# Logo, company name Description automatically generated

# **A Distributed IoT System for Non-Destructive Evaluation of Fitness and Integrity of Concrete Buildings and Structures. Architectural Design and Practical Demonstration.**

A dissertation submitted in partial fulfilment of the requirements of Glasgow Caledonian University for the degree of Master of Science in Big Data Technologies

This project report is my own original work and has not been submitted elsewhere in fulfilment of the requirements of this or any other award.

Guillermo Carlos Navarro Magana

S1933686

# Abstract

Non-destructive evaluations (NDE) are methods used in infrastructure to analyse and audit health of infrastructures in a non-invasive manner. Among these techniques, visual inspection is a popular method due to the ease, reduced cost, and lack of specialized tools to inspect infrastructures. Auditors and professionals in the construction fields can be exposed to unnecessary risks where these NDE are carried in exposed areas, such as high places or areas with little to no protection for the professionals. Moreover, in regions where erosion or damage to infrastructures and buildings is constant, such as areas prone to periodic earthquakes, high frequency movements, or to corrosive environments, periodic audits may be required, elevating costs associated with infrastructure maintenance.

Internet of Things (IoT) are devices with embedded sensors that have network capabilities to connect to internet or networks. IoT devices have revolutionized a multitude of industries due to the low costs, accessibility and simplicity, size, power requirements, and flexibility these devices offer. Moreover, with the aid of data analytics and machine learning (ML), the ability to retrieve and uncover information, as well as to improve processes and tools can increase in different sectors. However, existing architectural designs for distributed system, related to distributed systems integrating IoT and ML tends to limit the abstraction and detail levels that describe these distributed systems and emphasize on the edge components of IoT systems.

This dissertation proposes a 7-layer architecture design for a distributed system that incorporates two principal components, IoT and ML, through a cloud platform to evaluate and demonstrate an NDE distributed system capable of identifying and informing structural integrity. Additionally, a practical demonstration of a prototype was developed to recommend a viable and low-cost NDE additional tool for professionals in the construction industry. The prototype incorporates a simulation IoT sandbox emulating an IoT device that, connected through a cloud platform, transmits images through a binary classification ML model, and detects fissure events in concrete structure from images transmitted by the emulated IoT device. The information that is generated from the emulated devices is presented to end-users through a dashboard developed through a Platform-as-a-Service (PaaS) cloud tool.

# Acknowledgments

I would like to acknowledge and express my gratitude to Peter Barrie, who has guided and aided me throughout the creation of this dissertation, whose feedback and comments helped me through all the challenges and difficulties of this work.

Lastly, I thank my family and my friends, whose support, help, and encouragement has been a ray through my studies and work.

Contents

[Abstract 2](#_Toc92472979)

[Acknowledgments 3](#_Toc92472980)

[Acronyms and abbreviations 6](#_Toc92472981)

[Table of Figures 7](#_Toc92472982)

[Table of Tables 9](#_Toc92472983)

[1. Introduction and system definition 10](#_Toc92472984)

[1.1. Background 10](#_Toc92472985)

[1.2. Problem statement and scope 10](#_Toc92472986)

[1.3. Structure of the Dissertation 12](#_Toc92472987)

[2. Literature review 13](#_Toc92472988)

[2.1. Structures Inspection Techniques 13](#_Toc92472989)

[2.1.1. Traditional Inspection Techniques 13](#_Toc92472990)

[2.1.2. Emerging and Experimental Inspection Techniques 15](#_Toc92472991)

[2.2. Machine Learning in Structure Analysis 16](#_Toc92472992)

[2.3. IoT System Architecture Patterns 17](#_Toc92472993)

[3. Technology review 22](#_Toc92472994)

[3.1. Cloud Platforms 22](#_Toc92472995)

[3.1.1. PaaS 22](#_Toc92472996)

[3.1.2. Database Integration 24](#_Toc92472997)

[3.1.3. IoT Services 25](#_Toc92472998)

[3.1.4. ETL Pipeline System and Orchestration 26](#_Toc92472999)

[3.1.5. AI platform 27](#_Toc92473000)

[3.2. IoT Simulation Sandbox 28](#_Toc92473001)

[3.3. Low Energy Communication Protocols 29](#_Toc92473002)

[4. Proposed design 32](#_Toc92473003)

[4.1. Technical requirements 32](#_Toc92473004)

[4.1.1. Cloud Platform Provider 32](#_Toc92473005)

[4.1.2. IoT Simulation Sandbox 34](#_Toc92473006)

[4.1.3. Machine Learning 35](#_Toc92473007)

[4.1.4. Dashboard 36](#_Toc92473008)

[4.2. End-to-end architecture design 37](#_Toc92473009)

[4.2.1. Edge Layer 38](#_Toc92473010)

[4.2.2. Middleware layer 39](#_Toc92473011)

[4.2.3. Connectivity Layer 40](#_Toc92473012)

[4.2.4. Cloud Infrastructure Layer 41](#_Toc92473013)

[4.2.5. Data Ingestion Layer 42](#_Toc92473014)

[4.2.6. Data Analysis Layer 43](#_Toc92473015)

[4.2.7. Application Layer 44](#_Toc92473016)

[4.2.8. People & Process Layer 44](#_Toc92473017)

[5. Development and testing 46](#_Toc92473018)

[5.1. IoT Device Simulation and Integration 46](#_Toc92473019)

[5.2. Data Pipeline 49](#_Toc92473020)

[5.3. Crack-detection AI Model 50](#_Toc92473021)

[5.3.1. Training, testing, and validation datasets 50](#_Toc92473022)

[5.3.2. Model performance metrics 55](#_Toc92473023)

[5.4. User Dashboard 57](#_Toc92473024)

[5.5. Limitations during implementation 63](#_Toc92473025)

[6. Conclusions and Further Work 65](#_Toc92473026)

[References 68](#_Toc92473027)

[Appendices 77](#_Toc92473028)

[Source codes 77](#_Toc92473029)

[Calculated Costs 126](#_Toc92473030)

[AI Model Results 127](#_Toc92473031)

# Acronyms and abbreviations

|  |  |
| --- | --- |
| Acronym | Description |
| EU | European Union |
| US | United States of America |
| USD | US Dollars |
| EUR | Euros |
| MXN | Mexican Pesos |
| GCP | Google Cloud Platform |
| AWS | Amazon Web Services |
| PaaS | Platform as a Service |
| SLA | Service-level agreement |
| BI | Business Intelligence |
| ML | Machine Learning |
| NPL | Natural Language Processing |
| IoT | Internet of Things |
| AI | Artificial Intelligence |
| ETL | Extract Transform Load |
| API | Application Programming Interface |
| UAV | Unmanned Aerial Vehicles |
| AE | Acoustic Emission |
| VR | Virtual Reality |
| NDT | Non-destructive techniques, synonym to NDE or Non-destructive evaluation |
| DE | Destructive evaluation |
| IR | Infrared spectrum |

# Table of Figures

[Figure 1: Ultrasonic pulse velocity test, an NDT sonic-based device to measure wave propagation in concrete structures (Adapted from J. Helal et. al) [Image] 14](#_Toc92473032)

[Figure 2: NDT Techniques used by bridge inspection contractors surveyed respondents (from D. D. Rolander, G. M. Phares, B. A. Graybeal, M. E. Moore, and G. A. Washer) [Image] 15](#_Toc92473033)

[Figure 3: System design for industrial monitoring integrating UAV (by Marouane Salhaoui et al) [Image] 16](#_Toc92473034)

[Figure 4: Diagram representing a 5-layer IoT architecture design (adapted from Al-Qaseemi) [Diagram] 19](#_Toc92473035)

[Figure 5: Diagram representing a 5-layer IoT architecture design (adapted from Mohammad et. al.) [Diagram] 19](#_Toc92473036)

[Figure 6: TOM sample configuration (by Pahl, C., Ioini, N., and Lee, B.) [Image] 20](#_Toc92473037)

[Figure 7: Edge cloud architecture design (by Claus Pahl et al.) [Image] 21](#_Toc92473038)

[Figure 8:Average power consumption comparison between low-energy wireless network on low-traffic and high-traffic conditions (by Morin, E. et al) [Image] 31](#_Toc92473039)

[Figure 9: 8-layer design for the cloud system prototype development [Diagram] 37](#_Toc92473040)

[Figure 10: Visual representation of the edge device attached with a camera and wireless sensor [Image] 39](#_Toc92473041)

[Figure 11: Edge layer and Middleware layer components diagrams [Diagram] 40](#_Toc92473042)

[Figure 12: Fog Layer diagram representing the interaction between lower layers in the fog and connectivity to the cloud infrastructure layer [Diagram] 40](#_Toc92473043)

[Figure 13: Cloud infrastructure layer representation, showing the layer’s inner services interaction and its contextual relation to the other layers [Diagram] 41](#_Toc92473044)

[Figure 14: ETL Layer contextualization and description [Diagram] 42](#_Toc92473045)

[Figure 15: Data Stream flow describing the process for the classification ETL process [Diagram] 42](#_Toc92473046)

[Figure 16: Diagram representing the model training/testing process flow [Diagram] 43](#_Toc92473047)

[Figure 17: Application Layer contextualization [Diagram] 44](#_Toc92473048)

[Figure 18: Classification verification process flow diagram [Diagram] 46](#_Toc92473049)

[Figure 19: Implemented Node-Red flow displaying the snapshot flow [Image] 48](#_Toc92473050)

[Figure 20: Implemented Node-Red flow displaying the intervalometer flow [Image] 48](#_Toc92473051)

[Figure 21: Implemented Node-Red flow displaying the command flow [Image] 49](#_Toc92473052)

[Figure 22:Cracked concrete images dataset sample (By ) [Images] 51](#_Toc92473053)

[Figure 23:Normal (no-crack) concrete images dataset sample, taken with a smartphone device [Images] 51](#_Toc92473054)

[Figure 24:Vertex dataset preview 52](#_Toc92473055)

[Figure 25: Dataset creation process in Vertex AutoML [Image] 53](#_Toc92473056)

[Figure 26:AutoML model creation process in Vertex AutoML [Image] 54](#_Toc92473057)

[Figure 27:Confidence threshold of the model based on Precision-recall curve and Precision-recall by threshold values of the model [Image] 55](#_Toc92473058)

[Figure 28:Classification integrated gradients results of crack identified images [Image] 56](#_Toc92473059)

[Figure 29:Classification integrated gradients results of no-crack identified images [Image] 57](#_Toc92473060)

[Figure 30: Cloud App data flow representation [Diagram] 59](#_Toc92473061)

[Figure 31: Analytics Dashboard View displaying general summary and statistics [Image] 60](#_Toc92473062)

[Figure 32: Analytics Dashboard View displaying data by registry and device [Image] 61](#_Toc92473063)

[Figure 33: Registries and Devices View displaying available registries' ID's and the available devices [Image] 61](#_Toc92473064)

[Figure 34: Device Dashboard main view, including device's logs, dataset view, and command centre [Image] 62](#_Toc92473065)

[Figure 35: Device Dashboard Recent Images preview displaying fissure and non-fissure events and image highlights [Image] 62](#_Toc92473066)

[Figure 36: Image Viewer Dashboard preview, displaying images from a date range query. The result tags images by fissure and non-fissure events and image highlights [Image] 63](#_Toc92473067)

# Table of Tables

[Table 1: AWS, Azure, and GCP PAAS platforms technologies and services comparison chart 23](#_Toc92473068)

[Table 2: DBMSs support provided as supported by AWS, Azure, and GCP cloud platforms 24](#_Toc92473069)

[Table 3: AWS, Azure, and GCP IoT platforms integrated technologies and services comparison chart 25](#_Toc92473070)

[Table 4: AWS, Azure, and GCP platform Data Pipeline Services comparison chart 27](#_Toc92473071)

[Table 5: Comparison chart of AI Services offered by AWS, Azure, and GCP Cloud Platforms 27](#_Toc92473072)

[Table 6:Cost comparison between SageMaker, AI, and Vertex AI machine learning platforms. Prices are per hour of use. Any machine above 4 vCPUs and 16 GiB of RAM is considered a large machine. 28](#_Toc92473073)

[Table 7:IoT Sandbox Platforms Comparison 29](#_Toc92473074)

[Table 8: BLE, ZigBee, and LoRaWAN protocols specifications comparison chart 30](#_Toc92473075)

[Table 9: Cloud provider’s specification requirements for the end prototype 33](#_Toc92473076)

[Table 10: Technical Evaluation Matrix of Cloud Providers 33](#_Toc92473077)

[Table 11: Cloud Platform Providers' Costs Breakdown 34](#_Toc92473078)

[Table 12: IoT simulation sandbox evaluation matrix 35](#_Toc92473079)

[Table 13: AI products evaluation matrix 36](#_Toc92473080)

[Table 14: Node-RED components used in the final prototype and the use-case relative to the system 47](#_Toc92473081)

[Table 15: Cloud Functions scenarios descriptions and requirements 50](#_Toc92473082)

[Table 16: Cloud App Engine Components 58](#_Toc92473083)

[Table 17: HTTP Request Paths in the system 60](#_Toc92473084)

[Table 18: Implementation Limitations and Problems Matrix 65](#_Toc92473085)

[Table 19: Python Labeller Program 77](#_Toc92473086)

[Table 20: Python Code to retrieve Vertex AutoML evaluations raw data 78](#_Toc92473087)

[Table 21: Image Classification Cloud Function Source Code and Requirements 79](#_Toc92473088)

[Table 22: IoT Pub/Sub Pre-process Cloud Function Source Code and Requirements (registry 1) 80](#_Toc92473089)

[Table 23: IoT Pub/Sub Pre-process Cloud Function Source Code and Requirements (registry 2) 81](#_Toc92473090)

[Table 24: Main Application App Engine Program Source Code and Requirements 85](#_Toc92473091)

[Table 25: IoT Controller App Engine Program Source Code and Requirements 88](#_Toc92473092)

[Table 26: Data Controller App Engine Program Source Code and Requirements 90](#_Toc92473093)

[Table 27: Storage Controller App Engine Program Source Code and Requirements 91](#_Toc92473094)

[Table 28: Analytics App Engine HTML Template Source Code 94](#_Toc92473095)

[Table 29: Devices View App Engine HTML Template Source Code 100](#_Toc92473096)

[Table 30: Images View App Engine HTML Template Source Code 107](#_Toc92473097)

[Table 31: Registries and Devices View App Engine HTML Template Source Code 109](#_Toc92473098)

[Table 32: Image rendering function App Engine HTML Template Source Code 109](#_Toc92473099)

[Table 33: Node-RED Flow Code 126](#_Toc92473100)

[Table 34: AWS Calculator Cost Estimate (in USD). Available services in the AWS calculator are included. Created and downloaded using the GCP calculator utility made available online. 126](#_Toc92473101)

[Table 35: Azure Calculator Cost Estimate (in USD). Available services in the Azure calculator are included. Created and downloaded using the GCP calculator utility made available online. 127](#_Toc92473102)

[Table 36: GCP Calculator Cost Estimate (in USD). Available services in the GCP calculator are included. Created and downloaded using the GCP calculator utility made available online. 127](#_Toc92473103)

[Table 37: Vertex Auto AI Evaluation Detailed Results 144](#_Toc92473104)

# Introduction and system definition

## Background

On September 19th of 2017, an earthquake of magnitude 7.1 hit the central region of Mexico, causing infrastructure damage, from superficial/cosmetic damage to some buildings and total structural collapse of 44 buildings [1]. As noted by Quinde, Pablo et al [2], the cumulative damage of these events may have played some role in the fatigue and eventual collapse of some structures, and conclude that by improving and updating standards, structural resilience can increase, especially considering the old structures in the area. It is estimated that an approximated MXN$13.6 million pesos were required for repairments after this event [3].

In the European Union, an annual budget of £215 M is allocated for bridge infrastructure maintenance [4]. In the United States, ~27.1% of the existing bridges are labelled as deficient, and which replacement costs are estimated to be of USD$9.4 billion [5]. According to the OECD, in 2019, the top 3 countries that invested the most in infrastructure maintenance [6] spent an approximated amount of €72 billion in infrastructure maintenance.

As defined by the American Society for Non-destructive Testing [6], non-destructive tests or evaluations (NDT or NDE) are techniques and methods used in the construction industry to inspect, test, or evaluate material, components without destroying the serviceability of the part or system. Among NDT techniques, visual testing is a common test method used in the industry, where an inspector looks at the surface to identify the health of the structure, and identify for surface fissures, defects, corrosion signs, erosion, or any sign of defects in the surface. Visual techniques can be enhanced with optical instruments and computer assisted tools [7].

## Problem statement and scope

NDE or NDT methods are techniques that allows inspection of buildings and infrastructures without affecting, modifying, or destroying a part or whole components of a structure [7]. Among NDE techniques, visual inspection of structures is a popular method [10] used in the industry, where an inspector or consultant does visual inspections, directly or with the aid of image devices such as cameras, to determine the condition of a structure. Depending on the area and frequency, implementations of this nature can be expensive or of high risk to inspectors [11].

The dissertation goal is to propose a distributed architectural design and demonstrate, through a practical application of an NDE system, the incorporation of IoT and ML through a Cloud platform. The aim of the system will be providing, through integrated services, visibility, and feedback of structure surface integrity by analysing and classifying structural images to help structure inspectors and building managers to make informed decisions based on the information provided by the system.

The architectural design choices must be sustained and argued over the course of this dissertation. Subsequently, the end-design will also be presented by clearly stating the abstractions of each component of the system and the system end-to-end interaction.

The end-prototype must integrate the following components as part of the demonstration:

* IoT device and connectivity between the edge systems and a cloud platform through a network connection
* An ETL data pipeline to transform datasets from the edge devices
* Machine Learning integration through the implementation a classification model
* Data storage, including objects storage and a DBMS integration
* Interactive dashboards that allow user to interact with the edge devices and visualize data from the edge devices

Specifically, the system prototype must be able to demonstrate a practical application of an IoT platform capable of demonstrating a use-case for structural integrity inspections for concrete structures through the integration of IoT edge devices and machine learning. This prototype should be able to demonstrate a viable, cost-effective, and simple system that can facilitate periodic inspections in the construction industry.

The scope of the prototype of the system should incorporate the following requirements:

* **IoT Device simulation system**
  + **Data transmission:** the prototype system must integrate a simulated IoT device or devices that can demonstrate periodic transmission of data from the edge into a cloud platform through an MQTT broker.
  + **Data type:** the goal of the system is to classify images based on cracks or fissures present in the system. Thus, the data type the system must produce and transmit are image objects.
  + **Data local processing:** captured images must be pre-process images to meet a cloud system criterion.
  + **Data reception:** the prototype must be able to receive commands from the cloud platform, interpret the commands, and execute an action based on the command. The list of commands must be established in the design section in chapter 4.
* **Cloud Platform**
  + **Edge device management:** the cloud platform must be capable of managing and registering edge devices, as well as implementing security protocols to ensure data encryption between the cloud platform and the edge devices.
  + **Data Storage:** the cloud platform must be capable of storing the data transmitted by the edge devices, as well as storing any data generated by the system platform, such as classification tags and system logs. The data can be stored in different formats and sources within the cloud platform.
  + **ETL Data Pipeline:** the ETL data pipeline must be responsible of processing the data generated from the edge and cloud systems, transforming the data in an adequate format specified by the requirements of the machine learning model, and storing the resulting dataset into the storage platforms of the cloud system. The ETL pipeline main objective is to clean and prepare the data for any use within the system.
  + **Machine Learning Model:** the machine learning main task will be classifying the datasets produced by the edge devices. Thus, the model that will be used is a classification model based on image recognition. Dataset availability, nature, conditions, and uses are described in this chapter.
  + **User Dashboard:** the user dashboard must be capable of presenting data to the users and enabling edge command transmission for device control by the users from the cloud system. Additionally, the dashboard must provide log data and metrics from the edge devices through an interactive UI system.

The dataset required for the demonstration should include a variety of images that can allow classification of crack or no-crack in a structure. The images conditions and qualities shouldn’t be uniform to simulate real-life conditions, assuming diverse variables can affect the final image that will be transmitted to the cloud system.

For model training, testing and validation, the dataset that will be used for the model from DeepCrack by Liu, Y et al. [12] combined with images taken with a mobile device. The DeepCrack dataset contains a dataset of 527 images sized at 544px width by 384px of fissured surfaces, with a variety of image quality such as angle of picture, focus, illumination, image noise, among other variables. However, there is a lack of images of surfaces without fissures present; to compensate and to train the classification model to identify crack and no-crack surfaces, a set of 236 images with a size of 544px width by 384px high were taken using a mobile device, aiming to recreate similar conditions of image quality by introducing different lighting conditions, focus, image noise, angle of pictures, and adding patterns that can be mistaken as fissures such as gaps between surfaces and vegetation.

Lastly, the devices logs and data, image datasets, classification results, and any other additional data available in the system must be presented to end-users through a user dashboard. The dashboard must be available accessible through a web interface. Additionally, a functionality that grants a level of remote control of the IoT devices must be enabled for the end-users through the dashboard.

## Structure of the Dissertation

The dissertation is structured in 6 chapters.

Chapter 2 covers literature review describing traditional structural evaluation methods and explores emerging techniques in the construction industry, which leads to a description of existing machine learning models for crack detection, discussing different NDE classification models. Lastly, it compares and discusses diverse architecture patterns to model IoT end-to-end systems.

Chapter 3 includes the technical, including a comparison and analysis to cloud platform services and IoT simulation sandbox systems. It finalizes with a technical description and comparison of common communication protocols in the IoT industry.

Chapter 4 presents the system architecture design of the prototype, including details on each layer design. It also presents details on the technical requirements of the system, expanding details of the sandbox, the data pipeline process from data ingestion do data storage and use, the machine learning integration, including a discussion on the datasets used for the model training and validation, and finally, the dashboard implementation design.

Chapter 5 discusses the details of the implementation and results of the final prototype, including the system final scope, use, and integration from an end-to-end point of view, model performance and results, as well as limitations encountered during the implementation.

Lastly, chapter 6 includes a conclusion statement of the work, and further work.

# Literature review

The literature review will cover three general areas, structure inspection techniques which includes traditional and emerging inspection techniques in the construction industry, existing machine learning models in concrete structure analysis, and lastly, systems architecture design patterns for IoT systems, exploring different patterns that integrates the different components for IoT systems.

## Structures Inspection Techniques

### Traditional Inspection Techniques

According to an US survey, 55,710 bridges have been declared structurally deficient [13]. There are broadly 4 inspection categories, Safety Inspection, Principal Inspection, Special Inspection, where each category can vary in terms of frequency, duration, and accuracy, but in combination, these inspections are undertaken to ensure timely identification of defects which can lead to accidents or high maintenance costs [13].

There are also many techniques that aids the inspection tasks of infrastructure inspectors to analyse and evaluate structural integrity that vary from simple models and visual inspections to complex systems that incorporates sonar devices that can detect micro fractures and fissures as well as bring light to deep structural integrity that superficial inspections cannot detect. A Normally, the inspections are carried with a technique or set of techniques and are classified as non-destructive (NDT) and destructive tests (DT) [14][15]. As defined by J. Helal et al [14], NDT methods are tests that help determining structural integrity without affecting the usefulness and serviceability of a structure, while DT methods explore failure mechanisms to determine mechanical properties of a material.

Some examples of non-destructive techniques are concrete electrical resistivity, where an electrical current is applied to concrete to measure the resistivity of the surface and assess them [16], acoustic sensing, used in oil and gas refineries to inspect for leaks and prevent explosions and toxic gas leaks [17], acoustic emission testing or impact-echo systems, where an emitter sends a vibration through a structure to light inner structure integrity [18], thermographic methods, where, through thermal imaging, an analysis of the structure can be made based on thermal distribution [19], visual evaluations, where an observer directly inspects the visual condition of a structure to find visible damage, corrosion, or material degradation of an infrastructure [20].

A picture containing device, meter, gauge

Description automatically generated

Figure 1: Ultrasonic pulse velocity test, an NDT sonic-based device to measure wave propagation in concrete structures (Adapted from J. Helal et. al) [Image]

As explained by J. Helal et al [14], an important factor in visual inspections is vertical and lateral resolutions, signal-to-noise ratio, and existing information about the structure. They also mention the importance of advances in sensors and new materials that can lead to new, modern NDT methods, where effective data-acquisition, processing, and interpretation will be important, and leading interest of the industry is devoted to acoustic techniques aided by software data analysis algorithms. Sooyong Park et al. [15] layout some limitations of acoustic techniques where conditions such as moisture and temperature have some effect on the results, which leads to their argument where an NDT technique should be capable of consider changes in the environmental conditions, locating the damage, and be incorporated to the level IV NDT.

E. Sheils et al [21]. introduce a two-stage inspection technique with NDT tools; on the first stage, the evaluation aims to detect a defect in a structure, and, in a subsequent test, a sizing inspection measures the defect size and impact to the structure. This method aims to minimize service life costs and open the field to determine an optimum combination of techniques to detect structure integrity. An insight in the research is that there is a direct frequency-cost implication which, in turn, has a direct impact on failures. The higher the frequency, the higher the cost and the lower the failure rate; the lower the frequency, the lower the cost, and the higher the failure rate. This is explained as the longer an interval between inspections, defects are more likely to fail and less likely to be repaired between inspections. Their proposed method finds that resources can be better allocated by focusing more resources on sizing phase or phase 2, while phase 1 should have a detection-repair threshold set by managers.

The National Bridge Inspection Program is a mandate in the United States to inspect all the highway bridges on public roads. The program was developed to ensure the safety of traveling public and aiming to maintain an inventory of federal-aid highway system bridges. As presented by D. D. Rolander et al in figure 2, visual inspection is the primary tool for inspection [22], where ~30% of State DOT visual inspections and ~66% of County DOT visual inspections were carried. It’s important to mention that, in the context of visual inspections, the FHWA concludes that the assessment can vary and directly related to the training of inspectors, and while computer software tools is important for data accuracy, it is not adequate alone [23][24].

Table

Description automatically generated

Figure 2: NDT Techniques used by bridge inspection contractors surveyed respondents (from D. D. Rolander, G. M. Phares, B. A. Graybeal, M. E. Moore, and G. A. Washer) [Image]

### Emerging and Experimental Inspection Techniques

New inspection tools are being proposed to replace some traditional tools and mitigate risks and costs associated with inspections, such as the Unmanned Aerial Vehicle Bridge Inspection Demonstration Project [11] by Brent M. Phares et al. and with the Department of Transportation of Minnesota, where an UAV inspected a ~24m (80 ft.) bridge and a ~56m (185 ft.) train bridge above St. Croix River to demonstrate both, practicality of inspection tasks, increased safety, and cost reduction associated with inspection tasks. Among the project’s conclusion, it argues a practical effectiveness of the technology by providing high quality details for medium sized concrete bridges with little risk for inspectors and the public, but also rises challenges on small bridge inspections, as well as the limitations on local air space regulations.

In Application for Crack Identification Techniques for an Aging Concrete Bridge Inspection Using an Unmanned Aerial Vehicle [25], In-Ho Kim et al. proposes another UAV system for image acquisition of structures using a high-resolution camera attached to the UAV. An interesting consideration by In-Ho Kim et al. is defining guidelines aiming to prioritize safety of inspections, where in the paper they recommend a flight altitude within 150 m to avoid collisions with the aircraft; an important consideration to maintain consistency with safety in the field, however they also mention that safety measures adds difficulty to image acquisition precision due to the area of focus is affected by the distance between the UAV and the subject, although they also propose some mitigation measures using gimbals to stabilize the image. Additionally, to image acquisition techniques, they expand their application by integrating a deep learning model for image classification, where the model can detect and classify images upon crack detection. There is a lack of clarity on model accuracy, and while it concludes that the model performs above traditional image processing techniques, it is both unclear what traditional model the proposed model was measured against, and what methodology was used for the experiment.

