Driver Drowsiness Detection

Software Requirements Specification

Document

Submitted by:
Nidhi Uppoor - E053
Deboparna Banerjee - E065

Under Software Engineering Course for partial fulfillment of B.Tech Program

Under Guidance of
Dr. Dhirendra S Mishra
Professor, Computer Engineering
&
T. Vijayetha
Assistant Professor, Computer Engineering

At

SVKM's NMIMS
Mukesh Patel School of Technology Management and Engineering
Vile Parle West –Mumbai-56
A.Y. 2020-2021

Version: (1) **Date: (04/03/2021)**

Table of Contents

1. Introduction

Innovation in technology has continually impacted our lives positively, be it VR-aided surgery or connected cars. However, as always, there is still room for improvement. Road accidents, for example, especially those caused by driver drowsiness, is one of the biggest killers today. In an attempt to address this problem with technology, this report proposes a system based on 'Behavioral Metrics for Driver Drowsiness Detection' and provides a solution which can bring down the fatality rates. By capturing visual information of the driver and using this behavioral data with Artificial Intelligence, we are developing a real-time drowsiness detection system based on Convolutional Neural Networks. Here, we create a classification network to determine if the driver's eyes are closed or not and whether the driver is yawning or not. The proposed system captures the driver's images and analyses eye and mouth positions, it then uses this data and our CNN model to build a fatigue detection system. The system continuously monitors the driver's state. As soon as the driver gives any indication of drowsiness, the system sounds the alarm to alert the driver.

1.1 Purpose

We use this SRS to express the behavioral, performance, and development requirements of our project - Driver Drowsiness Detection. This is a safety technology that can prevent accidents that are caused by drivers who got drowsy while driving. It serves as the fundamental requirements document for the development of the project. It includes a description of every input into the system, every output from the system and all functions performed by the system in response to input or in support of an output. It is the exclusive requirements document to be used in development; all design and testing choices are compatible with this document.

1.2 Scope

The software product to be produced is a Driver Drowsiness Detection and Monitoring System. This software is a real time drowsiness detection system, which will constantly monitor the driver's eyelids and detect patterns of sleepiness. If it detects any signs of drowsiness, it will ring an alarm in hopes of making the driver alert.
One of the most prevailing problems across the globe nowadays is the booming number of road accidents. Driver's drowsiness or lack of concentration is considered as a dominant reason for such mishaps. This project therefore proposes an approach to implement a driver drowsiness alert system which would calculate the driver's sleepiness and prevent such accidents and deaths caused by the same.
The real time system will utilize a webcam to capture images continuously and measure the state of the eye according to the specified algorithm and give a warning if required. The user must have access to a device with a working camera and audio facilities and must grant the application the permission to access the same.The system must also allow continuous video capture.

1.3 Definitions, Acronyms, and Abbreviations.

1.3.1 Abbreviations and Acronyms

CNN - Convolutional Neural Network
COCOMO - Constructive Cost Model
KDSI - Kilo Delivered Source Instructions
Tdev - Time required for development
EAF - Effort Adjustment Factor
FP - Function Point
FPA - Function Point Analysis
PC - Product Complexity
PCA - Product Complexity Analysis
UFP - Unadjusted Function Points

1.3.2 Definitions

COCOMO - Constructive Cost Model which is a regression model based on LOC, i.e number of Lines of Code.

1.4 References

1. S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, “YawDD: A Yawning Detection Dataset”, Proc. ACM Multimedia Systems, Singapore, March 19 -21 2014.
2. Jabbar, R., Shinoy, M., Kharbeche, M., Al-Khalifa, K., Krichen, M., & Barkaoui, K. (2020). Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application. 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020, February, 237–242. https://doi.org/10.1109/ICIoT48696.2020.9089484
3. Machaca Arceda, V. E., Cahuana Nina, J. P., & Fernandez Fabian, K. M. (2020). A survey on drowsiness detection techniques? CEUR Workshop Proceedings, 2747, 152–161.
4. Kim, W., Lee, Y.-K., Jung, W.-S., Yoo, D., Kim, D.-H., & Jo, K.-H. (2020). An Adaptive Batch-Image Based Driver Status Monitoring System on a Lightweight GPU-Equipped SBC. IEEE Access, 8, 206074–206087. https://doi.org/10.1109/access.2020.3035393
5. Reddy, B., Kim, Y. H., Yun, S., Seo, C., & Jang, J. (2017). Real-Time Driver Drowsiness Detection for Embedded System Using Model Compression of Deep Neural Networks. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2017-July, 438–445. https://doi.org/10.1109/CVPRW.2017.59

1.5 Overview

Section 2 of this SRS document tells us the requirements in easy to understand language for the consumption of the customer. It explains the system, software and hardware interfaces for our product, along with memory constraints which are of more importance for customer/potential users. Here we also discuss the product functionalities, user characteristics, etc..
In section 3 we dive in deeper about our product and discuss various specifications like requirements, security needs, maintainability, etc..

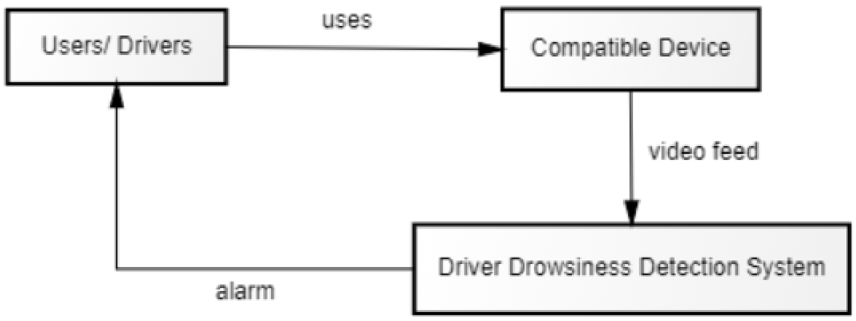
2. The Overall Description

In this section of the SRS document, a general description of the factors which influences the system and its requirement is involved. It supplies diagrams and models which gives a view of how the Driver Drowsiness Detection is going to behave, respond and interact with the customer.

