SOFTWARE REQUIREMENTS SPECIFICATION

for

Speed Detection Project

Version 1.0

Prepared by : Group G3

Submitted to : Prof. Saurabh Tiwari
Prof. Manish Khare
Lecturer

April 28, 2023

Contents

1 Introduction

1.1 Purpose

The basic purpose of this document is to provide a detailed specification of the requirements and the functionalities of our speed detection system which we have named "Swift Sense". This system is primarily designed to measure the speed of vehicles in order to promote road safety, reduce accidents, and prevent dangerous driving behavior such as speeding. Apart from above, this system can also be used for sports application, industrial application etc.
For example, this system can be used in sports to measure the speed of athletes, such as sprinters or race car drivers, to analyze their performance.
Furthermore, this system can also be used in industrial environments to monitor the speed of machinery and equipment to ensure they are functioning correctly and safely.

1.2 Intended Audience and Reading Suggestions

The intended audience of "Swift Sense" would primarily be law enforcement agencies, traffic control authorities, and other relevant government organizations responsible for road safety and traffic management. Additionally, the system can be used by sports organizations, research institutes, and industries to measure the speed of athletes, things such as ball, shuttle and speed of machinery and equipment and other objects. Basically, any environment which is highly sensitive on or requires the speed of the various objects involved could be an intended audience.

1.3 Project Scope

"Swift Sense" creates a space for users to upload a video of formats:[mp4, mkv, avi] for speed detection. After user logs in on the website, he/she can upload the video file to detect the speed of the object in it. After the file is uploaded the python speed detector script is run. The current version requires that we upload the file, but by making a small change in code we could also run the speed detection script on a real-time feed from a webcam etc.
In the script, first the dataset loading takes place. These datasets(which are in form of .xml files) are generated from the positive and negative images of the object to be detected.
There are 2 python scripts which search the positive and negative images from search engines like Bing, Google etc, and store them in a dataset folder. Then this folder is

passed to a Cascade Trainer software which generates the required dataset xml files.
In the current version, we already generate several xml files for different objects, but it could also be possible to automate the process of generating the xml file depending on the object the user wants to detect. The reason for above is that the generation of xml files from the images takes a lot of time and processing.
After, dataset loading video capture takes place & the video is analyzed frame by frame. Then the object is detected and tracked over multiple frames, and the speed of the object is calculated after passing through 2 quality checks.
This data is then stored on the server DB. The user can access the speed data like average speed, speed graph, notification for speed violation etc through the website dashboard.