An advantage that UAV have, and possibly why these are emerging components in NDT systems, is that UAV can cost effective devices that can capture high-resolution images, easily integrated to different systems, as well cover large areas as Marouane Salhaoui et al [25] point out in Smart Industrial IoT Monitoring and Control System Based on UAV and Cloud Computing Applied to a Concrete Plant. An important consideration highlighted by Marouane Salhaoui et al is that, due to the diverse components in the fog layer, a processing delay is introduced that punishes latency and affects data transmission from UAV data to the cloud. The implementation in this research paper is relevant in the context of this dissertation even though the final application is specific to industrial concrete factory environment.

Chart, diagram

Description automatically generated

Figure 3: System design for industrial monitoring integrating UAV (by Marouane Salhaoui et al) [Image]

Another system proposed by Muhammad Omer et al in Use of gaming technology to bring bridge inspection to the office [13] demonstrates a VR system to facilitate access and reach for infrastructure inspections through a combination of portable devices capable of supporting VR such as android smart phones, and the use of LIDAR scan systems to inspect and maintain a historic record of a structure at any moment in time. While the proposed technique shows some promise by facilitating reach and data quality, it also comes at high costs on processing requirements, highly specialized equipment, and special training to operate the system.

## Machine Learning in Structure Analysis

Classification and detection of fissures and fractures in structures is a topic of interest in the industry; among some advantages of a fissure image classification system are reduction of costs associated to specialized tools required for inspections, frequency of inspections, and accessibility to the knowledge and tools required to carry inspections.

There are some important challenges for crack detection systems, some of which have some partial solutions that improve accuracy of existing image classification models. Commonly, structural images can carry noise data when used without any filtering. As described by Xiangzeng Liu et al [27], existing algorithms require low-noise images with smooth surfaces, for which they propose an algorithm that combines a multi-scale filter that enhances the images, regardless of illumination conditions, which yielded an improvement of accuracy and performance against existing algorithms (OTSU and morphology-based algorithm and percolation model-based method).

Other important considerations for image classification are image resolution. An assumption is that high-resolution images can yield better results at the cost of performance – the higher the resolution, the higher the computing and memory costs for an algorithm to analyse. Minami Takahiro et al [28] compared results from super high-resolution cameras (defined as capable of capturing photographs of 100 megapixels) against existing datasets of ~65 pixels implies that there are possible accuracy gains for models, though it is unclear what is the performance penalty.

Another consideration for image quality, as pointed by Y. Fujita et al [23], is closely related to the different conditions a photograph is exposed such as noise ISO levels from a sensor sensitivity level to exposure and aperture, to illumination conditions of the object, which can have impact in a model accuracy as shown by their results. By applying a filtering and cleaning technique on a pre-process stage on the images used in their research increased the accuracy of the model significantly.

Considering both research results from Xiangzeng et al and Y. Fujita et al [23] [27] it is possible to draw conclusions regarding the dataset required for the model – high-quality images can improve and aided with a pre-processing technique, yields of a model can increase. It is important to highlight that there can be penalties associated with image resolution, and thus resources must be considered when designing and implementing a model. Additionally, on-site devices can have different limitations and restrictions, and thus, data engineer pre-processing techniques can have a wider application in end-to-end systems.

An interesting technique which, according to I. Garrido et al in Thermographic methodologies used in infrastructure inspection: A Review-Post-processing procedures [20], has become increasingly important is an infrared technique imaging combining ML to classify and detect subsurface anomalies in a structure. Infrared imaging provides an additional dimension of information that traditional images cannot provide through the thermal markers that can highlight defects and subsurface defects. However, as its also pointed out, there are some significant costs associated with specialized cameras, current imaging techniques have low resolution which can prejudice an AI model, climatic conditions necessary to obtain imaging through IR, among others.

While image processing techniques and quality have been previously established by many authors, Wilson Ricardo Leal Da Silva and Diogo Schwerz de Lucena expands on the subject with a CNN model to detect fissures and fractures in infrastructure by segmenting the nature of the cracks, a work that is close to this dissertation goal. An important conclusion they draw is well-reasoned – existing datasets are limited, and existing models are yet to mature. As datasets become available, it is possible to enrich a system to create a well-rounded system. Applying a time-series based model can push forward the field, however as noted by the authors, expertise to classify a supervised model is important to increase data quality.

## IoT System Architecture Patterns

ACM [32] specifies a rigorous method to clearly define *what* is desired or goal of a system, and *how* a system can arrive to the goals at each level of its components. This method is a solution-oriented design that frames a problem to solve, and clearly states how it will deliver. System architectural designs, then, requires decomposition of a problem to analyse and define a clear set of steps to achieve to a solution. This defined process creates an environment where different methods and patterns to solve similar problem blooms.

As explained by Hironori Washizaki et al [31], systems and software design processes have two main phases to uncover the different abstraction levels, architectural design, and detailed design. They argue that the level of abstraction of existing patterns has some shortcomings due to the lack of detail on the IoT abstraction. They present three domains in IoT patterns which can be presented through high, medium, or low abstraction levels:

* Non-IoT patterns or general systems that can be designed to integrate IoT systems.
* General IoT patterns, where components and patterns are applicable to any IoT system or software not specific to a certain technical domain.
* Domain specific IoT patterns, where a pattern addresses a specific problem domain, such as healthcare.

The scope of this dissertation involves an end-to-end ecosystem by integrating different domains, IoT, Cloud, and ML. This requires, then, an architectural design that reflects this integration. Currently, the industry has defined two main designs in IoT-specific architectures called three-layer design [34] and five-layer design [33] [34].

The three-layer design is defined by three main components:

* **Sensing, perception, or physical layer** – This layer encompasses IoT devices that typically embedded with a sensor or mesh of sensors such as cameras, accelerometers, gyroscopes, sonars, thermometers, among many other sensors. The main purpose of this layer is to capture data from the edge, although it can also incorporate additional activities outside data capturing through external commands.
* **Network layer –** The network layer main purpose is connecting the sensing and application layers together. This layer can be represented by any network device and protocol, and the physical connection can be represented by either a wired or wireless connection. Normally, a secure communication protocol is integrated as part of this layer to enable a secure end-to-end connection between the sensing and application layers.
* **Application layer –** the application layer represents the services that consumes and typically transforms and use the information consumed from the sensing layer. The application layer can be subsequently divided in several integrated services that provides decision-making capabilities to the system. This layer can also send commands to the sensing layer and react to events based on the data obtained from the sensing layer.

While the three-layer offers some level of generalization of conventional IoT architectures, it can leave finer details of complex system invisible. Thus, a five-layer design is also normally defined in architectural designs as explained by Mohammad et al [34]. However, as explained by Al-Qaseemi et al. [33], there is a lack of standards in the IoT industry which is seen in five-layer designs; the industry presents two distinct designs with a five-layer topology; a five-level design as defined by Al-Qaseemi et al. that expands upon the three-layer design by incorporating two layers, the gateway layer between the sensing layer and network layer, and the middleware layer, between the network layer and application layer. Mohammad et al. defines a middleware layer, or processing layer, similarly as Al-Qaseemi, but incorporates a business layer on top of the application layer as shown in figure 4; this design maintains the gateway and communication layers under a single layer called transport layer.

Figure 4: Diagram representing a 5-layer IoT architecture design (adapted from Al-Qaseemi) [Diagram]

Figure 5: Diagram representing a 5-layer IoT architecture design (adapted from Mohammad et. al.) [Diagram]

R. Praveen Kumar et al [35] model is based on the 5-layer design and proposes a 7-layer design. This design is formed by 4 main layers containing sub-layers which are as it follows:

* **Fog Layer:** The bottom-most layer which incorporates the sublayers of things and connectivity. It also abstracts the outer edge, which helps to highlight the level of interaction between the two sub-layers.
* **Cloud Layer:** This single layer represents the cloud infrastructure, which connects with the fog layer to upper layers for data processing and presentation.
* **Big Data Layer:** this layer includes two sub-layers, data ingestion and data analysis, to represent ETL pipelines and ML, data analytics, and mining tasks.
* **Business Layer:** the upper-most layer of this design, which includes two sub layers, the application, and people and subprocess layer, which abstracts the applications developed to integrate and present the things sub-layer data, and the transformational decisions obtained from the data.

A different architectural style, proposed by Claus Pahl et al. [36] in an architecture pattern for trusted orchestration in IoT edge clouds, bases its pattern with the principle of edge orchestration supported by a blockchain mechanism to enable a trusted orchestration management, or TOM, which consists in a trustworthiness identity verification layer to handle the edge and fog infrastructures. The principles of TOM are identification, where all *things* must authenticate, *data provenance* or traceability, and non-repudiation of architecture operations, while TOM requirements consider the following includes *things* might dynamically join, *data* is generated, but must be traceable, and *decisions and orchestration* must be agreed by all participants.

An argument they make for this design proposal is that, on principle, data transmission from the edge to the cloud can and should be limited, while encouraging filtering and processing activities over the edge to reduce, and this way achieve an optimization of cloud resources. However, an argument against this principle are resources limitations over fog or edge layers, where data pre-processing can add a burden that can translate to an increase in latency between the edge and cloud systems, which can have a negative impact for time-sensitive systems, as noted by Marouane Salhaoui in a UAV system application [26]. However, a possible benefit to this architecture proposal is TinyML which can open, at some limited capacity, local decision-making processes, and tasks through model implementation in the fog layers.

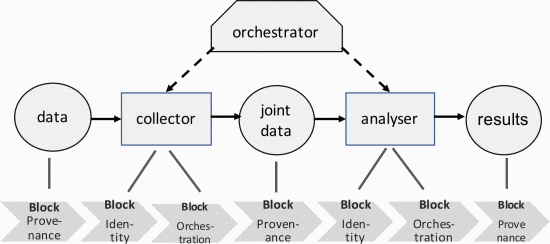


Figure 6: TOM sample configuration (by Pahl, C., Ioini, N., and Lee, B.) [Image]

In A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters, Claus Pahl et al. [35] acknowledges the importance of low-latency and mobility that IoT systems often requires, as well as the limitations in hardware and system resources that lightweight fog and edge systems normally have. In their architecture proposal, the common fog layer can be considered a micro-cloud system by providing similar services to cloud systems in a small scale, thus they argue that the fog layer requires the right abstraction. This micro-cloud, the Edge Cloud layer, is comprised of three sub-layers: the smart things network, the lowest layer of the micro-cloud containing all edge sensors and ad-hoc networks and protocols; the field area network layer or middle layer which includes long-range networks, WAN, or internet connections; at the top layer, the virtual compute and storage cloud, operates the management of the architecture in centralized providers.

What makes this architecture unique is the integration of a container principle, where the edge cloud can allocate containers to orchestrate in the edge by creating a cloud layer and subsequently, reducing implementation costs with this approach. However, there are restrictions such as special devices that can host the containers and, from an architectural standpoint, it lacks abstractions of upper layers as seen in other IoT architecture designs; also, this abstraction could be considered as part of middleware abstractions in other patterns.

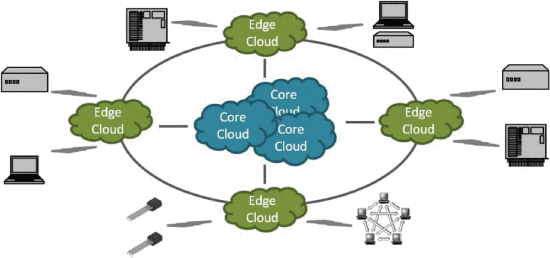


Figure 7: Edge cloud architecture design (by Claus Pahl et al.) [Image]

# Technology review

The technology review contained in this dissertation covers a Cloud Platforms which contrasts cloud platforms providers and the services offered, an analysis of IoT simulation platforms, and lastly, low energy communication protocols.

## Cloud Platforms

Aleem and Ryan Sprott [38] argue that cloud computing and cloud are interchangeable concepts and adapts the National Institute of Science and Technology definition, which is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (Ryan and Aleem Sprot, 2012). Cloud Computing, as defined by Luis M. Vaquero et al [39], is a paradigm that shifts the location of computing infrastructure to the network, aiming to reduce costs associated with hardware and software resources management and maintenance. They also define the terminology cloud as a large pool of easily usable and accessible virtualized resources that can be reconfigured to adjust to a variable load (Luis. M. Vaquero et al, 2008, p 51). Another definition for Cloud Platform is an entity that manages a shared pool of manufacturing resources and capabilities over a network (Ren L. et al, 2013 [40]). For this chapter, we will then define Cloud Platforms as a ubiquitous grid of elastic, scaling, and configurable computing resources, tools, and services that are available shared and on-demand through an external network.

There are many commercial cloud platform services which integrates off-the-shelf solutions to build end-to-end systems by combining an array of technologies, which offerings can vary and compete on costs, offered services and toolsets, SLA, end-to-end integration, among other areas which can influence architectural and development decisions. As of the first quarter of 2021 [41], the most popular companies with cloud platform offerings in the market are Amazon with Amazon Web Services (AWS) with a market share of 32%, Microsoft Azure service with a market share of 20%, and Google with Google Cloud Platform (GCP) with a market share of 9%. In this chapter we will review each system integrated technology through the contextual scope of this dissertation, thus the chapter will be divided in the following sub-sections:

* PaaS
* Database integration
* IoT service
* ETL Pipeline System and Orchestration
* ML/AI platform

### PaaS

PaaS (Platform as a Service), as defined by IBM [42], is a cloud computing model that provides a fully integrated and managed hosting service that developers can use to create, deploy, maintain, and scale applications and systems on-demand. From all three platform, the PaaS offerings are GCP App Engine [43], AWS elastic Beanstalk [44], and Azure App Service [45]. From a general perspective, these platforms have similar functionalities as seen in *table 1*.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS Elastic Beanstalk [44] | Azure App Service [45] | GCP App Engine [43] |
| **Supported Programming Languages** | Java  .NET  PHP  Node.js  Python  Ruby  Go | .NET  .NET Core  Node.js  Java  Python  PHP | C#  Java  Ruby  Node.js  Python  Go  PHP |
| **Third party programming**  **language extension** | Yes | Yes | Yes |
| **Standard Database Integration** | Amazon RDS  Amazon DynamoDB  Microsoft SQL Server  Oracle | Azure SQLDB  MongoDB  MySQL | Google BigQuery  Cloud Firestore  Cloud SQL for MSQL and PostgreSQL |
| **API integration** | SDK API through tools for AWS | RESTful API through CORS | RESTful API and client APIs integration through Google Cloud APIs |
| **Security integration** | AWS IAM credentials  User-specific secret key IDs with key rotation  Firewalls | Azure WebApp Firewalls  Active Directory integration | Access control with GCP IAM integration  User and service accounts with secret key and OAuth 2.0 integration  Firewalls  Domain-wide delegation of authority with Google Workspace [46] |
| **Included services** |  | Streamlining  Visual Studio, GitHub, Bit Bucker, and Docker Hub integrations  Deployment slots  App Service Diagnostics |  |
| **Container deployment support** | Yes | Yes (custom window containers) | Yes |
| **SLA** |  | 99.95% uptime | At least 99.95% uptime, and a crediting system for uptimes of less of 99.95% per month |
| **Free tier** | 750 hours of t2.micro instances through EC2 pricing model | 10 apps and up to 1GB of storage  Shared computing [47] | Up 5 GB cloud storage, up to 1GB code storage, and up to 28 free hours for instance uptime |

Table 1: AWS, Azure, and GCP PAAS platforms technologies and services comparison chart

A key difference between these services resides in default programming languages; however, the three platforms allow extensions to third party languages, which adds flexibility to the PaaS services for a multitude of development requirements. Contextually, the design of a system and chosen programming language are the drivers for decisions when selecting a specific platform.

On a security level, Azure provides an additional function that the other platforms do not offer, active directory, and all three platforms include firewall and IAM integrations.

### Database Integration

There are different definitions of databases, and, for this section, we will use the term defined by Britannica [48] as any collection of data or information that is specially organized for search and retrieval by a computer, and are structured to facilitate storage, retrieval, modification, and deletion of data. The management systems that power the databases to manage and extract data are database management systems or DBMS.

Data stored in DBMS can be structured in many ways, however, the two most popular storage mechanisms are:

* **Relational databases**: these are structures in a tabular fashion [49], represented by rows and columns, where each record can store a unique ID called a key. Some popular relational DBMS examples are MySQL, PostgreSQL, and Microsoft SQL Server.
* **NoSQL databases**: non-relational structures [50], where data does not require a schema, allowing for rapid scalability, high distribution, and large amounts unstructured data storage. Some popular examples of NoSQL DBMS are MongoDB, Apache Cassandra, Azure Cosmos DB, and Neo4J.

The three platforms offer different DBMS solutions integrated in their systems for both DB structures, offering both, in-house and allow third party DBMS integration. From a general standpoint, all three have similar capabilities and offerings as shown in *table 2*.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS [51][54] | Azure [52][55] | GCP [53][56][57] |
| **Relational DBMS support** | Amazon Aurora  Amazon RDS (PostgreSQL, SQL Server, MySQL)  Amazon Redshift | Azure SQL Database  PostgreSQL  SQL Server  MySQL  MariaDB | PostgreSQL  SQL Server  MySQL |
| **NoSQL DBMS support** | Amazon DocumentDB  Amazon Dynamo  Amazon Cassandra | Azure Cosmos DB | Cloud BigTable  BigQuery  Firestore  MongoDB Atlas |
| **Graph DBMS support** | Amazon Neptune | Azure Cosmos DB | Neo4j |
| **Cache DBMS** | Amazon ElastiCache | Azure Cache for Redis | Memorystore |
| **Additional Services** | Amazon Timestream  Amazon QLDB  Database Migration Service | Database Migration Service | Database Migration Service |
| **Free tier available services** | Amazon RDS  Amazon DynamoDB  Amazon Redshift  Amazon ElastiCache | Azure SQL DB  Azure Cosmos DB  Azure PostgreSQL  Azure MySQL | BigQuery  Firestore |

Table 2: DBMSs support provided as supported by AWS, Azure, and GCP cloud platforms

### IoT Services

Oracle [58] defines IoT as a network of physical objects or things embedded with sensors, software, and other technologies, and expands on IoT prebuilt applications as software-as-a-service (SaaS) applications, which includes services such as data real-time presentation, machine learning algorithms integration, among others. This definition will be used as an IoT Service for this chapter.

Three different providers IoT service offerings are diverse, and, as seen in table 3, while AWS [59], Azure [60], and GCP [61] share similar properties, the end IoT services and applications are varied among the three platforms. Among shared similarities, the three platforms offer device management platforms, secure communication channels, and remote control on edge devices; however, in closer contrast, the application of these premises have many differences among the three platforms.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS IoT Core | Azure Internet of Things | GCP IoT Core |
| **Message exchange platform** | HTTP  MQTT  LoRaWAN | HTTP  MQTT  AMQP | HTTP  MQTT |
| **Backend services** | LoRaWAN network gateway  Integrated device logs  Amazon Sidewalk Integration  Device Shadow | Firmware management  Remote device controls [62] | Integrated device logs  Dashboard monitors  Metric thresholds  Remote device controls |
| **Security integration** | SigV4 certificate-based authentication.  CA Certificate  Token access authentication  IoT policies  AWS credentials integration | Active Directory authentication  CA Certificates  Shared access signatures  Symmetric device keys  Token access authentication [63][64] | TLS 1.2  CA certificates |
| **Free tier** | Up to 2,250,000 minutes of connection, 500,000 messages sized at 5kb or less, and 225,000 device registrations [65] | Up to 8,000 messages per day per IoT unit sized at 0.5kb [66] | Up to 250mb of data transmission free |

Table 3: AWS, Azure, and GCP IoT platforms integrated technologies and services comparison chart

Highlighting some hey differences between each service:

* AWS offers [67] an integration with LoRaWAN through IoT core and integrates a bi-directional secure service to enable a high-available, low-power communication channel.
* AWS and Azure expands over HTTP and MQTT protocols by offering AMQP, which is an open standard communication ecosystem that offers a richer set of scenarios and superset messages [68][69].
* GCP and Azure simplifies the security management by combining CA certificates and device keys for device and gateway management. Azure expands through by also including token authentication by device, shared access signatures, and symmetric device keys.

### ETL Pipeline System and Orchestration

Extract Transform and Load or ETL Pipelines [70] are a set of processes to move data from multiple sources into a database or storage system. Among some benefits of ETL pipelines is preparing the data for analysis and consumption by BI, AI, or any reporting application such as CRMs. Orchestration, as defined by Red Hat [71], is the automated configuration, management, and coordination of computer systems, applications, and services, which can help IT teams to manage complex servers and applications by allowing software systems to perform tasks automatically as well as streamline periodic concurrent workflows.

The three platforms have built-in orchestrators integrated in the ETL pipelines as part of the services, which helps developers to create complex and automatic end-to-end systems that, through integration with other in-house services, can enable seamless integrated systems.

AWS ETL pipeline service, called AWS Glue ETL [72], is a serverless environment to create, run, and monitor ETL jobs, and which can consume data from diverse AWS storage systems, including AWS DBMS and object storage. Azure data pipeline service, called Azure Data Factory [73], is a serverless data integration system that, according to Microsoft, can construct ETL processes code-free. GCP offers a streaming and batch data processing service through Dataflow [74], powered by Apache beam, which is a fully managed service that automates and auto scales depending on the needs and requirements of the system implementation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS Glue ETL [75] | Azure Data Factory [76] | GCP Dataflow |
| **Native Data Source Support** | Amazon Aurora  Amazon RDS  Amazon Redshift  DynamoDB  Amazon S3 | More than 90 built-in connectors, including:  Azure data services  Salesforce  AWS Redshift  Google BigQuery  Oracle Exadata  DB2  SAP Warehouse | Cloud Storage  BigQuery  BigTable  Google Sheets |
| **Programming language for ETL** | Scala  Python | .NET  Python  REST | Java  Python |
| **Data processing backend platform** | Apache Spark | Apache Spark | Dataflow  Apache Beam |
| **Streaming integration** | Amazon Kinesis  Apache Kafka  Amazon MSK | Azure Stream Analytics | Apache Beam  Dataflow Streaming |
| **Free tier** | 40 free interactive sessions included | Not included in free tier | Not included in free tier |

Table 4: AWS, Azure, and GCP platform Data Pipeline Services comparison chart

Some differences by the three platforms are:

* AWS and GCP offer a low-volume, free-tier data services for ETL, AWS Lambda [77] and GCP Cloud Functions [78], which work in a similar fashion to Glue ETL and Dataflow but aimed to consume less resources with serverless, autoscaling platforms. Both offer support to the same programming languages, while GCP allows triggers through other services events like HTTP and Pub-Sub, which Lambda does not offer.
* GCP and Azure does not include Dataflow and Data Factory respectively as part of their free tiers, while AWS includes a limited use of Glue under the free tier.

### AI platform

AI is a growing branch of systems that offers many benefits in automating tasks such as identifying, classifying, or in general, analyse data from different sources.

All the major cloud platforms are offering AI tools that, combined with the data pipeline services integrated in their systems, adds an end-to-end synergy from a data analytics lifecycle point of view, from data exploration to ETL process, to model training, to deployment of a trained model, all through a single platform. While the capabilities and tools integrated to the ETL pipeline across the three platforms offer different advantages as described in the previous chapter, the AI services have similar capabilities from all three platforms, AWS SageMaker, Azure AI, and GCP Vertex AI.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS SageMaker [79] | Azure AI [80] | GCP Vertex AI [81] |
| **Off-the-shelf pre-trained models** | NPL  Vision  Data Classification  Linear Regression  Forecasting | NPL  Vision  Data Classification  Linear Regression  Forecasting | NPL  Vision  Data Classification  Linear Regression  Forecasting |
| **Hyperparameter Optimization** | Yes | Yes | Yes |
| **Model training** | Yes | Yes | Yes |
| **Notebook integration** | Jupyter Notebook  SageMaker Studio Notebooks | Jupyter Notebook | Jupyter Labs |
| **Mainstream data analytics libraries and frameworks** | TensorFlow  Apache MXNet  PyTorch  SciKit-learn  Keras | TensorFlow  Apache Spark  PyTorch  SciKit-learn  Keras | Tensorflow  Apache Spark\*  PyTorch  Scikit-learn  Keras |

Table 5: Comparison chart of AI Services offered by AWS, Azure, and GCP Cloud Platforms

As shown in table 5, the advantage that the three platforms offer are the pre-trained models, which can be used in integration with systems of general purpose and have an end-to-end system with an AI integration faster than building and training models from the ground up.

The libraries that all three platforms offer are suited for all the tasks in the context of this implementation. Additionally, experimentation tasks can be eased through the notebook integration that all three offers.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS SageMaker [82] | Azure AI [83] | GCP Vertex AI [84] |
| **Free Tier** | 250 hours of notebooks instances  25 hours of data wrangler (ETL interface) [85]  10M write units, 10M read units, and up to 25 GB of data lake storage.  50 hours for model training  125 hours for inference training | 1 machine learning studio workspace which includes 1 hour per experiment, single node execution [86].  Up to 10 GB for dataset space [87] | Vertex ML Metadata (model studio and model tuning) [88]  Vertex TensorBoard (ML experimentation visualization) [89]  1000 units per month [90] |
| **Standard machine costs** | USD$0.05 up to USD$4.896 | USD$0.073 up to USD$26.688 | USD$0.1900 up to USD$8.7970 |
| **GPU-accelerated machine costs** | USD$0.7364 up to USD$28.152 | USD$0.396 up to USD$1.742 | USD$0.35 up to USD$3.10 |

Table 6:Cost comparison between SageMaker, AI, and Vertex AI machine learning platforms. Prices are per hour of use. Any machine above 4 vCPUs and 16 GiB of RAM is considered a large machine.

Standard machines can serve training purposes, however, GPU-powered machines can offer high yields in performance from an architectural standpoint of GPUs for model training, where GPU’s have a higher number of computing cores, and better parallel performance overall against traditional CPU’s. As shown in table 6, from a cost perspective, while prices depend on the machine type and hours of use on all three platforms, AWS has the highest price on the largest GPU-accelerated machine, while the lowest price is from GCP with small machines.

## IoT Simulation Sandbox

IoT simulation sandbox systems or testbeds are environments and tools that helps the development and timely decisions over the design and requirements for complex IoT environments. As Maxim Chernyshev et al [91] stated, the research in IoT should give attention and include all components and elements that make up the IoT ecosystem (Chernyshev, Maxim, et al 2018, p 1), which can be greatly benefited by testbeds that can scope the requirements for an end-system. In their research, they qualify diverse IoT simulators and classify sandboxes in the following categories:

* **Full Stack:** systems that describe IoT applications and simulate networks.
* **Big Data Processing Simulators:** systems focused on cloud performance and data processing
* **Network Simulators:** systems focused to research network systems, layout, and topology of IoT systems and, according to Maxim and Chernyshev et al, the most common type of IoT simulators.

