BeamEye

AI-Powered Pedestrians Detector

Graduation Project Report

CMSE 406

Team members:

Khawlah Alshubati 19701557

Omar El Khadiji 19700103

Supervisor:

ASSIST.PROF.DR. ADNAN ACAN

Computer Engineering Department

Eastern Mediterranean University

Fall 2023-2024

ABSTRACT

This report plays a key role in providing detailed information about the development phases of our project "AI-Powered Pedestrians Detector" called "BeamEye" where we integrated the SSD MobileNet detection model provided by TensorFlow to a windows application using Tkinter Framework. Project management tools were frequently used to keep up with the project implementation starting from planning up to testing and maintaining stages. The main aim of our project is to integrate AI (Artificial Intelligence) detection technology into a windows desktop app to make it easier for users to track pedestrians and thus provide more safety and security features to their surveillance system. This report demonstrates the details of the system's functional/non-functional requirements, planning activities, detailed description of code and algorithms used and quality matrices which were followed to ensure successful completion of the system. As for the software development life cycle, we followed the agile methodology. While writing this report and codes for the project we followed the IEEE standards as shown throughout the project code and report.

Keywords: Object Detection, ML, TensorFlow, Computer Vision, OpenCV, Tkinter, Pedestrians Safety

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

As cities grow, the safety of pedestrians becomes increasingly challenging. Our proposed solution plays a crucial role in monitoring pedestrian flow. BeamEye can serve the traffic authorities to better understand pedestrians' behavior and enhance security purposes. We have successfully integrated an object detection pretrained model provided by TensorFlow, SSD MobileNet, into a user-friendly Tkinter interface which not only makes this solution smart but also easy to use.

The main reason we created this app is to help in the security of pedestrians and enhance the traffic management by providing an accessible user-friendly windows app which users can easily use to detect and track pedestrians. In addition to security personnel, this system can be further improved to help urban city planners utilize the system for better understanding, managing, and tracking pedestrians.

While various pedestrian detection systems have been developed, this project's unique contribution lies in the combination of AI model with a user-friendly interface where user can also use prerecorded videos of pedestrians to be detected. Previous attempts in this domain have often focused on either improving the accuracy of detection algorithms or developing applications for specific platforms or focusing on detecting people live.

This report explains the planning, design, development, and testing processes of BeamEye which serves not just as a documentation of our effort, but as a blueprint for future endeavors in realm of AI and public safety.

2. PROJECT PLANNING AND MANAGEMENT

2.1 Preliminary Project Information

Project No	**01**
Project Name	**BeamEye, AI-Powered Pedestrians Detector**
Start Date	15.03.2023
End Date	30.12.2023
Time	8 Months - 270 Days

Project Manager			
Name Surname	**Khawlah Alshubati**	**ID No**	**19701557**
Title/Role	Software Engineer - Project Manager		
Address	TRNC Famagusta		
Phone	+905338341999		
Email	19701557@emu.edu.tr		

Team Member			
Name Surname	**Omar El Khadiji**	**ID No**	**19700103**
Title/Role	Computer Engineer – System Analyst		
Address	TRNC Famagusta		
Phone	+212613860047		
Email	19700103@emu.edu.tr		

Table 1: Projects of The Team

List of Completed / Ongoing Projects of Team
- BeamEye, AI-Powered Pedestrians Detector. (CMSE 406) - Web Scrapping Tool to Scrap Physicians Data. (CMSE 322) - Student Portal Website. (CMPE 312) - E-Commerce website. (CMPE 344) - Pharmacy Management System. (CMSE 321) - Rating Websites Using Data Mining. (CMSE 201)

2.2 Aim of the Project

In our BeamEye Project we aim to create a windows-based AI-powered pedestrian detection system that can recognize and categorize pedestrians in real-time using previously recorded videos without the use of physical cameras. The system will examine the uploaded videos, identify and track pedestrians' behavior using deep learning algorithms. The ultimate objective is to offer an adaptable, user-friendly, and scalable solution to enhance public safety and security in a variety of settings, including public parks, parking lots, commercial areas such as shopping centers and malls, pedestrians' walkways, and university campuses.

2.3 Reasons of Starting the Project

We - as a team - are fascinated by object detection technology and desire to expand our knowledge in the fields of artificial intelligence (AI), machine learning (ML), and deep learning by developing a real-world project. We aim to gain expertise in these areas by working on a project that uses these technologies, particularly in the detection and tracking of pedestrians from video footage. Most importantly, the potential real-world applications of this project are numerous, including improving self-driving automobiles, enhancing surveillance systems, and enhancing public safety through the detection and tracking of pedestrian's behavior

2.4 Output & Success Criteria

The main **purpose** of this project is to create an AI-based pedestrian detection system named BeamEye that can be integrated into a desktop application with the goal of enhancing safety and security in public areas. The system utilizes artificial intelligence algorithms to analyze prerecorded videos of public spaces to identify pedestrians. The project is mostly useful in public safety and surveillance systems, researching and city planning.

The **output** of the project will be a functional system that can detect pedestrians from a prerecorded video of a surveillance camera using AI technologies. The system should be able to analyze the video's data, identify the presence of pedestrians, and track their movements.

The project's specific objectives are to develop and implement an accurate and reliable pedestrian detection system, to create a user-friendly interface for system users, and to improve safety and security in public areas.

The **success criteria** of this project will be measured by:

- The accuracy and reliability of the resulting system.
- The user-friendly interfaces and effectiveness of the system.

- Completing the project within the given time.
- The system should be able to detect pedestrians with an accuracy of at least 80%.
- The system should be able to track pedestrians through occlusions, such as when they are temporarily blocked by other objects.
- The system should have a fast-processing speed, capable of analyzing video data in real time.
- The system should be reliable and able to operate under various lighting and weather conditions.

2.5 Comparison To other Projects

Compared to other similar projects, the AI-powered pedestrian detection system has several differences, advantages, and areas of superiority.

In terms of **differences**, our AI-powered pedestrian detection system uses state-of-the-art deep learning algorithms and computer vision techniques to accurately detect and track pedestrians from prerecorded videos. This sets it apart from other projects that may use more traditional machine learning algorithms or rely on human operators for pedestrian detection. Secondly, this system can be used by researchers of pedestrians' behavior and modern city planners, unlike other projects that only aim to detect pedestrians live from surveillance cameras.

In terms of **advantages**, the AI-powered pedestrian detection system offers improved accuracy and reliability compared to other projects. The use of deep learning algorithms and computer vision techniques allows the system to quickly and accurately detect and track pedestrians from input videos that come from various resources. Additionally, the system can operate 24/7, making it an ideal solution for surveillance and public safety applications where constant monitoring is required.

In terms of **superiority**, the AI-powered pedestrian detection system has several advantages over other projects. Firstly, the system can handle enormous amounts of data and can process video footage from various resource videos, which is not always possible with other projects. Secondly, the system can be trained to recognize and track people until they are out of the frame of video footage. Thirdly and most importantly, the system is built as a desktop app where we have not seen similar projects implementing it this way.

Overall, our AI-powered pedestrian detection system offers an important level of accuracy, reliability, and flexibility compared to other similar projects. It will be useful for various real-life scenarios that require pedestrian detection.

2.6 Gantt Chart

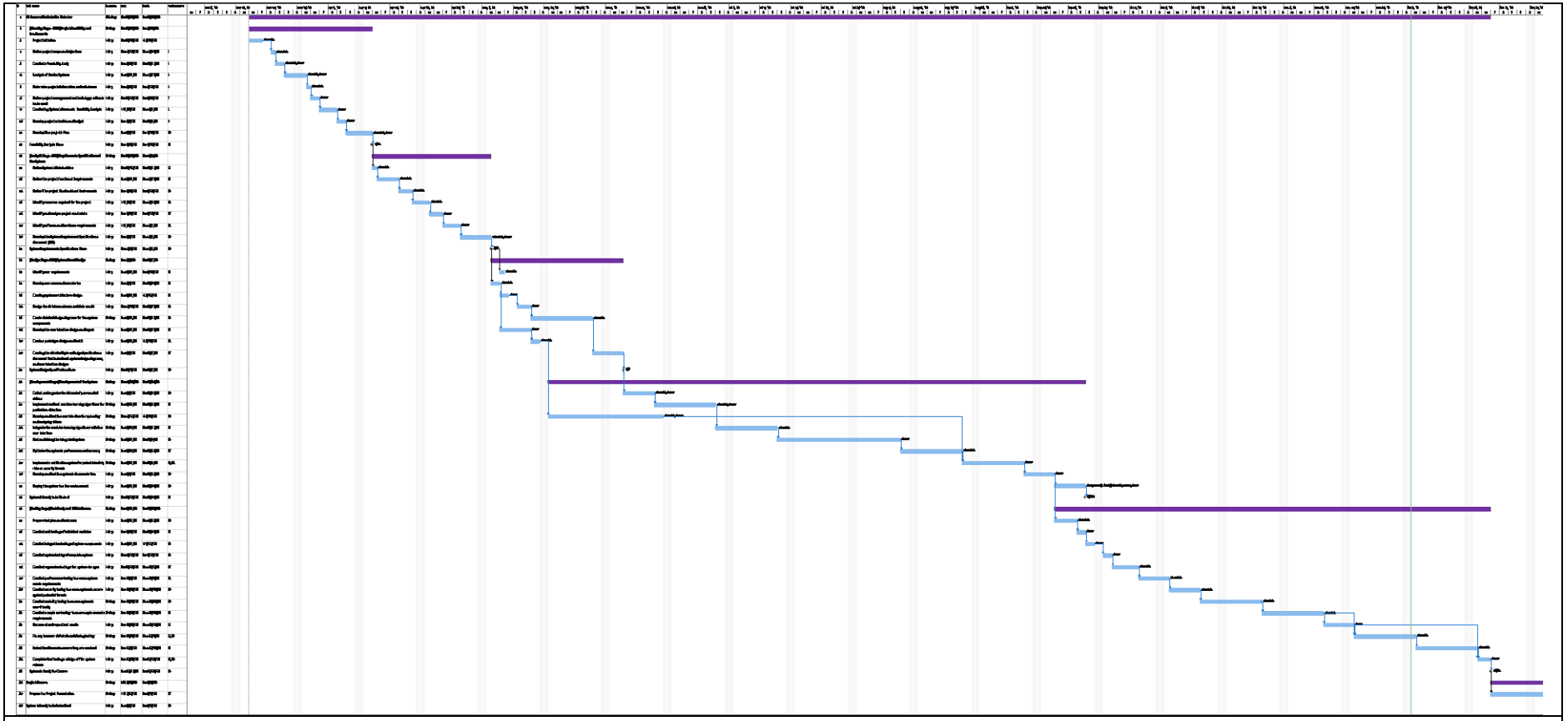


Figure 1: Gannt Chart

2.7 Work Packages Tables

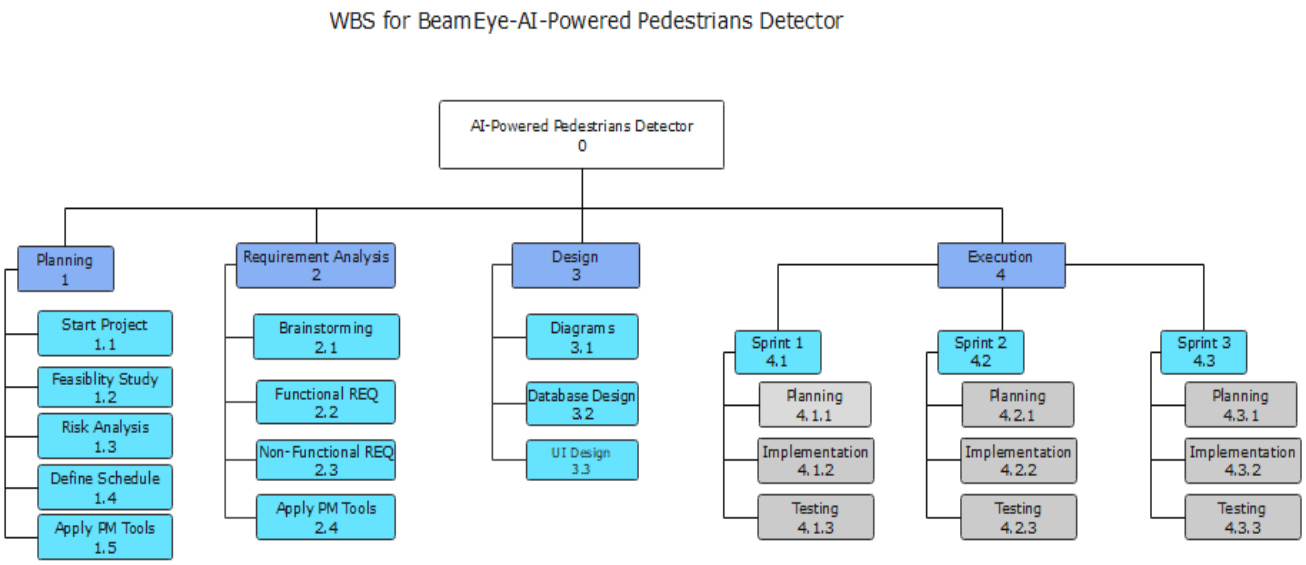


Figure 2: WBS Structure

Table 2: List of Work Packages

Work Package No	1
Work Package Name	**Project Feasibility and Pre-Research (Feasibility Analysis)**
Start-End Date and Time	15.03.2023 – 11.04.2023
Related Organizations	

1- List the activities of work packages.
1.1 Project Process and Economic Feasibility: At this stage we need to have an idea of the project's scope, objectives, and deliverables to ensure that they are achievable within the project timeline and budget. Market Analysis should be conducted to identify the demand for the AI-powered pedestrian detection system and potential customers. Financial Analysis should be conducted to estimate revenue projections, and COCOMO. Risk Assessment should be conducted to identify potential risks and develop risk mitigation strategies.

Resource Planning to identify the resources required for the project, including personnel, equipment, and technology.

1.2 Technological Feasibility:

The technological feasibility activities for Project Feasibility and Pre-Research (Feasibility Analysis) involves evaluating the technical requirements, resources, and capabilities necessary for developing and implementing the proposed project. These activities include analyzing technical requirements, assessing resources and capabilities, and evaluating technologies. These activities aim to ensure the proposed project is technically feasible and can be successfully implemented within available resources and capabilities.

2- Describe the methods and parameters that will be used for the work package.

The methods and parameters used for this work package will include market research, financial analysis, risk assessment, resource planning and various project management tools utilization. Moreover, multiple meetings with the project advisor will be conducted.

3- List the experiments, tests, and analysis in the work package.

Market research experiments, financial analysis tests, risk assessment analysis, resource planning experiments, similar system experiments, feasibility analysis, cost-benefit analysis.

4- List the output of work package and its success criterias.

Outputs:

Output of this phase is a PPM report (Project Planning Management Report) which includes all details about the project and the team.

Success Criterias:

The PPM is approved, and we can start gathering requirements for analysis stage.

5- Explain the relation of output with other work packages

This work package ensures the success of the project through the planning and feasibility study as it is the first stage to help understand the project feasibility.

Work Package No	2
Work Package Name	**Based System Design Technology (Analysis & Design stage)**
Start-End Date and Time	12.04.2023 – 07.06.2023
Related Organizations	

1- List the activities of work packages.

(Analysis Stage -SRS) Requirements Specifications of the System

System's Stakeholders will be defined

Define the project Functional Requirements

Define The project Nonfunctional Requirements

Identify resources required for the project

Identify and analyze project constraints

Identify software and hardware requirements

Develop the System Requirement Specifications document (SRS)

System Requirements Specifications Done

(Design Stage -SDS) System Based Design

Identify user requirements

Develop use cases and user stories

Creating system architecture design

Design the database schema and data model

Create detailed design diagrams for the system components

Develop the user interface design and layout

Create a prototype design and test it

Creating the detailed System Design Specifications document that includes all system design diagrams, and user interface designs

System Design Specification Done

2- Describe the methods and parameters that will be used for the work package.

- Functional and Nonfunctional requirements gathering and analyzing.
- Detailed design of the system through UML modeling
- System Architecture design and prototyping.
- User interface design

3- List the experiments, tests, and analysis in the work package.

- Usability testing: Testing the user interface design with a sample of end-users to ensure it is user-friendly and intuitive.
- Prototype testing: Testing a preliminary version of the system to identify design flaws and opportunities for improvement.

4- List the output of work package and its success criterias.

Outputs:

- Well organized System Requirements Specification document.
- Detailed System Design Specification document

Success Criterias:

- Well-structured system design and intuitive diagrams for use cases and flow of the operation in the system.
- Clear system architecture and modular hierarchy diagrams.
- Output documents are well organized and detailed.
- User-friendly user interfaces of the system are prototyped.

5- Explain the relation of output with other work packages

This work package is critical to ensure success of the project's development and guide us when starting its implementation. We will refer to this work package when testing to ensure that the system meets the requirements specified.

Work Package No	3
Work Package Name	**Development of System Software (Development Stage)**
Start-End Date and Time	09.06.2023–20.09.2023
Related Organizations	

1- List the activities of work packages.

- Collect and organize the dataset of prerecorded videos
- Implement and test machine learning algorithms for pedestrian detection
- Develop and test the user interface for uploading and analyzing videos
- Integrate the machine learning algorithms with the user interface
- Test and debug the integrated system
- Optimize the system's performance and accuracy
- Implement a notification system for potential safety risks or security threats
- Develop and test the system's documentation
- Deploy the system to a live environment
- System it Ready to be Tested

2- Describe the methods and parameters that will be used for the work package.

- Algorithms Development: PyCharm and VS code.
- Training the model.
- Deep learning for datasets.
- User interface: Figma or Adobe XD.
- Programming Languages: Python.
- Continuous Integration and deployment: GitHub.
- Agile Methodology for frequent feedback and adjustments.

3- List the experiments, tests, and analysis in the work package.

- Unit testing: testing individual software components or modules to ensure that they are functioning as expected.
- Integration testing: testing the integration of software components and modules to ensure that they are working together as expected.

- System testing: testing the complete system to ensure that it is functioning as expected and meeting all requirements.
- Performance testing: testing the performance of the system under various conditions, such as high traffic or heavy load.
- Usability testing: testing the system for ease of use and ensuring that it meets all user interface requirements.
- Code review: reviewing the code written.
- Software analysis: analyzing the software code and architecture to identify potential issues or areas for improvement.

4- List the output of work package and its success criterias.

Outputs:

- Well trained Model that is integrated to a software system.
- Software documentation.
- Software test results.

Success Criterias:

- Completion of all software modules on time and within budget.
- Successful integration of software modules into a complete software system.
- Passing of all software tests, including unit testing, integration testing, system testing, performance testing, security testing, and usability testing.
- Development of high-quality software documentation.
- Meeting all software development standards and best practices.

5- Explain the relation of output with other work packages

With this work package output we can start the prototyping and testing of the project for the next phase.

Work Package No	4
Work Package Name	**Prototype Implementation and Test Study and Maintenance (Test & Maintenance stage)**
Start-End Date and Time	14.09.2023–22.12.2023
Related Organizations	

1- List the activities of work packages.

- Prepare test plan and test cases
- Conduct unit testing of individual modules
- Conduct integration testing of system components
- Conduct system testing of complete system
- Conduct regression testing after system changes
- Conduct performance testing to ensure system meets requirements
- Conduct usability testing to ensure system is user-friendly
- Conduct acceptance testing to ensure system meets requirements
- Document and report test results
- Fix any bugs or issues.
- Retest fixed issues to ensure they are resolved
- Complete final testing and sign-off for system release

2- Describe the methods and parameters that will be used for the work package.

- Testing methods: Various testing methods will be used to evaluate the system's functionality, accuracy, and performance, including unit testing, integration testing, system testing, and acceptance testing.
- Test cases: Test cases will be developed to verify the functionality and accuracy of the system.
- Test data: Real-world and simulated test data will be used to test the system under different scenarios and conditions.
- Performance parameters: Performance parameters such as processing time, accuracy, and memory usage will be measured and evaluated.
- Hardware and software requirements: The system's hardware and software requirements will be identified and tested.

- Prototyping: A proof-of-concept prototype will be developed to test the system's functionality and identify any technical challenges.
- Iterative testing: Testing will be conducted in an iterative manner to identify and address any issues as they arise.

3- List the experiments, tests, and analysis in the work package.

- Unit testing.
- Integration testing.
- System testing.
- Performance testing.
- Usability testing.
- Compatibility testing.
- Data analytics.
- Prototype development.

4- List the output of work package and its success criterias.

Outputs:

- A Detailed and well-organized Test Report that documents all the results of all testing activities.
- Technical specifications for system that include all the necessary hardware and software requirements of the system

Success Criterias:

- Successful completion of all test cases and verification of the system's functionality and performance.
- The prototype meets all technical specifications and requirements.
- Identification and resolution of all system issues and bugs encountered during testing.
- The development of a comprehensive test report that documents all testing activities and results.
- Project is ready to be presented.

5- Explain the relation of output with other work packages

All the previous work packages have led to this last work package. Once the testing is completed the system is ready to be put on the market.

2.8 List of Milestones

Table 3: Milestones List

	Description of Output	Expected Time Interval
1	Project Feasibility and Pre-Research	15th.03.2023-11th.04.2023
2	Systems Requirements Specifications.	12th.04.2023-8th.05.2023
3	System Design Specifications	9th.05.2023-7th.06.2023
4	Development of the System	22nd.06.2023-20th.09.2023
5	Test Study and Maintenance	14th.09.2023-21st.12.2023
6	Project Closure	30th.12.2023 - 4th.01.2024

2.9 Risk Analysis

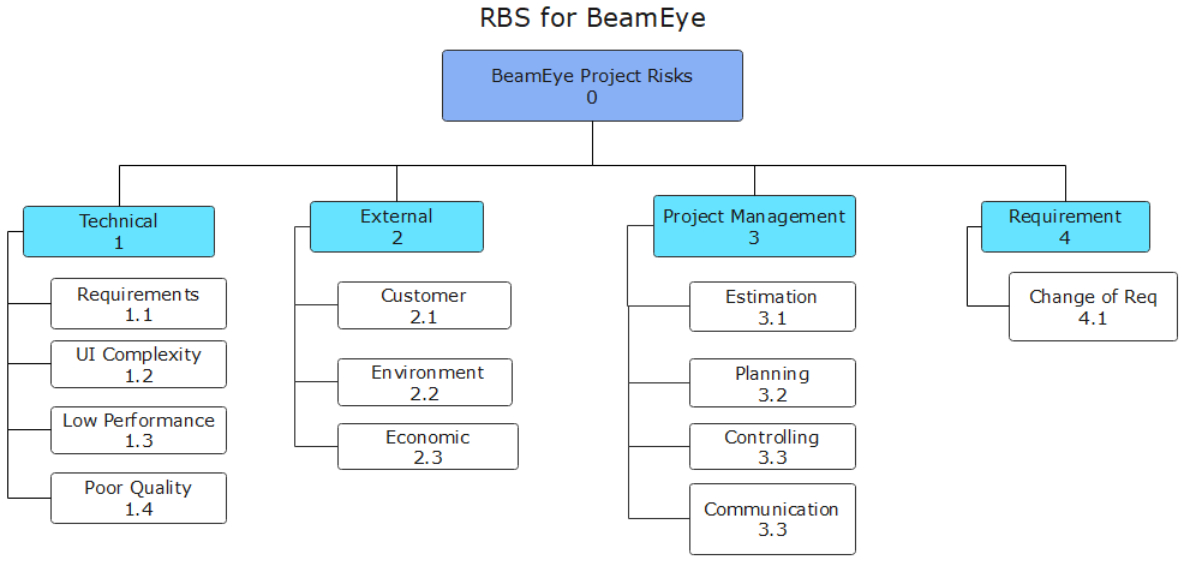


Figure 3: Risk Breakdown Structure

Table 4: Risks List

	Risk	Probability	Effects	Mitigation Strategy
1	Project Delays, increased costs, and decreased satisfactions of the project	High	Serious	To mitigate this risk, a thorough testing and quality assurance plan should be developed and implemented. This includes testing the system at every stage of development to ensure that it meets the requirements of the system. A contingency plan should also be developed in case any technical issues arise.
2	Software tools cannot work together in an integrated way.	Moderate	Serious	To mitigate this risk: conduct a comprehensive analysis of the software tools and their compatibility before procurement as well as appropriate training for the project team.
	The rate of defect repair is underestimated.	Moderate	Tolerable	Replace potentially defective components with more reliable bought-in components.
3	The size of the software is underestimated.	High	Serious	Investigate buying SW components. Investigate use of a program generator.
4	The database used in the system cannot process as many transactions per second as expected.	Moderate	Serious	Investigate the possibility of buying a higher-performance database.
5	The time estimated to complete the system is underestimated	High	Serious	Will need to search through similar systems project management plans.
6	Not finding enough datasets to train the model	Moderate	Tolerable	We will need to find datasets from the internet through research.
7	Any of team members leaves the group for sudden reasons	Moderate	Serious	We will need to put more effort and consider working more hours on the project to be able to finish.

2.10 Project Organization

Table 5: Project Team Organization

Personnel Name	Title	ID	Education Status	Graduation Date	Date of Starting Work	Idea Owner
Omar El Khadiji	Computer Engineer - System Analyst- UI	19700103	Undergrad.	25.02.2024	15.03.2023	Yes
Khawlah Alshubati	Software Engineer - Project Manager- ML	19701557	Undergrad.	25.02.2024	15.03.2023	Yes

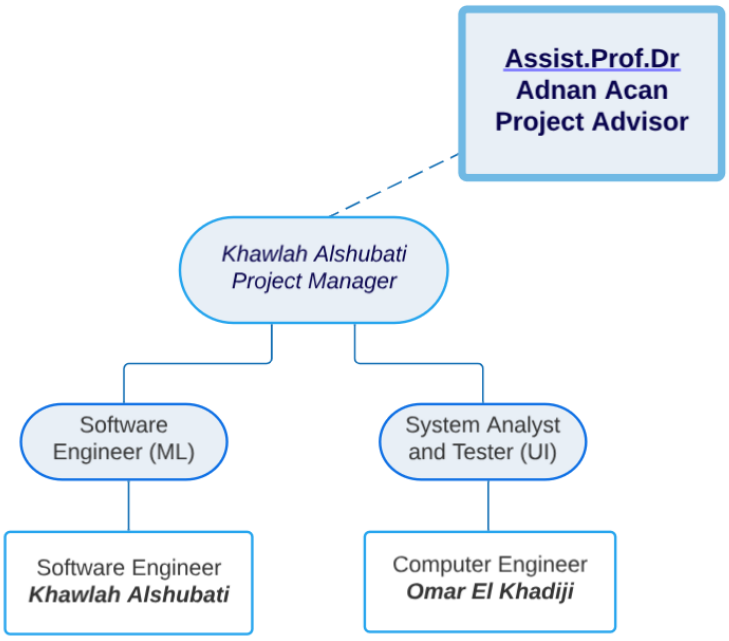


Figure 4: Organization Schema

2.11 Project Economic Expectation & National Outcomes

Time-to-market (month):	6 Months
The expected increase in sales revenue (%):	20%
The expected increase in market share (%):	30%
Time to start to gain:	July 2024

2.12 Equipment/Software/Release Purchases

Table 6: Software & Instruments Purchases

Project				**AI-Powered Pedestrians Detector**						
Line no	Instrument / Equipment / Software / Publication Name	No. / of Item	Capacity	Technical specification	Purpose of Project Activities	Post-Project Place of Use / Purpose		Unit Price (USD)	Unit Price (TL)	Total Amount (TL)
						R & D	Production			
1	Laptops	4		Min 16 GB RAM	Used For Programming and Searching and Training the Model			7000$	133000 TL	133000 TL
2	Internet Connection	4			Used For Research			300$	5880 TL	5880 TL
3	Surveillance Cameras	1			To get video Footage			100 $	1967 TL	1967 TL
4	Modelio Software	1			Used for Designing and drawing diagrams		Design	Free		
5	Lucid Chart	1			Used for Drawing some of the Diagrams		Drawing	Free		
6	Visual Studio Code	2		Implementing Algorithms	Used for Implementation and writing code	Coding	Implementation	Free		
7	MS Project	1			Used for Managing the Project		Management	Free		

8	MS Office	3			Used for Reports		Documenting	Free		
9	OneDrive	1			Used for Uploading		Communication	Free		
10	PyCharm	1		Coding	Used for Python code and AI algorithms		Implementation	Free		
									TOTAL	**140,847 TL**

2.14 Quarterly Estimated Cost Form (TL)

Table 7: Quarterly Estimated Cost Form

Project Name: AI-Powered Pedestrians Detector				
Cost Item	**2023-2024**		**TOTAL (TL)**	**TOTAL COST RATE OF CONTENTS (%)**
	I	**II**		
Personnel	5000	5000	10000	10%
Travel	6000	6000	12000	12%
Instrument / Equipment / Software / Publications	7000	7000	14000	14%
Domestic Works Made by R & D and Testing Institutions	1000	1000	2000	2%
International Works Made by R & D and Testing Institutions	4000	4000	8000	8%
Domestic Services Procurement	3000	3000	6000	6%
Overseas Service Procurement	5000	8000	13000	13%
Material	15000	20000	35000	35%
TOTAL COST	46000	54000	100000	100%
CUMULATIVE COST			100000	100%
IN THE PROJECT TOTAL PERSON-MONTH			**100000**	

2.15 Advanced COCOMO Analysis

2.15.1 Calculating KLOC:

0. KLOC = FB*Language Ratio
1. FP = UFP*[0.65+0.01*DI]

2.15.2 Calculating UFP:

Table 8: Unadjusted Function Points

Business Functions	Simple	Simple weight	Average	Average weight	Complex	Complex weight	UFPs
User Input (IT)	3	3	5	4	3	6	47
User Output (OT)	1	4	3	5	2	7	33
User Inquiries (QT)	2	3	1	4	7	6	52
Internal Files (FT)	1	7	1	10	3	15	62
External Interfaces (ET)	3	5	2	7	1	10	39
UFP =							**233**

2.15.3 Calculating DI:

Table 9: Development Instrumentation Table

Number	Factors	Complexity	Complexity Value
1	Data Communication	Average	3
2	Distributed data processing	Significant	4
3	Performance Criteria	Average	3
4	Online Data Entry	Incidental	1
5	High Transaction rate	Average	3
6	Heavily Utilized Hardware	Significant	4
7	Maintainability	Average	3
8	Online Updating	Incidental	1
9	Complex Computation	Incidental	1
10	Reusability	Average	3

11	Ease of installation	Essential	5
12	Ease of operation	Essential	5
13	Portability	Essential	5
14	End User Efficiency	Significant	4
DI =			**46**