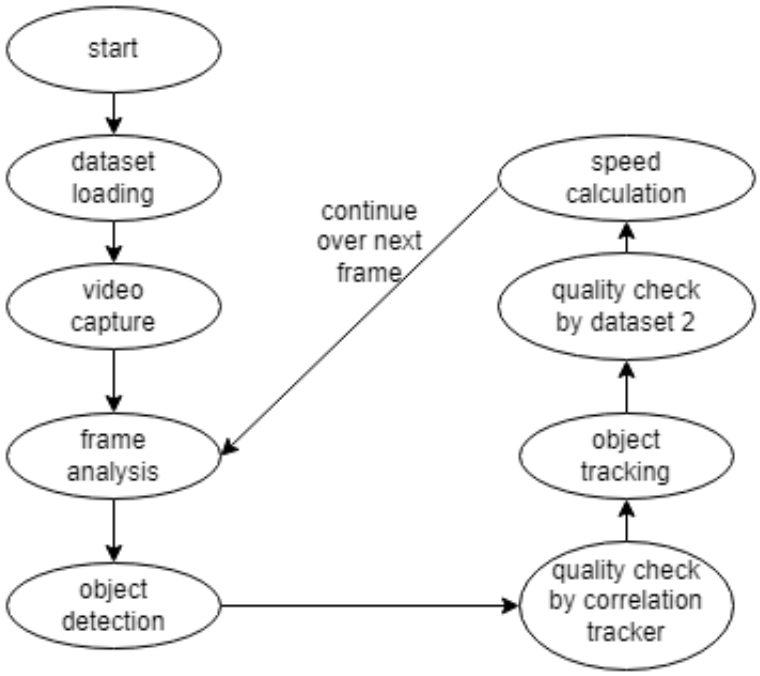

Figure 1.1: Speed detection basic flow diagram

2 Overall Description

2.1 Product Perspective

"Swift Sense" is designed to accurately measure the speed of moving objects in real-time or through a video feed. It employs computer vision and object detection/tracking and speed detection using the Haar Cascade Classifier model.
From the user's perspective, the system provides valuable data for a variety of applications. For example, law enforcement agencies can use it to enforce speed limits and reduce accidents on the roads. Industrial engineers can use it to monitor the speed of the machinery and equipment to ensure they are functioning correctly and safely. Sports coach and analysts can use it to measure the speed of athletes to analyze their performance, and to measure ball speed, shuttle speed etc.
The main goal of this project is to make a standardized system to measure the speed of many different types of objects instead of using different systems to measure speeds of different objects.

2.2 User Classes and Characteristics

The speed detection website can have users such as:

- Users
  - Traffic authorities
  - Sports Analysts, coaches
  - Industrial engineers

For example, traffic authorities could get a notification when the system detects an overspeeding vehicle.
Sports analysts/coaches could get a graph of the speed of the athlete, in addition to the average speed to analyze the athlete's performance.

