B.Sc. (Hons) in Software Development

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

TrafficVision

By
Conor Murphy

April 22, 2024

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

1 Introduction **2**
1.1 Project GitHub Repository . . . 2
1.2 Project Context . . . 2
1.3 Scope . . . 3
1.3.1 Objectives: . . . 4
1.3.2 Deliverables: . . . 4

2 Methodology **6**
2.1 Agile Methodology . . . 6
2.1.1 What is Agile methodology . . . 6
2.1.2 Agile in Use . . . 7
2.2 GitHub Use . . . 8
2.2.1 Branching & Merging . . . 8
2.2.2 Research and Collaboration . . . 8
2.3 Jira . . . 9
2.3.1 Gantt Chart . . . 9
2.3.2 Kanban Board . . . 9

3 Technology Review **11**
3.1 Technology Diagram . . . 11
3.2 Backend Technologies . . . 12
3.2.1 Python . . . 12
3.2.2 C++ . . . 13
3.2.3 YOLO by Ultralytics . . . 14
3.2.4 Flask . . . 15
3.2.5 Django . . . 16
3.2.6 Express . . . 17
3.2.7 MongoDB . . . 18
3.3 Frontend Technologies . . . 19
3.3.1 JavaScript . . . 19
3.3.2 REACT . . . 20
3.3.3 Angular . . . 21
3.3.4 Chart.js . . . 22
3.3.5 Benefits of Charts.js . . . 22
3.4 Hosting . . . 22
3.4.1 Railway . . . 22
3.4.2 Github Pages . . . 22

4 System Design **23**
4.1 System Architecture . . . 23
4.1.1 Introduction . . . 23
4.1.2 Architecture Overview . . . 23
4.1.3 Programming Languages Utilised . . . 23
4.1.4 Detailed Architecture Descriptions . . . 24
4.1.5 Deployment Environment . . . 26
4.2 TrafficVision Application Design . . . 26
4.2.1 User Interface Design . . . 26
4.2.2 Data Model & Schema . . . 26

5 System Evaluation **29**
5.1 Testing . . . 29
5.2 Performance Metrics . . . 30
5.2.1 Processing Speed . . . 30
5.2.2 Performance Bottlenecks . . . 30
5.2.3 Reliability . . . 30
5.3 Objective Evaluation . . . 31
5.3.1 Objectives . . . 32
5.4 Limitations . . . 36
5.4.1 Live Camera Feed . . . 36
5.4.2 Bus Provider Times . . . 36
5.4.3 Time Limitation . . . 37
5.5 Future Work . . . 37
5.5.1 Live Camera Feed & TensorFlow Lite . . . 37
5.5.2 Road Traffic News . . . 37
5.5.3 Accident & Road Closure Reporting . . . 38

6 Conclusion **39**
6.1 Summary of Context . . . 39
6.2 Summary of Objectives . . . 39
6.3 System Evaluation Findings . . . 40
6.4 Future Work Opportunities . . . 40
6.5 Conclusion . . . 41

List of Figures

1 Example: TrafficVision Project Management Tools . . . 10
2 Example: Image of TrafficVision System Architecture/Technology Diagram . . . 11
3 TrafficVision Home Page . . . 27
4 TrafficVision Traffic Page, with Data visualised for a Sunday . . . 28
5 TrafficVision Upload Page where the users pick the time and date the video was recorded as well as its location . . . 28
6 Example: Car being detected using Python machine-learning process 34
7 Example: Bus being detected using Python machine-learning process 34
8 Graph of visualised data for traffic and buses . . . 36

List of Tables

1 Sample of what a data entry looks like within MongoDb . . . 27

1 Introduction

1.1 Project GitHub Repository

GitHub Repository: `https://github.com/ConorPadraigMurphy/FYP`
GitHub Description:

- ConorMurphys-Demo&Docs: This folder contains the video demonstration, A0 Poster and Dissertation belonging to Conor Murphy.
- Conor-Research: This folder contains research that was done by Conor Murphy for the computer vision process portion of TrafficVision, as well as working examples and a program for detection(not including tracking).
- Rohans-Research: This folder contains all research done by Rohan Sikder during the development of the Traffic Vision application.
- backend: This folder contains the express.js server for retrieving the data from the Mongo database for the frontend as well as other files relevant to the backend.
- frontend: This folder contains all of the necessary code for the frontend part of the TrafficVision application and any other relevant files.
- `video_processing`: This folder contains all of the necessary code for the Python computer vision process to allow for the detection and tracking of vehicles for the application.
- README.md: The README contains information on the application as a whole as well as instructions to run the application.

1.2 Project Context

The project aims to develop an innovative application that can efficiently and accurately process traffic videos. This application will allow users to upload traffic videos for analysis, which will help count the number of cars and buses appearing in the video. The analysis will further compare the bus times mentioned in the TFI report with the times that buses actually appear in the video. This will allow the app to determine if the buses were on time or not, and if all the buses arrived at their respective stops as expected.

The collected data will be used to generate two graphs. The first graph will display the traffic congestion at different times of the day, based on the number of

cars appearing in the video. This information will help commuters plan their travel and avoid busy roads. For instance, if the graph shows that traffic congestion is highest during peak hours, commuters can plan their travel accordingly and avoid those hours. This will help reduce traffic congestion, save time, and improve the overall commuting experience.

The second graph will compare the expected bus arrival times and the actual number of buses appearing in the video. This will help commuters and transport authorities determine the reliability of the public transport system. For instance, if the graph shows that the number of buses appearing in the video is lower than the expected number of buses, it indicates that the public transport system is not reliable. This information can be used by transport authorities to improve the public transport system and make it more reliable for commuters.

The project has the potential to greatly improve traffic data collection and analysis. It will provide valuable insights into traffic patterns, improving traffic management and making public transport more reliable. The application can also be used by students, who often face issues with inaccurate bus timings in ATU. The app will provide accurate information on bus timings, which will help students plan their travel better and avoid waiting for long hours at the bus stop.

Overall, the project is an excellent initiative that will benefit commuters, transport authorities, and students alike. It has the potential to make a significant impact on traffic management, public transport, and commuting experience.

1.3 Scope

The term scope refers to the boundaries or limits of a project. It is used to define what is included in the project and what is not. In the context of this application, the scope is related to the specific objectives, deliverables, tasks, and timelines that are defined and agreed upon for completing the project successfully. By defining the project's scope, there will be a clear understanding of what needs to be done, how much time and resources will be required, and what the expected outcome will be. This will help to avoid any misunderstandings or conflicts during the execution phase of the project.

The aim of this project is to develop an application that can efficiently and accurately process traffic videos, count the number of vehicles appearing in the video, generate a graph based on estimated TFI(Transport For Ireland) bus data vs the actual real-world bus data, generate a graph plotting traffic congestion throughout the day, and have an easy-to-use application that gives users useful traffic information and makes the data graphed easy to understand. In the following sections, we will discuss each of these objectives in detail and the methodology we will use to achieve them.

1.3.1 Objectives:

Our project has several objectives, which we will discuss below:

- Create a user-friendly application that is capable of taking uploaded videos and processing them: The primary objective of this application is to provide a user-friendly interface that enables users to upload traffic videos and extract useful data from them. The application should be intuitive and easy to use, allowing users to upload videos quickly and efficiently.
- Enable users to upload traffic videos that get processed for the relevant data: The application should be capable of processing traffic videos uploaded by users, extracting useful data such as vehicle counts and expected bus times.
- Collect the data and present it in two graphs: The application should generate two visually informative graphs that are easily understood by users. One graph should display traffic congestion data plotted on a graph based on times and vehicle counts, while the other graph should compare expected bus arrival times with the actual bus arrival times.
- Provide commuters with insights into traffic patterns: The application should provide commuters with useful insights into traffic patterns, enabling them to plan their travel more efficiently and avoid peak hours.
- Offer bus timing information to users: The application should provide users with bus timing information, allowing them to plan their travel and reduce waiting times at bus stops.

1.3.2 Deliverables:

Our project has several deliverables, which we will discuss below:

- Deliver a functional application that allows users to upload a video, which is processed for the relevant data. The application should be fully functional, allowing users to upload videos and extract useful data from them.
- A process that is used by the application that is capable of counting vehicles accurately in videos uploaded by the user and collects the relevant data: The application should be capable of accurately counting vehicles in videos uploaded by users, collecting relevant data such as vehicle counts and expected bus times.