2.15.4 Calculating FP:

2. FP = UFP*[0.65+0.01*DI]
3. **FP** = 233*[0.65+0.01*46] = **258.63**

2.15.5 Calculating KLOC (Python):

4. KLOC = FP * Language Ratio/1000
5. **KLOC** = 258.63 * 64/1000 = **16.55 KLOC**

2.15.6 Since KLOC = 16.55 then it is under organic mode and Intermediate Model:

6. **EAF** = DATA (Normal) * TIME(High) * TURN (High) * PCAP (Very High) * LEXP (High) * TOOL (Normal) * SCED (High) = 1*1.11 * 1.07*0.70*0.95*1.04 = **0.82**

Calculate Effort Estimate:

⇨ **E** = ai $(KLOC)^{bi}$ * EAF = 3.2*$(16.55)^{1.05}$*0.82 = **60.93 Person/Month**

Calculate Duration:

⇨ **D** = ci $(E)^{di}$ = 2.5 * 60.93 $^{\mathbf{0.38}}$ = **11.91 Months**

Calculate Staff Size:

⇨ **SS** = E/D = 60.93/11.91 = **5.11 Persons**

Calculate Productivity:

⇨ **P** = KLOC/E = 16.55/60.93 = **0.27 KLOC/PM**

2.15.7 Phases Calculations (detailed COCOMO)

Since KLOK is 16.55 which means is under organic small (≈2) category

The calculations for effort and schedule fractions in each phase of the life cycle will be as following:

Plan and requirements:

E = 60.93 * 0.06 = 3.65 PM

D = 11.91 * 0.10 = 1.19 Months

System Design:

E = 60.93 * 0.16 = 9.74 PM

D = 11.91 * 0.19 = 2.26 Months

Detailed Design:

E = 60.93 * 0.26= 15.84 PM

D = 11.91* 0.24 = 2.85 Months

Module Code and Test:

E = 60.93 * 0.42 = 25.59 PM

D = 11.91* 0.39 = 4.64 Months

Integration & Testing:

E = 60.93 *0.16= 9.74PM

D = 11.91 *0.18 = 2.14 Months

2.16 CPM & Pert Analysis

Table 10: Tasks Duration

Task ID	Task name	Duration (days)	Dependency
A	Project Initiation	3	
B	Feasibility Study and Project Planning	17	A
C	System Requirements Specifications	22	A, B
D	System Design Specifications	19	C, B
E	User Interface Development	10	C, D
F	Developing Software Modules	30	D

G	Integrating Software Modules	20	F, D
H	System Testing	71	G
I	Deployment of the System	10	G, H

Table 11: Tasks Path Duration

Path Number	Path	Duration (days)
1	A B D G H I	140
2	A B D F G H I	170
3	A B D F G I	99
4	A B D G I	69
5	A C D F G H I	175
6	A C E D F G H I	185
7	A C D F G I	104
8	A C B D F G H I	192
9	A C B D G I	91
10	**A B C E D F G H I**	**202**

2.16.1 Tasks Variance and Deviation:

Table 12: Tasks Variance and Standard Deviation

Task ID	Dependency	O (Min)	Duration	P (Max)	Duration	Variance	Standard Deviation
A	-	1	3	5	3	0.44	0.67
B	A	10	17	20	16.33	2.78	1.67
C	A, B	17	22	25	21.67	1.78	1.33
D	C, B	15	19	27	19.7	4	2
E	C, D	5	10	20	10.83	6.25	2.5
F	D	20	30	40	30	11.11	3.33
G	F, D	10	20	30	20	11.11	3.33

H	G	55	71	85	17.67	25	5
I	G, H	5	10	15	10	2.78	1.67

2.16.2 Paths Variance and Deviation:

Table 13: Paths Variance and Deviation

Path	Total Expected Time for Each Path	Total Variance	Total Standard Deviation
A B D G H I	86.7	46.11	6.79
A B D F G H I	116.7	57.22	7.56
A B D F G I	99.03	32.22	5.68
A B D G I	69.03	21.11	4.59
A C D F G H I	122.02	56.22	7.49
A C E D F G H I	132.87	62.47	7.90
A C D F G I	84.67	31.22	5.59
A C B D F G H I	138.37	55	7.42
A C B D G I	90.7	22.89	4,78
A B C E D F G H I	**149.2**	**65.25**	**8.07**

2.16.3 Probability:

Table 14: Paths Probability of Completion

Path	Specified Time	Total Expected Time	Path Standard Deviation	Z-Value	Probability of Finishing
A B D G H I	140	86.7	6.79	7.84	100%
A B D F G H I	170	116.7	7.56	7.05	100%
A B D F G I	99	99.03	5.68	-0.05	0.4801 (48%)
A B D G I	69	69.03	4.59	-0.06	0.4681 (47%)
A C D F G H I	175	122.02	7.49	7.07	100%
A C E D F G H I	185	132.87	7.90	6.59	100%
A C D F G I	104	84.67	5.59	3.46	0.9997 (99%)

A C B D F G H I	192	138.37	7.42	7.23	100%
A C B D G I	91	90.7	4.78	0.06	0.5239 (52%)
A B C E D F G H I	**202**	**149.2**	**8.07**	**6.54**	**100%**

2.17 Network Diagram

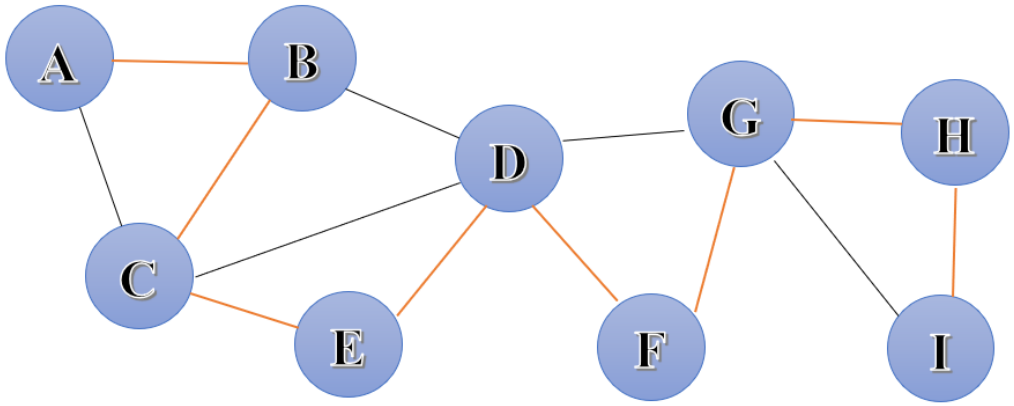


Figure 5: Network Diagram and Critical Path

2.18 Project Crashing Approach

The method will be applied for reducing 19 days (about 2 and a half weeks)

Table 15: Crashing Approach Table

Task ID	Normal Time (days)	Crash Time (days)	Normal Cost ($)	Crash cost ($)	Max # Days to reduction	Reduced Cost Per day
A	3	2	500	1500	1	1000
B	17	10	2500	5000	7	357
C	22	18	3000	5000	4	500
D	19	15	2500	4500	4	500
E	10	8	1000	2000	2	500
F	30	27	4000	6000	3	666

G	20	18	2800	5500	2	1350
H	71	66	11000	14500	5	700
I	10	8	1000	2000	2	500
Total	202	172	28300	46000	30	

Project Normal Cost: **28300$**

Tasks chosen to reduce 19 days (about 2 and a half weeks) are B, C, D, E, and I

Accordingly: for B: 7*357 = **2499,** for C: 4*500 = **2000,** for D = 4*500 = **2000, for** E= 2*500 = **1000,** for I = 2 * 500 = **1000**

Total for 19 Days reduction **= 8499$**

Project final crashing cost **= 28300 + 8499 = 36799 $**

Additional Cost **= 36799 – 28300 = 8499**

Final Project Completion **Time = 202 – 19 = 183 Days**

2.19 Project Log

Project Title: AI-powered Pedestrian Detector

Start Date: 15-03-2023 **End Date**: 22-01-2024

Date	Tasks Done / To Do
15-03-2023	İnitiate the project and discussed timeline and tasks
30-03-2023	Finished ppm and started working in functional requirements
15-04-2023	Started design and implementation
20-04-2023	Discussed Design issues
30-04-2023	Discussed and solved design issues
21-05-2023	Finilized srs and sds reports for submission
01-06-2023	Submitted final report
01-07-2023	Discussed implementation of model
20-07-2023	Discussed progress of the implementation
30-08-2023	Worked on solving technical issues faced during development
30-09-2023	Checked model and started the user interface
15-10-2023	Doing more research on implementation
20-10-2023	Solving problems we faced in implementaion
07-11-2023	Met with supervisor for model confirming
15-11-2023	Corrected Issues faced with model
21-11-2023	Discussed the integration issues
29-11-2023	Solved Technical Issues faced last meeting
11-12-2023	Finilazed SRS and SDS as well as implemented the new algorithm
23-12-2023	Confirmed the final report with supervisor
30-12-2023	Finilized the project with implementation

Figure 6: Project Log

3. REQUIREMENTS ANALYSIS

This system is designed for one actor who can upload video footage for pedestrians walking in street with different postures and positions, then see the result where people are detected and labeled.

3.1 Functional Requirements

1. Uploading a video

1.1 Description and Priority

This functionality allows users to upload video footage capturing pedestrians in different postures and positions. This is of High priority as it is the primary function of the system

1.2 Stimulus/Response Sequences

Stimulus: user clicks on the “Upload Video” button

Response: system opens a file dialog for video selection

1.3 Functional Requirements

REQ-1: system shall provide “Upload Video” button in the main screen

REQ-2: system shall support the upload of video files of any type

REQ-3: system shall validate uploaded videos for compatibility and provide appropriate error messages for incompatible formats.

2. Viewing Detected Pedestrians

2.1 Description and Priority:

This functionality allows the user to view the uploaded video alongside the detection results, including labeled pedestrians. This function is also of High priority for core functionality.

2.2 Stimulus/Response Sequence:

Stimulus: User selects a video for processing.

Response: System displays the original video and the processed video with labeled pedestrians.

2.3 Functional Requirements:

REQ-4: system shall display original video on one side of the screen

REQ-5: system shall display processed video with labeled pedestrians on the other side

REQ-6: number of detected pedestrians should be shown

REQ-7: system shall provide buttons for other actions like download, screenshot or upload another video

REQ-8: system shall allow user to choose whether to keep result of input video or not.

3. Downloading Video and Taking Screenshots

3.1 Description and Priority:

This functionality enables the user to download the processed video and capture screenshots of the detection results. Medium priority for enhancing user experience.

3.2 Stimulus/Response Sequence:

Stimulus: User clicks on the "Download Video" or "Take Screenshot" button.

Response: System initiates the download or captures and saves a screenshot.

3.3 Functional Requirements:

REQ-9: system shall provide a "Download Video" button for the user to save the processed video.

REQ-10: system shall provide a "Take Screenshot" button for capturing and saving screenshots.

REQ-11: Screenshots shall include the labeled pedestrians and relevant information.

REQ-12: system shall allow users to specify the destination for downloaded files.

4. User Preference and Settings

4.1 Description and Priority:

This feature allows users to customize their preferences for the detection display. It is of Low priority since it is just for better user experience.

4.2 Stimulus/Response Sequence:

Stimulus: user clicks on the settings gear icon and adjusts preferences.

Response: system applies the selected preferences to the detection display.

4.3 Functional Requirements:

REQ-13: settings menu shall include options to toggle the inclusion of labels in the detection display.

REQ-14: settings menu shall include options to toggle the inclusion of accurate information in the detection display.

REQ-15: settings menu shall include an option to enable or disable crowd detection option.

REQ-16: settings menu shall include an option for user to choose a color for crowd labels.

REQ-17: settings menu shall include an option for user to choose a color for single pedestrian labels.

REQ-18: settings menu shall include a "Save Changes" button to confirm and save user preferences.

REQ-19: settings menu shall include a "Change Settings" button to discard changes and return to the previous settings.

3.1.1 Use case Glossary

Table 16: Use Case Glossary

Use Case ID	Description	Priority	Actor
UC-01	Uploading a Video of any type and should be validated by system	High	System's User
UC-02	Viewing Detected Pedestrians and Displaying their number	High	System's User
UC-03	Saving Video and Taking Screenshots from the detected pedestrian's video	Medium	System's User
UC-04	User Preference and Settings where user can change colors of bounding boxes and enable, disable crowd detection	Low	System's User

3.1.2 Use Cases Narratives

Table 17: Use Case 01 Narrative Table

Use Case ID	**UC-01**
Description	Uploading a Video
Priority	High
Actor	User
Precondition	System is running and settings are set
Trigger	User clicks on the "Upload Video" button
Basic Flow	1. User clicks on the "Upload Video" button on the main screen. 2. System responds by opening a file dialog for video selection 3. System validates the video path. 4. If the video is valid, the system proceeds to process it. 5. If the video invalid or corrupt, the system provides an appropriate error message.
Postcondition	The selected video is uploaded and ready for processing or an error message is displayed if the video is incompatible.
Alternate Flow	If the user cancels the file selection, the use case terminates.
Functional Requirements	REQ-1: system shall provide “Upload Video” button in the main screen REQ-2: system shall support the upload of video files REQ-3: system shall validate uploaded videos for compatibility and provide appropriate error messages for incompatible formats.

Table 18: Use Case 02 Narrative Table

Use Case ID	**UC-02**
Description	Viewing Detected Pedestrians
Priority	High
Actor	User
Precondition	System is running and video is uploaded
Trigger	User selects a video for processing
Basic Flow	1. User selects a video for processing 2. System displays the original video on the side of the screen.

	3. System displays the processed video with labeled pedestrians on the other side of the screen. 4. The number of detected pedestrians is shown on the screen 5. The system provides buttons for other actions like Save video, screenshot or upload another video
Postcondition	The user can view the original video alongside the detection results, including labeled pedestrians. The user can access additional actions through the provided buttons.
Alternate Flow	If the user chooses to take alternative actions, they can access other features or exit the viewing process.
Functional Requirements	REQ-4: system shall display original video on one side of the screen REQ-5: system shall display processed video with labeled pedestrians on the other side REQ-6: number of detected pedestrians should be shown as well as the title of the video REQ-7: system shall provide buttons for other actions like save video, screenshot or upload a video

Table 19: Use Case 03 Narrative Table

Use Case ID	**UC-03**
Description	Saving Video and Taking screenshots
Priority	Medium
Actor	User
Precondition	System is running and Video Processing is completed
Trigger	User clicks on “Save Video” or “Screenshot”
Basic Flow	1. After video processing is completed, the user is presented with options to either save the processed video or take screenshots of the current frame with detected people. 2. If the user clicks on the "Save Video" button: a. The user is prompted to specify the destination for the downloaded video file. c. The system saves the processed video to the specified destination.

	d. The system provides a confirmation message to the user upon successful saving. 3. If the user clicks on the "Screenshot" button: a. The system captures and saves a screenshot of the detection results, including labeled pedestrians and relevant information. b. The system saves the screenshot to the screenshots folder in the app's directory. d. The system provides a confirmation message to the user upon successful screenshot capture.
Postcondition	Processed Video and Screenshots are saved to the specified destinations
Alternate Flow	If user encounters any technical issue, an error message indicates the problem should appear
Functional Requirements	REQ-8: system shall provide a "Save Video" button for the user to save the processed video. REQ-9: system shall provide a "Screenshot" button for capturing and saving screenshots. REQ-10: Screenshots shall include the labeled pedestrians and relevant information. REQ-11: system shall allow users to specify the destination for Saved files.

Table 20: Use Case 04 Narrative Table

Use Case ID	**UC-04**
Description	User Preference and Settings
Priority	Low
Actor	User
Precondition	System is running
Trigger	User clicks on the settings gear icon and adjust preferences
Basic Flow	1. The user wishes to customize their preferences for the detection display and clicks on the settings gear icon. 2. The system responds by opening the settings menu, presenting various options for adjusting preferences. 3. Within the settings menu, the user can: a. Toggle the inclusion of labels in the detection display. b. Toggle the inclusion of accuracy information in the detection display. c. Enable or disable crowd detection. d. Choose a color for crowd bounding-boxes. e. Choose a color for non-crowd bounding-boxes. f. Choose output folder for processed video g. Choose whether or not to change the resolution of the processed video 4. After making their desired selections, the user can either: a. Click the "Save Changes" button to confirm and save their preferences. b. Close window without saving to discard changes and return to the previous settings. 5. The system applies the selected settings for the next upload
Postcondition	The user now can view the detection results according to their chosen settings in the next upload

Alternate Flow	If user encounters any technical issue, an error message indicates the problem should appear If user attempts to choose same colors for non-crowd and crowd, system will not allow them.
Functional Requirements	REQ-12: settings menu shall include options to toggle the inclusion of labels in the detection display. REQ-13: settings menu shall include options to toggle the inclusion of accuracy information in the detection display. REQ-14: settings menu shall include an option to enable or disable crowd detection option. REQ-15: settings menu shall include an option for user to choose a color for crowd bounding-boxes. REQ-16: settings menu shall include an option for user to choose a color for non-crowd bounding-boxes. REQ-17: settings menu shall include a "Save Changes" button to confirm and save user preferences. REQ-18: settings menu shall include a "Discard Changes" option to discard changes and return to the previous settings. REQ-19 user can choose to change the resultion of the output video.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

Performance requirements are crucial for our system since it influences the user's overall experience. We will focus more on systems response time and scalability to provide better performance requirements.

i Response Time:

- The loading of input video should take no more than a second which we implemented it by using OpenCv's "VideoCapture" function which takes the index of the first frame instead of reading the entire video. [1]
- The time for merging video should not take more than the duration of the input by creating a "VideoWriter" object from OpenCv, looping through the processed frames and adding each to that object.[1]
- The processing of the video should be done in a reasonable time-no more than 1.5xthe duration of the video by compressing the original video and lowering the number of frames processed in each second.

- The user interface should take less than 1 second to load which is implemented by starting a thread that opens the UI as soon as user opens the app.
- Buttons' actions and feedback should be instantaneous by using Tkinter's built-in parameter "command" That starts the associated function with buttons as soon as they are clicked. [2]
- Video slicing and handling time should be instantaneous by using OpenCv's frame by frame analysis.

ii Scalability:

- Utilizing Thread Manipulation and garbage disposal, the application should use system resources efficiently.
- OpenCv's Frame by Frame techniques are used for efficient memory management to handle varied sizes of videos. [1]
- Large input sizes should not cause output quality degradation by conserving the properties of the input video.
- Utilizing OpenCv's color conversion functions ensures the desired output matches the user's settings.

3.2.2 Safety Requirements

Safety requirements are needed in our system to ensure that the system operates reliably and accurately preventing unintentional harm to users or environment [3]. Where we will ensure that our system is safe by implementing User Data Protection practices.

i User Data Protection:

- The system shall not access data other than what users provide. This ensured using Tikinter's File Dialog Module. [4]
- The system shall not modify the original user's data by applying the necessary changes to a different copy of that data.
- Any notifications or warnings presented to users must be clear.

3.2.3 Security Requirements

Security requirements are important to prevent unauthorized access to sensitive information and protecting against malicious attacks [3]. in our system, since it is a desktop app that does not require any authorization or authentication process, we need to ensure security through securing the settings and files handling processes.

i Secure File Handling:

- By specifying the input data to be of video type, we ensure that no suspicious files can be used to endanger the user or their data.
- By specifying the output folder, the user can choose a secure location to keep their data.

3.2.4 Software Quality Attributes

The software quality attributes are crucial in shaping the overall performance, reliability, and usability of the system. Our system should be available, reliable, and usable and this will be implemented throughout the following:

i Usability:

- The user interface should be intuitive, and the application should be easy to navigate for users with varying levels of technical expertise.
- User preferences, including color choices and visibility options, should enhance the overall user experience.

ii Reliability:

- The system should be robust and stable, minimizing crashes or unexpected behavior.
- The system will be tested to ensure it behaves predictably and consistently across different scenarios.

iii Maintainability:

- The codebase should be well-organized and documented to facilitate future updates or modifications.
- Git and GitHub are used to track changes made to code collaboration.

iv Availability:

- The system should be available 24 hours, 7 days a week for use.

v Testability:

- It should be easy to write and execute automated tests for the system by putting the needed variables in separate files.
- Debugging and troubleshooting will be implemented through logging and monitoring functionalities

vi Interoperability:

- system can interact seamlessly with other systems or components in case integrated with external services like a surveillance camera. (TODO)

3.2.5 Business Rules

Business rules influence how the system operates and what constraints it adheres to. Our system must meet the user expectations when it comes to accuracy and time to upload and process the video. System will be in English language only and should comply with relevant privacy regulations. Additionally, only those in charge of street safety or smart city planning or human research can use the system for pedestrians' safety and security purposes.

3.3 Realistic constraints

The system must adhere to the realistic constraints of time and budget that we have as students and make it scalable, adaptable, and user-friendly to meet the needs of different organizations and communities. Other constraints may include:

Limited hardware resources: The project may need to be implemented on a system with limited computing resources, which may affect the complexity of the algorithms used for pedestrian detection and tracking.

Limited dataset: The performance of the system may depend on the availability and quality of the dataset used for training and testing the AI models.

Ethical considerations: The system should be designed in a way that does not infringe on privacy rights.[5]

Only People in charge can use the system such as police officers or researchers since they have the authority to inspect people walking in streets through surveillance cameras. Such a product does not consume much power, so it has no effect on pollution. Additionally, the system can be used in the long term.

3.4 Ethical issues

Such system can raise several ethical issues such as:

Privacy Concerns: The recorded videos might have unintentionally taken pictures or videos of people who did not want to be seen or recorded, infringing their right to privacy.[5]

Algorithmic bias: The AI model may be prejudiced and behave discriminatorily, misidentifying pedestrians according to their gender, color, or other attributes, which could cause unexpected behavior.[6]

4. DESIGN

4.1 High level design (architectural)

Figure 7: High Level Design (Architectural Diagram)

Looking at the architectural diagram above, it is shown that there is only one user who can access the system via a desktop app user interface. The user uploads the video of people to be detected through the UI and this video will be processed using some processing algorithms and techniques after the video processing completion, the video will go through the ML model that detects people. If people were detected, the processed video will be returned to user, showing the detected people with colored bounding boxes and number of people displayed. Then the user can view the results and accuracy of detected people.

4.2 Software design

4.2.1 Use Case Diagram

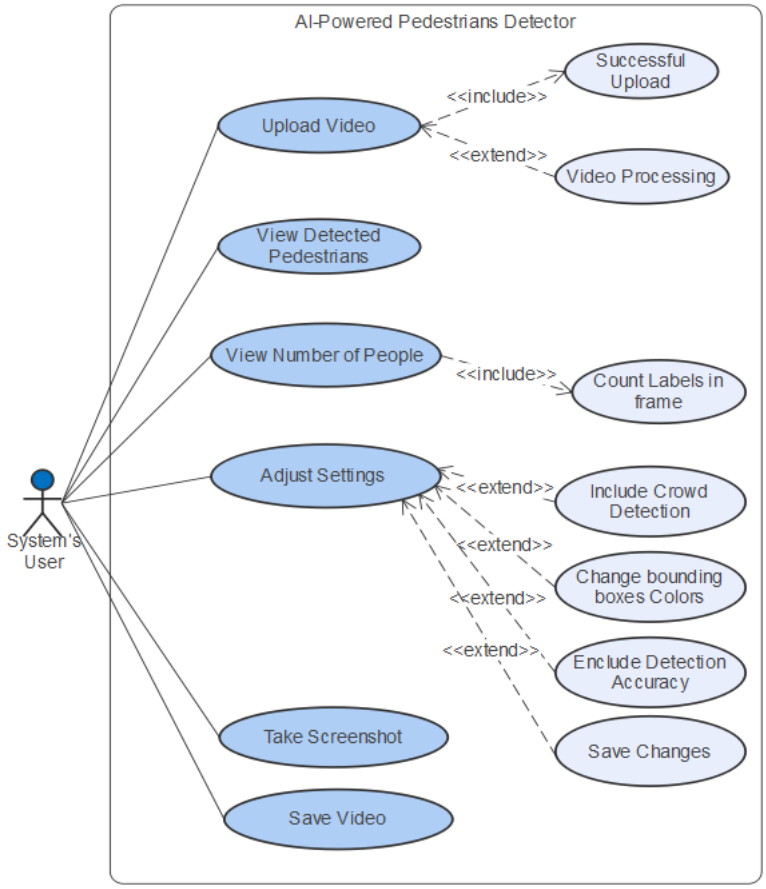


Figure 8: Use Case Diagram

This diagram shows the functional requirements of the system, and it works as a visual representation of how the user will interact with the system. The actor is the system's user who can be a police officer or a researcher and the ovels show the use cases which are the functionalities of the system. The extend and include relationships show which functions should appear because of others and which ones must include others. As shown uploading video includes a successful video upload and extends to video processing. The view number of people functionality includes counting and labeling people in frames. The preference settings extend to inclusion of crowd detection and detection accuracy, saving changes and changing bounding boxes of single pedestrians or grouped pedestrians.

4.2.2 Context Diagram

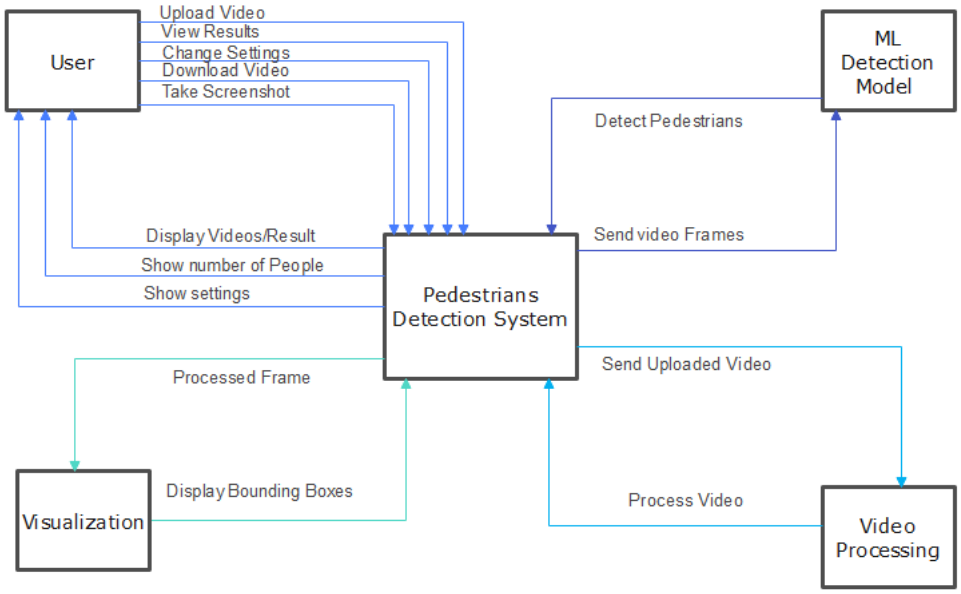


Figure 9: Context Diagram

This is a high-level visual representation that provides an overview of the system and its interactions with different entities. The center is the system which has three main components:

- User who can upload video, view results, change settings, download video, and take a screenshot. While the system returns to the user the resulting video displayed, number of people and settings.

- Video processing Module which interacts directly with the system where videos are uploaded and processed.

- The machine learning model module that interacts directly with the user interface of the system after video is processed. It returns to the system the detected people, bounding boxes around them and number of people as well as detecting crowds.

4.2.3 Modular Hierarchy Diagram

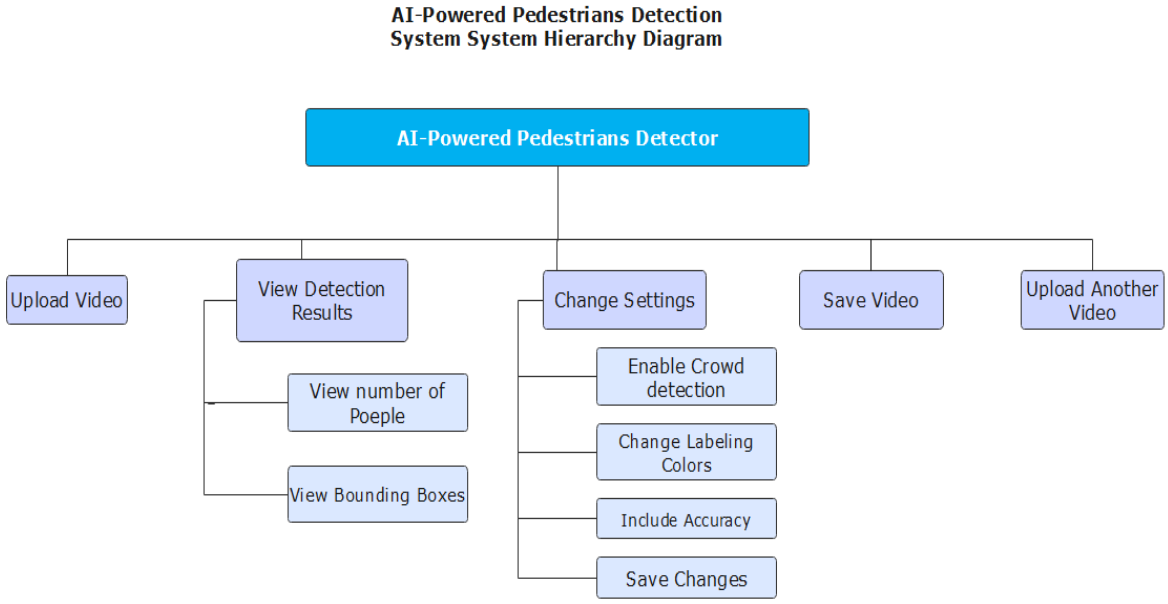


Figure 10: Modular Hierarchy Diagram

This is the module dependency diagram that shows the hierarchal structure of the system. The main modules are uploading video, viewing detection results, changing settings, saving video, and uploading another video. While the sub modules are viewing number of people, viewing bounding boxes, enabling crowd detection, changing labeling colors, including accuracy, and saving changes. This diagram breaks down the system into smaller manageable modules that we can look at while developing the system and plays a pivotal role when doing the integration testing of these different modules.

