Video Surveillance System

Software Requirements Specification

Version 1.0.0
Status: draft

Prepared by Jainam Shah(2017csb1107)

Dilip Sharma(2017csb1073)

Revision History

Name	Date	Reason For Changes	Version	Date of Approval
	29 Sep 2020	Initial Edit	1.0.0	--

Introduction

Purpose

The purpose of this document is to create a standalone desktop application for video analysis of security cameras with automated scene/object detection from video files. This will ease the process of reviewing the video for wrong activities manually by reducing inactive video frames. This document will explain the purpose and features of the software, the interface of the software, what the software will do, the constraints under which it must operate. It also identifies functional and non-functional requirements with a use case diagram. This document is intended for both the end-users and the developers of software.

Intended Audience and Reading Suggestions

The document is expected to be read by the end-users and the developers of the application. The document starts by briefing the purpose of the product being developed. Thereafter it goes on to describe the Functionalities and the Operating Environment. The developers are advised to read the whole document.

Product Scope

The Video Surveillance System will analyze the video footprint of security cameras. This product is based on computer vision which is a field of Artificial Intelligence. Using various deep learning models, this application will detect and identify objects given in a video frame. The product will be available for all major OS(Windows, macOS, Ubuntu). Above all, we hope to provide a comfortable user experience with the interface.

References

1. https://github.com/Breakthrough/PySceneDetect
2. https://medium.com/@abulka/electron-python-4e8c807bfa5e
3. https://github.com/argman/EAST

Overall Description

Product Perspective

Common techniques used for the above scenarios are tedious and consume a lot of time as well as manpower. Further, there are many software available and are also being developed but they are designed only for specific purposes. We plan to deliver software that can be adapted to various scenarios. This is a new self-contained product that provides the user an option to design complex workflows on the detection of desired object/movement.

Product Functions

We plan to deliver a thick client-based software. This product aims to provide the functionality to the user for extracting relevant information from a given photo or video which he/she can then use to proceed further with his/her defined workflow. The software will take input an image or a video URL and then give output in a specific format which will contain the relevant information. This information will include the image id and other details about the persons/objects identified. After receiving this information, the user can proceed further with whatever workflow he has defined.

User Classes and Characteristics

ID	User classes	Description
U-1	Registered user	A user can directly access the software to perform the functions offered.
U-2	Trial user	A user who is given the service free of cost. He has access to limited features.

Operating Environment

- **Client**
    - Operating System - Windows/Ubuntu/macOS

Design and Implementation Constraints

Languages - Python3, JavaScript
Libraries/Frameworks - Tensorflow(with Keras), NumPy, OpenCV, Electron, Django
Software Technologies - SQLite, Postgres
Data Exchange Format - JSON(tentative)
Hardware Limitations - Good RAM, Recommended good GPU(optional)
Communication Protocols - HTTP, zeroRPC
Security considerations - We will be using Gmail authentication.

User Documentation

- The clients will be presented with an Intro at the first time application Startup.
- There will be a webpage describing how to use the app in detail.
- There will be a help form, where the user can interact with the developers for technical help.

Assumptions and Dependencies

Assumptions

- We assume that all the input will be in the same specific format
- We suppose that the workflow should be fast enough to handle the continuous stream of frames in real-time.

- The user has available resources to run our product.
- If the algorithm terminates due to some unexpected reason then the data must not be lost. At Least one backup of data must be present.

Dependencies

- This is an independent project, we are introducing it for the first time. There is no dependency on the version of the software.

External Interface Requirements

User Interfaces

Before installing the software, the user would have to agree with our legal requirements. The user can refuse to enter into the agreement.

Authentication

After installation, a pop-up will appear.

- Registered User: Buy the software from the website. Users will receive a unique premium one-time code in the mail.
- Free-Trial User: Use free features for 7 days.

Additional files will be downloaded for the registered user after he has entered correct code.

Video Mining

This interface will allow users to add input video and specify parameters of the scene that he would like to detect in the video.

- Input: Video, Scene parameters (text, objects, face, etc.)
- Output: Detected scenes from ML-based model
    - Allow download containing output information

Hardware Interfaces

The software doesn't require any special hardware. The thick client supports hardware that can run Windows/macOS/Ubuntu and has hardware fulfilling the recommended requirements are good RAM, recommended good GPU(optional).