- Two visually informative graphs that are easily understood: The application should generate two visually informative graphs that are easy to understand, displaying traffic congestion data plotted on a graph based on times and vehicle counts and comparing expected bus times with actual bus times.
- An intuitive user interface: The application should have an intuitive user interface that allows for the easy uploading of videos, viewing of graphs, and overall simple navigation.

In conclusion, by defining the scope, objectives, and deliverables of the project, we can ensure that the project is successful and meets the needs of its users. Our traffic analysis application aims to provide commuters with useful insights into traffic patterns and bus timing information, allowing them to plan their travel efficiently and reduce wait times. The application's primary objective is to be user-friendly, providing an intuitive interface that allows users to upload videos quickly and efficiently while extracting relevant data.

2 Methodology

2.1 Agile Methodology

2.1.1 What is Agile methodology

Agile methodology is an iterative and flexible approach to software development that prioritises adaptability, collaboration, and customer satisfaction. It emerged as a response to traditional, plan-driven development methodologies that often struggled to accommodate changing requirements and priorities. Below are the key principles of agile methodology relevant to the TrafficVision project.

The Key Principles of Agile Methodologies are:

- **Stakeholder Satisfaction**: Agile methodology prioritises stakeholder satisfaction through early and continuous delivery of valuable work. In the case of TrafficVision, the customer can be seen as the team members involved in the development process.
- **Embrace Change**: Agile welcomes the change of requirements even in the later stages of development, which can allow for improvements in the project.
- **Iterative Development/frequent Delivery**: In Agile, the aim is to deliver a working piece of software in increments, whether those increments are weeks or months, with a preference for bi-weekly increments.
- **Collaboration and Communication**: Stakeholders, which in this case are the team members developing the project, must work together on the project.
- **Supportive Environment**: Agile works best when building projects around individuals (team members), giving them the tools and support they need and trusting in them to complete a task/project, etc.
- **Face-to-Face Communication**: Rather than documentation, agile focuses on conveying information to individuals in a face-to-face interaction.
- **Working Increments**: Working increments are the primary measure of progress.
- **Sustainable Development**: Promotes a sustainable development environment so that stakeholders and team members can maintain a constant pace of work or as close as possible.
- **Technical Expertise**: It is important to consistently focus on technical expertise and make sound design choices.

- **Simplicity**: The ability to minimise unnecessary work is essential.
- **Self-Organising Teams**: The best architectures, requirements, and designs emerge from self-organising teams.
- **Reflection and Adjustment**: The team regularly reflects on its performance and adjusts its behaviour accordingly to become more effective.

2.1.2 Agile in Use

During the development of the TrafficVision project, the team made an effort to utilise as many of the key principles of agile methodologies as was feasible. This is how several of the methodologies were applied:

- **Iterative Development/Frequent Delivery**: During the development of the TrafficVision application, Jira's Gantt charts and Kanban boards were utilised in order to break down the project into small, manageable increments. This allowed for each task to be planned out and discussed thoroughly before implementation, ensuring that all potential issues were addressed ahead of time. Once each increment was complete, it was tested to make sure that it met the necessary requirements and functionality. This approach allowed for a more efficient and effective development process.
- **Collaboration and Communication & Stakeholder Satisfaction**: Regular communication with the project supervisor was maintained throughout the project development process. The meetings involved discussing project ideas and progress, and receiving feedback. Such interactions with the supervisor and team members were deemed crucial forms of stakeholder communication and collaboration.
- **Face-to-Face Communication**: Throughout the project, frequent face-to-face communication took place through weekly meetings with both the project supervisor and members of the team. These meetings provided a platform for effectively communicating information and ideas to all stakeholders involved. Through weekly meetings, information was conveyed in a clear and concise manner, which helped us avoid the need for extensive documentation. This approach facilitated an efficient and effective development process. Prioritising communication through regular meetings rather than conveying all information through documentation.
- **Embracing Change**: The project underwent several changes throughout its development lifecycle. The team carefully considered and discussed each modification before implementation, welcoming proposed changes. Despite

the challenges that arose, the team remained committed to delivering a high-quality product by staying flexible and open-minded. Ultimately, the team's collaborative and solution-oriented approach led to the project's success.

- **Continuous Delivery**: With the aid of software solutions such as Jira's Gantt and Kanban boards, software was delivered seamlessly during the project's development. The project was divided into tasks to facilitate iterative development and continuous delivery.

2.2 GitHub Use

GitHub is a widely used and powerful platform for collaborative software development and version control. It provides developers with a centralised space to host, manage, and collaborate on code repositories. GitHub uses the Git version control system, which enables users to track changes, manage branches, and collaborate seamlessly. TrafficVision utilised GitHub in two main ways during its development.

2.2.1 Branching & Merging

The team successfully applied the agile principle of iterative development by utilising GitHub's branches. Whenever a team member undertook a task, they created their own branch, separate from the main code branch. This practice prevented any major issues from arising with the working code when implementing new features. Furthermore, it ensured that no development on the same functionality or process would interfere with another team member's code. Once the team member completed their task, they would merge their changes back into the main code branch, which allowed for seamless integration of the new feature or functionality.

2.2.2 Research and Collaboration

GitHub is not just a platform for technical purposes but also serves as a central hub for collaboration and research within our development ecosystem. Apart from hosting code repositories, GitHub also serves as a documentation space for discussions, research findings, and insights guiding decisions throughout the development cycle. This use of GitHub allows developers to make informed choices, maintain project alignment, and encourage a collaborative environment. The importance of this approach was evident during development, where team members could actively track each other's progress, making information readily accessible for issue resolution and discussions. This transparency ensured informed decision-making and facilitated a more cohesive and interconnected development process.

2.3 Jira

During the development process, two pieces of Jira software were used, which have already been mentioned in this chapter. One being Jira's Gantt chart and the other being their Kanban board. Let's discuss:

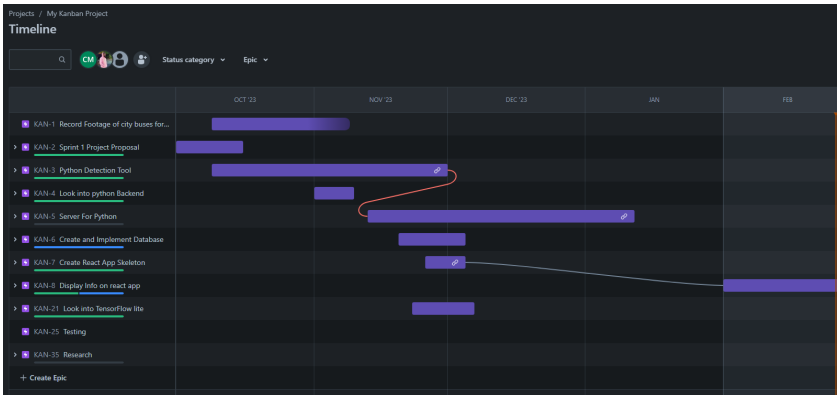
2.3.1 Gantt Chart

A Gantt chart is a visual representation of a project schedule that shows how long each epic will take from start to finish. An epic is a piece of functionality that needs to be implemented and contains smaller iterative tasks. In Figure 1a, you can see an example of the TrafficVision Gantt chart. It displays all the tasks from the beginning of development and how one epic flows into another.

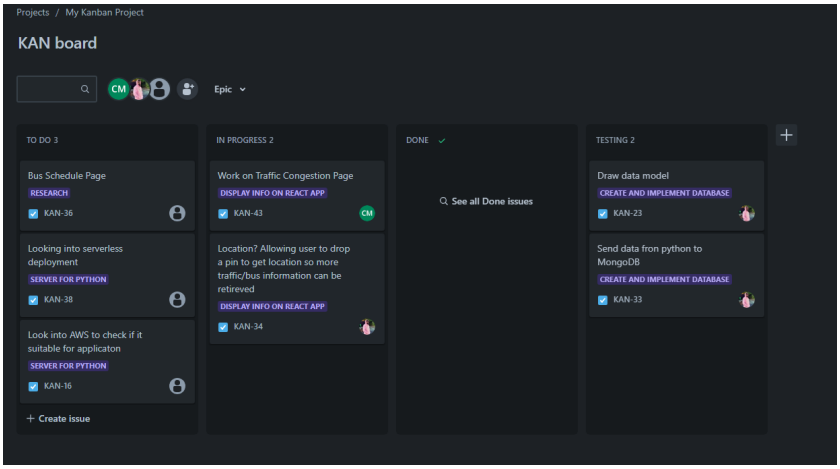
2.3.2 Kanban Board

A kanban board is a useful visual management tool that helps optimise workflow within a development team. It offers a clear visual representation of the work that needs to be done in the current sprint and who is responsible for each task. In the attached image, which you can see in 1b, you will find a screenshot of the kanban board used during the development cycle of the TrafficVision application.