4.2.4 State Transition Diagram

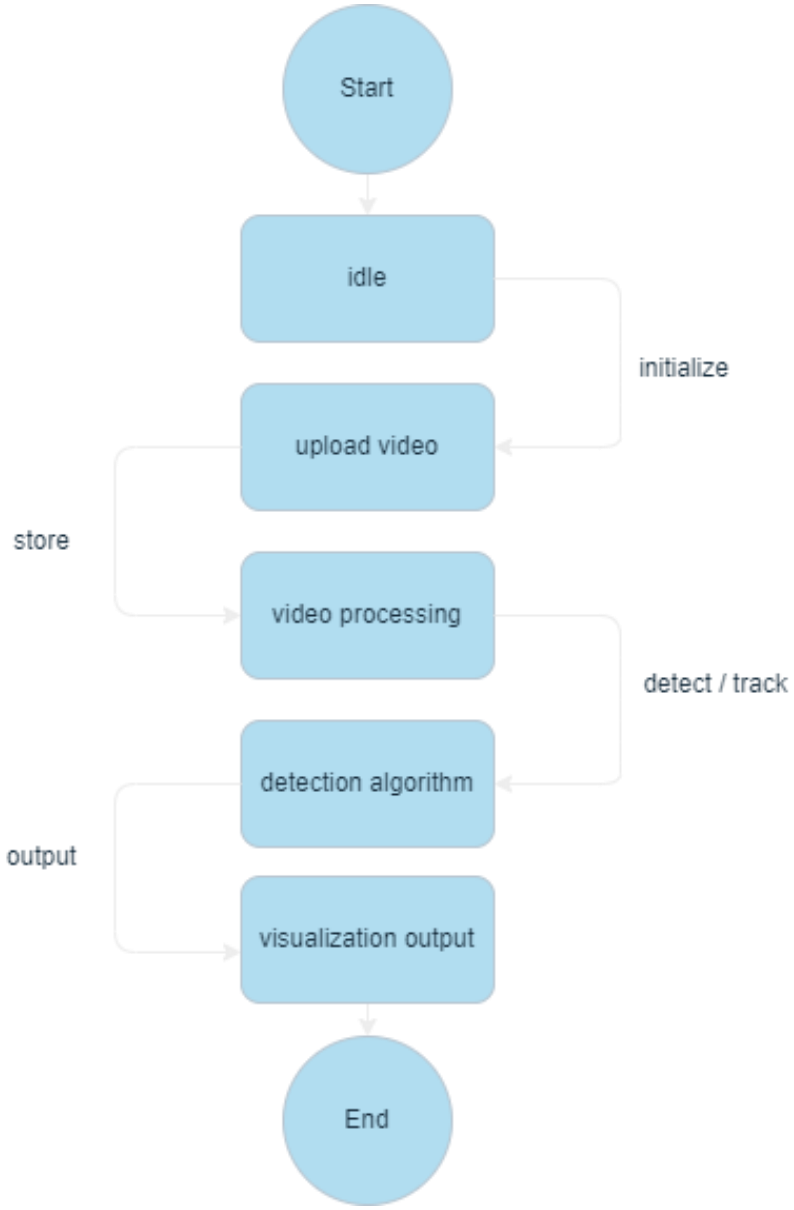


Figure 11: State Transition Diagram

This diagram was drawn to represent the system's behavior using states and transitions. The circles are the nodes showing the states and arrows are the transitions between these states. So, the transition flow goes as follows: first user runs the system, uploads a video, the video is processed and passed through a detection model that uses various detection algorithms after that the video is returned to user as a visualization output.

4.2.5 Sequence Diagram

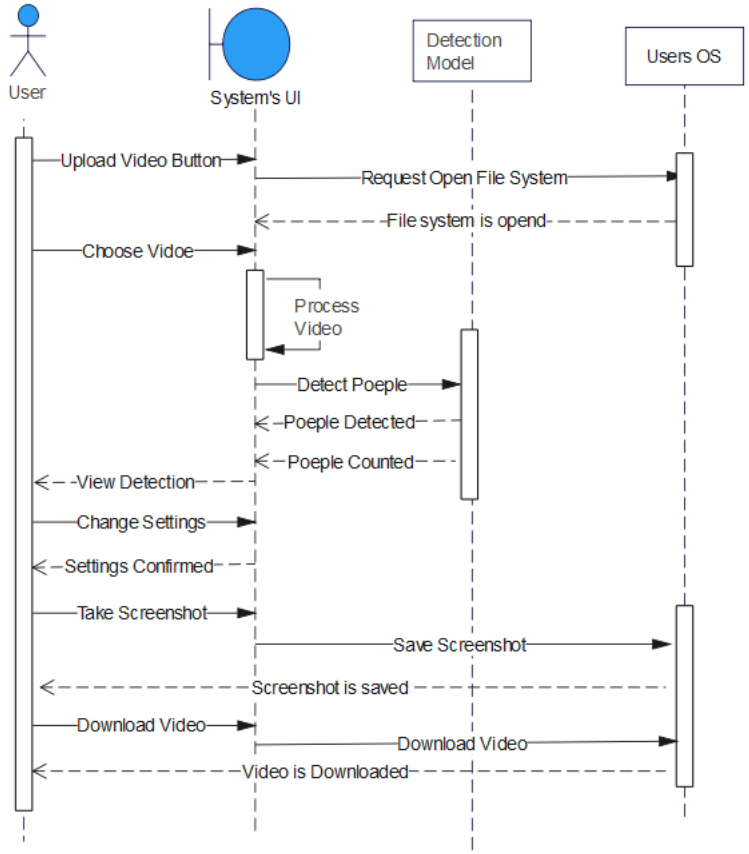


Figure 12: Sequence Diagram

This diagram visualizes the interaction between the components of the system over time. We used this diagram to understand and document the dynamic behavior of the system. The components here are the user interacting directly with the user interface by uploading a video and viewing the resulted processed video that has the bounding boxes around people and crowd are detected with a distinct color. The user after that can chose to change settings according to their preferences, they can download the video, upload another video, or take a screenshot which will be saved to the users file system storage.

4.2.6 BPMN Diagram

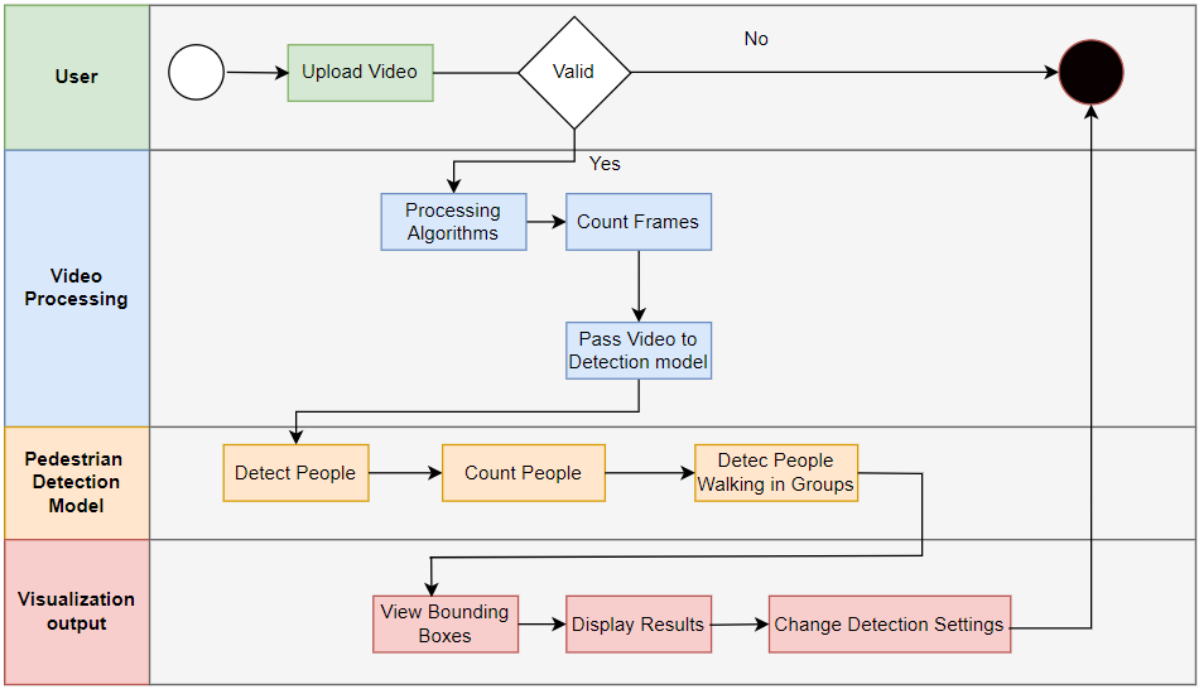


Figure 13: Business Process Model and Notation Diagram

The BPMN diagram was drawn to provide notations for the representing business processes within the system and their communication. The circles indicate the start and end events while boxes show the activities down and arrows are to represent the sequence flow between these activities. Starting form uploading a video, checking for its validity, and ending with user changing the settings according to his preferences and viewing the results with bounded boxes around them.

4.2.7 Activity Diagram

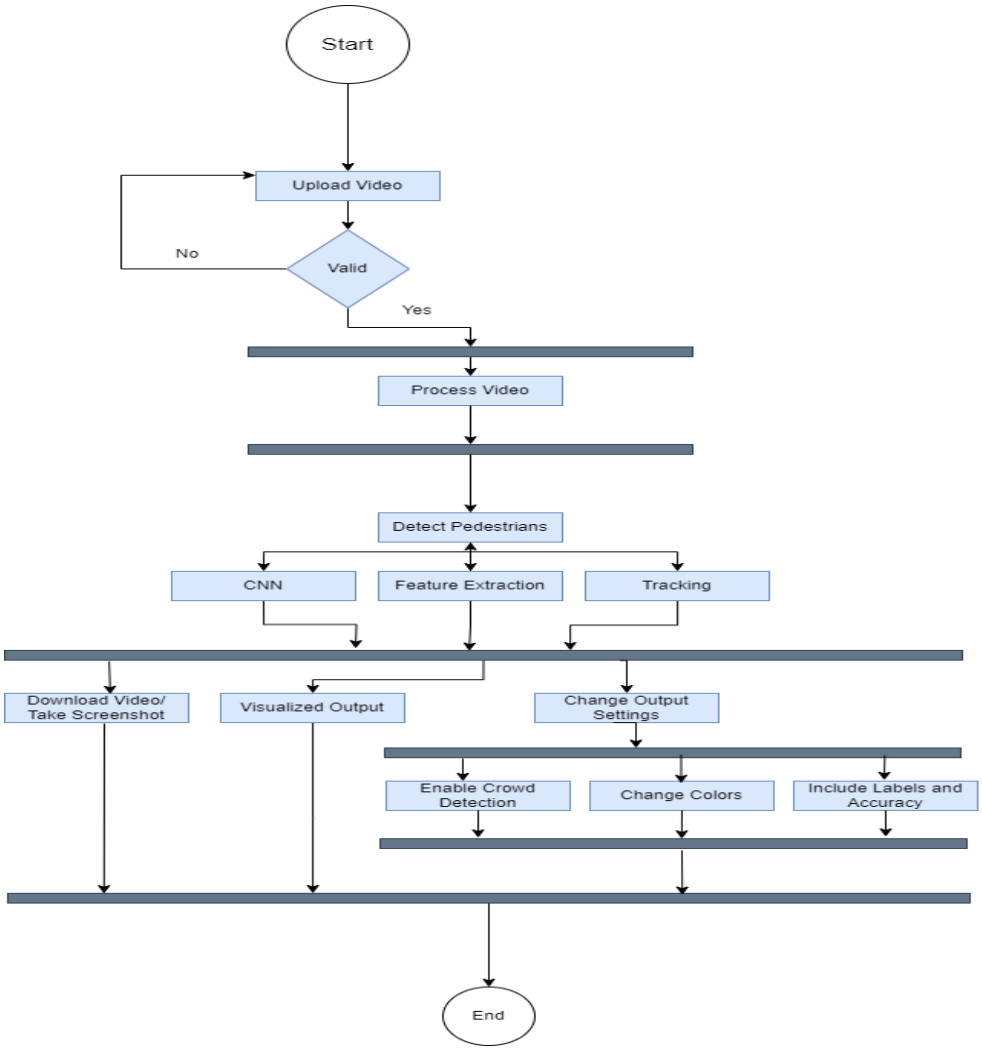


Figure 14: Activity Diagram

The activity diagram was made to model the dynamic aspects of the system, more specifically the flow of activities, actions, and processes withing the system. This diagram shows the different activities that represent the system's behavior, in which sequence each activity occurs, and the relationship between the activities. First, we have the task of uploading a video then this task is checked with a control flow arrow and a decision node if video is valid then it is processed and then passed to the model for people to be detected. After people are detected, the processed video is shown to the user, who can choose to download the video or take a screenshot. Users can change the settings as well according to their preferences and this part is shown with fork and join relationships. There are also the initial and final nodes and flow of control as shown in the diagram in figure12.

5. IMPLEMENTATION

5.1 Tools, technologies and platforms used

5.1.1 For Model

TensorFlow (Open-Source Library) [7]

SSD MobileNet Model (Pretrained Model available in TensorFlow) [8]

COCO Dataset (Common Object Common Context, over 200000 labeled images with over 1.5 million object instances across 80 object categories) [9]

5.1.2 For User Interface

Tkinter (GUI library for Windows Applications) [10]

OpenCV (Open-Source Computer Vision Library) [11]

NumPy (Numerical Processing Library for Data Manipulation) [12]

Pillow (Python Imaging Library) [13]

Python (Primary Programming Language Supports TensorFlow and ML Libraries) [14]

5.1.3 Environments and Platforms

OS: Windows

IDE: VS code and PyCharm, Google Colab

Version Control: GitHub

5.2 Algorithms

5.2.1 Object Detection and Frame Processing

Table 21: Algorithm 1- Object Detection and Frame Processing

```
1.  FUNCTION Detect()
2.      Wait until the user input video is ready
3.      IF video needs to be resized THEN
4.          Resize video for faster processing
5.      Initialize video capture with the input video
6.      Load user settings from 'userSettings.txt'
7.      Set color coding for object detection based on settings
8.      Initialize frame count and object count lists
9.      WITH TensorFlow detection graph
10.         Initialize TensorFlow session
11.         WHILE video has frames
12.             Read frame from video
13.             IF read is successful THEN
14.                 Expand dimensions of frame for TensorFlow compatibility
15.                 Run detection model on the frame
16.                 Draw bounding boxes and get object counts from frame
17.                 Save the processed frame if needed
18.                 Update progress every few frames
19.             ENDIF
20.         END WHILE
21.         Convert processed frames back to video
22.     Release video capture
23.     Update sharedVariables with object counts
24.     Set detection completion flag in sharedVariables
```

5.2.2 Frames to Video Algorithm

Table 22: Algorithm 2- Frames to Video

1. FUNCTION frames_to_video(fps, output_folder='videoOut//')
2. Set the output folder and ensure it ends with "//"
3. Set the image folder where frame images are stored
4. Define the video file name and path
5. Gather all JPEG images from the image folder
6. Read the first image to determine frame properties (height, width, layers)
7. Define the video encoding format with fourcc
8. Initialize a VideoWriter object with the video name, encoding, fps, and frame size
9. FOR EACH image in the sorted list of images DO
10. Read the image from the file
11. Write the image frame to the video
12. END FOR
13. Copy the created video to a backup file with "_copy" in the name
14. Release all OpenCV windows
15. Release the VideoWriter object
16. Update shared variables to indicate progress, completion, and output video path
17. END FUNCTION

5.2.3 Frame by Frame Video Analysis

Table 23: Algorithm 3- Frame by Frame Analysis

```
1.  FUNCTION begin()
2.      CALL empty  frames  folder()
3.      DEFINE nonlocal variables frame  count, video  frame  count, pedestrian  color, crowd  color,
4.      SET TensorFlow detection graph as default graph
5.      CREATE a TensorFlow session within the detection graph
6.      INITIALIZE frames  left to 100 (percent of frames to process)
7.      INITIALIZE pedestrian  count  frame, crowd  count  frame to 0
8.      INITIALIZE increment  progress  bar to 0
9.      WHILE True
10.         INCREMENT frame  count by 1
11.         IF increment  progress  bar >= 2.28 * video  frames  total / 100
12.             DECREMENT frames  left by 2.28
13.             INCREMENT User.frames  progress by 2
14.             PRINT progress status
15.             RESET increment  progress  bar to 0
16.         READ next frame from video capture object 'cap'
17.         IF frame read unsuccessful
18.             PRINT 'EOF' and BREAK the loop
19.         EXPAND dimensions of frame for TensorFlow processing
20.         DEFINE tensors for image, boxes, scores, classes, num  detections
21.         RUN TensorFlow session to detect objects in frame
22.         DRAW bounding boxes on frame, count detected objects
23.         IF frame  count >= frame  rate
24.             CALCULATE and store max counts of pedestrians and crowds in second
25.             RESET pedestrian  count  frame, crowd  count  frame to 0
26.             RESET frame  count to 0
27.         INCREMENT video  frame  count by 1
28.         SAVE processed frame using save  frame function
29.         INCREMENT increment  progress  bar
30.     SET User.frame  rate to fps
31.     CALL frames  to  video to assemble processed frames into video
32.     RELEASE resources and update necessary flags/variables
33. END FUNCTION
```

5.2.4 Resizing Video Algorithm

Table 24: Algorithm 4- Resizing Video

```
1.  FUNCTION resize_video(input_video_path)
2.      Define the output video path
3.      Open the input video file using cv2.VideoCapture
4.      Retrieve original width, height, and frames per second (fps) from     input video
5.      Set new width and calculate new height maintaining aspect ratio
6.      Create a VideoWriter object with the output path and new dimensions
7.      WHILE there are frames to read from the input video
8.          Read a frame from the input video
9.          IF no frame is read (end of video)
10.             Break the loop
11.         Resize the frame to the new dimensions
12.         Write the resized frame to the output video
13.     Release the VideoCapture and VideoWriter objects
14.     Print a success message
15.     Return the path of the resized video
16. END FUNCTION
```

5.2.5 Object Detection Label Map Processing

Table 25: Algorithm 5- Label Map Processing

```
1.  FUNCTION load_labelmap(path)
2.      Read and parse label map file into a protobuf object
3.      Validate each item in the label map (IDs should be >= 1)
4.      Return the parsed label map
5. FUNCTION convert_label_map_to_categories(label_map, max_num_classes,
use_display_name)
6.      Initialize categories list and list of added IDs
7.      IF label_map is not provided THEN
8.          Generate default categories up to max_num_classes
9.      ELSE
10.         FOR each item in label_map DO
11.             IF item's ID is within the max_num_classes range THEN
12.                 Determine the category name (use display_name if
                            available)
13.                 Add to categories if ID not already added
14.     Return categories list
15. FUNCTION create_category_index(categories)
16.     Initialize category index as a dictionary
17.     FOR each category in categories DO
18.         Add category to index using its ID as the key
19.     Return category index
20. FUNCTION object_detection_label_map_processing(path, max_num_classes,,
use_display_name)
21.     Load label map from file
22.     Convert label map to categories
23.     Create a category index from categories
24.     Return category index
```

5.2.6 Object Detection Visualization

Table 26: Algorithm 6- Object Detection Visualization

1. FUNCTION object_detection_visualization(image, boxes, classes, scores, category_index, additional_params)
2. Initialize a map for box display strings and parameters for processing
3. Filter and process each detected object (box)
4. Calculate the coordinates of the box
5. Determine the class and score of the object
6. Generate display strings based on class, score, and additional parameters (labels, accuracy)
7. Process crowd detection if enabled
8. Identify intersecting boxes and mark them as crowds
9. For each processed object:
10. Determine the appropriate color for the box (pedestrian or crowd)
11. Call convert_and_draw_box to draw the box on the image
12. Return the annotated image along with object counts (pedestrian and crowd)

5.2.7 Crowd Detection Algorithm

Table 27: Algorithm 7- Crowd Detection Algorithm

1. FUNCTION do_boxes_intersect(box1, box2)
2. Calculate the overlap in the x and y coordinates between box1 and box2
3. Return True if there is an overlap in both x and y coordinates (indicating intersection)
4. FUNCTION bounding_box(boxes)
5. Initialize a bounding box based on the first box in the list
6. Expand the bounding box to include any intersecting boxes
7. Return the expanded bounding box
8. FUNCTION check_intersected_boxes(boxes)
9. Initialize the number of crowds detected to 0
10. FOR each box in the array
11. FOR each other box in the array
12. IF the two boxes intersect
13. Create a bounding box that includes both intersecting boxes
14. Remove the intersecting boxes from the array
15. Add the newly created bounding box to the array
16. Increment the crowd count and restart the loop
17. END IF
18. END FOR
19. END FOR
20. Return the modified array and the number of crowds detected

5.3 Standards

While developing our project we adhered to different standards according to tools and technologies we used:

PEP (Python Enhancement Proposals) 8 – Python Style Guide for formatting and layout of python code [15]

TensorFlow Coding practices for the pretrained model [16]

ML Model Development Best Practices [17]

IEEE Std 830-1998 for software requirements specifications [18]

IEEE Std 14764-2006 for SDLC guidelines [19]

5.4 Detailed description of the implementation (coding)

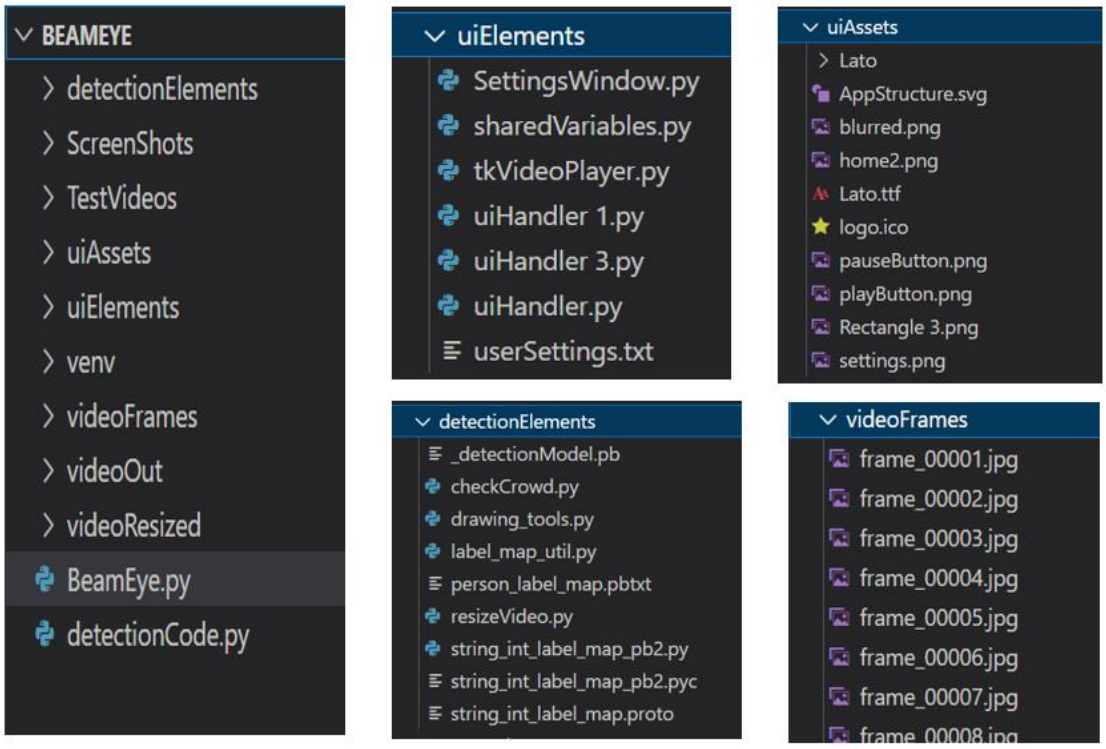


Figure 15: Code Structure in Code Editor

Figure 15 shows the code organization in our editors. It is shown that DetectionElements folder includes the model protocol buffer “detectionModel.pb” that will be called later. The “VideoFrames” folder has the frames extracted from each video uploaded and for a better garbage collection, every time user uploads a video the previous frames are deleted. uiAssests folder contains all related images and icons needed for the project while uiElements folder includes all python .py programs that handle the user interface.

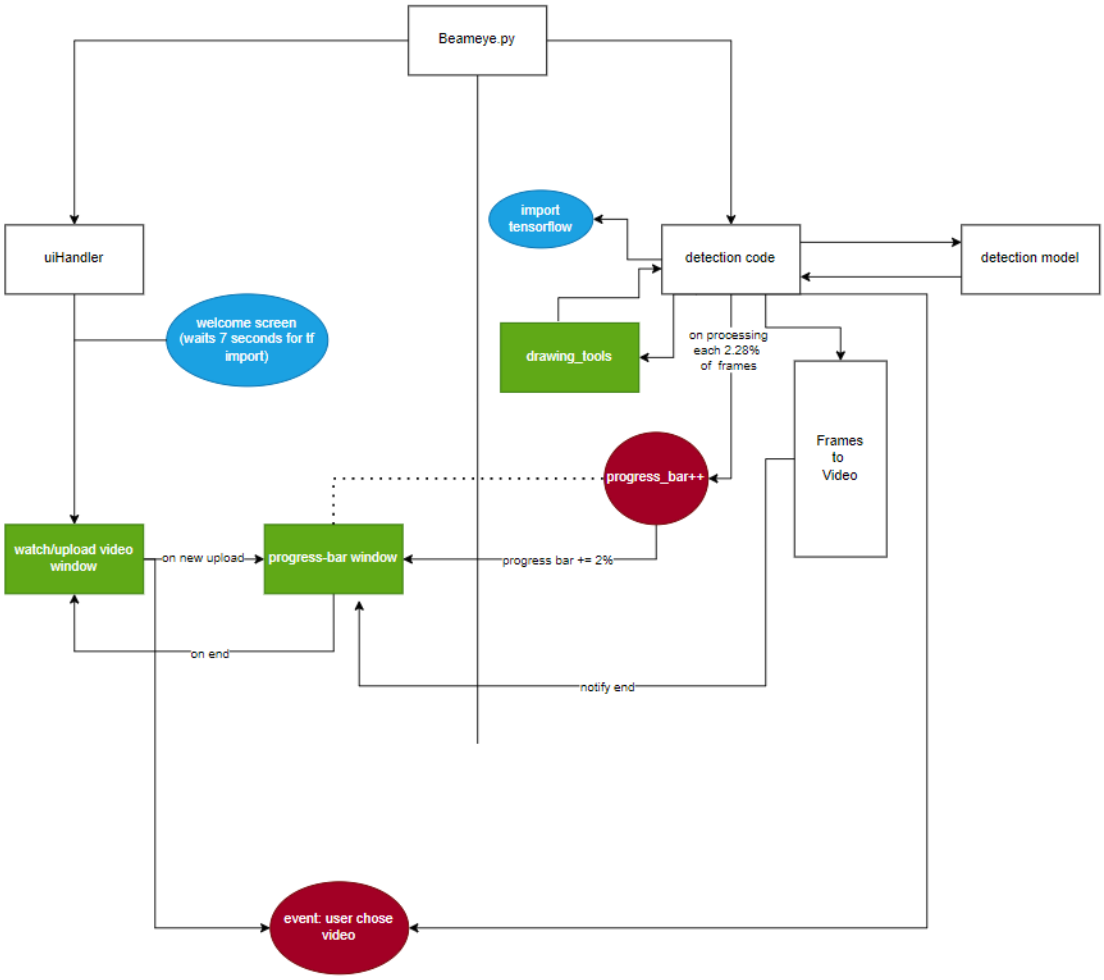


Figure 16: UI Str`ucture Flow Diagram

BeamEye.py calls the uiHandler and detectionCode.py, shows the loading screens straight to upload window, on new upload the progress bar window is shown and checks the progress bar then checks the progress bar in sharedVariables.py if true then detectionCode.py will start and TensorFlow is imported, then each frame is passed to detection model and drawing tools then it goes to frames to be put back into video, while doing so the progress bar is incremented, when done the video notifies the progress bar window to end and goes back to upload. Both

uiHandler.py and detectionCode.py can access sharedVariables.py as a communication line. The uiHanler.py functions writes and the detectionCode.py reads and writes.

Model Reading

"_detectionmodel.pb" is the pretrained model, the. bp is for Google's Protocol Buffer which is a method of serializing structured data, like JSON or XML but more efficient. TensorFlow often uses protobufs to define and exchange data structures.

The .pyc is for python compiled file,

The. proto is for protocol buffers for serializing data

The .pbtxt is for protocol buffers text formats

Now for detectionCode.py

```
17  detection_graph = tf.Graph()
18  with detection_graph.as_default():
19      od_graph_def = tf.compat.v1.GraphDef()
20      tf.compat.v1.disable_v2_behavior()
21      with tf.io.gfile.GFile('detectionElements/_detectionModel.pb', 'rb') as fid:
22          serialized_graph = fid.read()
23          od_graph_def.ParseFromString(serialized_graph)
24          tf.import_graph_def(od_graph_def, name='')
25
26  NUM_CLASSES = 50
27  label_map = label_map_util.load_labelmap('detectionElements/person_label_map.pbtxt')
28  categories = label_map_util.convert_label_map_to_categories(
29      label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
30  category_index = label_map_util.create_category_index(categories)
31
32
33  def save_frame(frame_number, frame):
34      frame_name = "0" * (5 - len(str(frame_number))) + str(frame_number)
35      cv2.imwrite(f"videoFrames//frame_{frame_name}.jpg", frame)
36
37
38  def empty_frames_folder():
39      for frame in os.listdir("videoFrames"):
40          os.remove(f"videoFrames//{frame}")
41
```

Figure 17: Calling the Model

This is where the model is called

1. Create a New TensorFlow Graph

detection_graph = tf.Graph()

This line creates a new TensorFlow graph. A TensorFlow graph is a series of TensorFlow operations arranged into a graph of nodes.

2. Set the New Graph as Default

with detection_graph.as_default():

This line sets detection_graph as the default graph for all operations defined in its context.

Context Manager: creates a context manager, ensuring that all operations defined within the block are added to detection_graph.

3. Create a Graph Definition Object

od_graph_def = tf. compat. v1. GraphDef()

This Initializes a GraphDef object. In TensorFlow, GraphDef is a protocol buffer that represents the model architecture (i.e., the structure of the graph).

4. Disable TensorFlow 2.x Behavior

tf. compat. v1. disable_v2_behavior ()

This Disables TensorFlow 2.x behaviors to ensure compatibility with TensorFlow 1.x code. This is necessary because the following code is written for TensorFlow 1.x, and there are significant differences between TensorFlow 1.x and 2.x.

5. Read and Load the Pre-Trained Model

```
with tf.io.gfile.GFile('detectionElements/_detectionModel.pb', 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
```

Reading the Model File: Opens the binary file detectionElements/_detectionModel.pb (which contains the pre-trained model) for reading ('rb' stands for 'read binary').

The model file is read into serialized_graph and then parsed into the od_graph_def object using ParseFromString. This process converts the binary file into a graph definition.

Finally, the import_graph_def function is used to import the od_graph_def into the current default graph (detection_graph). This step loads the pre-trained model into the TensorFlow graph.

with detectionElements/person_label_map.pbtxt:

6. Loading the Label Map

label_map = label_map_util.load_labelmap('detectionElements/person_label_map.pbtxt')

This line loads a label map from a specified file.

The function load_labelmap from a utility module is called. It reads the label map file located at 'detectionElements/person_label_map.pbtxt'.

A label map in TensorFlow is typically a .pbtxt file that maps integer IDs to category names. In object detection, these categories represent different objects that the model can recognize e.g., 'person' (in our case), 'car'.

7. Converting Label Map to Categories

categories = label_map_util.convert_label_map_to_categories(

label_map, max_num_classes=NUM_CLASSES, use_display_name=True)

Converts the label map to a list of categories.

The convert_label_map_to_categories function takes the loaded label map and processes it into a format used for model evaluation.

NUM_CLASSES is a variable that should be defined elsewhere in your code.

use_display_name=True: Indicates that the function should use the display_name field in the label map as the name of the category. If display_name is not available or use_display_name is set to False, it will use the name field instead.