The technical research in this section focuses on the second class defined by Maxim Chernyshev et al, the Big Data processing simulators and covers three open sourced IoT simulation sandbox, Node-RED, an open-sourced browser-based flow programming language built on Node.JS [92], NoFloJS, a flow-based programming system similar to Node-RED that aids to separate flows and activities of an application through their sandbox [93], and ThingsBoard, an IoT platform for data collection, processing and visualization as well as a device management system [94].

|  |  |  |  |
| --- | --- | --- | --- |
|  | Node-RED [92] | NoFloJS [93] | ThingsBoard Community Edition [94] |
| **Platform** | Node.js | Node.js | Java |
| **Sandbox Style** | Flow based | Flow based | Event based |
| **Base programming language** | JavaScript (flow-based programming) | JavaScript (flow-based programming) | Java with microservices integration and Angular frontend. |
| **Default communication protocols supported** | MQTT  HTTP | MQTT  HTTP | MQTT  HTTP  CoAP  SNMP  LwM2M |
| **Community Support** | Yes | Yes | Yes |
| **Plug-ins** | Yes – palette | Yes - packages | Yes - widgets |
| **Docker container compatibility** | Yes | Yes | Yes |
| **Cloud Platform Integration** | Yes – through Node-RED plugins |  |  |
| **3rd party integrations** | Yes, through Node.JS, JavaScript, and plug-ins | Yes, through Node.JS, JavaScript, and plug-ins | Yes, through microservices and widgets |

Table 7:IoT Sandbox Platforms Comparison

Some additional characteristics of each sandbox platform are the following:

* Node-RED incorporates an ability to share flows which are stored in a JSON format to increase portability of a development. This characteristic is not present under NoFloJS and ThingsBoard.
* NoFloJS and Node-RED are highly flexible and scalable flexibility by incorporating direct JavaScript and Node.JS support for 3rd party tools for the solutions.
* ThingsBoard is a complete ecosystem that also incorporates a seamless device management and registration services and cloud services.
* ThingsBoard is oriented towards providing full end-to-end services, from a sandbox system to cloud and device management solution.
* Node-RED and NoFloJS share many core aspects, contributing toward a wider and effort-less support on cloud platforms that supports Node.JS.
* Node-RED community is the largest and broadest of the three platforms, reflecting a bigger repository of community plug ins.

## Low Energy Communication Protocols

Wireless networks provide advantages to systems where wired infrastructures may be restricted or inexistent and for systems where high mobility is priority. Among the different wireless networks in the telecommunications industry, low-energy wireless networks are designed to optimize energy consumption of devices and are of particular interest for devices for systems that operate with limited energy supplies, such as batteries.

There are three main Wireless Network categories [95], which are classified based on the range and coverage of the protocols:

* **Wireless Personal Area Networks (WPAN)**: These networks are limited in power consumption and range, which are commonly used in IoT devices due to their seamless connectivity, low power, low bandwidth, low computation power capabilities [96]. Two popular protocols in this category are ZigBee and Bluetooth Low Energy (BLE).
* **Wireless Land Area Networks (WLAN)**: with a greater coverage, WLAN protocols are power-efficient wireless networks where various applications can communicate between each other and to other systems connected to the network and, in some scenarios, provide connection with the internet. In IoT context, the IEEEE in the 802.11ah standard defines a low-power and long-range operations that must be maintained by using a set wake-up operations to optimize power usage [100].
* **Wide Area Networks (WAN):** the greatest coverage networks that can span over kilometres. In IoT context, among some used technologies are Low Powered WAN (LP-WAN) which make use of existing cellular low-bandwidth technologies, for example GSM, radio services, or satellite connections [106]. Two popular platforms in LP-WAN are LoRaWAN and Sigfox. In the context of the scope of this dissertation, WAN technologies are not relevant for further discussion.

|  |  |  |  |
| --- | --- | --- | --- |
|  | BLE [97] | ZigBee [98] | LoRaWAN [107] |
| Supported Topologies | Point-to-point  Piconet  Broadcast  Star  Mesh | Star  Mesh | Star-of-stars |
| Range | Up to 400 meters [102] | Up to 300 meters in line of sight, up to 100 meters indoor | Line of sight up to 15 kilometres |
| Radio Spectrum | 2.4 GHz | 2.4 GHz | Spread Spectrum |
| Channels | 40 | 16 |  |
| Data rate | 2 Mb/s (LE 2M)  1 Mb/s (LE 1M) | 250 Kb/s | 0.5 Kb/s to 50 Kb/s |
| Encryption | Optional [99], AES-CMAC and AES-CCM supported | 128-AES supported (at network and application layers) | 128-bit Network Session Key  128-bit Application Session Key |

Table 8: BLE, ZigBee, and LoRaWAN protocols specifications comparison chart

As presented by Morin, E. et. al. [101] In their paper *Comparison of the Device Lifetime in Wireless Networks for the Internet of Things* they analyse and compare energy consumption of low energy wireless networks, short and long, and energy costs associated with packet retransmissions. The paper concludes how particular technologies offer advantages in specific scenarios, and remarks that range is not the only factor, but traffic volume in networks has a direct impact in each system. For systems for ultra-low traffic, Bluetooth Low Energy (BLE) and 802.15.4 performs well, while in larger data systems, 802.11ah has a good performance. BLE is also ranked as adequate in low and medium traffic intensity in this study.

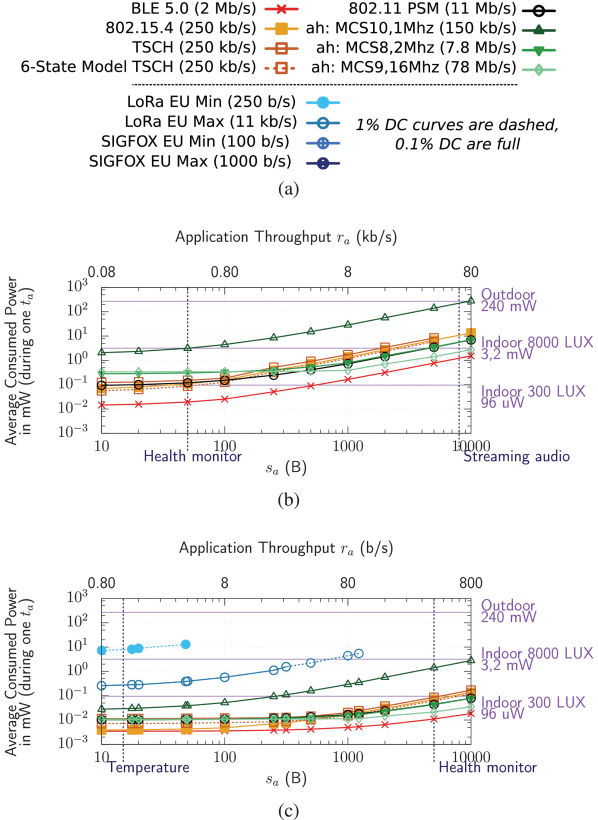


Figure 8:Average power consumption comparison between low-energy wireless network on low-traffic and high-traffic conditions (by Morin, E. et al) [Image]

An important restriction that comes with BLE is its limited range, which is of up to 100m for BLE 4, and up to 400m for BLE 5 [102], however conditions can affect the range in field. An approach to tackle the physical limitations of the network is through a mesh network implementation style [102][103], a topology where a network device can be connected through distributed gateways, which can connect to internet and provide web services through these mesh networks, or where devices can connect with other devices to form a larger network that can broadcast messages across a larger area. There are important considerations to take in high-congested networks that can increase collision and interference between devices as Mohammad, Omar et al [104] concludes, where high-noise conditions can cause system failure and, in a mesh-style network, if a single device acts as a single node bridge between two areas, it be reasonable to conclude that the network can partially collapse.

# Proposed design

This chapter will discuss the technical requirements and architectural design for the prototype aiming to demonstrate an approach to the problem statement of the previous chapter; this chapter includes the specific technical requirements for the prototype and the end-to-end architecture design and its components. Both, the technical requirements, and architecture design will argue on the choices made.

## Technical requirements

This chapter describes and justifies the technical decisions, from the IoT sandbox simulation system to the cloud platform service provider, as well as the components required to build the prototype in the scope defined by both the problem statement and the architectural definition for this dissertation.

### Cloud Platform Provider

As discussed in previous chapters, the cloud platform provider is a key actor to enable storage, data processing, ETL pipeline services, AI integration, and dashboard hosting. In essence, the cloud platform provider will centralize, host, and manage most of the functionalities of the prototype.

For an architectural decision, there are many variables to consider when selecting a specific provider based on the scope of the project. *Table 9* lists the specific requirements that governs the provider required for the dissertation.

|  |  |
| --- | --- |
| Requirement | Problem |
| PaaS Dashboard | End-user dashboard available to consume, track, review, and interact with the data and IoT devices  Allow scalability to quickly incorporate new components  Quick deployment for the developers  Minimal system and hardware maintenance |
| Database and blob storage | NoSQL or relational database manager  Blob Storage container |
| ETL Pipeline | Event-driven  Quick and simple  Capable of image processing  Integrate with storage services |
| IoT service | Capable of providing IoT device management services  Integrate a reliable and consistent messaging broker  Integrate security protocols and services between the edge and the cloud |
| Machine Learning service | Cross-service availability  Fast training and deployment  Low-cost |
| IoT Sandbox Integration | Integration through communication protocols, API, tools available, or otherwise |
| Costs | The final cost must be either within the free-tier usage limits or as low as possible |

Table 9: Cloud provider’s specification requirements for the end prototype

Additional to the requirements set by the scope, there can be additional requirements that may be relevant due to commercial, industrial, or project specific reasons such as SLA, licencing, specific technologies, tools and services offerings, discounts, among others; however, under the context of this dissertation, these are not relevant and not under consideration.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS | Azure | GCP |
| **PaaS** | Yes | Yes | Yes |
| **NoSQL DB Engine** | Yes | Yes | Yes |
| **Blob Storage** | Yes | Yes | Yes |
| **ETL Pipeline** | Yes | Yes | Yes |
| **IoT services** | Yes | Yes | Yes |
| **Machine Learning services** | Yes | Yes | Yes |
| **IoT Sandbox Integration** | Available | Available | Available |

Table 10: Technical Evaluation Matrix of Cloud Providers

From a cost perspective, *table 11* matrix contains a breakdown of estimated costs of services. The estimation was done using the providers’ monthly costs calculators assuming a west Europe zone implementation and within the free-tier offerings. It is important to highlight that all the estimated costs are not necessarily the end-cost; however, these estimates will help for decision making. Additional to free tier offering, the following requirements were also considered:

* 1 PaaS instance for dashboard implementation
* ETL Pipeline integration with 3 instances running in average 200 milliseconds, and up to 300 executions per month
* 2 IoT devices transmitting 24 messages average per day, with an average message size of 3 MB
* ML Training for 8 hours minimum monthly with on-demand prediction deployment

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS | Azure | GCP |
| PaaS service | $0.00 | $0.00 | $0.00 |
| NoSQL DB Engine | $63.47 | $23.36 | $0.00 |
| Blob Storage | $0.00 | $1.00 | $0.00 |
| ETL | $0.00 | $0.00 | $0.00 |
| IoT Services | $0.00 | $0.00 | $0.47 |
| ML service | $146.12 | $64.02 | $2.34 |
| **Total estimated cost** | **$209.59** | **$88.38** | **$2.81** |

Table 11: Cloud Platform Providers' Costs Breakdown

Considering the requirements of table, and the costs breakdown *table 11*, the chosen cloud platform provider will be Google Cloud Platform. From a technical standpoint, all three providers have all the requirements needed for the system, however, from a cost perspective, the implementation through GCP will be lower per month, which is an advantage to reduce overall costs of the prototype deployment. Considering the chosen provider, the end-to-end process will require the following services from GCP:

* IoT Core for IoT device management
* Google Firestore for NoSQL database storage
* Pub/Sub for IoT broker and trigger events in GCP for ETL processes
* Cloud Functions for ETL process
* App Engine for dashboard development
* Cloud Storage for blob storage

### IoT Simulation Sandbox

An important component of the end-to-end system is the IoT simulation sandbox; it can enable simulation of IoT devices interacting from and to the edge, removing physical IoT devices dependence for a full test and its different variables for consideration such as network layer integrations at different levels (device to gateway, gateway to middleware, middleware to cloud). Additionally, combined with the cloud platform provider, and depending on the sandbox, the system can be fully deployed in the same environment, reducing the number of variables that can increase interconnection across systems, such as external API integration, user and password management for different services, implementation of additional security layers, among many other variables. The sandbox must be capable of reproducing IoT behaviour by simulating devices that transmit images from the edge as well as receiving and processing commands based on external events from the dashboard.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Node-RED | NoFloJS | ThingsBoard Community Edition |
| Free tier/open edition | Yes | Yes | No (30-day free trial) |
| IoT device simulation | Yes | Yes | Yes |
| Local storage access and support | Yes | Yes | Yes |
| Cloud Platform integration | Yes (plug-in)[[1]](#footnote-2) | No (plug-in available tagged as failing)[[2]](#footnote-3) | Yes (built-in)[[3]](#footnote-4) |
| Image pre-processing ETL plug-in | Yes | Yes | Yes |
| Active community support and resources | Yes | Yes | Yes |

Table 12: IoT simulation sandbox evaluation matrix

The three platforms offer most of the requirements for the end-to-end system. However, as shown in the evaluation matrix in *table 12*, there are two core components, cost and GCP integration, which are relevant for the scope and nature of the implementation of the prototype; Node-RED fulfils all the requirements for the final prototype:

* MIT Open-source licence
* Rich library of plug-ins for data extraction and processing
* Google Cloud platform integration support, including Pub/Sub and IoT Core plug-ins
* Active community and resources available

### Machine Learning

The machine learning component of the prototype aim is to identify and classify images of concrete structures between crack and no-crack images. Thus, the model will be a binary classification supervised machine learning model, which will use TensorFlow libraries for training, testing, and validating the model for the prototype.

For the training and testing process, a repository of images must be used, which can be enhanced using data augmentation processes to increase data quantity to improve model performance during training phases. The repository of images can have diversity of conditions to reflect real-world conditions, which will be described under chapter 5 of this dissertation.

The final model must ideally be capable of classifying images based on these conditions; however, the main goal of the prototype is to demonstrate a big data system capable of integrating edge devices with a cloud system, thus the model accuracy isn’t the focus, but demonstrating the integration of a machine learning model component into the prototype end-to-end system.

|  |  |  |
| --- | --- | --- |
|  | Vertex AI | Manual Model Deployment |
| Cloud Storage access | Yes | No |
| API Integration | Yes | No |
| Model hyperparameters fine-tuning | Yes (limited) | Yes |
| Data Augmentation | No | Yes |
| Notebook Integration | Yes | Yes |
| Image Classification Costs | $3.46 per hour (training)  $1.37 per hour (deployment) | From $0.1143 per hour up to $7.18 per hour, plus $0.56 to $3.56 per hour for GPU acceleration services |

Table 13: AI products evaluation matrix

For a more accurate and controlled model, a manual model deployment can prove to have better performance and more accurate results compared to Vertex AI, which automatically adjusts the hyperparameters and minimizes the variable adjustment for model training. Additionally, some data augmentation can be done in either a GCP VM container or in a local instance to improve the model performance. However, from a cost-perspective, API integration, and available pre-trained models, Vertex AI can be a better choice, depending on the application.

The scope of the application is aimed at educational and architectural demonstration of an end-to-end system. This translates to greater benefits of using Vertex IA over a manual model deployment based on the evaluation matrix in *table 13*:

* The overall cost for deployment can be minimal for the simple model training with a standard and controlled pricing. This pricing already includes GPU acceleration, which can be benefited for the type of application and dataset size.
* The API integration proves to be a critical component for the dashboard development. Specifically, re-training the model can be done by directly calling the Vertex-AI, which can automatically deploy the model in the pipeline. A manual model deployment requires either the development of an API to emulate this process, or the model must be manually re-trained and re-deployed periodically.

### Dashboard

The dashboard is a component of the application layer where data is presented to the end-user, and a space where the end-user can interact with the edge devices and data directly through the cloud system.

The dashboard functionalities must include:

* Monitor status of edge devices, including connection status and transmission status
* Sending commands to edge devices
* Monitor alerts of the system (crack identified events)
* Identify sensors where a crack event has been triggered
* Review image blobs from the cloud storage

The dashboard must also incorporate security layers from the cloud platform service provided. This means that users must only be able to use the services while connected to an account to the service. Otherwise, service access must be denied.

From a technical standpoint, the dashboard should be served via a website to the users. This means that the hosting can be done either through a virtual machine or through a serverless application through the cloud system; the result can be the same with either approach.

## End-to-end architecture design

From a contextual standpoint, the Kumar R. et al [35] 7-layer design has a level of abstraction that can best reflect the design required for the system prototype for the following points:

* High level of detail for the edge systems in the fog layer by including the network bridge between the edge layers with the cloud layers.
* Outlines the abstraction and details of Big Data components, relevant in the context of this prototype by encapsulating the ETL and ML components.
* Includes an application and business layers that allows to abstract the details for the dashboard set by the problems statement in the context of the prototype.
* A modification to the original design was done to abstract a middleware layer under the Fog Layer, which is inspired by the design pattern proposed by Al-Qaseemi et al. [33].

Figure 9: 8-layer design for the cloud system prototype development [Diagram]

Thus, the design for the prototype will include the following layers:

* **Edge layer:** this layer represents and includes the ‘things’ devices and its components. The devices in the edge layer will have image recollection and network functions to recollect images from concrete structures and transmit the images over from the device and into the middleware layer.
* **Middleware layer:** the middleware layer contains components that can allow image recollection storage in the local fog of the system if there is a communication interruption between either connectivity layer and middleware layer, or between connectivity layer and cloud infrastructure layer. Some pre-processing functions can be implemented in this layer before data transmission from the edge to the cloud infrastructure layer. Commands can be received in this layer, which can then distribute it to the edge layer devices as required.
* **Connectivity layer**: the connectivity layer is the communication bridge between edge layers and cloud and application layers. At this level, specific details of the communication and message protocols are detailed.
* **Cloud infrastructure layer:** the cloud infrastructure layer contains cloud platform services excluding data processing, machine learning services, and IaaS. It includes cloud details such as data storage system, integrated DBMS’s, API’s, security integration, regionalization, and zoning.
* **Data ingestion layer:** this layer encapsulates all the ETL pipelines and processes related to data engineering tasks. The details of ETL does not exclude ETL processes for other requirements such as data storage through the cloud infrastructure, however it makes emphasis for machine learning and data analytics purposes. Data ETL is not necessarily exclusive for edge layer data, however, for the context of this prototype, it will be limited to edge devices data.
* **Data analysis layer:** the data analysis layer main purpose is to highlight tasks specific to data analysis and machine learning models. In this layer, the model training and testing process is described, including details of the chosen model. Additionally, this layer lays the presentation of data analytics from the data from the edge layer. The end-result of this layer affects the application layer. Additionally, like the data ingestion layer, the data analysis layer scope will be limited to the edge layer data exclusively.
* **Application layer:** the application layer includes any application development and integration that presents data from upper layers to the final users. It can also contain any application that enables some level of communication and control with upper layers, as well as additional integrations of functions and libraries that can expand and enhance user experience within the scope of the system. The layer excludes services, applications, and views readily available in the cloud infrastructure layer which are not developed and integrated to the applications developed. Contextually with the prototype, this layer includes the user dashboard that enables feedback from the edge layer.
* **People & Process layer:** an abstract layer that incorporates the end-user of the system and the decision-making process based on the information provided in the application layer.

### Edge Layer

The Edge Layer represents the outer-most layer of the system where the edge or *things* devices reside. The output of this layer are images captured from a camera attached to the edge device, which is transmitted through a WLAN network between the edge layer and the middleware layer.

The main actor in this layer is the edge device, which can be any IoT device so long its capable of capturing and transmitting images from structures, represented in figure 11. While any edge device can be incorporated in this layer, the representation suggests a UAV as the main edge device particularly for the movability advantages it offers; adding mobility to the system can be beneficial to increase the application range, assuming scenarios where stationary or land-based devices cannot reach such as high, inaccessible altitudes such as bridges or oil rigs. However, while the representation implies an UAV use, the nature of this architectural design choice is agnostic to the final edge device type use.

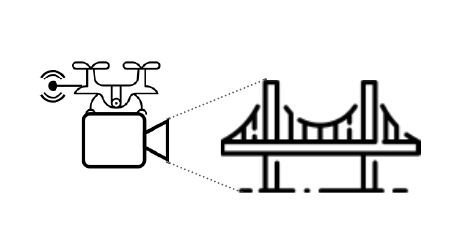


Figure 10: Visual representation of the edge device attached with a camera and wireless sensor [Image]

The advantage of the chosen design is the added flexibility on the implementation. An important finding is that the construction industry can have different device and techniques[citations], which will ultimately impact the final implementation for a system.

Another aspect that this design offers is the possibility to mix and grow as the need requires it – different devices can be used in conjunction as the edge layer offers some level of independence from the rest of the system so long it meets the two main requirements, image acquisition and data transmission.

The edge component has a PAN network component to enable communication between the edge layer and the middleware layer. The specific requirements for the PAN network will be discussed in detail in chapter 4.2.

### Middleware layer

The middleware layer is a central actor between the edge and the network components. This component has multiple uses and goals, including:

* **Pre-processing unit:** the middleware layer adds pre-processing and filtering capabilities to the fog super layer. This can translate to higher data quality by improving image quality of edge devices through the application of filters, such as noise reduction filters, image resizing, and application of different imaging techniques that can increase the image quality before transmitting the images to the cloud system.
* **Central node:** the middleware is a central node that offers a PAN or LAN network where multiple edge devices can get connected. The local connection hardware components and protocols requirements can be of any nature. Additionally, the middleware acts as a bridge of the edge layer and the communication layer.
* **Local repository:** data from edge devices can be temporary stored in the middleware component as means to mitigate network connection problems and transmission errors.

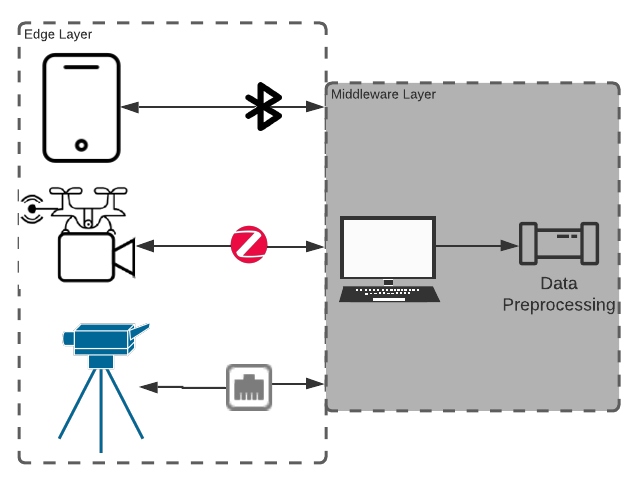


Figure 11: Edge layer and Middleware layer components diagrams [Diagram]

The added middleware component is a flexible abstraction of a computing unit which can add extended services which are not covered in this prototype; however, a highlight of this layer is that adds elasticity and scalability to the fog layer by allowing the addition of new devices through any devices, which can be registered to the cloud through middleware services. Additionally, by removing networking tasks from the edge devices, power consumption can be optimized and, in turn, extending the life of battery powered IoT. It also allows to better system resource usage of cloud systems by pre-processing the data. This design, however, can introduce latency to the system, which isn’t a concern for the specific end-use set by the scope of this system.

### Connectivity Layer

The connectivity layer represents a frontier between the fog super layer, which is formed by the edge, the middleware, and the connectivity layers, and the cloud infrastructure layer. This frontier manages the communications from and to the lower and upper layers of the system, thus it also represents a critical component in the system. The connection allows a two-way communication link between the cloud infrastructure layer and the fog layer, which translates to data transmission from and to the outer-most layers of the prototype.

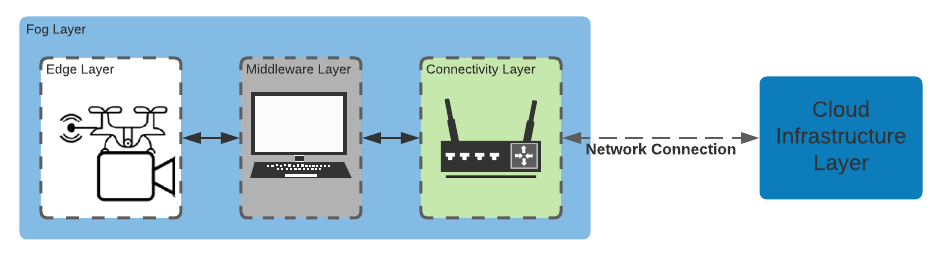


Figure 12: Fog Layer diagram representing the interaction between lower layers in the fog and connectivity to the cloud infrastructure layer [Diagram]

This layer contains the hardware components to connect the middleware layer and cloud layer, specifically routers and gateways that enable a direct network connection to the cloud platforms from the fog layer. It also includes the communication protocol required to communication between the fog layer and the cloud infrastructure which, for the context of this prototype, the communication protocol that is implemented is MQTT protocol, which will be explained in detail in chapter 4.2.

### Cloud Infrastructure Layer

The Cloud Infrastructure layer contains all the components of the cloud platform services, except for services related specifically to the Big Data Layer (Data Ingestion Layer and Data Analysis Layer). The Cloud Infrastructure layer is a central layer which connects all the different layers and acts as the main bridge and orchestrator. The lower layer, the fog layer (including the Edge, Gateway, and Connectivity layers) transmits and receives data from the cloud infrastructure layer through the connectivity layer, and the Big Data Layer and Business Layer consumes and updates the information stored in this layer.

The main components of the layer, from a contextual standpoint, are:

* **IoT management services component:** these services manage the security of edge devices and enables a connection point for the fog layer through a network protocol (MQTT). This is an entry point of data from the edge.
* **Storage:** this component only goal is to store data of objects, which are defined as files (text, binary, or media files) that are made available and accessible to different layers in the system. The storage system can also be considered the data lake or data store of a system.
* **DBMS:** The Database Management System or DBMS component is represented by the database or databases systems integrated in the system. This component can store data from different sources and in different shapes through any DBMS service integrated to the platform, which can be available to different components and layers of the end-to-end system.
* **IAM and Security:** a central security control management service of the system which goal is to ensure security policies of the system are enforced across the cloud platform.
* **API:** a set of interfaces that enables the communication and integration of all the services, tools, and components in the cloud platform. This component can also enable external integration to the cloud layer.

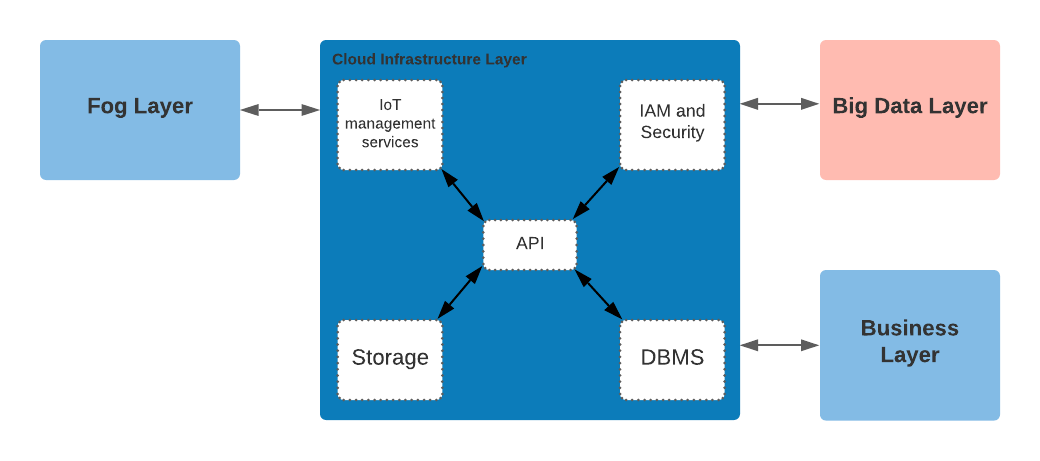


Figure 13: Cloud infrastructure layer representation, showing the layer’s inner services interaction and its contextual relation to the other layers [Diagram]

### Data Ingestion Layer

The Data Ingestion Layer is owner of the ETL process and where the data transformation from the system will occur, and it’s a sub-layer contained under the cloud infrastructure layer where the system processes and storage are managed. The data source can be any data from the system, from upper layers to lower layers datasets, which the main goal is to enhance and improve data quality for the end-to-end system. The main dataset, and most important for the scope of the application, will be the dataset from the edge layers to improve model performance in the Data Analysis Layer.

