# Road Surface Status Using Machine Learning

**Final Engineering Report**

**Preet Patel 100708239**

**Tiwaloluwa Ojo 100700622**

**Faraaz Mohsin 100659110**

**Janajan Jeyabalan 100698148**

**Waleed El Alawi 100764573**

Final engineering report submitted in partial fulfillment for the final year Capstone Project in the Faculty of Engineering and Applied Science.

Advisor: Dr. Mohamed El-Darieby, P.Eng.

Coordinator: Dr. V.K. Sood, P.Eng.

Submitted to Ontario Tech University

April 2023

## Abstract

Road surface condition and deterioration are crucial factors to monitor for the maintenance of roads, which entails the safety and comfort of the drivers and pedestrians who use those roads. Road surface inspection is currently conducted using manual human effort and expensive machinery. Many municipalities lack the resources to conduct inspections thoroughly and frequently, which increases the risks posed by road surface conditions and deterioration. Collection and maintenance of updated data of road surface damage and conditions is crucial for the municipalities to make decisions regarding the maintenance of the roads. Due to the recent advancements in object detection and computer vision, a system using machine learning can be created to detect road surface damage and its various conditions. The goal of this project is to leverage object detection in machine learning to create an application that can detect different road conditions and damages to collect data which municipalities can utilize to allocate their resources for road maintenance appropriately. The following literature discusses project objectives, the background and prior work conducted on the topic. It explores the various research conducted on image processing technology to detect different road conditions; how they were utilized to seek a cheaper and more reliable alternative to road surface inspection; and other methods used to capture road deterioration. The report includes the various use cases for this project, and the collected functional and non-functional requirements. Furthermore, the literature also discusses the detailed design of the software system developed. It provides a detailed explanation of the design decisions and method of implementation for each component of the system. It also covers the integration and unit tests designed to test each component of the system and provides the results of these tests. The literature also provides the results for the acceptance tests and validates if the product developed for this project meets the defined business requirements. Lastly, the report covers the ethical and safety considerations made by the team for this project.

## Dedication

To all our professors for teaching and aspiring us over the course of the software engineering program at Ontario Tech University.

## Acknowledgements

We would like to express our gratitude to Dr. Mohamed El-Darieby from Ontario Tech University for his guidance and encouragement throughout this project. We would also like to thank the Capstone Coordinator Dr. Vijay Sood from Ontario Tech University for providing us with the opportunity to work on this project as part of our capstone project. We would like to express our genuine gratitude for the valuable learning experience and abundant opportunities that have been extended to us throughout the past year.

# Table of Contents

## List of Figures 23

# List of Tables 13

## List of Acronyms

# 1 Introduction

The Canadian landscape is vast and requires kilometers of roadway to connect cities, communities, and businesses together. The maintenance of these roads is a tough task that requires vigilance as it can impact both the health and performance of the vehicle and the safety of the drivers that travel on the road every day. Due to the cost and time-consuming nature of road inspections, it is difficult for the government to maintain a database containing information on the deterioration of the roads and highways in the country. During the inspection, besides identification of the presence or absence of road damage, the inspection procedure must provide actionable insights such as the characteristics, type, and extent of the road damage for the information to be useful. With the advancement of image-processing in recent years, image recognition through machine learning can be utilized to visually inspect a scene for deterioration and decay. This method can pose a cost-effective alternative to the current methods of deploying a road maintenance unit. The goal of the project is to create a system that relies on the personnel that utilize the roads daily and the instruments they are already familiar with.

## 1.1 Problem Statement

The Canadian weather is harsh, and its geography is wide. Due to this complexity, the Canadian government are tasked with maintaining its kilometers of road and pavement. This often indicates that some fixes are often not applied on time, which places its drivers and pedestrians at risk. This risk also feedbacks onto the governments as cost of fixing increases for as long as the deterioration is not attended to. An easier and cost-effective method using machine learning models is required to allow for faster reactivity from the government and participation from its citizens. This method will also provide an automated means of alerting the road maintenance units as quickly as the conditions are discovered.

## 1.2 Objective

The goal of our project is to design a system that can detect and classify road surface damage and different road conditions using image-processing technology. The system should display the collected data on a dashboard in different forms, such as a live map leveraging the geolocation of the data or tables that can be filtered so the data can be utilized to help with decisions regarding resource allocation to fix road damages. The system should leverage the advancement of smartphone devices, which are now equipped with high resolution cameras, various sensors, and powerful processors to create a cost-effective solution in comparison to the current approaches of road inspection.

# 2 Background and Research Review

## 2.1 An Asphalt Damage Dataset and Detection System Based on RetinaNet for Road Conditions Assessment

The analysis of road infrastructure using image processing technology and artificial intelligence techniques has recently been receiving increased attention in research. The advancement of such technology is important for road managers and government agencies as roads are one of the most important civil infrastructures in a country and contribute directly and indirectly to the country's economy and to the safety of its citizens [1]. Traditional methods for road inspections involve manual inspections by experts used in combination with sophisticated technologies containing various sensors and traditional imaging techniques, which are time-consuming, cost-intensive, inconsistent, and prone to errors. Initial approaches to aid government agencies in automating road inspection and data collection processes include the use of the mobile measurement system, which uses various sensors such as inertial profilers and scanners, in tandem with sophisticated imaging techniques [1]. Although this approach is highly accurate, it is also considerably more expensive and complex to operate, resulting in many governments not being able to afford such an approach. Recently, computer vision acquisition systems have emerged as a promising solution to these problems when integrated with georeferenced devices such as smartphones [1]. These computer vision-based systems are a lot cheaper and can be used to make solutions that are more cost-effective.

Current vision-based technologies only focus on the presence or absence of road damage and ignore the more useful information such as the characteristics, type, and extent of the road damage [1]. Such information is required by road managers to clearly understand the type of damage and the extent thereof, to identify the effective action required to solve the problem and to allocate the necessary required for a solution [1]. Although vision-based approaches are promising, many of them are targeted to specific damages and use a small dataset, which means these solutions do not function well in complex scenarios and are also costly for governments. The biggest hurdle for automating road damage detection and classification is to consistently achieve high accuracy in challenging environmental conditions. In recent years, comprehensive datasets have been created and, with deep learning-based methods, outstanding results have been obtained in image analysis and classification problems [1]. The biggest concern with deep learning is the problem of overfitting when working with a small dataset [1]. Data augmentation can be used to generate training data from examples in a dataset to address the problem of overfitting [1]. Another frequent problem with datasets can be the poor quality of images, which can contain blurriness, low contrast, and other characteristics that can lower the accuracy of the model. To solve this issue, the images need to be filtered based on quality scores [1]. In conclusion, a detection system created using a generic object detector trained on a quality-aware filtered dataset can detect damage with high accuracy and low inference times.

## 2.2 A New Road Damage Detection Baseline with Attention Learning

Automated detection of road damage and road conditions is a challenging problem in road maintenance. Automated detection involves automatically detecting road damage and assessing its severity by using deep learning. Due to the sparse distribution of characteristic pixels, this problem is far more challenging than object detection [2]. Deep learning-based systems, which target the problem of automated road damage detection, have begun to gradually appear and have become an emerging task in computer vision [2]. These systems were created using public datasets from specific countries. However, these methods were unsuccessful when used directly in other countries due to the differences between the training data and the actual testing data [2]. Different climates and road construction standards lead to differences in the damage characteristics. In machine learning, transfer learning or semi-supervised training can address the problem of the differences between the training and testing data, but it requires rich datasets for training, which are private and unpublished [2]. Based on existing detection algorithms, this paper proposed road damage detection using edge detection and attention learning, which can effectively alleviate the sparse distribution and unclear boundaries of road damage [2].

## 2.3 Transfer Learning-based Road Damage Detection for Multiple Countries

Governments and road authorities continuously seek solutions for automated evaluation of road damage and conditions. Countries like Japan have been successful in developing smartphone-based methods for automatic road condition monitoring, but it is only limited to Japan as the accuracy of the system is compromised in different countries [3]. Roads experience wear and deterioration from factors such as location, age, traffic volumes, harsh weather conditions, engineering faults, and materials used to construct the roads [3]. It is critical to maintain a database containing knowledge of the deterioration of roads to preserve these roads in good and safe conditions. The traditional methods for inspection include manual and semi-automated surveys, which suffer from the subjective judgement of the inspectors and are also time-consuming [3]. Recently developed fully automated solutions involve the employment of vehicles equipped with expensive sensors, image processing, and pattern recognition software [3]. However, these vehicles are expensive, and the costs can reach half a million dollars [3]. With the advancement of smartphones containing high-resolution cameras, sensors, and powerful processors, these devices can be used to create efficient and cost-effective road surface inspection tools [3]. Convolutional Neural Networks (CNN) are one of the main categories of deep neural networks that are used for image recognition and classification [3]. In Convolutional Neural Networks, input images are processed using convolutional operations such as gradients, edge detection filters, blobs, with learnable weights and biases, which allows them to automatically detect critical features without human supervision after the training stage [3]. In conclusion, the study explains that using convolutional neural networks with a dataset that has images from different countries such as Japan, India, and the Czech Republic, we can efficiently detect and classify road damage globally.

## 2.4 A new model for road-condition forecasting in Canada

This article focuses on the new model that was established for road-condition forecasting in Canada. Road weather forecasting helps reduce the number of accidents that may occur, and with technology it will further reduce the risks and possible negative impacts on the economy [4]. Road Weather Information System (RWIS) consists of roadside observation stations that continuously collect weather and road-condition information; a system to forecast these conditions in the future; and a mechanism to communicate the pertinent information to the road maintenance staff to aid in the planning of deicing and snow removal. The RWIS infrastructure helps from two points of view; one from the point of view of safety and the other for economic reasons [4]. The economic benefits that come from this model are the fewer accidents and the more readily available roads because of more prompt maintenance. Metro has a complete road-conditioning system that can forecast both the temperature of the road's surface and the state of the pavement. The surface energy balance model helps in evaluating the energy fluxes at the road surface. The road heat-conduction model helps to calculate the one-dimensional heat diffusion by establishing the temperature profile. The forecast process order is atmospheric forcing, then the initialization phase, then the coupling phase, and then essentially the forecast phase. The results are then modelled by error feedback and coupling, and the system progresses by doing the manual versus automatic modes. It essentially concludes the results by showing the time of freezing or thawing, then focuses on the model accuracy, and finishes by viewing the road-condition forecast [4].

## 2.5 Accident risk of road and weather conditions on different road types

This study was designed to investigate the relative accident risk of different road weather conditions and to examine combinations of conditions. Palm probability was a notion that helped the study to develop a method that originated from the theory of random point processes, which essentially chooses a random automobile from the traffic [5]. The method consists of calculating the Palm distribution of different conditions and comparing it with the distribution of the same conditions as seen by accidents. This study's goal was to investigate the frequency and relative accident risk of various weather and road conditions from the perspective of drivers, both generally and when broken down by accident type and road type. The road surface temperature, wind speed, wind direction, road weather code, and road weather category were all included in the road weather data. This study's investigation concentrated on the road weather category and code [5]. The four-character road weather code, which consists of one letter and three numbers, expresses cloudiness, precipitation type, and intensity. The overall relative accident risks for the road types (two-lane roads, multiple-lane roads, and motorways) were compared for various locations to provide an overview of the general relative accident risk for the various road types and climatic areas. Then, based on the kind of road and accident type, the relative accident risks for the categories of precipitation type, precipitation intensity, and road weather were compared. According to a comparison of the relative accident risks associated with different types of precipitation [5], Snowfall had the highest relative accident risks of all road kinds and accident types.

## 2.6 Models for predicting pavement deterioration

This article focuses on models that can be used to predict pavement deterioration. Following an examination of the numerous prediction model types, the authors concluded that an empirical-mechanistic model, with a systematic database containing the structural details, traffic volume, and condition data for each "homogeneous" segment of road, is the most appropriate [6]. Pavements are intricate physical constructions that react intricately to a variety of environmental and load-related factors and their interconnections. Therefore, a prediction model should consider the development of different distresses and how they might be impacted by both normal and planned maintenance. The authors created performance equations for three types of pavements using the road system's condition data: flexible pavements without an overlay; flexible pavements with an overlay; and composite pavements [6]. In essence, this article showed different ways in which individuals can predict pavement deterioration.

## 2.7 Pavement Deterioration and its causes

This article focuses on understanding the meaning behind pavements and the different types of pavements. However, it also focuses on pavement deterioration and its causes. The factors that influence the performance of a pavement are traffic, moisture, subgrade, construction quality, and maintenance [7]. The primary functions of a pavement are to provide a reasonably smooth riding surface, provide adequate surface friction, protect the subgrade, and provide waterproofing. The types of pavement deterioration are cracking, surface deformation, disintegration, and surface defects [7]. Reinforcing steel is an option for these pavements and is typically utilized to minimize or completely remove joints. Through a process known as beam action, the enhanced stiffness of concrete enables the concrete surface layer to bridge minor weak spots in the supporting layer. The most common types of cracking are fatigue cracking, longitudinal cracking, traverse cracking, block cracking, slippage cracking, reflective cracking, and edge cracking [7]. Some of the causes of pavement deterioration are temperature, sudden increases in traffic loading and temperature variations.

## 2.8 Automatic Pavement Damage Predictions Using Various Machine Learning Algorithms: Evaluation and Comparison

An effective method for addressing concerns about road deterioration has been machine learning algorithms. A few studies have made use of tensor flow, image processing using ML, 3D image-based screening techniques, and trajectory clustering algorithms [10]. Tensor flow technology has produced better predictions of surface deterioration. Numerous initiatives have used machine learning techniques to find deterioration on the road surface. However, most of these studies concentrated on crack isolation. The term "deterioration" refers to a wide variety of road damage, including numerous sorts of cracks [10]. Effective algorithms that recognize the kinds of damage are crucial. The techniques and analysis employed in this article reveal that recent research has improved performance and efficiency that accurately detects the state of deterioration in real-time [10]. Additionally, when developing such projects, we must consider their financial and environmental implications. Both economical and environmentally beneficial methods should be used. Regression-based techniques, such as those based on trees, are used to find errors [10]. These algorithms and other strategies have demonstrated increased performance and outcomes. Strong project and analysis foundations are required for an effective solution to this problem. For example, detailed photographs of the road surface that would work well in this method are needed in the collected datasets. A strategy to determine the physical effects of road surfaces was announced by the National Cooperative Highway Research Program in America [10]. "Pavement-ME" contains two parts: one that focuses on the physical harm brought on by weather conditions, and the other is the numerical part that determines the model's correctness [10].