Each column on the board represents a different category of tasks. The first column, "To-Do", contains tasks that have not yet been assigned to a member of the team. Any member of the development team can pick up tasks from this column. Once a task is assigned, it is moved to the "In-Progress" column. Finally, when a task is completed, it is moved to the "Done" column, which indicates that it is ready to be tested.



(a) Example: Image of TrafficVision Gantt Chart



(b) Example: Image of TrafficVision Kanban board

Figure 1: Example: TrafficVision Project Management Tools

3 Technology Review

3.1 Technology Diagram

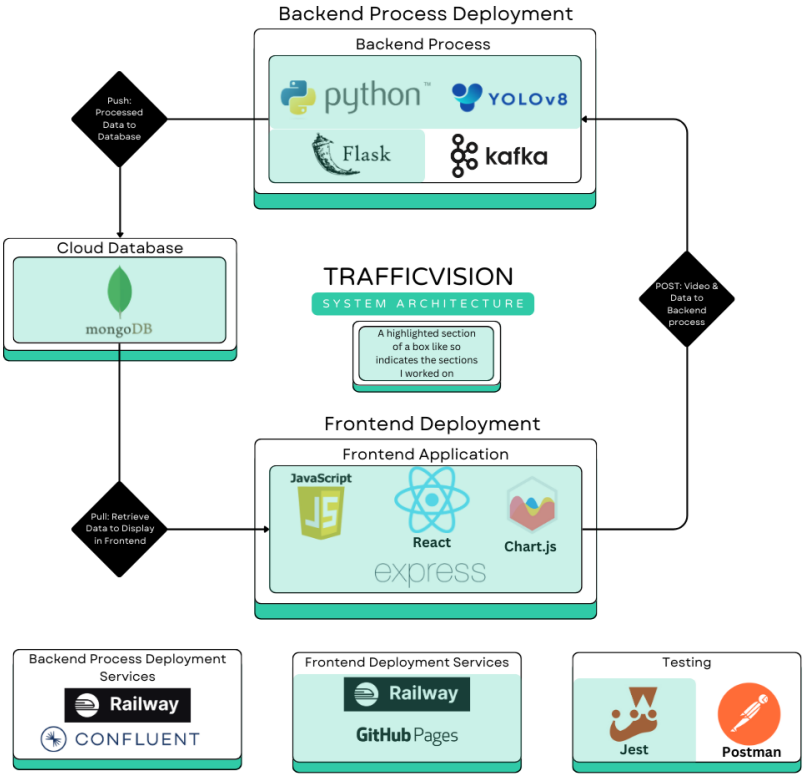


Figure 2: Example: Image of TrafficVision System Architecture/Technology Diagram

The system architecture diagram 2 above displays the core technologies that are integrated into the TrafficVision application. In the following sections, the selected technologies will be discussed in detail, along with the reasons for choosing them over other available technologies. This will provide a comprehensive understanding of the strengths and advantages of these technologies and how they contribute to the successful implementation of the TrafficVision application. The discussion will cover additional technologies that have been utilised in the TrafficVision application in addition to the core technologies mentioned previously. These technologies are not listed in the 2 depicting the system architecture.

3.2 Backend Technologies

3.2.1 Python

Python, which was chosen for the development of the TrafficVisions computer vision tool, is a widely used programming language known for its simplicity and readability. Guido van Rossum developed it. Python is a versatile language that is used in various areas, including computer vision. When it comes to computer vision, Python is the preferred programming language due to its wide availability of libraries and frameworks designed for image processing and analysis. Some of these include OpenCV, TensorFlow, and PyTorch, which are known for their exceptional support for computer vision tasks. OpenCV offers an extensive set of tools for image and video analysis, while TensorFlow and PyTorch provide robust machine-learning capabilities. Python's flexibility and the availability of specialised libraries make it an ideal choice for solving diverse challenges in computer vision, from simple image processing tasks to more complex deep learning-based algorithms. [1] [2]

Advantages of using Python for Computer Vision

- **Availability of Libraries**: Python offers a wide variety of libraries and frameworks focused on computer vision tasks, such as OpenCV, scikit, PyTorch and TensorFlow, which provide a wide variety of functions and algorithms for image processing, object detection and object tracking.
- **Machine Learning Integration**: Python integrates with machine learning frameworks like TensorFlow, PyTorch, and Yolo via Ultralytics. This enables the use of advanced computer vision techniques, such as CNNs, for object detection, tracking, and classification.
- **Community Support**: Python has a large community of developers who specialise in computer vision. This community actively contributes to open-source projects, tutorials, and documentation, providing an abundance of resources for new developers in the field of computer vision.
- **Simplicity and Readability**: Python is a programming language that offers an easy-to-understand syntax, making it simple for developers to write and comprehend code. Its readability means that developers can concentrate on creating the computer vision functionality without getting bogged down in complex syntax issues.

Disadvantages of using Python for Computer Vision

- **Performance**: Python can be great for development, but it may not be as performant as lower-level languages such as C++ and Java. This could be a concern when developers are dealing with processing very large datasets in real-time.
- **Memory Consumption**: Python dynamic typing and garbage collecting techniques can lead to higher usage of memory compared to statically typed languages. This may become an issue when dealing with memory-hungry computer vision applications/tasks.
- **Deployment Complexity**: Occasionally, when deploying a Python computer vision application, there can be challenges with dependency issues, versioning issues and runtime overhead. Containerisation can help alleviate this issue but requires careful management and setup.

3.2.2 C++

C++, which was considered for the TrafficVision computer vision but ultimately was not chosen, is a language known for its performance and versatility. It offers low-level control but high performance, which makes the language well-suited for computationally complex tasks like those seen in computer vision.

Advantages of using C++ for Computer Vision

- **Performance**: C++ is a compiled language that offers developers high performance and low overhead, which makes it great for computationally complex computer vision tasks.
- **Low-Level Control**: C++ provides direct access to available hardware resources and memory management, allowing the developers using it to optimise their functionality and algorithms for specific hardware/architectures.
- **Libraries**: C++ offers libraries and tools to assist in the development of computer vision applications. Although it may not offer as extensive amount of libraries and tools as Python does, it does offer libraries such as OpenCV and Dlib that provide a wide variety of functions and algorithms for image processing, detection and tracking.
- **Versatility Across Platforms**: C++ can be compiled and run on various OS and hardware platforms, which makes it suitable for deploying computer vision applications across different environments.

Disadvantages of using C++ for Computer Vision

- **Learning Curve**: For developers new to development in computer vision, C++ can have more of a learning curve when coming from a higher-level language such as Python as its syntax, as well as its features, require a deeper understanding, which in turn requires more time to learn.
- **Documentation and Community**: C++ does have a large community of developers and enthusiasts, although documentation can sometimes be less comprehensive compared to other languages that are used for computer vision development, such as Python.
- **Development Time**: C++ can take up more time due to the fact that a lot more manual coding has to be done, especially so compared to languages like Python.

3.2.3 YOLO by Ultralytics

Who are Ultralytics? Ultralytics is a company founded by Glenn Jocher. It focuses on developing cutting-edge AI models, particularly for computer vision tasks like object detection, object tracking and classification, and more. More important for this paper is that they are the creators of the YOLO(You Only Look Once). [3] [4] [5]

YOLO Version 8
Ultralytics YOLO v8 is the newest iteration of the YOLO model. It builds upon the success of previous versions to further enhance the model's performance and flexibility. Unlike some traditional models that take a multi-stage approach to detection, YOLOv8 take a single-stage approach, which makes it much faster. It uses a single-stage approach by analysing an image or frame of a video just once while at the same time predicting bounding boxes around the objects in the image or video frame while classifying them. Even though YOLOv8 is known mainly as a detection model, its capabilities go beyond that of detection. It offers developers other functionality such as:

- **Classification**: YOLO can detect objects while also precisely classifying them.
- **Pose Estimation**: The posture and orientation of an object within an image or frame can be estimated.