2.1 Product Perspective

The proposed product is a Driver Drowsiness Detection and Monitoring System created to ensure safety of drivers against accidents caused due to fatigue. This research-oriented project is independent and self-contained. It only requires a compatible device with the ability to capture real time video feed to use. The application once started, will interact with the driver real time for checking their drowsiness state. It will continue to do so unless heavy drowsiness is detected upon which it will alert the user with the help of an alarm.

Out of all drowsiness detection methods, the behavioral measure that our system uses is the most accessible and efficient measure. It has proven to be highly effective and non intrusive. In comparison, the other measures require extra equipment. This also makes them more expensive.



2.1.1 System Interfaces

The proposed software is an independent system. The main functionality that it aims to accomplish is to efficiently detect fatigue in the driver and alert them about the same. In order to do this, we interact with the camera on the user's device using OpenCV and capture a video feed. This captured video feed is then used to detect drowsiness. Once drowsiness has been detected, our system once again interacts with the device audio interface using Pygame to ring an alarm.

2.1.2 Interfaces

- Our product, Driver Drowsiness Detection, has a command line interface. In order to deploy our product, users must have a python interpreter.
- It is designed for the general student population and is not yet suitable for a general audience.
- It is not designed in accordance with ADA (American with Disabilities Act).

2.1.3 Hardware Interfaces

No extra hardware is required except for a PC with a working web-cam which can capture and record video-feed.

2.1.4 Software Interfaces

The device would require a python interpreter along with OpenCV, TensorFlow, Keras libraries for Driver Drowsiness Detection.

2.1.5 Communications Interfaces

Our product has no communication requirements.

2.1.6 Memory Constraints

Application should need small memory for storing the data. It will process the real time data using a very small amount of memory as it does not save the video frames on the device. It will also take up a small amount of memory space for saving the detection model.

2.1.7 Operations

The primary mode of operation of our system is the active mode when the user is driving and the system is on. In this mode, Once the user starts using the system, the system then loads the classification model trained with driver drowsiness image data. This model is then fed with the enhanced and cropped images real-time.

In the passive, off state the system does not carry out any operations and waits till the user runs it again.

2.1.8 Site Adaptation Requirements

- The camera should be frontal faced for optimal results i.e. it should be placed on the driver's dashboard.

2.2 Product Functions

1) Capture user video feed and enhance the individual frames.
2) Detect face and eye and crop the image accordingly.
3) Send cropped images into the detection model.
4) Get the result based on values of the prediction model.
5) Ring the appropriate alarm based on the level of drowsiness detected.

2.3 User Characteristics

Software team needs to provide a simple and user friendly interface that is easy to cope with. A mid-level user must comprehend the system just by reading a simple instruction manual about the command list. Command list should not be very complex. An average user should easily understand and memorize commands.

2.4 Constraints

1. The user must have access to a device with a working camera and audio facilities and must grant the application the permission to access the same.
2. The device would require a python interpreter along with OpenCV, TensorFlow, Keras libraries for Driver Drowsiness Detection.

2.5 Assumptions and Dependencies

1) The lighting conditions in the use case must be bright enough.
2) The device on which the monitoring system will be run will be placed on the dashboard of the car to provide an optimal angle.
3) The face of the driver should be visible at all times during the use of the drowsiness detection system as the driver drives their car.
4) The device would require a python interpreter along with OpenCV, TensorFlow, Keras libraries for Driver Drowsiness Detection.

2.6 Apportioning of Requirements

The future version of our product might include a full GUI integrated software. This is so that we can include users with or without a python interpreter and can be accessed on a wider range of devices. This feature was delayed due to lack of resources.

