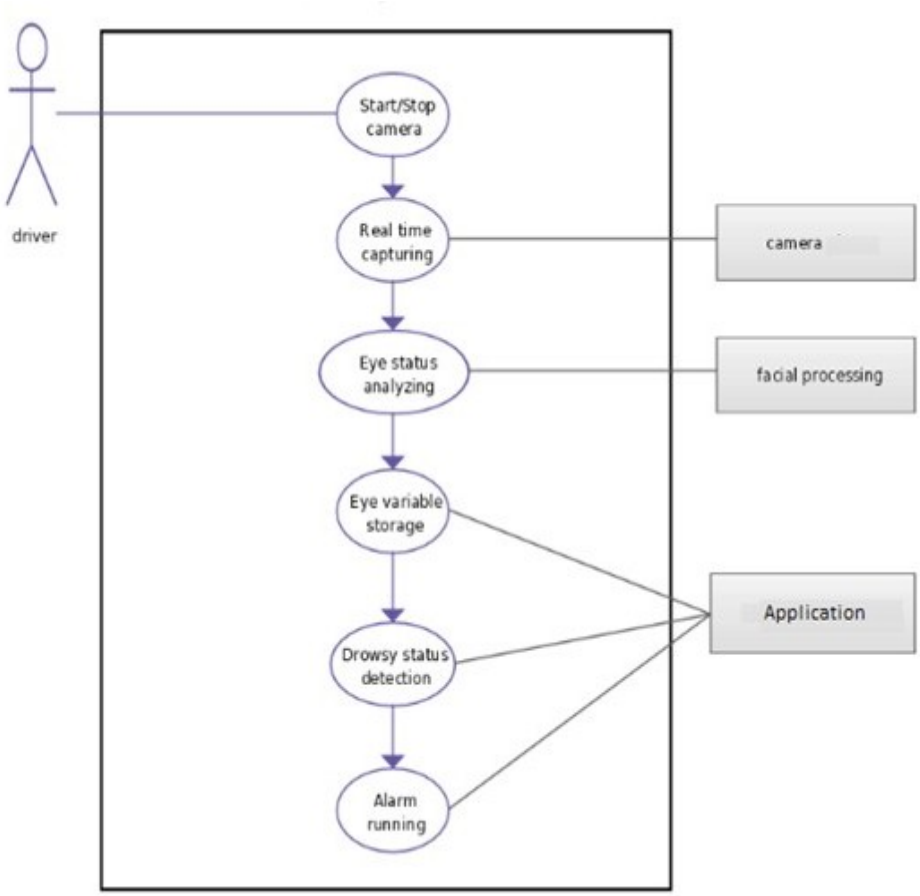


Figure 5.4 Use case Diagram

5.4.2 Activity Diagram

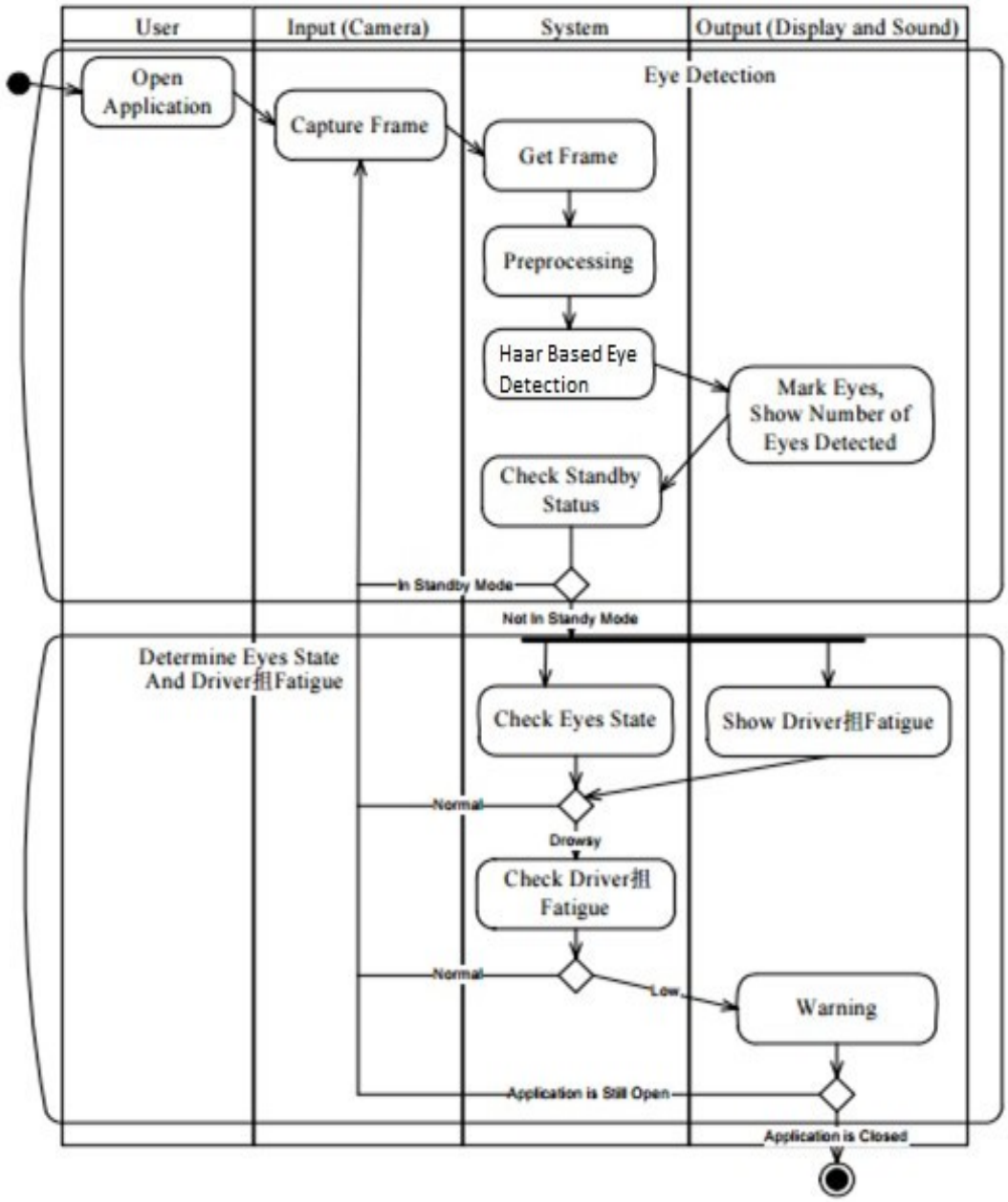


Figure 5.5 Activity Diagram

5.4.3 Sequence Diagram

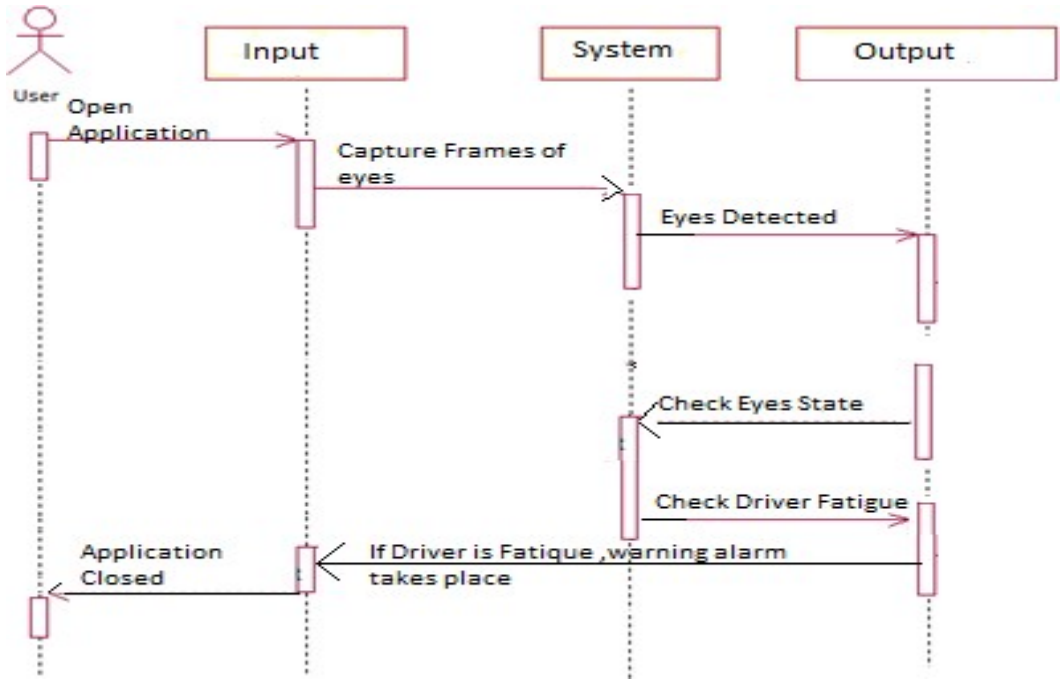


Fig 5.6 Sequence Diagram

CHAPTER 6
SOFTWARE TEST DOCUMENT

The Software Testing Document (STD) gives an overview of the testing approach for the given project Driver Drowsiness Detection System. It describes the basic test approach undertaken along with the Test Plan which will include the following: features to be tested, features not to be tested testing tools and environment. It also includes the test specifications.

6.1 Overview

This document is a high-level overview defining our testing strategy for the Driver Drowsiness Detection System. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. We will utilize testing criteria under the white box, black box, and system-testing paradigm. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process we will be applying the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation.

6.1.1 Target Audience

The document is targeted towards the reviewing the team who will validate whether the system works as specified based on the test cases. Also the development team would remain conscious as to whether the said system meets all the specified requirements.