## 2.9 Developing a near real-time road surface anomaly detection approach for road surface monitoring

Road surface deterioration has been detected and measured using a variety of approaches during the past few years. To improve road surface monitoring, this paper focuses on real-time anomaly detection on the road surface [11]. When employing such methods, it is important to consider several issues, including cost and other elements that could slow the procedure down. This article, for instance, claims that maintaining low-traffic highways has proven expensive [11]. The storage-collected videos go over the 1 GB/km restriction [11]. During data processing, the constraints have detrimental effects. Many government departments and municipalities are implementing precise and cost-effective technologies because of reasons like these. Vibration-based techniques have been discovered to be inexpensive by researchers, engineers, and various investigations. Due to sensors that aid in detection, the introduction of smartphone-based screening has expanded the idea of the method of detecting impacts on the road surface. Cellphones can detect damage, but a study has revealed that signals from embedded sensors like smartphones make it difficult to identify surface conditions due to differing sensor qualities [11]. The two aspects of detecting road surfaces are the main topics of this essay. One of them is that real-time methods to find abnormalities like cracks, bumps, and potholes have been studied. The other factor is that it is hard to tell the difference between different impact categories because of the way a vehicle and road contact affect impact categories like size and depth in different ways [11].

## 2.10 Road Surface Damage Detection Using Fully Convolutional Neural Networks and Semi-Supervised Learning

Road surface damage can be very dangerous for the drivers on the road. As it is extremely hard for any driver to recognize and avoid potholes on a rainy day or snowy day, which can cause damage to cars and other properties. The following section will be about the approaches that have been implemented to detect road surface damage. There are three major approaches: A Vibration sensor-based method that uses sensors to detect vibration amplitude [13]. The performance of this technique can vary for many reasons, as it depends on the vibration from vehicle to vehicle and the road surface itself. The second method, the Laser scanning- based method, can detect road damage with higher accuracy than other methods, but the main issue with this method is that you need to drive at a slow speed, which can cause traffic congestion. The third method is the computer vision-based method, which uses the optical flow of road surfaces to detect damage to the road surface. Its performance may vary depending on the weather conditions [13].

Image processing technologies based on convolution neural networks (CNN) are widely used among various neural network structures. The algorithm used with this technology shows better performance than conventional image processing in terms of regression problems, object detection, and semantic segmentation. CNN technologies detect road surface damage using object detection [13]. The object detection finds the object and puts a box around to detect and classify what the object is. The CNN layer shown below is a fully connected neural network, b and c illustrate 1D and 2D convolution neural networks. All the neural layers on the previous layer and the next are connected to each other [13]. Full neural networks must be turned into one dimension, which gives us information about space.

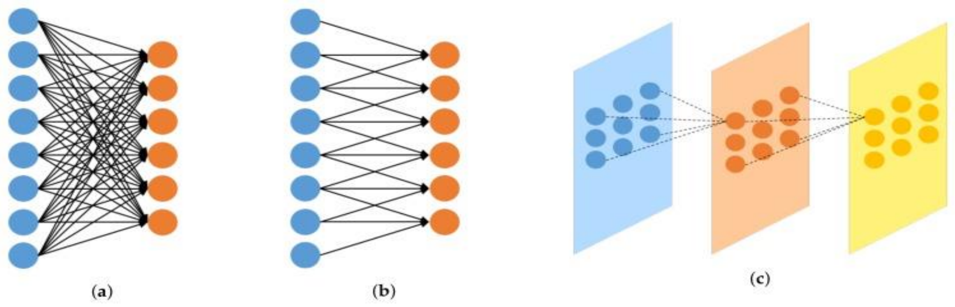


*Figure 1: (a) Fully connected neural network and (b) 1D and (c) 2D convolutional neural networks*

Semi-Supervised learning using Pseudo Labels refers to the use of many unlabeled datasets. To use this technique, we need to train the system using labeled data. Then generated 5 different models to improve the performance [13]. Using these models will obtain predicted output datasets. This process will train the system to be trained each time we use the system.

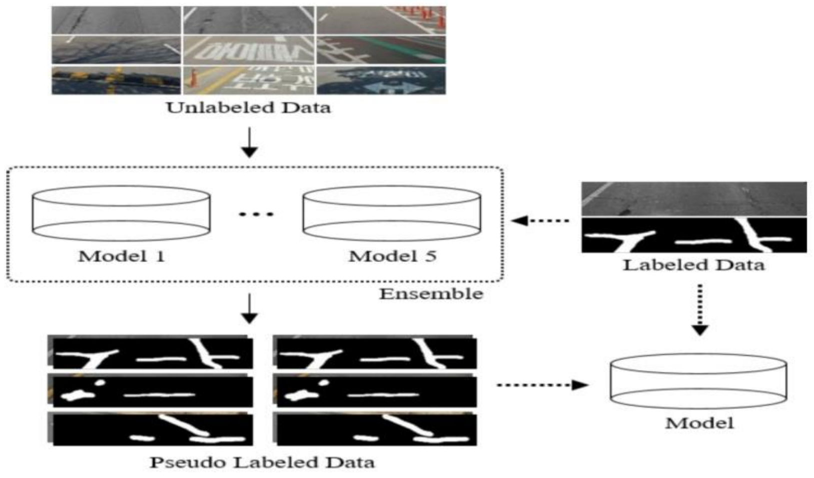


*Figure 2 Structure of road surface damage detection using pseudo labels.*

# 3 Scenario and Use Cases

## 3.1 Detecting road surface damages

**Scenario:** Detecting different types of road damages from images captured by a smartphone.

**Purpose:** To detect and classify different types of road damage.

**Individuals:** Public transit, Car owners, Pedestrians

**Equipment/Software:** Smartphone

*Table 1 Use Case 3.1 – Detecting Road surface damages*

| **Use Case ID 01** | |
|---|---|
| Use case name | Detecting different types of road damages from images captured by smartphone |
| Primary Actor | Regional Works Department, Ministry of Transportation |
| Summary Description | The system should be able to identify and classify the road damage from images captured by the actors. The data collected should be streamed to the cloud for road damage detection. |
| Priority | High |
| Pre-Condition | The user must install the system on a smartphone device with a camera beforehand. |
| Post-Condition | System should allow users to capture images and stream the data to the cloud where object detection machine learning models can be leveraged to detect road damages from the captured image. |
| Basic Path | 1. Install the system on the smartphone.<br>2. The user uses the app to start the camera to capture an image of possible road damage.<br>3. The system should stream the collected data to the cloud.<br>4. The system should use an object detection machine learning model to detect any road damage from the image. |
| Alternate Path | N/A |
| Exception | The use case is dependent on the system being able to capture images and detecting road damages from the captured images. If the road damage cannot be detected by the model, the system will not be able to fulfil the primary functionality. |

## 3.2 Display detected road damage and condition locations on a live map dashboard for users to track status of different roads

**Scenario**: Display detected road damages and conditions using geolocation of the data on a live map on the system dashboard.

**Purpose:** Users should be able to view and track the detected road damage and conditions on a map view.

**Individuals**: Ministry of Transportation, Regional Works Department, Road Contractors

**Equipment/Software**: Cloud computing

*Table 2 Use Case 3.2 – Display detected road damage and condition locations on a live map dashboard for users to track status of different roads*

| **Use Case ID 02** | |
|---|---|
| Use case name | Display detected road damages locations on a live map dashboard for users to track status of different roads |
| Primary Actor | Ministry of Transportation, Regional Works Department, Road Contractors |
| Summary Description | The user will be able to locate the detected road and conditions on a live map to enable users to track the status of different roads. |
| Priority | Medium |
| Pre-Condition | The user captured images or video of poor road condition or damage on a smartphone device using the app and streamed the data to the cloud for model inference to detect road damages from the image. |
| Post-Condition | The user will be able to view the detected road damage on a live map dashboard. |
| Basic Path | 1. Captured image containing possible road damage is streamed to the cloud with the geolocation of the image from the smartphone device.<br>2. Machine learning model is used to collect data related to road damages from the image.<br>3. System uses the geolocation associated with the data to update a map, to display the detected damage on a map view. |
| Alternate Path | N/A |
| Exception | Data captured containing possible road damage should be streamed to the cloud with the associated geolocation of the smartphone device. Users must be able to locate and review where the damage is on a map and track the status of different roads. |

# 4 Revised Requirements and Constraints

## 4.1 Requirements

*Table 3 Revised Requirements with Additions*

| Requirement ID | Requirement Type | Requirement Description | Priority | Risk |
|---|---|---|---|---|
| Req 1 | Functional | System should be able to capture images from phone gallery or camera | H | L |
| Req 2 | Functional | System should be able to georeferenced collected data | H | L |
| Req 3 | Functional | System should be able to utilize the accelerometer, gyroscope, and other sensors on smartphone devices to support the model's accuracy | L | H |
| Req 4 | Functional | System should allow user devices to upload data to the cloud | H | L |
| Req 5 | Functional | System should be able to perform image classification and object detection on a cloud hosted service | H | L |
| Req 6 | Functional | System should be able to detect and classify distinct types of road damages and conditions | H | M |
| Req 7 | Functional | System should be able to identify the road damages and conditions in different weather conditions | M | H |
| Req 8 | Functional | System should be able to store image footage as it might be paired with the detection report | M | L |
| Req 9 | Functional | System should store all data collected from the cloud hosted AI image processing service | H | L |
| Req 10 | Functional | System Dashboard should be able to read events from the event streaming platform | H | L |
| Req 11 | Functional | System should have a dashboard to view all the captured data on a map view or in tabular form | H | L |
| Req 12 | Functional | System should generate polls for users for road damage analysis in geolocation areas with most incoming traffic | L | M |
| Req 13 | Non-Functional | System should have user-friendly features and usability tutorials | H | L |
| Req 14 | Non-Functional | System should encrypt and securely store end-user information and other data | M | M |
| Req 15 | Non-Functional | System should encrypt data for security during data transmission | H | M |
| Req 16 | Non-Functional | System should perform object detection efficiently and quickly | H | M |
| Req 17 | Non-Functional | System should only store relevant data | H | L |
| Req 18 | Functional | Multiple consumers should be able to subscribe from a single cluster | H | L |
| Req 19 | Non-Functional | Events should retain their data for at least 7 days | H | L |

## 4.2 Concerns and Constraints

*Table 4 Revised Concerns and Constraints with additions*

| ID | Concerns | Constraints |
|---|---|---|
| CON 01 | Read and Write speed to cluster partitions | Mobile device storage might limit how much data can be stored on the device |
| CON 02 | Poor resolution images and videos might need to be discarded if they pose a threat to the accuracy of the A.I model | The user will need to accrue network costs |
| CON 03 | Data consistency among the systems services | User device types will impact the deployment of the web app. E.g., Apple and their iOS devices are bureaucratic with their deployment practices |

# 5 Detailed Design

The following project utilizes computer vision to replace expensive equipment involved in road surface damage detection. Expensive equipment and methods limit the abilities of governments around the globe to keep an updated set of data on road surface damage around their cities. This often results in economic and physical repercussions for the drivers and their passengers. For this project, different components such as the mobile application, Firebase cloud storage, Kafka cluster, You Only Look Once (YOLOv5) object-detection model, Neo4j graph database, and a web dashboard will be used to implement the design outlined in the figure below.

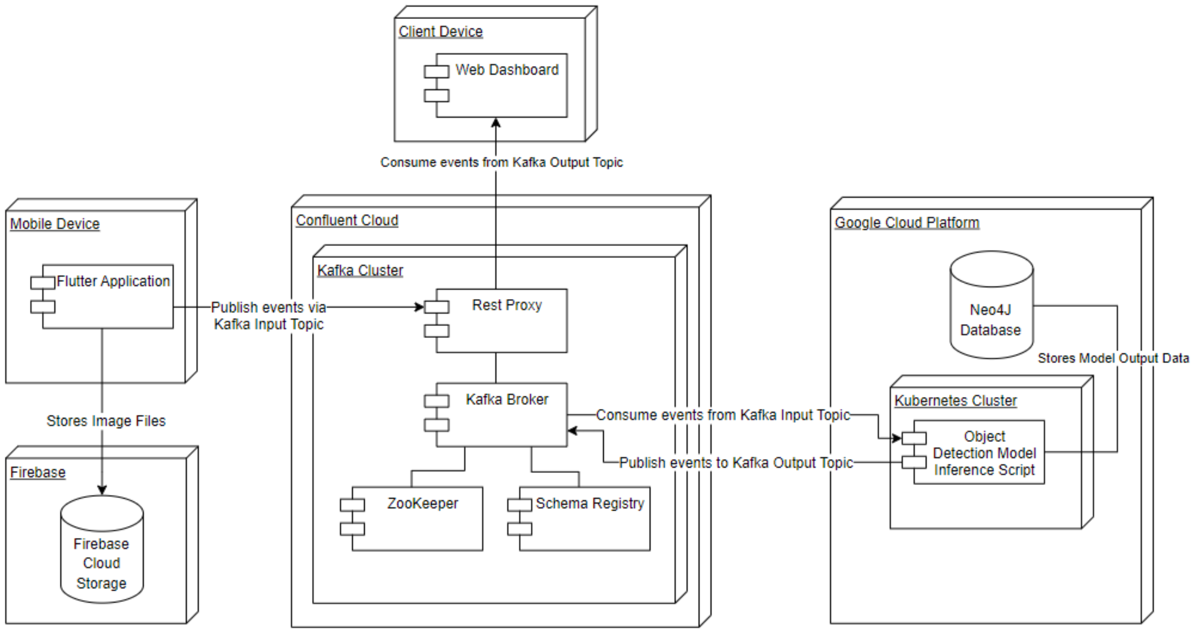


*Figure 3 Detailed Deployment Diagram*

In this section of the report, the detailed design and implementation for each of the components shown in the design above will be described in further detail. The various design decisions taken to create a product from the project conceptual design will be discussed for each component.

## 5.1 Mobile Flutter Application

According to our deployment design as referenced in our previous reports, the mobile application publishes data from the Kafka Producer to the Kafka Cluster hosted on the cloud. The mobile app will be developed for taking photographs and videos, cropping the important aspects of the scene before streaming to the Kafka Cluster. Our mobile application's user interface design is essential for the rest of the system as it not only enhances the user's visual experience, but also represents a vital component as an edge device in our system. Form-based, menu-driven, and touch user interfaces are some of the several types of user interfaces that will be used. Data entry into the program and a constrained menu of options will be used to construct the form-based user interface. Moreover, we will use the menu-driven user interface to take advantage of the computer power and rely on the camera recording capabilities of the users' mobile devices by using them. As we no longer need to distribute recording devices to every potential user of our system, this will save our stakeholders money.