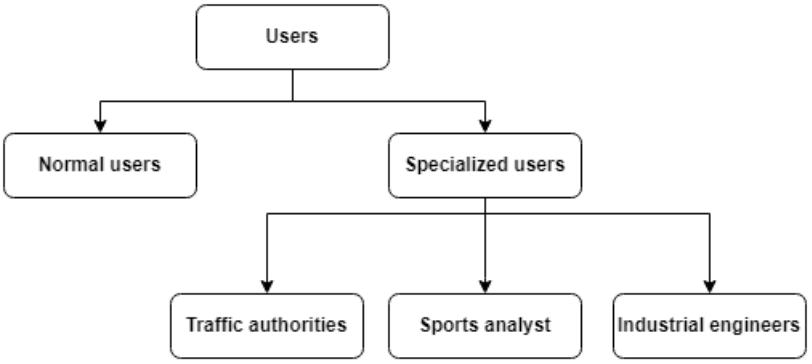

Figure 2.1: type of users

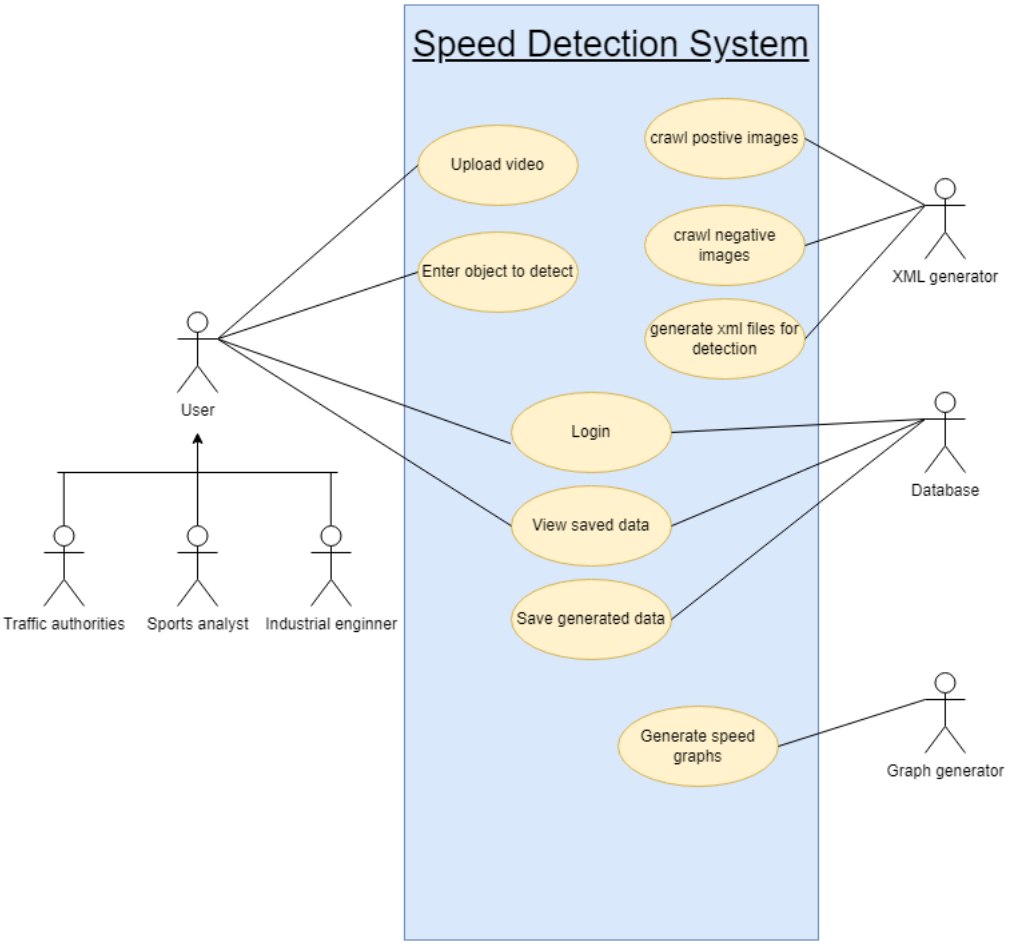

Figure 2.2: Use case diagram

2.3 Use case stories

Actor: User

Pre-condition: The speed detector system(consisting of a camera) is ready to use, and the user has given the access permissions to the system.

Use-case 1: Camera operation Normal Flow:

1. Turn ON the system
2. Point the system towards the object
3. Camera in the detection system detects the object
4. System calculates the object speed
    a) Frame by frame object detection
    b) Image Processing
    c) Speed calculation
5. Display the speed detected
6. Turn OFF the system

Alternate/Exception flows:

1.a If the system does not turn ON, make sure the system is charged and then continue.

3.a If the system does not detect the object, restart the system

5.a If the user is unsure about the value of the speed detected, detect again

Post-conditions: The speed detection values are also added to the database of the user.

Pre-condition: The user has the website URL and internet access.

Use-Case 2: Login functionality and access previous speed data

1. User opens the system website
2. User logins to the website if already a registered user
    a) Enter username and password
    b) Click on Login
3. User signups to the website if not a registered user
    a) Enter required signup information
    b) Click on SignUp

4. User enters the website
5. Click on the dashboard and navigate to speed database
6. User access the required data

Post-conditions: After the user gets the required data, he/she closes the website

2.4 Product Functions

"Swift Sense" stores all the speed related data of the users. The user first logs in on the "Swift Sense" website. If the user is not registered he/she can sign up. After log in, the user can upload a video file for speed detection(The current version does not have the function for live video feed from the user, because it would require us to take permissions from user to access their camera, and live camera feed is usually of poor quality which leads to poor speed detection.)
While uploading user also has to enter the object to detect the speed of. If the xml file for that object already exists then the xml file is directly used for detection, but if the xml file for that object does not exist then the system generates positive and negative images from the prompts that user gives and passes them to a Cascade Trainer software, which then generates the required xml; and then the speed detection start
The user can also access the dashboard to get a history of their speed calculations.

2.5 Operating Environment

The website can operate in any Operating Environment - Mac, Windows, Linux etc.
The website will require a server environment to host the web application, store data, and manage user accounts. The server should have sufficient processing power and memory to handle the expected traffic and user load.
It also requires a database like MongoDB to store unstructured data like the speed graphs of the object, and to store the username and password for login.
It also requires that all the sensitive url like the mongoDB connection strings for user authentication and data be stored as secret keys or environment variables.

2.6 Design

"SwiftSense" website has the following components:

- Login/Signup page
- Home/Dashboard page
- Upload page
- History page to show the speed data of user

- About us page

User can do the following activities in order to get their speed data:

- Login/Signup
- Go to the upload page
- Upload the video, and give object input
- Go to history page to view the speed related data.