Software Interfaces

The software is written in *Python3* and *Node.js*. It uses *Electron (a Node.js framework)* for creating GUI and python for video processing. The python code is compiled using *Pyinstaller* and will communicate with GUI through RPC. For scene detection, a library based on OpenCV will be used. The web framework will be created using Django and its extensions. The thick client contacts its server for user authentication for purchase verification. The communication

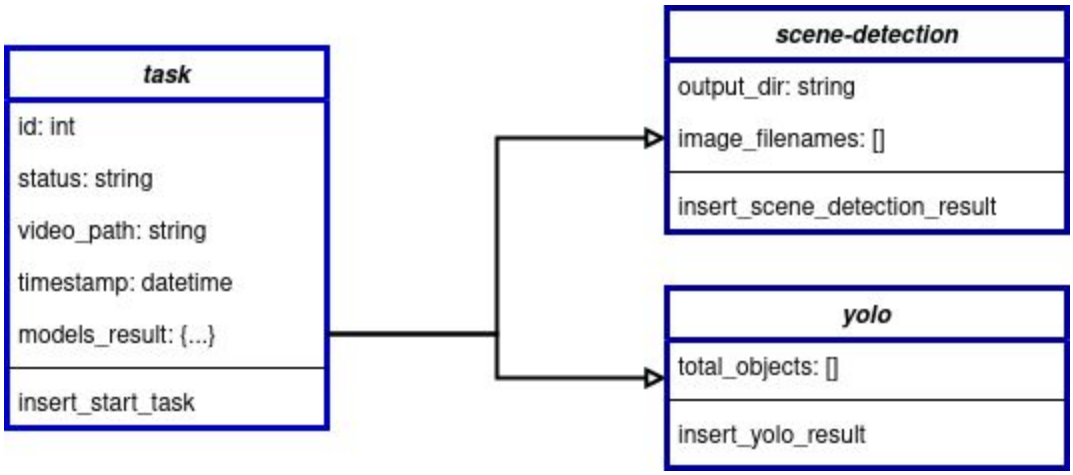
with the server is done using the HTTP protocol. Rest-Auth is used for the authentication of the user and is also used for API authentication.

Communications Interfaces

Web Server acts as our communication interface. Clients need to register and buy the software from our web portal inorder to use the thick client application. We plan to deploy our web server on Google Cloud Run.
Web application provides following functionalities:

1. User Registration
2. Software Purchase
3. Reset Password
4. Documentation and Tutorial
5. Support

Domain Model



System Features (Use Cases)

Upload Video

Brief Description:	The user uploads video for a model prediction.

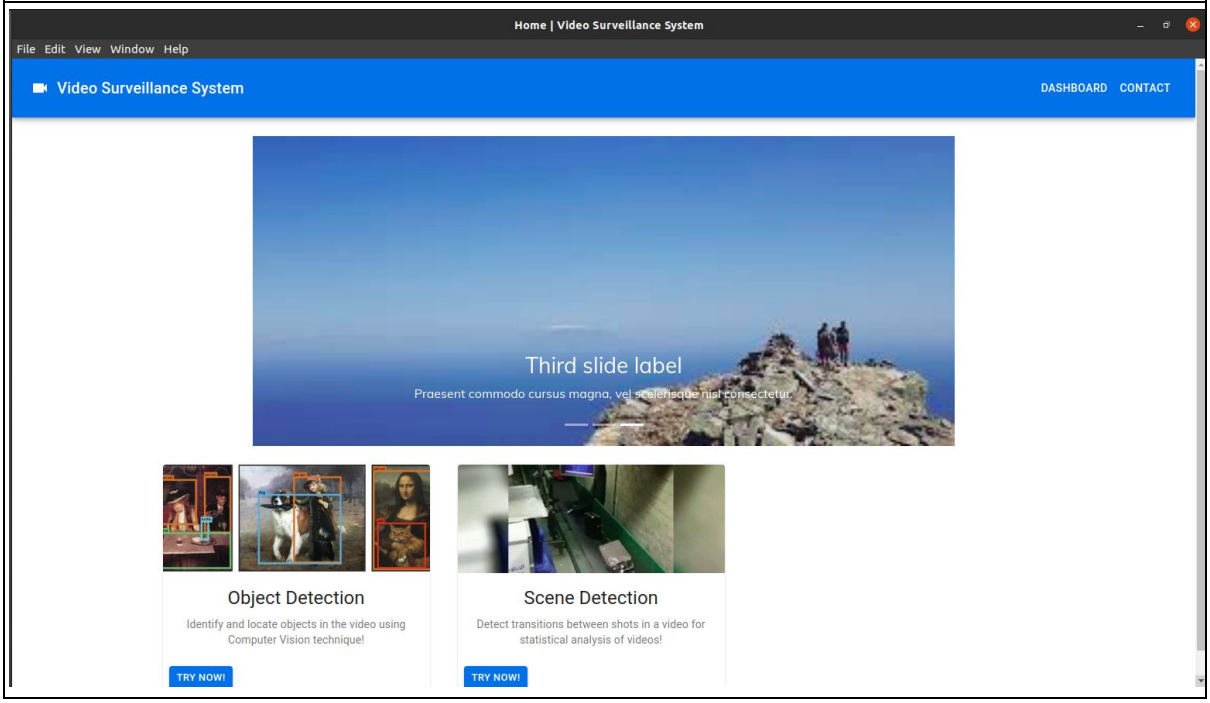
Preconditions:	The software is installed in the user's system.

