AI-Powered Acoustic Wave and Computer Vision Analysis for Rail Defect Detection and Monitoring

Capstone Project Report

End-Semester Evaluation

Submitted by:

(102203043) Himanshu Bhumbla

(102203040) Divrose Kaur

(102203279) Pranav Dev Khindria

(102203480) Kezia

(102383006) Lalit Singla

BE Fourth Year- COE

CPG No. 51

Under the Mentorship of

Dr. Garima Singh

Assistant Professor - I

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

Jan-Dec 2025

ABSTRACT

The condition of railway track infrastructure plays a crucial role in ensuring safe and reliable train operations. Over time, railway tracks are subjected to heavy mechanical loads, environmental exposure, and continuous wear, which can lead to the development of surface-level defects such as cracks, flaking, and material degradation. Conventional inspection methods rely largely on manual surveys and periodic monitoring, making them time-consuming, costly, and unable to provide continuous or real-time assessment. Automated inspection systems based on computer vision offer a promising alternative; however, their effectiveness is often limited by background interference from ballast and the scarcity of labeled defect data, particularly for rare defect types.

This project presents a two-stage computer vision–based railway track defect detection system designed to operate under realistic field conditions. In the first stage, railway track regions are isolated from surrounding background using an instance segmentation model, enabling effective removal of ballast and other irrelevant elements. In the second stage, defect detection is performed using two parallel learning approaches: a supervised convolutional neural network for common defect categories and a few-shot learning model based on prototypical networks to improve detection of rare defects with limited training samples.

The system follows a client–server architecture, where image acquisition is carried out using a Raspberry Pi and computationally intensive processing is handled on a central server through REST-based APIs. Detected defects are visualized using a simple dashboard interface to support inspection and maintenance activities. Experimental evaluation indicates that rail masking significantly improves detection reliability, while the few-shot learning approach demonstrates better adaptability to class imbalance. The proposed system provides a scalable and practical solution for automated railway track monitoring and supports data-driven maintenance planning.

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DECLARATION

We hereby declare that the design principles and working prototype model of the project entitled **AI-Powered Acoustic Wave and Computer Vision Analysis for Rail Defect Detection and Monitoring** is an authentic record of our own work carried out in the Computer Science and Engineering Department, TIET, Patiala, under the guidance of **Dr. Garima Singh** during 7th semester (2025).

Date: 23-12-2025

Roll No.	Name	Signature
102203043	Himanshu Bhumbla	
102203040	Divrose Kaur	
102203279	Pranav Dev Khindria	
102203480	Kezia	
102383006	Lalit Singla	

Counter Signed By:

Dr. Garima Singh
Assistant Professor - I
Computer Science & Engineering Department
Department TIET, Patiala

ACKNOWLEDGEMENT

We would like to express our thanks to our mentor Dr. Garima Singh. She has been of great help in our venture, and an indispensable resource of technical knowledge. She is truly an amazing mentor to have.

We are also thankful to Dr. Neeraj Kumar, Head, Computer Science and Engineering Department, entire faculty and staff of Computer Science and Engineering Department, and also our friends who devoted their valuable time and helped us in all possible ways towards successful completion of this project. We thank all those who have contributed either directly or indirectly towards this project.

Lastly, we would also like to thank our families for their unyielding love and encouragement. They always wanted the best for us and we admire their determination and sacrifice.

Date: 23-12-2025

Roll No.	Name	Signature
102203043	Himanshu Bhumbla	
102203040	Divrose Kaur	
102203279	Pranav Dev Khindria	
102203480	Kezia	
102383006	Lalit Singla	

LIST OF FIGURES

- Fig 1: System Block Diagram
- Fig 2: Technology Stack
- Fig 3: MVC/Tier Architecture
- Fig 4: Deployment Diagram
- Fig 5: Sequence Diagram
- Fig 6: Component Diagram
- Fig 7: Activity Diagram
- Fig 8: Use-Case Diagram
- Fig 9: Class Diagram
- Fig 10: ER Diagram
- Fig 11: Data Flow Diagram (DFD Level-0)
- Fig 12: Dashboard Wireframe
- Fig 13: Defect Detail Wireframe
- Fig 14: Workflow of The Project
- Fig 15: Dashboard Images
- Fig 16: Dashboard Analysis

