CLEVELAND STATE UNIVERSITY

Software Requirements Specification (SRS) Document

For

"Real-Time Surveillance using Object Detection"

Team – 2

Members:

1. *Vishal Shailesh Mandalia - 2808337*
2. *Sabeel Ashfaq Khan - 2829233*
3. *Shrikant N Satle - 2793600*
4. *Kavya Kolla - 2842895*

Contents

1. Introduction

1.1 Introduction

The purpose of this document is to define and describe the requirements of the project and to spell out the system's functionality and its constraints.

1.2 Scope of this Document

The developers, classmates and the professor are the stake holders of the software developed. This document is to define the scope of the software and its usage guide.

1.3 Overview

The product is build a python based software which shows fatigue in the driver that functions in a way were the opening and closing of eyes or blink rate figure out whether the driver is drowsy. Another factor which is the yawn rate figures out if the individual is drowsy and when these features are captured an alarm is set off to alert the driver to prevent fatal accidents. By applying our computer vision algorithm with the help of OpenCV we can clearly detect whether the person is drowsy or awake. If the driver is fatigued then an alarm is set off to wake the driver up.

2. General Description

2.1 Product Functions

The product should make input of data from the external source via web cam, or it can take the input from the stored data library.

2.2 User Characteristics

The users include the Professor and students at the Cleveland State University as they can provide us with inferential user data. For this system, the user must know the basic usability of accessing webpages as well as a very base level understanding of Artificial Intelligence, which hopefully will be eased by the software team through training.

2.3 User Problem Statement

According to FMCA (Federal Motor Carrier Safety Administration) in the USA it is said that Fatigue is the major factor to cause accidents where fatigue shows about 64% of truck drivers experience some kind of fatigue regularly. Earlier studies show that drowsy driving could be involved in upwards of 40% of truck crashes and about 50% of accidents involving driver fatigue take place between midnight and 8 am. By applying our computer vision algorithm with the help of Open CV we can clearly detect whether the person is drowsy or awake. If the driver is fatigued, then an alarm is set off to wake the driver up.

2.4 User Objectives

The user wants a system that could be used to monitor the moment of driver in case of drowsy and fatigue to prevent accidents.

2.5 General Constraints

Constraints include an easy-to-use interface, an web based platform or, at bare minimum, a Mac/PC based local system.

3. Functional Requirements

1. **Items provided to the system shall be stored in the Database / Library.**

    - Items shall be stored on the laptop machine.
    - Extremely high criticality
    - Limited network / wi-fi availability could present a technical challenge
    - The above stated factor is a risk we have met. Drop it by reducing the dependency of our program on these things.
    - This requirement is the basis of the project; all other aspects depend on it.

2. **The items shall be accessible via web view.**

    - Users of the software should be able to run the application on the web browsers.
    - Extremely high criticality
    - We do not foresee any technical issues preventing the implementation of this.
    - Given the capabilities of Flask, this requirement can be satisfied.
    - This requirement depends on requirement number one.

3. **The data stored should be accessible.**

    - Items and other data should be able to be accessed.
    - Extremely high criticality
    - We do not foresee any technical risks involved in this requirement.
    - The only factor we can meet here is the user of the system not being able to use it correctly. We will overcome this by training those who will be using it.
    - This requirement is dependent on requirement one.

4. Interface Requirements

4.1 User Interfaces

- **4.1.1 GUI**
  The user interface for this program is the screen on the PC where user can view the output and program processing. There will also be a webpage with basic HTML, CSS and JavaScript that uses Flask to render the python code into the website.
- **4.1.2 CLI**
  There is Anaconda command line interface
- **4.1.3 API**
  There is no API for the product

- **4.1.4 Diagnostics or ROM**
  There is a troubleshooting and help section provided by Anaconda CLI
- **4.1.5 libraries**
  There are many libraries used such as cv2, MediaPipe, numpy, matplotlib, time, pygame

4.2 Hardware Interfaces

The program uses webcam as input for live monitoring.

4.3 Software Interfaces

The system may be used to import image data. This functionality is built into the user interface.

5. Performance Requirements

The software is designed to be run on Flask to render the python code into the website. There are almost 5 main libraries that are needed to run the software algorithms. There are no added system requirements beyond those needed to run software, except for a negligible amount of hard drive space to store the image and video streaming data.