This flutter mobile application design would further evolve to be an effective tool for identifying road surface deterioration and notifying the relevant authorities. It makes use of the latest computer vision techniques to ensure accurate identification of road surface damage. The location data would allow for targeted maintenance of the road network, improving road safety and reducing the risk of accidents. The following features will be implemented in the mobile application, which includes camera access, location access, firebase storage upload, Kafka Producer, and user interface. In terms of camera access, the application would require access to the user's camera to capture images of the road surface. The camera would be activated by a button in the application's user interface. With regards to the location access, the application would use the user's location data to identify the location of the road surface images. The location would be automatically detected and stored in the application's database along with the images. The firebase storage bucket will allow our team to upload pictures of the surfaces that are being inspected to inspect for deterioration. The Kafka producer will be responsible for sending data records, or messages, to Kafka topics. In terms of the user interface, the application would be designed to be easy to use and appealing. It would feature a button to activate the camera and a map showing the location of the captured images.

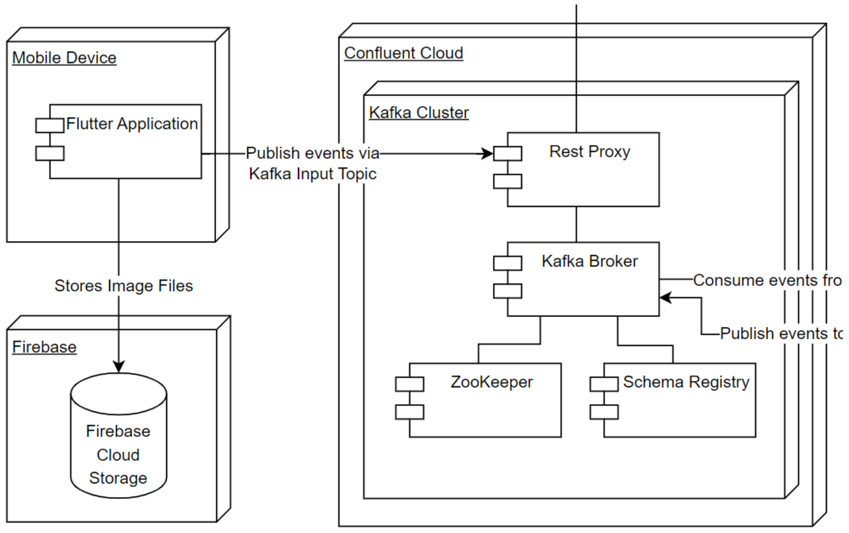


*Figure 4 Shared relationship between Kafka, Firebase, and the Mobile App*

## 5.2 Kafka Cluster

Event streaming is a tool used by many fortune 500 companies around the world that assesses vast volumes of data that need to flow between their services and applications. Event streaming presents a means of decoupling and delivering data in real-time. Its data can be abstracted so that the type of data received will cause a different behavior within the system. Event streaming platforms are often used for publish-subscribe (pub/sub) systems. Its use cases span industries such as automotive, banking, and logistics [1]. Event streaming platforms are driven by an event-driven architecture, which can handle a vast throughput of data and can scale to the demand of its consumers. They often utilize a broker to maintain the system and its components. For instance, a broker will distribute the events it receives across its cluster as well as maintain information regarding its publishers and subscribers.

The core of our system will be utilizing Apache Kafka, as explained in section 1.2. We will be able to develop agnostically and de-couple our services as a result. Our clients' mobile devices will represent the producer in the system, while our A.I. model service will act as the topic's consumers. The producer will be responsible for streaming its events to the cloud-based Kafka Cluster, and the A.I. service will handle its consumption for further processing via a preset topic on the broker. Kafka was selected due to its capabilities of performing both stream and batch processing in real-time. We want to make use of Kafka's low latency of 10ms, scalability using partitions, and storage capacity of 5 TB [2] [3]. Although this method might appear excessive, it was economical and suitable for a system that needed to assimilate a lot of data quickly. Moreover, the team had prior experience using the platform. Although there are several different types of event streaming platforms in the industry today, such as, Amazon Web Services (AWS) Kinesis, Solace, and Google Cloud Platform (GCP) Cloud Pub/Sub.

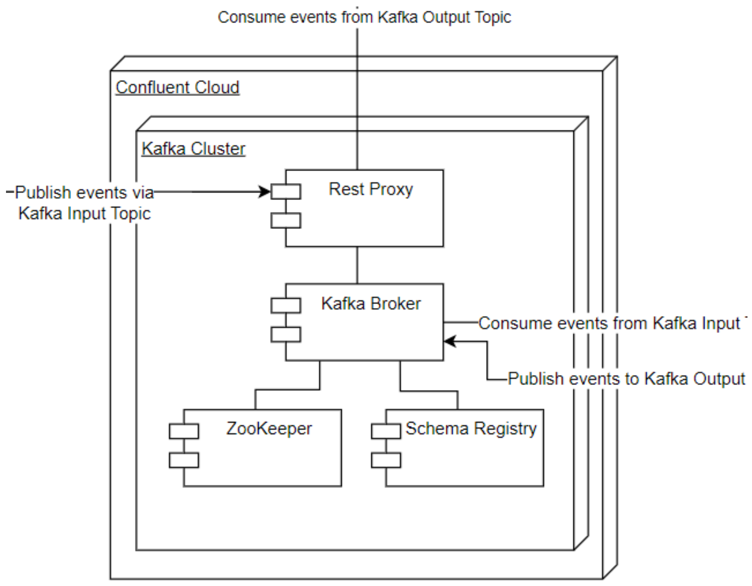

Consume events from Kafka Output Topic
Confluent Cloud
Kafka Cluster
Publish events via Kafka Input Topic
Rest Proxy
Kafka Broker
Consume events from Kafka Input
Publish events to Kafka Output
ZooKeeper
Schema Registry

*Figure 5 An illustration detailing Kafka on Confluent Cloud*

## 5.3 Firebase Cloud Storage

The system utilizes Firebase Storage to store the photos taken by the mobile application user. After an image is captured, the mobile application uploads the image to a cloud storage called Firebase Cloud Storage. Firebase Cloud Storage is a cloud object storage service that stores uploaded images, videos, or files to a Google Cloud Storage bucket. Firebase Storage provides various benefits to the project such as security to the stored files, encryption during uploading and downloading files, high scalability, and robust upload and download operations, all at a fair price. With robust upload operations, upon the mobile device losing connection, the application can continue uploading from the point where the operation was interrupted. The purpose of the Firebase Cloud Storage component is to indefinitely store all the images that the object detection model will process. The YOLOv5 model that is used in the project is trained using datasets containing images from various countries. As the images are from various global locations, the road conditions in each of the images can affect the accuracy of the model. The model accuracy can be improved if it is retrained or further trained using data that is local to the area of operation. As the goal of the capstone project is to create a road damage detection system that can detect road damage in Canada, the model will benefit from being trained using images that originate in Canada. The Firebase Cloud Storage will hold valuable data that is captured by the users which can be used for continuous learning for the model used in the project or retraining a new model that can perform better by being trained using data that originates locally.

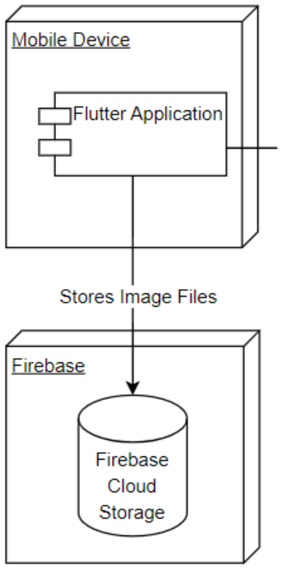


*Figure 6 A shared relationship between the Flutter Mobile app and the Firebase Cloud Storage*

### 5.4 YOLOv5 Object Detection Model

For the implementation of the object detection model, several different models were considered and statistically compared. The models that were considered for implementation of the object detection model were the ResNet50, MobileNet V2, and the YOLOv5. After comparing the statistical results of the models from other projects that attempted to create object detection models for the purpose of road damage detection, YOLOv5 was considered as the clear favorite by the team due to its competitive statistical performance compared to the other models and the unique nature of performing better than the other models in real-time object detection. The YOLOv5 model is trained using supervised learning on a labelled dataset that will be obtained from online resources. Once the model is trained, it is integrated into the project for inference. The inference script and the trained model are containerized and deployed on a cloud platform such as the Google Cloud Platform (GCP). Containerization of the inference script and model provides benefits such as portability, efficiency, and most importantly scalability. Portability allows the deployment of the model on different platforms such as the cloud. Containerization provides efficiency in deployment as only the necessary resources are included in the container to allow for rapid deployment. Lastly, it also provides improved scalability as multiple instances of the containers can be deployed independently to work together to meet the demands of any increase in the traffic of images that need to be processed by the model.

The model and the script are containerized together so that the inference script can utilize the model to process images taken by the mobile application and received from the Kafka topic. Firstly, the script imports and initializes the trained YOLOv5 model with the best weights obtained through training. Next, the script consumes the Kafka topic that the mobile application will produce messages containing the URL of the image that is stored on the firebase container and requires processing. The script then uses the obtained URL to download the image temporarily from firebase and processes the using image using the YOLOv5 model to obtain an output containing data of any road damage detected in the image. It then pushes the obtained output data from the model inference including data such as location coordinates which are included in the Kafka message with the URL to the Neo4J database to store the data permanently. Lastly, the successful create query response will trigger an event on the script to query the Neo4J database and produce a message on the separate web dashboard Kafka topic containing the queried data so the web dashboard can update the view with the latest data. Although, such a script can be written in any language, the team utilized python as the team used PyTorch, which is a python package for training the model.

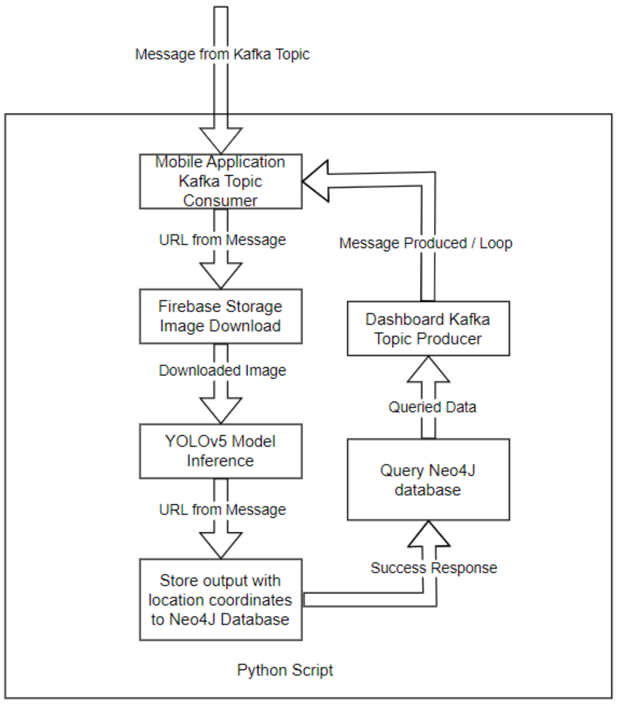


*Figure 7 Python Script Job Structure*

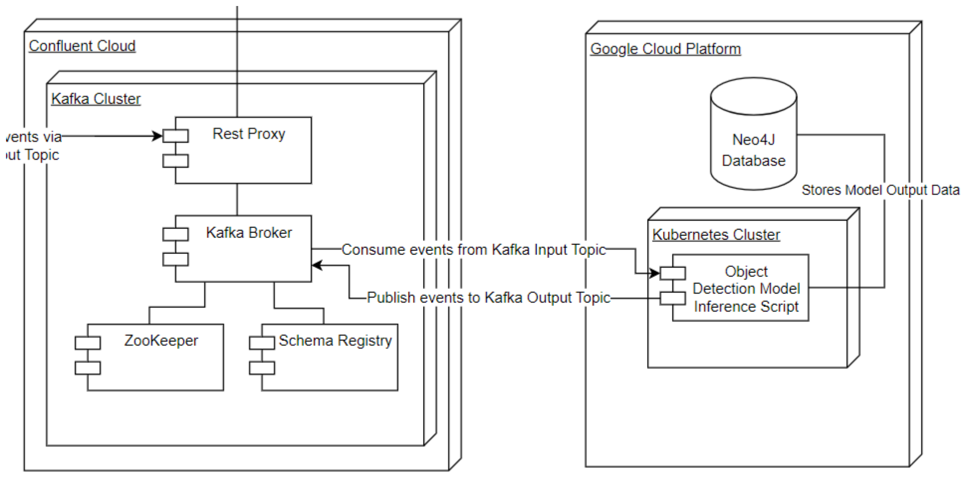


*Figure 8 Relationship between Kafka, Neo4J, and AI Model*

## 5.5 Neo4J Database

We will be building our graph database management system using Neo4j. This database's primary functions will be to assist in the development of the AI model and store data as nodes in native graph storage. The database design that will be employed is a single-node architecture, or a separate Neo4j server running on Google Cloud Platform (GCP). The Neo4j database will need to be installed as a single instance on a virtual machine within the cloud architecture. One of the key benefits of having a single-node architecture is to create and manage a simple cloud topology. We will be able to guarantee that the data being processed and saved is consistent by configuring a single instance of the database. Also, it will assist us in avoiding synchronization problems while using a distributed database. Additionally, we believe that utilizing a standalone architecture will enable us to enhance the database performance and modify it to better accommodate the unique workload and data access patterns. We will concentrate on a standalone architecture for testing needs. However, there can be some restrictions that have an impact on our application's fault tolerance and scalability. If the workload exceeds the virtual machine's capacity, we may need to add more resources or even think about switching to a multi-instance architecture style. Building a Neo4j cluster on Google Cloud Platform could be another potential scaling approach.

The activities of other application components will start a detailed flow of data that is processed and stored in our Neo4j database. The image will be delivered to the Kafka cluster as soon as the user uploads or takes a picture using the mobile application. The image will be delivered to the YOLOv5 object detection instance from the Kafka cluster. A script that connects to the Neo4j database, stores model data based on nodes, verifies the data, and exports the results as a graph model will be present in the YOLOv5 object detection instance. We will be able to export the data into other parts of our application, such as the client interface dashboard one the create query for the nodes is executed to store model output. The data will be used to display the nodes on a map once it has been extracted from the graph model. Each node would be an image, with the nodes' subsequent characteristics being data like classification, size, type, etc. The query result will be added as a message and published to the web dashboard Kafka topic and consumed back to the front end to view the displayed data. An interactive dashboard with map visualization and additional graphs to show the data model will be made using Neo Dash.