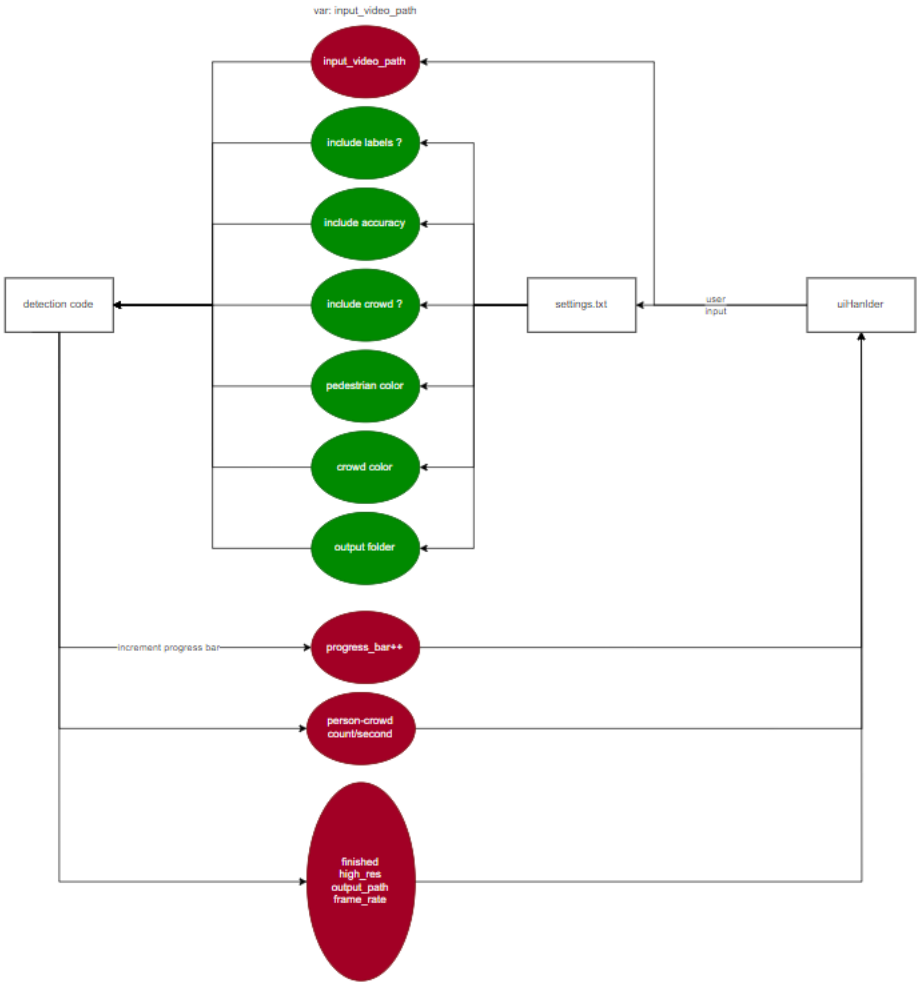


Figure 18: Connect UI Settings with Detection Code

Connecting the UI with detection

Red circles show that both the detection and the uiHandler.py code write and read to settings while the green ones only uiHandler write to settings and detection reads

And shared variables are used by both the uiHandler.py and detectionCode.py

```
def detect():
    import uiElements.sharedVariables as User
    User.finished = False
    while User.wait:
        sleep(2)
        print("waiting")

    else:
        print("Got Video")
        User.wait = True
        from detectionElements.resizeVideo import resize_video
        high_res = User.high_res
        if not high_res:
            resized_video = resize_video(User.input_video_path)
            if resized_video:
                cap = cv2.VideoCapture(resized_video)
                print("resized video")
                User.input_video_path = resized_video
                print(cap)
        else:
            cap = cv2.VideoCapture(User.input_video_path)
            print("didnt resize video")
            # User.input_video_path = None
            print(User.input_video_path)
            print(cap)
```

Figure 19: Detect Function

Code in figure19 Waits for user to upload video and starts detection

```

root = tk.Tk()
root.title("BeamEye")
root.iconbitmap(uiAssets + "logo.ico")
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
# define window size percentage, max is 1 == screen size
resize_ratio = 0.88
# setting window size 75% screen size
width, height = int(screen_width * resize_ratio), int((screen_width * resize_ratio) * 9 / 16)
# keeping 16:9 aspect ratio to match videos' placeholders
root.geometry(f"{int(screen_width * resize_ratio)}x{int((screen_width * resize_ratio) * 9 / 16)}")
# disable resizing
root.resizable(False, False)
root.configure(bg="black")

```

Figure 20: TKinter Container

Code in figue20 shows Tkinter container definition

Figure 21: Tkinter Container Picture

Figure 21 shows the container all elements are added to.

```
12  def settings_inherit_root(root):
13      global window
14      window = root
15
16
17  def open_settings_window(root=window):
18      # settings will only be used in and by this function
19      with open(uiElements + "/userSettings.txt", "r") as f:
20          settings = f.read()
21          settings = [line.split(" ")[-1] for line in settings.split("\n")]
22          include_labels, include_crowd, include_accuracy, pedestrian_color,
            crowd_color, output_path = settings
23          output_path = output_path.replace("_SPACE_", " ")
24          include_labels, include_crowd, include_accuracy, pedestrian_color,
            crowd_color = int(include_labels), int(
25              include_crowd), int(include_accuracy), int(pedestrian_color), int
                (crowd_color)
26
27      color_dict = {1: "#0094FF", 2: "#FF00F6", 3: "red", 4: "#FF6A00", 5: "yellow",
        6: "#26FF5C"}
28      color_dict_key = 1
```

Figure 22: Setting Window Code

Settings window with main colors defined and it inherits from root

Everything is mentioned in Tkinter Documentation [9]

Labels are texts that have certain parameters like font, size, background, and foreground in Tkinter the. place places elements wherever wanted. Tkinter and custom Tkinter have functionalities like CSS.

```
127     # choosing colors
128     # pedestrian box colors
129     def reset_pedestrian_checkboxes():
130         undo_saved()
131         nonlocal pd_color_checkBox_list, pedestrian_color, crowd_color, same_color_error
132         same_color_error.place_forget()
133         pd_color_checkBox_list[pedestrian_color - 1].deselect()
134         if all(not box.get() for box in pd_color_checkBox_list):  # for no empty checkbox
135             pd_color_checkBox_list[pedestrian_color - 1].select()
136 
137         else:
138             for box in pd_color_checkBox_list:
139                 print(box.get(), end=", ")
140                 if box.get():
141                     tmp = pd_color_checkBox_list.index(box) + 1
142                     if tmp == crowd_color:
143                         same_color_error.place(x=20, y=560)
144                         pd_color_checkBox_list[tmp - 1].deselect()
145                         pd_color_checkBox_list[pedestrian_color - 1].select()
146                     else:
147                         pedestrian_color = tmp
148 
```

Figure 23: Coloring Function

This is for choosing colors for pedestrians and crowd

a call to the undo_saved function, which might reverse some prior state.

retrieving non-local variables, such as color variables and checkbox lists.

Use conditional logic to choose and deselect checkboxes in response to specific circumstances.

printing to the console each checkbox's status.

detecting and responding to a situation in which the color of the pedestrians and the crowd match.

All elements in settings are defined through functions like the ones in figure23.

```
268     # output folder
269     def change_output_folder():
270         undo_saved()
271         nonlocal output_path, current_output_text
272         settings_window.withdraw()
273         output_path = filedialog.askdirectory()
274         print("got:", output_path)
275         disp_output = output_path
276         current_output_text.configure(text=f'Current: {disp_output}')
277         settings_window.deiconify()
278 
279     topy -= 20
280     left_x = 0
281     tk.Label(settings_window, text='Output folder', font=top_font, foreground="white",
        background="#071F46").place(x=20,
282                                 y=increment_topy() + 20)
283     display_output = "\\" + output_path.replace("//", "\\").replace('"', "")
284     current_output_text = tk.Label(settings_window, text=f'Current: {display_output}', font=
        (Lato, 10),
285                                    foreground="white", background="#071F46")
286     current_output_text.place(x=20, y=increment_topy())
287     output_folder_change_button = CTk.CTkButton(master=settings_window,
288                                                 width=120,
```

Figure 24: Changing Output Folder

This one to change folder functionality

```
303     # save settings
304     def undo_saved():
305         nonlocal settings_save_button
306         settings_save_button.configure(text_color=settings_bg_color,
307                                        text='Save',
308                                        bg_color=settings_bg_color,
309                                        fg_color="white",
310                                        hover_color="#24EA3F", )
311 
312     def save_settings():
313         nonlocal include_labels, include_crowd, include_accuracy, pedestrian_color, crowd_color,
            output_path, settings_save_button
314         output_path = output_path.replace(" ", "_SPACE_")
315         settings = f"labels {include_labels}\ncrowd {include_crowd}\naccuracy {include_accuracy}
            \npedestrian_color {pedestrian_color}\ncrowd_color {crowd_color}\nout_dir {output_path}"
316         print(settings)
317         with open(uiElements + "/userSettings.txt", "w") as f:
318             f.write(settings)
319         settings_save_button.configure(text='Saved!', fg_color="#24EA3F")
320 
321     settings_save_button = CTk.CTkButton(settings_window,
322                                          height=40,
323                                          width=120,
324                                          border_width=2,
325                                          corner_radius=8,
326                                          border_color="white",
327                                          font=("Lato", 20),
328                                          command=save_settings,
329                                          text_color=settings_bg_color,
```

Figure 25: Saving Settings Functionality

For saving settings functionality

Shared Variables

```
import threading
from time import sleep
import random

input_video_path = None

wait = True

pedestrian_count_second, crowd_count_second = [0], [0]  # [0 for _ in range(30)], [0 for _ in range(30)]

finished = False

high_res = False

frames_progress = 0  # percent

output_video = None

frame_rate = 1
```

Figure 26: SharedVariables.py Content

Figure 26 shows the shared variables that read and written by codeDetecton.py and uiHandler.py.

tkVideoPlayer to handle the video display [9]

The UI handler

```
# if the parent folder isn't GP ==> a sub-folder of GP
while not os.path.isdir(str(parent) + '\\uiAssets\\'):
    # go back to its parent
    parent = parent.parent
GP_path = parent
uiAssets = str(GP_path) + '\\uiAssets\\'
```

Figure 27: Handling Parent Folder for Project

Figure 27 code shows how Tkinter keeps the parent folder as it is other laptops

```python
def open_hello_window():
    global current_canvas, main_root, w, h

    # upload canvas
    img_ = Image.open(uiAssets + 'home2.png')
    resized_image_ = img_.resize((root.winfo_width(), root.winfo_height()))
    tk_image_ = ImageTk.PhotoImage(resized_image_)
    background_image_hello = tk_image_
    hello_canvas = tk.Canvas(root, width=root.winfo_width() - 4, height=root.winfo_width() - 10)
    current_canvas = hello_canvas
    hello_canvas.place(x=0, y=0)
    hello_canvas.create_image(root.winfo_width() / 2, root.winfo_height() / 2,
    image=background_image_hello, anchor="c")

    # settings in upload window

    progressbar_placeholder = ctk.CTkProgressBar(master=hello_canvas, height=20,
                                                 width=400, bg_color="#041632",
                                                 fg_color="#041632",
                                                 progress_color="#30A8E6", border_color="#30A8E6",
                                                 border_width=2, indeterminate_speed=0.01,
                                                 mode='determinate'
                                                 )
    progressbar_placeholder.place(x=root.winfo_width() / 2 - 200, y=root.winfo_height() / 2 + 60)
    progressbar_placeholder.set(0)
```

Figure 28: Main Window Code

Code in figure 28 is for handling the main window.

```python
    # settings canvas

    def wait_for_tenserflow_import():
        sleep(1)
        for _ in range(7):  # takes around 7 seconds to import tensorflow
            for __ in range(7):  # each .step() increases the bar by 2%, 7x7x2 = 98% of the bar
            after 7 seconds
                progressbar_placeholder.step()
            sleep(1 / 7)
        progressbar_placeholder.set(1)  # set the bar to 100%
        sleep(1)
        hello_canvas.destroy()

        return

    threading.Thread(target=wait_for_tenserflow_import).start()
```

Figure 29: Waiting Bar Logic

Code in figure 29 shows the waiting bar logic

```
159  def open_video_window():
160      global root, current_video_canvas
161
162      # threading.Thread(target=open_hello_window).start()
163      video_canvas = tk.Canvas(root, bg="#051735")
164      settings_inherit_root(root)
165      img = Image.open(uiAssets + 'blurred.png')
166      resized_image = img.resize((root.winfo_width(), root.winfo_height()))
167      tk_image = ImageTk.PhotoImage(resized_image)
168      background_image_loading = tk_image
169
170      def open_load_window():
171
172          global progressbar, progressbar_progress, \
173              progressbar_placeholder_label, current_loading_canvas, \
174              current_video_canvas, ended, input_video_path
175          bg_color = "#031532"
176          loading_canvas = ctk.CTkCanvas(root, width=root.winfo_width() - 4, height=root.
             winfo_height() - 4,
177                                         bg=bg_color)  # , bg="#031532")
178          loading_canvas.place(x=0, y=0)
179
180          current_loading_canvas = loading_canvas
181          loading_canvas.create_image(0, 0, image=background_image_loading, anchor="nw", )
182          progressbar = ctk.CTkProgressBar(master=loading_canvas, height=int(20 * root.winfo_height
             () / 750),
183                                           width=int(400 * root.winfo_width() / 1300),
                                              bg_color="#3C3E46",
184                                           fg_color="#4A4C51",  # bg_color: for corner
185                                           # edges, fg_color inside the bar (inactive part)
```

Figure 30: Second Page Code

This is to open the second page where user can see the video display and buttons for actions

```
363     def load_video(video: str, video_player: TkinterVideo):
364
365         nonlocal current_pd_number, current_crowd_number, \
366             current_crowd_number_off, max_people_number, \
367             max_crowd_number, max_crowd_number_off
368         import uiElements.sharedVariables as User
369
370         color_dict = {1: "#0094FF", 2: "#FF00F6", 3: "red", 4: "#FF6A00", 5: "yellow",
371                       6: "#26FF5C"}  # {1: "blue", 2: "purple", 3: "red", 4: "orange", 5:
                          "yellow", 6: "green"}
372         with open(str(GP_path) + "\\uiElements\\userSettings.txt", "r") as f:
373             settings = f.read()
374             settings = [line.split(" ")[-1] for line in settings.split("\n")]
375             include_labels, include_crowd, include_accuracy, pedestrian_color, crowd_color,
                output_path = settings
376             include_labels, include_crowd, include_accuracy, pedestrian_color, crowd_color = int
                (include_labels), int(
377                 include_crowd), int(include_accuracy), int(pedestrian_color), int(crowd_color)
378
379             global crowd_is_included
380             crowd_is_included = include_crowd
381
382             if bool(include_crowd):
383                 current_crowd_number_off.place_forget()
384                 max_crowd_number_off.place_forget()
385
386                 current_crowd_number.place(x=480 / 1300 * video_canvas.winfo_width(),
387                                            y=600 / 750 * video_canvas.winfo_height())
388                 current_crowd_number.configure(text_color=color_dict[crowd_color])
389
```

Figure 31: Loading Video Function

Code in figure 31 shows the loading video functionality and logic

```
431     def resize_video_canvas():
432         video_canvas.config(width=root.winfo_width(), height=root.winfo_height())
433 
434     Lato = "Lato"
435 
436     video_title_label = tk.Label(root, text='Video Title', font=("Lato", int(20 / 750 * root.
        winfo_width())),
437                                  foreground="white", background="#051736")
438     current_timestamp = ctk.CTkLabel(root, text="00:00:00", font=("Lato", int(25 / 750 * root.
        winfo_width())),
439                                      fg_color="#051635", bg_color="#051635",
440                                      corner_radius=8, text_color="white")
441     current_timestamp.place(x=105, y=472)
442 
443     pil_image2 = Image.open(uiAssets + 'settings.png')
444     settings_button_image = ImageTk.PhotoImage(pil_image2)
445 
446     label_a = ctk.CTkLabel(root, text="Max # of People/sec: ", font=("Lato", int(20 / 750 * root.
        winfo_width())),
447                            fg_color="transparent",
448                            bg_color="#051635", text_color="white",
449                            corner_radius=8)  # .place(x=225, y=550)
450     current_pd_number = ctk.CTkLabel(root, text="", font=("Lato", int(30 / 750 * root.winfo_width
        ()), "bold"),
451                                      fg_color="transparent",
452                                      bg_color="#051635", corner_radius=8,
453                                      text_color="#30A8E6")
```

Figure 32: Resize Video Code

Code in figure 32 is to handle the resizing of videos.

```
537     def save_video_func():
538
539         def video_saved_feedback():
540             import uiElements.sharedVariables as User
541             save_to = filedialog.askdirectory()
542             if not save_to:
543                 return
544             try:
545                 move(User.output_video[:-4] + "_copy.avi", save_to)
546             except shutil.Error:
547                 pass
548
549             nonlocal save_video_button
550             save_video_button.configure(text="Saved in Specified Folder!", fg_color="#36BC27",
                hover_color="#36BC27",
551                                         text_color="#071B3D")
552             sleep(2)
553             save_video_button.configure(text='Save Video   ' + "\U0001F4E5", fg_color="#30A8E6",
                hover_color="#061C42",
554                                         text_color="white")
555             return
556
557         threading.Thread(target=video_saved_feedback).start()
558
559     save_video_button = ctk.CTkButton(root,
560                                       height=60,
561                                       width=333,
562                                       border_width=2,
563                                       corner_radius=8,
564                                       border_color="#30A8E6"
```

Figure 33: Saving Video Functionality

Code in figure 33 is to save video functionality.

```
574     def screenshot():
575
576         import uiElements.sharedVariables as User
577
578         def screenshot_saved_feedback():
579             nonlocal screenshot_button
580             screenshot_button.configure(text="Saved in ScreenShots/", fg_color="#36BC27",
581                                         hover_color="#36BC27", text_color="#071B3D")
582             sleep(2)
583             screenshot_button.configure(text='ScreenShot    ' + "\U0001F4F7", fg_color="#30A8E6",
                hover_color="#061C42",
584                                         text_color="white")
585             return
586
587         video_path = User.output_video
588         cap = cv2.VideoCapture(video_path)
589         frame_number_to_save = vid_player2.current_frame_number()
590         cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number_to_save - 1)
591         ret, frame = cap.read()
592         if ret:
593             output_folder = str(GP_path) + '\\ScreenShots\\'
594             frame_filename = f'ScreenShot_{frame_number_to_save}.jpg'
595             output_path = output_folder + frame_filename
596
597             cv2.imwrite(output_path, frame)
598
599             threading.Thread(target=screenshot_saved_feedback).start()
600             print(f"Saved frame {frame_number_to_save} as {frame_filename} in {output_folder}")
601         else:
```

Figure 34: Screenshot Code

Code in figure 34 is to handle screenshot functionality

```
291     def upload_button_func():
292
293         global input_video_path, new_video, thread_people, thread_crowd, threads_started
294         input_video_path = filedialog.askopenfilename(initialdir=str(Path(__file__).resolve().parent.parent),
295                                                       filetypes=[("Videos", "*.mp4")])
296         if not input_video_path:
297             return
298
299         nonlocal video_title_label
300         new_video = True
301         sleep(0.002)
302         if (thread_people is not None and thread_crowd is not None) and threads_started:
303             thread_people.join()
304             thread_crowd.join()
305             threads_started = False
306
307         video_title = input_video_path.split("/")[-1]
308         video_title_label.configure(text=video_title)
309         video_title_label.place(x=45, y=30)
310         nonlocal play_pause_button
311
312         play_pause_button.place(x=60 * root.winfo_width() / 1300, y=470 * root.winfo_height() / 750 + 10)
313         if input_video_path:
314             import uiElements.sharedVariables as User
315             User.input_video_path = input_video_path
316             User.wait = False
317             sleep(0.6)
318
319             open_load_window()
```

Figure 35: Upload Video Code

Code in figure 35 is to handle uploading video functionality.

```
711     def update_people_label():
712
713         nonlocal current_pd_number, vid_player
714         import uiElements.sharedVariables as User
715         tmp_stamp = 0
716         while True:
717
718             current_duration = vid_player.current_duration()
719             stamp = int(current_duration)
720             if tmp_stamp == stamp:
721                 pass
722             else:
723                 print("updating people")
724
725                 current_pd_number.configure(text=User.pedestrian_count_second[stamp]
726                                             )
727                 tmp_stamp = stamp
728             sleep(.001)
729             if new_video:
730                 return
731
732     def update_crowd_label():
733         nonlocal current_crowd_number, vid_player
734         import uiElements.sharedVariables as User
735         tmp_stamp = 0
736         while True:
737
738             current_duration = vid_player.current_duration()
739             stamp = int(current_duration)
740
```

Figure 36: Labels and Crowd Inclusion Handling

Code in figure36 is to handle Updating functions for inclusion of labels, crowd detection and other functionalities

```
BeamEye.py > ...
        Click here to ask Blackbox to help you code faster
 1    import threading
 2
 3
 4    def startUI():
 5        import uiElements.uiHandler
 6
 7
 8    threading.Thread(target=startUI).start()
 9    import detectionCode
10
```

Figure 37: Running App Code

Figure 37 shows the app running code. A thread is imported and the uiHander.py along with the detectionCode.py are called and the program initializes.

5.5 AI Detection Model Architecture

The model we used utilizes SSD Architecture which is a single convolution network that learns to predict bounding box locations and classify these locations in one pass. This network consists of base architecture and in our case, it is the MobileNet followed by several convolution layers

As shown in figure 38:

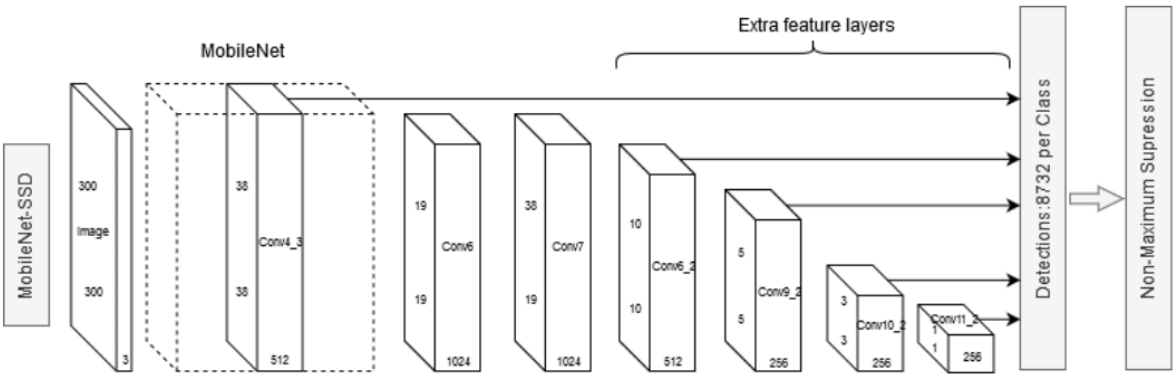


Figure 38: Single Shot Multiplex Detector MobileNet model Architecture

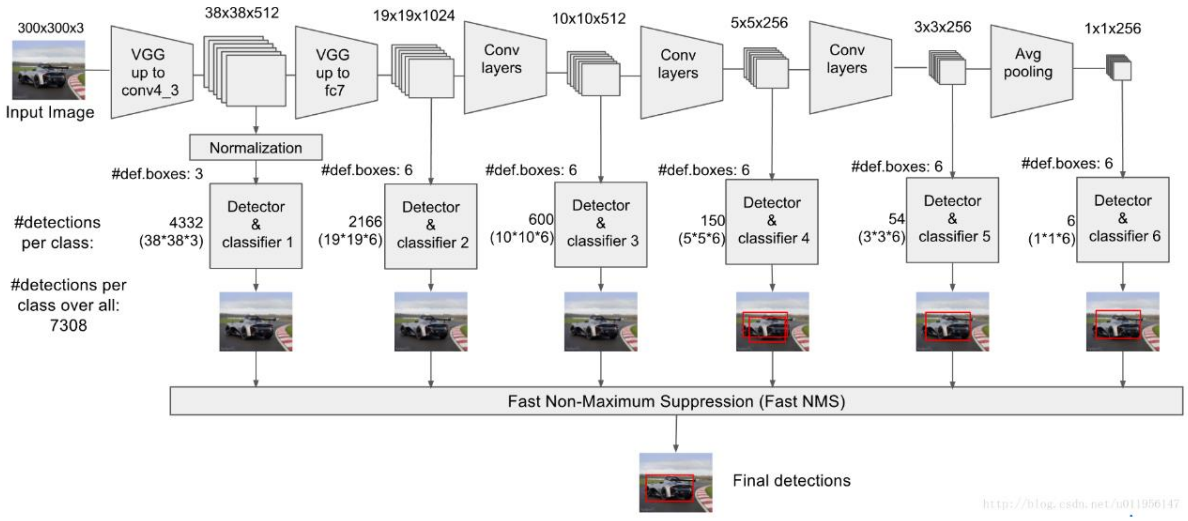


Figure 39: More Detailed SSD Architecture with VGG16 for Clarification

"In order to detect multiple objects within image, a one single shot is taken using SSD unlike RPN (regional proposal network) based approaches like R-CNN which need two shots one for generating region proposals and one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches." [7]

Additionally, the SSD approach is based on a feed-forward convolutional network that produces fixed-size collection of bounding boxes and scores for the presence of object class instance in those boxes. The first network layer (base network) utilizes MobileNet network as a base which is truncated before any classification layers. Then feature layers are added to the end of this truncated base network. These layers decrease in size progressively and allow predictions of detections at multiple scales and each convolutional model for predicting detection is different for each feature layer. [7]

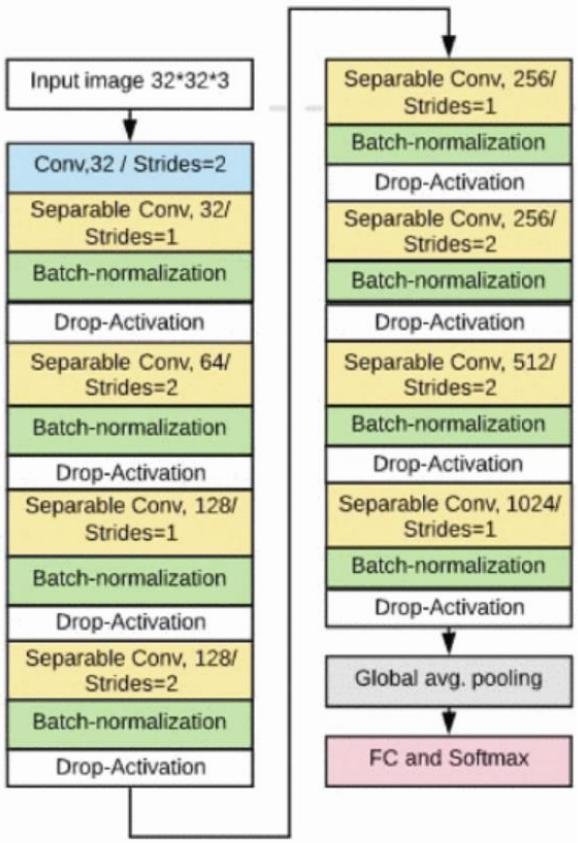


Figure 40: MobileNet Architecture Diagram

As shown in figure 40, this is the MobileNet architecture which is used in our model as a base neural network. Combining the SSD with Mobile net makes the detection system lightweight (has fewer parameters and requires less computational power to run), fast (processes image quickly) and accurate (performs well in various object detection tasks). While traditional convolutional layers use filters to perform both spatial filtering and combination of features simultaneously which requires high number of computations, MobileNet breaks the process into two layers: Depth-wise convolution and Point-wise convolution. The first layer is a depth-wise convolution where a single filter is applied to each input channel, so if the input image has 32 channels, there will be 32 filters each processing one channel. This handles spatial filtering which is the detection of features like edges, textures, and colors within an image. The second Layer, which is the point-wise convolution, takes the output of a depth-wise convolution and applies 1x1 convolution which combines the features from different channels. It is a way to create a new set of features by missing the existing ones.

Since Model taken integrates both MobileNet and SSD, the model achieves an efficient and effective object detection system capable of detecting multiple objects in many sizes and can be used for real-time object detection. See and illustration for MobileNet Architecture in figure 41:

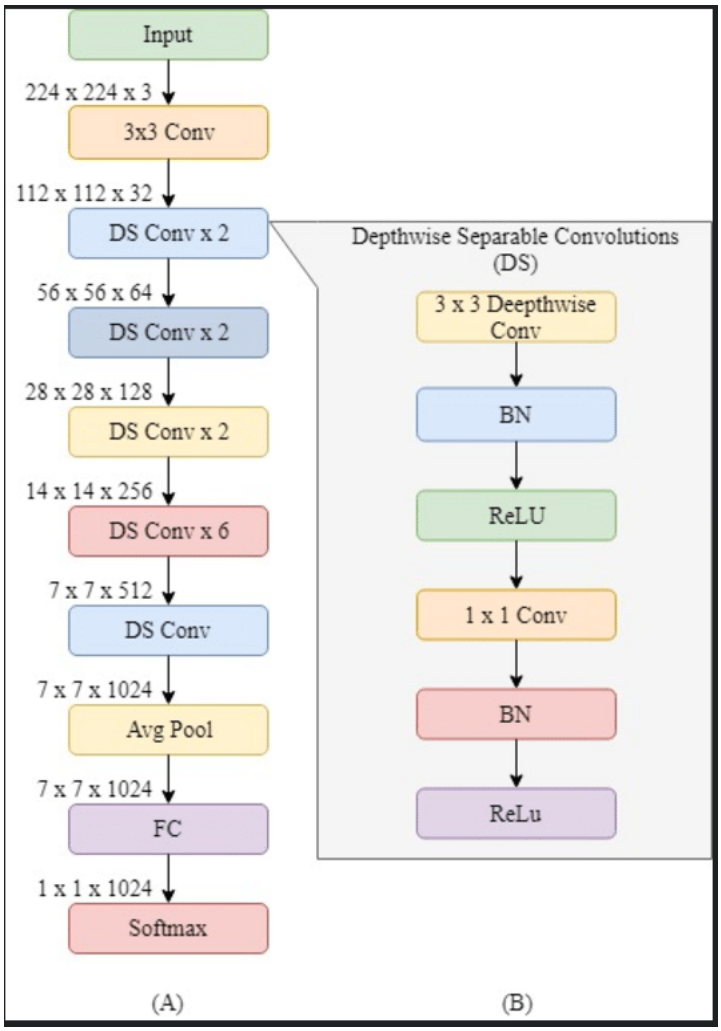


Figure 41: Illustration of MobileNet Architecture

In our project, we used the pretrained model from TensorFlow Object Detection API framework which is referred to as Model Zoo which includes collection of pretrained models trained on the COCO (common object in context) dataset. [6]

COCO-trained models

Model name	Speed (ms)	COCO mAP[1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes

Figure 42: TensorFlow Pretrained Models on COCO Dataset

The way these models are trained is as follows:

First the dataset is prepared, in this case the COCO data set, and preprocessed to organize images and corresponding annotations into a compatible format with training pipeline. This includes converting the annotations into tensors representing bounding boxes and class labels

Second the model is trained and configured. This can be done by separating the dataset into training and testing data set.