Basic host system requirements are as follows:

- 1.5 GHz processor or higher
- 4GB RAM or higher
- 100GB Available Hard Drive Space
- Windows 10 or later operating system.
- Webcam (inbuild or external)

6. Other non-functional attributes

6.1 Security

The system shall be designed with a level of security proper for the sensitivity of information enclosed in the storage base. More interaction is needed with users about the volatility of the information. Since there is no obvious information that is of a high security level as of now at this stage, the only requirements that could be implemented in future are encrypting the confidential image or video storage of users by asking their consent.

6.2 Binary Compatibility

This system will be compatible with any computer that has latest operating system and python 3 or later installed (whether PC or Mac) and will be designed with more than one computer in mind.

6.3 Reliability

Reliability is one of the key attributes of the system. It can be stored and reinstalled so that restoration with minimal data loss is possible in unforeseen events. The system will also be thoroughly tested by all team members to ensure reliability.

6.4 Maintainability

The system shall be supported by teammates and potential stakeholders.

6.5 Portability

The system shall be designed in a way that shall allow it to be run on multiple computers with Python 3 or later installed.

6.6 Extensibility

The system shall be designed and documented in such a way that anybody with an understanding of using Python, libraries used, and PC as well as basic knowledge of Computer vision shall be able to extend the system to fit their needs with the team's basic instructions.

6.7 Reusability

The system should be designed in a way that allows the source code to be re-used regularly for the various silent auctions that the organization shall hold.

6.8 Resource Utilization

The resources used in this system include open-source python libraries and computer vision algorithms and formulas.

6.9 Serviceability

The maintenance of the system should be able to be sufficiently performed by any person with a basic understanding of system usage.

7. Use Cases

- Users can view and interact with the Real-Time Surveillance using Object Detection via webpage hosted on Flask web framework.
- For the user's drowsiness detection, we declare two threshold values and a counter in the algorithm:
    - EAR_thresh: A threshold value to check whether the current EAR value is within range.
    - D_TIME: A counter variable to track the amount of time passed with current EAR < EAR_THRESH.
    - WAIT_TIME: To determine whether the amount of time passed with EAR < EAR_THRESH exceeds the permissible limit.
    - The application starts, we record the current time (in seconds) in a variable t1 and read the incoming frame.
    - We preprocess and pass the frame through Mediapipe's Face Mesh solution pipeline which outputs 468 3-D landmark points.

- Our second use case for facial landmarking, We retrieve the relevant (Pi) eye landmarks if any landmark detections are available. Otherwise, reset t1 and D_TIME (D_TIME is also reset over here to make the algorithm consistent).
- Moving to our third use case in retrieving EAR from both eyes of user's:
    - If detections are available, calculate the average EAR value for both eyes using the retrieved eye landmarks.
    - If the current EAR < EAR_THRESH, add the difference between the current time t2 and t1 to D_TIME. Then reset t1 for the next frame as t2.
    - If the D_TIME >= WAIT_TIME, we sound the alarm or move on to the next frame
- Lastly for our fourth use case:
    - The user's will have an interactive experience with GUI that has been created using basic HTML, CSS and JavaScript and we have used Flask to render the python code into the website.
- User's Experience:
    - A user will be given a custom URL on which our application will be running such as example :"`http://127.0.0.1:5000/`", "`127.0.0.1`" is the IP that represents our machine's `localhost` and `:5000` is the port number.
    - Once the user is able to open the URL, the user will come across our GUI, designed for easy viewing with two buttons for Running the code and the other for Exit.
    - After clicking on the "Run code" button, a window will appear which has the webcam enabled. The user's face will not have facial landmarks in real-time to be able to detect drowsiness or not.
    - No two users can view the real-time surveillance at the same time, as this is designed for one individual user (which is the 'driver').
    - It is highly recommended to not use the URL from multiple devices at the same time as it can cause technical issues for the system.
    - Lastly, when the user has finished with the experience, they can click the 'Exit' button to end the session and exit to our webpage for more information on the project.

Scenario A: Live Detection Based Monitoring

The system hardware can be installed in the dashboard of the vehicle using which it will take the input of the video data and then according to the situational analysis it will give output.

Scenario B: Manual Input Based Monitoring

System can be used on the local system or cloud to run the monitoring by taking an image as an input and according to the situational analysis it will give output on screen.