6.2 Introduction

6.2.1 System Overview

This System Test Document (STD) will provide a detailed description of the testing strategy for the Driver Drowsiness Detection System. This STD will allow for a complete understanding of what is to be expected of the system to be constructed. The clear understanding of the system and its functionality will allow for the correct software to be developed for the end user and will be used for the development of the future stages of the project. This document covers all the tests that are to be conducted to ensure that the said system meets its requirements, both functional and non-functional.

6.2.2 Test Approach

The objective of our testing plan is to find and report as many bugs as possible to improve the integrity of our project. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to

utilize these functions is designed to be user-friendly. The system will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

The following represents the overall flow of the testing process:

- Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
- Identify which particular test(s) will be used to test each module.
- Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
- Identify the expected results for each test.
- Document the test case configuration, test data, and expected results.
- Perform the test(s).
- Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the System Test Report (STR).
- Unsuccessful testing requires a Bug Report to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem.
- Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

6.3 Test Plan

The diagram below outlines the Test Process approach that will be followed.

a. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

b. **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

c. **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

d. **Build Test Environment** includes requesting/building hardware, software and data set-ups.

e. **Execute System Tests –** The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

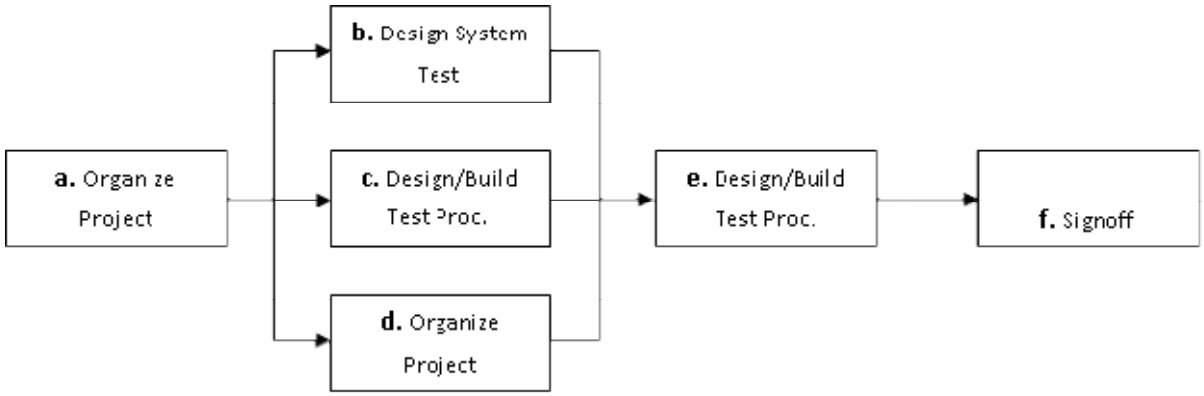


Figure 6.1 Test Process Flow

The following outlines the types of testing that will be done for system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the System Design Document. The template that will be used for designing test cases is shown in Figure 2.