Third an appropriate loss function for detection is used, usually a combination of localization loss (like Smooth L1) for bounding box predictions and a classification loss (like cross entropy) for class predictions. The loss is calculated during training when model outputs are compared with main annotations and model weights are updated to minimize the loss. Then hyperparameters like learning rate, batch size and number of epochs are selected.

After that, the model is evaluated using the metric shown in figure 42 (mAP) which means Average Precision. The training process involves iterations of training and validation phases where model performance is periodically checked on a validation set not used during training.[6]

Comparing our chosen model architecture to other architectures:

Table 28: Comparison of Used model's Architecture to Others

Feature	SSD MobileNet	SSD with VGG-16	YOLO (Various Versions)
Accuracy (mPA)	Moderate	High	Varies (Early versions lower, newer versions higher)
Speed (FPS)	High (Real-time on mobile devices possible)	Moderate (Real-time on powerful devices possible)	Very High (Optimized for real-time processing)
Model Size	Small (Fewer parameters due to depth-wise separable convolutions)	Large (More parameters due to fully connected layers)	Varies (Depends on the version)
Computational Efficiency	High (Designed for mobile and edge devices)	Lower (Requires more computational resources)	High (Optimized for efficiency)
Use Case	Real-time applications on resource-constrained devices	High-accuracy applications where model complexity is less of a concern	Real-time video processing and applications requiring very fast detection
Typical Applications	Embedded systems, mobile apps, IoT devices	High-end surveillance systems, cloud-based applications	Surveillance with real-time feedback, autonomous vehicles

Results of chosen Model:

As figure 43 shows, the combination of SSD with MobileNet can give accurate object detection for various scenarios. In our project, we only chose one object class, Person, and added Crowd detection as shown throughout the project report.[7]

Figure 43: Output from SSD MobileNet Object Detection Model

Additionally, it is important to draw the attention to the fact that one of the major limitations of this model is the low accuracy in complex environments or reduced performance in varying

lighting conditions as it also takes time to process images when integrated to desktop applications. In case model's performance is improved, it will provide better results. This can be achieved through hyperparameter tuning where model is experimented with different hyperparameters to find most effective configurations for desktop applications and incorporating more sophisticated data augmentation techniques will improve the model's robustness to real-world variations.

6. QUALITY AND TESTING

6.1 Quality Assurance Activities During Project Life Cycle

To ensure the quality of our project and follow up on it during the SDLC, we have used some project management tools that helped us and made our work easier when implementing.

6.1.1 Affinity Diagram

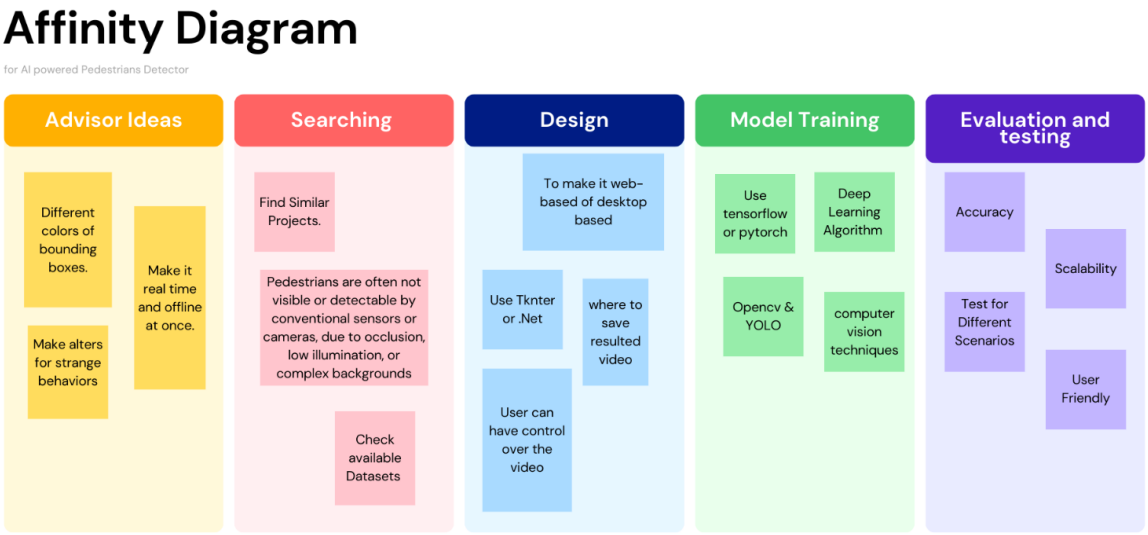


Figure 44: Affinity Diagram

We used Affinity diagram to help us organize our ideas while brainstorming and simplify our work. We mostly used the planning stage of the project to generate ideas for each stage of the system.

6.1.2 Process Map

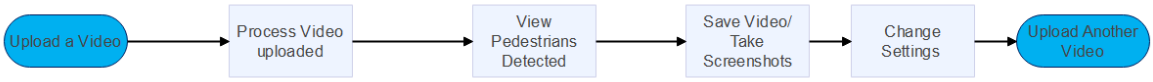


Figure 45: Process Map

The process map tool was used to illustrate the workflow of the general processes of the entire system to help us while testing to check at which stage, we are. Additionally, mapping out the process helped in identifying risks at various stages of development.

6.1.3 SWOT Analysis

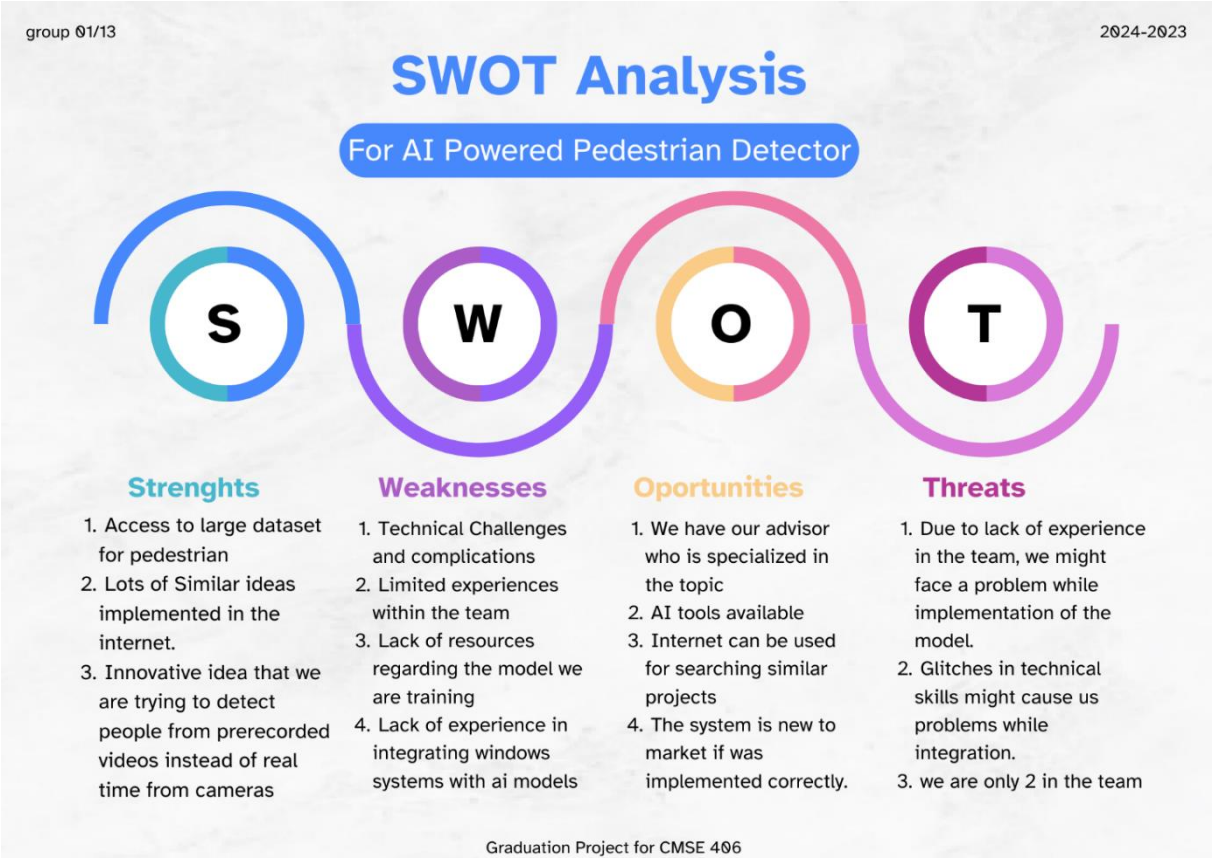


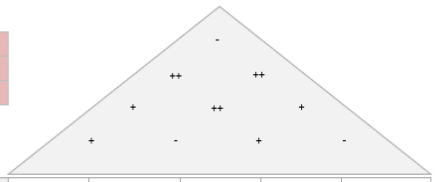
Figure 46: SWOT Analysis

We applied SWOT analysis to analyze our strengths, weaknesses, opportunities, and threads. This tool offered us a comprehensive way to evaluate both internal and external factors that can impact the project's success.

6.1.4 Quality Funcion Deployment

Quality Function Deployment

Project Title:	**AI-Powered Pedestrians Detector**
Project Leader:	**Khawlah Al-shubati**
Date:	**December 3rd 2023**



Correlation:

+	.	-
Positive	No correlation	Negative

Relationships:

9	3	1	
Strong	Moderate	Weak	None

	Desired direction of improvement (↑,0,↓)	↑	↑	↑	↑	↑					
1: low, 5: high	Functional Requirements (How's) →							Competitive evaluation (1: low, 5: high)			
Customer importance rating	Customer Requirements - (What's) ↓	Advanced Detection Algorithm	Video Processing Techniques	Intuitive User Interfaces	Model Accuracy	Preferences Settings	Weighted Score	Satisfaction rating	Competitor rating 1	Competitor rating 2	Competitor rating 3
5	Upload Video		9				45	4	4	3	1
5	View Detected People	9	9		3	1	110	3	4	2	3
3	Fast Response Time		9				27	1	4	1	4
2	Download Video		3			1	8	3	4	5	3
3	Change Labeling Colors	3		9	3		45	4	5	5	5
4	Take Screenshot			3		1	16	3	1	2	3
4	Change Settings		3	9		9	84	1	1	2	5
4	Detect Groups	9			3		48	1	3	1	2
5	Count People	9			9		90	1	1	3	4
	Technical importance score	135	135	75	81	47	473				
	Importance %	29%	29%	16%	17%	10%	100%				
	Priorities rank	1	1	4	3	5					

Figure 47: Quality Function Deployment

We used this approach to define our projects requirements and prioritize features that we should focus on. After doing the calculations needed, it is shown that having an advanced detection algorithm and video processing techniques has the priority over others.

6.1.5 Kano Model

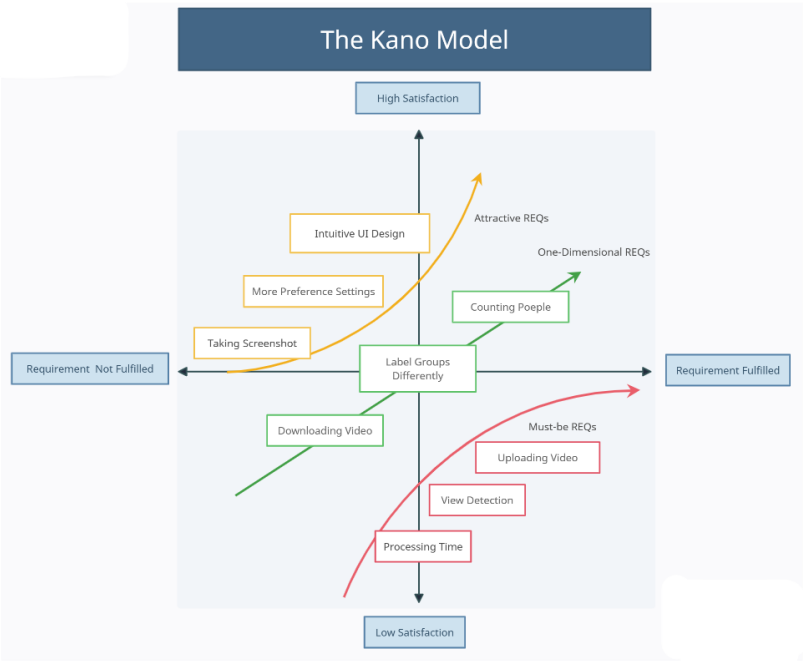


Figure 48: Kano Model

This tool was used to understand and categorize customer needs where it aided us in feature prioritization, enhancing customer satisfaction and guides resource allocation. It is shown that processing time and viewing detection along with uploading a video are a must-be requirements that we should focus on more while having an intuitive user interface and more preference settings are attractive requirements.

6.1.6 Quality Matric Table

Table 29: Quality Matric

No.	Quality Metric Name	Description
1	Detection Accuracy	Measures the percentage of correctly identified pedestrians versus total pedestrians in the test set. High accuracy is crucial for the reliability of the system.
2	Detection Speed	Evaluates the time taken by the SSD MobileNet model to detect pedestrians per frame. Faster detection is essential for real-time applications.
3	False Positive Rate	The frequency at which the system incorrectly identifies a pedestrian. Lower rates are desirable to minimize false alarms.
4	System Responsiveness	Measures the time taken for the Tkinter interface to respond to user inputs. Quick responsiveness enhances user experience.
5	Interface Usability	Assesses the ease with which users can navigate and utilize the Tkinter interface, including clarity of instructions and intuitiveness of controls.
6	Resource Utilization	Evaluates the number of computational resources (CPU, GPU, memory) used by the system. Efficient use of resources is important for the system's sustainability.
7	Adaptability to Different Conditions	Measures the model's accuracy and performance under varying environmental conditions, such as different lighting or weather conditions.
8	Integration Efficiency	Assesses the effectiveness of integrating the SSD MobileNet model with the Tkinter UI, in terms of data flow, error handling, and overall system stability.
9	Scalability	The ability of the system to handle increasing numbers of users or to be easily upgraded with new features or performance improvements.
10	System Reliability	Measures the overall reliability of the system, including uptime and the absence of critical failures during operation.
11	User Satisfaction	This metric focuses on the overall satisfaction with the system's performance and user interface.

12	Maintenance and Support Ease	Assesses how easily the system can be maintained and updated, including the ease of fixing bugs and updating the SSD MobileNet model or the Tkinter interface.

6.1.7: Check Quality Metrics

Table 30: Check Quality Metric

Check	Issue/Topic
√	Test accuracy of pedestrian detection under various scenarios.
√	Evaluate the speed of pedestrian detection per frame for real-time capability.
√	Analyze the rate of false positives and negatives in detection.
√	Check responsiveness of the Tkinter user interface to user interactions.
√	Assess usability and navigation ease of the Tkinter interface.
√	Monitor CPU, GPU, and memory usage to assess resource efficiency.
√	Test model performance in different environmental conditions (light, weather).
√	Verify integration efficiency and data flow between SSD MobileNet and Tkinter UI.
√	Conduct scalability tests for increased load or feature upgrades.
√	Monitor system reliability and uptime.
√	Send satisfaction forms to customers for feedback on system performance and interface.
√	Review maintenance, update, and support effectiveness for the system.
√	Ensure compliance with relevant standards and regulations; update documentation as needed.
√	Assess system for security vulnerabilities and adherence to data privacy standards.
√	Verify backup and recovery procedures for system data and operation continuity.
√	Run search/insert/update queries of varying complexity to calculate Average Response Time.

6.1.8 Quality audit Checklist

Table 31: Quality Audit Checklist

Audit Date	Planned/ Unplanned Audits	Number of Compliant Issues	Number of Actions Taken on non-Compliant Issues	Number of High Priority Risk Items in Project List Risk	Hours to Prepare and Apply the Audit	Auditor
2023-12-05	Planned	2	3	1	1	KH
2023-12-10	Unplanned	3	4	2	3	KH
2023-12-15	Planned	1	2	0	2	KH
2023-12-18	Planned	0	0	0	1	KH
2023-12-22	Unplanned	4	5	3	5	KH

6.1.9 Quality Process Analysis

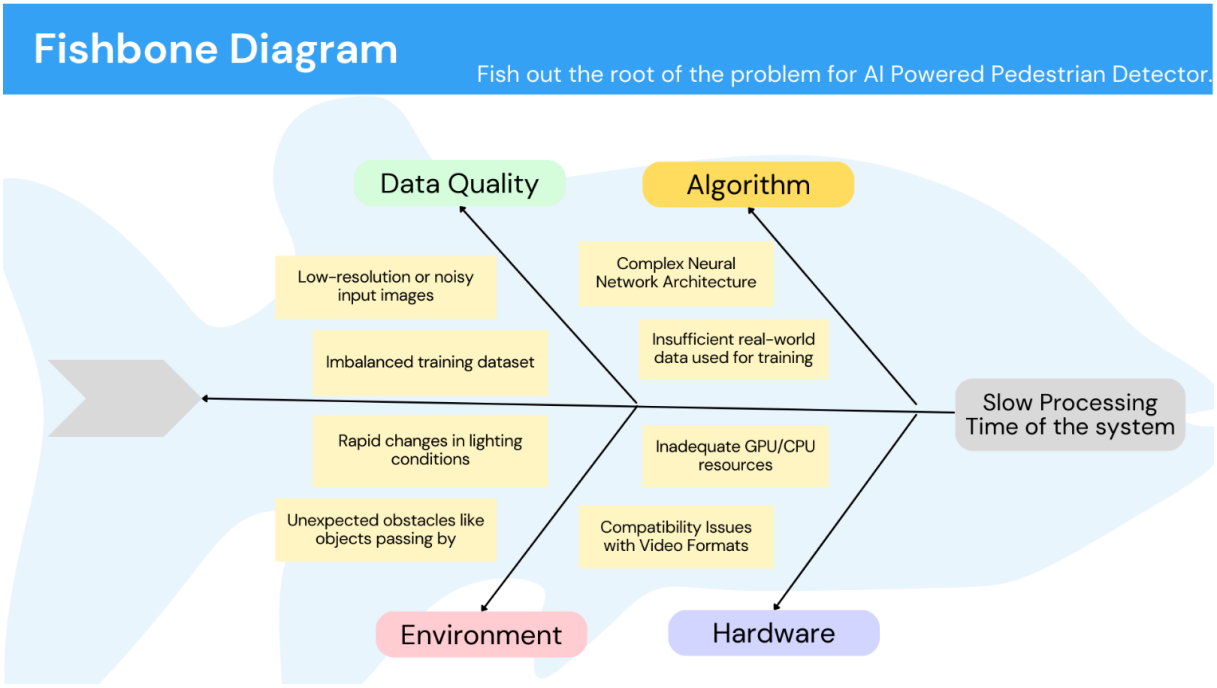


Figure 49: Fishbone Diagram

Fishbone Diagram was made to understand the causes of slow processing time of the system and fix that issue. Such issues can arise due to multiple factors which are hardware, environment, algorithm, and data quality. Under each of these factors there are multiple causes which we investigated to solve this issue.

6.1.10 Statistical Process Analysis

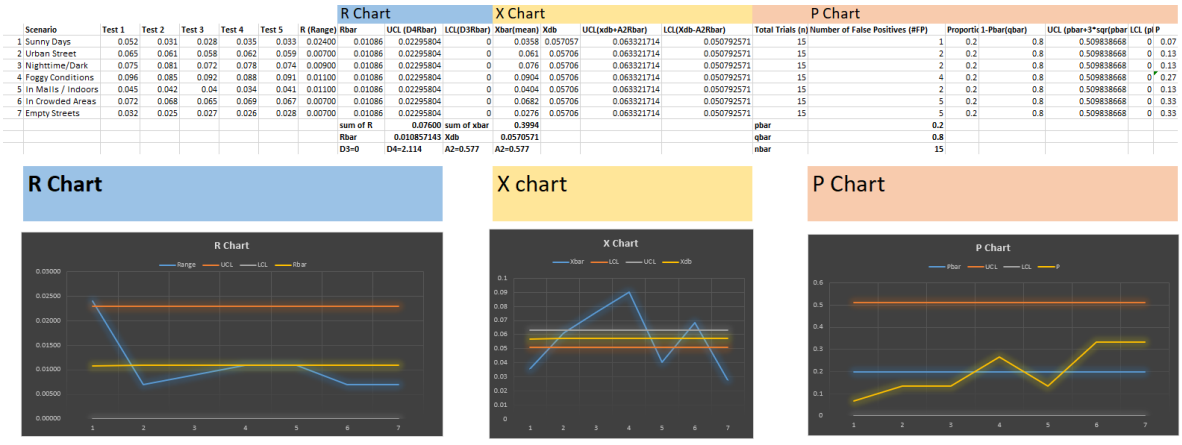


Figure 50: Control Charts

As shown in the figure, control charts of type R, X and P are shown for statistical process control to monitor various aspects of a process. The Range chart (R) shows that the process of detecting people in different scenarios is out of control, the same as the Mean chart (X) that represents that this process is also out of control. And for the Proportion chart (P), is for the process of checking how many False Positive cases would occur.

6.1.11 Process Failure Mode and Effect Analysis

Risk Score (RS) = Probability * Impact

Risk Probability Number (RPN) = Probability * Impact* Detection

Table 32: PFMEA Table

Risk ID	Risk Description	Probability	Impact	Detection	RS	RPN
1	Inaccuracy in pedestrian detection in groups	6	7	4	42	168
2	System latency leading to delayed pedestrian detection	5	8	5	40	200
3	False positives in crowded environments	4	5	6	20	120
4	UI unresponsiveness under high system load	3	6	7	18	126
5	Inadequate system scalability for future upgrades	2	7	5	14	70
6	Integration issues between SSD MobileNet model and Tkinter	4	6	8	24	192
7	Reliability of the system	5	9	3	45	135
8	System failure due to hardware limitations	3	8	6	24	144
9	Inaccurate detection due to environmental factors	6	7	4	42	168

6.1.12 Pareto Analysis

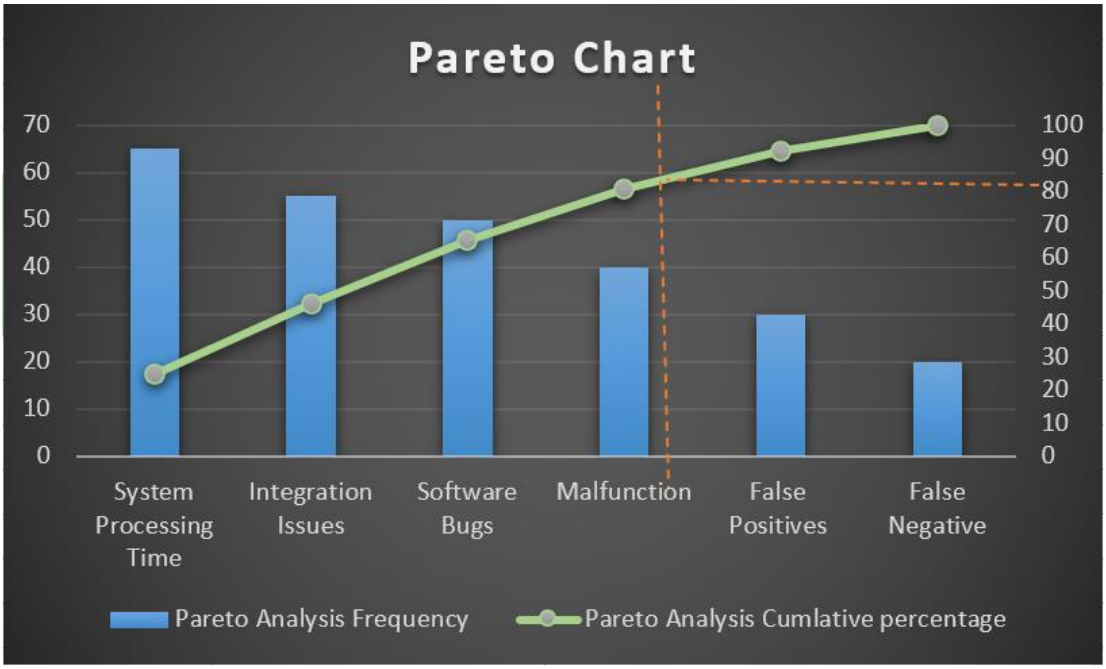


Figure 51: Pareto Chart

As the figure shows 80 percent of software problems come from system processing time, integration issues, software bugs and Malfunction which in development we should focus on more. The other 20 percent are for false positives and false negatives which are part of the model that is out of our control to fix.

6.2 Quality Control (QC) Activities After Implementation Completed

Table 33: TC01-Test Case 01

Test Case ID	**TC01**
Module Name	Video Upload Functionality
Test Case Scenario	Verify the functionality of uploading a video to the system
Test Case Name	Test Video Uploading
Pre-requisite	System is running
Test Data	Video File in mp4 format
Test Steps	1. Launched the system 2. Verify that upload button is available 3. Click on the button 4. Check if a file dialog opens 5. Select valid video file and check for appropriate error messages 6. Repeat with different video file formats
Expected Results	1. App launches successfully 2. Upload Video button is present and functional 3. File dialog opens for video selection 4. Compatible video is uploaded without errors

	5. Video is being processed for detection
Actual Results	The system had the button and video uploads correctly
Status	Pass
Comments	Video loading takes time and longer videos are not tested
Test Executed By	Omar