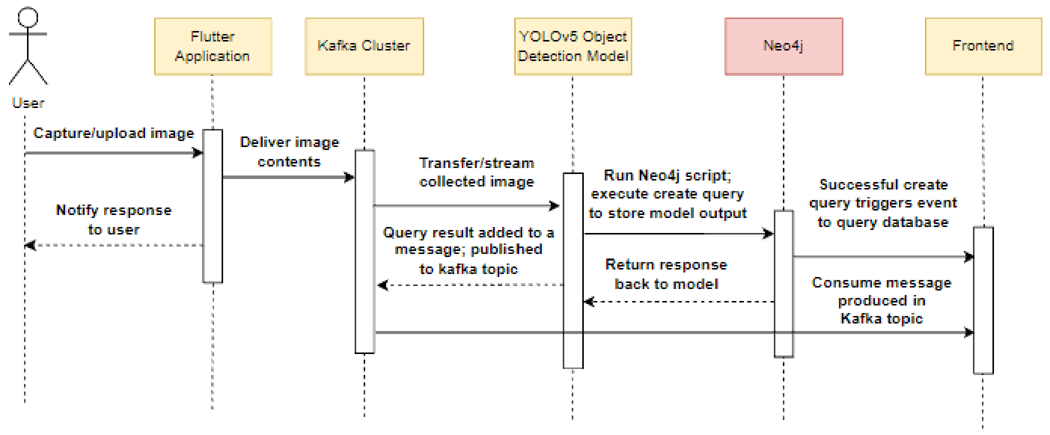

User
Flutter Application
Kafka Cluster
YOLOv5 Object Detection Model
Neo4j
Frontend
Capture/upload image
Deliver image contents
Transfer/stream collected image
Run Neo4j script; execute create query to store model output
Successful create query triggers event to query database
Notify response to user
Query result added to a message; published to kafka topic
Return response back to model
Consume message produced in Kafka topic

*Figure 9 Data storing to Neo4j Sequence Diagram*

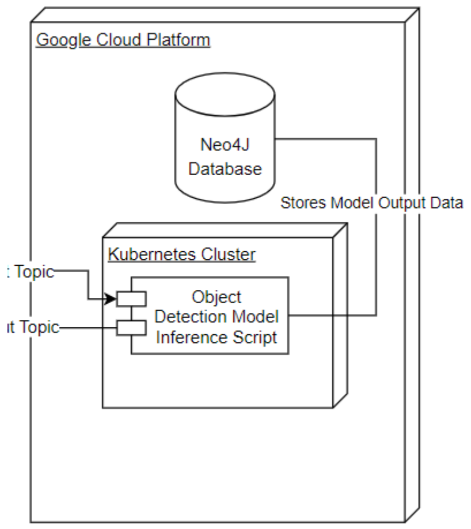


*Figure 10 Relationship between Neo4J and AI Model*

## 5.6 Web Dashboard

Our system requires a dashboard that will be used by our stakeholders (City Employees). It can be accessed by our users from any platform or browser. The main benefit of creating and designing a dashboard is to analyze the data that has been collected and to classify the damages and where it is located. This can be done using open street maps API and data processing tools. When the user logs in to the dashboard, the employee would be able to see many options such as when and where the data has been collected by whom it was collected, and where the damage is on the map. By using the dashboard, the user will analyze the data collected and locate the damages, the time, and the dimensions of a pothole (length, depth, and width). For development, flutter has been used in the implementation of the dashboard. The map will be integrated using flutter map APIs which will allow for real-time updates and visualization. On the main page, some video tutorials will pop up to help the user with the steps required to launch the dashboard and start using it. After logging in with the correct credentials the user would have some options such as start discovering button the button will show all data that has been collected and the exact location on the street open maps this will provide valuable insights for city employees to locate the damage and do the necessary repairs in a timely manner. The data on the map will be updated if any message containing data is consumed by the Kafka consumer running on the web dashboard. This will help the dashboard maintain the live view of the data collected by the system. In terms of future goals, we plan on essentially having a notification system that would notify the relevant authorities, such as the local municipality, of the identified road surface deterioration. This would be done automatically based on the location of the road surface images.

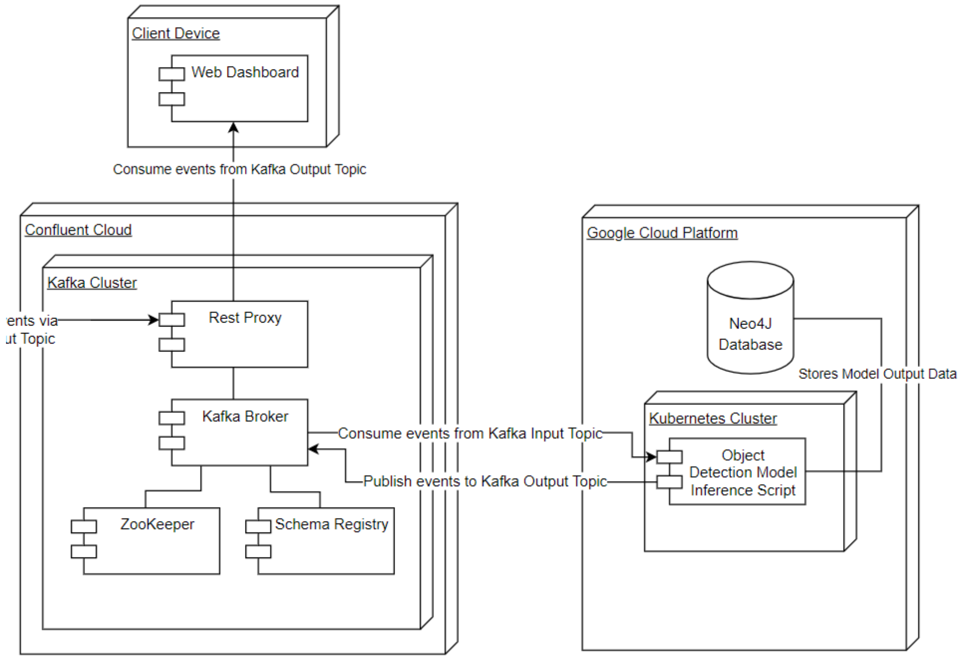


*Figure 11 Relationship between Apache Kafka, Neo4J, AI Model, and Client Web Dashboard*

## 6 Unit Tests

Unit testing is the process in which individual components of the software application are tested separated from the whole system to improve the overall design of the software. The following unit tests are created to ensure that each component or unit of code works as expected.

### 6.1 YOLOv5 Object Detection Model Testing

The dataset that was used to train the YOLOv5 object detection model used in the system was split into a training set and validation set. As supervised learning was used to train the model, testing against a validation set can help predict how well the model will perform on data outside of the training set. The following image below shows the graphs of the various metrics that were used to measure the performance of the model on both the training and validation sets to ensure that it can correctly detect road damage from images.

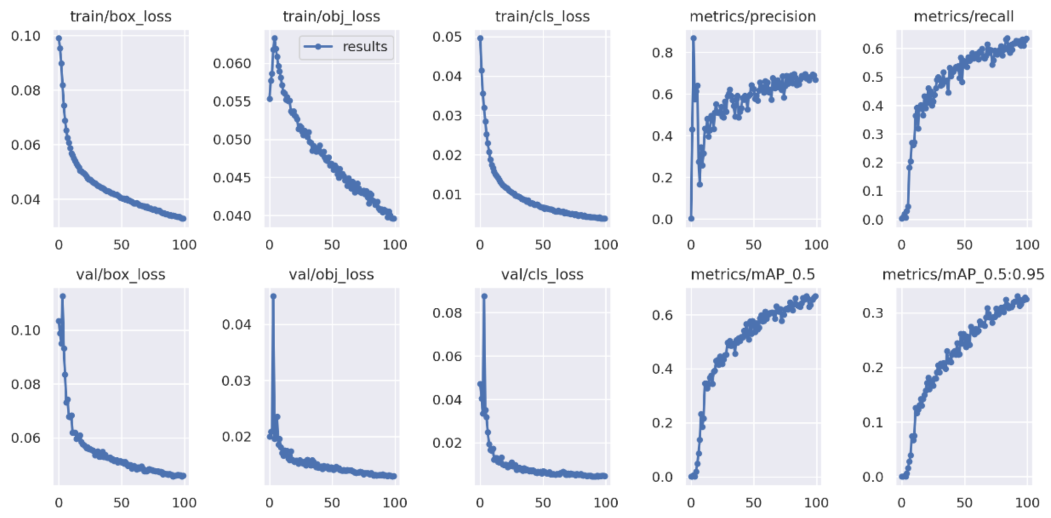


*Figure 12 YOLOv5 Model Performance Metrics*

Box loss is defined as the metric to measure how well a predicted bounding box is able to center and cover the detected object. From the graphs in the first column of the figure above, it was observed that as the model is trained further the value of the box loss in both the training and validation sets approaches zero. This is the desired behavior for an object detection model as the model is able to accurately detect and bound the object tightly. The objective loss shown in the graphs in the second column of the figure show the model's ability to detect all the damages present in the image. It can be observed that as the model was trained further on the training set it was able to detect all the present objects as it learned the patterns to detect the damage. However, on the validation set the model was able to immediately detect more amounts of data which is expected as a trained model will perform better during validation than training. The next most important metric to consider is the classification loss as shown in the third column of the figure. Classification loss is defined as a metric to measure how accurately the model can classify the detected objects between the different damage type class labels. Similar to the box loss graph, it was observed that the classification loss also approaches zero which is desired and means that the model is able to accurately classify the detected objects. Lastly, the graphs in the last two columns of the figure demonstrate the various precision and the recall related metrics. Precision is defined as the ability of the model to classify positive (damages) observations and recall is defined as the ability of the model to detect positive instances from all the actual positive instances. It can be observed from the graphs that as the model is trained further, it is able to accurately detect and classify the damages in both the training and validation sets.

## 6.2 Flutter Mobile Application

The following table describes all the unit test cases implemented on the Flutter Mobile Application:

*Table 5 Unit Tests for mobile application*

| Test Case ID | Test Case | Test Case Description | Test Steps | Test Data | Results |
|---|---|---|---|---|---|
| TC01 | User Permission Test | Check if the user has granted permission for accessing the device's camera and location | 1. Install application<br>2. Configure account<br>3. Accept user permission prompts for devices camera and location | Location services permission = granted<br><br>Camera services permission = granted | **Expected**<br>Users can use the camera and location services<br><br>**Actual**<br>As expected |
| TC02 | UI Test | Verify that the user interface displays correctly, and all buttons are functional | 1. Email access into the app<br>2. Click Submit button to check functionality<br>3. Click camera button to check functionality | Submit button<br><br>Camera button | **Expected**<br>Once buttons are clicked user will be navigated to the next widget<br><br>**Actual**<br>As expected |
| TC03 | Location Accuracy Test | Ensure that the application accurately detects the device's location | 1. Click camera button to check functionality<br>2. AI instance will display the device's location | Image taken from camera<br><br>Device's current location | **Expected**<br>Once camera button is clicked the device's current location should be displayed<br><br>**Actual**<br>As expected |
| TC04 | Kafka Test | Publishing a record or event to the Kafka broker. | 1. Publish the record<br>2. Check if the result code is 200. | The record or event<br><br>Result Code | **Expected**<br>If the result code ends up being 200 the test passes |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  |  |  | 3. If it is 200 it passes, if not it fails. |  | **Actual**<br>As expected |
| TC05 | Firebase Test | Ensure the user is able to upload an image to firebase | 1. Upload the image<br>2. Check if the image has been uploaded | Image taken from camera | **Expected**<br>If the result code ends up being uploaded the test passes<br><br>**Actual**<br>As expected |

## 7 Integration Tests

Integration testing is the process of testing if the individual components of the software system can interact and work together to achieve the intended function as designed. The following integration tests were used to test if the components can function together to achieve and meet the requirements of the project:

### 7.1 Integration Test 1: Mobile Flutter Application – Firebase Cloud Storage – Kafka Broker – YOLOv5 Object Detection Model Inference Script Integration Test

*Table 6 Integration Test 1: Mobile Flutter Application – Firebase Cloud Storage – Kafka Broker – YOLOv5 Object Detection Model Inference Script*

| **Test Name** | Mobile Flutter Application – Firebase Cloud Storage – Kafka Broker – YOLOv5 Object Detection Model Inference Script Integration Test |  |  |  |  |  |
|---|---|---|---|---|---|---|
| **Test Description** | The system should be able to capture an image using the mobile Flutter application and upload the image to Firebase Cloud Storage. Next, the mobile application should send a message containing the URL of the captured image and the current location of the mobile device to a Kafka topic. The python script that is used for the YOLOv5 model inference should consume the Kafka topic and receive the produced message containing the URL required to download the image from Firebase Cloud Storage and the location of the device when the image was captured. |  |  |  |  |  |
| **Test** | **Action** | **Expected Output** | **Pass** | **Fail** | **N/A** | **Comments** |
| 1 | Mobile Flutter Application captures an image and uploads the image to the Firebase Cloud Storage. | The uploaded image should appear on the cloud storage and the Mobile Flutter Application should | ✓ |  |  | Uploaded image was observed on the Firebase Cloud |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  |  | receive the URL to download that image. |  |  |  | Storage Bucket |
| 2 | Mobile Flutter application sends a message containing the location of the device and the URL of the uploaded image to a Kafka topic. | The produced message should be received by the Kafka broker hosted on Confluent Cloud. | ✓ |  |  | Message was received and observed on the Kafka Broker |
| 3 | The script job created for YOLOv5 Object Detection Model inference consumes the Kafka topic. | The message produced by the mobile application containing the URL and the location of the device when the image was captured. | ✓ |  |  | The script logs the message was received and the message data contains the expected values |
| 4 | The model inference script downloads the image from Firebase Cloud Storage. | The image is downloaded successfully and processed by the YOLOv5 Object Detection Model. | ✓ |  |  | The image was processed by the model and the expected output from the model was observed |
| **Overall Test Result** |  |  | ✓ |  |  |  |

## 7.2 Integration Test 2: YOLOv5 Object Detection Model Inference Script – Neo4J Database – Web Dashboard Integration Test

*Table 7 Integration Test 2: YOLOv5 Object Detection Model Inference Script – Neo4J Database – Web Dashboard*

| **Test Name** | YOLOv5 Object Detection Model Inference Script – Neo4J Database – Web Dashboard Integration Test | | | | | |
|---|---|---|---|---|---|---|
| **Test Description** | The system should be able to store the output from the YOLOv5 Object Detection Model produced by processing an image to the Neo4J database. After storing the extracted data to the database, the script should produce an event to query the database to update the view on the web dashboard. | | | | | |
| **Test** | **Action** | **Expected Output** | **Pass** | **Fail** | **N/A** | **Comments** |
| 1 | The YOLOv5 Object Detection Model Inference script executes a create query to store the model output to the database. | The output data should be observed on the Neo4J database. | ✓ | | | The expected output data was observed on the Neo4J database dashboard. |
| 2 | The successful create query on the inference script should trigger an event to query the Neo4J database. | The event trigger log should be produced by the script and Neo4J database should be queried successfully. | ✓ | | | The logs for the event triggering and the database query being successful are observed. |
| 3 | The result of the query should be added to a message that is published to the web dashboard Kafka topic. | The produced message should be received by the Kafka broker hosted on Confluent Cloud. | ✓ | | | Message was received and observed on the Kafka Broker using the Confluent Cloud Dashboard |
| 4 | The web dashboard consumes message produced to the Kafka topic. | The web dashboard updates it view to display the latest data collected by the system. | ✓ | | | The web dashboard view was updated to show the |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | latest information. |
| **Overall Test Result** | | | ✓ | | | |

## 8 Acceptance Tests and Results

Acceptance tests are used to validate if the developed product meets the business requirements and support the identified use cases of the project. In this section, the acceptance tests for the road damage detection system are refined as the project requirements have evolved as the project progressed due to changing requirements. The acceptance tests have been categorized into 4 different stages of the system. The results and validation for each of the acceptance tests are also provided in this section to validate if the developed system meets the business requirements of this project.