2.7 Feasibility Study

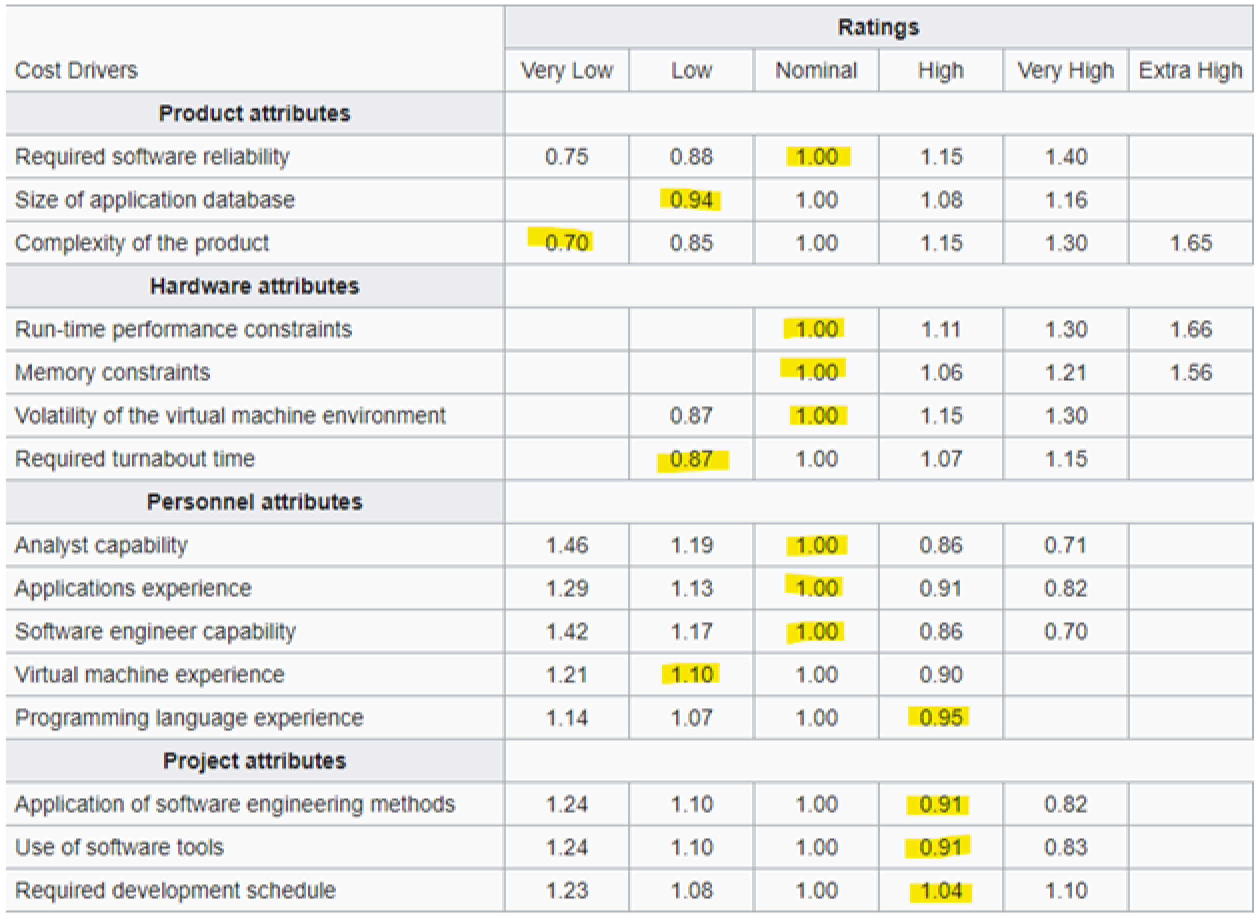
2.7.1 COCOMO Model

2.7.1.1 Basic COCOMO Model:

1. Real time Video Capture:
    a. Organic Development Project
    b. KDSI - 60 lines or 0.06
    c. Effort – 2.4*0.6^1.05 = 1.40 PM
    d. Tdev – 2.5*1.40^0.38 = 2.84 months
2. Ringing the Alarm to alert driver:
    a. Semi Detached Development Project
    b. KDSI – 15 lines or 0.015
    c. Effort – 3*0.015^1.12 = 0.027 PM
    d. Tdev – 2.5*0.027^0.35 = 0.70months
3. Processing the Images & Detecting Drowsiness:
    a. Embedded Development Project
    b. KDSI – 1500 lines or 1.5
    c. Effort – 3/6*1.5^1.2 = 5.85 PM
    d. Tdev – 2.5*5.85^0.32 = 4.4 months

2.7.1.2 Intermediate COCOMO Model:

1. Ringing Alarm/Semi-detached:



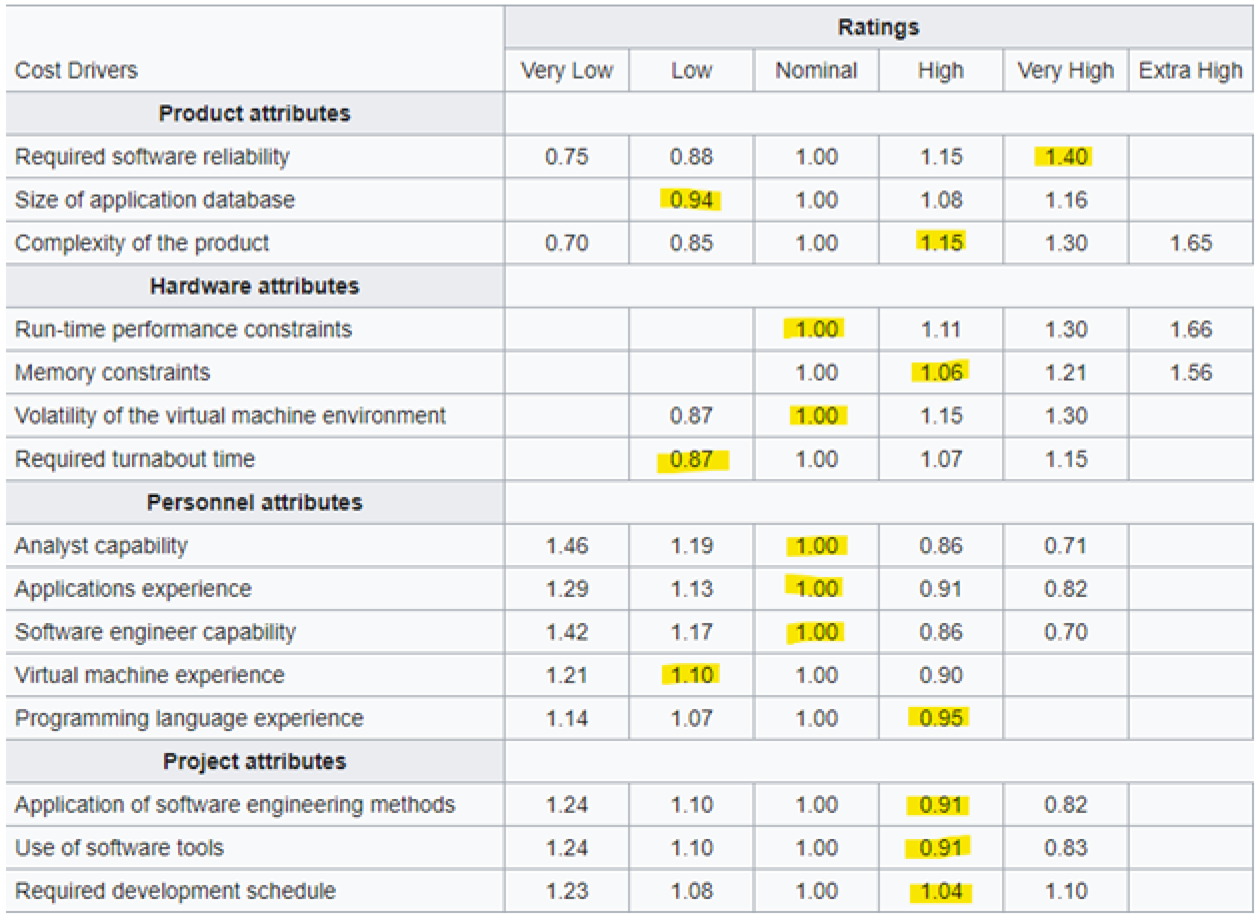
Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