6.3.1 Features to be tested

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function, validation and performance. And in both cases we will use the black-box method of testing.

6.3.2 Function Validation testing

The Driver Drowsiness Detection System will be tested based on the requirements to ensure that we built the right system. In doing this test, we will try to find the errors in the inputs and outputs, that is, we will test each function to ensure that it properly implements the various algorithms, and that the visual scene displays the description properly.

Table 6.1 Function Validation testing

Module	Expected Behavior
Face Detection	Detect the face of automobile driver when it is at the center.
Face Detection with Left Motion	Detect the face of automobile driver when it is positioned at the left.
Face Detection with Right Motion	Detect the face of automobile driver when it is positioned at the right.
Eye Detection within the face	Detect the automobile driver's eyes.
Firing Alarm	When system detect drowsiness for more than set period of time.

6.3.3 Testing tools and environment

6.3.3.1 Testing Team

Test Planner / Controller – Shashank Rai, Himanshu Gupta

- Ensure Phase 1 is delivered to schedule and quality
- Produce expected results
- Co-ordinate review and signoff of test conditions

Lead Tester – Anurag

- Identify test data
- Execute test conditions and mark-off results
- Prepare software error reports
- Ensure test systems outages/problems are reported immediately and followed up.

6.3.3.2 Hardware requirements

Computer with a webcam OR a laptop

6.3.3.3 Software requirements

1. Visual Studio
2. Open CV

6.4 Test Cases

Test Case Number	1	
Test Case Name	Face Detection	
Test Case Description	Check whether the face of the automobile driver is positioned at the center or not.	
Item(s) to be tested		
1	Input is accepted.	
2	Input is forwarded to the next module.	
Specifications		
Input	**Expected Output**	**Actual Output**
The face of the automobile driver is positioned at the center.	Input is accepted and the processing continues to the next module.	Input is accepted.
Procedural Steps		
1	Start the system.	
2	Position your Head at the center.	

Test Case Number	2
Test Case Name	Face Detection with Left Face profile
Test Case Description	Check whether the face of the automobile driver is detected when it is left profile.

Item(s) to be tested	
1	Input is accepted.
2	Input is forwarded to the next module.

Specifications		
Input	**Expected Output**	**Actual Output**
The face of the automobile driver is positioned at the left.	Input is accepted and the processing continues to the next module.	Input is accepted.

Procedural Steps	
1	Start the System
2	Position your Head in left profile.

Test Case Number	3
Test Case Name	Face Detection with Right Face Profile
Test Case Description	Check whether the face of the automobile driver is detected while in the right profile.

Item(s) to be tested	
1	Input is accepted.
2	Input is forwarded to the next module.

Specifications		
Input	**Expected Output**	**Actual Output**
The face of the automobile driver is positioned at the right.	Input is accepted and the processing continues to the next module.	Input is accepted.

Procedural Steps	
1	Start the System
2	Position your head in the Right profile.

Test Case Number	4
Test Case Name	Eye Detection within the face
Test Case Description	Check whether the eyes of the automobile driver has been detected or not.

Item(s) to be tested	
1	Input is accepted.

Specifications		
Input	**Expected Output**	**Actual Output**
.Eyes of the automobile driver.	Input is accepted and the processing continues to the next module.	Input is accepted.

Procedural Steps	
1	Start the System.
2	Position your head properly.

Test Case Number	5
Test Case Name	Firing Alarm
Test Case Description	When system detects drowsiness for more than set period of time.

Item(s) to be tested	
1	Input is accepted.

Specifications		
Input	**Expected Output**	**Actual Output**
System observes drowsiness of the driver for a specific period of time.	If the input detect drowsiness for more than specified time then firing alarm will take place.	Firing alarm will take place.