### 8.1 Data Collection and Event Streaming

*Table 8 Test Case 1*

| **Test Case ID** | TC1 | | | | | |
|---|---|---|---|---|---|---|
| **Test Case Description** | The system should be able to capture images using mobile devices, record the geographical location of the device, and publish an event containing the collected data. | | | | | |
| **Test** | **Action** | **Expected Output** | **Pass** | **Fail** | **N/A** | **Validation** |
| 1 | User captures image using mobile device with possible road damages | The mobile application should collect the location data of the device when the image was captured and store that data along with the captured image. | ✓ | | | The system can capture images and collect location data when the image is captured (See Figure 15). |
| 2 | Mobile application collects the image and location of the mobile device | The model should store the image in a cloud hosted storage and retrieve a reference URL to access the image. | ✓ | | | The system stored the image in firebase storage. Firebase with URL reference (See Figure 16). |
| 3 | Mobile application successfully stores the image | The mobile application should create an event containing the location data, and the URL to access the captured | ✓ | | | The system successfully created an event containing location, URL to |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  | to a cloud hosted storage | image to an event streaming topic. |  |  |  | the image (See Figure 17). |
| **Overall Test Result** |  |  | ✓ |  |  |  |

## 8.2 Road Damage Detection and Classification

*Table 9 Test Case 2*

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| **Test Case ID** | TC2 |  |  |  |  |  |
| **Test Case Description** | The system should be able consume the events created from data collection; download the captured image; process the image using a trained object detection model to detect, classify, and extract features of any existing road damages. |  |  |  |  |  |
| **Test** | **Action** | **Expected Output** | **Pass** | **Fail** | **N/A** | **Validation** |
| 1 | System consumes a published event on the event streaming topic. | The system should download the captured image from the cloud hosted storage using the image URL in the message data. | ✓ |  |  | The system can consume and process events from “rss_topic” Kafka topic correctly and respond appropriately (See Figure 23). The system downloads the image from firebase storage using the provided URL in the consumed event. |
| 2 | Systems run inference using a trained object detection model on the downloaded image to detect, classify, and extract features of any road damage. | The trained object detection model should detect road damages, correctly classify the type of damages and extract features such as dimensions of the road damages from the image. | ✓ |  |  | The system uses a YOLOv5 (You Only Look Once) model accurately and efficiently to detect, classify, and extract features of road damage from the download image (See Figure 22). |
| **Overall Test Result** |  |  | ✓ |  |  |  |

## 8.3 Data Storage

*Table 10 Test Case 3*

| **Test Case ID** | TC3 | | | | | |
|---|---|---|---|---|---|---|
| **Test Case Description** | The system should be able to store location data, image URL, and the data collected from the model inference in a cloud hosted database. | | | | | |
| **Test** | **Action** | **Expected Output** | **Pass** | **Fail** | **N/A** | **Validation** |
| 1 | Model inference returns extracted data from the image and the system stores that data with the data from the consumed data collection event. | The system should store the data from the consumed data collection event and the model inference into a cloud hosted database. | ✓ | | | The System ensures that the YOLOv5 model has extracted relevant data correctly and stores the extracted data with the data collection event in Neo4j database which stores the data in the form of nodes (See Figure 20). The system can query the database to ensure the data has been stored correctly and will retrieve specific node properties (See Figure 21). |
| **Overall Test Result** | | | ✓ | | | |

## 8.4 Dashboard

*Table 11 Test Case 4*

| **Test Case ID** | TC4 | | | | | |
|---|---|---|---|---|---|---|
| **Test Case Description** | The system should display the stored data in the database on a dashboard containing an interactive map that maps the collected data to geographical locations. | | | | | |
| **Test** | **Action** | **Expected Output** | **Pass** | **Fail** | **N/A** | **Validation** |
| 1 | The system stores the data from the consumed data collection event and data from the | The system should update the dashboard map view by adding a marker to the map for the | ✓ | | | The dashboard should be updated by adding new markers using collected data that |

| | | | | | | |
|---|---|---|---|---|---|---|
| | model inference in a cloud hosted database. | newly added data using the location data collected during the image capture. | | | | is stored in the Neo4J database (See Figure 18). |
| **Overall Test Result** | | | ✓ | | | |

## 9 Ethical Considerations

Developing an IoT application has many ethical dilemmas. Some of these implications include but are not limited to the user's right to privacy, and request to access resources on user's device. The following sections below further describe these considerations.

### 9.1 Data Privacy

Ensuring that the users' data is safe during, and post transit was a critical portion of the system. Although the team utilized many resources during development, we used due diligence to ensure they did not have vulnerabilities that could be exploited. Moreover, we also ensured the applications did not have root privileges to reduce avenues that can be taken advantage of.

### 9.2 Resource Requests

Due to the system's dependency on the user's mobile device, our mobile device application required the use of some sensors on the device. These sensors such as the camera and the GPS, could not be accessed without the user granting our application permission to perform a task.

## 10 Safety Considerations

Due to the system requiring the capturing of road damages, users may need to get as close as possible to the damage to capture it using their mobile cameras. This may inadvertently mean a user may walk on the road in order to capture the damage, possibly obstructing or increasing the risk of a collision. As we have no control over this, the best solution we can implement is to warn the user to be vigilant of traffic.

Another safety consideration the team encountered was protecting the application keys and secrets of our application to protect the user's data. After researching the various methods developers and other applications use to protect their keys and secrets, we were able to implement as well as follow best practices in defending the keys to our applications. We prevented the committing of the keys to our source control, alleviating the avenue for unauthorized use. Moreover, we also employed the use of an encrypted third-party tool that allowed the sharing of application keys and secrets by encrypting an online link and adding a short Time to Live (TTL) on the shared links.

## 11 Conclusions

In conclusion, as a group we were able to successfully implement an image-processing system using machine learning technology to provide a cost-effective and efficient alternative to current road inspection methods. We were able to gain valuable experience and knowledge in software design, development frameworks, and tools such as Flutter, Kafka, YOLOv5, Firebase, and Neo4j. Overall, the proposed product we have developed can have a significant impact on the safety and performance of Canada's road network, ultimately benefiting the country. Moreover, we also learned various software development and management best practices. These include but are not limited to the use of a matured source control (GitHub), utilizing Docker developer containers for consistent development environments, the use of Continuous Integration – Continuous Delivery (CI/CD) tools such as (GitHub Actions) for automating repetitive tasks, and ClickUp for Agile development.

## 12 Updated Project Plan

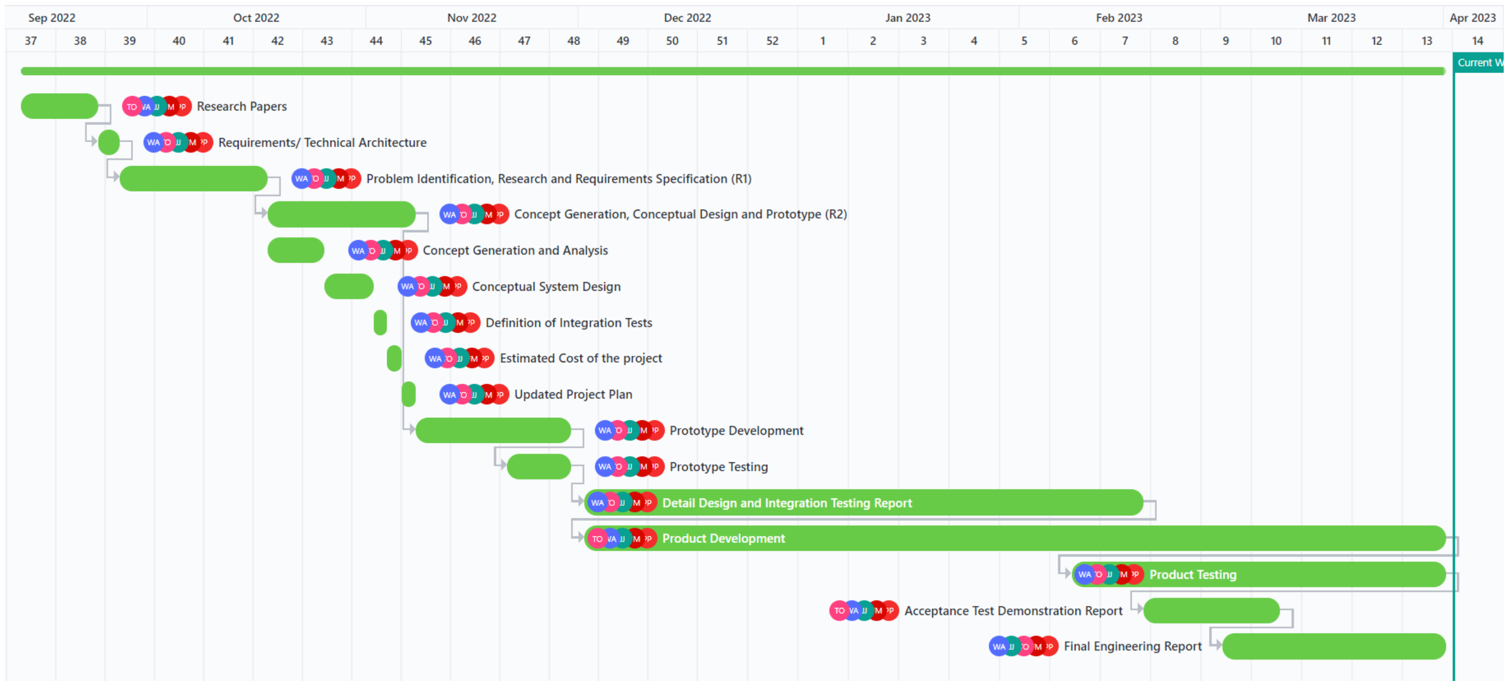
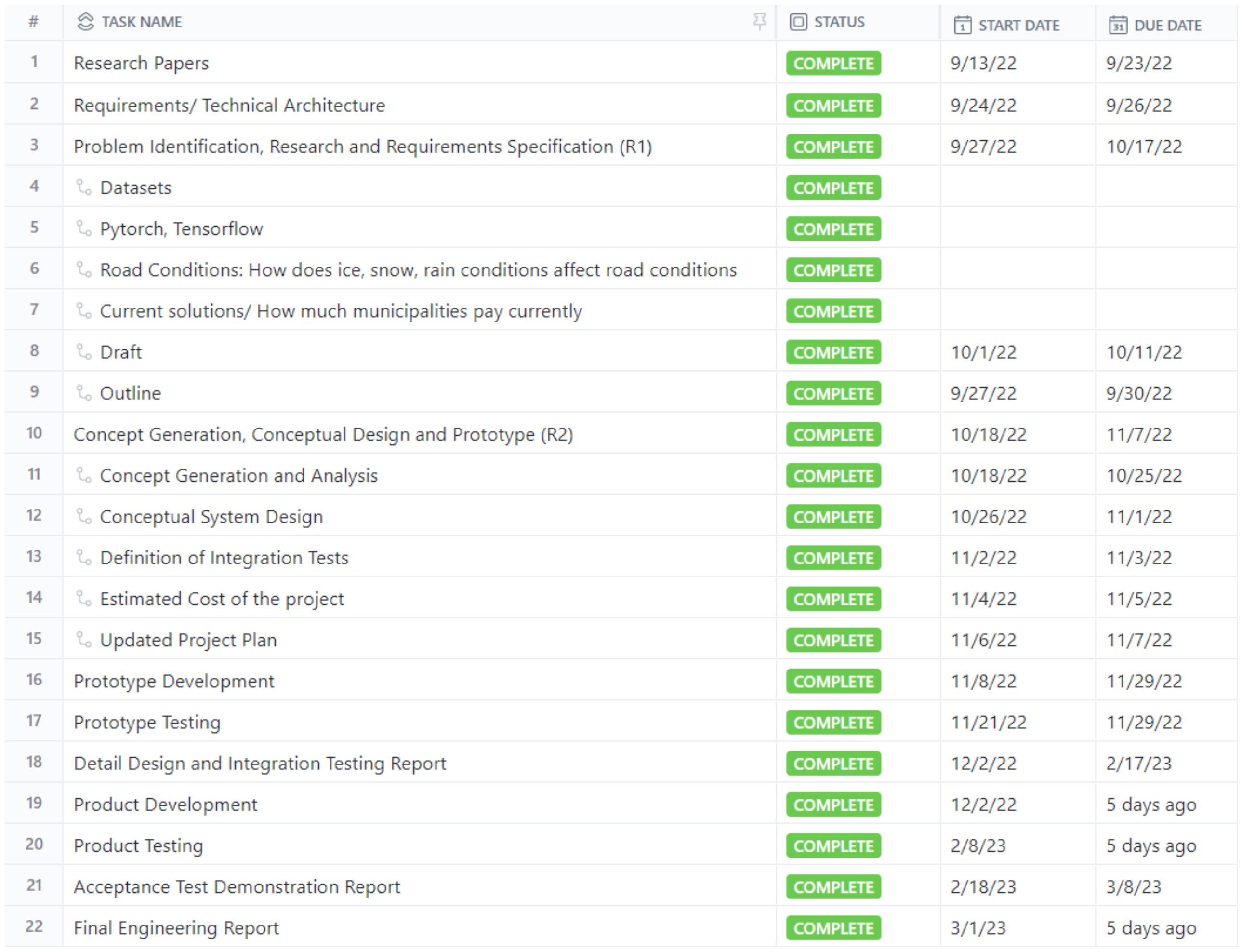