- **Tracking**: YOLOv8 can be used to track objects across frames, allowing developers to develop basic tracking applications.

Advantages of YOLOv8

- **Speed**: YOLOv8 has excellent processing speeds thanks to its single-stage approach, which makes it ideal for real-time applications as well as applications like self-driving cars
- **Accuracy**: Despite the speed at which YOLO works, it maintains excellent accuracy for detention and classification.
- **User-Friendly**: Ultralytics focuses on making its framework and model easy to use. YOLO comes with a Python package and command line interface, making it accessible to developers and researchers from beginner level to expert.
- **Versatility**: YOLO is not just used for object detection. It can be utilised for other applications where classification, pose estimation, and tracking may be needed. Which can make a perfect choice for most computer vision needs.

Disadvantages of YOLOv8

- **Small Object Detection**: While YOLO can be very accurate for most objects, YOLOv8 may struggle with detecting very small objects within an image or frame, especially so in images or video frames with occlusions.
- **Computational Requirements**: Although YOLOv8 is fast and has great accuracy, it still needs quite a bit of processing power, which could be a limitation, depending on the device.

3.2.4 Flask

During the development of the TrafficVision application, the Python backend utilised Flask as the web framework. Flask is widely known for its simplicity and flexibility, which made it an ideal choice for the Python side of the TrafficVision application. Below, you will find some of the advantages and disadvantages of using Flask. [6]

Advantages of using Flask

- **Lightweight and Minimalistic**: Flask is a microframework, meaning it has a small core that keeps things nice and simple, which makes it great for beginners.
- **Scalable**: Applications can quickly grow from their small beginnings, so Flask allows you to add extensions for extra functionality for database and authentication capabilities.

Disadvantages of using Flask

- **Security**: Security is of the utmost importance for web applications. Flask provides tools to build secure applications, but it places a lot of responsibility on the developers of the application to make sure that security features are implemented properly.
- **Lack of Built-in Features**: Although Flask is flexible, it comes with a minimal amount of features out of the box. This means developers have to add extensions for extra functionality, which could get unwieldy as an application grows.

3.2.5 Django

Early in the development of TrafficVision, Django was considered a framework for that application, but ultimately, it was not chosen as it was decided that Flask was more lightweight and perfectly suited to the application's needs. Although, Django is a full-featured framework, which means that it comes with a lot of functionality right out of the box, which can help developers get straight to work and save time.

Advantages of using Django

- **Security**: Django provides built-in features for authentication, authorisation, and security against common web vulnerabilities.
- **Scalability**: Django is suitable for larger and more complex web applications. It also provides additional features to assist with scalability, such as database scaling and performance optimisation.
- **Structure and Organisation**: Django offers a well-organised project structure that helps developers maintain code consistency, particularly in larger teams.

Disadvantages of using Django

- **Learning Curve**: Due to how comprehensive Django is, it can have a deeper learning curve compared to other frameworks like Flask, which could delay development for newer developers.
- **Overhead**: Due to the larger feature set that Django provides, it can introduce more overhead, which may make it less suitable for smaller-scale projects.

3.2.6 Express

Express.js was used in the TrafficVision application to pull data from the database to display to the users of the application. Express is a commonly used web framework built on top of Node.js. It provides features for building APIs and web applications. Below you can see some of expresses advantages and disadvantages. [7] [8]

Advantages of Express

- **Simplicity and Minimalism**: Express is simple and lightweight and provides the essential building blocks that are needed to develop a web application.
- **Large Community and Support**: Express has a great community of active developers, which means there are plenty of resources such as documentation, libraries, and tutorials to be found online to help experts and beginners with Express.
- **Performance**: Express is built on top of Node.js, and because of this, Express benefits from the asynchronous and non-blocking I/O model. This makes it very efficient at handling larger numbers of simultaneous requests, which leads to high-performance web applications.

Disadvantages of Express

- **Lack of Built-in Features**: Unfortunately, Express does not come with a lot of additional functionality out of the box. It lacks several features that are common among other web frameworks, such as authentication, validation, and authorisation.
- **Security**: Unfortunately, Express has fallen victim to several security vulnerabilities over the years. Because of this, developers are left to rely on the best practices when developing and expressing applications.

- **Complexity at Scale**: Express is a good choice for developing small to mid-sized applications. However, for larger and more complex applications, it may not be the best option. This is because Express lacks certain tools and features that can aid developers in handling scale and complexity.

3.2.7 MongoDB

MongoDB is a document-oriented database that uses JSON-like documents to store data. These documents can have a flexible schema, which means that depending on the data you store, different fields can be allowed in one collection. This makes it a good fit for storing data that doesn't fit into a tabular structure or data that may change frequently. [9] [10]

Advantage of MongoDB

- **Flexability**: MongoDB's document-oriented structure allows developers to easily represent complex data models by storing data in a way that resembles real-world entities.
- **Performance**: Compared to some relational databases, Mongos performance optimisation is much simpler thanks to its architecture and the way it manages data internally.
- **Sharding**: Developers can horizontally expand their databases by distributing data across multiple servers using Mongo. This process, called sharding, ensures efficient data retrieval through data distribution and separation across servers.

Disadvantages of using MongoDB

- **Limited ACID Compliance**: MongoDB provides atomicity, consistency, isolation, and durability (ACID) guarantees at the document level but lacks support for ACID transactions across multiple collections or documents.
- **Limited Joins**: Performing complex joins across different collections in MongoDB can be challenging and inefficient compared to traditional databases. Although it is possible to execute more complex join-like queries through code, doing so can slow down performance.

3.3 Frontend Technologies

3.3.1 JavaScript

JavaScript, also written as JS, is a high-level interpreted programming language. This means that it is written in a way that closely resembles human language, making it easier for developers to pick up and work with. JS is translated and executed line by line within the web browser, which allows for quicker development cycles. It also enables developers to create dynamic applications with static web pages, resulting in a more interactive experience. That is why JS was chosen for the frontend of the TrafficVision application.

Advantages of using JavaScript

- **Easy to Learn**: JS has a simple syntax, making it great for beginner developers to get up and running with their first web app quickly.
- **Universal Language**: JavaScript is also a universal language. In the context of JavaScript, this means that it is supported by all the modern web browsers we use today and will work consistently across multiple different platforms.
- **Abundance of Packages**: JavaScript has a massive collection of packages offering lots of different functionality for tasks such as animations, UI components and network requests.
- **Frameworks**: JavaScript also has greater frameworks such as React, Angular and Vue that have an abundance of prebuilt components and functionality to help developers streamline the development process.

Disadvantages of using JavaScript

- **Performance**: Sometimes, developing complex JavaScript code can slow down web page loading. However, there are several optimisation techniques developers can employ to maintain good performance.
- **Security Issues**: In JavaScript, security is a concern as bad actors can inject malicious code into a web page to exploit vulnerabilities which may compromise an application's functionality or sensitive data.

3.3.2 REACT

React is a popular JavaScript library used in web development. It was utilised to build the UI of the TrafficVision application. React divides the UI into reusable components. Each of these components handles its own state and logic, making the code easier to comprehend and more organised for developers working on it.[11] [12]

Advantages of using React

- **Performance**: React's virtual DOM and component-based architecture allow for optimised rendering, enhancing the application's responsiveness and user experience.
- **Reusable Components**: React utilises reusable components, which helps in making the code modular and simplifies code management for developers.
- **Community and Support**: React has a huge developer community, which results in React having an abundance of support in terms of learning and libraries along with docs and tutorials that help beginner and expert developers with making their first React web app.

Disadvantages of using React

- **Security Issues**: Just like any JavaScript code, security vulnerabilities can exist in React applications if developers don't follow secure coding practices and stay up to date with the latest security practices to make sure that bad actors don't exploit and vulnerabilities in a React application.
- **Learning Curve**: For developers who are new to JavaScript, react can have a somewhat steep learning curve as it requires previous knowledge of JavaScript, HTML, CSS, as well as the React library itself.
- **React Updates**: As React is a very popular technology, it is always being improved upon. This means developers generally have to keep up with the latest updates to the React library, which may cause delays and challenges when it comes to maintaining existing applications.