LIST OF TABLES

Table 1: Cost Analysis of Components
Table 2: Test Cases
Table 3: Peers Assessment
Table 4: Student Outcomes

LIST OF ABBREVIATION

- AI – Artificial Intelligence
- CNN – Convolutional Neural Network
- DFD – Data Flow Diagram
- DGPS – Differential Global Positioning System
- EC – Eddy Current Testing
- EMI – Electromagnetic Interference
- EN – European Norm (Standard)
- ESP32 – Espressif Systems Microcontroller
- GAN – Generative Adversarial Network
- GDPR – General Data Protection Regulation
- GPS – Global Positioning System
- IEEE – Institute of Electrical and Electronics Engineers
- IoT – Internet of Things
- ISO – International Organization for Standardization
- IT Act – Information Technology Act (India)
- ML – Machine Learning
- NDT – Non-Destructive Testing
- Pi – Raspberry Pi
- RAM – Random Access Memory
- SRS – Software Requirement Specification
- TRV – Track Recording Vehicle
- UT – Ultrasonic Testing
- YOLO – You Only Look Once (Deep Learning Model)

TABLE OF CONTENTS

ABSTRACT… ..... **i**
DECLARATION… ..... **ii**
ACKNOWLEDGEMENT… ..... **iii**
LIST OF FIGURES ..... **iv**
LIST OF TABLES ..... **v**
LIST OF ABBREVIATIONS ..... **vi**

CHAPTER……Page No.

1- INTRODUCTION
1.1 Project Overview 1
1.1.1 Technical terminology…. 5
1.1.2 Problem statement
1.1.3 Goal
1.1.4 Solution
1.2 Need Analysis
1.3 Research Gaps
1.4 Problem Definition and Scope
1.5 Assumptions and Constraints
1.6 Standards
1.7 Objectives
1.8 Methodology Used
1.9 Project Outcomes and Deliverables
1.10 Novelty of Work
2 - REQUIREMENT ANALYSIS
2.1 Literature Survey
2.1.1 Related Work
2.1.2 Research Gaps of Existing Literature
2.1.3 Detailed Problem Analysis
2.1.4 Survey of Tools and Technologies Used
2.1.5 Summary
2.2 Software Requirements Specification
2.2.1 Introduction
2.2.1.1 Purpose
2.2.1.2 Intended Audience and Reading Suggestions
2.2.1.3 Project Scope
2.2.2 Overall Description
2.2.2.1 Product Perspective
2.2.2.2 Product Features
2.2.3 External Interface Requirements

2.2.3.1 User Interfaces
2.2.3.2 Hardware Interfaces
2.2.3.3 Software Interfaces
2.2.4 Other Non-functional Requirements
2.2.4.1 Performance Requirements
2.2.4.2 Safety Requirements
2.2.4.3 Security Requirements
2.3 Cost Analysis
2.4 Risk Analysis
3 -METHODOLOGY ADOPTED
3.1 Investigative Techniques
3.2 Proposed Solution
3.3 Work Breakdown Structure
3.4 Tools +and Technologies Used
4 -DESIGN SPECIFICATIONS
4.1 System Architecture (Eg. Block Diagram / Component Diagram)
4.2 Design Level Diagrams
4.3 User Interface Diagrams
5 -IMPLEMENTATION AND EXPERIMENTAL RESULTS
5.1 Experimental Setup (or simulation)
5.2 Experimental Analysis
5.2.1 Data
5.2.2 Performance Parameters
5.3 Working of the project
5.3.1 Procedural Workflow
5.3.2 Algorithmic Approaches Used
5.3.3 Project Deployment
5.3.4 System Screenshots
5.4 Testing Process
5.4.1 Test Plan
5.4.2 Features to be tested
5.4.3 Test Strategy
5.4.4 Test Techniques
5.4.5 Test Cases
5.4.6 Test Results
5.5 Results and Discussions
5.6 Inferences Drawn
5.7 Validation of Objectives
6 -CONCLUSIONS AND FUTURE DIRECTIONS
6.1 Conclusions
6.2 Environmental, Economic and Societal Benefits
6.3 Reflections
6.4 Future Work