*Figure 13 Updated Project Gantt Chart*



| # | TASK NAME | STATUS | START DATE | DUE DATE |
|---|---|---|---|---|
| 1 | Research Papers | COMPLETE | 9/13/22 | 9/23/22 |
| 2 | Requirements/ Technical Architecture | COMPLETE | 9/24/22 | 9/26/22 |
| 3 | Problem Identification, Research and Requirements Specification (R1) | COMPLETE | 9/27/22 | 10/17/22 |
| 4 | Datasets | COMPLETE | | |
| 5 | Pytorch, Tensorflow | COMPLETE | | |
| 6 | Road Conditions: How does ice, snow, rain conditions affect road conditions | COMPLETE | | |
| 7 | Current solutions/ How much municipalities pay currently | COMPLETE | | |
| 8 | Draft | COMPLETE | 10/1/22 | 10/11/22 |
| 9 | Outline | COMPLETE | 9/27/22 | 9/30/22 |
| 10 | Concept Generation, Conceptual Design and Prototype (R2) | COMPLETE | 10/18/22 | 11/7/22 |
| 11 | Concept Generation and Analysis | COMPLETE | 10/18/22 | 10/25/22 |
| 12 | Conceptual System Design | COMPLETE | 10/26/22 | 11/1/22 |
| 13 | Definition of Integration Tests | COMPLETE | 11/2/22 | 11/3/22 |
| 14 | Estimated Cost of the project | COMPLETE | 11/4/22 | 11/5/22 |
| 15 | Updated Project Plan | COMPLETE | 11/6/22 | 11/7/22 |
| 16 | Prototype Development | COMPLETE | 11/8/22 | 11/29/22 |
| 17 | Prototype Testing | COMPLETE | 11/21/22 | 11/29/22 |
| 18 | Detail Design and Integration Testing Report | COMPLETE | 12/2/22 | 2/17/23 |
| 19 | Product Development | COMPLETE | 12/2/22 | 5 days ago |
| 20 | Product Testing | COMPLETE | 2/8/23 | 5 days ago |
| 21 | Acceptance Test Demonstration Report | COMPLETE | 2/18/23 | 3/8/23 |
| 22 | Final Engineering Report | COMPLETE | 3/1/23 | 5 days ago |

*Figure 14 Updated Project Plan Table*

*Table 12 Team roles and contributions for each developmental task*

| | YOLOv5 Object Detection Model and Script | Neo4J - Graph Database | Kafka - Event Streaming | Mobile application | Web Dashboard |
|---|---|---|---|---|---|
| Preet Patel | 40% | 15% | 15% | 15% | 15% |
| Faraaz Mohsin | 15% | 40% | 15% | 15% | 15% |
| Tiwaloluwa Ojo | 15% | 15% | 40% | 15% | 15% |
| Janajan Jeyabalan | 15% | 15% | 15% | 40% | 15% |
| Waleed El-Alawi | 15% | 15% | 15% | 15% | 40% |

# 13 References

[1] G. Ochoa-Ruiz, A. A. Angulo-Murillo, A. Ochoa-Zezzatti, L. M. Aguilar-Lobo, J. A. Vega-Fernández and S. Natraj, "An Asphalt Damage Dataset and Detection System Based on RetinaNet for Road Conditions Assessments," *Applied Sciences,* vol. 10, no. 11, 2020.

[2] H. Zhang, Z. Wu, Y. Qiu, X. Zhai, Z. Wang, P. Xu, Z. Liu, X. Li and N. Jiang, "A New Road Damage Detection Baseline with Attention Learning," *Applied Sciences,* vol. 12, no. 15, 2022.

[3] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, A. Mraz, T. Kashiyama and Y. Sekimoto, "Transfer Learning-based Road Damage Detection for Multiple Countries".

[4] L. P. Crevier and Y. Delage, "A new model for road-condition forecasting in Canada," *Meteorological Service of Canada,* vol. 1, p. 12, 2020.

[5] F. Malin, I. Norros and S. Innamaa, "Accident risk of road and weather conditions on different road types," *Accident Analysis and Prevention,* vol. 1, pp. 181-188, 2019.

[6] K. P. George, A. S. Rajagopal and L. K. Lim, "Models for Prediciting Pavement Deterioration," *The TRIS and ITRD database,* vol. 1, p. 7, 1989.

[7] S. Adlinge and A. K. Gupta, "Pavement Deterioration and its Causes," *Journal of Mechanical and Civil Engineering,* vol. 1, p. 7, 2019.

[8] R. Nyirandayisabye, L. Huixia and Q. Dong, "Automatic pavement damage predictions using various machine learning algorithms: Evaluation and comparison," *Results in Engineering,* p. 49, 2022.

[9] S. Sattar, S. Li and M. Chapman, "Developing a near real-time road surface anomaly detection approach for," *Measurement,* vol. 185, p. 18, 2021.

[10] C. Chanjun and K. R. Seung, "Road surface damage detetion using fully convolutional neural network demi-supervised," *Road surface damage detetion using fully convolutional.*

[11] "Event streaming: Build applications that keep up with business," 2022. [Online]. Available: https://tanzu.vmware.com/event-streaming#:~:text=What%20is%20event%20streaming%3A%20Streaming,will%20affect%20any%20resulting%20action. [Accessed 6 November 2022].

[12] "Confluent Clound - Pricing," Confluent, [Online]. Available: https://www.confluent.io/confluent-cloud/pricing/. [Accessed 6 November 2022].

[13] P. Nagy, "Kafka for real time stream processing in the real world," 4 August 2021. [Online]. Available: https://www.quix.io/blog/set-up-kafka-for-real-time-stream-processing/#:~:text=Kafka%20is%20lightning-fast%3A%20it%20blows%20one%20second%20out,not%20detectable%20by%20human%20standards.%20The%20short%20answer%3F. [Accessed 07 November 2022].

[14] C. Dulanga, "APIs vs. WebSockets vs. WebHooks: What to Choose?," 17 Feburary 2021. [Online]. Available: https://blog.bitsrc.io/apis-vs-websockets-vs-webhooks-what-to-choose-5942b73aeb9b. [Accessed 6 November 2022].

[15] J. Juviler, "REST APIs: How They Work and What You Need to Know," 30 August 2022. [Online]. Available: https://blog.hubspot.com/website/what-is-rest-api. [Accessed 6 November 2022].

[16] V. Singh, "Flask vs Django in 2022: Which Framework to Choose?," 7 June 2022. [Online]. Available: https://hackr.io/blog/flask-vs-django. [Accessed 6 November 2022].

[17] A. Lee, "Solace Topics vs. Kafka Topics: What's the Difference?," 31 May 2019. [Online]. Available: https://solace.com/blog/solace-topics-vs-kafka-topics/?utm_source=youtube&utm_medium=socialmedia&utm_content=topicvs&utm_campaign=youtube_pubsub. [Accessed 6 November 2022].

[18] "AWS Pricing Calculator," Amazon Web Services, [Online]. Available: https://calculator.aws/#/addService/KinesisDataStreams. [Accessed 6 November 2022].

[19] "Plans and Pricing," Solace, [Online]. Available: https://solace.com/products/prices/. [Accessed 6 November 2022].

[20] V. Shrimali, "PyTorch for Beginners: Image Classification using Pre-trained models," 3 June 2019. [Online]. Available: https://learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/. [Accessed 6 November 2022].

[21] V. Narayanan, "Tutorial — Image Classifier using Resnet50 Deep Learning model (Python Flask in Azure)," 13 October 2019. [Online]. Available: https://medium.com/@venkinarayanan/tutorial-image-classifier-using-resnet50-deep-learning-model-python-flask-in-azure-4c2b129af6d2#:~:text=Why%20are%20we%20using%20Resnet50,useful%20for%20the%20new%20task.. [Accessed 7 November 2022].

[22] Y. Gao and K. M. Mosalam, "Deep Transfer Learning for Image-Based Structural Damage Recognition," *Computer-Aided Civil and Infrastructure Engineering,* vol. 33, no. 9, pp. 748-768, 16 April 2018.

[23] N. Reddy and A. Rattani, "Comparison of Deep Learning Models for Biometric-based Mobile User Authentication," *Ocular Biometrics in Visible Spectrum,* p. 7, 2018.

[24] M. Li, Z. Zhang, L. Lei, X. Wang and X. Guo, "Agricultural Greenhouses Detection in High-Resolution Satellite Images Based on Convolutional Neural Networks: Comparison of Faster R-CNN, YOLO v3 and SSD," *MDPI Open Access Journals,* vol. 20, no. 17, pp. 1-13, 31 August 2020.

[25] O. G.Yalcin, "Towards Data Science," 23 September 2020. [Online]. Available: https://towardsdatascience.com/4-pre-trained-cnn-models-to-use-for-computer-vision-with-transfer-learning-885cb1b2dfc. [Accessed 2 November 2022].

[26] M. Nowak, "Vue vs React in 2022 - Comparison of Two Most Popular JS Frameworks," 08 April 2022. [Online]. Available: https://www.monterail.com/blog/vue-vs-react.

[27] H. Bandyopadhyay, "V7 Labs," 7 October 2022. [Online]. Available: https://www.v7labs.com/blog/yolo-object-detection. [Accessed 2 November 2022].

[28] "Vertex AI Pricing," [Online]. Available: https://cloud.google.com/vertex-ai/pricing#automl. [Accessed 07 November 2022].

[29] "Google Colaboratory," [Online]. Available: https://colab.research.google.com/. [Accessed 7 November 2022].

[30] "Introduction to Vertex AI," [Online]. Available: https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform. [Accessed 7 November 2022].

[31] A. Khan, "How to use mobile camera on raspberry pi," *linuxhint,* vol. 1, no. 1, 2022.

[32] "Alan Blog," [Online]. Available: https://alan.app/blog/types-of-user-interface/. [Accessed 2 November 2022].

[33] X. Li, B. Cui and Y. Chen, "MLog: towards declarative in-database machine learning," *Proceedings of the VLDB Endowment,* vol. 10, no. 12, p. 4, 2017.

[34] D. Joshi, P. T. Singh and S. Gargeya, "Automatic surface crack detection using segmentation-based," *Engineering Fracture Mechanics,* vol. 268, p. 16, 2022.

[35] C. Romero-Ramos, P. Rios-Leon, B. Al-Hadithi, L. Sigcha and C. Asensio, "Identification and mapping of asphalt surface deterioration by tyre-pavement interaction noise measurement," *Measurement,* vol. 146, p. 10, 2019.

[36] S. Bhutad and P. Kailas, *Dataset of road surface images with seasons for machine learing application .*

[37] K. Harjit and K. Rajandeep, "Crack detection and parameters," *Crack detection and parameters estimation on road images.*

[38] T. J. Kwon, L. Fu and C. Jiang, "Effect of Winter Weather and Road Surface Conditions on Macroscopic Traffic Parameters," *Transportation research record,* vol. 2329, no. 1, pp. 54-62, 2013.

[39] M. Kangas, M. Heikinheimo and M. Hippi, "RoadSurf: a modelling system for predicting road weather and road surface conditions," *METEOROLOGICAL APPLICATIONS,* vol. 22, no. 3, pp. 544-553, 2015.

[40] E. Toivonen, M. Hippi, H. Korhonen, A. Laaksonen, M. Kangas and J.-P. Pietikäinen, "The road weather model RoadSurf (v6.60b) driven by the regional climate model HCLIM38: evaluation over Finland," *Copernicus GmbH,* vol. 12, no. 8, pp. 3481-3502, 2019.

[41] "Know and use a deliberate design process | Designing a Solution | 3-5, 6-8," The University of North Carolina at Chapel Hill, 2022. [Online]. Available: https://iste.web.unc.edu/activity/designing-a-solution-3-8/. [Accessed 11 October 2022].

[42] "nvisia," 16 September 2020. [Online]. Available: https://www.nvisia.com/insights/agile-methodology. [Accessed 11 October 2022].