Before Uploading Video

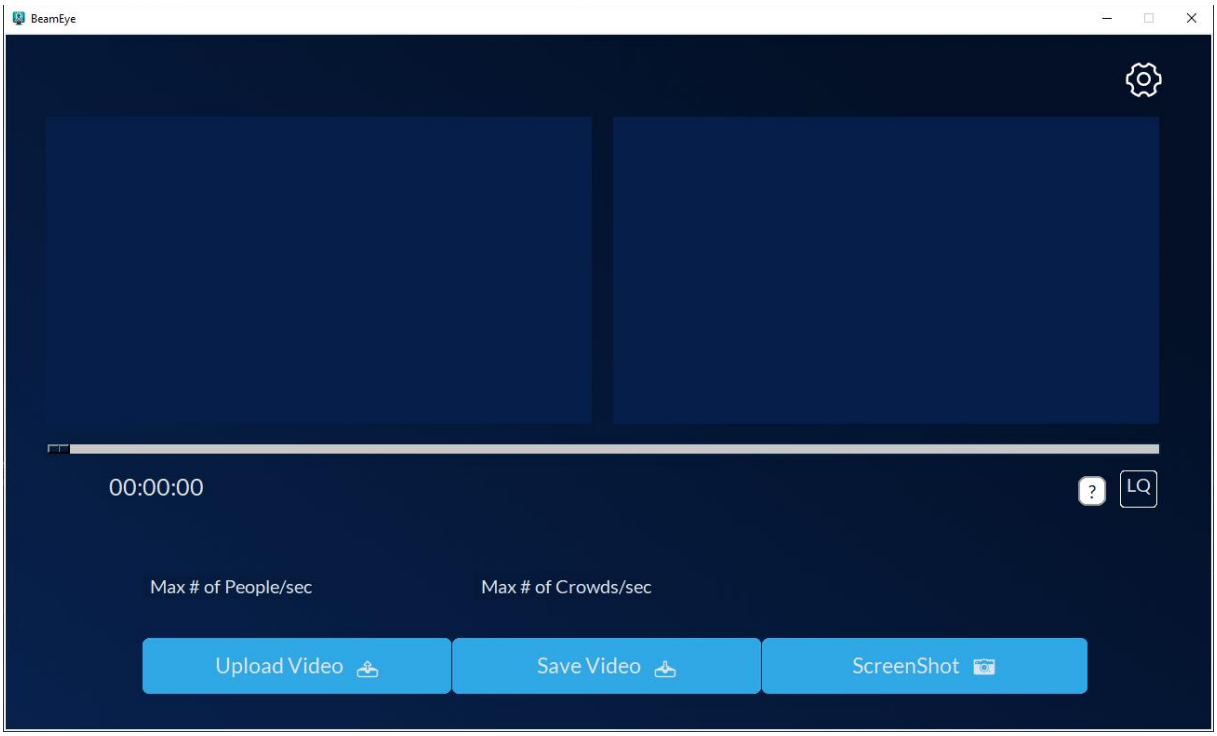


Figure 52: TC01- Screen Before

User is prompted to choose video file

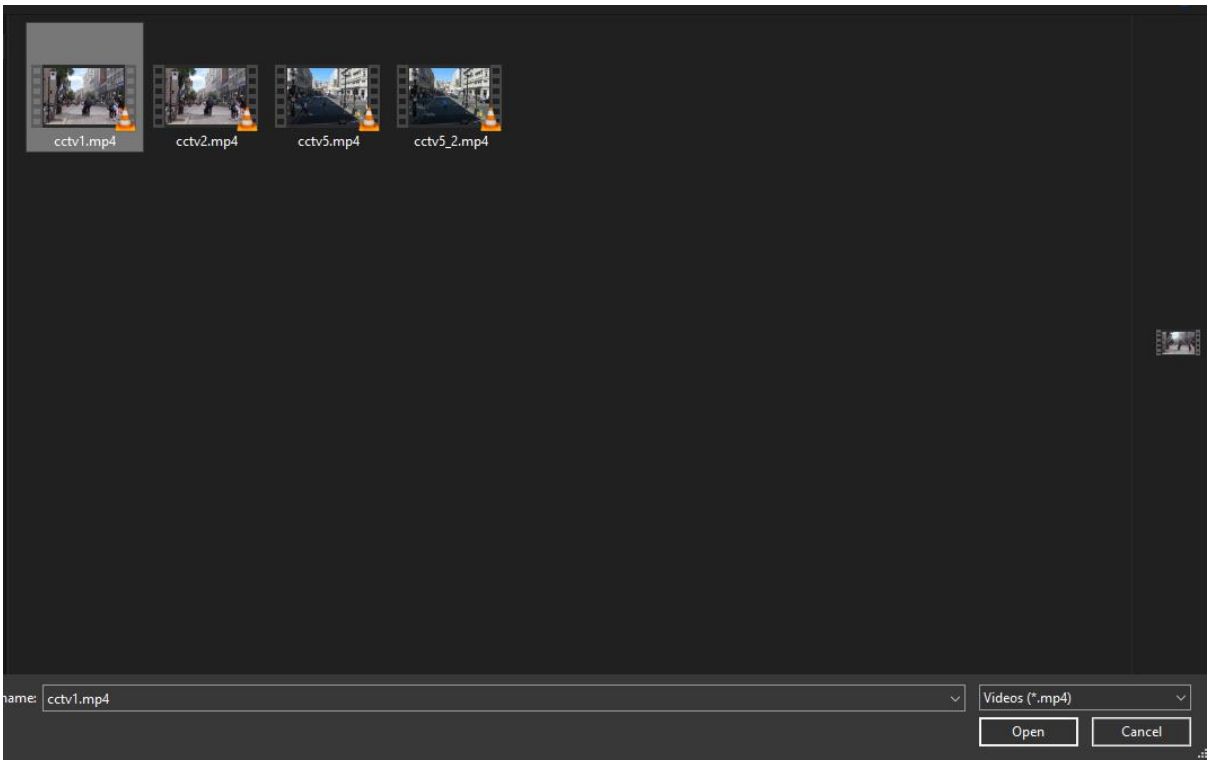


Figure 53: Prompt User to Upload Video

Video and its title are displayed

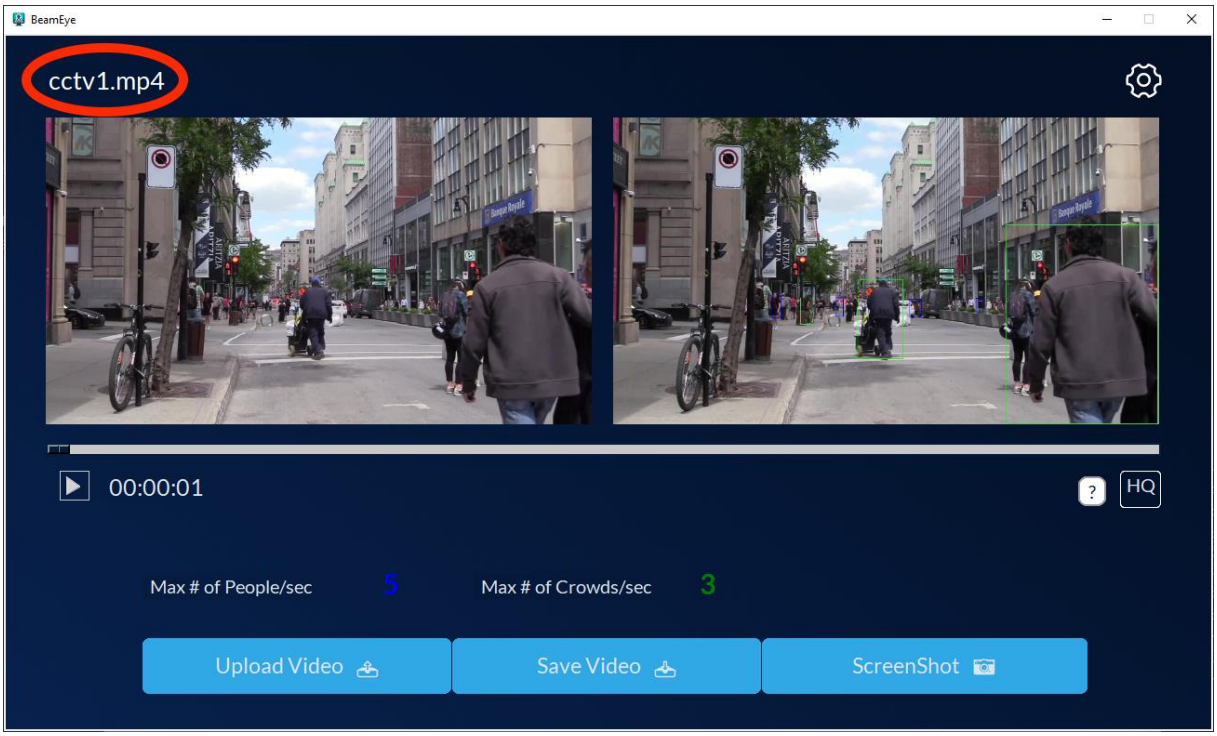


Figure 54: Displayed Video

Table 34: TC02- Test Case 2

Test Case ID	TC02
Module Name	Viewing Detected Pedestrians Functionality
Test Case Scenario	Verify the functionality of viewing uploaded video alongside the detection results, including labeled pedestrians.
Test Case Name	Test Viewing Detection Results
Pre-requisite	System is running and Video is uploaded and processed
Test Data	Video File in mp4 format uploaded to system and processed
Test Steps	1. Verify the original video is displayed on one side of the screen and processed video on the other 2. Confirm the display of the number of detected pedestrians 3. Confirm the display of bounding boxes of different colors for people's walking in groups and single walkers 4. Confirm the presence of other buttons like Save Video, Screenshot or upload video
Expected Results	1. Original and processed video are correctly displayed side by side 2. Pedestrians in the processed video are accurately labeled 3. Number of pedestrians whether in groups or individuals is correctly displayed. 4. Other buttons are present and functional 5. Different color of bounding boxes for crowed and individuals
Actual Results	The system displayed the videos side by side and pedestrians are detected
Status	Pass
Comments	The video quality changes according to what user chooses
Test Executed By	Omar

TensorFlow Import Time

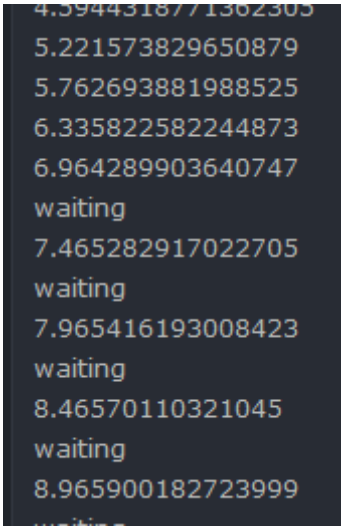


Figure 55: Import Time of TensorFlow

Test Case 02 Result

A. Label with accuracy displayed

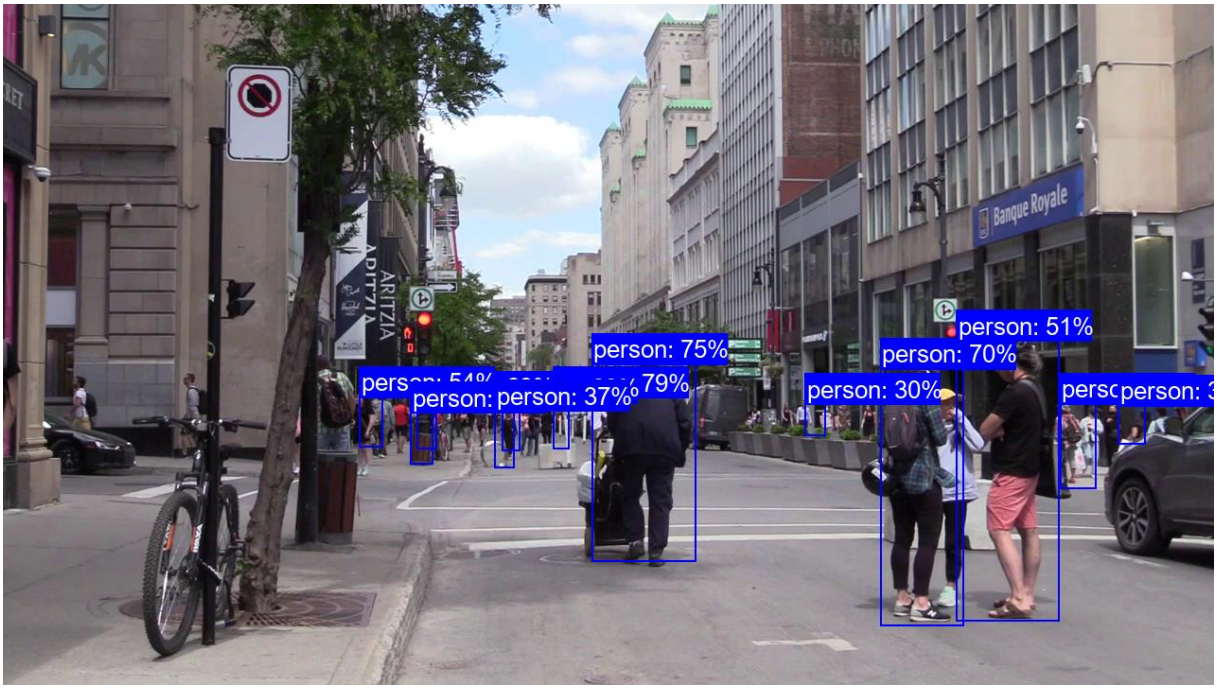


Figure 56: TC02 Results A

B. Labels without accuracy Displayed

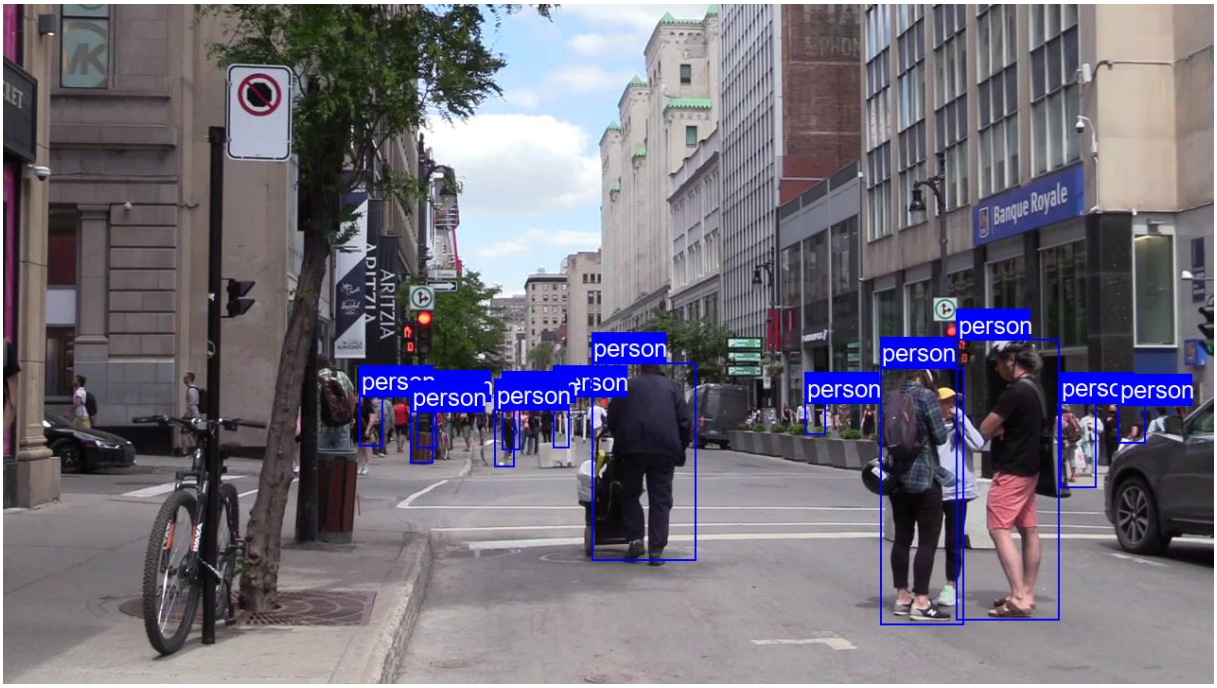


Figure 57: TC02 Results B

C. Only Labels Displayed

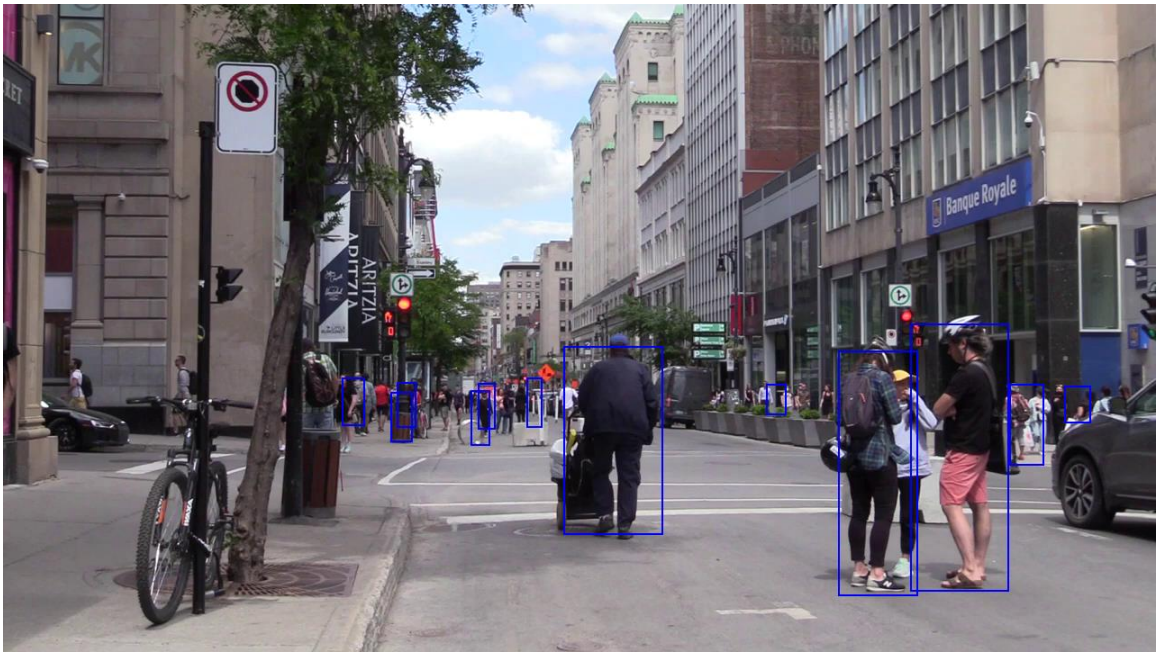


Figure 58: TC02 Results C

Table 35: TC03-Test Case 3

Test Case ID	TC03
Module Name	Save Video and Screenshot Functionality
Test Case Scenario	Verify the functionality of downloading the processed video and capturing screenshots of the detection results
Test Case Name	Test Saving Video and Take Screenshot Features
Pre-requisite	System is running and video is processed with detected pedestrians available for download(save) and screenshot
Test Data	The processed video with pedestrian detection results
Test Steps	1. Locate and click on 'Save Video' button 2. Specify destination for the download video and confirm the download 3. Do same for Screenshot 4. Confirm the video and screenshots are saved in specified location
Expected Results	1. Save Video and Screenshot buttons are functional 2. Processed video is downloaded to specified location 3. Screenshot is captured correctly 4. Video or Screenshots are saved correctly
Actual Results	Both buttons were functioning as expected
Status	Pass
Comments	User does not need to specify a destination for screenshots since they are saved directly to screenshots folder of the system
Test Executed By	Omar

Folder to where images are saved

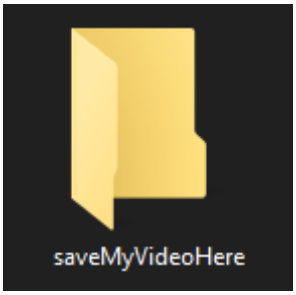


Figure 59: TC03 Saving Folder

After Button is clicked, user is prompted to specify destination

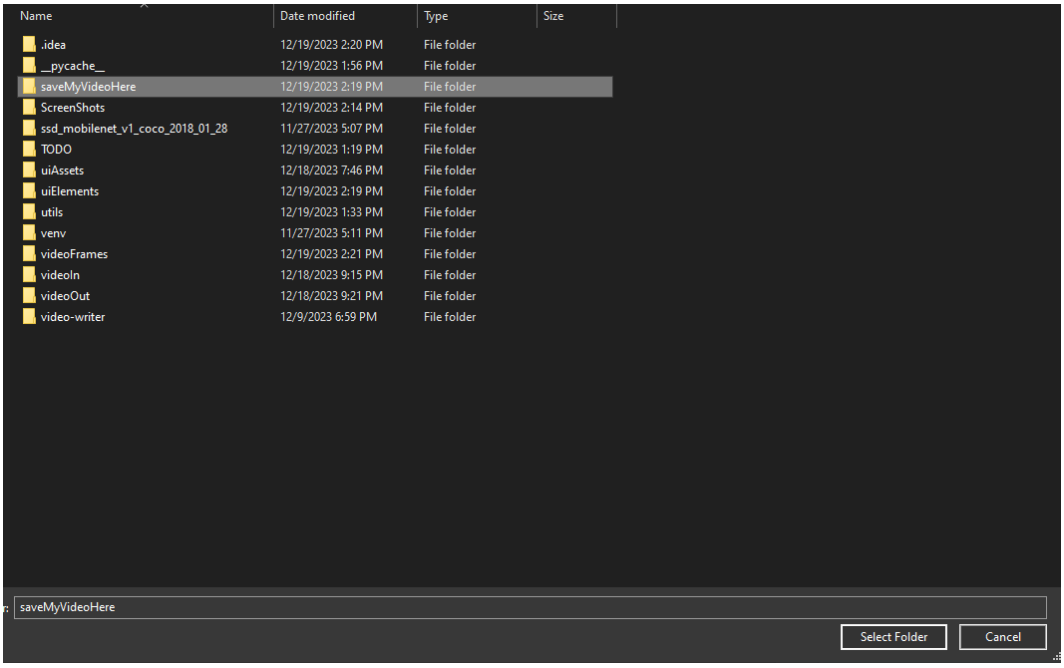


Figure 60: TC03 Prompt to Change Destination

After File is saved Button Status

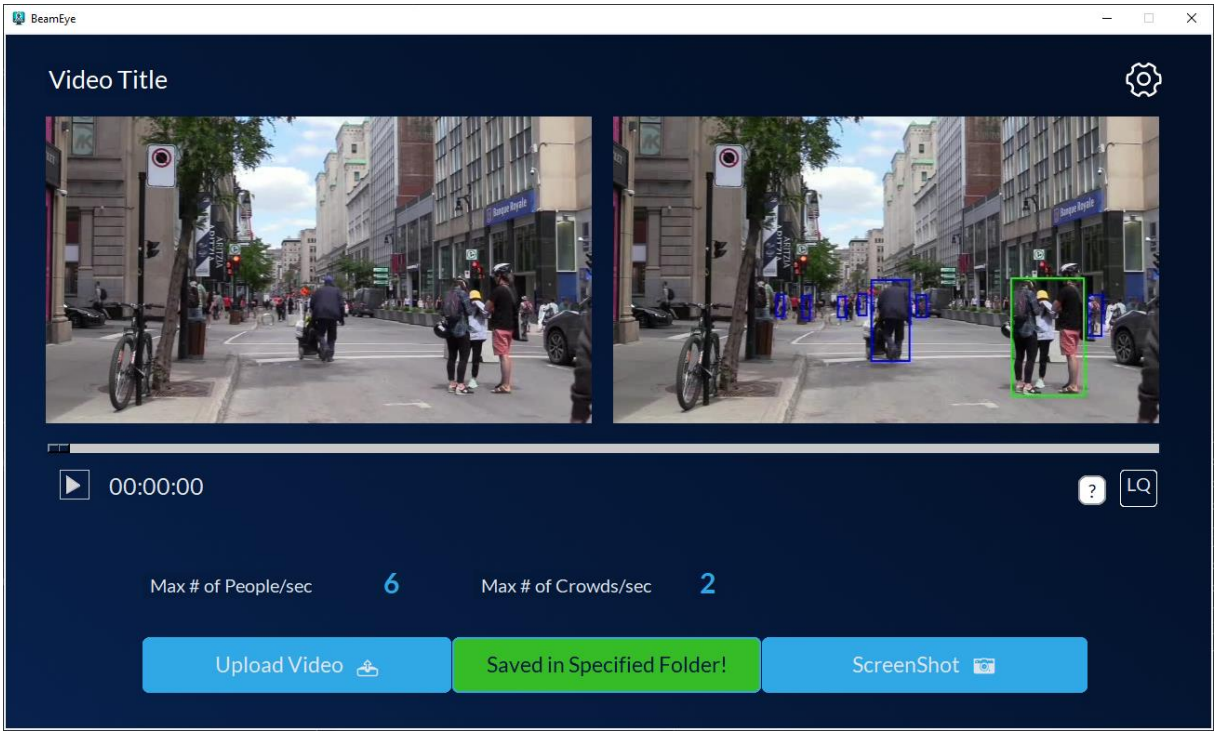


Figure 61: TC03 Button Status

And Video is saved as shown

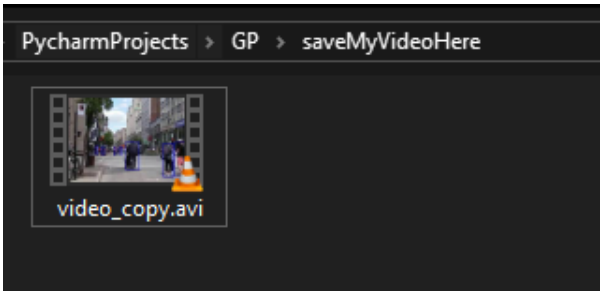


Figure 62: TC03 Video Saved

Screenshot Testing

Before Taking Screenshot, screenshots folder is empty

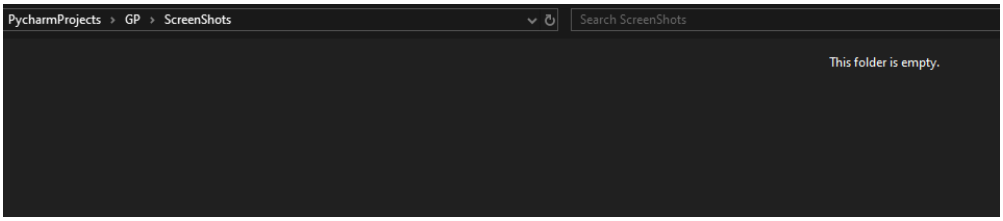


Figure 63: TC03 Screenshots Folder Empty

Clicking on Button

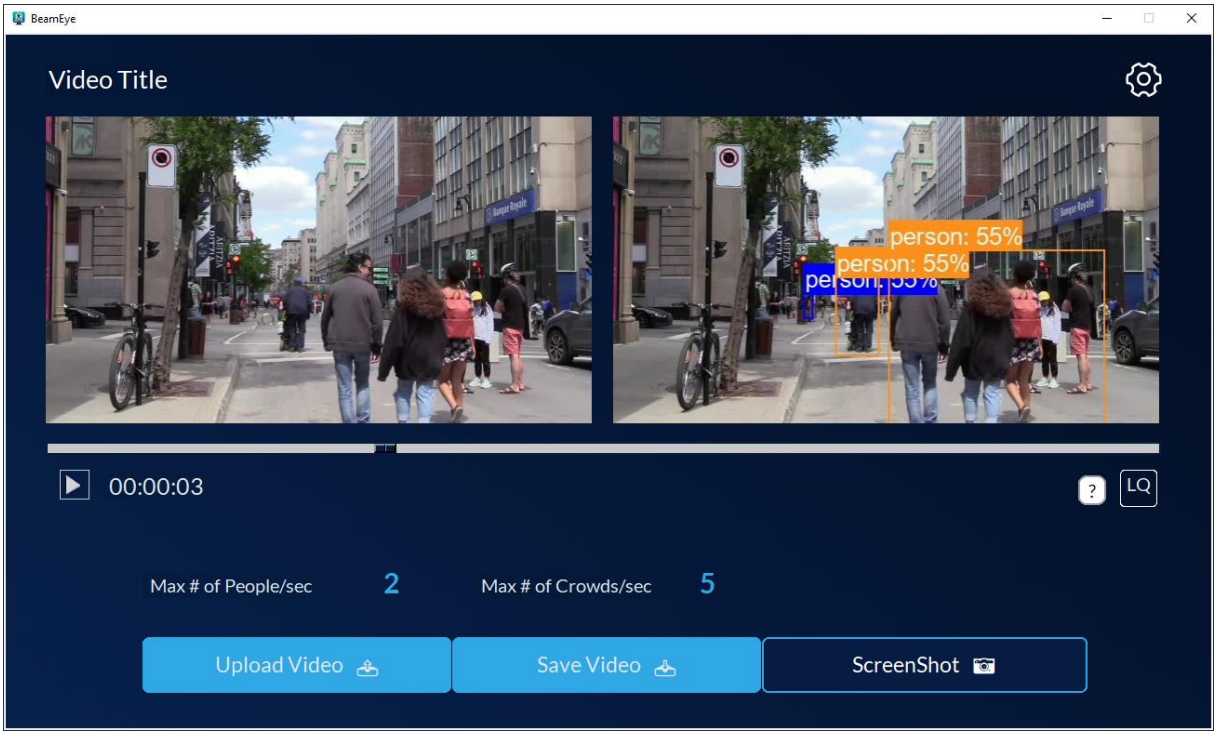


Figure 64: TC03 Screenshot Button Clicked

Folder after taking a screen shot, it is saved

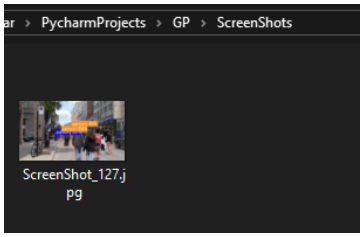


Figure 65: TC03 Screenshot Saved

The Screenshot Saved

Figure 66: TC03 The Saved Screenshot

After button is clicked and screenshot is saved, the button status change

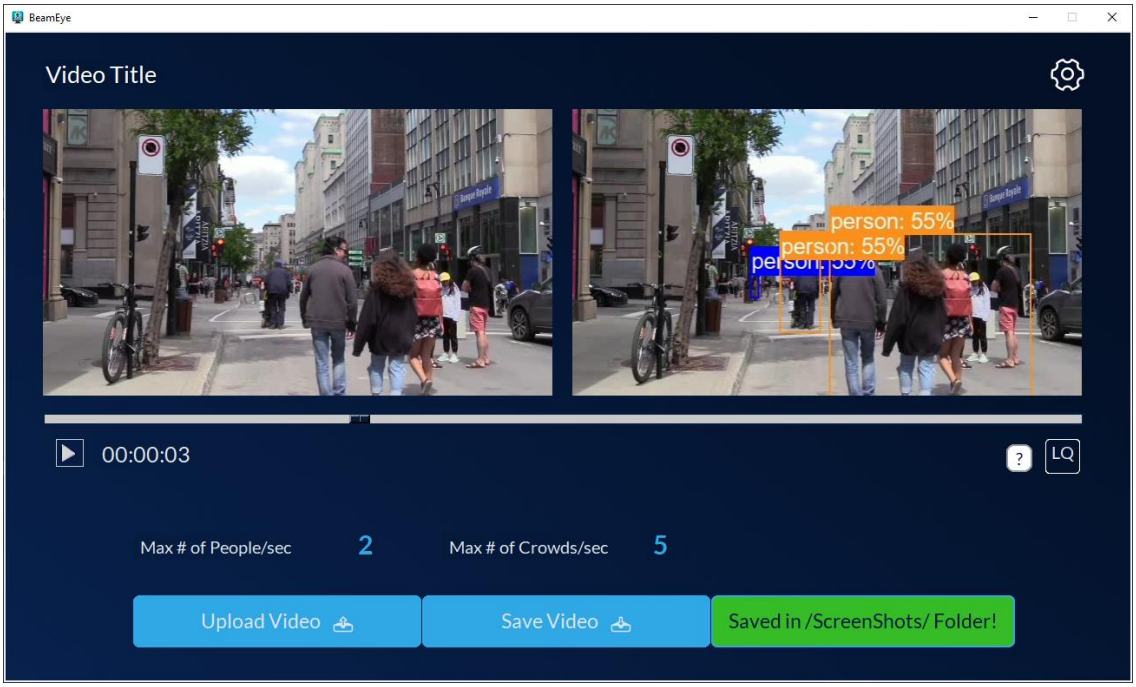


Figure 67: TC03 Screenshot Button Status

Table 36: TC04A- Toggling Labels

Test Case ID	TC04A
Module Name	User Preferences and Settings – Toggling Labels
Test Case Scenario	Verify the functionality of including labels in the detection display
Test Case Name	Test Inclusion of Labels in Detection Display
Pre-requisite	System is operational with available settings menu
Test Data	None
Test Steps	1. Click on the setting gear icon 2. Check the option of including labels 3. Save Changes and observe detection display
Expected Results	The detection video correctly reflects the inclusion of labels
Actual Results	Expected result was achieved
Status	Pass
Comments	The label for crowd is also different from individual pedestrians
Test Executed By	Omar/Khawlah

Table 37: TC04B- Toggling Accuracy

	TC04B
Module Name	User Preferences and Settings – Toggling Accuracy Information
Test Case Scenario	Verify the functionality of including accuracy in the detection display
Test Case Name	Test Inclusion of Accuracy in Detection Display
Pre-requisite	System is operational with available settings menu
Test Data	None
Test Steps	1. Click on the setting gear icon 2. Check the option of including accuracy 3. Save Changes and observe detection display
Expected Results	The detection video correctly reflects the inclusion of Accuracy
Actual Results	Expected results were achieved
Status	Pass
Comments	label, accuracy, crowd have cross dependency so testing of all cases in needed unlike colors, which only need to be tested once
Test Executed By	Omar/Khawlah

Table 38: TC04C- Crowd Detection

Test Case ID	TC04C
Module Name	User Preferences and Settings – Crowd Detection
Test Case Scenario	Verify the functionality enabling/disabling crowd detection option
Test Case Name	Test Enabling/Disabling Crowd Detection
Pre-requisite	System is operational with available settings menu
Test Data	None
Test Steps	1. Click on the setting gear icon 2. Enable or disable the crowd detection option 3. Save Changes and observe detection display
Expected Results	The detection video correctly reflects the enabled or disabled state of the crowd detection.