3 Domain Analysis Model

A domain analysis model for a speed detection system is representation of the system's domain, inputs, outputs, and its components. It defines the key concepts, relationships and behavior within the domain and it helps to identify the requirements and constraints that the system must satisfy.
Below is our system's current domain analysis model

1. **Inputs:** Currently, the system requires the following inputs:
   - A video file from the local system or a video feed from the webcam
   - Name of the object to be detected
   - Positive and negative object detection prompts
2. **Processing:** Currently, the system processes the video file/feed using the following steps
   - Detect the object in each frame of the video
   - Tracks the position of the object across frames
   - The above 2 points are done by using various xml files which are obtained from training the object detector model with images of objects like car, human etc
   - Calculates the speed of the object on the basis of the distance travelled over a frame period
3. **Outputs:** Currently, the system outputs the following
   - A csv file containing the speed over different frames of the video of multiple objects is generated, and then speed graphs of the objects detected is generated.
4. **Constraints and Assumptions:** Currently, the system operates under the following constraints and assumptions
   - The camera is stationary and its position and orientation do not change during the video feed
   - The object is clearly visible in the video feed and is not obstructed by any other objects
   - The frame rate and resolution of the camera are sufficient to capture the movement of the object accurately

5. **Actors:** Currently, the system has following actors
    - Traffic control authorities
    - Camera to capture the video feed
    - Speed analyst
    - Industrial engineers

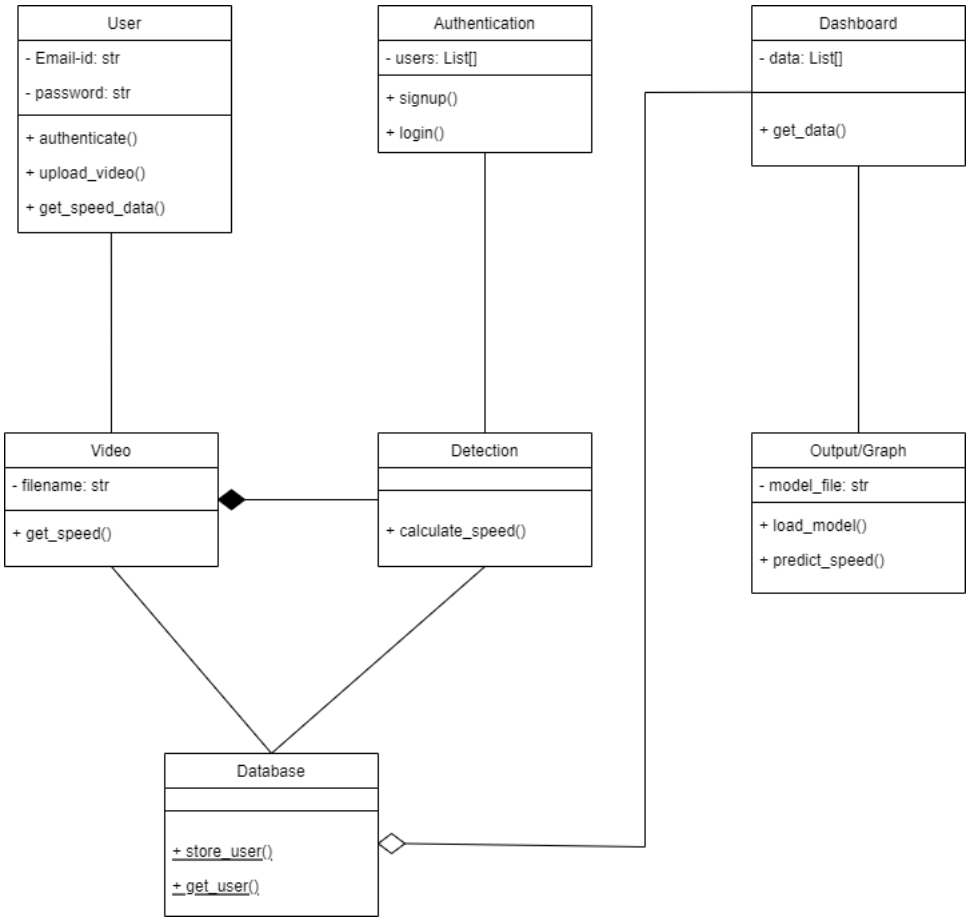

Figure 3.1: Class diagram

3.1 Boundary, Entity and Control Object:

Boundary: VideoInputBoundary (interfaces with external video source), SpeedOutputBoundary (interfaces with external data storage or display)