EAF = 0.5152

Effort $_{\text{corrected}}$ = 0.027*0.5152 = 0.0139 PM

Tdev $_{\text{corrected}}$ = 2.5 * (0.0139^0.35) = 0.5597 months

1. Processing the Images & Detecting Drowsiness/Embedded:



Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

EAF = 1.256

$\text{Effort}_{\text{corrected}} = 5.85*1.265= 7.40025$ PM

$\text{Tdev}_{\text{corrected}} = 2.5 * (7.40025\text{\^{}}0.32) = 4.7434$ months

2.7.2 FPA Analysis

The files types in our project can be counted as follows:

		Weighting Factor			Count
		Simple	Average	Complex	
Inputs	Video Feed	3			3
Outputs	Alarm Ringtone	3			
	Alarm Text on Screen	3			6
Inquiries	-	-	-	-	-
Internal Logical Files	Image RGB to Grayscale		10		
	Image Resize		10		
	Image Cropping		10		30
External Interface Files	Yawn Classification Model			10	
	Eye Classification Model			10	20
Total UFP					59

Table Adjusted Function Points:

Number	Complexity Weighting Factor	Value
1	Backup and recovery	2
2	Data communications	0
3	Distributed processing	0
4	Performance critical	5
5	Existing operating environment	4
6	On-line data entry	0
7	Input transaction over multiple screens	0
8	Master files updated online	0
9	Information domain values complex	5
10	Internal processing complex	5
11	Code designed for reuse	5
12	Conversion/installation in design	4

13	Multiple installations	4
14	Application designed for change	5
	Total complexity adjustment value	39

- Total Unadjusted Function Points (UFP) = 59
- Product Complexity Adjustment (PC) = 0.65 + (0.01 * 39) = 1.04
- Total Adjusted Function Points (FP) = UFP * PC = 61.36

Burdened labor rate = Rs 800 per month, the cost per FP is approximately Rs 123.

Based on the FP estimate and the historical productivity data,

1. The total estimated project cost is [61.36 X 123 = Rs. 7547.28]
2. The estimated effort is [7547.28/800 = 9.4341 person-months.]