Actual Results	Expected results were achieved
Status	Pass
Comments	If crowd detection is enabled, they have different bounding boxes from individuals.
Test Executed By	Omar/Khawlah

Table 39: TC04D-Bounding Boxes Colors

Test Case ID	**TC04D**
Module Name	User Preferences and Settings – Bounding Boxes Colors
Test Case Scenario	Verify the functionality of choosing different colors for crowd and single pedestrians bounding boxes and labels
Test Case Name	Test Choosing Bounding Boxes Colors
Pre-requisite	System is operational with available settings menu
Test Data	None
Test Steps	1. Click on the setting gear icon 2. Select colors for crowd and single pedestrian bounding boxes 3. Save Changes and observe detection display
Expected Results	The detection video correctly reflects the chosen colors for crowd and single pedestrian labels
Actual Results	Expected results were achieved
Status	Pass
Comments	The color of crowd should be different from individuals and system does not allow for choosing same color for both
Test Executed By	Omar/Khawlah

Test Cases Figures for TC04A, TC04B, TC04C, TC04D are shown bellow

The combinations for Settings

```
(Label: False, Accuracy: False, Crowd: False)
(Label: False, Accuracy: False, Crowd: True)
(Label: False, Accuracy: True, Crowd: False)
(Label: False, Accuracy: True, Crowd: True)
(Label: True, Accuracy: False, Crowd: False)
(Label: True, Accuracy: False, Crowd: True)
(Label: True, Accuracy: True, Crowd: False)
(Label: True, Accuracy: True, Crowd: True)
```

Figure 68: Settings Combination

No Labels nor accuracy nor crowd are choses (label False, Accuracy False, Crowd False)

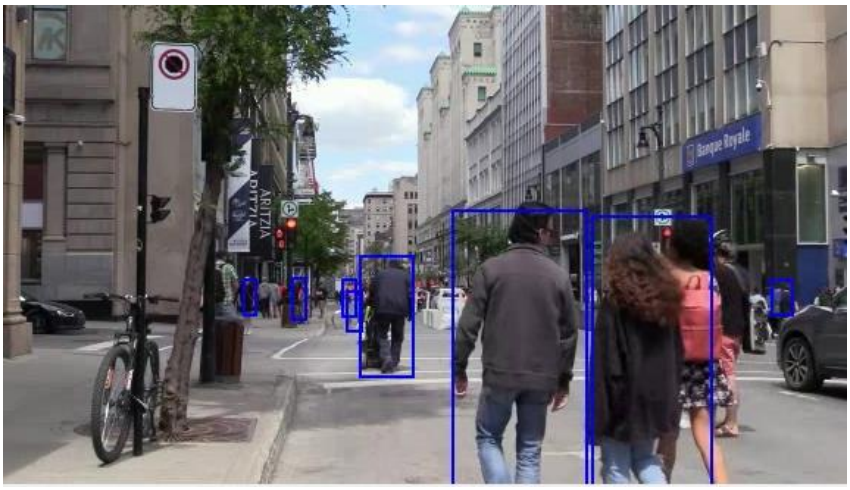


Figure 69: TC04 Labels + Accuracy + Crowd = False

Crowd is true, Labels and accuracy False

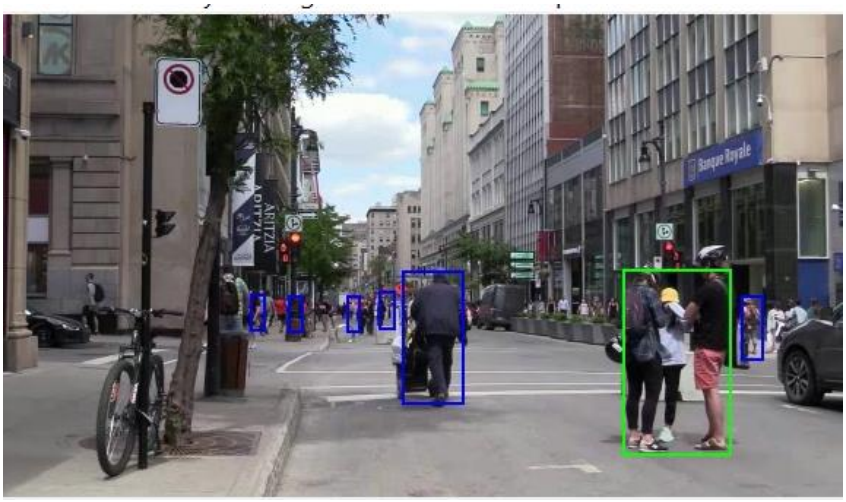


Figure 70: TC04 Labels + Accuracy = False, Crowd = True

Accuracy True, Labels False, Crowd False

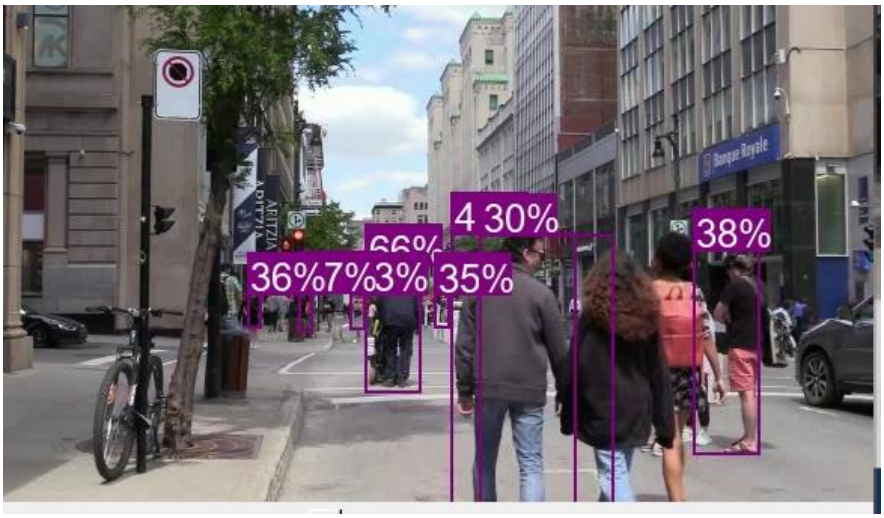


Figure 71: TC04 Labels + Crowd = False, Accuracy = True

Labels False, Accuracy True, Crowd True

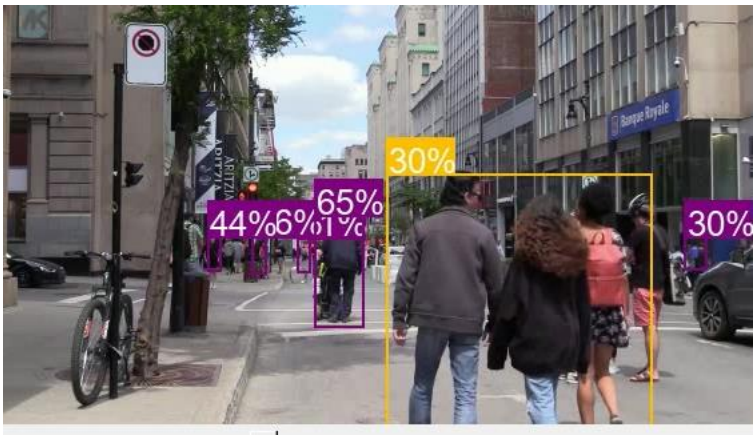


Figure 72: TC04 Labels = False, Accuracy, Crowd = True

Labels True, Accuracy False Crowd False

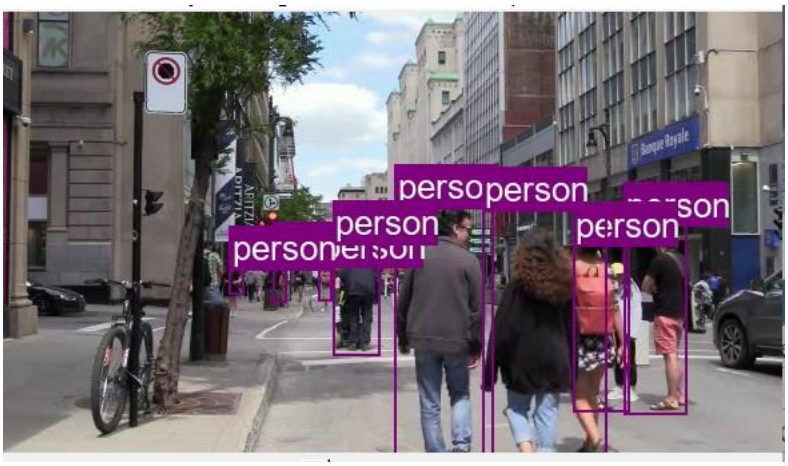


Figure 73: TC04 Labels = True, Accuracy + Crowd = False

Label True, Accuracy False, Crowd True

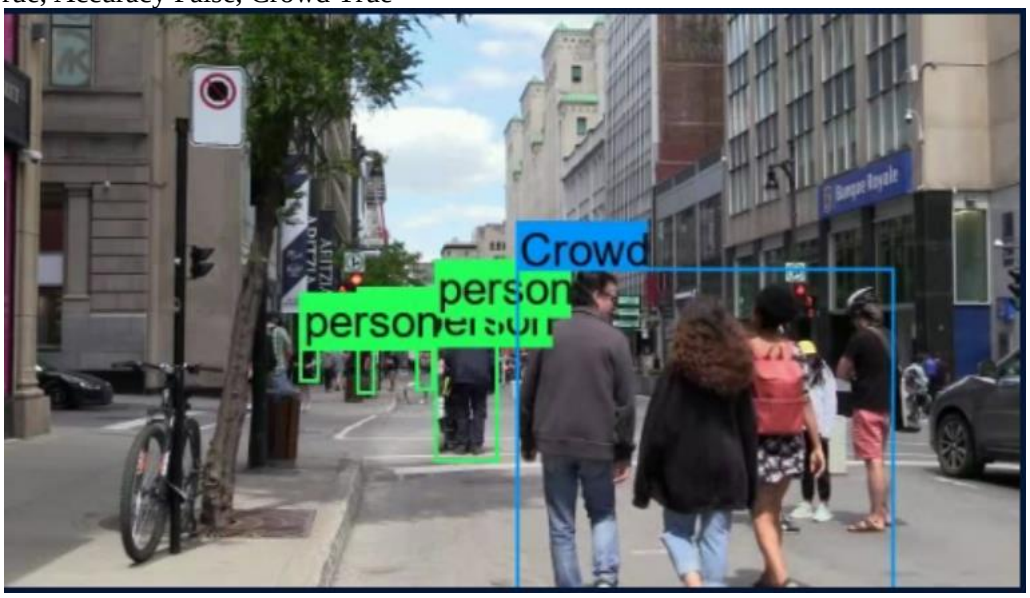


Figure 74: TC04 Label + Crowd = True, Accuracy = False

Labels True Accuracy True Crowd False

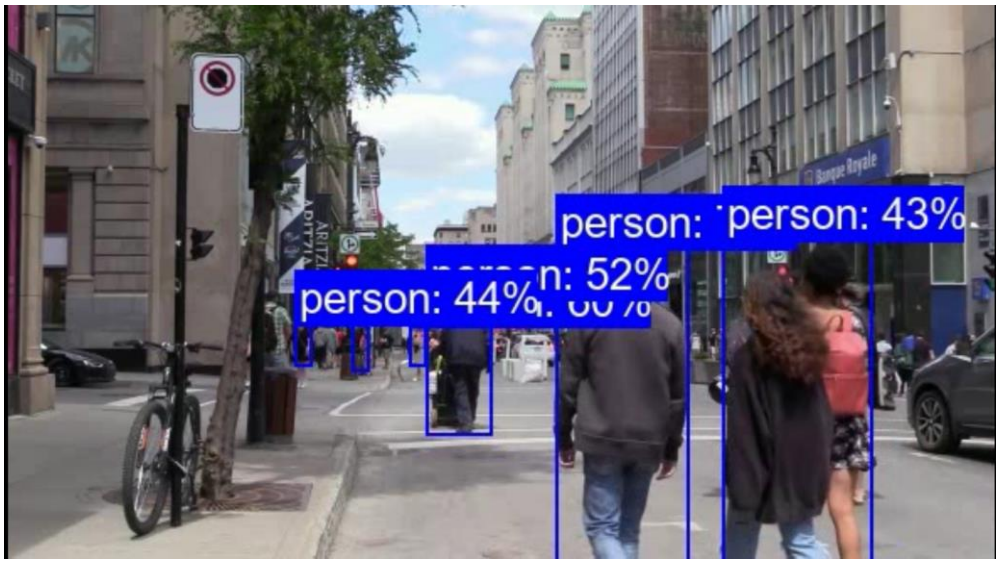


Figure 75: TC04 Labels + Accuracy = True, Crowd = False

Labels True, Accuracy True, Crowd True

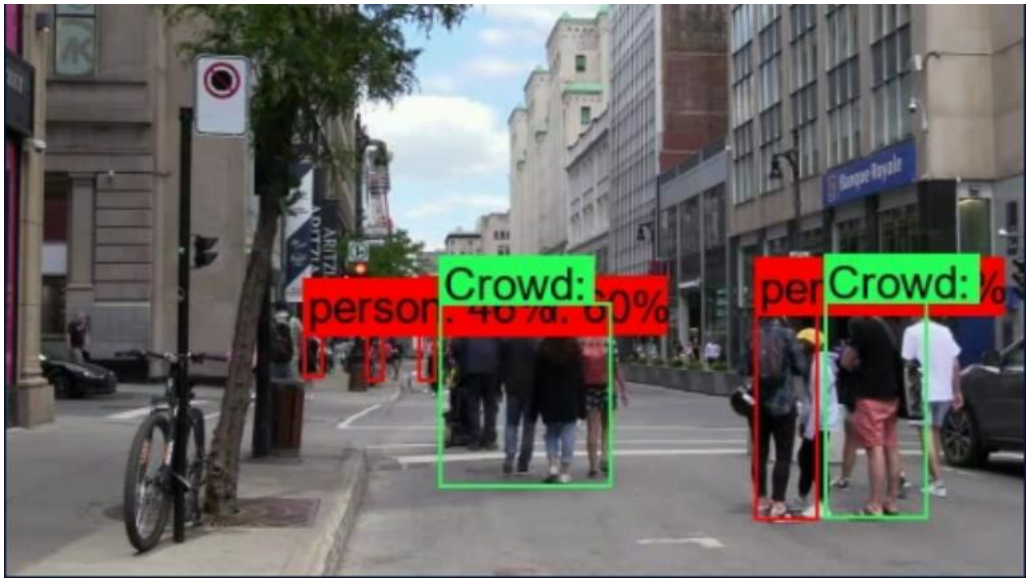


Figure 76: TC04 Labels + Accuracy + Crowd = False

Error Message Appears if user chooses same colors

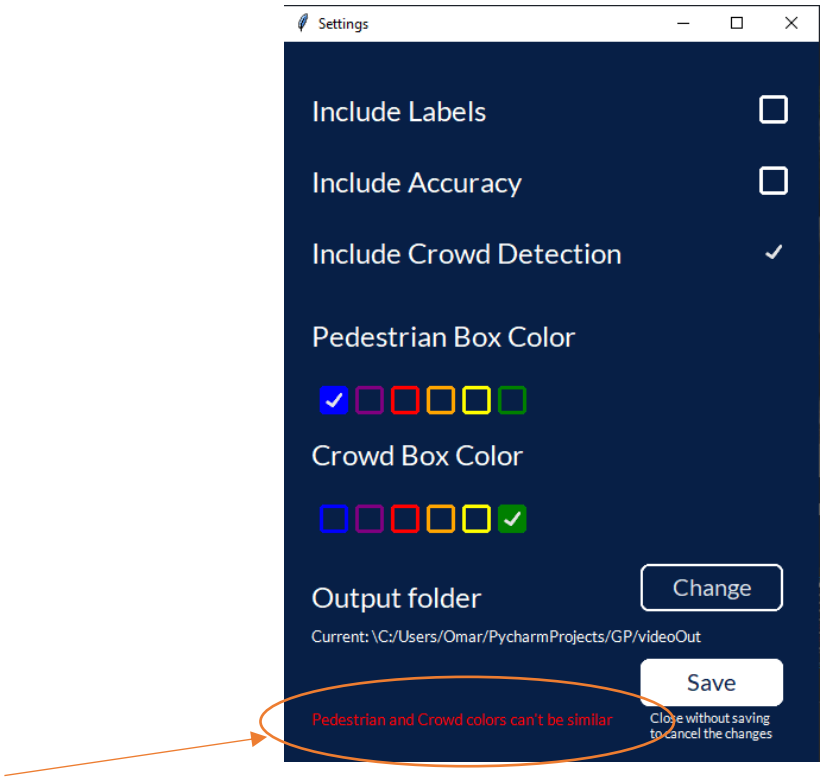


Figure 77: TC04 Settings Message Changes

Crowd Color is Different from Pedestrians Color

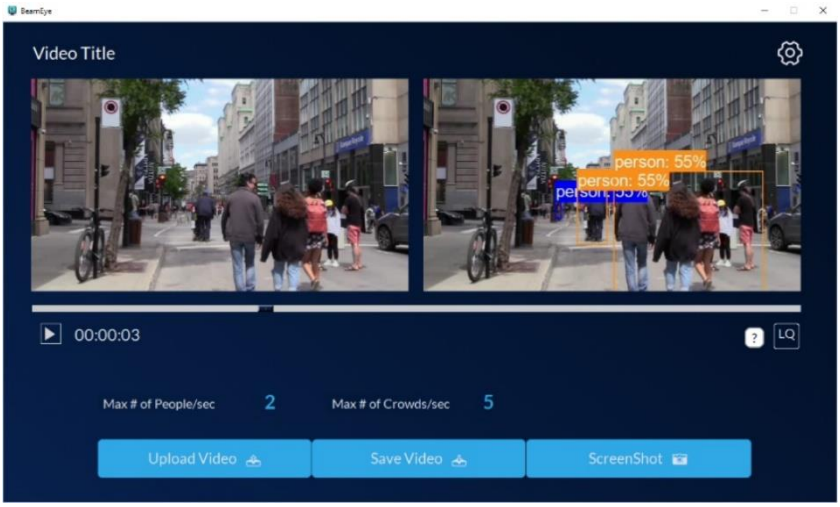


Figure 78: TC04 Different Colors for Crowd and Pedestrians

If Crowed Detection is Off, they are not counted

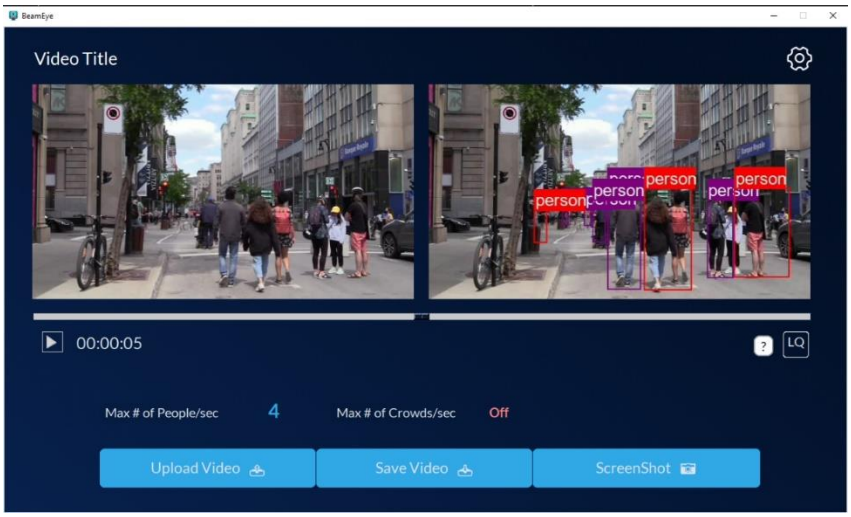


Figure 79: TC04 Crowd Detection is Off

Crowd on different coloring

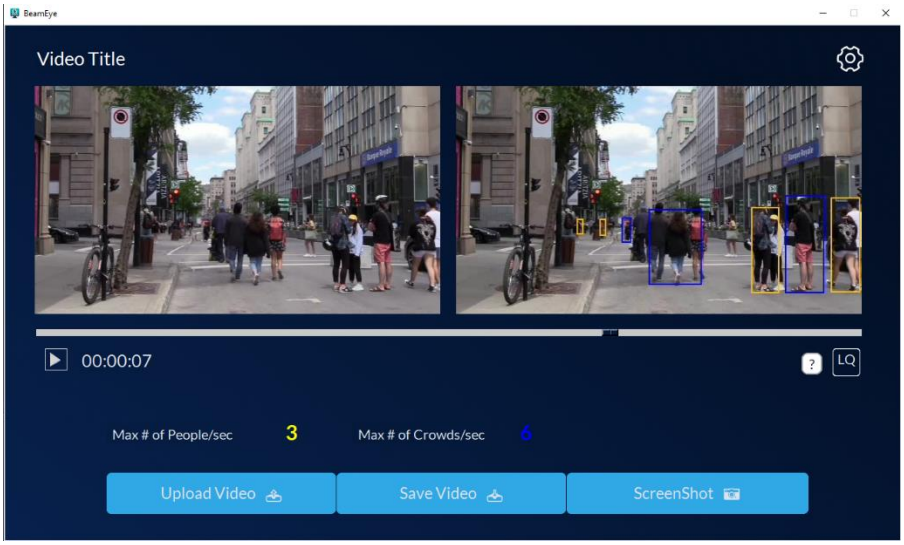


Figure 80: TC04 Different Coloring for Crowd and Pedestrians

Table 40: TC04E Test Changing Quality Function

Test Case ID	**TC04E**
Module Name	User Preferences and Settings – Change Quality Of Video
Test Case Scenario	Verify the functionality Changing the quality of a video according to user preferences
Test Case Name	Test Changing Quality of Video Functionality
Pre-requisite	System is operational with available settings menu
Test Data	Videos
Test Steps	1. Click on LQ icon 2. Select different video to check 3. Look up the changes

	4. Change LQ to HQ and observe Results
Expected Results	The lower quality video will run faster than higher quality
Actual Results	Expected results were achieved
Status	Pass
Comments	Test code used here as shown in figures bellow
Test Executed By	Omar

The user can see this information for either to choose lower quality video or higher quality video

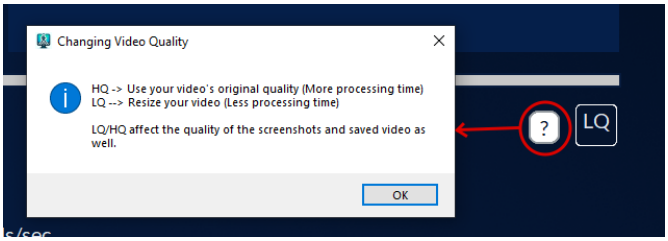


Figure 81: Information Window

This is the icon user can click to change quality

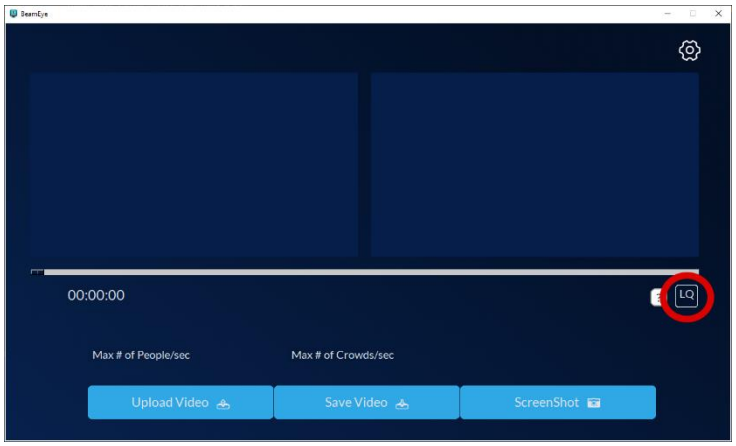


Figure 82: TC04E Quality Icon

Test Code

```
start = time.time()
begin()
end = time.time()
width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
print(f"Video resolution: {width}x{height}")
print(f"Took {end-start} seconds to read video, slice, process, and put frames back together")
```

Figure 83: TC04E Quality Code Test

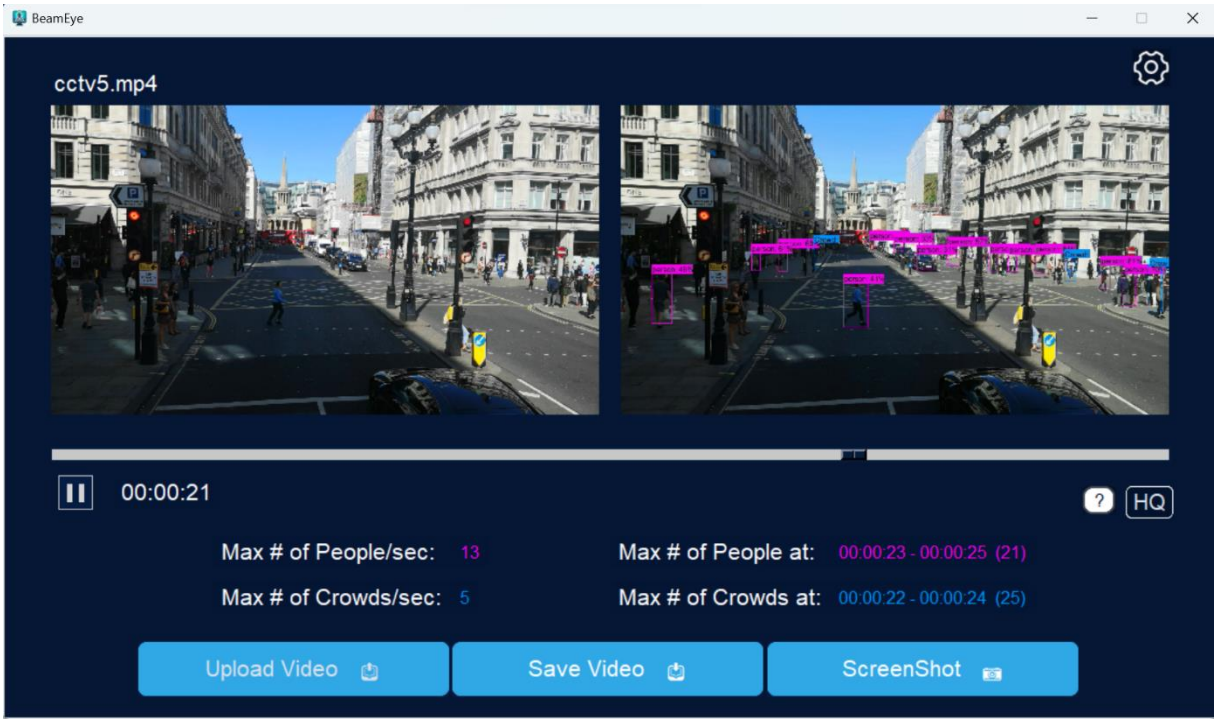


Figure 84: TC04E Higher Quality Video

Original Video Info

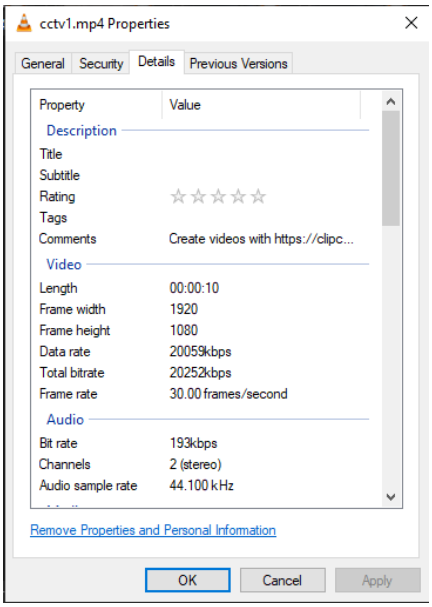


Figure 85: TC04E Original Video Prosperities

Lower Quality (LQ)

LQ Test Code output

```
Video resolution: 586.0x330.0
Took 22.665955543518066 seconds to read video, slice, process, and put frames back together
```

Figure 86: TC04E LQ Test Code Output

Resized Video Info

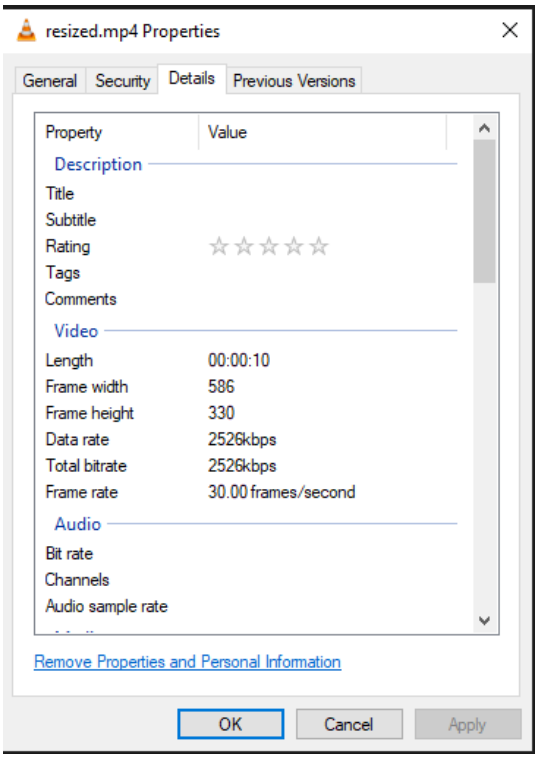


Figure 87: TC04E Resized Video Information

LQ Frame Output Info

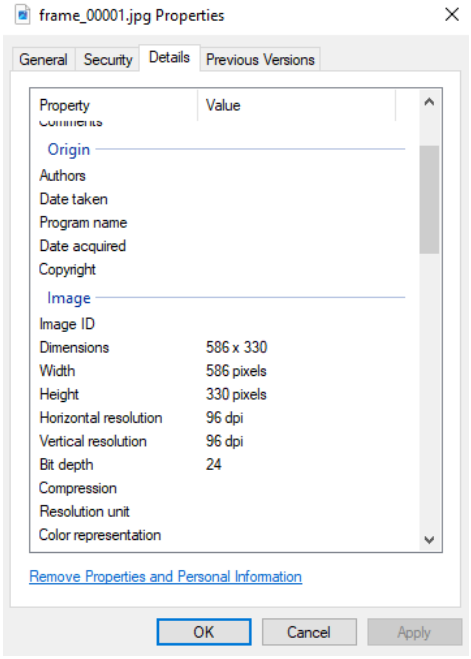


Figure 88: TC04E LQ Frame Output Information

High Quality Testing

High Quality Test Code Output

```
Video resolution: 1920.0x1080.0
Took 104.89993190765381 seconds to read video, slice, process, and put frames back together
```

Figure 89: TC04E HQ Test Code Output

HQ Frame Output

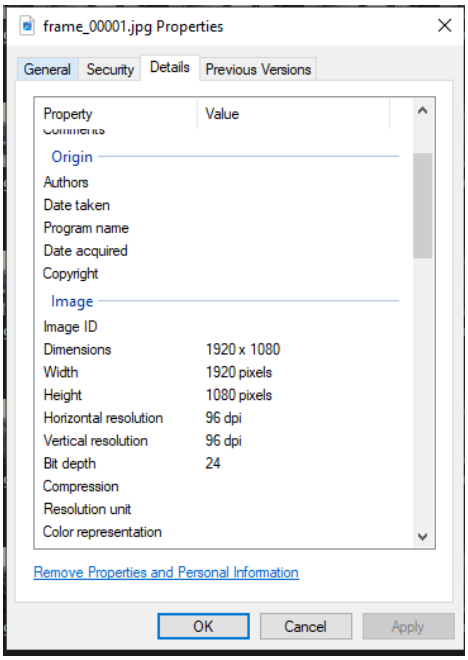


Figure 90: TC04E HQ Frame Prosperities

HQ video Output

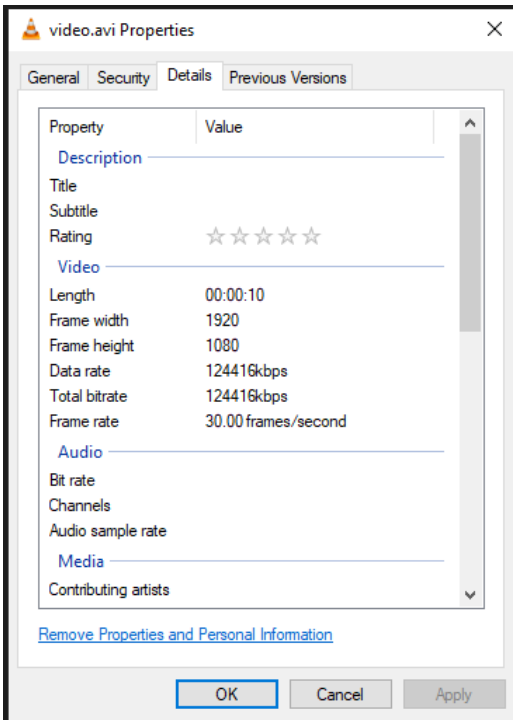


Figure 91: TC04E HQ Video Prosperities

Table 41: TC05A Testing Loading Time

Test Case ID	TC05A
Module Name	Response Time- Loading Input Video
Test Case Scenario	Verify that loading an input video takes no more than duration of video
Test Case Name	Test Loading Time of Input Video
Pre-requisite	System is operational
Test Data	Multiple video files of varying lengths and sizes
Test Steps	1. Select and load different video files 2. Measure the time taken to load each video
Expected Results	Each video should load fast, expectedly in less than duration of original video
Actual Results	Some videos took longer than expected to load
Status	Pass
Comments	If user puts quality to reduction, the video loads faster
Test Executed By	Omar/Khawlah

Table 42: TC05B Testing Resource Efficiency

Test Case ID	TC05B
Module Name	Scalability- System Resource Management
Test Case Scenario	Verify that the application uses system resources efficiently
Test Case Name	Test Resource Efficiency

Pre-requisite	System is operational
Test Data	None
Test Steps	1. Operate system under varying video sizes 2. Monitor system resources usage for CPU and Memory and Storage
Expected Results	System efficiently manages resources without significant performance degradation
Actual Results	The system works as intended
Status	Pass
Comments	This was performed by garbage collection, so every time user uploads a new video, previous video frames will be deleted
Test Executed By	Omar/Khawlah

Table 43: TC05C Testing Data Protection

Test Case ID	**TC05C**
Module Name	Safety- User Data Protection
Test Case Scenario	Verify that system does not access or modify data beyond what is provided by user
Test Case Name	Test User Data Protection
Pre-requisite	System is operational
Test Data	Simple user data for testing
Test Steps	1. Check specific data for processing 2. Observe system interaction with data
Expected Results	System only accesses and Modifies data provided without affecting other data
Actual Results	Expected results were achieved
Status	Pass
Comments	Since this is a windows application, the user can only upload a video and system only access the file user specifies
Test Executed By	Omar/Khawlah

Table 44: TC05D Testing Secure File Handling

Test Case ID	**TC05D**
Module Name	Security- Secure File Handling
Test Case Scenario	Verify system securely handles files, preventing unauthorized access or use of suspicious files.
Test Case Name	Test Secure File Handling
Pre-requisite	System is operational
Test Data	Video files
Test Steps	1. Select and load different video files 2. Specify and use various output folders
Expected Results	Only video files are processed and output is saved in user-specified locations securely
Actual Results	System behaves as expected
Status	Pass

Comments	There should have been more security considerations but since our product is just a windows app we did not care about this much.
Test Executed By	Omar/Khawlah

Table 45: TC05E Testing System's Usability

Test Case ID	**TC05E**
Module Name	Software Quality- Usability
Test Case Scenario	Verify the usability of the application, focusing on the intuitiveness of the interface and ease of navigation
Test Case Name	Test System's Usability
Pre-requisite	System is operational
Test Data	None
Test Steps	1. Interact with various features of the application 2. Observe the intuitiveness and ease of navigation 3. Validate user preference settings.
Expected Results	The application interface is user-friendly and navigation is straightforward for users with varying technical expertise
Actual Results	System behaves as expected
Status	Pass
Comments	We achieved our goal of creating an intuitive user interface
Test Executed By	Omar/Khawlah

Table 46: TC05F Testing System's Reliability

Test Case ID	**TC05F**
Module Name	Software Quality- Reliability
Test Case Scenario	Verify the system's reliability under various conditions and scenarios
Test Case Name	Test System's Reliability
Pre-requisite	System is operational
Test Data	Various video files and user interaction scenarios
Test Steps	1. Test the system with different video files and user actions 2. Observe system's behavior for stability and consistency
Expected Results	The system operates robustly and stably, with minimal crashes or unexpected behavior
Actual Results	System behaves as expected
Status	Pass
Comments	This part requires more testing with large video files like 2 hours long video
Test Executed By	Omar/Khawlah

By showing all the Quality Control Testing Cases, we trust that our system performs as intended, offering the user a great User Interface and detection capabilities. The model's accuracy is too low, yet this can be solved by fine-tuning it to increase its accuracy.