Entity: Frame (represents a single image captured from the video), Speed (represents the calculated speed of a moving object)

Control Object: FrameExtractor (controls the process of separating frames from the video input), SpeedCalculator (controls the process of applying ML algorithms to estimate speed from frames)

3.2 High level system design

1. **Input:** The system will need a video feed, name of the object to be detected and the positive/negative object prompt from the user to detect the speed.

2. **Data acquisition:** In this module the acquired video feed is converted into frames so that it is easy to process.

3. **Data processing:** This module will take raw data we got from the data acquisition module and convert it into meaningful information. In our system, after detecting the object this module tracks the object over various frames, and sends the frame data to the speed detection module.

4. **Speed detection:** This module analyzes the frame data sent by the data processing module, and calculates the speed of the object by finding how much distance the object moves over different frames.

5. **Output display:** This module displays the detected speed of the objects on the video feed. The speed output is also given in the form csv file, which is then used to calculate the average speed and the speed graphs of the detected objects, and this is viewed by the user.

6. **Data storage:** The system also stores the above data about the detected speed of the object for future reference or analysis.

3.3 Architecture:

1. **Physical Layer:** This layer includes the hardware components of the system such as the camera or the upload box to upload the video feed.The physical layer is responsible for collecting data from the environment and processing it into a format that can be understood by the upper layers.

2. **Data processing:** This layer includes the data processing unit. It is responsible for processing the data received from the physical layer. In our system, this layer receives the video feed from the physical layer and converts it into frames, detects the object(using the pre-generated/generated xml files), tracks the object over various frames of the feed, calculates its speed, stores the data into a csv format and uses this data to give average speed and speed graphs of the detected object.

3. **Application Layer:** This layer includes the software that runs on top of the data processing layer. The application layer is responsible for providing the user the

interface for the system. In our system, the data obtained from data processing layer is passed to the speed database which can be accessed through an web application(application layer). This web app has features like calculating average speed, graph of the speed of the object etc

3.4 Subsystems

Subsystem 1: Object detection
This subsystem will be responsible for detecting objects like vehicles, humans etc in the video feed. The detection subsystem will need t o be fast and accurate to ensure all the objects are correctly identified.

Subsystem 2: Speed calculation
Once an object is detected, this will be responsible for measuring its speed. This is done by capturing the video feed, converting the feed into frames, tracking the identified object over different and finally calculating the speed by dividing the distance travelled by the object by the frame time. The speed calculation subsystem will need to be accurate and precise, with low error rates.

Subsystem 3: Data analysis, storage and reporting
The final subsystem will be responsible for analyzing the speed data(csv data) collected by the system, and generating meaningful output (eg average speed of the object, graph of the speed of the object). This subsystem will need to be reliable(and fast and low false positive rates in case of speed limit tracking of the vehicles)

4 System Features

4.1 Description and Priority

"Swift Sense" is a speed detecting web application. So the main feature of this product is to detect the object,track its speed and generate data regarding the speed.
The main priority of the web application is to detect the speed of various different type of objects accurately.
The features with priority up to down -

1. **Detect different objects:** The goal of the web application is this feature itself. This feature allows to detect different objects on the basis of what the user wants to measure the speed of.
2. **Accurate detection:** It is very important that the speed detected is accurate enough to be close to the actual speed value even if harsh conditions.
3. **Multi detection:** This feature allows to detect multiple objects of same type at the same time.
4. **Quality check:** This feature first checks the quality of the detection, and only if the it is high enough is the object detected.
5. **Data generation and storage:** The speed data like speeds graphs, average speed etc are generated and stored so that the user can access it later.
6. **Real-time detection:** This is an important feature, but since it is hard to achieve it is kept lowest in the priority. This feature is difficult to achieve because to capture real-time video feed we need to get user permission, also the real-time detection could be worse depending on the camera quality that user uses.