7 -PROJECT METRICS

7.1 Challenges Faced
7.2 Relevant Subjects
7.3 Interdisciplinary Knowledge Sharing
7.4 Peer Assessment Matrix
7.5 Role Playing and Work Schedule
7.6 Student Outcomes Description and Performance Indicators
7.7 Brief Analytical Assessment

APPENDIX A: REFERENCES
APPENDIX B: PLAGIARISM REPORT

1. Introduction

1.1 Project Overview

Railway transportation continues to be one of the most economical and widely used modes of travel, making the condition of railway track infrastructure a critical factor in ensuring operational safety. Railway tracks are subjected to continuous mechanical stress due to heavy axle loads, repeated vibrations, environmental variations, and prolonged operational cycles. Over time, these factors contribute to the development of surface-level defects such as cracks, flaking, spalling, and wear. If such defects are not detected at an early stage, they may propagate further and compromise track integrity, leading to increased safety risks and maintenance costs.

Conventional railway track inspection practices primarily rely on manual visual surveys and periodic inspections using specialized track recording vehicles. While these methods have been used for decades, they are inherently limited in their ability to provide continuous monitoring. Manual inspections are labor-intensive, time-consuming, and dependent on human judgment, which can vary across inspectors and working conditions. As railway networks expand and train frequencies increase, the limitations of traditional inspection approaches become more pronounced, highlighting the need for automated and scalable inspection solutions.

Advancements in computer vision and deep learning have enabled the development of automated inspection systems capable of analyzing visual data captured from cameras. However, the railway environment presents several practical challenges for vision-based approaches. Track images typically include ballast, stones, and surrounding elements that visually resemble rail surfaces, making it difficult for models to accurately isolate defect-prone regions. In addition, real-world railway defect datasets are highly imbalanced, with certain critical defects occurring far less frequently than others, which limits the effectiveness of conventional supervised learning models.

This project addresses these challenges by developing a practical computer vision–based railway track defect detection system tailored for real-world conditions. The proposed system follows a two-stage processing pipeline in which railway track regions are first isolated from the background using segmentation-based masking, followed by defect detection using deep learning models. The system is designed with deployment feasibility in mind, adopting a client–server architecture that separates image acquisition from computational processing. Detection results are presented through a visualization interface to assist inspection and maintenance activities, supporting more reliable and data-driven railway track monitoring.

1.1.1 Technical Terminology

- **Railway Track Defects:** Physical abnormalities or damage present on the surface of railway tracks, such as cracks, flaking, spalling, squats, and shelling, which may adversely affect track integrity and operational safety.
- **Ballast:** Crushed stone or gravel laid around railway tracks to provide structural stability and drainage. In visual inspection systems, ballast often introduces background noise due to its visual similarity to rail surfaces.
- **Computer Vision:** A domain of artificial intelligence that focuses on enabling machines to interpret, analyze, and extract meaningful information from images or video data.
- **Instance Segmentation:** A computer vision technique that identifies and delineates individual objects within an image at the pixel level. In this project, it is used to precisely isolate railway track regions from surrounding background elements.
- **Binary Segmentation Mask:** A pixel-level mask generated by applying a fixed threshold to segmentation confidence values, where pixels belonging to the railway track are retained and all non-rail pixels are removed.
- **Convolutional Neural Network (CNN):** A deep learning model widely used for image-based tasks such as feature extraction and classification by learning spatial patterns from visual data.
- **Few-Shot Learning:** A learning approach that enables models to recognize and classify categories using a very limited number of labeled training examples, making it suitable for handling rare defect classes.

- **Prototypical Networks:** A metric-based few-shot learning technique that represents each class using a prototype vector and performs classification based on distance measures in an embedding space.
- **Client–Server Architecture:** A system architecture in which data acquisition is handled by a client device, while processing and analysis are performed on a centralized server.
- **Raspberry Pi:** A compact and low-cost single-board computer used in this project for image capture and data transmission in the inspection pipeline.