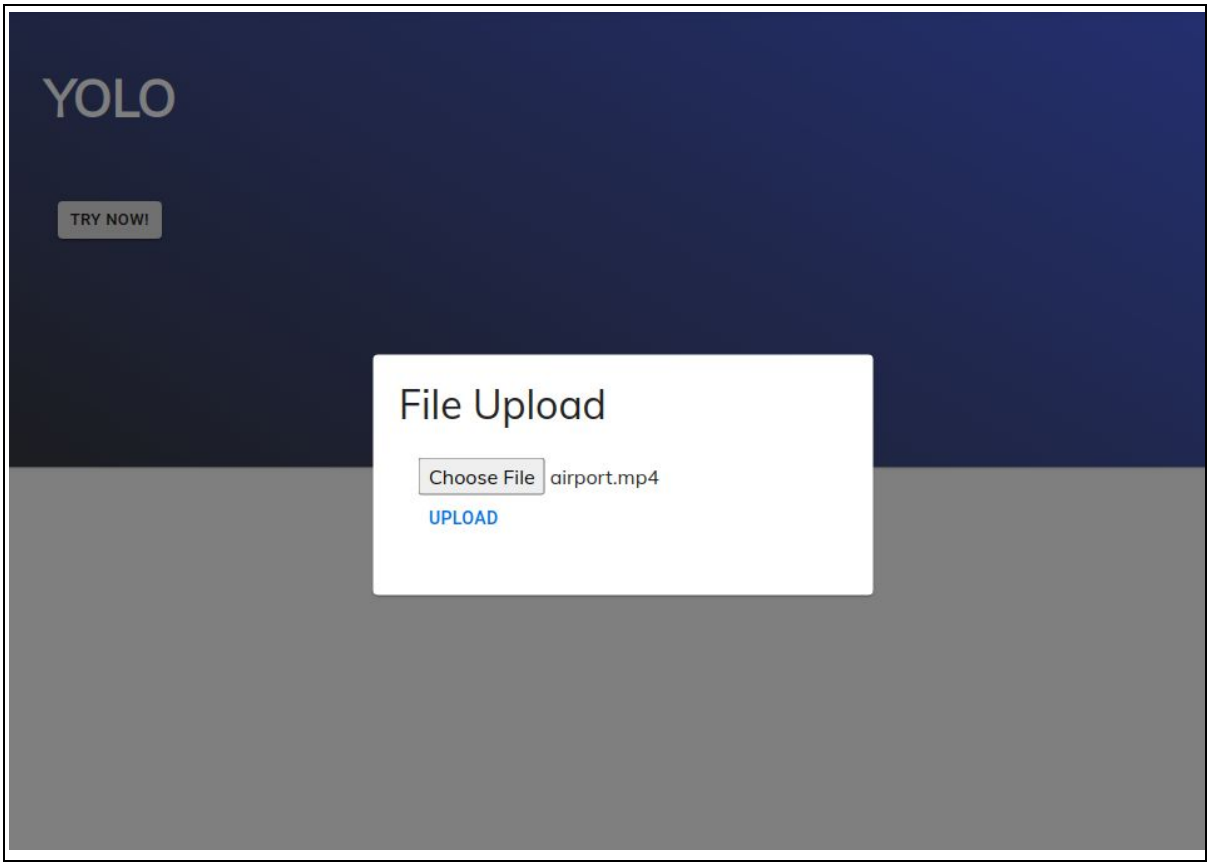
Basic Flow: The user opens the VSS software in his system.		
Assumptions: The user should choose a valid video.		
Line	**System Actor Action**	**System Response**

1	The user opens the VSS software.	It presents the main screen of VSS.
2	The user chooses the desired model to run.	The software opens the model page.
3	User clicks on the `Try Now` Button.	System will pop for file upload.
4	User clicks on browse for selecting video.	The system opens the file browser.
5	The user selects the video.	Captures the video path.
6	Clicks on Upload Button.	Send the video for the selected model worker and start processing. And shows the user to wait for the processing.
Post Condition:	Now the user can see progress of the task in Dashboard.	