The Data Ingestion Layer goals are:

* Transport datasets from the edge into the data lake
* Dataset transformation processes
* Dataset classification trigger

Diagram

Description automatically generated

Figure 14: ETL Layer contextualization and description [Diagram]

While this layer includes an event trigger that helps classification of datasets occur, the main component, the classification model, is contained in the Data Analysis layer. However, as part of the ETL process, the Data Ingestion Layer closely interacts with the Data Analysis layer to fulfil its role in the ETL process. This is also reflected in the Data Analysis layer, which will use the end-result of the ETL process, particularly the result of the pipeline which is the cleansed datasets which will be used for model training.

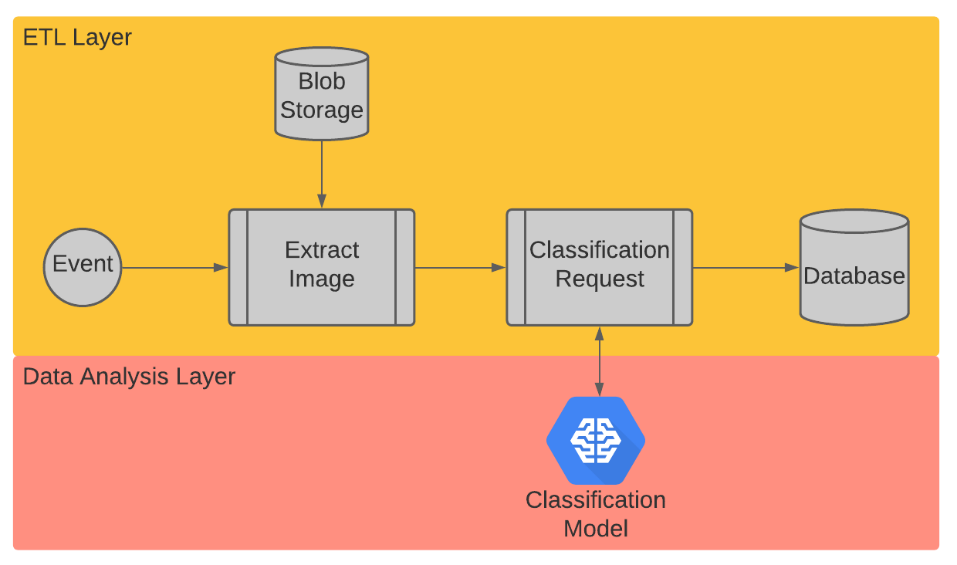


Figure 15: Data Stream flow describing the process for the classification ETL process [Diagram]

### Data Analysis Layer

The Data Analysis Layer, part of the Big Data Layer, is an important component that connects the data that benefits the most and exploits the data from the edge layer. In this layer, all the tasks related to machine learning models are contained, including:

* Data analytics
* Dataset distribution for model training and testing
* Model experimentation
* Model training
* Classification and clustering tasks
* Predictions

While in other integrated systems the Data Analysis can be capable of providing additional functionalities, such as clustering and making predictions, the scope of this system is limited to heuristic classifications and thus, from a contextual standpoint, this layer’s main purpose is to classify images from the edge devices so upper layers can make informed decisions based on the insight provided by this layer.

From a training and testing process standpoint, this layer consumes data from the storage services from the platform that was pre-processed and moved into the correct locations from the lower layer, the Data Ingestion Layer. Once data is ingested, the machine learning service will split the data between train and test and validation data, which will be split in an 80% of the available dataset for training purposes and 20% for test purposes. These proportions are set based on the industry standards. Once data is split, the model training process can start using the dataset, which result will then be tested against the test dataset and log the results and model weights. This flow is described in figure 17.

Diagram

Description automatically generated

Figure 16: Diagram representing the model training/testing process flow [Diagram]

### Application Layer

The application layer contains the upper-most layer of the system from a technical standpoint. In this layer, data is presented to a user of the system, including processed data relevant in the scope of the system. In general, this layer is a gateway between users and systems to communicate the result of the end-to-end process; it enables the decision-making process by presenting the data from lower layers in the system. The main functions for the application layer are:

* **Access to end-user:** the application layer will enable a user GUI through a serverless cloud application which must incorporate user access control mechanisms to prevent unauthorized access to the system.
* **Enable IoT devices remote control from the cloud system:** communication between the edge and the dashboard will be handled via the MQTT broker of the cloud system.
* **Data presentation (user dashboard):** datasets that are required for data presenting will be obtained from the main databases and data lake where all the generated information is stored in the cloud solution. The data will include registered IoT devices, IoT devices logs, images obtained from the edge layer, model results, system performance and metrics, among any relevant data for the scope of this dissertation.

Diagram

Description automatically generated

Figure 17: Application Layer contextualization [Diagram]

While the layer is represented as a single, independent layer, in the context of this dissertation scope, most technical services will be provided and hosted by the Cloud Infrastructure layer, including serverless hosting, API access, user access control, and MQTT broker services. Some functions will be handled directly in a web interface, including the final data presentation and user dashboard.

### People & Process Layer

The People & Process Layer, part of the business super layer, is an abstraction of the business decisions from the results and information that the application layer presents. Contextually, in this layer, the insights the application lights drive *what* will happen based on the information available. Unlike other layers, this layer is an abstract concept of the process after the system has provided information. This layer can also be considered a decision-making process layer.

From a user standpoint, the following scenarios are considered as part of the decision-making process of the prototype:

* **Classification process**
  + **Verification process:** The verification process from the ETL layer when the system has classified an image (crack/no-crack). Once the classification result is ready, the user will be able to verify the classified images in the dashboard to confirm the accuracy of the classification (false or negative classification). The following outcomes will result from the verification process:
    - **Misclassification**: a classification will be considered a misclassification when the outcome of the model does not successfully classify an image, called a false outcome. Regardless of if a misclassification nature is a false positive, or an image classified as crack is misleading, or if it’s misclassified as a false negative, the image will be re-classified in the database, and the re-classification will be logged to improve model performance.
    - **Correct classification:** if a classification is correct, the user won’t have to execute any action – the system will assume under this circumstances that the classification was correct. Once the user has verified this classification, for true positive classifications, the stakeholders can make an informed decision of next steps (e.g., evaluate the degree of crack, carry additional inspections, carry correction measures set by the stakeholders, etc). The steps to follow a true positive classification of fissures takes place are to be set by the stakeholders and experts in the field.
* **Data analytics process:** the data analytics process covers any data analysis task from the data provided by the system, from dashboard data to systems log, and any other data stored anywhere in the platform. The availability of dataset and governance of data is set by the stakeholders of the system, and any data can be used for this process.
* **Device management process:** the device management process includes task related to the management of the edge devices in the system. While policies are not considered for this system, in a real setting, policies should be established to enforce decision making and management of edge devices of a system. The management of said devices can include:
  + **Device monitoring:** the cloud platform should be able to inform the device alive status (connected/not connected) and MQTT call frequency. Log data will also be available in the cloud platform for user consumption to track and monitor devices from the fog layer connected to the cloud infrastructure layer.
  + **Device commands:** through the dashboard, stakeholders with access will be able to send pre-defined commands to the edge devices. This enables some control level of lower layers in the system.
  + **Device onboarding process:** the device onboarding process will be managed by a stakeholder defined in the process. This functions as a control access measure that increases security in the end-to-end system, which can help to decrease potential intrusions in the system.

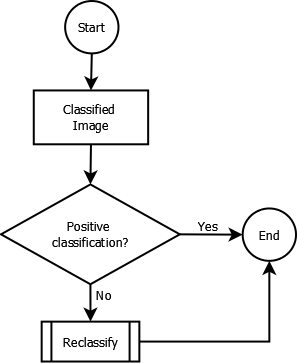
****

Figure 18: Classification verification process flow diagram [Diagram]

# Development and testing

## IoT Device Simulation and Integration

The Device Simulation component main goal is to simulate the edge devices in the end-to-end process. From the design standpoint, the sandbox covers the fog layer, which includes the edge, middleware, and network sub-layers. The most important functions of the sandbox are:

* Consume images from a local repository
* Pre-processing of the dataset
* Subscribe to MQTT broker to transmit and receive messages into and from the cloud
* Emulate some IoT functions

For the implementation of the Node-RED sandbox, the following components were installed in a local Windows 10 system:

* Node.js version 12.20.1
* Node-RED version 1.2.9

Most of the nodes used in the simulation system uses standard nodes included in the installed version of Node-RED. Additionally, Image nodes and Google Cloud nodes were installed via the Node-Red palette to enhance Node-RED.

|  |  |  |  |
| --- | --- | --- | --- |
| Node | Version | Description | Use cases |
| Inject | Included in Node-RED v1.2.9 | Injects a message into a flow, either manually or at regular intervals | Simulates a regular injection of messages in the flow to start the flow process. |
| Functions | Included in Node-RED v1.2.9 | A JavaScript function to run against messages received by the node | * Manages an internal sequence of images to read and transmit through the system * Sets the directory path where the local image repository is * Prepares the byte array and topic to transmit the image over the IoT core device |
| File In | Included in Node-RED v1.2.9 | Reads contents of a file of the hosting system | Reads the image from the source directory and transforms it into a byte array |
| Image | v2.0.1 | Image manipulation node built with Jimp | Resizes the byte array of the image from the original size to a 128 width by 128 height pixels image |
| Google Cloud IoT Core device | v0.1.1 | Node that simulates a registered device in GCP IoT core platform. | * Transmits over MQTT over GCP cloud the images using the topic set in a function node * Receives messages through Pub/Sub and process the request to control the local device |

Table 14: Node-RED components used in the final prototype and the use-case relative to the system

The sandbox implementation was done using a multi-flow system that consists of the following flows:

* **Snapshot flow:** the snapshot flow consists of the main flow that obtains a random set of local images and transmits it to GCP via the IoT broker core node. The flow sequence is as it follows:
  1. **Dataset consumption:** the flow has access to a local repository of images that the flow consumes randomly when a ‘snapshot’ message is triggered. The snapshot trigger works in two different ways, a periodic transmission, which is handled under the intervalometer flow, and a manual snapshot, handled under the commands flow. Access to images is done randomly.
  2. **Image transformation:** once an image is read, the image is resized into a 128x128 image using a Jimp node and subsequently transforms the image into a byte array to prepare it for transmission. The image resizing was done to reduce the size of each message and subsequently, maximize space utilization of cloud storage.
  3. **Message transmission:** the message transmission is an outbound endpoint that consists of a GCP IoT device. The node stores a registry ID, device ID, and a previously registered and signed RS256 key. The registration and key creation were done through GCP IoT Core as described in section. The message transmission sends the byte array from the image transformation nodes and injects it into the MQTT broker.

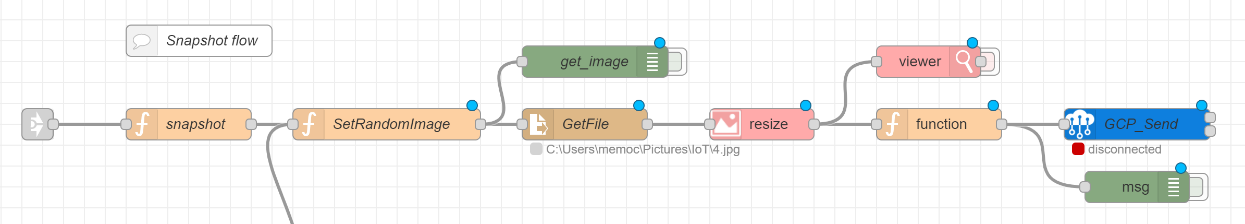


Figure 19: Implemented Node-Red flow displaying the snapshot flow [Image]

* **Intervalometer flow:** The intervalometer flow is a simple controller that regulates the periodicity of the intervals between each ‘snapshot’ automatic request. The node interval rate can be modified via manual override via messages sent from the command flow. The default value of the intervalometer is set to 10 seconds per snapshot.

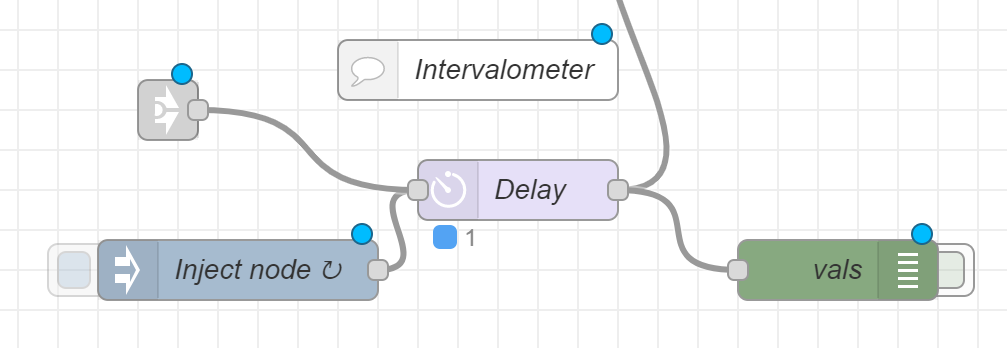


Figure 20: Implemented Node-Red flow displaying the intervalometer flow [Image]

* **Command flow:** the command flow is a central actor between messages received from the MQTT and the interval, snapshots, and to some extent, the simulated IoT device. The command flow redirects received messages from the GCP IoT MQTT broker into the corresponding flow. The command flow consists of:
  1. **Message reception:** a GCP IoT node storing the same data as the outbound node, a registry ID, device ID, and a previously registered and signed RS256 key. The broker is subscribed to the command Pub/Sub to consume the messages transmitted from the dashboard and pushes it into the Switch node.
  2. **Switch Node:** a simple switch that interprets the inbound message from the broker. The switch expects one of the following commands in the message:
     1. **Crack Identified:** if an image is tagged as ‘cracked’ by the model, a Boolean value is passed through in the message via crackidentified value. It can either be True or False. If True, it will trigger the ‘onboard alert light’ to be turned on.
     2. **Alert:** a manual alert that available through the user dashboard. The alert can manually override the onboard alert lights and turn them on.
     3. **Snapshot:** a manual override available through the user dashboard that triggers the snapshot flow.
     4. **Interval:** command to modify the intervalometer rate. The value must be stored in the same command string separated by a dash (e.g., interval-1), where the value must be an integer that represents the number of seconds between each automatic snapshot.
     5. **Reset:** command that resets the interval flow and onboard alert lights through the user dashboard.

|  |
| --- |
|  |

Figure 21: Implemented Node-Red flow displaying the command flow [Image]

## Data Pipeline

The data pipeline is the cloud component that manages the ETL process over the cloud. While some partial transformations are done over the edge to maximize storage utilization of the cloud buckets as well as to decrease message size during data transmission. Most of the pipeline was developed using Cloud Functions using Python 3.8 as the core programming language for the functions, combined with Pub/Sub and event triggers for the orchestration of the pipeline through the system.

The following scenarios were developed as functions for the pipeline:

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Library requirements | Trigger | Results |
| Blob Creation | google-cloud-storage | Pub/Sub IoT message | When a message in the IoT Core topic is listened, the message is processed in the program.  The system confirms the message structure and message content and, if it coincides with an image message:   * Prepares and creates filename by combining registry ID and timestamp * Transforms byte array into a blob image object * Saves the blob into the bucket   Creates a JSON with the blob metadata and stores it into Firestore as a new document. |
| AI classification | google-cloud-firestore  google-cloud-storage  google-cloud-aiplatform | Blob creation event | The system sends the image into the Vertex model to classify (crack/no crack).  The obtained results from the classification are stored in the blob document entry in the database, appending the result and estimation of the model.  Lastly, if the image is classified as crack, the function sends a Pub/Sub to trigger the AI Signalling scenario. |
| AI Signalling |  | Pub/Sub | When the model classifies an image as crack, an alert must be sent to the dashboard and edge devices. The signalling starts through a Pub/Sub message that triggers the function which sends a message via MQTT to two different topics, the edge topic to indicate an alarm from a detection, and the dashboard topic to track in real-time events in the AI classification. |

Table 15: Cloud Functions scenarios descriptions and requirements

## Crack-detection AI Model

### Training, testing, and validation datasets

For the structure crack detection model training, experiments were executed using concrete surface images with a mix of fissure and non-fissure datasets. The dataset is comprised of a total of 763 labelled images, where 537 are labelled as cracked images as seen in figure 24, and 236 as non-cracked datasets, as seen in figure 26, with varying picture quality (different angles, distance from object, focus of object, and illumination). Each image was stored in either one of two directories, the crack dataset pool directory and no-crack dataset pool directory, to subsequently label images for model training through a python program that read filenames and saved the tag based on the directory location of the image file.

Images used for training, testing, and validation tasks were limited to 544 pixels width by 384 height pixels. The dataset distribution was of 80% for training represented by 610 images, 10% for tests represented by 76 images, and 10% for validation represented by 77 images. The images were randomly selected for each task through Vertex AI import toolset which randomly selects images from the source directory (figure 24).



Figure 22:Cracked concrete images dataset sample (By ) [Images]



Figure 23:Normal (no-crack) concrete images dataset sample, taken with a smartphone device [Images]

The crack classification is of binary nature where either one of two tags are considered, crack or no crack. While classification is simple, some variables can affect the nature of the images as discussed in section 2.2, such as fissure-like patterns in structures, optics effects (blurry image, illumination and shading, contrast, noise level)l; additionally, the model is limited to classify whether a crack is found in a structure or not, ignoring the level of damage present in the structure, a variable out of scope for this implementation which is not considered as part of the scope of the end-model.

Vertex AutoML has a three-step configuration setup to train and deploy an AI model:

* **Dataset selection:** The dataset path must be set under this section for model creation. At this stage, the end-goal of the model must be determined. For this dissertation, Image classification with a single label was set. The nature of the process is a simple binary classification, where only one of two results can occur, either a fissure is present in an image or not. Dataset location can either be online, from GCP Storage, or manually uploaded form a local machine. Additionally, a text script containing the filename and classification of the file must be made available; the nature of these models is supervised. For this dissertation, both script and files were made available through GCP Storage. This process is shown in figure 25.
* **Model creation:** the simplest method, when no manual model is loaded, is by selecting Model Creation services. This is a simple service which enables model training from the dataset that was pre-loaded in the service. There are not many functions or options available from a machine learning perspective, for example hyper parameters, convolution layers, or any fine tuning; rather the options are oriented to naming, train-test-validate split selection, and setting a highlighting function to visualize and help to understand hotspots in the image that the model takes into consideration for the classification process. This process is shown in figure 26.
* **Model deployment:** once model has been trained and reviewed, the model can be deployed and made available through an endpoint, a server which enables an online prediction service. Once the endpoint is deployed and while the endpoint remains active, the model can be called via GCP API. Endpoints can be disabled at any given moment.

Graphical user interface, application

Description automatically generated

Figure :Vertex dataset preview

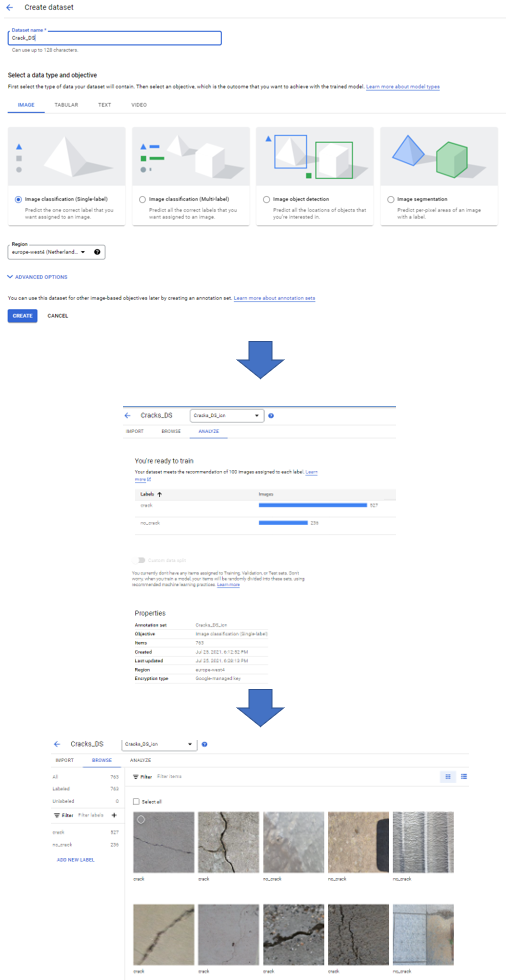


Figure : Dataset creation process in Vertex AutoML [Image]

Graphical user interface, application

Description automatically generated

Figure :AutoML model creation process in Vertex AutoML [Image]

Once a model is trained in Vertex AutoML, one additional parameter which can be maintained is the confidence threshold. This is the confidence level of the model’s certainty of a prediction, whether is a true positive or false positive. The higher the value of the confidence threshold, the higher the precision, but lowers recall, which the number of successful predictions made by the model that is calculated during validation and test process. Vertex integrates additional values to fine-tune this parameter, aiding the user to maximize the results and sensitivity of the model depending on model accuracy and nature. The threshold was set to 0.8 based on the Precision-recall curve and threshold, where recall and precision remains balanced based on the available data. This is shown under figure 27.

Graphical user interface

Description automatically generated with low confidence

Figure :Confidence threshold of the model based on Precision-recall curve and Precision-recall by threshold values of the model [Image]

### Model performance metrics

Model performance is difficult to measure with the data available. The only information that is provided by Vertex AutoML, both through console and API, are:

* **Average precision:** a calculation based on a precision-recall matrix; it estimates the quality of the model.
* **Precision:** proportional number of correct predictions of the classification.
* **Recall:** the proportion of successful predictions by the model. The higher the recall, the lower of false negatives.
* **Precision Recall Curve:** trade-off of precision and recall at different confidence threshold to better select the optimal threshold for model deployment.
* **Precision-recall by threshold:** model performance of the top-scored labels with full range of confidence threshold. Along with the Precision Recall Curve, it aids to select the optimal threshold for model deployment.

Based on these two values provided, recall and precision, Vertex AutoML seems to suggest high accuracy based on precision and recall values, with a 98.7% precision and a recall of 98.7%. Out of all the image, one image resulted in a false positive under the crack classification and, accordingly, one image is a false negative under no\_crack classification.

When image integrated gradients and XRAI is enabled during model creation, the system creates visualization of hot spots in the images that the model used during testing and validation. These hotspots are landmarks or objectives which the model infers are relevant for the image type which results in the classification. As seen in figure 28, the model seems to infer areas near cracks are the main characteristic to classify, not the fissures themselves. Moreover, as seen in figure 29, it seems the model is randomly assuming different surface characteristics as relevant for the no-crack classification.

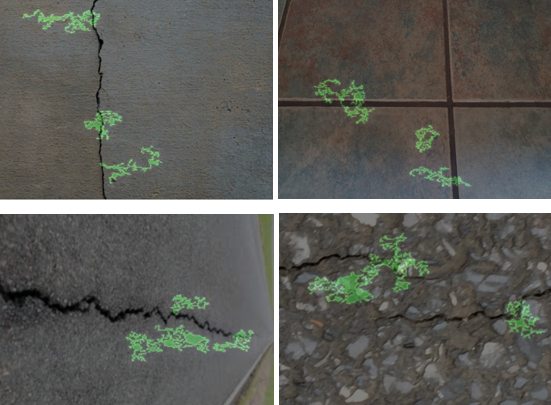


Figure :Classification integrated gradients results of crack identified images [Image]



Figure :Classification integrated gradients results of no-crack identified images [Image]

These results suggests that the model is not necessarily accurate or reliable, however, lack of dataset availability and data provided by Vertex AutoML reduces the amount of information to understand the model and, subsequently, to fine-tune the hyperparameters for better performance. It is probably that some over-fitting can explain the high precision of the model, however, without additional information, this argument cannot be sustained.

## User Dashboard

The User Dashboard is an important component that acts as a bridge between end-users, data, and edge devices in the end-to-end application. Through the dashboard, end-users are enabled to:

* Review data generated in the system
* Oversee and control IoT registries and devices used in the system
* Check devices logs
* View classification results

For the implementation of the dashboard, the following components were used to implement the application:

|  |  |  |
| --- | --- | --- |
| Component | Description | Libraries and frameworks |
| App Engine | Serverless host for dashboard development. The main application was developed using Python 3.7 hosted in a flask server. | * Cryptography 3.4.7 * Flask 2.0.1 * Google Auth 2.0.2 * Google Cloud AI Platform * Google Cloud Firestore * Google Cloud IoT 2.2.1 * Google Cloud Pub/Sub 2.7.0 * Google Cloud Storage * NumPy * Pandas * Pillow 8.4.0 * Requests 2.26.0 |
| Web Application | The main dashboard application web UI was developed using a combination of HTML 5, CSS, and JavaScript. | * Bootstrap 4.0.0 * Datatables 1.11.3 * Datatables Stylesheet 1.11.3 * jQuery 3.5.1 * jQuery UI 1.10.4 * Popper 1.12.9 |

Table 16: Cloud App Engine Components

The web UI system obtains data through direct HTTP calls and consumes it using JSON to arrange and present the information in the system. Data handling and database and datasets access is done through Python functions hosted under the Flask server. The App Engine uses Google IAM authentication to restrict use access to registered users under the GCP platform.

The main web app is divided in three four main programs that handles all the requests and dataflow of the dashboard:

* **Main**: the core python application divided in different functions that handles diverse incoming HTTP requests, handles final data preparations, and serves the data and HTML templates back to the user. Most of the data acquisition and handling is done through the rest of the controllers.
* **IoT Controller**: the main controller that connects with IoT core to obtain all data pertaining to devices registered in GCP IoT Core, including registries, devices, and devices’ logs. Additionally, the Pub/Sub messages are handled through the IoT controller to transmit commands directly to registered devices.
* **Data Controller:** the data controller handles queries to GCP Firestore and transforms and presents query results for the main function as list objects to handle it directly through Python code. Under some analytics functions, data is summarized to decrease data handling from the main functions.
* **Storage Controller:** a controller which only handles blob access and transforms data from the Cloud Storage. Blobs are downloaded and transformed into byte arrays to ease data transfer between functions.

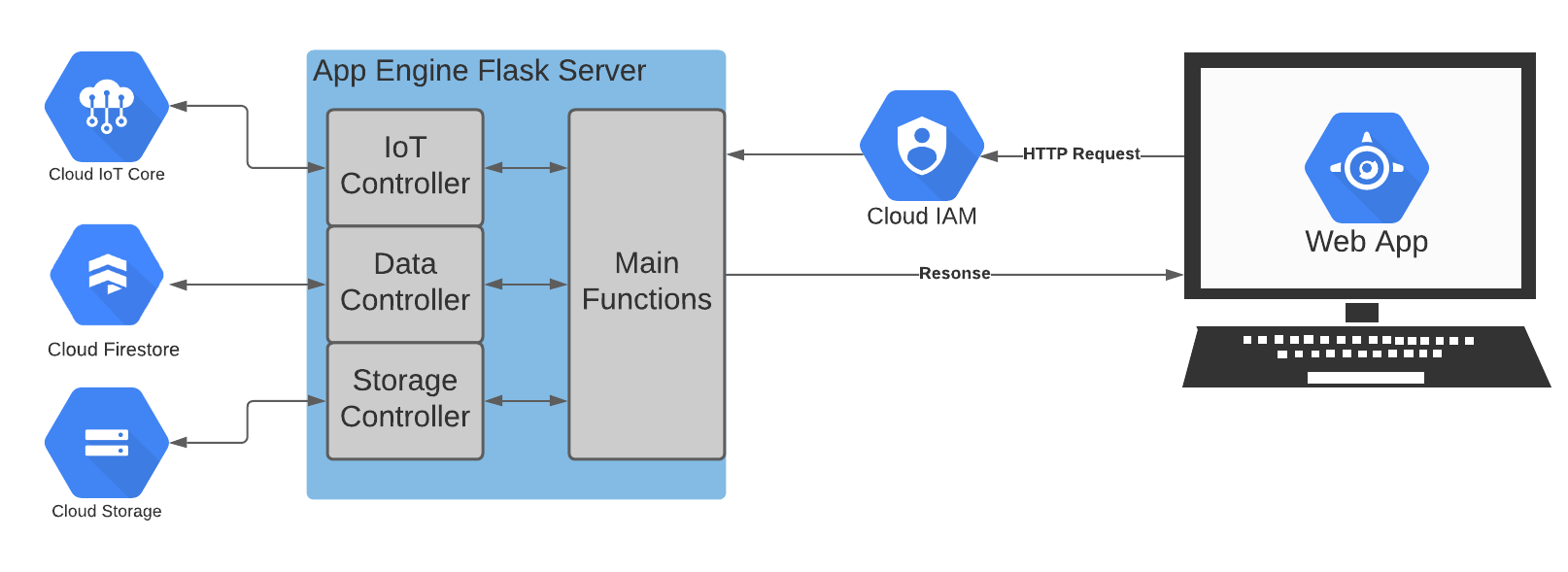


Figure 30: Cloud App data flow representation [Diagram]

The following requests are available in the system:

|  |  |  |
| --- | --- | --- |
| Request Path | Arguments | Description |
| / | N/A | Root path which contains:   * List of all the images, its classification result and confidence level, brief analytics of all the dataset of the system, * IoT device activity dashboard * Chart of detections by devices and registries |
| /listDevs | N/A | Registries List that displays devices by registry. |
| /device | * registryId * deviceId | Device view that contains all data and objects of a specific device, including:   * Device log * Command centre, which enables users to send commands directly to the selected device * Device’s images list, including the filenames, classification, and confidence level * The latest images sent from the device with a highlight function that marks detected cracks in the system. |
| /imageList | N/A | Image viewer that enables users to query the system to obtain the last 99 images by date, registry, or device ID. The dashboard automatically highlights an image if it has been classified with a crack. |

Table 17: HTTP Request Paths in the system

From a functional perspective, the dashboard was developed with the following final functions as part of the scope requirements:

* **Analytics dashboard:** the main landing page where data is condensed, summarized, and presented for the final user. The included data contains miscellaneous data by registry and devices excluding image datasets as shown in figures 31 and 32. The data is obtained from a query to the Firestore database. Available data presented is made available to the end-user after ETL and classification processes. Datasets presented in table format allows user to search and sort through any available column except for statistics and log tables.