1.1.2 Problem Statement

Automated railway track defect detection using computer vision faces significant challenges when applied to real-world railway environments. Railway track images are heavily affected by ballast, stones, and surrounding background elements that introduce visual noise and make it difficult for models to accurately focus on rail surfaces. This background interference often results in false detections and unreliable inspection outcomes.

In addition, real-world railway defect datasets exhibit severe class imbalance, where commonly occurring defects dominate the dataset while rare but safety-critical defects, such as surface cracks, have very limited representation. Conventional supervised deep learning models tend to perform poorly under such conditions, leading to biased predictions and reduced generalization. These challenges highlight the need for a defect detection system that explicitly isolates rail regions and adopts learning strategies capable of handling limited defect data.

1.1.3 Goal

The goal of this project is to design and implement an automated, computer vision–based railway track defect detection system capable of identifying surface-level defects under realistic environmental conditions. The system aims to minimize the impact of background interference and limited defect data while supporting practical deployment through clear, interpretable inspection outputs for maintenance personnel.

1.1.4 Solution

To address the identified challenges, this project proposes a two-stage computer vision–based defect detection pipeline designed for real-world railway

environments. In the first stage, an instance segmentation model is used to identify railway track regions and generate pixel-wise confidence values. These values are converted into a binary segmentation mask using a fixed threshold, ensuring that only rail surface pixels are retained while all background elements such as ballast are completely removed.

In the second stage, defect detection is performed on the extracted rail region using two complementary learning approaches. A supervised convolutional neural network is employed to detect commonly occurring surface defects, while a few-shot learning model based on prototypical networks is used to improve recognition of rare defect types with limited training data.

The system follows a client–server architecture in which image acquisition is handled by a Raspberry Pi, and computationally intensive processing is performed on a central server using REST-based APIs. The detection results are visualized through a dashboard interface, enabling efficient interpretation of inspection outputs and supporting maintenance decision-making.

1.2 Need Analysis

Railway infrastructure forms the backbone of mass transportation systems, and the safety of railway operations is directly dependent on the condition of the track. Railway tracks are continuously subjected to heavy axle loads, high-speed vibrations, temperature variations, and environmental exposure such as rain, dust, and debris. Over extended periods of operation, these factors contribute to the gradual development of surface-level defects including cracks, flaking, spalling, and wear. If such defects are not identified and addressed at an early stage, they can propagate further, increasing the risk of derailments, service disruptions, and costly maintenance interventions.

At present, railway track inspection is largely based on manual visual inspections and periodic surveys using specialized inspection vehicles. Although these approaches are widely adopted, they suffer from several limitations. Manual inspections are labor-intensive, time-consuming, and dependent on human judgment, which may vary between inspectors. Periodic inspection schedules also mean that defects developing between inspection cycles can remain undetected for

extended durations. With the expansion of railway networks and the increasing frequency of train operations, relying solely on traditional inspection methods has become increasingly impractical and expensive.

In recent years, computer vision–based inspection systems have emerged as a promising alternative due to their ability to automate defect detection using image data captured during regular railway operations. Camera-based inspection allows non-contact monitoring and can be deployed without interrupting train schedules. However, the real-world railway environment presents unique challenges that limit the effectiveness of many existing vision-based approaches. Railway track images often contain heavy ballast, stones, and surrounding objects that visually resemble rail surfaces, making it difficult for models to accurately focus on defect-prone regions. As a result, background interference frequently leads to false detections and reduced reliability.

Another critical challenge is the severe class imbalance present in real railway defect datasets. Certain defect types, such as surface cracks, occur relatively infrequently compared to more common defects like flaking or wear. Conventional deep learning classifiers tend to favor majority classes, resulting in poor detection performance for rare but safety-critical defects. This imbalance reduces the practical usefulness of automated systems, as missing rare defects can have serious safety implications.

The need for this work arises from the requirement to develop a railway defect detection system that is both technically robust and practically deployable. There is a clear demand for solutions that can effectively isolate railway track regions from complex backgrounds, reduce false detections caused by ballast interference, and improve detection reliability for rare defect categories. Additionally, inspection outputs must be presented in a form that is easy to interpret by maintenance personnel, enabling faster decision-making and timely corrective actions.