Procedural Steps	
1	Start the system
2	Keep your Head intact let the system detect eyes.
3	Close or blink your eyes for 1 second or more.

CHAPTER 7
REFERENCES

[1] Jihong Liu, Zhongfan Li ,Li Yang, Zhan Mei," Study of the Eye-tracking Methods Based on Video", 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks.

[2] Lu Yufeng, Wang Zengcai ,"Detecting Driver Yawning in Successive Images" IEEE 2009.

[3] Shabnam Abtahi, Behnoosh Hariri, Shervin Shirmohammadi,"Driver Drowsiness Monitoring Based on Yawning Detection" IEEE 2011.

[4] Xianghua Fan, Fuyou Zhang, Haixia Wang, Xiao Lu, "The System of Face Detection Based on OpenCV" , IEEE 2012.

[5] Weirwille, W.W. (2006), "Overview of Research on Driver Drowsiness Definition and Driver Drowsiness Detection," 14th International Technical Conference on Image Processing.

[6] Zeynep Orman , Abdulkadir Battal and Erdem Kemer, "A STUDY ON FACE, EYE DETECTION AND GAZE ESTIMATION", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.3, August 2011.

CHAPTER 8
DATA PREPARATION CODE

```
import os
import shutil
import glob
from tqdm import tqdm
import random

Raw_DIR= r'D:\Python37\Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\mrl
Eyes_2018_01'
for dirpath, dirname, filenames in os.walk(Raw_DIR):
for i in tqdm([f for f in filenames if f.endswith('.png')]):
if i.split('_')[4]=='0':
shutil.copy(src=dirpath+'/'+i, dst=r'D:\Python37\Projects\CV_Driver_Drowsiness_Detectio
n\MRL Eye Data\Prepared_Data\Close Eyes')

elif i.split('_')[4]=='1':
shutil.copy(src=dirpath+'/'+i,dst=r'D:\Python37\Projects\\CV_Driver_Drowsiness_Detectio
n\MRL Eye Data\Prepared_Data\Open Eyes')

raw_data = 'D:\Python37\Projects\CV_Drowsiness_Detection\Driver-Drowsiness-
Detection-using-Deep-Learning-main\MRL Eye Data\mrlEyes_2018_01'
for dirpath, dirname, filename in os.walk(raw_data):
for file in tqdm([f for f in filename if f.endswith('.png')]):
if file.split('_')[4] == '0':
path='D:\Python37\Projects\Driver-Drowsiness-Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\Close Eyes'
if not os.path.exists(path):              os.makedirs(path)
```

```
shutil.copy(src=dirpath + '/' + file, dst= path)
elif file.split('_')[4] == '1':
path='D:\Python37\Projects\Driver-Drowsiness-Detection-using-Deep-Learning-main\MRL Eye Data\Prepared_Data\Open Eyes'
if not os.path.exists(path):
os.makedirs(path)
shutil.copy(src=dirpath + '/' + file, dst= path)

def create_test_closed(source, destination, percent):
'''
divides closed eyes images into given percent and moves from
source to destination.

Arguments:
source(path): path of source directory
destination(path): path of destination directory
percent(float): percent of data to be divided(range: 0 to 1)
'''
path, dirs, files_closed = next(os.walk(source))
file_count_closed = len(files_closed)
percentage = file_count_closed * percent
to_move = random.sample(glob.glob(source + "/*.png"), int(percentage))

for f in enumerate(to_move):
if not os.path.exists(destination):
os.makedirs(destination)
shutil.move(f[1], destination)
print(f'moved {int(percentage)} images to the destination successfully.')
def create_test_open(source, destination, percent):
'''
divides open eyes images into given percent and moves from
source to destination.
```

```
Arguments:
source(path): path of source directory
destination(path): path of destination directory
percent(float): percent of data to be divided(range: 0 to 1)
'''
path, dirs, files_open = next(os.walk(source))
file_count_open = len(files_open)
percentage = file_count_open * percent
to_move = random.sample(glob.glob(source + "/*.png"), int(percentage))

for f in enumerate(to_move):
if not os.path.exists(destination):
os.makedirs(destination)
shutil.move(f[1], destination)
print(f'moved {int(percentage)} images to the destination successfully.')
create_test_closed(r'D:\Python37\Projects\CV_Drowsiness_Detection\Driver-Drowsiness-
Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\Close Eyes', r'D:\Python37\Projects\Drive
r-Drowsiness-Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\test\Close Eyes', 0.2)

create_test_open(r'D:\Python37\Projects\CV_Drowsiness_Detection\Driver-Drowsiness-
Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\Open Eyes', r'D:\Python37\Projects\Driver-
Drowsiness-Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\test\Open Eyes', 0.2)
```