3.3.3 Angular

Another framework that could have been chosen for the frontend of the TrafficVision application is Angular. Angular is a JavaScript framework built on TypeScript. It is very popular among developers in web development and is a strong alternative to React. Angular enforces a Model-View-Controller architecture, which can be considered an advantage or disadvantage depending on the application's needs. [13]

Advantages of using Angular

- **Performance**: Similar to React, performance is a very high priority for Angular. It utilises a technique called ahead-of-time compilation (AOT). This allows for the translation of code into more efficient machine code before the browser runs it; in turn, this can lead to much faster load times and a better user experience.
- **Code Architecture**: Angular MVC architecture implements a clear distinction between the model, view and controller. This structure of Angular promotes organised and maintainable code among the developers of Angular applications.
- **Community and Support**: Much like React, Angular has a large active community, which gives developers access to an abundance of docs, libraries and tutorials to help developers with their applications no matter what level of expertise they have.

Disadvantages of using Angular

- **SEO Capabilities**: In Angular, client-side rendering can sometimes introduce issues for SEO(Search Engine optimisation). Specifically, search engines can have a difficult time indexing content that has been dynamically generated on the client side.
- **Complexity at a Small Scale**: When developing smaller basic applications, Angular MVC architecture might make it more challenging compared to Reacts component-based approach to web applications.

3.3.4 Chart.js

Chart.js is a free, open-source JavaScript library that gives the ability to developers display data using various different graphs such as bar charts, line graphs and pie charts. [14]

3.3.5 Benefits of Charts.js

- **Ease of Use**: Charts has easy-to-understand syntax, making it accessible to both beginner and expert developers.
- **Browser Support**: Charts operates across different web browsers, which ensures that no matter the browser being used, Charts looks consistent across all of them.
- **Community Support**: Much like the other technologies that have been discussed, Charts too shares a large active community of developers, providing an abundance of documentation, tutorials and plugins that provide charts with extra functionality.

3.4 Hosting

3.4.1 Railway

Railway is a modern platform used to deploy web applications. In the case of TrafficVision, it has been used to host the express server that enables the react frontend to display data using chart.js.[15]

3.4.2 Github Pages

GitHub Pages, which is integrated with GitHub, is a great way to host a React application's frontend user interface.[16]

4 System Design

4.1 System Architecture

4.1.1 Introduction

The TrafficVision application has been designed to process data from a video that has been uploaded by a user of the application with the help of machine learning to extract the relevant data, which is then displayed on the frontend user interface.

4.1.2 Architecture Overview

The TrafficVision application consists of the following key technologies:

- **React**: The React frontend is responsible for all of the user interaction in the TrafficVision application. Without the React user interface, users would not be able to upload the videos needed to collect the relevant data and would not be able to view the data after it has been processed.
- **Flask & Kafka**: Sends the video to the Python machine learning process so that it can be processed for the relevant data.
- **YoloV8 by Ultalytics**: YoloV8 by Ultralytics is used with Python to create the Python machine learning process that retrieves the necessary data from the uploaded videos.
- **MongoDB**: Stores the data that gets processed by the Python machine learning code.
- **ExpressJS Server**: Retrieves the data from MongoDB and serves it to the frontend user interface, which gets visualised using Chart.js.
- **Railway**: Railway was utilised to host the express.js server.
- **GitHub Pages**: GitHub pages was utilised to host the React Frontend.

4.1.3 Programming Languages Utilised

- **Frontend**: JavaScript
- **Backend - Express.js**: JavaScript
- **Backend - Flask/Kafka**: Python
- **Machine Learning Process**: Python

4.1.4 Detailed Architecture Descriptions

React Frontend

- **Functionality**:
  - When the user interacts with the React frontend, it dynamically renders its components to facilitate the different functionality that the TrafficVision application provides.
  - The React frontend utilises JavaScript as well the React library to create a responsive user interface.
- **Connectivity**:
  - **Video Upload Flow**: When a user selects a video for upload a requests are initiated to the Flask server for video processing.
  - **Data Visualisation**: The data is received from the Express.js server and is visualised with the use of the charts.js library.

Flask & Kafka

- **Functionality**:
  - The Flask acts as a backend server for receiving uploaded videos from the React frontend to be processed by the Python machine learning process. And Kafka facilitates the real-time messaging between the frontend and backend.
- **Connectivity**:
  - **Video Upload Handling**: Listens for a video being uploaded from the React frontend. When it receives an upload, it is sent to the Python machine-learning code to be processed.
  - **Kafka Integration**: The Flask is integrated with Kafka to send the videos to the Python machine learning process. Kafkas messaging queues are utilised to ensure fault-tolerant and scalable communication.

Python Machine Learning Process

- **Functionality**:
  - Performs the object detection and tracking on the video that was uploaded using Ultralytics YoloV8. The relevant data is also extracted from the video and sends the relevant data to the mongo database.
- **Connectivity**:
  - **Kafka Integration**: Listens for the uploaded video, and once received, the Python machine learning process processes the video.
  - **MongoDB**: Once the video has been processed for the relevant data, the data gets sent to the MongoDB collection.

MongoDB

- **Functionailty**:
  - MongoDB serves as the database for the TrafficVision application. It is utilised for storing the data that the Python machine-learning code extracted from the uploaded video, and it provides very flexible and scalable storage, which is perfect for this application.
- **Connectivity**:
  - **Data Storage**: The Mongo collection receives the data sent from the Python code and stores it in a structured format.
  - **Data Retrieval**: With the use of Express.js, the React frontend can fetch data from the Mongo database so that it can be visualised.

Express.js Server

- **Functionality**:
  - Acts as the backend server for retrieving the processed data from the Mongo database and serving it to the React frontend where it gets visualised.
- **Connectivity**:
  - **Data Retrieval**: Fetches the processed data from the Mongo database in response to a request made by the React frontend.
  - **Data Serving**: Sends the retrieved data to the React frontend to be visualised using charts.js.

4.1.5 Deployment Environment

Railway

- **Functionality**: Railway is the platform used to host the TrafficVision Express.js backend server. This server provides the application with a scalable and easy way to deploy backend services. Whenever the Express.js server is updated, and the updated version is pushed to the main GitHub branch, Railway automatically picks it up and deploys the newer version of the server.

GitHub Pages

- **Functionality**: GitHub Pages serves as a PaaS (Platform as a Service) that hosts the TrafficVision React frontend, making it accessible to users from anywhere. It offers a simple and direct approach to hosting the application frontend. This is achieved by hosting a build of the frontend on a specific GitHub branch called gh-pages. To update the frontend, a developer must create a new version of the build, commit it, and push it to the gh-pages branch. Once done, the new version of the frontend will be automatically deployed.

4.2 TrafficVision Application Design

4.2.1 User Interface Design

The TrafficVision application's user interface is designed to provide an intuitive and user-friendly experience. As previously discussed, the frontend is built using the React library, which offers a simple and responsive layout. This layout enables users to upload their videos to the application and view visualised data with chart.js. The UI components are organised in a clear and structured way, providing users with an easy method of navigation.

4.2.2 Data Model & Schema

MongoDB is used as TrafficVision's database system. Mongo provides the application with a flexible and scalable storage solution for storing the various types of data that the TrafficVision application needs to store. The data stored in Mongo is the data that the Python machine-learning process extracts from the uploaded videos.

The table above showcases a typical data entry stored within MongoDB for the TrafficVision application. Each data entry contains essential information such as the object ID, class ID, timestamp of entry, direction of movement, geographical address, latitude, and longitude coordinates. The structured format ensures that

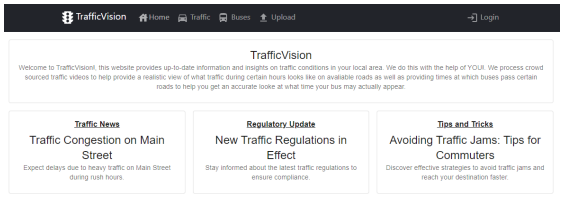


Figure 3: TrafficVision Home Page

Data Name	Sample Data	Data Type
object_id	1	Int32
class_id	"Bus"	String
entered_time	2024-03-31T14:30:41.000+00:00	Date
direction	"Right"	String
address	"College Rd, Galway, Ireland"	String
latitude	"53.2788461"	String
longitude	"-9.0381477"	String

Table 1: Sample of what a data entry looks like within MongoDb

frontend components can effectively interpret and visualise data to users in an easy-to-understand format. For instance, the class ID provides insights into the type of object detected, allowing users to identify and analyse different types of traffic elements accurately. Timestamps, coordinates, and addresses enable users to track the movement patterns of objects over time and visualise data across locations and times. Moreover, the consistent data type formatting ensures seamless integration with frontend visualisation libraries.