# 14 Appendix

## 14.1 Validation

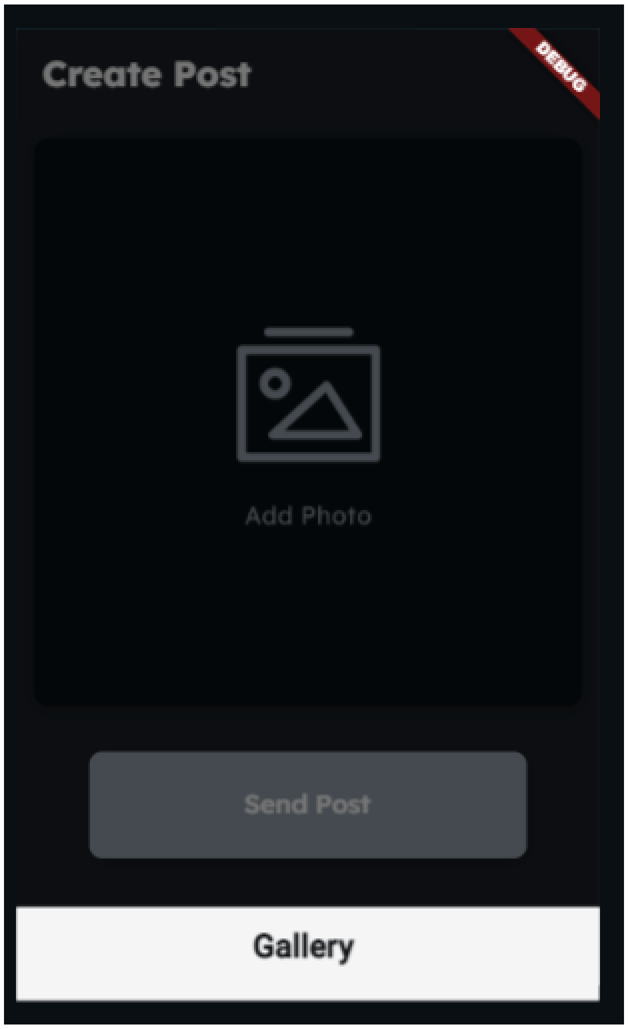


*Figure 15 Capture image from flutter mobile application*

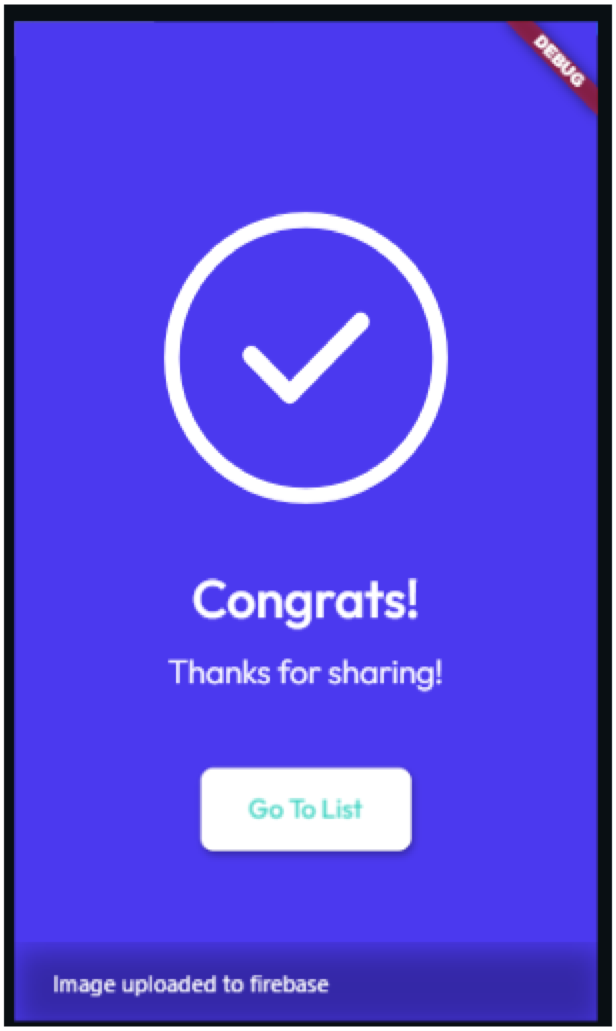
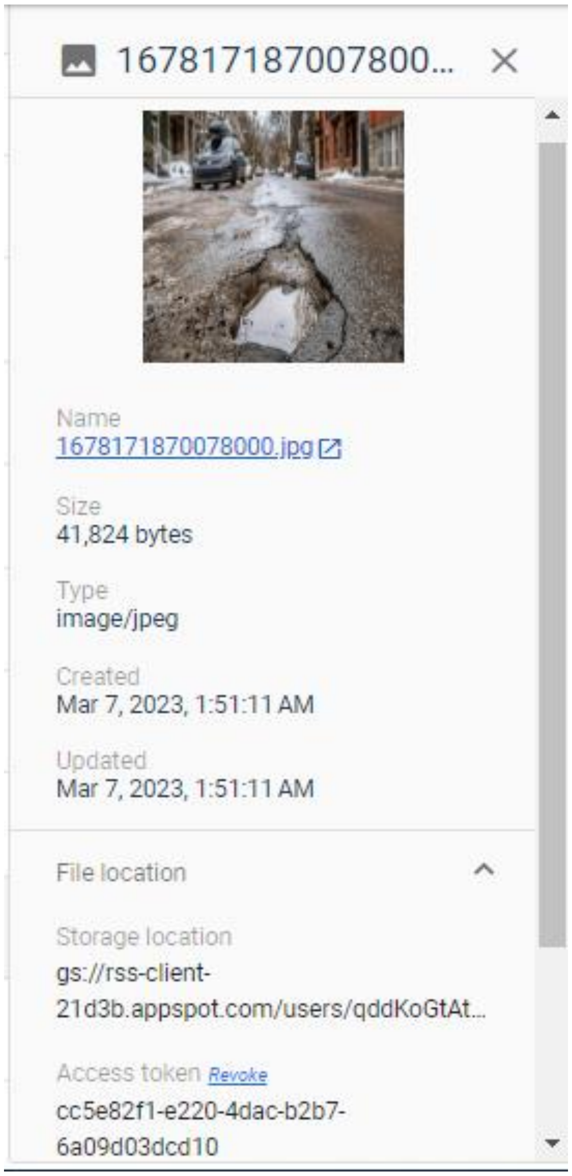


Image uploaded to firebase

*Figure 16 Image uploaded to Firebase*



167817187007800...

Name
1678171870078000.jpg

Size
41,824 bytes

Type
image/jpeg

Created
Mar 7, 2023, 1:51:11 AM

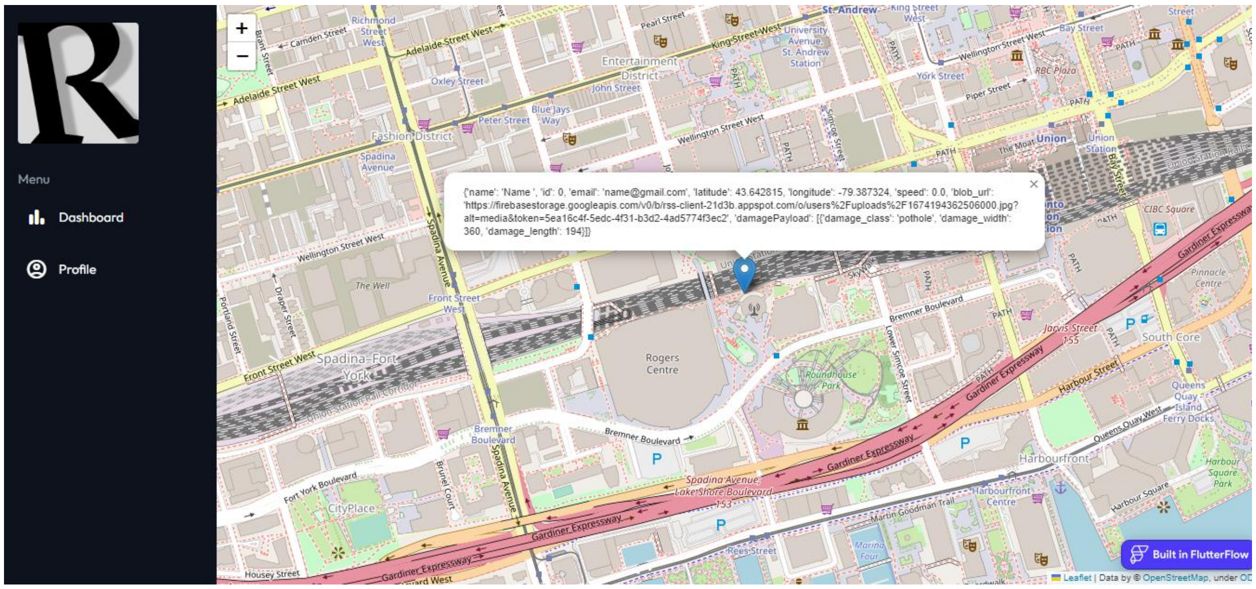
Updated
Mar 7, 2023, 1:51:11 AM

File location

Storage location
gs://rss-client-21d3b.appspot.com/users/qddKoGtAt...

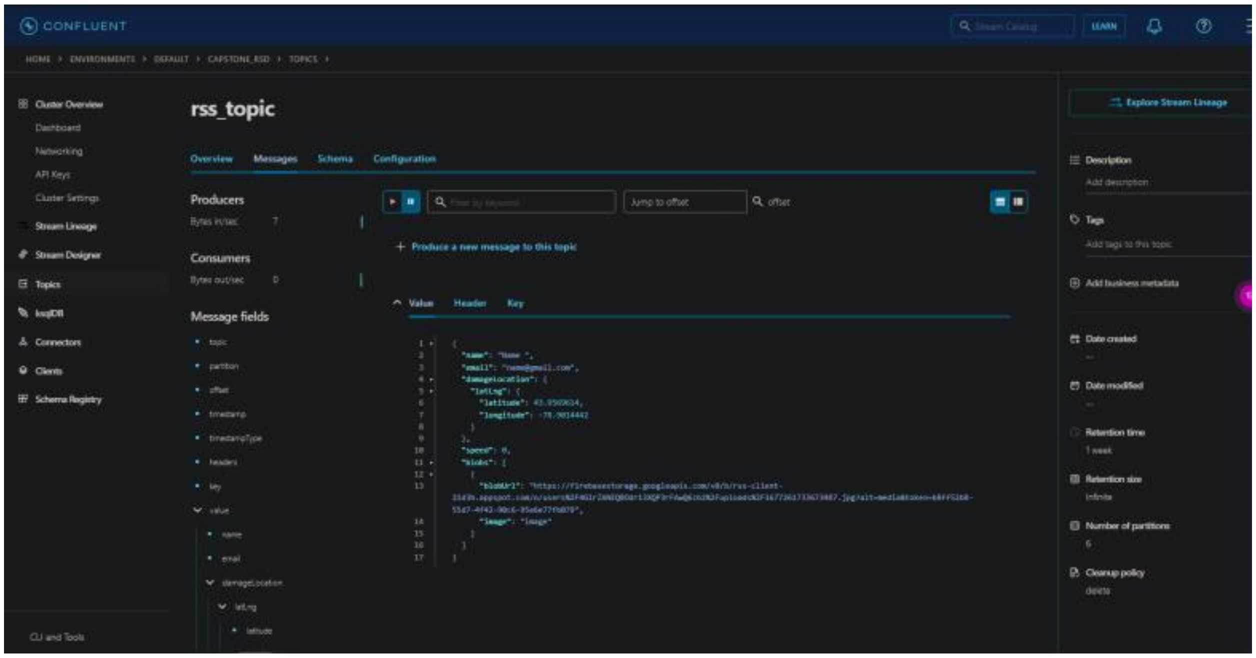
Access token Revoke
cc5e82f1-e220-4dac-b2b7-6a09d03dcd10

*Figure 17 Firebase URL*



```
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.642815, 'longitude': -79.387324, 'speed': 0.0, 'blob_url': 'https://firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4f31-b3d2-4ad5774f3ec2', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 360, 'damage_length': 194}]}
```

*Figure 18 Dashboard*



```
{
  "name": "Name ",
  "email": "name@gmail.com",
  "damageLocation": {
    "latLng": {
      "latitude": 43.6509634,
      "longitude": -79.3814442
    }
  },
  "speed": 0,
  "blobs": [
    {
      "blobUrl": "https://firebasestorage.googleapis.com/v0/b/rss-client-
21d3b.appspot.com/o/users%2F[illegible]%2Fuploads%2F[illegible].jpg?alt=media&token=[illegible]",
      "image": "image"
    }
  ]
}
```

*Figure 19 Kafka Topic and its consumed message*

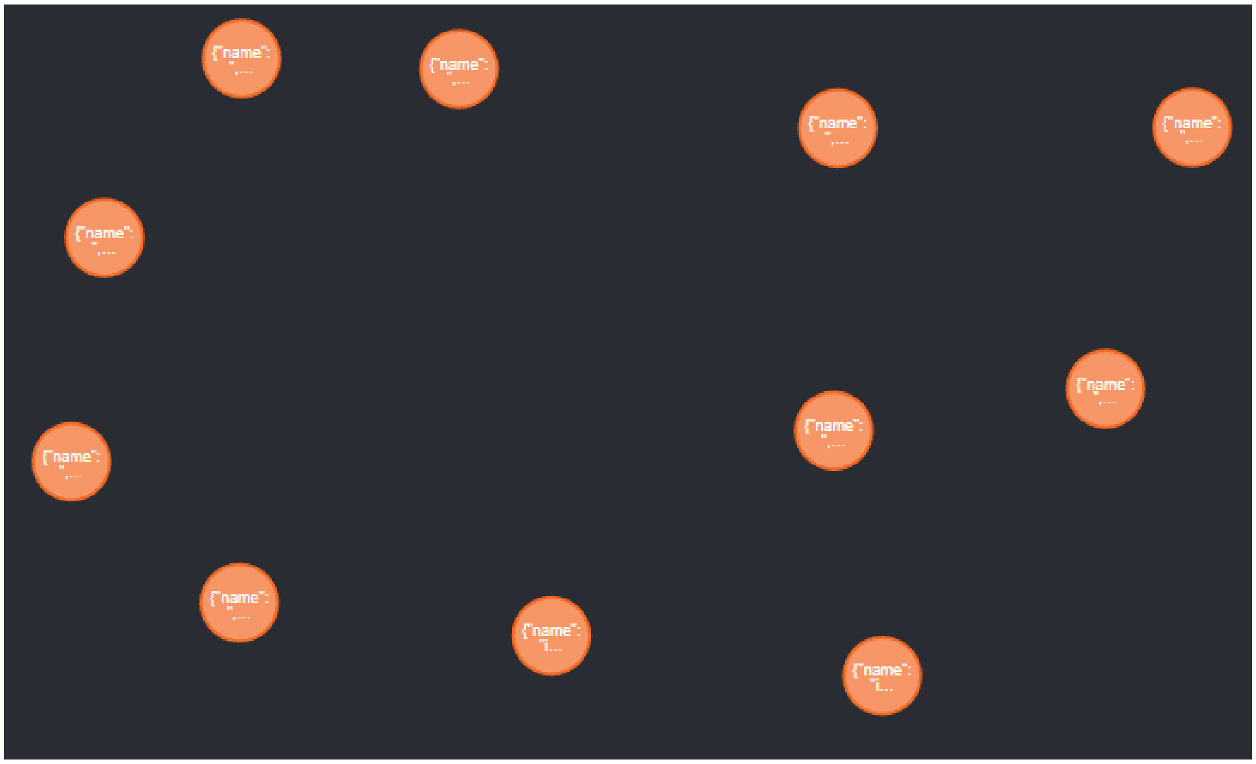
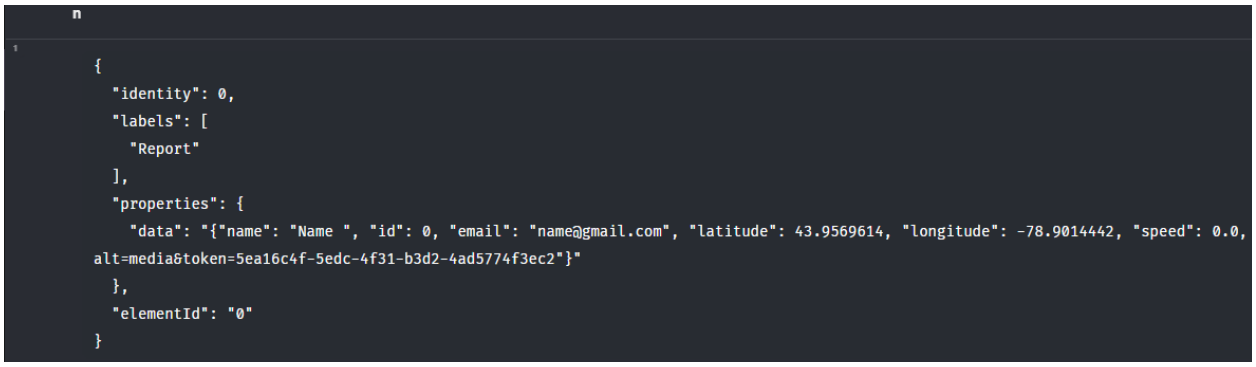
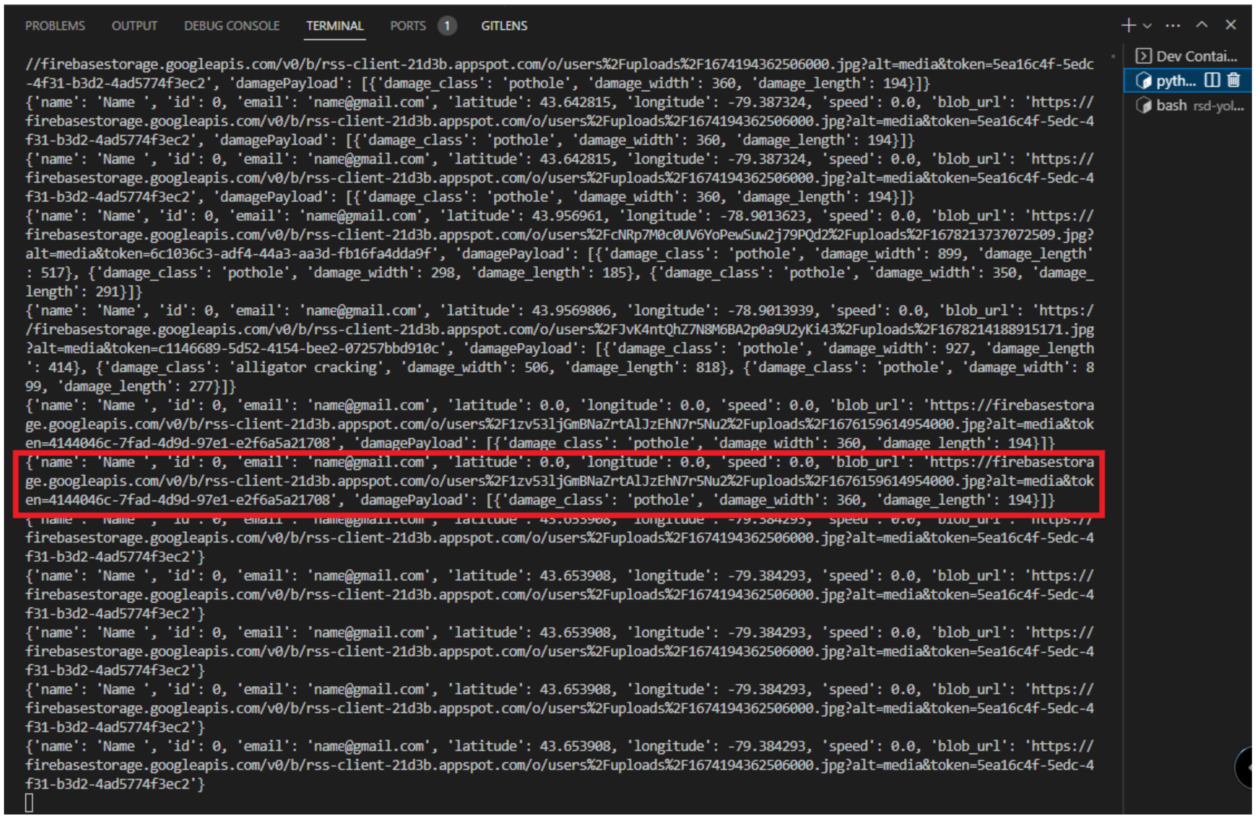