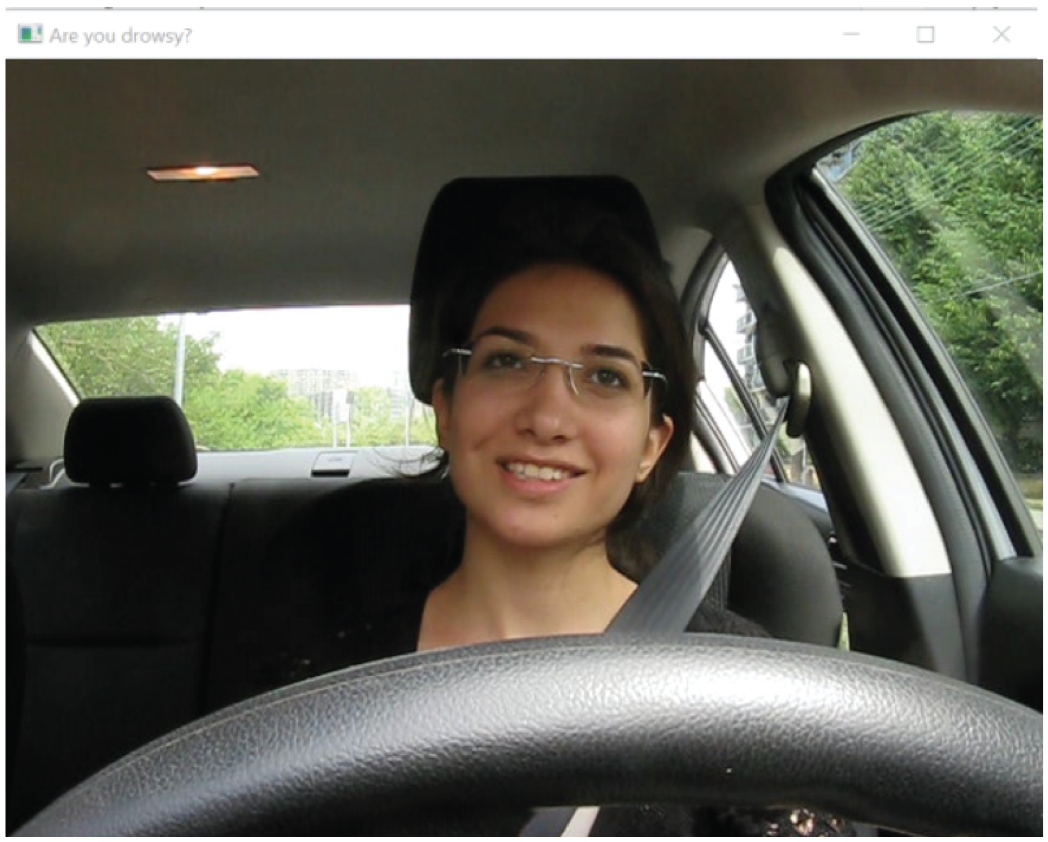
3. Specific Requirements

3.1 External interface requirements

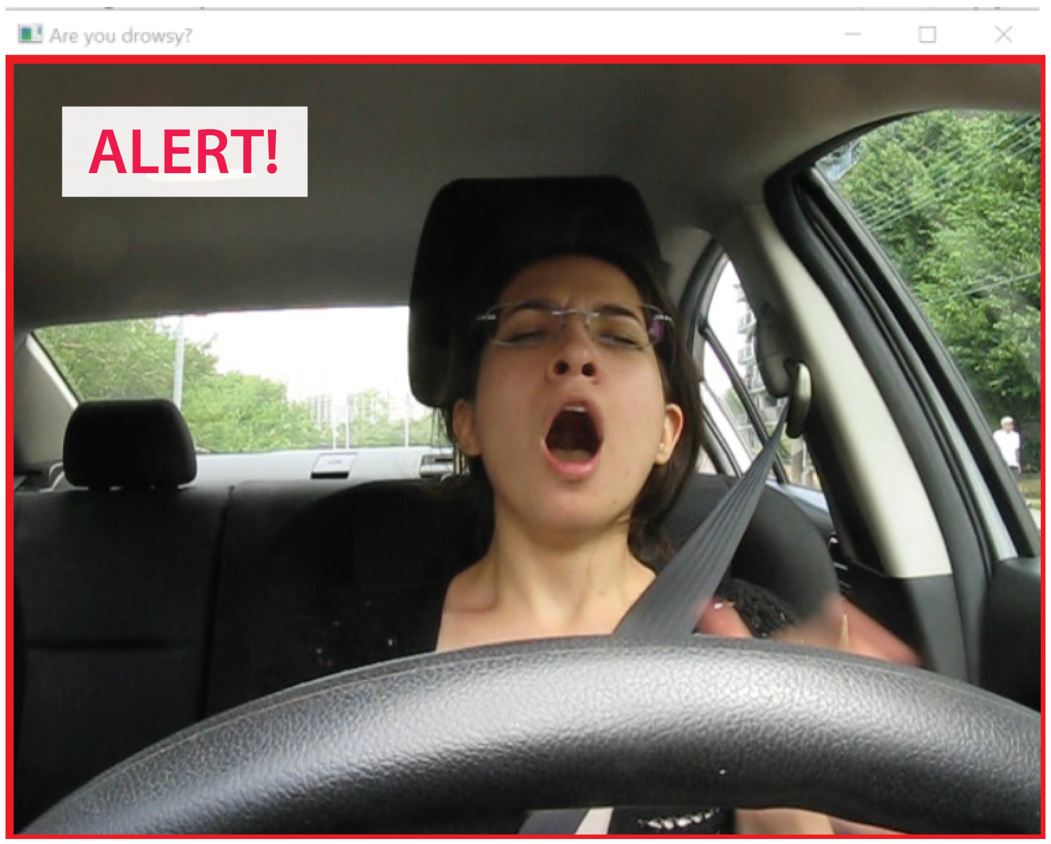
3.1.1 Specific Requirements

3.1.2 User interfaces

When the driver is found to be not drowsy, the system will not display any text as it could be distracting to the user. It will continue in this waiting and detecting loop until drowsiness is detected.



When drowsiness is detected, the system will display an alert message in red color in order to draw the attention of the user and make them aware of the need to be more careful and take a break if they must. It will also simultaneously ring an alarm for as long as the user is in a drowsy state in hopes of waking or alerting the user so that accidents and fatalities could be avoided.



3.1.3 Hardware interfaces:
No extra hardware is required except for a PC with a working web-cam which can capture and record video-feed.

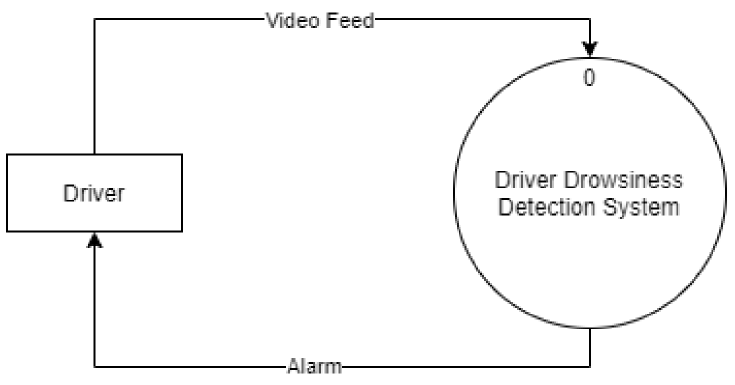
3.1.4 Software interfaces:
The device would require a python interpreter along with OpenCV, TensorFlow, Keras libraries for Driver Drowsiness Detection.