CHAPTER 9
MODEL TRAINING CODE

```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data Augument
ation
tf.config.list_physical_devices('GPU')
batchsize=8


train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
zoom_range=0.2,width_shift_range=0.2, height_shift_range=0.2, validation_split=0.2)



train_data= train_datagen.flow_from_directory(r'D:\Projectc\Drowsiness\Driver-
Drowsiness-Detection-using-Deep-Learning-main\MRL Eye Data\Prepared_Data\train',
 target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='training' )


validation_data= train_datagen.flow_from_directory(r'D:\Projectc\Drowsiness\Driver-
Drowsiness-Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\train', target_size=(80,80),batch_size=batchsize,class_
mode='categorical', subset='validation')


test_datagen = ImageDataGenerator(rescale=1./255)


test_data = test_datagen.flow_from_directory(r'D:\Projectc\Drowsiness\Driver-Drowsiness-
Detection-using-Deep-Learning-
main\MRL Eye Data\Prepared_Data\test',target_size=(80,80),batch_size=batchsize,class_m
ode='categorical')
```

```
bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(
80,80,3)))
hmodel = bmodel.output
hmodel = Flatten()(hmodel)
hmodel = Dense(64, activation='relu')(hmodel)
hmodel = Dropout(0.5)(hmodel)
hmodel = Dense(2,activation= 'softmax')(hmodel)


model = Model(inputs=bmodel.input, outputs= hmodel)
for layer in bmodel.layers:
    layer.trainable = False


model.summary()


from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPlat
eau



checkpoint = ModelCheckpoint(r'D:\Python37\Projects\CV_Drowsiness_Detection\Driver-
Drowsiness-Detection-using-Deep-Learning-main\models',
                             monitor='val_loss',save_best_only=True,verbose=3)


earlystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3, restore_best_weigh
ts=True)


learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )


callbacks=[checkpoint,earlystop,learning_rate]


model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])


model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
```

```
                validation_data=validation_data,
                validation_steps=validation_data.samples//batchsize,
                callbacks=callbacks,
                 epochs=5)

acc_tr, loss_tr = model.evaluate_generator(train_data)
print(acc_tr)
print(loss_tr)
```