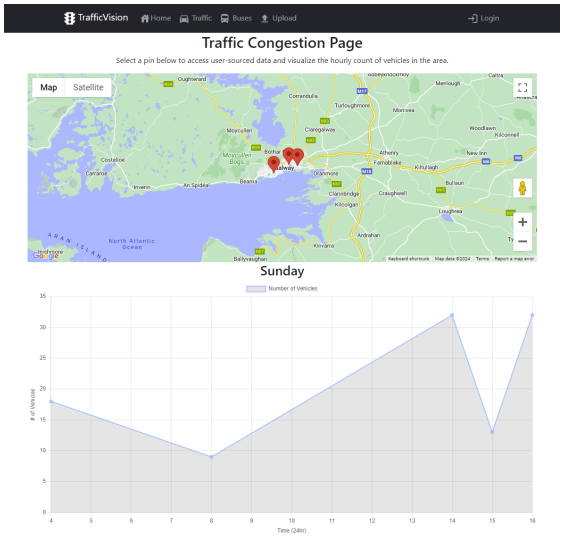


Figure 4: TrafficVision Traffic Page, with Data visualised for a Sunday

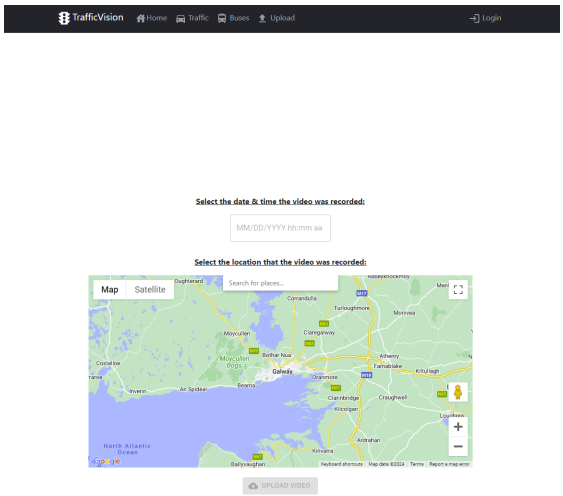


Figure 5: TrafficVision Upload Page where the users pick the time and date the video was recorded as well as its location

5 System Evaluation

5.1 Testing

Python Machine-Learning Manual Testing

During the development phase of the Python machine learning process, a lot of manual testing was done to ensure that the correct results were being extracted from the process. It was also important to make sure that the tracking and detection algorithm was working correctly. This involved verifying that only the necessary objects were being extracted from the data. To achieve this, test videos were processed and watched to ensure that the objects were being picked up correctly. Apart from verifying that the objects themselves were being detected and tracked accurately, it was also essential to ensure that the extraction of the additional data (as shown in 4.2.2) was being done correctly. Initially, this was a bit of a tedious process, but as the development progressed, it became less time-consuming.

React Frontend - Manual Testing

During the development process of TrafficVisions React Frontend, various testing methods were employed to ensure that the application was user-friendly and easy to navigate. One of these methods was manual testing of UI components. By conducting manual testing, it was ensured that TrafficVisions React Frontend was intuitive and easy to use for all users, regardless of their technical expertise. This approach helped create a positive user experience, which ultimately led to increased user engagement and satisfaction with the application.

Jest Tests

Jest is a popular testing framework used to test JavaScript applications, specifically in frontend development. In the case of the TrafficVision application, Jest was utilized to ensure that frontend components were functioning as expected. For instance, the Upload page of the application was tested with Jest as it was the most crucial area of the application where there was a higher risk of issues arising. By utilizing Jest, developers at TrafficVision were able to write test cases that could be run automatically, which helped catch any issues before they became major problems.

5.2 Performance Metrics

5.2.1 Processing Speed

- **Frame Processing Rate**: During the Python machine-learning process of uploaded videos, the Frames Per Second (FPS) at which the video is being processed is outputted in the window that appears while the process is running. Although this information isn't displayed for the user, it is useful during the development phase to measure the rate at which the video frames are being processed.

- **Algorithm Execution Time**: The time that it takes the YOLOv8 detection and tracking algorithm to execute is largely dependent on the length of the video that is uploaded, e.g. a five-minute video uploaded will be processed much quicker than a twenty-minute video. But another factor that influences it is the CPU of the machine that the Python machine-learning process is being run on.

5.2.2 Performance Bottlenecks

As briefly mentioned in the 5.2.1, a factor that can influence the performance of the Python machine-learning is the hardware of the machine that the process is being executed on. As of writing this, the Python machine-learning process utilises the machine's CPU. The process has been tested on three different machines, all of which ran the process reliably and efficiently. Although this process utilises the machine's CPU, it can be configured with some work to utilise a machine's GPU as well. However, during development, this method was not necessary to execute the process, so it was not pursued. With this information, we can suggest to team members that this process be hosted on a system that is within the ranges specified.

5.2.3 Reliability

Algorithm Reliability

- Throughout the development of the Python machine-learning process that uses YOLOv8 for detection and tracking, it was tested with various different videos which were recorded in different conditions, such as Day time videos, nighttime videos, videos that had occlusions such as hedges, trees and people walking back in forth in front of the camera. All of these videos also varied in conditions such as lighting, background objects and object sizes. Under all of these conditions, the Python machine-learning process performed very well.

- Although the machine-learning process performs well when put under various conditions, there are some conditions where issues may arise when trying to detect and track objects. An issue like this may arise when there are large occlusions that completely block the view of traffic, which would cause the algorithm not to be able to extract the objects out of the image. Or if a video is taken at a very poor camera angle, which may make detection and tracking difficult for the algorithm, although, in testing, various different camera angles were tested without running into this issue, that's not saying it is impossible for it to happen. Lastly, there is extremely poor camera quality, whether it may be the age of the camera or damage that may have occurred to a camera at some point. It may cause distortions that may negatively affect the machine-learning process.

- Another aspect that was important to consider when developing the Python machine-learning portion of the TrafficVision application was making sure that only the necessary objects were being detected and tracked. Luckily, YOLOv8 allows for the selection of what objects you want to detect and track, which ensures that the machine-learning process is only getting the specified objects.

React Reliability

- When developing the React frontend of the TrafficVision application, it was important to ensure that users were provided with a reliable frontend that guaranteed to work. To do this, a clean user interface was developed with a navbar leading to other components of the application that were clearly labelled, which allowed for easy navigation. On every page, clear instructions were listed for users to understand how they can visualise the data they would like to view as well as how to upload a video to the Python machine-learning process.

5.3 Objective Evaluation

In order to evaluate the overall effectiveness of the TrafficVision application, it is necessary to revisit the set of objectives established at the beginning of the development phase and assess what has been achieved. The application needs to be assessed against the objectives to determine its effectiveness in meeting the desired goals. This will help identify the strengths and weaknesses of the application and highlight areas where improvements are necessary to ensure that it meets the requirements that were set out. Therefore, conducting a critical assessment of the TrafficVision application is essential to ensure that it delivers the intended results and provides value to its users.

5.3.1 Objectives

Objective 1

- **Objective**: Create a user-friendly application that is capable of taking uploaded videos and processing them. The primary objective of this application is to provide a user-friendly interface that enables users to upload traffic videos and extract useful data from them. The application should be intuitive and easy to use, allowing users to upload videos quickly and efficiently.

- **Objective Evaluation - User Interface**: During the development phase of the TrafficVision application, it was crucial to have a working machine-learning process before starting the frontend development. This was done to ensure that the user interface was clean, intuitive, and user-friendly. The goal was to make it easy for users to interact with the application, even if they didn't have a deep understanding of different technologies. The user interface for TrafficVision has successfully met all of these objectives and provides the desired data to users.