4.2 Functional Requirements

The "Swift Sense" website is build on the following:

- Front-End - React, Javascript, CSS, HTML5, Flask
- Back-End - Python, NodeJS
- Database - MongoDB

Following are the functional requirements of a speed detection system:

1. **User login and Authentication:** The website should provide user login and authentication to ensure that only authorized users(eg traffic control authorities) can access the system.

2. **Speed calculation:** The system should yield the speed of different objects on the basis of distance travelled over a frame time.

3. **Image processing:** The system backend should have the logic to process the frames of the video feed uploaded(or live video feed)

4. **Multi-detection:** The system should be able to detect and handle multiple (same type) objects at the same time in a video feed and give their individual speeds.

5. **Real-time detection:** The system should detect and track objects in real-time.

6. **Data generation:** The system should process the video, generate the required speed data like object speed csv file.

7. **Custom reporting and storage:** The system should process the data generated to give meaningful information like average speed, speed graphs etc and store it in a database so that the user can access and analyze it on the basis of various parameters.

4.3 USP of the product

There are many speed detection system on internet but they are for detecting a particular type of object only. "SwiftSense" web application is able to detect the speed of many different type of objects.
If the user wants to detect the speed of an object whose xml file is already generated, then the detection starts right away. But if the user wants to detect an object whose xml file is not pre-generated, then the user can give positive and negative object prompts, which will then be used to crawl the positive and negative images from the web browsers like Bing, Google etc, and then these images will be passed to a Cascade Trainer which will generate a new file xml file for the object the user wants to detect.
In this way, the "SwiftSense" web application can even detect objects for which it does not have the pre-generated xml, thus allowing the user to detect the speed of a wide range of objects.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

"Swift Sense" website could be used by the traffic control authorities or industrial engineers, so it needs to be **accurate and reliable**(ie low rates of erroneous data and malfunctions) even under challenging conditions.
If the detection system is inaccurate, it may not accurately detect when a vehicle is exceeding the speed limit or driving dangerously. Also, if the system is inaccurate it may use more resources and thus affect the performance of the system.

It should also be able to analyze large volume of video feed/speed data **efficiently**, and give the output report with **low latency**.
If the processing time is high, the system may miss the detection or provide incorrect information which can be dangerous in situations where speed limits are enforced, for eg highways or in high traffic areas.
Also, the database must have the up-to-date information at all times.

5.2 Security Requirements

Only the authorized users(eg traffic control authorities) should be able to access the system.
The user login details should be encrypted and then stored in the database.
Also, the database connections strings should not be directly used but rather should be stored as secret keys to prevent unauthorized access or tampering.

5.3 Other Requirements

"Swift Sense" system should also be user-friendly. If the user wants to process a live video feed, the quality of the video feed should be high to decrease detection and speed calculation errors even in unfavorable conditions like bad weather, heavy traffic, very fast moving objects etc.

To ensure the system can handle the increased volume of speed data it receives, processes it quickly and delivers results in a timely manner, it must be scalable. If the system is not scalable it may become unresponsive which would severely impact its ability to accurately detect and report speed violations, and thus affect the road safety.

5.4 Business Rules

Some possible business rules are as follows:

1. **Data Ownership:** The speed data captured by the system should be owned by the organization responsible for the system and not shared with third parties without proper consent or legal authorization.

2. **Accuracy Requirements:** The system should be designed to capture speed data accurately and reliably, with a predetermined level of accuracy specified in the system's requirements.

3. **System Availability:** The system should be designed to ensure maximum uptime and availability.

4. **Speed Limit Thresholds:** The system should be configured with predetermined speed limit thresholds, and should generate alerts or notifications when vehicles exceed these limits.