Table

Description automatically generated

Figure 31: Analytics Dashboard View displaying general summary and statistics [Image]

Graphical user interface

Description automatically generated

Figure 32: Analytics Dashboard View displaying data by registry and device [Image]

* **Registries and devices:** the registries and devices view summarize all the registries and their corresponding devices available in the IoT core of the project. The registries and devices information are consumed through the GCP IoT core API available in Python.

Graphical user interface, text, application

Description automatically generated

Figure 33: Registries and Devices View displaying available registries' ID's and the available devices [Image]

* **Device Dashboard:** a dashboard subset view where devices’ datasets, including image datasets, and statistic summary are presented for the end-user. Additional functionalities of the device dashboard include:
  + **Classification highlights:** highlights classification results of the images of the device where crack events were detected
  + **Command centre:** gateway to communicate and interact with the IoT device. The available commands are snapshots, where the IoT can get a new snapshot of the structures on demand, alert, where the IoT device can generate an on-board alert through its available functions and hardware, interval, where the interval of period snapshots can be updated, and reset to re-establish the default device settings.

Graphical user interface, text, application, email

Description automatically generated

Figure 34: Device Dashboard main view, including device's logs, dataset view, and command centre [Image]

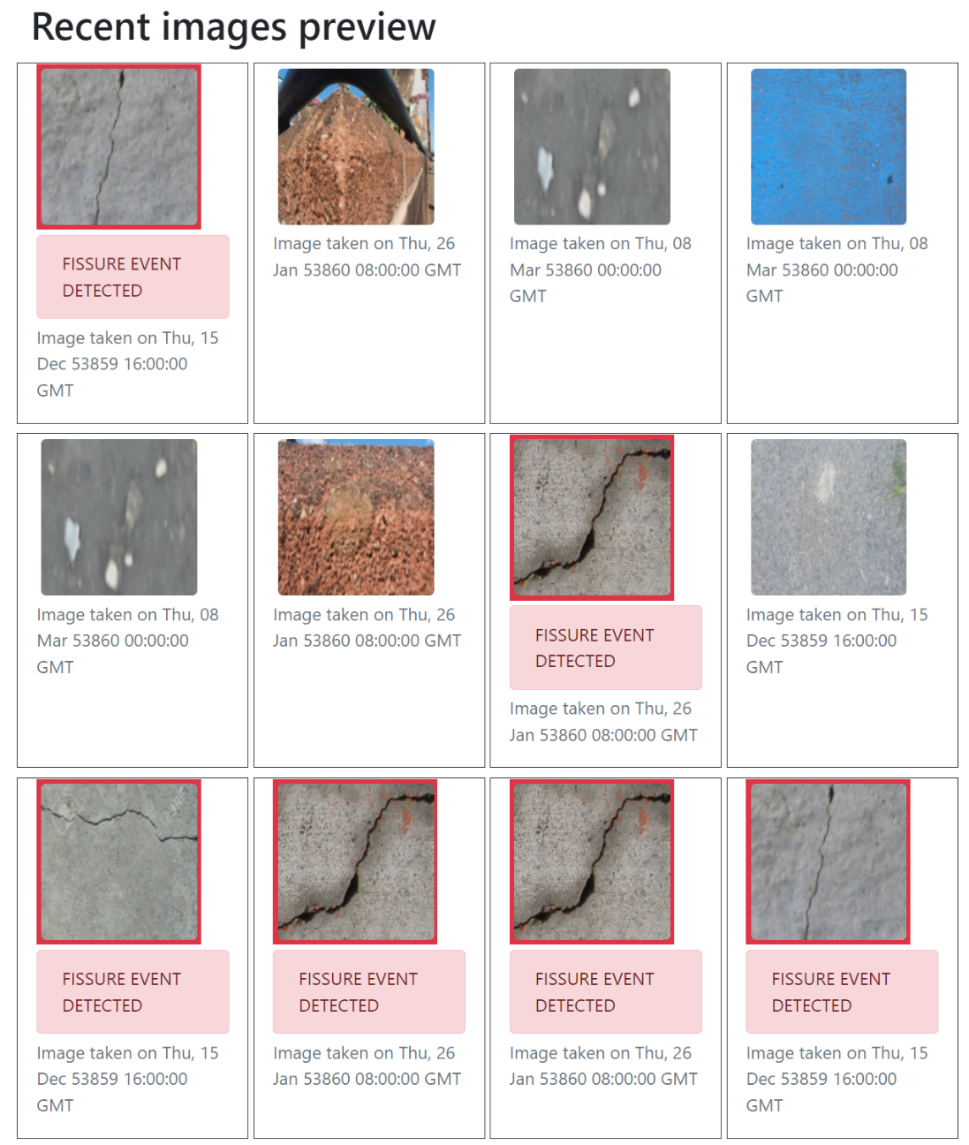


Figure 35: Device Dashboard Recent Images preview displaying fissure and non-fissure events and image highlights [Image]

* **Image Dashboard:** a dashboard where images’ datasets are made visible to the end-user. This dashboard has filters available for users to search by date range, registry ID, or device ID. The dataset query is limited up to 100 images per query for performance and visualization.

Graphical user interface, application

Description automatically generated with medium confidence

Figure 36: Image Viewer Dashboard preview, displaying images from a date range query. The result tags images by fissure and non-fissure events and image highlights [Image]

## Limitations during implementation

The implementation of the design of the final prototype faced diverse challenges during the development, most of which were successfully overcome. Nonetheless, some of these challenges proved an obstacle, causing a divergence from the original scope of this dissertation with the end-result. It is important to highlight, however, that both, success and failure, lights information that can result in more robust and better system implementations:

|  |  |  |
| --- | --- | --- |
| Problem | Description | Resolution |
| **Pub/Sub Message Size** | * Image transmission from the emulated IoT devices was using high-fidelity images, some of which were over 10MB * Pub/Sub limits publish messages’ body to 10MB | * A node was incorporated in Node-RED to resize images and maintain messages under the limit. |
| **Pub/Sub throughput** | * Connection with Pub/Sub limits the number of message publication to 200 MB/s in europe-west2 region * Multiple devices streaming simultaneously, or multiple requests per second that exceeds 200 MB/s can overwhelm the connection, causing instability | * Added controllers in the Node-RED to avoid overwhelming a single connection * Maintain device streaming under 8 devices to maintain stability with some breathing room. |
| **Model Blackbox** | * Vertex AutoML creates a black box that hides and removes the hyperparameters of the model * Analytics and model data is limited for the system administrator and end-users | * No resolution was found with AutoML |
| **Model deployment costs** | * The system was implemented with online prediction in mind, which consumes credits over time while the endpoint remains active * Vertex AutoML deployment has two different deployment services, online   predictions and batch predictions.   * Online predictions are on-demand, synchronous evaluations of datasets through the API * Batch predictions are asynchronous evaluations of batches of datasets. * Online prediction requires a persistent endpoint availability, which consumes credits, increasing cost over time * Batch prediction doesn’t require endpoint availability | * Due to design, online prediction was used; however, the endpoint had to be activated and deactivated during tests to reduce costs during testing. |
| **Reclassification function** | * The scope of the dashboard included a reclassification function that could be triggered by end-users when a classification is done incorrectly. | * There was no resolution to this problem, as the final system did not include this functionality due to lack of knowledge of the Vertex AI API. |

Table 18: Implementation Limitations and Problems Matrix

# Conclusions and Further Work

This dissertation scope was defined to design and implement a distributed architecture design for a practical demonstration of an NDE system. This was achieved by incorporating an IoT edge device paired with a cloud system, which required a user dashboard and a classification model to monitor and audit concrete structure health and integrity. The architecture design goal was to integrate an IoT component and a big data and analytics elements through a cloud-based platform service for the beforementioned scope.

The final architectural design of choice revised and expanded upon Mohammad, Jamali et. al. and Al-Qaseemi, S. et. al. 5-layer architectural designs to propose a 7 layers design. The abstraction level allows to explain and highlight four of the main components covered by the scope of the system, the IoT system, the ETL and ML process, and the end-user dashboard, while retaining core elements which both 5-layer designs offer, interconnection and middleware components between the fog layer and cloud systems, the cloud platform infrastructure components, and the end-user abstraction which abstracts business and functional elements for decision-making processes. A demonstration prototype was subsequently developed based on the design. This development consisted in the implementation of an IoT simulation sandbox to emulate an edge device, an event driven ETL pipeline to process data from the fog layer, a binary classification model, a blob and database storage systems, and a PaaS application for the end-user dashboard.

The IoT simulation sandbox was chosen due to its fast and simple deployment that emulated an end-to-end system without requiring physical IoT devices. It incorporated all the required technical functions set in the scope and reach of the original design which demonstrates flexibility and lack of constraints where time or resources are limited, or for systems where edge devices choices have not been fully defined or finalized. However, among elements that cannot be tested or measured in simulation sandboxes are hardware-specific limitations and problems that physical devices can raise, as connectivity latency or limitations, operational problems, or specific concerns such as damage or restrictions of these devices, among many other scenarios that these systems do not or cannot incorporate.

For the Big Data layer, the ETL pipeline and AI model elements were developed and deployed for the prototype. The ETL pipeline was developed with Cloud Functions as an event-driven service which enabled reception and processing of datasets from the fog layer to classification of new datasets. There’s an important restriction in scalability through Cloud Functions, especially with the event-driven design that can create a bottleneck for high-volume streams of data. This wasn’t a limitation under the scope of this dissertation; however, for better resilience, growth, and reliability for systems, where high-volume, high-velocity data streaming is a core element of the scope, other alternatives should be pursued.

The AI model, a supervised binary classification, was trained as part of the NDE scope to identify whether structures presented fissures and/or cracks or not. It was implemented using AI Vertex, which automatically sets hyperparameters, reducing complexity for new, inexperienced, or simple model training and deployments. The end-result was adequate for the prototype demonstration where most of the images were classified successfully. However, the results seem to indicate an overfitting of the model due to the lack of image diversity.

An important challenge for ML models in NDE systems relates to diverse conditions and scenarios that can result in misclassifications, increasing uncertainty and unreliability of a model. This is also noted in previous research done in the area, creating an important obstacle for AI models. Still, the limitation of the scope of the dissertation did not aim for model performance. Further works can investigate improvements of model performance and reliance by incorporating and expanding over new techniques, datasets, and tools used in combination with the design and platform developed for this prototype.

Vertex AutoML presents an interesting, simple service that can help reduce time and costs associated with model training and deployment in implementations and systems where budget, time, or knowledge can be limited. This can be a viable and, for some applications, affordable option for diverse professionals and institutions in the data and IT industry. Nevertheless, it is worth mentioning that the oversimplification of details and minimal control and visibility of hyperparameters creates a black box for system integrations that can limit the understanding, models performance gain, or otherwise.

The final dashboard prototype included most requirements in scope. The development resulted in a simple, fast, and with minimal efforts in part due to the PaaS design of App Engine which removes complexity of server maintenance and configuration tasks. The functions which were incorporated includes data and image visualization dashboards, devices views, and devices remote control through commands. However, the incorporation of the reclassification function wasn’t included due to knowledge limitations of the API. This last missing functionality can benefit similar systems by improving accuracy and reliability of the model as time progresses by end-users by direct interaction and reclassification of false results made by the model.

The architecture design, technical requirements, tools, expertise, and scope of a project will govern the final costs. This dissertation demonstrated that costs can be minimal and limited to the ML services. Costs limitations such as these can aid diverse industries, institutions, and professionals to create end-systems that can meet complex needs. Still, during implementation, while costs were reduced, during the implementation the costs resulted higher due to an AutoML on-demand service chosen during development; these costs could have been reduced through a batch prediction service. Nonetheless, this implementation demonstrates a viable low-cost option, which can be replicated for different applications and uses, both industrial and academic

As an NDE tool, which is one of the main objectives of the scope, the prototype demonstrated that similar designs and systems have the potential to be incorporated for structural assessments and evaluation processes and tools in the construction industry. However, it’s important to recognize the improvement areas uncovered in this dissertation, highlighting the model performance and accuracy. The goal of any NDE is to detect early signs of deterioration in buildings and structures which is only possible with the expertise and involvement of experienced professionals of the field. While the final prototype is flawed from this perspective, it can improve by incorporating more and richer datasets, new and existing toolsets, incorporation of diverse NDE techniques, and the involvement of experts in the field, and ultimately further pushing the boundaries of current technologies.

Since this dissertation scope is to design and develop a prototype for demonstration of a practical distributed system that incorporates IoT and ML as part of its components, additional research and improvements for fissure detection models can be an interesting enhancement for similar systems. Additionally, exploring different and additional datasets for model improvements, such as infrared imaging and percussion-based datasets, can result in a more robust system and could overcome some of the limitations which this dissertation met. The end design of the end-system can accommodate different and better techniques and tools, which can ultimately result in a robust, inexpensive, and reliable NDE tool.

# References

[1] Galvis, F., Miranda, E., Heresi, P., Dávalos, H. and Ruiz-García, J., 2020. Overview of collapsed buildings in Mexico City after the 19 September 2017 (Mw7.1) earthquake. Earthquake Spectra, 36(2\_suppl), pp.83-109. Available at: <https://journals.sagepub.com/doi/abs/10.1177/8755293020936694>

[2] Quinde, P. and Reinoso, E., 2020. Long duration and frequent, intense earthquakes: Lessons learned from the 19 September 2017 earthquake for Mexico City’s resilience. *Earthquake Spectra*, 36(2\_suppl), pp.49-61. Available at: <https://journals.sagepub.com/doi/pdf/10.1177/8755293020942548>

[3] García-Rueda, G., Valdovinos, R., Valdés-González, J., Alejo, R., González-Ruiz, J. and Marcial-Romero, J., 2020. Analysis of Repair Costs of Scholar Buildings Affected by Earthquakes Using Data Mining. Case Study: Earthquakes of 2017 in Mexico. *Lecture Notes in Computer Science*, [online] pp.45-56. Available at: <https://link.springer.com/chapter/10.1007/978-3-030-49076-8\_5> [Accessed 13 July 2021].

[4] Pacheco-Torgal, F., Abdollahnejad, Z., Miraldo, S., Baklouti, S. and Ding, Y., 2012. An overview on the potential of geopolymers for concrete infrastructure rehabilitation. *Construction and Building Materials*, [online] 36, pp.1053-1058. Available at: <https://www.sciencedirect.com/science/article/pii/S0950061812004722> [Accessed 12 July 2021].

[5] Frangopol, D. and Liu, M., 2007. Maintenance and management of civil infrastructure based on condition, safety, optimization, and life-cycle cost∗. *Structure and Infrastructure Engineering*, [online] 3(1), pp.29-41. Available at: <https://www.tandfonline.com/doi/full/10.1080/15732470500253164?scroll=top&needAccess=true> [Accessed 11 July 2021].

[6] Transport, 2017. Infrastructure maintenance. [online] Available at: <https://data.oecd.org/transport/infrastructure-maintenance.htm#indicator-char> [Accessed 12 July 2021].

[7] American Society for Nondestructive Testing. 2021. *INTRODUCTION TO NONDESTRUCTIVE TESTING*. [online] Available at: <https://asnt.org/MajorSiteSections/About/Introduction\_to\_Nondestructive\_Testing.aspx> [Accessed 19 July 2021].

[8] H., Z., A., H. and M., M., 2015. Internet of Things (IoT): Definitions, Challenges and Recent Research Directions. International Journal of Computer Applications, [online] 128(1), pp.37-47. Available at: <https://www.researchgate.net/profile/Zozo-Hassan/publication/320532203\_Internet\_of\_Things\_IoT\_Definitions\_Challenges\_and\_Recent\_Research\_Directions/links/59ea1d4ba6fdccef8b08cc3e/Internet-of-Things-IoT-Definitions-Challenges-and-Recent-Research-Directions.pdf> [Accessed 19 July 2021].

[9] American Society for Nondestructive Testing. 2021. INTRODUCTION TO NONDESTRUCTIVE TESTING. [online] Available at: <https://asnt.org/MajorSiteSections/About/Introduction\_to\_Nondestructive\_Testing.aspx> [Accessed 19 July 2021].

[10] Govinfo.gov. 2021. Rules and Regulations. [online] Available at: <https://www.govinfo.gov/content/pkg/FR-2004-12-14/pdf/04-27355.pdf> [Accessed 13 July 2021].

[11] Zink, J., Holdhusen, B. and Lovelace, B., 2021. Unmanned Aerial Vehicles Enable Safe and Cost-Effective Bridge Inspection. [online] Dot.state.mn.us. Available at: <http://www.dot.state.mn.us/research/TS/2015/201540TS.pdf> [Accessed 13 July 2021].

[12] Liu, Y., Yao, J., Lu, X., Xie, R. and Li, L., 2019. DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. Neurocomputing, [online] 338, pp.139-153. Available at: <https://github.com/yhlleo/DeepCrack> [Accessed 4 July 2021].

[13] Omer, M., Margetts, L., Hadi Mosleh, M., Hewitt, S. and Parwaiz, M., 2019. Use of gaming technology to bring bridge inspection to the office. Structure and Infrastructure Engineering, [online] 15(10), pp.1292-1307. Available at: <https://www.tandfonline.com/doi/full/10.1080/15732479.2019.1615962> [Accessed 14 July 2021].

[14] Electronic Journal of Structural Engineering, 2015. Non-Destructive Testing of Concrete: A Review of Methods. [online] Available at: <https://www.researchgate.net/publication/282938427\_Non-Destructive\_Testing\_of\_Concrete\_A\_Review\_of\_Methods> [Accessed 17 July 2021].

[15] Park, S., Stubbs, N., Bolton, R., Choi, S. and Sikorsky, C., 2001. Field Verification of the Damage Index Method in a Concrete Box-Girder Bridge via Visual Inspection. Computer-Aided Civil and Infrastructure Engineering, [online] 16(1), pp.58-70. Available at: <https://onlinelibrary.wiley.com/doi/epdf/10.1111/0885-9507.00213?saml\_referrer> [Accessed 17 July 2021].

[16] Layssi, H., Ghods, P., Alizadeh, A. and Salehi, M., 2021. Electrical Resistivity of Concrete: Concepts, applications, and measurement techniques. Concrete International, [online] Available at: <https://www.giatecscientific.com/wp-content/uploads/2015/05/Concrete\_Electrical\_Resistivity.pdf> [Accessed 13 July 2021].

[17] Agnisarman, S., Lopes, S., Chalil Madathil, K., Piratla, K. and Gramopadhye, A., 2019. A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection. Automation in Construction, [online] 97, pp.52-76. Available at: <https://www.sciencedirect.com/science/article/pii/S0926580518303248> [Accessed 12 July 2021].

[18] Ono, K., 2008. Structural Integrity Evaluation by Means of Acoustic Emission. Acoustic Emission and Critical Phenomena, [online] pp.13-27. Available at: <https://www.researchgate.net/profile/Kanji-Ono/publication/268366237\_Structural\_integrity\_evaluation\_using\_acoustic\_emission/links/5779805d08ae4645d611f160/Structural-integrity-evaluation-using-acoustic-emission.pdf> [Accessed 4 July 2021].

[19] Garrido, I., Lagüela, S., Otero, R. and Arias, P., 2020. Thermographic methodologies used in infrastructure inspection: A review—Post-processing procedures. Applied Energy, [online] 266, p.114857. Available at: <https://www.sciencedirect.com/science/article/pii/S030626192030369X> [Accessed 13 July 2021].

[20] Chang, P., Flatau, A. and Liu, S., 2003. Review Paper: Health Monitoring of Civil Infrastructure. Structural Health Monitoring, [online] 2(3), pp.257-267. Available at: <https://journals.sagepub.com/doi/pdf/10.1177/1475921703036169> [Accessed 13 July 2021].

[21] Sheils, E., O'Connor, A., Schoefs, F. and Breysse, D., 2012. Investigation of the effect of the quality of inspection techniques on the optimal inspection interval for structures. Structure and Infrastructure Engineering, [online] 8(6), pp.557-568. Available at: <https://www.tandfonline.com/doi/full/10.1080/15732479.2010.505377?scroll=top&needAccess=true> [Accessed 17 July 2021].

[22] Rolander, D., Phares, B., Graybeal, B., Moore, M. and Washer, G., 2001. Highway Bridge Inspection: State-of-the-Practice Survey. Transportation Research Record: Journal of the Transportation Research Board, [online] 1749(1), pp.73-81. Available at: <https://journals.sagepub.com/doi/pdf/10.3141/1749-12> [Accessed 13 July 2021].

[23] Fujita, Y., Mitani, Y. and Hamamoto, Y., 2006. A Method for Crack Detection on a Concrete Structure. 18th International Conference on Pattern Recognition (ICPR'06), [online] Available at: <https://ieeexplore.ieee.org/document/1699671> [Accessed 4 July 2021].

[24] Farhidzadeh, A., Dehghan-Niri, E., Moustafa, A., Salamone, S. and Whittaker, A., 2013. Damage Assessment of Reinforced Concrete Structures Using Fractal Analysis of Residual Crack Patterns. Experimental Mechanics, [online] 53(9), pp.1607-1619. Available at: <https://link.springer.com/article/10.1007/s11340-013-9769-7> [Accessed 4 July 2021].

[25] Kim, I., Jeon, H., Baek, S., Hong, W. and Jung, H., 2018. Application of Crack Identification Techniques for an Aging Concrete Bridge Inspection Using an Unmanned Aerial Vehicle. Sensors, [online] 18(6), p.1881. Available at: <https://www.mdpi.com/1424-8220/18/6/1881/htm> [Accessed 16 July 2021].

[26] Salhaoui, M., Guerrero-González, A., Arioua, M., Ortiz, F., El Oualkadi, A. and Torregrosa, C., 2019. Smart Industrial IoT Monitoring and Control System Based on UAV and Cloud Computing Applied to a Concrete Plant. Sensors, [online] 19(15), p.3316. Available at: <https://www.mdpi.com/1424-8220/19/15/3316/htm> [Accessed 17 July 2021].

[27] Liu, X., Ai, Y. and Scherer, S., 2017. Robust image-based crack detection in concrete structure using multi-scale enhancement and visual features. 2017 IEEE International Conference on Image Processing (ICIP), [online] Available at: <https://ieeexplore.ieee.org/abstract/document/8296693> [Accessed 4 July 2021].

[28] Minami, T., Urata, W., Fujiu, M., Fukuoka, T., Sagae, M., Suda, S. and Takayama, J., 2019. Development of Automatic Concrete Cracks Detection System Using Average Shifted Mesh. Journal of the Eastern Asia Society For Transportation Studies, [online] 13. Available at: <https://www.jstage.jst.go.jp/article/easts/13/0/13\_1560/\_pdf/-char/en> [Accessed 4 July 2021].

[29] Fujita, Y., Mitani, Y. and Hamamoto, Y., 2006. A Method for Crack Detection on a Concrete Structure. 18th International Conference on Pattern Recognition (ICPR'06), [online] Available at: <https://ieeexplore.ieee.org/document/1699671> [Accessed 4 July 2021].

[30] Chang, P., Flatau, A. and Liu, S., 2003. Review Paper: Health Monitoring of Civil Infrastructure. Structural Health Monitoring, [online] 2(3), pp.257-267. Available at: <https://journals.sagepub.com/doi/pdf/10.1177/1475921703036169> [Accessed 11 July 2021].

[31] Washizaki, H., Ogata, S., Hazeyama, A., Okubo, T., Fernandez, E. and Yoshioka, N., 2020. Landscape of Architecture and Design Patterns for IoT Systems. IEEE Internet of Things Journal, [online] 7(10), pp.10091-10101. Available at: <https://ieeexplore.ieee.org/abstract/document/9120234> [Accessed 17 July 2021].

[32] Soderborg, N., Crawley, E. and Dori, D., 2003. System function and architecture. Communications of the ACM, [online] 46(10), pp.67-72. Available at: <https://dl.acm.org/doi/fullHtml/10.1145/944217.944241?casa\_token=3OGXk9qKVG0AAAAA:TmBHuhMiVDD6FY-UzQjstwAPoucQaHeEweKi150pPZAIELQl6WLHnETnuqBX7rpbkvjIQBR3om0KTg> [Accessed 10 July 2021].

[33] Al-Qaseemi, S., Almulhim, H., Almulhim, M. and Chaudhry, S., 2016. IoT architecture challenges and issues: Lack of standardization. 2016 Future Technologies Conference (FTC), [online] Available at: <https://ieeexplore.ieee.org/abstract/document/7821686> [Accessed 10 July 2021].

[34] Jabraeil Jamali, M., Bahrami, B., Heidari, A., Allahverdizadeh, P. and Norouzi, F., 2019. IoT Architecture. Towards the Internet of Things, pp.9-31.

[35] Kumar, R. and Smys, S., 2018. A novel report on architecture, protocols and applications in Internet of Things (IoT). 2018 2nd International Conference on Inventive Systems and Control (ICISC), [online] Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8398986> [Accessed 9 July 2021].

[36] Pahl, C., Ioini, N., Helmer, S. and Lee, B., 2018. An architecture pattern for trusted orchestration in IoT edge clouds. 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), [online] Available at: <https://ieeexplore.ieee.org/abstract/document/8364046> [Accessed 23 July 2021].

[37] Pahl, C., Helmer, S., Miori, L., Sanin, J. and Lee, B., 2016. A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters. 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), [online] Available at: <https://ieeexplore.ieee.org/abstract/document/7592710> [Accessed 24 July 2021].

[38] Aleem, A. and Ryan Sprott, C., 2012. Let me in the cloud: analysis of the benefit and risk assessment of cloud platform. Journal of Financial Crime, [online] 20(1), pp.6-24. Available at: <https://www.emerald.com/insight/content/doi/10.1108/13590791311287337/full/html> [Accessed 22 July 2021].

[39] Vaquero, L., Rodero-Merino, L., Caceres, J. and Lindner, M., 2008. A break in the clouds. ACM SIGCOMM Computer Communication Review, [online] 39(1), pp.50-55. Available at: <https://dl.acm.org/doi/pdf/10.1145/1496091.1496100> [Accessed 22 July 2021].