3.1.5 Communications interfaces
Our product has no communication requirements.

3.2 Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 0



3.2.1.1.1 Data entities

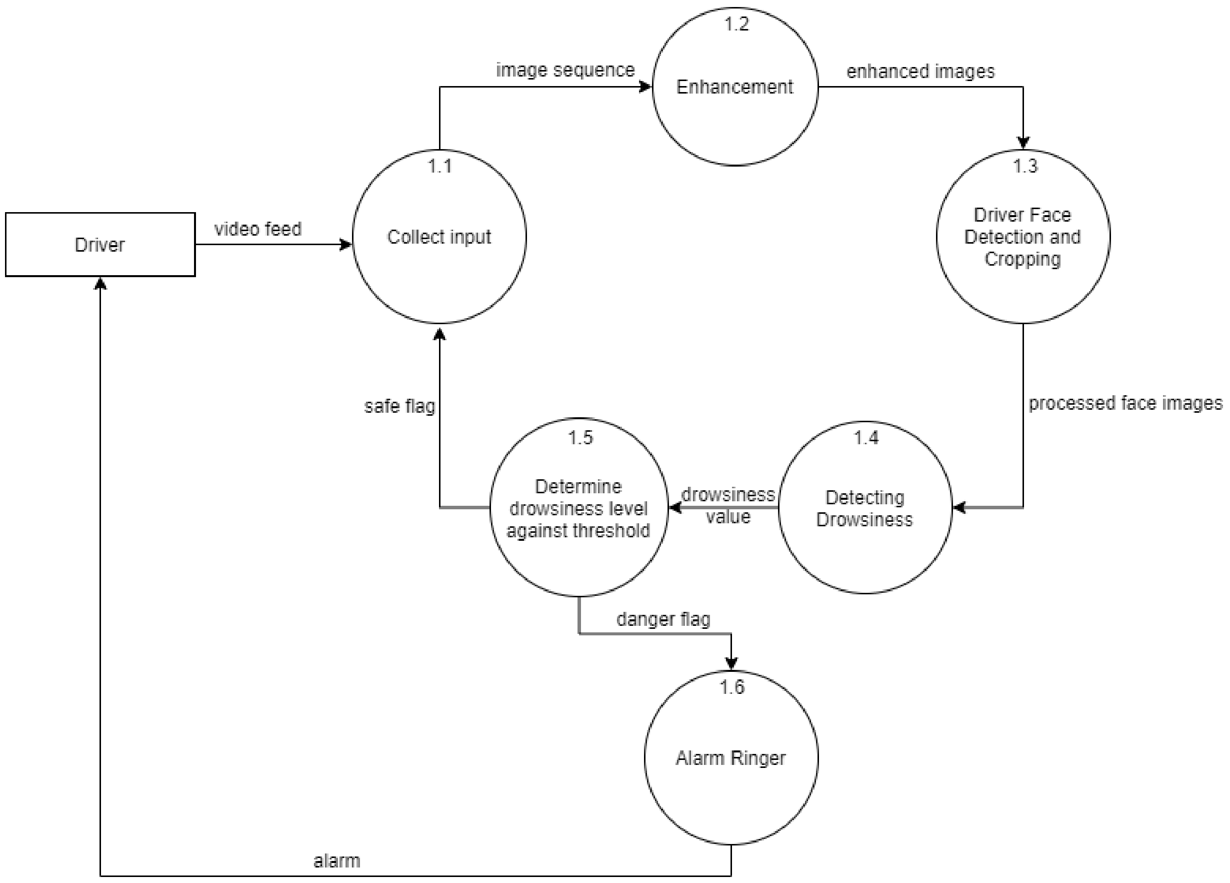
1) Video feed collected of the driver using the driver's device is fed into the drowsiness detection system.

2) Based on the results of the drowsiness detection system, an alarm is rung to alert the driver.

3.2.1.1.2 <u>Pertinent processes</u>

1) Driver Drowsiness Detection System takes in video feed as input and gives out alarm as output for when it detects drowsiness in the user.

3.2.1.2 Data flow diagram 1



3.2.1.2.1 **<u>Data entities</u>**

1) Driver's video feed is sent to the input collection process.

2) The image sequence of frames from the video feed is sent to the enhancement process.

3) The enhanced images are sent to the detection and cropping process.

4) The processed images in which faces have been detected and cropped are sent to the process that detects drowsiness.

5) The drowsiness value detected is sent to be checked against the threshold.

6) Based on the result obtained from the thresholding function, a safe flag is sent to collect input or a danger flag is sent to the alarm ringer.

7) Alarm is sent to the driver by the alarm ringer in order to alert them.

3.2.1.2.2 Pertinent processes

1) Collect Input: It takes in video feed from the user's device and breaks it down to frames for further processing.

2) Enhancement: It takes in one video frame at a time and applies enhancement techniques for better detection.

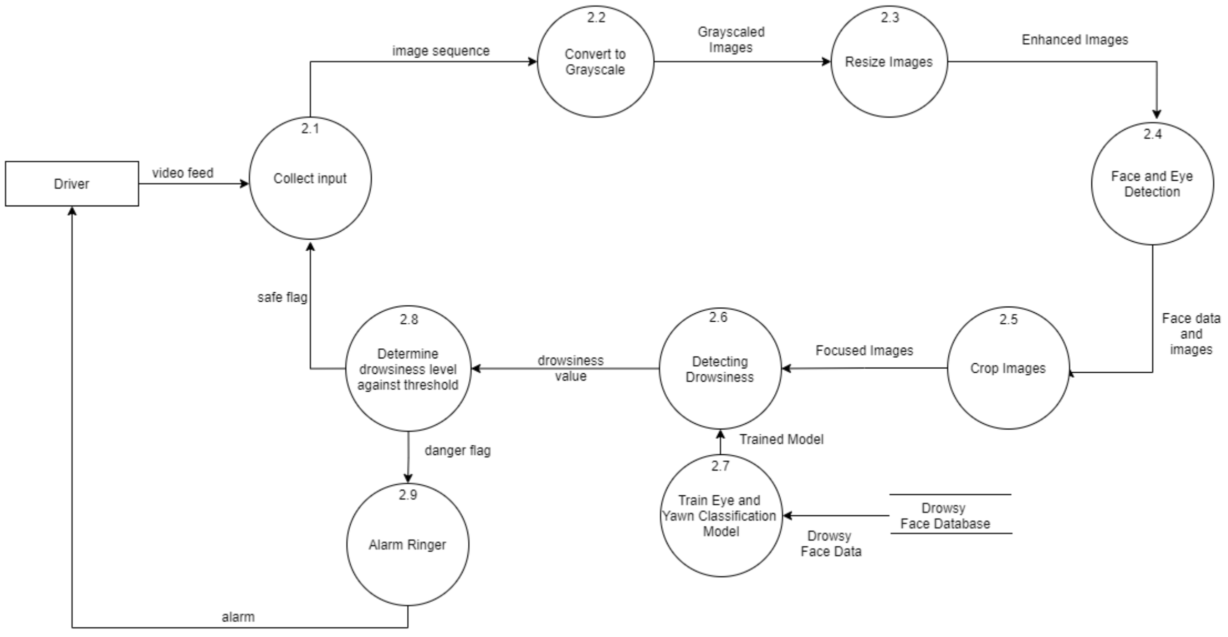
3) Driver Face Detection and Cropping: It detects the crucial features on the user’s face and crops it.