Alternate Flow (AF1): Invalid Video		
If the video is not valid or not selected.		
Line	**System Actor Action**	**System Response**
1	The user taps on Browse Button.	The system returns the user in basic flow.
Post Condition:	Now the user can see progress of the task in Dashboard.	

Prototype Screen:





View Dashboard

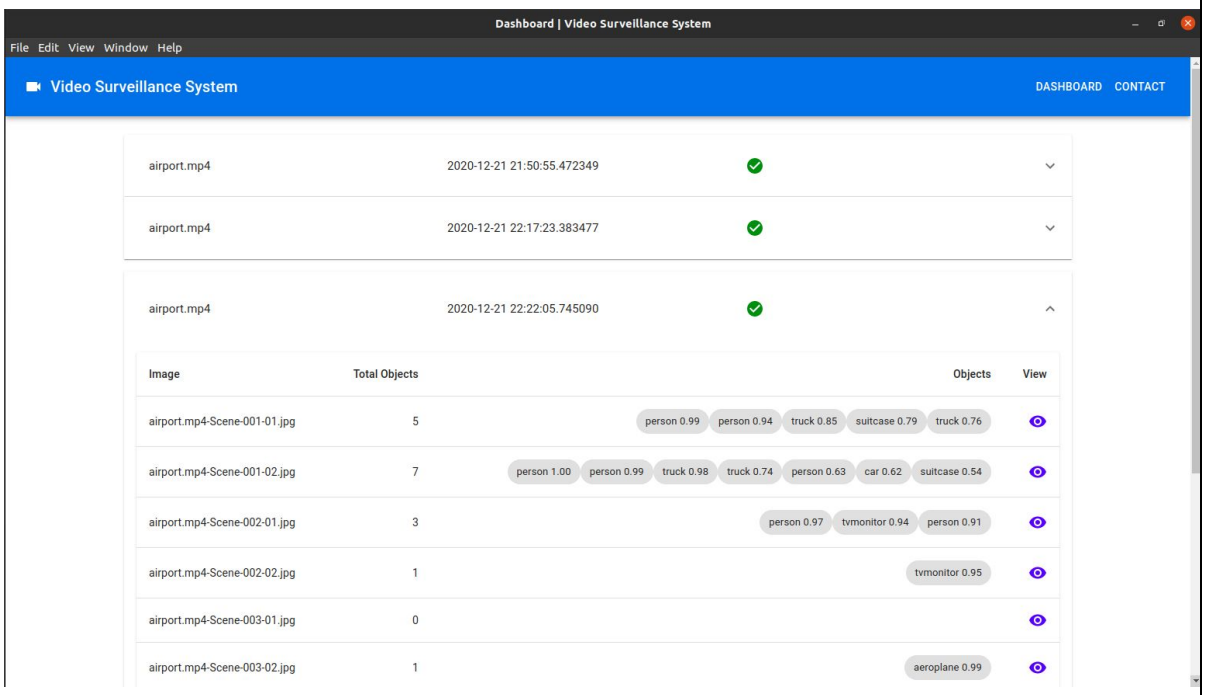
Brief Description:	View the results of videos uploaded and do the analysis.
Preconditions:	The users should have uploaded any videos to see the result.

Basic Flow: The user can view dashboard by using the *Dashboard* button on the NavBar		
Assumptions: There are results available		
Line	**System Actor Action**	**System Response**
1	The user opens VSS software.	The system shows the home page of software with NavBar at top.
2	The user clicks on the Dashboard button.	The software shows the list of all the videos uploaded by the user and their status (processing or successful)
3	The user click on any of the processed result	The list of scenes detected and objects detected are shown
4	The user clicks on any of the Image icon	The image corresponding to that result is shown
Post Condition:	Now the user can view all the results of the uploaded video.	

Alternate Flow (AF1): Video still processing		
If the video uploaded is not processed yet		
Line	**System Actor Action**	**System Response**
3	The user clicks on the video with processing state	The system shows the processing message.
Post Condition:	The outputs are shown once video is available.	

Prototype Screen:

Dashboard View



Viewing the image