[40] Ren, L., Zhang, L., Tao, F., Zhao, C., Chai, X. and Zhao, X., 2013. Cloud manufacturing: from concept to practice. Enterprise Information Systems, [online] 9(2), pp.186-209. Available at: <https://www.tandfonline.com/doi/full/10.1080/17517575.2013.839055?needAccess=true> [Accessed 22 July 2021].

[41] Richter, F., 2021. Infographic: Amazon Leads $150-Billion Cloud Market. [online] Statista Infographics. Available at: <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/> [Accessed 22 July 2021].

[42] Education, I., 2021. paas. [online] Ibm.com. Available at: <https://www.ibm.com/cloud/learn/paas> [Accessed 11 July 2021].

[43] Google Cloud. 2021. App Engine Application Platform  |  Google Cloud. [online] Available at: <https://cloud.google.com/appengine#all-features> [Accessed 11 July 2021].

[44] Amazon Web Services, Inc. 2021. AWS Elastic Beanstalk Product Details - Amazon Web Services (AWS). [online] Available at: <https://aws.amazon.com/elasticbeanstalk/details/> [Accessed 11 July 2021].

[45] Azure.microsoft.com. 2021. App Service | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/services/app-service/> [Accessed 11 July 2021].

[46] Google Cloud. 2021. Setting Up Access Control. [online] Available at: <https://cloud.google.com/appengine/docs/flexible/custom-runtimes/access-control> [Accessed 9 July 2021].

[47] Azure.microsoft.com. 2021. Pricing – App Service for Linux | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/pricing/details/app-service/linux/> [Accessed 9 July 2021].

[48] Encyclopedia Britannica. 2021. database | Definition, Types, & Facts. [online] Available at: <https://www.britannica.com/technology/database> [Accessed 24 July 2021].

[49] Oracle.com. 2021. What is a relational database?. [online] Available at: <https://www.oracle.com/uk/database/what-is-a-relational-database/#:~:text=A%20relational%20database%20is%20a,are%20related%20to%20one%20another.&text=The%20columns%20of%20the%20table,the%20relationships%20among%20data%20points.> [Accessed 24 July 2021].

[50] Ibm.com. 2021. nosql-databases. [online] Available at: <https://www.ibm.com/cloud/learn/nosql-databases> [Accessed 24 July 2021].

[51] AWS Databases: Break Free to Save, a., 2021. Databases on AWS. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/products/databases/> [Accessed 11 July 2021].

[52] Azure.microsoft.com. 2021. Azure Databases - Types of Databases on Azure | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-us/product-categories/databases/> [Accessed 11 July 2021].

[53] Google Cloud. 2021. Cloud Storage  |  Google Cloud. [online] Available at: <https://cloud.google.com/storage#all-features> [Accessed 4 July 2021].

[54] Amazon Web Services, Inc. 2021. Amazon RDS Free Tier | Cloud Relational Database | Amazon Web Services. [online] Available at: <https://aws.amazon.com/rds/free/> [Accessed 11 July 2021].

[55] Azure.microsoft.com. 2021. Azure Free Account FAQ | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-us/free/free-account-faq/> [Accessed 11 July 2021].

[56] Google Cloud. 2021. Google Cloud Free Program. [online] Available at: <https://cloud.google.com/free/docs/gcp-free-tier/#bigquery> [Accessed 11 July 2021].

[57] Google Cloud. 2021. Google Cloud Free Program. [online] Available at: <https://cloud.google.com/free/docs/gcp-free-tier/#firestore> [Accessed 11 July 2021].

[58] Oracle.com. 2021. What is the Internet of Things (IoT)?. [online] Available at: <https://www.oracle.com/uk/internet-of-things/what-is-iot/> [Accessed 24 July 2021].

[59] Amazon Web Services, Inc. 2021. AWS IoT Core Features - Amazon Web Services. [online] Available at: <https://aws.amazon.com/iot-core/features/?nc=sn&loc=3> [Accessed 11 July 2021].

[60] Docs.microsoft.com. 2021. Azure IoT Hub Documentation. [online] Available at: <https://docs.microsoft.com/en-us/azure/iot-hub/> [Accessed 11 July 2021].

[61] Google Cloud. 2021. Cloud IoT Core  |  Google Cloud. [online] Available at: <https://cloud.google.com/iot-core> [Accessed 11 July 2021].

[62] Azure.microsoft.com. 2021. Pricing – IoT Hub | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/pricing/details/iot-hub/> [Accessed 11 July 2021].

[63] Docs.microsoft.com. 2021. Control access to IoT Hub using SAS tokens. [online] Available at: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-dev-guide-sas?tabs=node#authenticating-a-device-to-iot-hub> [Accessed 11 July 2021].

[64] Docs.microsoft.com. 2021. Access control and security for IoT Hub. [online] Available at: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security> [Accessed 11 July 2021].

[65] Amazon Web Services, Inc. 2021. AWS IoT Core Pricing - Amazon Web Services. [online] Available at: <https://aws.amazon.com/iot-core/pricing/?nc=sn&loc=4> [Accessed 11 July 2021].

[66] Docs.microsoft.com. 2021. Azure subscription limits and quotas - Azure Resource Manager. [online] Available at: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits> [Accessed 11 July 2021].

[67] LoRa Alliance®. 2021. What is LoRaWAN® Specification - LoRa Alliance®. [online] Available at: <https://lora-alliance.org/about-lorawan/> [Accessed 10 July 2021].

[68] Amqp.org. 2021. AMQP is the Internet Protocol for Business Messaging | AMQP. [online] Available at: <https://www.amqp.org/about/what> [Accessed 11 July 2021].

[69] Lists.oasis-open.org. 2021. [online] Available at: <https://lists.oasis-open.org/archives/amqp/201202/msg00086/StormMQ\_WhitePaper\_-\_A\_Comparison\_of\_AMQP\_and\_MQTT.pdf> [Accessed 11 July 2021].

[70] Snowflake. 2021. ETL Pipeline. [online] Available at: <https://www.snowflake.com/guides/etl-pipeline#:~:text=An%20ETL%20pipeline%20is%20the,and%20move%20it%20to%20another.> [Accessed 24 July 2021].

[71] 2021. What is orchestration?. [online] Available at: <https://www.redhat.com/en/topics/automation/what-is-orchestration> [Accessed 24 July 2021].

[72] Amazon Web Services, Inc. 2021. AWS Glue - Managed ETL Service - Amazon Web Services. [online] Available at: <https://aws.amazon.com/glue/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc> [Accessed 24 July 2021].

[73] Azure.microsoft.com. 2021. Data Factory – Data integration service | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/services/data-factory/> [Accessed 24 July 2021].

[74] Google Cloud. 2021. Dataflow  |  Google Cloud. [online] Available at: <https://cloud.google.com/dataflow> [Accessed 4 July 2021].

[75] Amazon Web Services, Inc. 2021. AWS Glue FAQs - Managed ETL Service - Amazon Web Services. [online] Available at: <https://aws.amazon.com/glue/faqs/> [Accessed 24 July 2021].

[76] Docs.microsoft.com. 2021. Pipelines and activities in Azure Data Factory - Azure Data Factory. [online] Available at: <https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities> [Accessed 24 July 2021].

[77] Amazon Web Services, Inc. 2021. AWS Lambda – Product Features. [online] Available at: <https://aws.amazon.com/lambda/features/> [Accessed 24 July 2021].

[78] Google Cloud. 2021. Cloud Functions  |  Google Cloud. [online] Available at: <https://cloud.google.com/functions> [Accessed 24 July 2021].

[79] Amazon Web Services, Inc. 2021. Amazon SageMaker – Machine Learning – Amazon Web Services. [online] Available at: <https://aws.amazon.com/sagemaker/> [Accessed 15 July 2021].

[80] Azure.microsoft.com. 2021. AI Platform | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/overview/ai-platform> [Accessed 15 July 2021].

[81] Google Cloud. 2021. Vertex AI documentation  |  Google Cloud. [online] Available at: <https://cloud.google.com/vertex-ai/docs> [Accessed 15 July 2021].

[82] Amazon Web Services, Inc. 2021. Amazon SageMaker Pricing – Amazon Web Services (AWS). [online] Available at: <https://aws.amazon.com/sagemaker/pricing/> [Accessed 23 July 2021].

[83] Azure.microsoft.com. 2021. Pricing – Machine Learning | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/pricing/details/machine-learning/> [Accessed 23 July 2021].

[84] Google Cloud. 2021. Pricing  |  Vertex AI  |  Google Cloud. [online] Available at: <https://cloud.google.com/vertex-ai/pricing> [Accessed 23 July 2021].

[85] Amazon Web Services, Inc. 2021. Amazon SageMaker Data Wrangler – Aggregate and Prepare Data for Machine Learning – Amazon Web Services. [online] Available at: <https://aws.amazon.com/sagemaker/data-wrangler/> [Accessed 23 July 2021].

[86] Azure.microsoft.com. 2021. Pricing - Machine Learning | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-us/pricing/details/machine-learning-studio/> [Accessed 23 July 2021].

[87] Azure.microsoft.com. 2021. Azure Machine Learning offers free-usage tier | Azure updates | Microsoft Azure. [online] Available at: <https://azure.microsoft.com/en-gb/updates/azure-machine-learning-now-offers-free-usage-tier/> [Accessed 23 July 2021].

[88] Google Cloud. 2021. Introduction to Vertex ML Metadata  |  Vertex AI  |  Google Cloud. [online] Available at: <https://cloud.google.com/vertex-ai/docs/ml-metadata/introduction> [Accessed 23 July 2021].

[89] Google Cloud. 2021. Get started with Vertex AI TensorBoard  |  Google Cloud. [online] Available at: <https://cloud.google.com/vertex-ai/docs/experiments/tensorboard-overview> [Accessed 23 July 2021].

[90] Google Cloud. 2021. Pricing  |  Cloud Vision API  |  Google Cloud. [online] Available at: <https://cloud.google.com/vision/pricing> [Accessed 23 July 2021].

[91] Chernyshev, M., Baig, Z., Bello, O. and Zeadally, S., 2018. Internet of Things (IoT): Research, Simulators, and Testbeds. IEEE Internet of Things Journal, [online] 5(3), pp.1637-1647. Available at: <https://ieeexplore.ieee.org/abstract/document/8234579> [Accessed 20 July 2021].

[92] Nodered.org. 2021. Node-RED. [online] Available at: <https://nodered.org/> [Accessed 24 July 2021].

[93] Noflojs.org. 2021. NoFlo | Flow-Based Programming for JavaScript. [online] Available at: <https://noflojs.org/> [Accessed 24 July 2021].

[94] ThingsBoard. 2021. ThingsBoard - Open-source IoT Platform. [online] Available at: <https://thingsboard.io/> [Accessed 24 July 2021].

[95] Al-Fuqaha, A., 2021. [online] Ala.cs.wmich.edu. Available at: <https://ala.cs.wmich.edu/Fall11/cs6570/lectures/Bluetooth-ZigBee.pdf> [Accessed 16 July 2021].

[96] Aqeel-ur-Rehman, Mehmood, K. and Baksh, A., 2013. Communication Technology That Suits IoT - A Critical Review. Communications in Computer and Information Science, [online] pp.14-25. Available at: <https://link.springer.com/content/pdf/10.1007%2F978-3-642-41054-3\_2.pdf> [Accessed 15 July 2021].

[97] Bluetooth® Technology Website. 2021. Bluetooth Technology Overview | Bluetooth® Technology Website. [online] Available at: <https://www.bluetooth.com/learn-about-bluetooth/tech-overview/> [Accessed 15 July 2021].

[98] Zigbee Alliance. 2021. Zigbee - Zigbee Alliance. [online] Available at: <https://zigbeealliance.org/solution/zigbee/> [Accessed 15 July 2021].

[99] Bluetooth® Technology Website. 2021. Bluetooth MeshSecurity Overview | Bluetooth® Technology Website. [online] Available at: <https://www.bluetooth.com/blog/bluetooth-mesh-security-overview/> [Accessed 15 July 2021].

[100] Hong, H., Kim, Y. and Kim, R., 2018. A Low-Power WLAN Communication Scheme for IoT WLAN Devices Using Wake-Up Receivers. Applied Sciences, [online] 8(1), p.72. Available at: <https://www.mdpi.com/2076-3417/8/1/72/htm> [Accessed 15 July 2021].

[101] Morin, E., Maman, M., Guizzetti, R. and Duda, A., 2017. Comparison of the Device Lifetime in Wireless Networks for the Internet of Things. IEEE Access, [online] 5, pp.7097-7114. Available at: <https://ieeexplore.ieee.org/abstract/document/7894201> [Accessed 14 July 2021].

[102] Sponås, J., 2021. Things You Should Know About Bluetooth Range. [online] Blog.nordicsemi.com. Available at: <https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range> [Accessed 14 July 2021].

[103] Kim, H., Lee, J. and Jang, J., 2015. BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices. 2015 3rd International Conference on Future Internet of Things and Cloud, [online] Available at: <https://ieeexplore.ieee.org/abstract/document/7300867> [Accessed 14 July 2021].

[104] Omar Al Kalaa, M., Balid, W., Bitar, N. and Refai, H., 2016. Evaluating Bluetooth Low Energy in realistic wireless environments. 2016 IEEE Wireless Communications and Networking Conference, [online] Available at: <https://ieeexplore.ieee.org/abstract/document/7564809> [Accessed 14 July 2021].

[105] Google Cloud Blog. 2021. Using Node-RED with Google Cloud | Google Cloud Blog. [online] Available at: <https://cloud.google.com/blog/products/application-development/using-node-red-with-google-cloud> [Accessed 1 September 2021].

[106] Sanchez-Iborra, R. and Cano, M., 2016. State of the Art in LP-WAN Solutions for Industrial IoT Services. Sensors, [online] 16(5), p.708. Available at: <https://www.mdpi.com/1424-8220/16/5/708/htm> [Accessed 1 September 2021].

[107] LoRa Alliance®. 2021. What is LoRaWAN® Specification - LoRa Alliance®. [online] Available at: <https://lora-alliance.org/about-lorawan/> [Accessed 1 September 2021].

# Appendices

## Source codes

|  |  |
| --- | --- |
| Type | Python Program |
| **Name** | Labeller.py |
| **Programming Language** | Python 3.8 |
| **Requirements file** | pandas |
| **Code** | from os import listdir  from os.path import isfile, join  import pandas as pd  crackpath = "C:/Users/memoc/pool\_cracks"  noncrackpath = "C:/Users/memoc/pool\_noncracks"  getCracks = ['gs://gcu-dissertation/Model DS/Lake/'+f for f in listdir(crackpath) if isfile(join(crackpath, f))]  getNonCracks = ['gs://gcu-dissertation/Model DS/Lake/'+f for f in listdir(noncrackpath) if isfile(join(noncrackpath, f))]  cr = pd.DataFrame(data=getCracks,columns=['file'])  nc = pd.DataFrame(data=getNonCracks,columns=['file'])  cr['label'] = 'crack'  nc['label'] = 'no\_crack'  final = pd.DataFrame.append(cr,nc)  final.to\_csv(index=False) |

Table 19: Python Labeller Program

|  |  |
| --- | --- |
| Type | Python Program |
| **Name** | VertexAIEvals.py |
| **Programming Language** | Python 3.8 |
| **Requirements file** | google-cloud-aiplatform |
| **Code** | from google.cloud import aiplatform  project\_id = 'gcu-dissertation'  model\_id = '570426632490188800'  location = 'europe-west4'  parent\_eval = 'projects/'+project\_id+'/locations/'+location+'/models/'+model\_id  api\_endpoint = location +'-aiplatform.googleapis.com'  client\_args = { "api\_endpoint" : api\_endpoint}  client = aiplatform.gapic.ModelServiceClient(client\_options=client\_args)  evaluations = client.list\_model\_evaluations(parent=parent\_eval)  print(evaluations) |

Table 20: Python Code to retrieve Vertex AutoML evaluations raw data

|  |  |
| --- | --- |
| Type | Cloud Function program |
| **Name** | classification\_pipeline |
| **Programming Language** | Python 3.8 |
| **Requirements file** | # Function dependencies, for example:  # package>=version  google-cloud-firestore  google-cloud-storage  google-cloud-aiplatform |
| **Code** | from google.cloud import storage  from google.cloud import firestore  from google.cloud import aiplatform  from google.cloud.aiplatform.gapic.schema import predict as pr  import tempfile  import base64  import os  import datetime  def predict(event, context):    os.environ['GRPC\_DNS\_RESOLVER'] = 'native'  #modify the endpoint depending on the endpoint values in Vertex    endpoint\_location = 'europe-west4'    endpoint\_id = '3478133510655442944'    endpoint\_opt = {"api\_endpoint":'europe-west4-aiplatform.googleapis.com'}#set the endpooint where the model is deployed    model\_client = aiplatform.gapic.PredictionServiceClient(client\_options=endpoint\_opt)#get the AI client    storage\_client = storage.Client()    blob = storage\_client.bucket(event['bucket']).get\_blob(event['name'])    img\_uri = f"gs:L//{event['bucket']}/{event['name']}"    trsh,temp\_local\_filename = tempfile.mkstemp()    blob.download\_to\_filename(temp\_local\_filename)#local temporary storage for the image    try:      with open(temp\_local\_filename,"rb") as f:        fl = f.read()      encode\_content = base64.b64encode(fl).decode("utf-8")#encode the image and send it to predict    except Exception as e:      print(f"An error occurred while opening the image: {e}")    try:      instance = pr.instance.ImageClassificationPredictionInstance(        content = encode\_content,      ).to\_value()      instances = [instance]      parameters = pr.params.ImageClassificationPredictionParams(            confidence\_threshold=0.5, max\_predictions=5,      ).to\_value()      endpoint = model\_client.endpoint\_path(project=event['bucket'],location=endpoint\_location,endpoint=endpoint\_id)      response = model\_client.predict(endpoint=endpoint,instances=instances,parameters=parameters)      predictions = response.predictions      for prediction in predictions:        insert\_firestore(event['bucket'],event['name'],dict(prediction))    except Exception as e:      print(f"An error occurred while trying to predict: {e}")    def insert\_firestore(project,fn,prediction):    confidence = prediction['confidences'][0]    crack = True if prediction['displayNames'][0] == 'crack' else False    data\_values = fn.split('/')    data\_values = data\_values[1].split('-')    registry =  data\_values[0]    device = data\_values[1]    time = datetime.datetime.now()    client = firestore.Client(project=project)    doc\_ref = client.collection(u'IoT\_ImageList').document().set({      u'registry':registry,      u'device':device,      u'filename':fn,      u'time':time,      u'prediction':{        u'crack':crack,        u'confidence':confidence      }    }) |

Table 21: Image Classification Cloud Function Source Code and Requirements

|  |  |
| --- | --- |
| Type | Cloud Function program |
| **Name** | IoT\_response |
| **Programming Language** | Python 3.7 |
| **Requirements file** | # Function dependencies, for example:  # package>=version  google-cloud-storage |
| **Code** | import base64  from google.cloud import storage  import datetime  def save\_data(event, context):       try:            pubsub\_message = base64.b64decode(event['data'])            devId = event['attributes']['deviceId']            regId = event['attributes']['deviceRegistryId']            regLoc = event['attributes']['deviceRegistryLocation']            client = storage.Client()            bucket = client.bucket('gcu-dissertation')            fn,tme = file\_name(regId,devId)            target\_blob = bucket.blob(fn)            target\_blob.upload\_from\_string(pubsub\_message,content\_type='image/jpeg')            print(f"Finished uploading {fn}")       except Exception as e:            print(f"An error ocurred while uploading the image: {e}")  def file\_name(registry,device):       directory = "IoT\_imgs/"       tme =  str(datetime.datetime.now())       starttime = tme.replace('-','')       starttime = starttime.replace(' ','')       starttime = starttime.replace(':','')       starttime = starttime.split('.',1)[0]       name = registry + '-' + device + '-'       fn = directory+name+starttime+".jpg"       return fn,tme |

Table 22: IoT Pub/Sub Pre-process Cloud Function Source Code and Requirements (registry 1)

|  |  |
| --- | --- |
| Type | Cloud Function program |
| **Name** | IoT\_response\_region2 |
| **Programming Language** | Python 3.7 |
| **Requirements file** | # Function dependencies, for example:  # package>=version  google-cloud-storage |
| **Code** | import base64  from google.cloud import storage  import datetime  def save\_data(event, context):       try:            pubsub\_message = base64.b64decode(event['data'])            devId = event['attributes']['deviceId']            regId = event['attributes']['deviceRegistryId']            regLoc = event['attributes']['deviceRegistryLocation']            client = storage.Client()            bucket = client.bucket('gcu-dissertation')            fn,tme = file\_name(regId,devId)            target\_blob = bucket.blob(fn)            target\_blob.upload\_from\_string(pubsub\_message,content\_type='image/jpeg')            print(f"Finished uploading {fn}")       except Exception as e:            print(f"An error ocurred while uploading the image: {e}")  def file\_name(registry,device):       directory = "IoT\_imgs/"       tme =  str(datetime.datetime.now())       starttime = tme.replace('-','')       starttime = starttime.replace(' ','')       starttime = starttime.replace(':','')       starttime = starttime.split('.',1)[0]       name = registry + '-' + device + '-'       fn = directory+name+starttime+".jpg"       return fn,tme |

Table 23: IoT Pub/Sub Pre-process Cloud Function Source Code and Requirements (registry 2)

|  |  |
| --- | --- |
| Type | App Engine File Structure |
| **Structure** | \*Dissertation (root directory)  \*\*templates  \*\*\*Analytics.html  \*\*\*DeviceInfoNotAvailable.html  \*\*\*Devices.html  \*\*\*Images.html  \*\*\*RegistryCards.html  \*\*\*render\_image.html  \*\*app.yaml  \*\*data\_controller.py  \*\*iot\_controller.py  \*\*main\_test.py  \*\*main.py  \*\*requirements.txt  \*\*storage\_controller.py |

|  |  |
| --- | --- |
| Type | App Engine Program |
| **Name** | main.py |
| **Programming Language** | Python 3.8 |
| **Requirements file** | Flask==2.0.1  cryptography==3.4.7  google-api-python-client  google-auth==2.0.2  google-auth-httplib2==0.1.0  google-cloud-pubsub==2.7.0  google-cloud-iot==2.2.1  google-cloud-firestore  google-cloud-storage  google-cloud-aiplatform  requests==2.26.0  pillow-8.4.0  pandas  numpy |
| **Code** | [START gae\_python38\_app]  # [START gae\_python3\_app]  from flask import Flask,render\_template,request  #misc libraries  import base64 #needed to decode msgs  import json  import os  import pandas as pd  import numpy as np  import plotly  import plotly.express as px  import datetime  #google libraries  from google.cloud import pubsub\_v1  from google.oauth2 import id\_token  from google.auth.transport import requests  #from google.cloud import iot\_v1  #own libraries  from iot\_controller import devices  from data\_controller import IoT\_Data  from storage\_controller import Storage\_Central  app = Flask(\_\_name\_\_)  edge = devices()  idata = IoT\_Data()  imagec = Storage\_Central()  @app.route('/')  def main():  df = idata.getAnalytics(edge.list\_registires())  #prepare some tags for data search  registries = df['registry'].unique()  devices = df['device'].unique()      fig = px.bar(df,x='device',y='count',color='crack',barmode='group',title="Detections on devices")  dev\_graph = json.dumps(fig,cls=plotly.utils.PlotlyJSONEncoder)    fig = px.bar(df,x='registry',y='count',color='crack',barmode='group',title="Detections on registries")  reg\_graph = json.dumps(fig,cls=plotly.utils.PlotlyJSONEncoder)  range\_end = datetime.date.today()  range\_start = range\_end - datetime.timedelta(days=3)    date\_summary = df[df['crack'] == True].groupby(['time']).agg('count').drop(['device','filename','registry','confidence','crack'],axis=1).reset\_index()  print(date\_summary)  fig = px.line(date\_summary,x='time',y='count',hover\_data={'time':"|%B %d %Y, %H:%M"},range\_x=[str(range\_start),str(range\_end)],title="Cracks detected (T-3 days)")  date\_graph = json.dumps(fig,cls=plotly.utils.PlotlyJSONEncoder)  stats = df.describe().to\_html(table\_id="statistics")  df\_table = df.to\_html(table\_id='full\_ds')  return render\_template("Analytics.html",full\_tb=df\_table,statTable=stats,regGraph=reg\_graph,devGraph=dev\_graph,dateGraph=date\_graph)  @app.route('/callback',methods=['POST','GET'])  def cb():  return gm(request.args.get('data'))  def gm(test='123'):  #fig = px.line([1,2,3],x='123')  #graphJSON = json.dumps(fig,cls=plotly.utils.PlotlyJSONEncoder)  #return graphJSON  return test  @app.route('/deviceStates')  def getDeviceStatus():  list\_devices = edge.device\_states()  return list\_devices  @app.route('/image',methods=['GET'])  def getImage():  img = request.args.get('imageName')  imgs = imagec.getImage(img)  if imgs is not None:  dict\_image = {"image":imgs}  return json.dumps(dict\_image)  #return render\_template("render\_image.html",image=imgs)  else:  return error("Image not found")  @app.route("/queryImage",methods=['GET'])  def queryImage():  query = request.args.get('query')  value = request.args.get('value')  results = {}  if query == 'date':  lower = value.split("|")[0]  upper = value.split("|")[1]  results = idata.getByDate(lower,upper,limit=99)  elif query == 'registry':  results = idata.getAllByRegistry(value,limit=99)  elif query == "device":  results = idata.getAllByDevice(value,limit=99)  json\_imgs = pd.DataFrame(list(map(lambda x: x.to\_dict(),results))).to\_json(orient="index")    return json\_imgs  @app.route('/device')  def getDeviceState():  df = None  df\_table = None  images = None  registry = request.args.get('registryId')  device = request.args.get('deviceId')  edge.set\_registry(registry)  dv = edge.set\_device(device)  if dv is not None:  dev\_info = edge.get\_device\_info(device)  if len(dev\_info) > 0:  df = idata.getAnalyticsDevice(device)  if type(df) is not type(None):  df\_table = df.sort\_values(by=['time'],ascending=False).drop(['registry','count','device'],axis=1).to\_html(table\_id='ds\_device',classes=["hover","table-striped","table-bordered"])  images = df.sort\_values(by=['time'],ascending=False)[0:16].drop(['registry','count','device','confidence'],axis=1).to\_json(orient="index")  images = json.dumps(images)  print(dev\_info)  return render\_template("Devices.html",registry=registry,device=json.dumps(dev\_info),ds\_imgs=df\_table,img\_list=images)  else:  return error("Device not found")  @app.route('/listDevs')  def renderDevices():  return render\_template('RegistryCards.html')  @app.route('/command', methods=['POST'])  def sendCommand():  reg = request.form.get('RegistryID')  dev = request.form.get('DeviceID')  msg = request.form.get('Command')  print(reg)  print(dev)  response = { "Response" : edge.send\_command(reg,dev,msg) }  print(response)  return json.dumps(response)  @app.route('/devices')  def getDevices():  registry = request.args.get('Registry')  if registry != None:  edge.set\_registry(registry)  devices = edge.list\_devices()  return devices  @app.route('/registries')  def getRegistries():  registries = edge.list\_registires()  return registries  @app.route('/tracking')  def getTracking():  return "Tracking section"  @app.route('/imageList',methods=['GET'])  def images():    return render\_template("Images.html")  @app.route('/imgs')  def getImageList():  tst = idata.getAllByDevice('Device2')  return str(len(tst))  @app.errorhandler(500)  def error(e):  return f"Ooops...gotta er...fix that...{e}"  if \_\_name\_\_ == '\_\_main\_\_':  app.run(host='127.0.0.1', port=8080, debug=True)  # [END gae\_python3\_app]  # [END gae\_python38\_app] |