CHAPTER 10
USER INTERFACE And MAIN CODE

```
import numpy as np
import pandas
from pygame import mixer
import time
import cv2
from tkinter import *
import tkinter.messagebox
import tensorflow

root=Tk()
root.geometry('500x570')
root.resizable(False,False)
frame = Frame(root, relief=RIDGE, borderwidth=2)
frame.pack(fill=BOTH,expand=1)
root.title('Driver Cam')
frame.config(background='black')
label = Label(frame, text="Drowsiness Detector",bg='light blue',font=('Times 35 bold'))
label.pack(side=TOP)
filename = PhotoImage(file=r'D:\Projectc\Drowsiness\Driver-Drowsiness-Detection-using-Deep-Learning-main\eye2.png')
background_label = Label(frame,image=filename)
background_label.pack(side=TOP)
```

```
def hel():
  help(cv2)


def Contri():
  tkinter.messagebox.showinfo("Contributors","\n1. Anurag\n2. Himanshu Gupta \n3. Shas
hank Rai\n4. Vineet Singh")


def anotherWin():
  tkinter.messagebox.showinfo("About",'Driver Cam version v1.0\n Made Using\n-
OpenCV\n-Tensorflow\n-Numpy\n-Tkinter\n In Python 3')


 menu = Menu(root)
root.config(menu=menu)

subm1 = Menu(menu)
menu.add_cascade(label="Tools",menu=subm1)
subm1.add_command(label="Open CV Docs",command=hel)

subm2 = Menu(menu)
menu.add_cascade(label="About",menu=subm2)
subm2.add_command(label="Driver Cam",command=anotherWin)
subm2.add_command(label="Contributors",command=Contri)


  def Exit():
  exit()


def alert():
  mixer.init()
  alert=mixer.Sound('beep-07.wav')
  alert.play()
  time.sleep(0.1)
  alert.play()
def detect():
```

```
capture =cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
model = load_model(r'D:\Projectc\Drowsiness\Driver-Drowsiness-Detection-using-Deep-Learning-main\models')
blink_cascade = cv2.CascadeClassifier('CustomBlinkCascade.xml')
Score = 0

while True:
    ret, frame = capture.read()
    height,width = frame.shape[0:2]

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces= face_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)
    eyes= eye_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)

    cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3 )

    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3 )

        # preprocessing steps
        eye= frame[ey:ey+eh,ex:ex+ew]
        eye= cv2.resize(eye,(80,80))
        eye= eye/255
        eye= eye.reshape(80,80,3)
        eye= np.expand_dims(eye,axis=0)
        # preprocessing is done now model prediction
        prediction = model.predict(eye)
```

```
        cv2.imshow('Detector',frame)
        if cv2.waitKey(33) & 0xFF==ord('q'):
            break

    capture.release()
    cv2.destroyAllWindows()



def blink():
    capture =cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_
default.xml')
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
    model = load_model(r'D:\Projectc\Drowsiness\Driver-Drowsiness-Detection-using-Deep-
Learning-main\models')


    Score = 0


    while True:
        ret, frame = capture.read()
        height,width = frame.shape[0:2]


        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces= face_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)
        eyes= eye_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)


        cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)


        for (x,y,w,h) in faces:
            cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3 )


        for (ex,ey,ew,eh) in eyes:
            # preprocessing steps
```

```
        eye= frame[ey:ey+eh,ex:ex+ew]
        eye= cv2.resize(eye,(80,80))
        eye= eye/255
        eye= eye.reshape(80,80,3)
        eye= np.expand_dims(eye,axis=0)
        # preprocessing is done now model prediction
        prediction = model.predict(eye)

        # if eyes are closed
        if prediction[0][0]>0.50:
           Score=Score+3
           if(Score>15):
              cv2.putText(frame,'closed',(10,height-
35),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(0,0,255),
                 thickness=1,lineType=cv2.LINE_AA)
              cv2.putText(frame,'Score'-str(Score),(100,height-
35),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,2
55),
                 thickness=1,lineType=cv2.LINE_AA)

              try:
                 alert()
              except:
                 pass

        # if eyes are open
        elif prediction[0][1]>0.90:
           if(Score<15):
              cv2.putText(frame,'open',(10,height-
10),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(0,255,0),
                 thickness=1,lineType=cv2.LINE_AA)
              cv2.putText(frame,'Score'-str(Score),(100,height-
10),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,2
55),
```

```
                thickness=1,lineType=cv2.LINE_AA)
            if(Score>100):
              Score=Score-50

         Score = Score-10
        if (Score<0):
           Score=0


    cv2.imshow('Detector',frame)
    if cv2.waitKey(33) & 0xFF==ord('q'):
       break

   capture.release()
 cv2.destroyAllWindows()


but4=Button(frame,padx=5,pady=5,width=20,bg='cyan',fg='black',relief=GROOVE,comm
and=detect,text='fACE AND EYE DETECTION',font=('helvetica 15 bold'))
but4.place(x=127,y=300)

but5=Button(frame,padx=5,pady=5,width=20,bg='cyan',fg='black',relief=GROOVE,comm
and=blink,text='Open Cam & Detect',font=('helvetica 15 bold'))
but5.place(x=127,y=392)

but5=Button(frame,padx=5,pady=5,width=5,bg='cyan',fg='black',relief=GROOVE,text='E
XIT',command=Exit,font=('helvetica 15 bold'))
but5.place(x=210,y=478)

root.mainloop()
```