- **Objective Evaluation - Data Extraction**: At the beginning of the development of the TrafficVision application, the Python machine-learning process was heavily focused on. This process was crucial as it extracted the necessary data from the videos uploaded by users, which was then visualized on the user interface. Without this process, the application would not have been able to function properly, as the data extraction from the videos was the backbone of the application. Throughout the development phase, the machine-learning process underwent many changes and modifications to ensure that it extracted only relevant data from the videos. The process was fine-tuned so that the data that was outputted was accurate and useful. The latest version of the process was identified as the most suitable one to use, as it was the most efficient and reliable. It was made sure that the process was tested before implementing it in the final version of the application. Overall, the Python machine-learning process was a critical component of the TrafficVision application. Its importance cannot be overstated, as it was the foundation upon which the application was built. Without the process, the application would not have been able to deliver the traffic data that it does today.

Objective 2

- **Objective**: Enable users to upload traffic videos that get processed for the relevant data: The application should be capable of processing traffic videos uploaded by users, extracting useful data such as vehicle counts and expected bus times.
- **Objective Evaluation - Upload Page**: When it comes to the TrafficVision application, one of the most crucial aspects is the upload page. This is where users can upload videos that can be processed by the Python machine-learning system. The data extracted from the videos can provide valuable insights and help with traffic analysis, but it's equally important to know where and when the video was recorded. This additional data can provide context and help make the analysis even more accurate. To capture this information, the upload page of the TrafficVision application has been designed to gather the date, time, and location of the video by having the user input this information. Once this information is collected, it is sent to Python to be used by the Python machine-learning process. This ensures that all the relevant information is available for further analysis and can be used to generate visualisations later on the user interface.
- **Objective Evaluation - Extracted Data**: After a user uploads a video they recorded to the application's upload page, along with the video's time, date, and location details, the Python machine-learning process extracts and visualizes data for the user on the frontend. The process extracts important information such as the type of vehicle detected, the time the vehicle entered the frame, and the direction of the vehicle. This information is then used to calculate the time the vehicle was in the scene. The database contains a log for each vehicle, which enables the display of car counts on a particular road, day, and time for the user on the frontend. An example of how the vehicles are detected can be viewed in figure: 7 & figure: 6

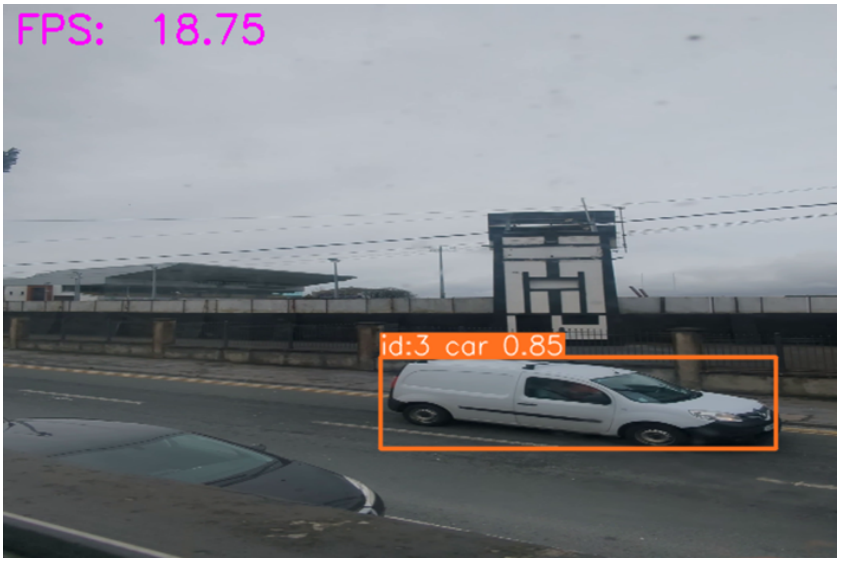


Figure 6: Example: Car being detected using Python machine-learning process

Figure 7: Example: Bus being detected using Python machine-learning process

Objective 3, 4, 5

- **Objective 3**: Collect the data and present it in two graphs: The application should generate two visually informative graphs that are easily understood by users. One graph should display traffic congestion data plotted on a graph based on times and vehicle counts, while the other graph should compare expected bus arrival times with the actual bus arrival times.

- **Objective 4**: Provide commuters with insights into traffic patterns: The application should provide commuters with useful insights into traffic patterns, enabling them to plan their travel more efficiently and avoid peak hours.
- **Objective 5**: Offer bus timing information to users: The application should provide users with bus timing information, allowing them to plan their travel and reduce waiting times at bus stops.
- **Objective Evaluation - Data Visulisation**: After implementing the machine-learning process and saving the data to the Mongo database, the next step was to present the data in a user-friendly and easy-to-understand manner. Chart.js's line graphs were used to display the collected data in two different graphs: one for traffic congestion and another for the arrival times of buses on these roads. The goal was to provide a clear and concise representation of the data that could be easily understood by all users. Both objectives were successfully achieved, as shown in Figure 8. The line graphs were well-designed and provided a comprehensive view of the traffic congestion and bus arrival times. The data was presented in a visually appealing manner, with clear labels that made it easy to interpret. This ensured that users could understand the visualised information. However, due to limited resources and time constraints, it was not possible to retrieve the expected bus arrival times and compare them with the actual bus arrival times extracted from the uploaded videos. This would have provided even more valuable information to provide the user with, but nonetheless, valuable information was still gathered.
- **Objective Evaluation - Graph Usability**: In objectives 4 and 5, the application aims to provide valuable insights into traffic conditions and offer user insights into bus timing information. Objective 4 primarily focuses on providing valuable data to the user, which they can use to plan their travel more efficiently by understanding the traffic congestion and peak traffic hours on specific roads. On the other hand, objective 5 focuses on providing the user with information to help them understand the reality of bus arrival times.
  The generated graphs on the frontend of the application have made it easier for the users to view essential information such as peak hours and traffic congestion on specific roads by merely selecting a pin. The traffic information feature provides users with a graph for each day of the week at that location. Based on this, it can be confidently said that objective 4 has been met. Objective 5 has also been met so users can view times at which buses have appeared in certain areas which a user can take into account when planning for travelling on a bus.

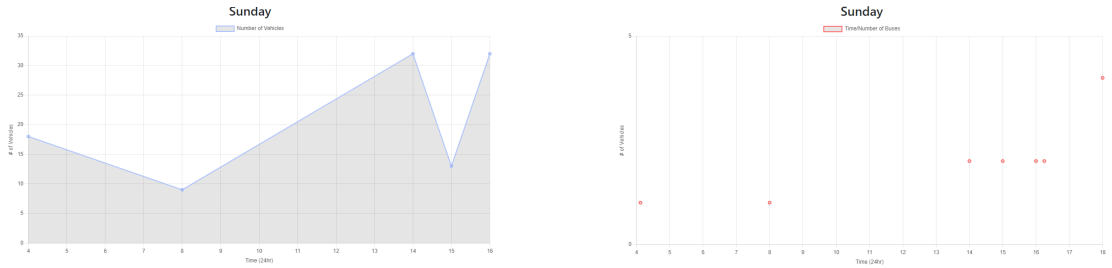


Figure 8: Graph of visualised data for traffic and buses

5.4 Limitations

5.4.1 Live Camera Feed

- During the development of the TrafficVision application, a limitation that was discussed was using a live camera feed to detect, track and extract data. This was because it would require setting up a capture device for an extended period of time at a suitable location, such as a bus stop, in order to capture the necessary data. However, due to the risks of setting up such a device in a public place like a bus stop, as well as cost constraints. It was also feared that a device like this would either be stolen or damaged, which would further drive up the costs as it would have to be replaced or repaired, and so this option was not considered viable.

5.4.2 Bus Provider Times

- As part of the efforts to improve the user experience of the TrafficVision application, a feature was initially planned to be implemented where users could compare the bus timings extracted from the uploaded videos with the expected arrival times provided by bus providers like TFI. This would have allowed users to get a better idea of when their bus is expected to arrive and consequently make better decisions regarding their travel plans. However, due to several factors, such as time constraints and difficulties in obtaining the necessary data through an API, as well as challenges in implementing the feature accurately, this feature had to be unfortunately limited. While it was understood that this feature would have been valuable to the users, other aspects of the application that were deemed more critical had to be prioritized.