Table 24: Main Application App Engine Program Source Code and Requirements

|  |  |
| --- | --- |
| Type | App Engine Program |
| **Name** | iot\_controller.py |
| **Programming Language** | Python 3.8 |
| **Requirements file** | google-cloud-pubsub==2.7.0  google-cloud-iot==2.2.1 |
| **Code** | from google.cloud import iot\_v1  from google.cloud import pubsub  from googleapiclient.errors import HttpError  import datetime  import json  class devices():      client = iot\_v1.DeviceManagerClient()      devices = []      device = None      project\_id = "gcu-dissertation"      cloud\_region = "europe-west1"      registry = "dissertation\_test"      private\_key\_file = ""#location of private key for MQTT      jwt = None      def \_\_init\_\_(self):          print("started")        def send\_command(self,registry,device,command):          if self.device is not None:              path = self.client.device\_path(self.project\_id,self.cloud\_region,registry,device)              encoded\_msg = command.encode("utf-8")              try:                  response = self.client.send\_command\_to\_device(                      request = {"name":path,"binary\_data":encoded\_msg}                  )                  if response == "":                      return True                  else:                      return False              except Exception as e:                  print(f"Error: {e}")                  return False          else:              return False      def get\_device\_info(self,id):          info = {}          print(self.device)          if self.device.id == id:              info["id"] = self.device.id              info["name"] = self.device.name              info["last\_heartbeat"] = self.device.last\_heartbeat\_time.timestamp() if self.device.last\_heartbeat\_time is not None else 0              info["last\_error\_time"] = self.device.last\_error\_time.timestamp() if self.device.last\_error\_time is not None else 0              info["last\_error"] = self.device.last\_error\_status.message            return info      def device\_states(self):          statuses = dict()          for device in self.devices:              info = dict()              path = self.client.device\_path(self.project\_id,self.cloud\_region,self.registry,device.id)              status = self.client.get\_device(request={"name":path})              info["name"] = status.name              info["num\_id"] = status.num\_id              info["last\_heartbeat"] = status.last\_heartbeat\_time.timestamp()              info["last\_event"] = status.last\_event\_time.timestamp()              info["last\_error\_time"] = status.last\_error\_time.timestamp()              info["last\_error"] = status.last\_error\_status.message              statuses[device.id] = info              print(statuses)          js = json.dumps(statuses)          return js      def set\_device(self,device):          try:              device\_path = self.client.device\_path(self.project\_id,self.cloud\_region,self.registry,device)              field\_mask = gp\_field\_mask.FieldMask(                  paths=[                      "id",                      "name",                      "num\_id",                      "last\_heartbeat\_time",                      "last\_event\_time",                      "last\_state\_time",                      "last\_config\_ack\_time",                      "last\_config\_send\_time",                      "blocked",                      "last\_error\_time",                      "last\_error\_status",                      "config",                      "state",                      "log\_level",                      "metadata",                      "gateway\_config",                  ]              )              device = self.client.get\_device(request={"name":device\_path,"field\_mask":field\_mask})              self.device = device              return device          except Exception as e:              print(e)              return None      def list\_registires(self):          parent = 'projects/' + self.project\_id + '/locations/' + self.cloud\_region          registries = {}          for idx,registry in enumerate(self.client.list\_device\_registries(parent=parent)):              reg = {}              reg['id'] = registry.id              reg['name'] = registry.name              registries[idx] = reg            js = json.dumps(registries)          return js      def list\_devices(self):          parent = self.client.registry\_path(self.project\_id,self.cloud\_region,self.registry)          devices = {}            for idx,device in enumerate(self.client.list\_devices(parent=parent)):              dev = {}              dev['id'] = device.id              dev['numid'] = device.num\_id              devices[idx] = dev          js = json.dumps(devices)          return js      def set\_registry(self,registry\_id):          self.registry = registry\_id      def get\_registry(self):          return self.registry |

Table 25: IoT Controller App Engine Program Source Code and Requirements

|  |  |
| --- | --- |
| Type | App Engine Program |
| **Name** | data\_controller.py |
| **Programming Language** | Python 3.8 |
| **Requirements file** | google-cloud-firestore  pandas |
| **Code** | from google.cloud import firestore  import json  import pandas as pd  from datetime import datetime  class IoT\_Data():      client = None      collection = None      project\_id = "gcu-dissertation"        jwt = None      def \_\_init\_\_(self):          self.client = firestore.Client(project=self.project\_id)          self.collection = self.client.collection(u"IoT\_ImageList")      def getByDate(self,lower\_date,upper\_date,limit=1):          lower\_date = datetime.fromtimestamp(int(lower\_date))          upper\_date = datetime.fromtimestamp(int(upper\_date))            docs = self.collection.where(u"time",u">=",lower\_date).where(u"time",u"<=",upper\_date).order\_by(u"time",direction=firestore.Query.DESCENDING).limit(limit).stream()          result = list(docs)          print(len(result))          return result      def getAllByDevice(self,device,limit=None):          if device is None:              return None          else:              docs = self.collection.where(u'device',u'==',device).stream()              result = list(docs)              return result[0:limit]      def getAllByRegistry(self,registry,limit=None):          if registry is None:              return None          else:              docs = self.collection.where(u'registry',u'==',registry).stream()              result = list(docs)              return result[0:limit]      def getAnalyticsDevice(self,device):          query\_result = None          docs = self.getAllByDevice(device)          if len(docs) > 0:              df = pd.DataFrame(list(map(lambda x: x.to\_dict(),docs)))              query\_result = pd.concat([df.drop(['prediction'],axis=1),df['prediction'].apply(pd.Series)],axis=1)              query\_result['count'] = 1              query\_result['time'] = pd.to\_datetime(query\_result['time'])              query\_result['time'] = query\_result['time'].dt.floor('H')            return query\_result      def getAnalytics(self,registries):          query\_result = pd.DataFrame(columns=['registry', 'device', 'time', 'filename', 'crack', 'confidence'])          registries = json.loads(registries)          for ix in registries:              reg = registries[str(ix)]              docs = self.getAllByRegistry(reg['id'])              if len(docs) > 0:                  df = pd.DataFrame(list(map(lambda x: x.to\_dict(),docs)))                    if len(df) is not 0:                      temp = pd.concat([df.drop(['prediction'],axis=1),df['prediction'].apply(pd.Series)],axis=1)                      query\_result = pd.DataFrame.append(query\_result,temp)            query\_result['count'] = 1          query\_result['time'] = pd.to\_datetime(query\_result['time'])          query\_result['time'] = query\_result['time'].dt.floor('H')          return query\_result |

Table 26: Data Controller App Engine Program Source Code and Requirements

|  |  |
| --- | --- |
| Type | App Engine Program |
| **Name** | storage\_controller.py |
| **Programming Language** | Python 3.8 |
| **Requirements File** | google-cloud-storage  pillow-8.4.0 |
| **Code** | from google.cloud import storage  import tempfile  import os  import base64  from PIL import Image  import io  class Storage\_Central():      client = None      image = None      project\_id = "gcu-dissertation"      bucket = "gcu-dissertation"      def \_\_init\_\_(self):          self.client = storage.Client(project=self.project\_id)        def getImage(self,path):          \_,temp = tempfile.mkstemp()          try:              blob = self.client.bucket(self.bucket).get\_blob(path).download\_as\_string()              bytess = io.BytesIO(blob)              base = base64.b64encode(bytess.read()).decode("utf-8")              return base          except Exception as e:              print(e)              return None |

Table 27: Storage Controller App Engine Program Source Code and Requirements

|  |  |
| --- | --- |
| Type | App Engine Template |
| **Name** | Analytics.html |
| **Programming Language** | HTML  Javascript  Python (Flask) |
| **Requirements** | JQuery 3.5.1  Datatables 1.11.3  Bootstrap 4.0  Popper 1.12.9 |
| **Code** | <!doctype html>  <html>      <head>          <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">          <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.11.3/css/jquery.dataTables.min.css">          <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>          <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>          <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>          <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>          <script src="https://cdn.datatables.net/1.11.3/js/jquery.dataTables.min.js"></script>      </head>      <body>          <div class='container'>              <div class = "row p-2">                  <div class="col"><a href="/">Analytics Dashboard</a></div>                  <div class="col"><a href="/listDevs">Registries & Devices</a></div>                  <div class="col"><a href="/imageList">Image Viewer</a></div>              </div>              <div class="row my-3">                  <div class ="col-sm-auto" id="list\_results">                      <h2>Analytics Dashboard</h2>                  </div>              </div>              <div class="row">                  <div class ="col-10" id="list\_results">                      <h4>General Summary</h4>                  </div>              </div>              <div class="row">                  <div class ="col-10" id="list\_results">                      <p><h6>Dataset</h6></p>                      <p>                      {{ full\_tb | safe }}                      </p>                  </div>              </div>              <div class="row">                  <div class ="col-sm" id="analytics\_data">                      <p><h6>Dataset Statistics</h6></p>                      <p>                      {{ statTable | safe }}                      </p>                  </div>                  <div class ="col-7" id="date\_graph">                      <div id="dateGraph" class="dateGraph"></div>                  </div>              </div>              <div class="row">                  <div class ="col-10" id="list\_results">                      <h4>Data by Registry</h4>                  </div>              </div>              <div class="row">                  <div class ="col-md" id="registry\_graph">                      <div id="regGraph" class="regGraph"></div>                  </div>              </div>              <div class="row">                  <div class ="col-10" id="list\_results">                      <h4>Data by Devices</h4>                  </div>              </div>              <div class="row">                  <div class ="col-md" id="device\_graph">                      <div id="devGraph" class="devGraph"></div>                  </div>              </div>          </div>      </body>      <script type='text/javascript'>          var regGraph = {{ regGraph | safe }};          var devGraph = {{ devGraph | safe }};          var dateGraph = {{ dateGraph | safe }};            Plotly.plot('regGraph', regGraph, {});          Plotly.plot('devGraph', devGraph, {});          Plotly.plot('dateGraph', dateGraph, {});          $(document).ready(function(){              $('#statistics').DataTable(                  {                      searching:false,                      paging:false,                      info:false,                  }              );          })          $(document).ready(function(){              $('#full\_ds').DataTable(              );          })      </script>  </html> |

Table 28: Analytics App Engine HTML Template Source Code

|  |  |
| --- | --- |
| Type | App Engine Template |
| **Name** | Devices.html |
| **Programming Language** | HTML  Javascript  Python (Flask) |
| **Requirements** | JQuery 3.5.1  Datatables 1.11.3  Bootstrap 4.0  Popper 1.12.9 |
| **Code** | <!doctype html>  <html>  <head>      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">      <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.11.3/css/jquery.dataTables.min.css">      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>      <script src="https://cdn.datatables.net/1.11.3/js/jquery.dataTables.min.js"></script>  </head>  <body>      <div class='container'>          <div class = "row p-2">              <div class="col"><a href="/">Analytics Dashboard</a></div>              <div class="col"><a href="/listDevs">Registries & Devices</a></div>              <div class="col"><a href="/imageList">Image Viewer</a></div>          </div>          <div class="row my-3">              <div class ="col-sm-auto" id="list\_results">                  <h2>Device Dashboard</h2>              </div>          </div>          <div class="row">              <div class="col-md-4">                  <div class="column" id="device">                  </div>                  <div class ="column" id="info">                      <div class="alert alert-danger" id="error">                          <button type="button" class="close" data-dismiss="alert">x</button>                          <strong>Something went wrong!</strong>                          <div id='ErrorMsg'></div>                       </div>                       <div class="alert alert-success" id="information">                          <button type="button" class="close" data-dismiss="alert">x</button>                          <div id='InfoMsg'></div>                       </div>                  </div>              </div>              <div class="col-md-7">                  <div class="column" id="images\_table">                      <p>                          {{ ds\_imgs | safe }}                      </p>                  </div>              </div>          </div>          <div class="row">            </div>          <div class="row">              <div class ="col-sm-auto" id="list\_results">                  {% if img\_list == none %}                  <h2>No images available to preview</h2>                  {% else %}                  <h2>Recent images preview</h2>                  {% endif %}              </div>          </div>          <div id="img\_container">            </div>      </div>  </body>  {% if device %}  <script>      var json = jQuery.parseJSON({{ device|tojson }});      const $device = $('#device');      var heartbeat = "Not available";      var etime = "Not available";      var le = "Not available";      let device\_id = json['id'];        if(json['last\_heartbeat'] > 0){          heartbeat = new Date(json['last\_heartbeat'] \* 1000).toUTCString();      }      if(json['last\_error\_time'] > 0){          etime = new Date(json['last\_error\_time'] \* 1000).toUTCString();      }      if(json['last\_error'] != ""){          le = json['last\_error'];      }            var inf = "<table class='table table-hover' id='tooltip'><caption>Device log</caption><tbody><tr class='table-primary'><th>Device ID</th><td>"+device\_id+"</td></tr>";        var seen = "<tr><th scope='row'>Last seen date</th><td>"+heartbeat+"</td></tr>"      var error\_time = "<tr><th scope='row'>Last error date</th><td>"+etime+"</td></tr>"      var last\_error = "<tr ><th scope='row'>Error msg.</th><td>"+le+"</td></tr></table>"        $device.append(inf+seen+error\_time+last\_error);      $device.append("<div class='form-group border p-1' id='commands'><label for='commands'><b>Command Center</b></label><input type='text' class='form-control form-control-sm' id='command' placeholder='Write a command and press enter'><small id='commandsHelp' class='form-text text-muted'>Write <b>commands</b> to list available commands.</small></div>");      $(document).ready(function(){              $('#ds\_device').DataTable({                  "order":[[2,"desc"]],                  "scrollY":"400px",                  "columnDefs":[                      {                          "targets":[0],                          "visible":false,                          "searchable":false                      }                  ]              }              );      });        $(document).ready(() => {          $("#error").hide();          $('#information').hide();          globalThis.temp = "";          const $imageContainer = $('#img\_container');          var json = jQuery.parseJSON({{ img\_list | safe }});          var i = 0;          var id2 = "a"+String(i);            $.each(json,function ( indx ){              var temp = "";              if(i % 4 === 0){                  id = i;                  var mcontainer = '<div class="py-1 row" id = "'+id+'">';                  $('#img\_container').append(mcontainer);              }              var fig\_div = '<div class="border border-dark col-sm-2 ml-1"><div class ="column " id="info"><figure class="figure">';                id2 = "a"+String(i);              var img\_src = "/image?imageName="+json[indx]['filename'];              img\_src = String(id2) + "+" + img\_src;              var string\_crack = "";              if(json[indx]['crack'] === true){                  string\_crack = "<div class='my-1 alert alert-danger'> FISSURE EVENT DETECTED </div>";                  fig\_div += '<div id="'+id2+'" class="bg-danger" style="width: 135px;">';              }else{                  fig\_div += '<div id="'+id2+'" style="width: 135px;">';              }                  //if(json[indx]['crack'] === true){                  fig\_div += '</div>';              //}132\*132              getValues(img\_src);              var timefound = new Date(json[indx]['time'] \* 1000);              fig\_div += '<figcaption class="figure-caption">'+string\_crack+'Image taken on '+timefound.toUTCString()+'</figcaption>';              fig\_div += '</figure></div></div>';                $('#'+id).append(fig\_div);              ++i;            })        });      function getValues(id\_in){          //console.log(a);          console.log(id\_in);          var id = id\_in.split('+')[0];          var img\_src = id\_in.split('+')[1];            $.getJSON({              url:img\_src,success:function(result){                    $.each(result,function(key,val){                      var fig\_div = '<img src="data:image/jpg;base64,'+ val +'" class="m-1 figure-img img-fluid rounded" alt="Placeholder">';                      $('#'+String(id)).append(fig\_div);                  });                }          })      }      $('#command').keyup(function(event){          if(event.keyCode === 13){              var reg = '{{ registry | safe }}';              var value = $("#command").val().toLowerCase();              switch(value){                  case "snapshot":                      $.post("/command",{"RegistryID":reg,"DeviceID":device\_id,"Command":value});                      $("#command").val("");                      setInformation("Picture command sent!",2000);                      break;                  case "alert":                      $.post("/command",{"RegistryID":reg,"DeviceID":device\_id,"Command":value});                      $("#command").val("");                      setInformation("Alert command sent!",2000);                      break;                  case "reset":                      $.post("/command",{"RegistryID":reg,"DeviceID":device\_id,"Command":value});                      $("#command").val("");                      setInformation("Reset command sent!",2000);                      break;                  case "commands":                      $("#command").val("");                      var commands = "The following commands are available for this device:<ul>";                      commands += "<li><b>Snapshot</b>: Take a new picture</li>";                      commands += "<li><b>Alert</b>: Send an alert</li>";                      commands += "<li><b>Interval-#</b>: Set automatic pictures interval to the set # (in int. seconds)</li>";                      commands += "<li><b>Reset</b>: Reset device setup</li>";                      commands += "<li><b>Commands</b>: List commands</li>";                      commands += "<ul>";                        setInformation(commands,5000);                      break;                  default:                      if(value.includes("interval")){                          $.post("/command",{"RegistryID":reg,"DeviceID":device\_id,"Command":value});                          $("#command").val("");                          setInformation("Alert command sent!",2000);                      }else{                          $("#command").val("");                          setError("This command does not exist. Enter <b>commands</b> to see the available commands.",3000)                      }                      break;              }          }      });      function setError(message,time){          $('#ErrorMsg').empty().append(message);              $('#error').fadeTo(time,500).slideUp(500,function(){                  $("#error").slideUp(500);          });      }      function setInformation(message,time){          $('#InfoMsg').empty().append(message);          $('#information').fadeTo(time,500).slideUp(500,function(){              $("#information").slideUp(500);          });      }    </script>  {% endif %} |

Table 29: Devices View App Engine HTML Template Source Code

|  |  |
| --- | --- |
| Type | App Engine Template |
| **Name** | Images.html |
| **Programming Language** | HTML  JavaScript  Python (Flask) |
| **Requirements** | JQuery 3.5.1  Datatables 1.11.3  Bootstrap 4.0  Popper 1.12.9 |
| **Code** | <!doctype html>  <html>  <head>      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">      <link href = "https://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel = "stylesheet">      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>      <script src = "https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>  </head>  <body>      <div class='container'>          <div class = "row p-2">              <div class="col"><a href="/">Analytics Dashboard</a></div>              <div class="col"><a href="/listDevs">Registries & Devices</a></div>              <div class="col"><a href="/imageList">Image Viewer</a></div>          </div>          <div class="row my-3">              <div class ="col-sm-auto">                  <h2>Image Viewer</h2>              </div>          </div>            <div class="row">              <div class ="rounded p-3 h-100 col-md-auto">                  <label for='date'><b>Search by date range</b></label>                  <div class='input-group input-daterange border p-1'>                      <input type='text' class='form-control' id='date1'>                      <div class="input-group-addon">to</div>                      <input type='text' class='form-control' id='date2'>                  </div>              </div>              <div class ="rounded p-3 h-100 col-md-auto">                  <label for='registry'><b>Search by Registry ID</b></label>                  <div class='form-group border p-1'>                      <select class="form-select" aria-label="Default select example" id="registry">                          <option selected value='9999'>Select a registry ID</option>                        </select>                  </div>              </div>              <div class ="rounded p-3 h-100 col-md-auto">                  <label for='device'><b>Search by device ID</b></label>                  <div class='form-group border p-1'>                      <input type='text' class='form-control form-control-sm' id='device' placeholder='Enter a device ID'>                  </div>              </div>          </div>          <div class="row" id='info\_main'>              <div class ="rounded p-3 bg-info col-md-auto text-white" id="info">                  Search by date, registry ID, or device ID to display results. The results are limited to the first 100 images.              </div>          </div>          <div id="img\_container">            </div>      </div>  </body>  <script>        $(document).ready(() => {          $( "#date1" ).datepicker(              { dateFormat: 'yy-mm-dd' }          );          $( "#date2" ).datepicker(              { dateFormat: 'yy-mm-dd' }          );          var regs = $("#registry");            $.getJSON('/registries', (registries) => {              $.each( registries, function( index ) {                  var fig\_list = '<option value="'+String(index)+'">' + registries[index]['id'] + '</option>';                  regs.append(fig\_list);              });          });          globalThis.temp = "";          });      $('#date1').on('change', function() {            var date2 = $("#date2").datepicker('getDate');          var date = $('#date1').datepicker('getDate');          $('#date2').datepicker("option", "minDate", date);          if(date !== null && date2 !== null){              $('#img\_container').empty();              var d1 = toTimestamp(date.getFullYear(),date.getMonth(),date.getDate());              var d2 = toTimestamp(date2.getFullYear(),date2.getMonth(),date2.getDate());              var dateRange = d1 + "|" + d2;              getImages("date",dateRange);          }          resetRegistry();          clearDev();      });      $('#date2').on('change', function() {          var date2 = $("#date2").datepicker('getDate');          var date = $('#date1').datepicker('getDate');          console.log(date.getTime());          $('#date1').datepicker("option", "maxDate", date2);          if(date !== null && date2 !== null){              $('#img\_container').empty();              var d1 = toTimestamp(date.getFullYear(),date.getMonth(),date.getDate());              var d2 = toTimestamp(date2.getFullYear(),date2.getMonth(),date2.getDate());              var dateRange = d1 + "|" + d2;              getImages("date",dateRange);          }          clearDev();          resetRegistry();      });      $('#registry').on('change', function() {          clearDev();          clearDates();          $('#img\_container').empty();          var selected = $("#registry option:selected").text();          getImages('registry',selected);      });      $('#device').keyup(function(event){          if(event.keyCode === 13){              $('#img\_container').empty();              clearDates();              resetRegistry();              getImages('device');          }      });      function resetRegistry(){          $("#registry").val('9999')//.change();      };      function clearDev(){          $("#device").val("");      };      function clearDates(){          $('#date1').datepicker("option", "maxDate", null);          $("#date1").val("");          $('#date2').datepicker("option", "minDate", null);          $("#date2").val("");      };      function getImages(queryType,value){          var args = '/queryImage?query='+queryType+'&value='+value;          getJson(args);      };      function getJson(value){          $.getJSON({              url:value, success:function(data){                  var i = 0;                  var id2 = "a"+String(i);                  $.each(data,function(key,val){                      var temp = "";                      if(i % 4 === 0){                          id = i;                          var mcontainer = '<div class="py-1 row" id = "'+id+'">';                          $('#img\_container').append(mcontainer);                      }                      var fig\_div = '<div class="border border-dark col-sm-2 ml-1"><div class ="column " id="info"><figure class="figure">';                        id2 = "a"+String(i);                      var img\_src = "/image?imageName="+val.filename;                      img\_src = String(id2) + "+" + img\_src;                      var crack = val.prediction.crack;                      var string\_crack = "";                      if(crack){                          string\_crack = "<div class='my-1 alert alert-danger'> FISSURE EVENT DETECTED </div>";                          fig\_div += '<div id="'+id2+'" class="bg-danger" style="width: 135px;">';                      }else{                          fig\_div += '<div id="'+id2+'" style="width: 135px;">';                      }                        fig\_div += '</div>';                      getValues(img\_src);                        var timefound = new Date(val.time \* 1000);                      var date = timefound.getDate();                      var month = timefound.getMonth();                      var year = timefound.getFullYear();                      var hours = timefound.getHours();                      // Minutes part from the timestamp                      var minutes = "0" + timefound.getMinutes();                      // Seconds part from the timestamp                      var seconds = "0" + timefound.getSeconds();                      // Will display time in 10:30:23 format                      var formattedTime = date + '-' + month + '-' + year + ' ' +hours + ':' + minutes.substr(-2) + ':' + seconds.substr(-2);                      fig\_div += '<figcaption class="figure-caption">'+string\_crack+'Image taken on '+val.time+' from device '+val.device+'</figcaption>';                      fig\_div += '</figure></div></div>';                        $('#'+id).append(fig\_div);                      ++i;                  });              }          });      };      function getValues(id\_in){          //console.log(a);          console.log(id\_in);          var id = id\_in.split('+')[0];          var img\_src = id\_in.split('+')[1];            $.getJSON({              url:img\_src,success:function(result){                    $.each(result,function(key,val){                      var fig\_div = '<img src="data:image/jpg;base64,'+ val +'" class="m-1 figure-img img-fluid rounded" alt="Placeholder">';                      $('#'+String(id)).append(fig\_div);                  });                }          })      };        function setError(message,time){          $('#ErrorMsg').empty().append(message);              $('#error').fadeTo(time,500).slideUp(500,function(){                  $("#error").slideUp(500);          });      }      function setInformation(message,time){          $('#InfoMsg').empty().append(message);          $('#information').fadeTo(time,500).slideUp(500,function(){              $("#information").slideUp(500);          });      }      function toTimestamp(y,m,d){          var datum = new Date(Date.UTC(y,m,d,0,0,0));          return datum.getTime()/1000;      }    </script> |

Table 30: Images View App Engine HTML Template Source Code

|  |  |
| --- | --- |
| Type | App Engine Template |
| **Name** | RegistryCards.html |
| **Programming Language** | HTML  JavaScript  Python (Flask) |
| **Requirements** | JQuery 3.5.1  Datatables 1.11.3  Bootstrap 4.0  Popper 1.12.9 |
| **Code** | <!doctype html>  <html>  <head>      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>      <script>          function cb(selection) {              $.getJSON({                  url: "/callback", data: { 'data': selection }, success: function (result) {                      Plotly.newPlot('chart', result, {});;                  }              });          };          function load\_devices(){              $.getJSON({                  url:"/devices",success:function(result){                      $.each(result,function(key,val){                          $("heya").append(val);                      });                  }              });          }        </script>  </head>  <body>      <div class='container'>          <div class = "row p-2">              <div class="col"><a href="/">Analytics Dashboard</a></div>              <div class="col"><a href="/listDevs">Registries & Devices</a></div>              <div class="col"><a href="/imageList">Image Viewer</a></div>          </div>          <div class="row my-3">              <div class ="col-sm-auto" id="list\_results">                  <h2>IoT Registries</h2>              </div>          </div>          <div class="row">              <div class="column">                  <div class="card-deck" id="registries">                    </div>              </div>          </div>      </div>  </body>  <script>      $(document).ready(() => {          const $registries = $('#registries');            $.getJSON('/registries', (data) => {              console.log(data);                $.each( data, function( index ) {                  var reg\_card = "<div class='card'><div class='card-body' id='"+ data[index].id +"'><h5 class='card-title'>Registry ID: " + data[index].id + "</h5></div></div>";                  $registries.append(reg\_card)                  var devices = '/devices?Registry=' + data[index].id;                    var itms = [];                  var crd = "#"+data[index].id;                  $.getJSON(devices,(dev) =>{                      itms.push('<p class="card-text"><ul>');                      console.log(dev);                      $.each(dev,function ( indx ){                          itms.push("<li><a href='/device?registryId=" + data[index].id + "&deviceId=" + dev[indx].id + "'>" + dev[indx].id +"</a></li>");                      })                      itms.push('</ul></p>');                      $(crd).append(itms);                  })              });            });        });  </script> |

Table 31: Registries and Devices View App Engine HTML Template Source Code

|  |  |
| --- | --- |
| Type | App Engine Template |
| **Name** | render\_image.html |
| **Programming Language** | HTML  Python (Flask) |
| **Requirements** | N/A |
| **Code** | <img id="imgTest" src="data:image/jpeg;base64,{{ image | safe }}" > |