7. USER GUIDE OF THE SYSTEM

This is the first Screen displayed to User, the aim of it is to load the program

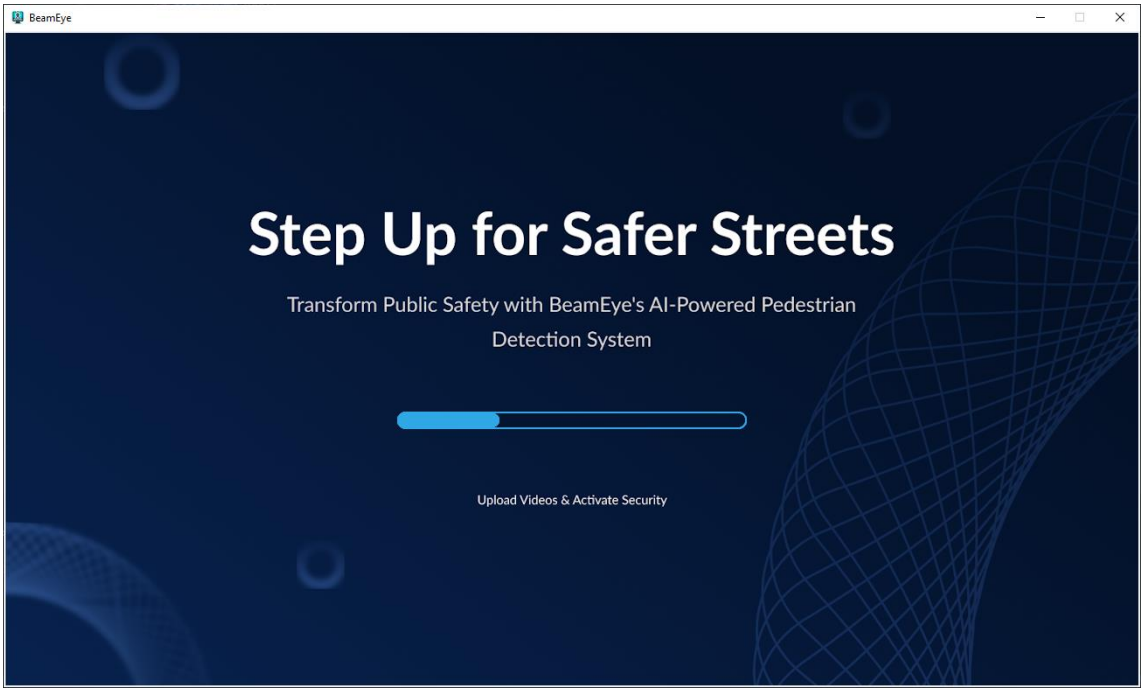


Figure 92: Hello Screen

After 1 to 2 Seconds The loading bar is full and next Screen will open

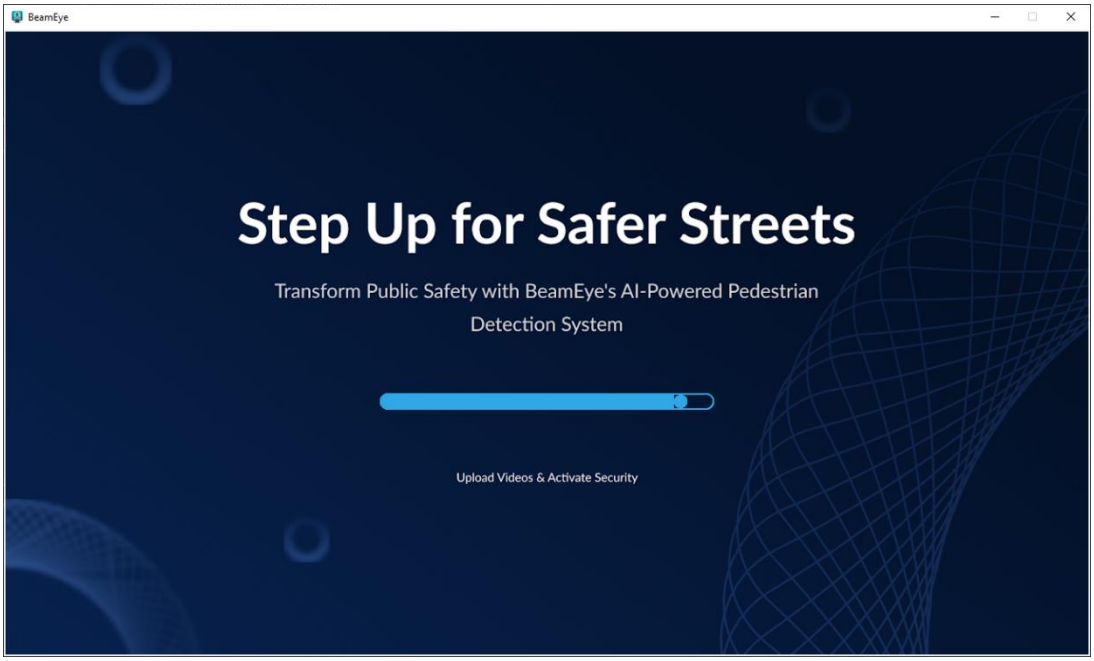


Figure 93: Loading Bar Completion for Hello Screen

After First screen loads this is the second screen. Information about elements is shown

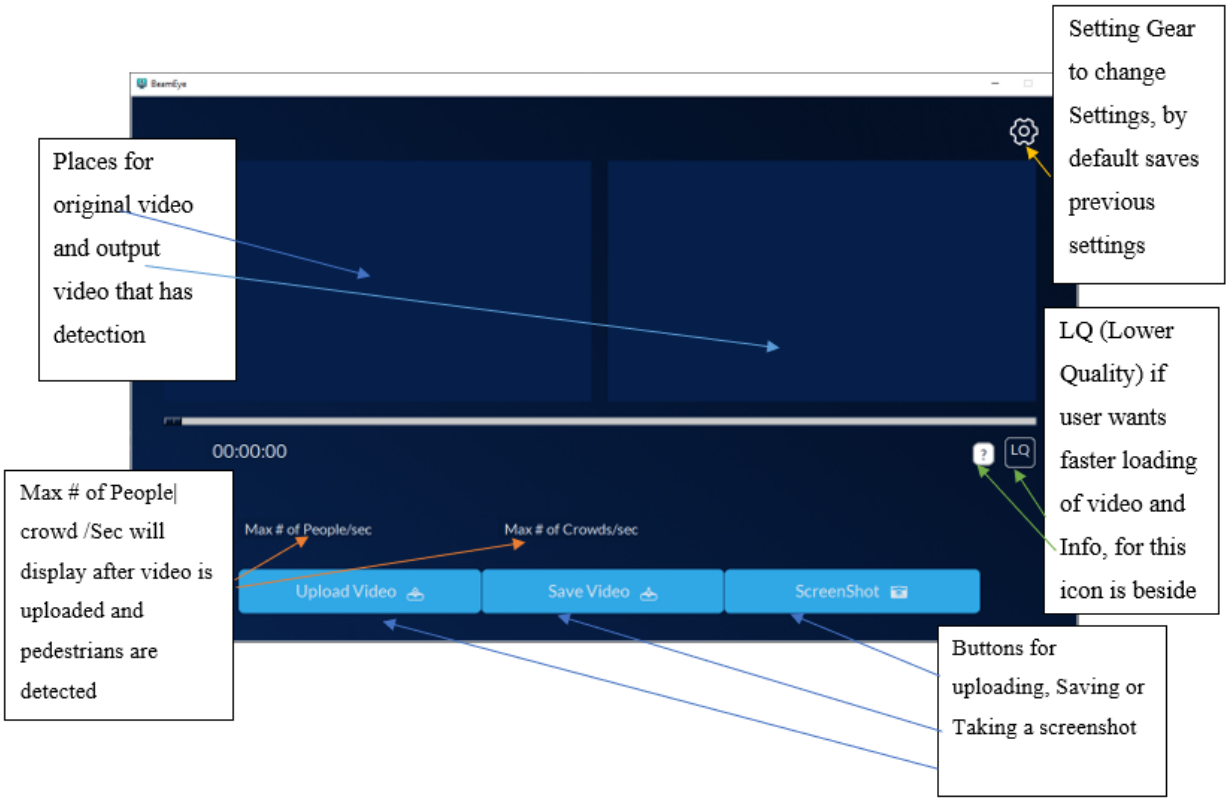


Figure 94: Second and Main Screen Guide

If there is a need to change quality of video for faster loading time

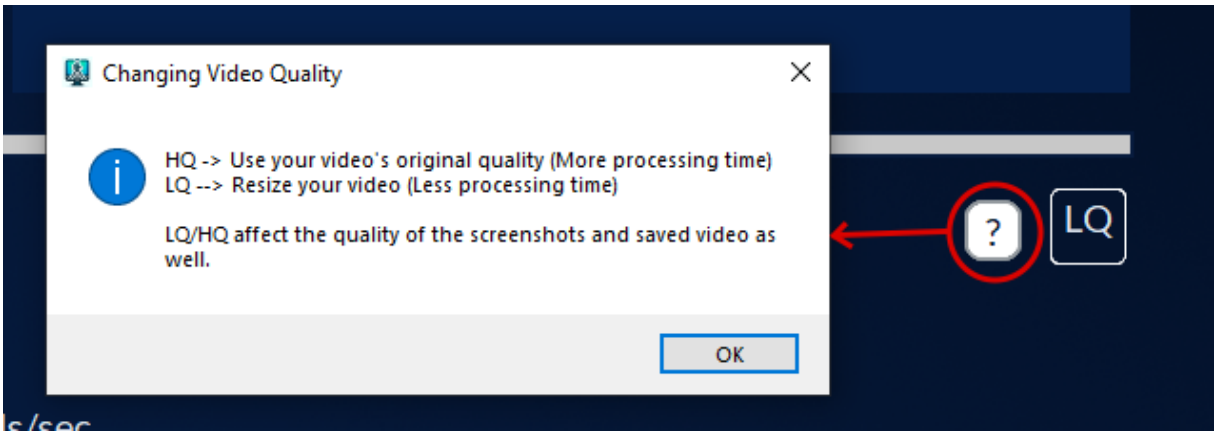


Figure 95: Changing Video Quality Information

Settings Menu Explained

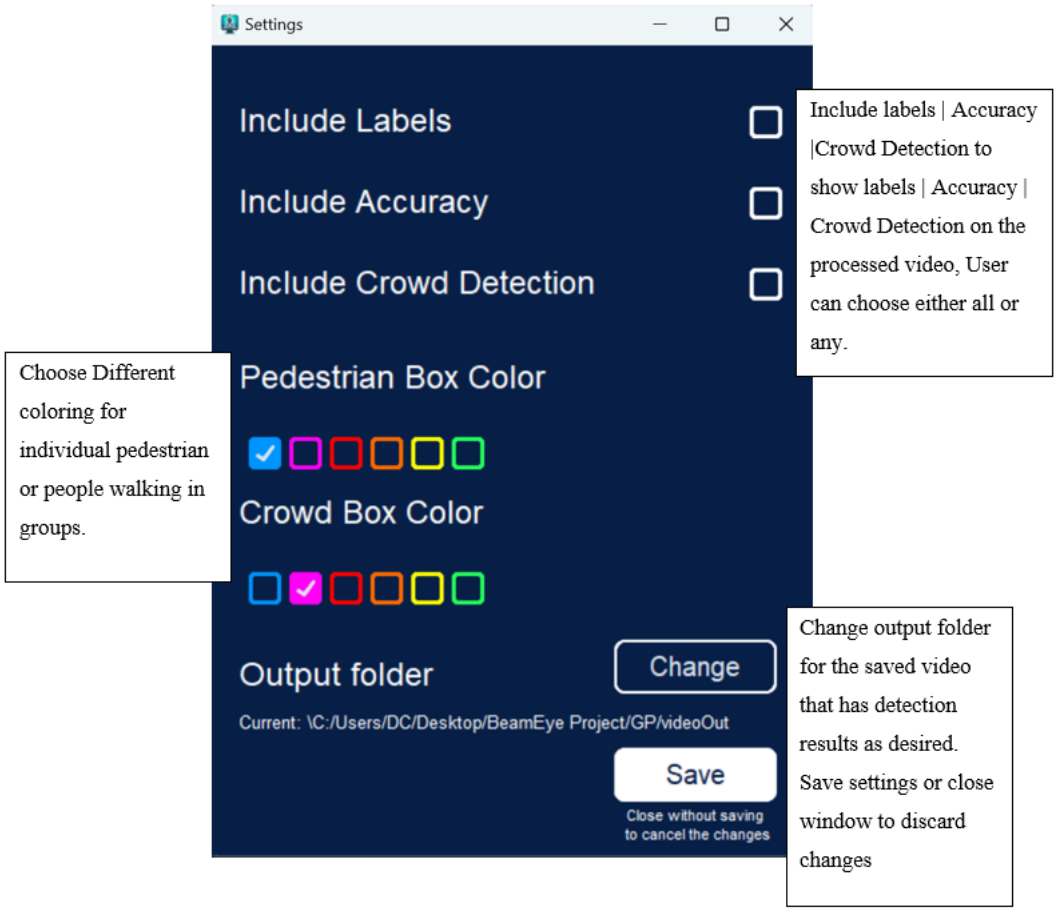


Figure 96: Settings Menu Elements

After Settings are Set

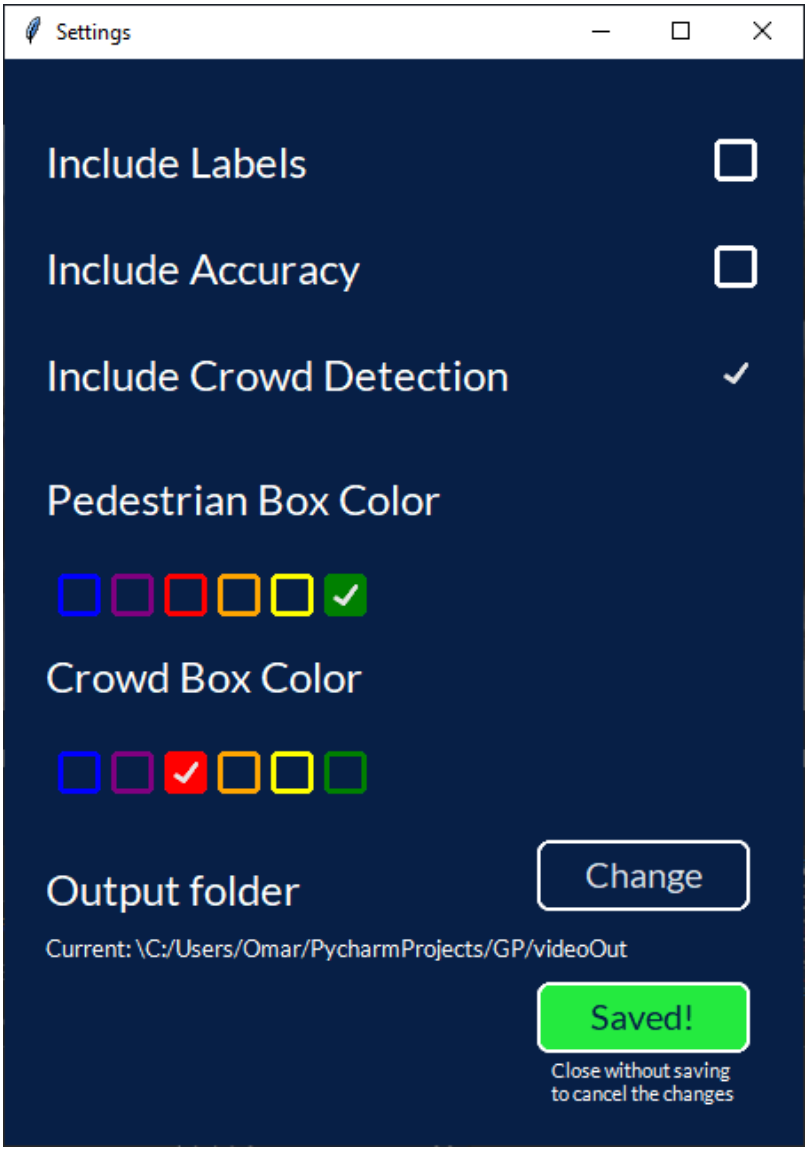


Figure 97: Saving Changes for Settings Screen

While waiting for video processing and pedestrians' detection these screens appear for the user not to feel bord while waiting and to inform user in which process the video is now.

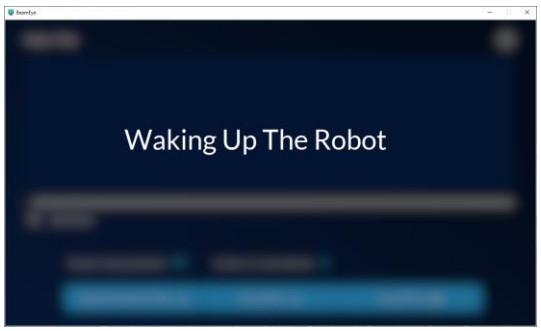


Figure 99: First Waking Up the Robot

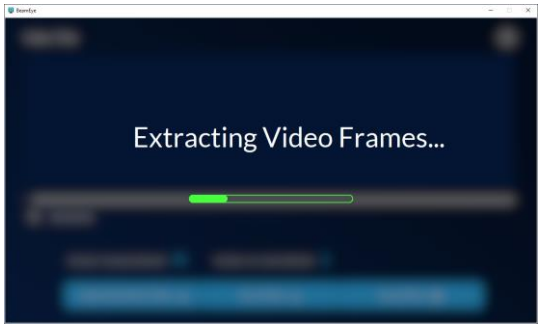


Figure 98: Second Extracting Videos

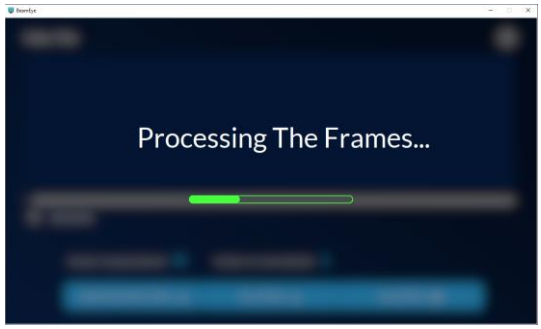


Figure 101: Processing the Frames

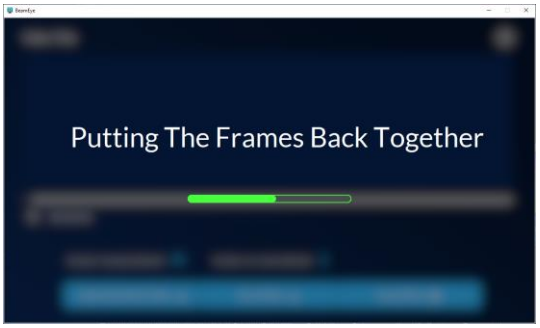


Figure 100: Putting Frames Back Together

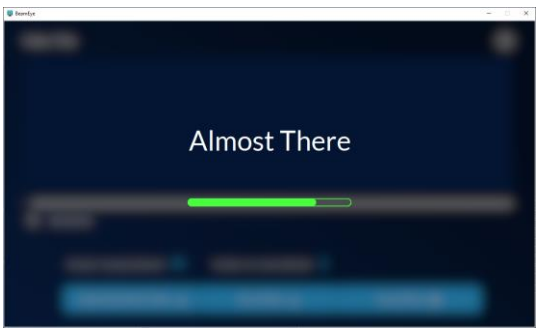


Figure 102: Almost There Screen

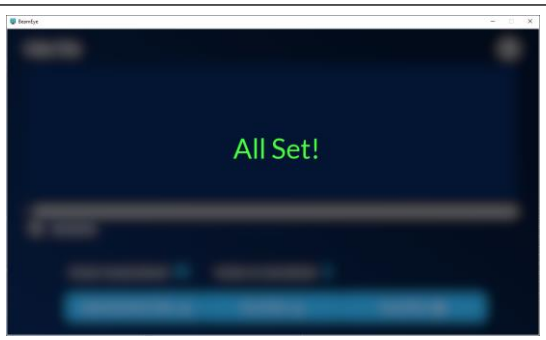


Figure 103: All Set Screen

8. DISCUSSION

1- The Impact of Our Solution Globally
This project will be useful for pedestrians' safety enhancement where if improved to include live detection, it will reduce accidents and fatalities especially in urban areas and crowded places. This project can also aid in better traffic flow management and planning for smart cities. Additionally, using the AI technology and integrating it to intuitive user interface in our project sets a precedent for further AI and ML applications for other purposes beneficial for humanity fostering a more technologically advanced global society.
2- The Impact of our Solution Economically and Socially
Our project can be used by authorized people like government or police officers which will help in saving costs related to traffic management and accident response by preventing pedestrian-related accidents. Furthermore, reduced accident rates can lower insurance claims, benefiting people financially. Not to mention that there can be a potential for job creation in the development, maintenance, and deployment of such a system. For social impact, by reducing risk of accidents our system directly contributes to safety and well-being of communities.
3- The Impact of our Solution Environmentally
For Environmental impact, our project can improve energy savings since efficient traffic management can lead to reduced fuel consumption and emissions. The stop-start cycles will be less if there is a smoother traffic flow. Moreover, if the system improved to collect data, this can assess in planning for environment friendly urban cities.

9. CONCLUSION

As we conclude this report along with the project, it is evident that the development of BeamEye is more than just a technical achievement; it is a step towards safer streets and a more aware society. The analysis made at the initial stage of planning up to user guide illustrates our dedication to creating a tool that is both technologically advanced and serves the need of the community.

Our project uses an AI model to accurately detect people, integrated into a user-friendly desktop app. As a result, we have succeeded in creating a product that not only meets technical expectations but also addresses a crucial need for safety. We have made the app in a way that it can be integrated to external surveillance systems for real time detection. Also, if improved to include facial recognition technologies, this system can be a more valuable tool for city planners, security personnel and the public.

As a team, creating an innovative solution that was not made before was immensely rewarding, challenging, and enlightening. Throughout the development, we have gained deeper understanding of machine learning, software development and project management. Dealing with challenges of integrating complex models to a user-friendly interface using a tool like Tkinter has enhanced our problem-solving skills and technical expertise. Moreover, working on each phase of the project has provided us with practical application of our theoretical knowledge, allowing us to experience bringing an AI project from concept to reality.

Finally, we extend our gratitude to all who have contributed to the completion of this project, and we anticipate that this report will serve as a valuable resource for future improvements in AI usability. BeamEye project has not only added a valuable tool to the realm of public safety but also has been a significant step in our professional and personal growth.

10. REFERENCES

[1]. OpenCV: Cv:VideoCapture class reference. *(n.d.). OpenCV documentation index. https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html*

[2]. *How to pass arguments to Tkinter Button command?* (2022, October 3). GeeksforGeeks. https://www.geeksforgeeks.org/how-to-pass-arguments-to-tkinter-button-command/

[3]. *Tkinter Dialogs*. (n.d.). Python documentation. https://docs.python.org/3/library/dialog.html

[4]. *Integrating ethical considerations into innovation design*. (n.d.). SpringerLink. https://link.springer.com/chapter/10.1007/978-3-031-08191-0_23

[5]. *Research shows AI is often biased. Here is how to make algorithms work for all of us*. (2021, July 19). World Economic Forum. https://www.weforum.org/agenda/2021/07/ai-machine-learning-bias-discrimination/

[6]. *TensorFlow/tensorflow: An open-source machine learning framework for everyone*. (n.d.). GitHub. https://github.com/tensorflow/tensorflow

[7]. Singhal, M. (2020, July 7). *Object detection using SSD MobilenetV2 using TensorFlow API: Can detect any single class from…*. Medium. https://medium.com/@techmayank2000/object-detection-using-ssd-mobilenetv2-using-tensorflow-api-can-detect-any-single-class-from-31a31bbd0691

[8]. (n.d.). COCO - Common Objects in Context. https://cocodataset.org/

[9]. *Tkinter — Python interface to Tcl/Tk*. (n.d.). Python documentation. https://docs.python.org/3/library/tkinter.html

[10]. (2023, October 11). OpenCV. https://opencv.org/

[11]. (n.d.). NumPy. https://numpy.org/

[12]. *Pillow*. (n.d.). PyPI. https://pypi.org/project/Pillow/

[13]. *PEP 8 – Style guide for Python code*. (2023, December 9). PEP 0 – Index of Python Enhancement Proposals (PEPs) | peps.python.org. https://peps.python.org/pep-0008/

[14]. (n.d.). Python.org. https://www.python.org/

[15]. Ogundepo, O. (2023, August 22). *MLOps checklist – 10 best practices for a successful model deployment*. neptune.ai.

[16]. *Use a pre-trained model*. (n.d.). TensorFlow. https://www.tensorflow.org/js/tutorials/conversion/pretrained_model

[17]. *Just a moment...* (n.d.). ResearchGate | Find and share research. https://www.researchgate.net/publication/220299404_Engineering_Safety_Requirements_Safety_Constraints_and_Safety-Critical_Requirements

[18]. *14764-2006 - ISO/IEC/IEEE international standard for software engineering - Software life cycle processes - Maintenance*. (n.d.). IEEE Xplore. https://ieeexplore.ieee.org/document/1703974

[19]. *830-1998 - IEEE recommended practice for software requirements specifications*. (n.d.). IEEE Xplore. https://ieeexplore.ieee.org/document/720574APPENDICES

APPENDICES

A. Instructions for installing the system

1. Clone the Repo: alshubati99/BeamEye (github.com)

2. Install Needed Dependencies and Libraries, TensorFlow, Tkinter, Custom Tkinter, Pillow, NumPy. In GitHub as 'Requirements.txt'

3. Run the 'BeamEye.py' file to start running the system.

More Documentation will be provided in the GitHub repo.

B. Code for the system

All Code Material and Documentation Can be Found at this repository: alshubati99/BeamEye (github.com)

C. Other relevant material

These are important GitHub Repositories, Notebooks and Resources were referred to when creating this project:

- https://github.com/pushprajkatiyar/person_detection_from_cctv_video
- Train_TFLite2_Object_Detction_Model.ipynb - Colaboratory (google.com)
- Deep learning approaches for pedestrian detection for enhancing safety of autonomous vehicles (youtube.com)
- naisy/train_ssd_mobilenet: Train ssd_mobilenet of the Tensorflow Object Detection API with your own data. (github.com)
- tensorflow/models: Models and examples built with TensorFlow (github.com)
- 1512.02325.pdf (arxiv.org)