4) Detecting Drowsiness: It takes the processed images and detects the drowsiness using the trained model.

5) Determine drowsiness level against threshold: After detecting the drowsiness level of the user, it checks against the decided threshold value in case the drowsiness reaches a critical value.

6) Alarm Ringer: If the algorithm crosses the decided threshold then it rings out an alarm with the intention of alerting the driver.

3.2.1.3 Data flow diagram 2



3.2.1.3.1 Data entities

1) Driver’s video feed is sent to the input collection process.

2) The image sequence of frames from the video feed is sent to the gray scaling process.

3) The grayscale images are sent to the resizing function.

4) Enhanced images that have been converted to grayscale and resized are then sent to the face and eye detection process.

5) The face and eye data and images are sent to the cropping function.

6) The cropped and thus focused images are then fed into the drowsiness detection process.

7) Drowsy face data from a drowsy face database is used to train eye and yawn classification models.

8) The trained model is given to the drowsiness detection process to use.

9) The drowsiness value obtained is then sent to the threshold checking function.

10) Based on the result obtained from the thresholding function, a safe flag is sent to collect input or a danger flag is sent to the alarm ringer.

11) Alarm is sent to the driver by the alarm ringer in order to alert them.

3.2.1.3.2 Pertinent processes

1) Collect Input: It takes in video feed from the user's device and breaks it down to frames for further processing.

2) Convert to Grayscale: Convert each individual video frame to grayscale.

3) Resize Image: Resize the image to a uniform dimension.

4) Face and Eye Detection: From the processed image, detect face and eye in the given frame.
5) Crop Image: After facial parts crucial to the algorithm gets detected, we crop the rest out.

6) Detecting Drowsiness: We feed these optimal cropped images to our detection models to detect drowsiness levels.

7) Train Eye and Yawn Classifier Model: Train the model to classify yawn from no_yawn and eye open or close.

8) Determine drowsiness level against threshold: After detecting the drowsiness level of the user, it checks against the decided threshold value in case the drowsiness reaches a critical value.

9) Alarm Ringer: If the algorithm crosses the decided threshold then it rings out an alarm with the intention of alerting the driver.

3.2.2 Process descriptions

3.2.2.1 Process 1 : Collect Input

3.2.2.1.1 Input data entities: Video feed

3.2.2.1.2 Algorithm or formula of process: We used OpenCV library's `cap.read()` function to convert video feed to its frames.

3.2.2.1.3 Affected data entities: Input video feed.

3.2.2.2 Process 2: Convert to Grayscale

3.2.2.2.1 Input data entities: Sequence of images

3.2.2.2.2 Algorithm or formula of process: We use OpenCV library's convert function to convert image color from BGR to grayscale.

3.2.2.2.3 Affected data entities: Input data and images.

3.2.2.3 Process 3: Resize image

3.2.2.3.1 Input data entities: Grayscale images

3.2.2.3.2 Algorithm or formula of process: We use OpenCV library's resize function to resize the images.

3.2.2.3.3 Affected data entities: Grayscale images that are now resized

3.2.2.4 Process 4: Face and Eye Detection

3.2.2.4.1 Input data entities: Enhanced Images

3.2.2.4.2 Algorithm or formula of process: We use the Dlib library to get the face and eye pointers.

3.2.2.4.3 Affected data entities: Input images for the CNN.

3.2.2.5 Process 5: Crop Images

3.2.2.5.1 Input data entities: Images with face and eye pointers.

3.2.2.5.2 Algorithm or formula of process: We take the feature coordinates and perform array slicing.

3.2.2.5.3 Affected data entities: Input images for the CNN.

3.2.2.6 Process 6: Detecting Drowsiness

3.2.2.6.1 Input data entities: Cropped and enhanced images.

3.2.2.6.2 Algorithm or formula of process: We `model.predict()` function provided by TensorFlow.

3.2.2.6.3 Affected data entities: Drowsiness value.

3.2.2.7 Process 7: Train Yawn and Eye classification model

3.2.2.7.1 Input data entities: Dataset of eye and face images.

3.2.2.7.2 Algorithm or formula of process: We implement concepts of the CNNs using TensorFlow and Keras.

3.2.2.7.3 Affected data entities: The binary classification CNN model.

3.2.2.8 Process 8: Determine Drowsiness Level Against Threshold

3.2.2.8.1 Input data entities: Drowsiness value.

3.2.2.8.2 Algorithm or formula of process: Simple if-else algorithm to find optimal value.

3.2.2.8.3 Affected data entities: Danger or safe flag which decides if the alarm rings or not.

3.2.2.9 Process 9: Alarm Ringer

3.2.2.9.1 Input data entities: Resulting drowsiness level value

3.2.2.9.2 Algorithm or formula of process: We use the pygame module in python to ring an alarm.

3.2.2.9.3 Affected data entities: None

3.2.3 Data construct specifications

3.2.3.1 Construct 1 : Model training image data

3.2.3.1.1 Record type: Folder

3.2.3.1.2 Constituent fields: Sub-folders consisting of yawn, no yawn, eye open and eye closed data.

3.2.3.2 Construct 2: Images read in using OpenCV

3.2.3.2.1 Record type: List of lists

3.2.3.2.2 Constituent fields: Numerical representation of an image with BGR values based on intensity or a grayscale version with only intensity values.