Table 32: Image rendering function App Engine HTML Template Source Code

|  |  |
| --- | --- |
| Type | Node-RED Flow |
| Name | IoT V2 |
| Programming Language | N/A |
| Requirements File | N/A |
| Code | [  {  "id": "a2ebda5b93a62082",  "type": "tab",  "label": "IoT V2",  "disabled": false,  "info": "",  "env": []  },  {  "id": "772673be21a44f92",  "type": "inject",  "z": "a2ebda5b93a62082",  "name": "CallFlow",  "props": [  {  "p": "payload"  },  {  "p": "topic",  "vt": "str"  }  ],  "repeat": "1",  "crontab": "",  "once": true,  "onceDelay": "0",  "topic": "default",  "payload": "123",  "payloadType": "num",  "x": 140,  "y": 360,  "wires": [  [  "a03091b435fb0310"  ]  ]  },  {  "id": "418b81626116cddd",  "type": "google-cloud-iotcore device",  "z": "a2ebda5b93a62082",  "name": "GCP\_Send",  "qos": "0",  "retain": "",  "topic": "events",  "broker": "8270c187.71d4d",  "x": 1190,  "y": 240,  "wires": [  [],  []  ]  },  {  "id": "2663292d4a8315ef",  "type": "file in",  "z": "a2ebda5b93a62082",  "name": "GetFile",  "filename": "",  "format": "",  "chunk": false,  "sendError": false,  "encoding": "binary",  "x": 640,  "y": 220,  "wires": [  [  "2bf0c8f69ced067f"  ]  ]  },  {  "id": "8258485d63599da2",  "type": "function",  "z": "a2ebda5b93a62082",  "name": "SetRandomImage",  "func": "var file\_path = \"C:\\\\Users\\\\memoc\\\\Pictures\\\\IoT\\\\\";\nvar image\_limit = 40;\nvar file\_name = 0;\n\nfile\_name = Math.floor(Math.random() \* (image\_limit - 0)) + 0;\nfile\_path = file\_path.concat(file\_name.toString(),\".jpg\");\nmsg = {\n \"filename\":file\_path,\n \"payload\":file\_path\n}\n\nreturn msg;",  "outputs": 1,  "noerr": 0,  "initialize": "",  "finalize": "",  "libs": [],  "x": 450,  "y": 220,  "wires": [  [  "5c438ddcc9199afc",  "2663292d4a8315ef"  ]  ]  },  {  "id": "797bfae623f5ad47",  "type": "google-cloud-iotcore device",  "z": "a2ebda5b93a62082",  "name": "GCP\_Receive",  "qos": "1",  "retain": "",  "topic": "",  "broker": "8270c187.71d4d",  "x": 120,  "y": 500,  "wires": [  [],  [  "8c2dc6b7e0459fba"  ]  ]  },  {  "id": "2bf0c8f69ced067f",  "type": "jimp-image",  "z": "a2ebda5b93a62082",  "name": "",  "data": "payload",  "dataType": "msg",  "ret": "buf",  "parameter1": "128",  "parameter1Type": "num",  "parameter2": "128",  "parameter2Type": "num",  "parameter3": "",  "parameter3Type": "msg",  "parameter4": "",  "parameter4Type": "msg",  "parameter5": "",  "parameter5Type": "msg",  "parameter6": "",  "parameter6Type": "msg",  "parameter7": "",  "parameter7Type": "msg",  "parameter8": "",  "parameter8Type": "msg",  "sendProperty": "payload",  "sendPropertyType": "msg",  "parameterCount": 3,  "jimpFunction": "resize",  "selectedJimpFunction": {  "name": "resize",  "fn": "resize",  "description": "resize the image. One of the w or h parameters can be set to automatic (\"Jimp.AUTO\" or -1).",  "parameters": [  {  "name": "w",  "type": "num|auto",  "required": true,  "hint": "the width to resize the image to (or \"Jimp.AUTO\" or -1)"  },  {  "name": "h",  "type": "num|auto",  "required": true,  "hint": "the height to resize the image to (or \"Jimp.AUTO\" or -1)"  },  {  "name": "mode",  "type": "resizeMode",  "required": false,  "hint": "a scaling method (e.g. Jimp.RESIZE\_BEZIER)"  }  ]  },  "x": 810,  "y": 220,  "wires": [  [  "42cdfde678ebefef",  "bbf292bf2e22383d"  ]  ]  },  {  "id": "42cdfde678ebefef",  "type": "function",  "z": "a2ebda5b93a62082",  "name": "ByteArray",  "func": "\ntst = msg.payload.toString('hex',0,29619)\n\nmsg2 = {\n \"payload\":msg.payload,\n \"topic\":\"projects/gcu-dissertation/topics/test\"\n}\nreturn msg2;",  "outputs": 1,  "noerr": 0,  "initialize": "",  "finalize": "",  "libs": [],  "x": 980,  "y": 280,  "wires": [  [  "0011f624d0b84a71",  "418b81626116cddd"  ]  ]  },  {  "id": "0011f624d0b84a71",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "ByteArray msg",  "active": false,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 1200,  "y": 300,  "wires": []  },  {  "id": "bbf292bf2e22383d",  "type": "image viewer",  "z": "a2ebda5b93a62082",  "name": "",  "width": 160,  "data": "payload",  "dataType": "msg",  "active": false,  "x": 950,  "y": 140,  "wires": [  []  ]  },  {  "id": "8c2dc6b7e0459fba",  "type": "switch",  "z": "a2ebda5b93a62082",  "name": "",  "property": "payload",  "propertyType": "msg",  "rules": [  {  "t": "cont",  "v": "crack\_status",  "vt": "str"  },  {  "t": "eq",  "v": "alert",  "vt": "str"  },  {  "t": "eq",  "v": "snapshot",  "vt": "str"  },  {  "t": "eq",  "v": "reset",  "vt": "str"  },  {  "t": "cont",  "v": "interval",  "vt": "str"  }  ],  "checkall": "true",  "repair": false,  "outputs": 5,  "x": 330,  "y": 500,  "wires": [  [  "77f297f0e41642cc"  ],  [  "78e9dc805713f227"  ],  [  "4a8d949137be57df",  "5f1d4be90169475c"  ],  [  "7237770d66e7684c"  ],  [  "5896aac36ca04345"  ]  ]  },  {  "id": "4a8d949137be57df",  "type": "link out",  "z": "a2ebda5b93a62082",  "name": "",  "mode": "link",  "links": [  "81c16aef86964f19"  ],  "x": 515,  "y": 600,  "wires": []  },  {  "id": "81c16aef86964f19",  "type": "link in",  "z": "a2ebda5b93a62082",  "name": "",  "links": [  "4a8d949137be57df"  ],  "x": 295,  "y": 160,  "wires": [  [  "8258485d63599da2"  ]  ]  },  {  "id": "5c438ddcc9199afc",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "get\_image",  "active": false,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 650,  "y": 120,  "wires": []  },  {  "id": "ed2241bd608e0b5c",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "Command flow",  "info": "The flow below is the main 'input' flow where commands are received and processed from GCP into the IoT devices. Through this flow the user is able to control and have interactions with the IoT devices remotely through the GCP Dashboard.\n\nThe following commands are available:\n - crack\_status:identified = If a crack is identified by the AI in GCP, a command will be received to turn on a warning light on the Pi emulated device. It will turn off if the user sends switch\_light action (locally or remotely)\n - interval-time = This emulates an approach to configure the devices. The configuration is limited to set the time frequency to send images from the device. The configuration is global for the flow.\n - alert = turns on and off the onboard device light\n - snapshot:device = The system will take an image from the bucket based on the device ID. This allows the system to take the correct image from the device\n - reset = resets the configurations of the device to its initial values (lights will turn off and intervalometer is set back to 10 seconds)",  "x": 340,  "y": 420,  "wires": []  },  {  "id": "1bce8c53dd9a167f",  "type": "rpi-sensehatsim out",  "z": "a2ebda5b93a62082",  "name": "",  "x": 780,  "y": 440,  "wires": []  },  {  "id": "78e9dc805713f227",  "type": "function",  "z": "a2ebda5b93a62082",  "name": "Control lights",  "func": "var msg1 = {\n \"payload\":\"\*,\*,red\"\n};\n\nif(context.get(\"lights\") === undefined || context.get(\"lights\") === 'Off'){\n context.set(\"lights\",\"On\");\n}\n\n\nreturn msg1;",  "outputs": 1,  "noerr": 0,  "initialize": "",  "finalize": "",  "libs": [],  "x": 570,  "y": 520,  "wires": [  [  "1bce8c53dd9a167f",  "dd410d83b920adfb"  ]  ]  },  {  "id": "dd410d83b920adfb",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "Lights",  "active": false,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 770,  "y": 520,  "wires": []  },  {  "id": "77f297f0e41642cc",  "type": "function",  "z": "a2ebda5b93a62082",  "name": "Crack Identified",  "func": "var test = msg.payload.split(\":\",2)[1];\nvar msg1 = {};\nif(test === 'true'){\n msg1 = {\n \"payload\":\"\*,\*,red\"\n };\n \n if(context.get(\"lights\") === undefined || context.get(\"lights\") === 'Off'){\n context.set(\"lights\",\"On\");\n }\n \n}\nelse if(test === 'false'){\n msg1 = {\n \"payload\":\"\*,\*,off\"\n };\n if(context.get(\"lights\") === undefined || context.get(\"lights\") === 'On'){\n context.set(\"lights\",\"Off\");\n }\n}\n\nreturn msg1; ",  "outputs": 1,  "noerr": 0,  "initialize": "",  "finalize": "",  "x": 580,  "y": 440,  "wires": [  [  "9d8e90ed4a2ddf32",  "1bce8c53dd9a167f"  ]  ]  },  {  "id": "9d8e90ed4a2ddf32",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "identified",  "active": false,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 760,  "y": 400,  "wires": []  },  {  "id": "5896aac36ca04345",  "type": "function",  "z": "a2ebda5b93a62082",  "name": "interval",  "func": "\n//if(msg.topic === undefined){\n var value = msg.payload.split(\"-\",2)[1];\n context.set(\"interval\", value\*1000);\n msg.rate = value\*1000;\n//}\n//else{\n// var interval = context.get(\"interval\");\n // msg2.delay = interval; \n//}\n\nreturn msg;",  "outputs": 1,  "noerr": 0,  "initialize": "",  "finalize": "",  "libs": [],  "x": 560,  "y": 800,  "wires": [  [  "e2f75503674b1ee7",  "d8871b42cbc8c419"  ]  ]  },  {  "id": "e2f75503674b1ee7",  "type": "link out",  "z": "a2ebda5b93a62082",  "name": "",  "links": [  "3eeb0ed0ca097666"  ],  "x": 695,  "y": 800,  "wires": []  },  {  "id": "3eeb0ed0ca097666",  "type": "link in",  "z": "a2ebda5b93a62082",  "name": "",  "links": [  "e2f75503674b1ee7",  "a81518bae9933ee9"  ],  "x": 155,  "y": 300,  "wires": [  [  "a03091b435fb0310"  ]  ]  },  {  "id": "07e0796af7a3b291",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "vals",  "active": false,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 490,  "y": 340,  "wires": []  },  {  "id": "7237770d66e7684c",  "type": "function",  "z": "a2ebda5b93a62082",  "name": "reset",  "func": "context.set(\"interval\", 10\*1000);//reset interval to 10 seconds\nmsg.rate = 10\*1000;\nmsg.payload = \"\*,\*,off\";\nif(context.get(\"lights\") === \"On\"){//reset lights\n context.set(\"lights\",\"Off\");\n}\n\nreturn msg;",  "outputs": 1,  "noerr": 0,  "initialize": "",  "finalize": "",  "libs": [],  "x": 550,  "y": 720,  "wires": [  [  "a81518bae9933ee9",  "1bce8c53dd9a167f",  "b334990002d59f54"  ]  ]  },  {  "id": "a03091b435fb0310",  "type": "delay",  "z": "a2ebda5b93a62082",  "name": "Delay",  "pauseType": "timed",  "timeout": "10",  "timeoutUnits": "seconds",  "rate": "1",  "nbRateUnits": "10",  "rateUnits": "second",  "randomFirst": "1",  "randomLast": "5",  "randomUnits": "seconds",  "drop": true,  "allowrate": true,  "outputs": 1,  "x": 310,  "y": 340,  "wires": [  [  "8258485d63599da2",  "07e0796af7a3b291"  ]  ]  },  {  "id": "28af65ce1377e524",  "type": "inject",  "z": "a2ebda5b93a62082",  "name": "Interval",  "props": [  {  "p": "payload"  }  ],  "repeat": "",  "crontab": "",  "once": false,  "onceDelay": 0.1,  "topic": "",  "payload": "interval-5",  "payloadType": "str",  "x": 110,  "y": 800,  "wires": [  [  "8c2dc6b7e0459fba"  ]  ]  },  {  "id": "d8871b42cbc8c419",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "Set interval",  "active": false,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 750,  "y": 840,  "wires": []  },  {  "id": "a81518bae9933ee9",  "type": "link out",  "z": "a2ebda5b93a62082",  "name": "",  "mode": "link",  "links": [  "3eeb0ed0ca097666"  ],  "x": 695,  "y": 740,  "wires": []  },  {  "id": "ec23b68135a56574",  "type": "inject",  "z": "a2ebda5b93a62082",  "name": "Reset",  "props": [  {  "p": "payload"  }  ],  "repeat": "",  "crontab": "",  "once": false,  "onceDelay": 0.1,  "topic": "",  "payload": "reset",  "payloadType": "str",  "x": 130,  "y": 760,  "wires": [  [  "8c2dc6b7e0459fba"  ]  ]  },  {  "id": "7a890d5ebaffd0de",  "type": "inject",  "z": "a2ebda5b93a62082",  "name": "Snapshot",  "props": [  {  "p": "payload"  }  ],  "repeat": "",  "crontab": "",  "once": false,  "onceDelay": 0.1,  "topic": "",  "payload": "snapshot",  "payloadType": "str",  "x": 120,  "y": 720,  "wires": [  [  "8c2dc6b7e0459fba"  ]  ]  },  {  "id": "08d9d145a9974a9e",  "type": "inject",  "z": "a2ebda5b93a62082",  "name": "alert",  "props": [  {  "p": "payload"  }  ],  "repeat": "",  "crontab": "",  "once": false,  "onceDelay": 0.1,  "topic": "",  "payload": "alert",  "payloadType": "str",  "x": 130,  "y": 680,  "wires": [  [  "8c2dc6b7e0459fba"  ]  ]  },  {  "id": "d5b193336990826e",  "type": "inject",  "z": "a2ebda5b93a62082",  "name": "Crack identified",  "props": [  {  "p": "payload"  }  ],  "repeat": "",  "crontab": "",  "once": false,  "onceDelay": 0.1,  "topic": "",  "payload": "crack\_status:true",  "payloadType": "str",  "x": 100,  "y": 640,  "wires": [  [  "8c2dc6b7e0459fba"  ]  ]  },  {  "id": "bc41dbf71c6463b5",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "Manual override",  "info": "Manual override of commands (local test)",  "x": 110,  "y": 580,  "wires": []  },  {  "id": "a4b2d189c39490af",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "Snapshot flow",  "info": "Main flow to obtain picture from a local repository and send it through to GCP IoT",  "x": 290,  "y": 80,  "wires": []  },  {  "id": "293664d20ddda83e",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "Intervalometer",  "info": "Intervalometer that controls the rate at which images are taken from the snapshot flow. \nNote that this flow only controls the automatic flow and can be overwritten through a snapshot command, which isn't affected by the intervalometer.",  "x": 260,  "y": 240,  "wires": []  },  {  "id": "b334990002d59f54",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "Reset",  "active": true,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 790,  "y": 760,  "wires": []  },  {  "id": "5f1d4be90169475c",  "type": "debug",  "z": "a2ebda5b93a62082",  "name": "Snapshot",  "active": true,  "tosidebar": true,  "console": false,  "tostatus": false,  "complete": "true",  "targetType": "full",  "statusVal": "",  "statusType": "auto",  "x": 560,  "y": 640,  "wires": []  },  {  "id": "47ed65f89f146d1b",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "GetFile",  "info": "The GetFile node accesses the local machine and opens the target file set under the image function.\n\nIf the system finds an image in the local repository of the machine, the node sends the file to the JIMP node; otherwise, the node throws an error for missing files.",  "x": 640,  "y": 180,  "wires": []  },  {  "id": "28105af6742343e0",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "RandomImage",  "info": "The SetRandomImage function creates a random filename string using the local repository path and the image name.\n\nThe local images must be named with an integer number, starting from 0 with no upper limit. The function requires the number of images in the repository to determine the range of images in the repository.\n\n - To modify the directory where images are stored locally, modify the variable \_file\_path\_ to the location of the local repository.\n - To set the number of image files in the local system, modify variable \_image\_limit\_ to the number of local image files of the repository.",  "x": 460,  "y": 180,  "wires": []  },  {  "id": "e71ba7f836f809ae",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "resize",  "info": "The Resize node uses a JIMP node to modify the size of the images to transmit through to IoT core. The size limit of messages with GCP is 10mb and up to 10mb/s for streaming, so downsizing images is critic to successfully transmit periodic messages from Node-RED into GCP IoT Core.\n\nThe default image rezising values are 128x128.",  "x": 800,  "y": 180,  "wires": []  },  {  "id": "9ab122c392f16a95",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "Viewer",  "info": "A node that enables a local viewer of resized images for visualization, testing, and debug activities. It can be enabled or disabled by pressing the switch of the node.",  "x": 960,  "y": 100,  "wires": []  },  {  "id": "64663ba58415fa57",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "ByteArray",  "info": "Pub/Sub rejects image files as it accepts only string and numeric values. To overcome this challenge, the ByteArray function transforms the resized image into a ByteArray string to transmit it through Pub/Sub.",  "x": 980,  "y": 240,  "wires": []  },  {  "id": "bd33f614aac6cc76",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "GCP\_Send",  "info": "GCP\_Send node is the GCP IoT Core API that connects the broker communication between the flows and Pub/Sub.\n\nNOTE: For local use, its important to use the correct registry ID and device ID, as well as enable access to the corresponding key file (RS256 encoded certificate) to authenticate with the GCP Pub/Sub broker. If the values are not set correctly, or if the certificate does not correspond to the key of the device, the connection will be rejected and messages will not transmit.",  "x": 1190,  "y": 200,  "wires": []  },  {  "id": "997d4790d78f1ea5",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "CrackIdentified",  "info": "The CrackIdentified function is called when the switch receives a message \_crack\_status\_. When the message is processed and the function called, the system triggers the SenseHat emulator to turn on the onboard lights if the value is \*\*true\*\*. The onboard light indicates that a crack has been identified in the last image processed by the model. ",  "x": 580,  "y": 400,  "wires": []  },  {  "id": "d404428e2512c376",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "snapshot",  "info": "Snapshot commands enables a manual override of the Snapshot flow by processing the command \*\*snapshot\*\* sent from the cloud dashboard to trigger the flow. This command will ignore the intervalometer process. ",  "x": 560,  "y": 560,  "wires": []  },  {  "id": "d7c91201063241c9",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "Control lights",  "info": "The CrackIdentified function is called when the switch receives a message \_crack\_status\_. When the message is processed and the function called, the system triggers the SenseHat emulator to turn on the onboard lights if the value is \*\*true\*\*. The onboard light indicates that a crack has been identified in the last image processed by the model. ",  "x": 570,  "y": 480,  "wires": []  },  {  "id": "84b46b7e1b72c933",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "reset",  "info": "The reset command is a simple function that will reset the system values to its initial state:\n - \*\*Sense HAT simulator:\*\* the onboard lights will be turned off\n - \*\*Intervalometer:\*\* the frequence will be reset to 5 seconds per snapshot.",  "x": 550,  "y": 680,  "wires": []  },  {  "id": "650e29d17c47528a",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "interval",  "info": "The interval function allows the dashboard to set the intervalometer frequency to a different value in seconds. The default value of the intervalometer is of 5 seconds, and it can be set to any other frequency through the interval command.\n\nThe interval command must contain an integer value greater than 0 to set the frequency to a different value from the default.",  "x": 550,  "y": 760,  "wires": []  },  {  "id": "5bab1376993cfc69",  "type": "comment",  "z": "a2ebda5b93a62082",  "name": "GCP\_Receive",  "info": "GCP\_Receive node is the GCP IoT Core API that connects the broker communication to receive commands from the Pub/Sub broker.\n\nNOTE: For local use, its important to use the correct registry ID and device ID, as well as enable access to the corresponding key file (RS256 encoded certificate) to authenticate with the GCP Pub/Sub broker. If the values are not set correctly, or if the certificate does not correspond to the key of the device, the connection will be rejected and messages will not transmit.",  "x": 110,  "y": 460,  "wires": []  },  {  "id": "8270c187.71d4d",  "type": "google-cloud-iotcore broker",  "name": "tst",  "commandsQos": "0",  "configQos": "1",  "brokerHost": "mqtt.googleapis.com",  "brokerPort": "8883",  "projectId": "gcu-dissertation",  "region": "europe-west1",  "registryId": "useful\_registry",  "deviceId": "EdgeDevice",  "algorithm": "RS256",  "privateKeyFile": "C:\\Users\\memoc\\Documents\\GCU\\Dissertation\\Certs IoT GCP\\ffs\_private.pem",  "keepAlive": "60",  "reconnectPeriod": "5000",  "connectTimeout": "30000",  "clean": "true"  }  ] |

Table 33: Node-RED Flow Code

## Calculated Costs

|  |  |  |  |
| --- | --- | --- | --- |
| Service Type | Region | Description | Estimated monthly cost |
| Amazon DocumentDB (with MongoDB compatibility) | EU (London) | ( 1 instances of type db t4g.medium ), Storage (4 GB), I/Os (1) | $63.47 |
| Amazon Personalize | EU (Ireland) | Average amount of data ingested per month (4 GB per month), Average training hours per month (8), TPS-hours per month for real time inference (720), Number of batch recommendations per month (1) | $146.120067 |
| AWS IoT Core | EU (London) | Number of devices (MQTT) (2), Number of messages for a device (720), Average number of actions executed per rule (1), Average size of each message (3000 KB), Average size of each record (1 KB) | $0.00 |
| AWS Lambda | EU (London) | Architecture (x86), Architecture (x86), Number of requests (4400 per month) | $0.00 |
|  |  | **Total** | $209.590067 |

Table 34: AWS Calculator Cost Estimate (in USD). Available services in the AWS calculator are included. Created and downloaded using the GCP calculator utility made available online.

|  |  |  |  |
| --- | --- | --- | --- |
| Service type | Region | Description | Estimated monthly cost |
| Azure Cosmos DB |  | Standard provisioned throughput (manual), Single Region Write (Single-Master) - UK West (Write region); 400 RU/s x 730 Hours; 4 GB transactional storage, 1 copies of periodic backup storage; Dedicated gateway not enabled | $24.36 |
| App Service | UK West | Free Tier; 1 F1 (0 Core(s), 1 GB RAM, 1 GB Storage) x 730 Hours; Windows OS | $0.00 |
| Azure IoT Hub | UK West | Standard Tier, Free: 500 devices, 8,000 msgs/day, $0.00/mo, 0 IoT Hub units | $0.00 |
| Azure Machine Learning | UK West | 1 DS1 v2 (1 Core(s), 3.5 GB RAM) x 730 Hours, Pay as you go | $64.02 |
| Azure Functions | UK West | Consumption tier, 256 MB memory, 200 milliseconds execution time, 300 executions/mo | $0.00 |
|  |  | **Total** | **$88.38** |

Table 35: Azure Calculator Cost Estimate (in USD). Available services in the Azure calculator are included. Created and downloaded using the GCP calculator utility made available online.

|  |  |  |  |
| --- | --- | --- | --- |
| Service type | Region | Description | Estimated monthly cost |
| App Engine standard environment | London (europe-west2) | Instances, 730 Instance Hours | USD 0.00 |
| Cloud Firestore | London (europe-west2) | 3041.66 Document Reads, 304116.66 Document Writes, 30416.66 Document Deletes | USD 0.00 |
| Cloud Functions | London (europe-west2) | 1440 invocations, Memory 256MB, CPU 200MHz | USD 0.00 |
| Pub/Sub | London (europe-west2) | Volume: 150 MiB, 4 subcriptions | USD 0.47 |
| AI Platform | London (europe-west2) | ML Training Units 0.407, Job Runtime 240 minutes, Online Prediction Mode, Total Node Hours per day 24 | USD 2.34 |
| Firestore | London (europe-west2) | A portion of your estimate fits within the Firestore free tier | USD 0.00 |

Table 36: GCP Calculator Cost Estimate (in USD). Available services in the GCP calculator are included. Created and downloaded using the GCP calculator utility made available online.

## AI Model Results

|  |
| --- |
| name: "projects/105067933135/locations/europe-west4/models/570426632490188800/evaluations/7305037607199571968"  metrics\_schema\_uri: "gs://google-cloud-aiplatform/schema/modelevaluation/classification\_metrics\_1.0.0.yaml"  metrics {  struct\_value {  fields {  key: "auPrc"  value {  number\_value: 0.9988157  }  }  fields {  key: "confidenceMetrics"  value {  list\_value {  values {  struct\_value {  fields {  key: "precision"  value {  number\_value: 0.5  }  }  fields {  key: "recall"  value {  number\_value: 1.0  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.05  }  }  fields {  key: "precision"  value {  number\_value: 0.96153843  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.1  }  }  fields {  key: "precision"  value {  number\_value: 0.96153843  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.15  }  }  fields {  key: "precision"  value {  number\_value: 0.97402596  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.2  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.25  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.3  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.35  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.4  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.45  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.5  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.55  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.6  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.65  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.7  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.75  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.8  }  }  fields {  key: "precision"  value {  number\_value: 0.9868421  }  }  fields {  key: "recall"  value {  number\_value: 0.9868421  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.85  }  }  fields {  key: "precision"  value {  number\_value: 0.9866667  }  }  fields {  key: "recall"  value {  number\_value: 0.9736842  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.875  }  }  fields {  key: "precision"  value {  number\_value: 0.9866667  }  }  fields {  key: "recall"  value {  number\_value: 0.9736842  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.9  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.91  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.92  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.93  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.94  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.95  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.96  }  }  fields {  key: "precision"  value {  number\_value: 0.9864865  }  }  fields {  key: "recall"  value {  number\_value: 0.9605263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.97  }  }  fields {  key: "precision"  value {  number\_value: 0.98630136  }  }  fields {  key: "recall"  value {  number\_value: 0.94736844  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.98  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.92105263  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.99  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.90789473  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.995  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.8947368  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.996  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.8684211  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.997  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.84210527  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.998  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.84210527  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 0.999  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.84210527  }  }  }  }  values {  struct\_value {  fields {  key: "confidenceThreshold"  value {  number\_value: 1.0  }  }  fields {  key: "precision"  value {  number\_value: 1.0  }  }  fields {  key: "recall"  value {  number\_value: 0.31578946  }  }  }  }  }  }  }  fields {  key: "confusionMatrix"  value {  struct\_value {  fields {  key: "annotationSpecs"  value {  list\_value {  values {  struct\_value {  fields {  key: "displayName"  value {  string\_value: "crack"  }  }  fields {  key: "id"  value {  string\_value: "133570871565484032"  }  }  }  }  values {  struct\_value {  fields {  key: "displayName"  value {  string\_value: "no\_crack"  }  }  fields {  key: "id"  value {  string\_value: "4745256889992871936"  }  }  }  }  }  }  }  fields {  key: "rows"  value {  list\_value {  values {  list\_value {  values {  number\_value: 53.0  }  values {  number\_value: 0.0  }  }  }  values {  list\_value {  values {  number\_value: 1.0  }  values {  number\_value: 22.0  }  }  }  }  }  }  }  }  }  fields {  key: "logLoss"  value {  number\_value: 0.053449232  }  }  }  }  create\_time {  seconds: 1637097247  nanos: 348111000  }  slice\_dimensions: "annotationSpec"  } |

Table 37: Vertex Auto AI Evaluation Detailed Results

1. https://flows.nodered.org/node/node-red-contrib-google-cloud [↑](#footnote-ref-2)
2. https://travis-ci.org/github/noflo/noflo-google [↑](#footnote-ref-3)
3. <https://thingsboard.io/docs/user-guide/install/gcp/> [↑](#footnote-ref-4)