*Figure 20 Nodes from Neo4j Database*



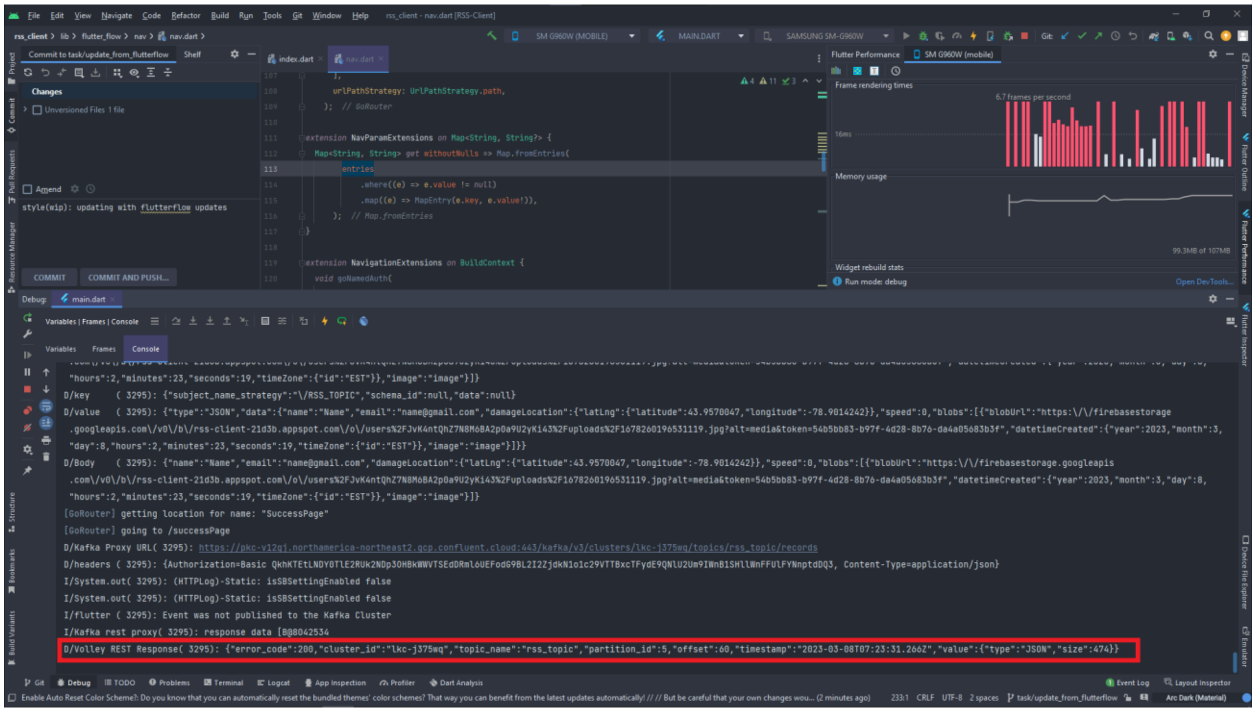
```
n

{
  "identity": 0,
  "labels": [
    "Report"
  ],
  "properties": {
    "data": "{"name": "Name ", "id": 0, "email": "name@gmail.com", "latitude": 43.9569614, "longitude": -78.9014442, "speed": 0.0,
alt=media&token=5ea16c4f-5edc-4f31-b3d2-4ad5774f3ec2"}"
  },
  "elementId": "0"
}
```

*Figure 21 Node properties from Neo4j Database*



```
//firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc
-4f31-b3d2-4ad5774f3ec2', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 360, 'damage_length': 194}]}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.642815, 'longitude': -79.387324, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 360, 'damage_length': 194}]}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.642815, 'longitude': -79.387324, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 360, 'damage_length': 194}]}
{'name': 'Name', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.956961, 'longitude': -78.9013623, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2FcNRp7M0c0UV6YoPewSuw2j79PQd2%2Fuploads%2F1678213737072509.jpg?
alt=media&token=6c1036c3-adf4-44a3-aa3d-fb16fa4dda9f', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 899, 'damage_length'
: 517}, {'damage_class': 'pothole', 'damage_width': 298, 'damage_length': 185}, {'damage_class': 'pothole', 'damage_width': 350, 'damage_
length': 291}]}
{'name': 'Name', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.9569806, 'longitude': -78.9013939, 'speed': 0.0, 'blob_url': 'https:/
/firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2FJvK4ntQhZ7N8M6BA2p0a9U2yKi43%2Fuploads%2F1678214188915171.jpg
?alt=media&token=c1146689-5d52-4154-bee2-07257bbd910c', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 927, 'damage_length
': 414}, {'damage_class': 'alligator cracking', 'damage_width': 506, 'damage_length': 818}, {'damage_class': 'pothole', 'damage_width': 8
99, 'damage_length': 277}]}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 0.0, 'longitude': 0.0, 'speed': 0.0, 'blob_url': 'https://firebasestora
ge.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2F1zv53ljGmBNaZrtAlJzEhN7r5Nu2%2Fuploads%2F1676159614954000.jpg?alt=media&tok
en=4144046c-7fad-4d9d-97e1-e2f6a5a21708', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 360, 'damage_length': 194}]}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 0.0, 'longitude': 0.0, 'speed': 0.0, 'blob_url': 'https://firebasestora
ge.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2F1zv53ljGmBNaZrtAlJzEhN7r5Nu2%2Fuploads%2F1676159614954000.jpg?alt=media&tok
en=4144046c-7fad-4d9d-97e1-e2f6a5a21708', 'damagePayload': [{'damage_class': 'pothole', 'damage_width': 360, 'damage_length': 194}]}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.653908, 'longitude': -79.384293, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2'}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.653908, 'longitude': -79.384293, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2'}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.653908, 'longitude': -79.384293, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2'}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.653908, 'longitude': -79.384293, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2'}
{'name': 'Name ', 'id': 0, 'email': 'name@gmail.com', 'latitude': 43.653908, 'longitude': -79.384293, 'speed': 0.0, 'blob_url': 'https://
firebasestorage.googleapis.com/v0/b/rss-client-21d3b.appspot.com/o/users%2Fuploads%2F1674194362506000.jpg?alt=media&token=5ea16c4f-5edc-4
f31-b3d2-4ad5774f3ec2'}
```

*Figure 22 Model Inference returning road image damages, classification of damages, and extracted features*



```
    urlPathStrategy: UrlPathStrategy.path,
  ); // GoRouter

extension NavParamExtensions on Map<String, String?> {
  Map<String, String> get withoutNulls => Map.fromEntries(
        entries
            .where((e) => e.value != null)
            .map((e) => MapEntry(e.key, e.value!)),
      ); // Map.fromEntries
}

extension NavigationExtensions on BuildContext {
  void goNamedAuth(
```

```
 "hours":2,"minutes":23,"seconds":19,"timeZone":{"id":"EST"}},"image":"image"}]}
D/key     ( 3295): {"subject_name_strategy":"\/RSS_TOPIC","schema_id":null,"data":null}
D/value   ( 3295): {"type":"JSON","data":{"name":"Name","email":"name@gmail.com","damageLocation":{"latLng":{"latitude":43.9570047,"longitude":-78.9014242}},"speed":0,"blobs":[{"blobUrl":"https:\/\/firebasestorage
 .googleapis.com\/v0\/b\/rss-client-21d3b.appspot.com\/o\/users%2FJvK4ntQhZ7N8M6BA2p0a9U2yKi43%2Fuploads%2F1678260196531119.jpg?alt=media&token=54b5bb83-b97f-4d28-8b76-da4a05683b3f","datetimeCreated":{"year":2023,"month":3,
 "day":8,"hours":2,"minutes":23,"seconds":19,"timeZone":{"id":"EST"}},"image":"image"}]}}
D/Body    ( 3295): {"name":"Name","email":"name@gmail.com","damageLocation":{"latLng":{"latitude":43.9570047,"longitude":-78.9014242}},"speed":0,"blobs":[{"blobUrl":"https:\/\/firebasestorage.googleapis
 .com\/v0\/b\/rss-client-21d3b.appspot.com\/o\/users%2FJvK4ntQhZ7N8M6BA2p0a9U2yKi43%2Fuploads%2F1678260196531119.jpg?alt=media&token=54b5bb83-b97f-4d28-8b76-da4a05683b3f","datetimeCreated":{"year":2023,"month":3,"day":8,
 "hours":2,"minutes":23,"seconds":19,"timeZone":{"id":"EST"}},"image":"image"}]}
[GoRouter] getting location for name: "SuccessPage"
[GoRouter] going to /successPage
D/Kafka Proxy URL( 3295): https://pkc-v12gj.northamerica-northeast2.gcp.confluent.cloud:443/kafka/v3/clusters/lkc-j375wq/topics/rss_topic/records
D/headers ( 3295): {Authorization=Basic QkhKTEtLNDY0TlE2RUk2NDp3OHBkWWVTSEdDRml6UEFodG9BL2I2ZjdkN1o1c29VTTBxcTFydE9QNlU2Um9IWnB1SHlLWnFFULFYNnptdDQ3, Content-Type=application/json}
I/System.out( 3295): (HTTPLog)-Static: isSBSettingEnabled false
I/System.out( 3295): (HTTPLog)-Static: isSBSettingEnabled false
I/flutter ( 3295): Event was not published to the Kafka Cluster
I/Kafka rest proxy( 3295): response data [B@8042534
D/Volley REST Response( 3295): {"error_code":200,"cluster_id":"lkc-j375wq","topic_name":"rss_topic","partition_id":5,"offset":60,"timestamp":"2023-03-08T07:23:31.266Z","value":{"type":"JSON","size":474}}
```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
rss_client - nav.dart [RSS-Client]
SM G960W (MOBILE)
MAIN.DART
SAMSUNG SM-G960W
index.dart
nav.dart
Commit to task/update_from_flutterflow
Shelf
Changes
Unversioned Files 1 file
Amend
style(wip): updating with flutterflow updates
COMMIT
COMMIT AND PUSH...
Flutter Performance
SM G960W (mobile)
Frame rendering times
67 frames per second
16ms
Memory usage
99.3MB of 107MB
Widget rebuild stats
Run mode: debug
Open DevTools...
Debug:
main.dart
Variables | Frames | Console
Variables
Frames
Console
Git
Debug
TODO
Problems
Terminal
Logcat
App Inspection
Profiler
Dart Analysis
Event Log
Layout Inspector
Enable Auto Reset Color Scheme?: Do you know that you can automatically reset the bundled themes' color schemes? That way you can benefit from the latest updates automatically! // // But be careful that your own changes wou... (2 minutes ago)
233:1 CRLF UTF-8 2 spaces
task/update_from_flutterflow
Arc Dark (Material)

*Figure 23 Mobile app publishes event to Kafka topic "rss_topic"*

## 15 Contribution Matrix

*Table 13 Contribution Matrix*

| Task | Preet Patel | Faraaz Mohsin | Tiwaloluwa Ojo | Janajan Jeyabalan | Waleed El-Alawi |
|---|---|---|---|---|---|
| Abstract | 60% | 10% | 10% | 10% | 10% |
| Introduction | 10% | 10% | 35% | 10% | 35% |
| Background and research review | 20% | 20% | 20% | 20% | 20% |
| Scenario and Use Cases | 5% | 20% | 5% | 20% | 50% |
| Detailed Design | 20% | 20% | 20% | 20% | 20% |
| Unit Tests | 25% | 10% | 15% | 25% | 25% |
| Integration Tests | 35% | 35% | 10% | 10% | 10% |
| Acceptance Tests | 30% | 30% | 10% | 15% | 15% |
| Ethical Considerations | 15% | 15% | 40% | 15% | 15% |
| Safety Considerations | 15% | 15% | 15% | 40% | 15% |
| Conclusion | 15% | 40% | 15% | 15% | 15% |