3.2.3.3 Construct 3: Sounds for alarms

3.2.3.3.1 Record type: MP3 or WAV

3.2.3.3.2 Constituent fields: A series of numbers representing the amplitude of variations in air pressure.

3.3 Performance Requirements

- It should process the Eye and Yawn Detection algorithm in real-time.
- The system should have a low false prediction rate.

3.4 Design Constraints

- The program will be implemented using Jupyter Notebook.
- User private data and actions must be kept private.

3.5 Software system attributes

3.5.1 Reliability

- Driver Drowsiness System shall be run properly at every time needed.
- High failure intensity is not acceptable.

3.5.2 Availability

- Driver Drowsiness System shall be implemented so that application crashes will be minimum.
- Recovery of the whole system should take minimum time.

3.5.3 Security

- Users also need to activate camera access permission on device.
- User's personal information shall not be accessed or reached by anyone as no data is being saved on the device.

3.5.4 Maintainability

- System shall respond to the real time and physical interactions.
- System shall respond to user commands properly.
- Codes shall be supported by some techniques like related implementation comments, naming conventions and coding standards for increasing readability for future developments

3.5.5 Portability

- The application shall be accessed also by different electronic devices which have a python interpreter and all the required libraries.

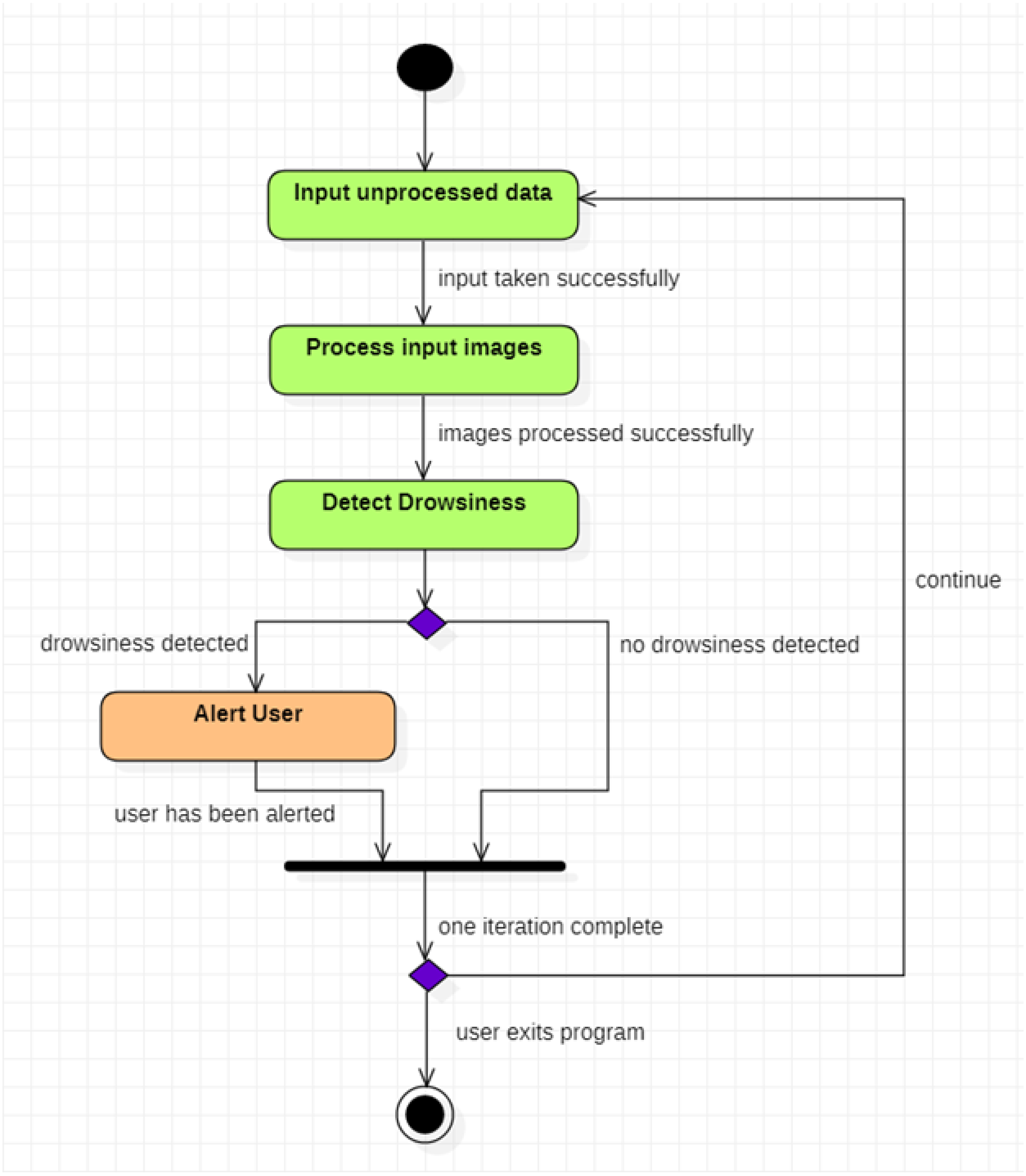
3.6 Other requirements

None

3.7 Organizing the Specific Requirements

3.7.1 Function Hierarchy

State transition diagram:



4. Change Management Process

In case of any changes required in the product, before or after its deployment, can be mailed to us at any of these two EmailIDs and we will get back to you as soon as possible:

- deboparna.banerjee04@nmims.edu.in
- nidhi.uppoor24@nmims.edu.in

5. Document Approvals

None

6. Supporting Information

None