5.4.3 Time Limitation

- During the development of the TrafficVision application, there were many ideas for new features that could have been valuable additions. However, due to time constraints, it was not feasible to add more features to the application. Time was a limiting factor that affected the overall development of the application. In order to ensure that the application was delivered within the allocated time, it was necessary to prioritize the features that were most important and relevant to the core functionality of the application. This required making some tough decisions and trade-offs. Some of the ideas that could have been valuable additions had to be put on hold for future iterations of the application. Despite the limitations, the development process focused on creating a functional and reliable application that met the objectives. While it may have been tempting to add more features, it was important to maintain a realistic approach that balanced the application's quality with the time constraints. The development process demonstrated the importance of balancing creativity with practicality and the need to work within constraints to achieve success.

5.5 Future Work

5.5.1 Live Camera Feed & TensorFlow Lite

- As mentioned previously, the traffic capture device was one of the main limitations of the application, as it would have been too expensive and risky to implement. However, an interesting option that can be explored is the use of live camera footage being processed with the help of Google's TensorFlow Lite. This will enable the application to process footage in real-time and provide more accurate data visualisation to users. Users could have the option to view multiple camera angles so that they can check the traffic status and see if their bus has arrived or is on its way. This feature would make the application more user-friendly and efficient, as users would be able to access real-time traffic data from multiple sources, leading to better-informed decisions and a more pleasant commuting experience.

5.5.2 Road Traffic News

- It has been suggested that the TrafficVision application could benefit from a new feature that would provide road and traffic news in the form of small blog posts or snippets of text. This would allow users to receive updates on traffic and collisions in real-time, similar to how news organizations deliver such information to their audience. However, this feature would require

manual recording to ensure accuracy or a person to validate the posts of other users. This was not a viable option during the development of the application. Alternatively, this feature could be used for commuters to share tips on commuting or provide advice for other users' vehicles. You can see a rough example of what this feature could look like on the TrafficVision application's Home Page.

5.5.3 Accident & Road Closure Reporting

- TrafficVision is an application that helps users to efficiently navigate through traffic. However, one of the limitations of the application is that it does not currently provide users with real-time updates on accidents or road closures. This limitation can sometimes cause users to unknowingly travel a route that may have delays, causing frustration and inconvenience. To address this issue, it would be extremely beneficial if TrafficVision could integrate a feature that allows users to report accidents and road closures in real-time. By doing so, the application would be able to provide users with up-to-date information on the status of the roads they plan on taking. This would enable users to make informed decisions on which route to take and avoid potential delays. Having this feature would also allow users to contribute to the app's overall functionality and reliability. By reporting accidents and road closures, users would be helping other users avoid any potential traffic issues and contribute to a more efficient traffic system overall.

[17]

6 Conclusion

6.1 Summary of Context

The TrafficVision application was developed after recognizing the need for a user-friendly platform that could provide traffic data to commuters. The application caters to users who want to plan their daily commutes or are simply interested in traffic patterns. In addition, it was designed to help urban and rural commuters by providing information on bus schedules, as anyone who regularly uses buses knows that they don't always stick to their advertised schedules, especially during peak hours. The goal of the TrafficVision application is to alleviate the frustration of waiting in traffic congestion or for a bus that's running late by providing users with reliable and informative data. It provides valuable insights into traffic patterns and helps users plan their routes accordingly. By harnessing state-of-the-art machine-learning technology like Ultralytics YoloV8, the app analyses traffic and generates easy-to-understand data that empowers users to make informed decisions. Overall, the TrafficVision application is a solution that addresses the needs of modern commuters. By providing traffic data, it helps users save time and avoid frustration, making their daily commute smoother and more efficient.

6.2 Summary of Objectives

At the beginning of the TrafficVision application, the objectives that were set were set to create a user-friendly application that was capable of processing a video of traffic, whether it be a busy urban street or a quiet rural road and providing the users with useful insights. The primary object was to develop an intuitive interface that enables users to upload videos, which would then be sent to the machine-learning process to be processed to extract the relevant data, such as vehicle type (e.g. Bus or Car), the time that the vehicle was detected and direction of the vehicle. Additionally, the application aimed to represent the data that was stored, which was the data that was extracted from the machine-learning process, as well as other user data that was inputted, such as time, date and location, to generate visually informative graphs that enabled the applications users to utilise these graphs to understand traffic patterns as well as making travelling decisions. While the application's initial objectives focused on data extraction and data visualisation, the findings in the system evaluation revealed the reliability of the application in detecting and tracking. Overall, the TrafficVision application has proven to be a useful application for traffic analysis and monitoring. Its intuitive interface, reliable detection and tracking capabilities, and visually informative graphs have made it a valuable application for users.

6.3 System Evaluation Findings

The TrafficVision application underwent a thorough system evaluation process, which revealed valuable insights into its functionality, reliability, and limitations. The findings that emerged through the testing and performance metrics from Chapter 5 will be discussed in detail below. Firstly, the application's reliability and robustness were tested by manually testing the Python machine-learning process and the React frontend. This testing ensured that the application's functionality worked as intended under various conditions, whether it be extracting data from the uploaded videos or providing the users with a user-friendly user interface. The manual testing process also helped to identify and address any issues that may have occurred during testing. In addition to manual testing, Jest testing was utilised to automate some of the react frontend tests to further ensure the application's reliability. The application's performance metrics were also analyzed to reveal its processing speeds and potential bottlenecks that may negatively affect it. The application's frame processing rate and algorithm execution time were measured, highlighting the algorithm's dependency on its CPU resource usage and the need for possible future optimisation. However, despite the possible optimisations that could be made, the TrafficVision application showcased its reliability in extracting the necessary data even when the videos uploaded had various different conditions. This shows that the algorithm is effective in different scenarios. Despite the small shortfall in displaying the expected times for bus services stated by services such as TFI, the TrafficVision application has been an overall success in terms of the initial objectives. Its user-friendly interface, valuable insights, and efficient data extraction have made it useful for commuters to stay informed about traffic patterns and make informed decisions. With continued updates and improvements, the TrafficVision application has the potential to become an even more powerful tool for commuters. The findings from the system evaluation process will help guide the development to further improve the application's reliability, processing speed, and overall performance. TrafficVision's future is bright, and we can expect even more innovations in the years to come.

6.4 Future Work Opportunities

The system evaluation has identified areas for future development and investigation to enhance the capabilities of the TrafficVision application and further improve its usefulness and effectiveness. Although the current version of the application has met its primary goals and provides value to its users, there are numerous opportunities for future improvements.

Optimizing processing speed, integrating real-time traffic data with live camera feeds, and adding bus service timing information are among the innovations

that can enhance the efficiency and usefulness of the application. Additionally, implementing new features such as user event reporting for collisions and road closures and adding alerts for specific traffic conditions can further enhance the application's utility and appeal. You can find more detailed information about the future work here: 5.5.

By continuously developing new versions of the application, TrafficVision strives to improve its reliability and efficiency, ensuring that users have access to an ever-improving solution. These updates could include new functionality and design improvements that make the app more intuitive and user-friendly. By staying up-to-date with the latest developments in the field, TrafficVision would be able to remain a viable solution for those looking for reliable traffic navigation assistance.

6.5 Conclusion

The TrafficVision application is an innovative approach towards traffic analysis, providing users with a user-friendly application for navigating urban and rural traffic and roads. The potential for further enhancements to the application is vast, ensuring that it will continue to play a vital role in helping commuters navigate the complexities of modern transportation networks and services. In conclusion, TrafficVision is a valuable asset for all commuters looking to make their daily travel more efficient and convenient.

References

[1] https://docs.python.org/3/.

[2] https://docs.opencv.org/4.x/index.html.

[3] https://github.com/ultralytics/ultralytics/pull/575.

[4] https://docs.ultralytics.com/reference/engine/results/ultralytics.engine.results.Boxes.

[5] https://www.augmentedstartups.com/blog/10-things-you-need-to-know-about-ultralytics-yolov8.

[6] https://flask.palletsprojects.com/en/3.0.x/.

[7] https://www.linkedin.com/pulse/expressjs-good-bad-ugly-aziz-taha/.

[8] https://expressjs.com/.

[9] https://www.mongodb.com/docs/.

[10] https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/.

[11] https://legacy.reactjs.org/docs/getting-started.html.

[12] https://www.linkedin.com/pulse/react-jsnode-jspros-cons-shagufta-naz/.

[13] https://angular.io/docs.

[14] https://www.chartjs.org/docs/latest/.

[15] https://docs.railway.app/.

[16] https://docs.github.com/en/pages/.

[17] https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/.