By combining rail-aware preprocessing, deep learning–based defect detection, and deployment-oriented system design, this project addresses key limitations of existing inspection approaches. The proposed system contributes toward safer

railway operations, reduced dependence on manual inspections, and a shift toward data-driven and condition-based maintenance strategies, which are increasingly emphasized in modern railway infrastructure management.

1.3 Research Gaps

Despite significant advancements in railway infrastructure monitoring and automated defect detection, several critical research gaps remain unresolved, particularly when transitioning from controlled experimental environments to real-world deployment. A detailed review of existing literature highlights the following gaps, which motivate the present work.

Research Gap 1: Inadequate Handling of Ballast and Background Interference

Many existing computer vision–based railway inspection systems process raw railway images without explicitly addressing background interference caused by ballast, stones, and surrounding track elements. Traditional image processing techniques and even some deep learning models struggle to differentiate between rail surfaces and visually similar background textures, leading to false detections and reduced inspection reliability. Although certain studies employ region-of-interest extraction, precise rail isolation using instance-level segmentation and strict background removal remains insufficiently explored in practical railway inspection systems [1], [2].

Research Gap 2: Limited Focus on Rare but Safety-Critical Defect Classes

Most railway defect detection models are trained using highly imbalanced datasets in which commonly occurring defects dominate the training samples. As a result, rare but safety-critical defects, such as surface cracks, receive limited representation during model training, leading to poor detection performance. Conventional supervised classification approaches tend to bias predictions toward majority classes, which is unsuitable for safety-sensitive railway applications. There is limited adoption of learning strategies specifically designed to address severe class imbalance in railway defect detection [3].

Research Gap 3: Over-Reliance on Conventional Supervised Learning Approaches

A significant portion of existing research relies exclusively on standard supervised deep learning models for railway defect classification. While such models perform well when large and balanced datasets are available, they exhibit reduced generalization in scenarios with limited labeled data. Learning paradigms such as few-shot learning and metric learning, which are well-suited for recognizing rare defect classes from scarce examples, remain largely underexplored in the context of railway surface defect detection despite their success in other application domains [4].

Research Gap 4: Limited Emphasis on Deployment-Oriented System Design

Several studies report high detection accuracy under controlled laboratory conditions but provide limited discussion on deployment feasibility. Practical concerns such as computational requirements, compatibility with edge devices, communication overhead, and system scalability are often overlooked. Consequently, many proposed solutions remain difficult to integrate into operational railway inspection workflows. There is a clear need for defect detection systems that are designed with deployment-oriented architectures, such as client–server models, and realistic processing constraints in mind [5].

Research Gap 5: Insufficient Integration of Rail-Aware Preprocessing with Defect Detection

While defect detection performance is frequently emphasized, fewer studies investigate the integration of rail-aware preprocessing techniques with downstream classification models. The lack of explicit preprocessing strategies that strictly isolate railway track regions prior to defect analysis often results in models learning background-dependent features. This gap highlights the need for structured pipelines that enforce a clear separation between rail and non-rail regions before defect detection [6].

Research Gap 6: Limited Support for Practical Visualization and Maintenance Decision-Making

Although automated defect detection accuracy is widely reported in literature, comparatively less attention is given to how inspection results are presented to end users. Many systems focus on technical predictions without providing intuitive visualization or inspection summaries suitable for maintenance personnel. The absence of user-friendly dashboards and interpretable outputs reduces the practical

applicability of automated railway inspection systems in real-world maintenance environments [7].

1.4 Problem Definition And Scope

Problem Definition

The core problem addressed in this project is the reliable detection of surface-level railway track defects using computer vision under real-world operating conditions. Although automated image-based inspection systems have demonstrated potential, their practical effectiveness is often limited when applied outside controlled environments. Railway track images typically contain heavy ballast, stones, and surrounding elements that introduce significant visual noise, making it difficult for models to accurately focus on defect-prone rail surfaces.

In addition to background interference, real-world railway defect datasets exhibit severe class imbalance. Certain defect types, such as surface cracks, occur far less frequently than others, resulting in limited training samples for these safety-critical categories. Conventional supervised deep learning models trained on such datasets tend to bias predictions toward majority classes, leading to missed detections and poor generalization for rare defects. As a consequence, many existing automated inspection approaches suffer from false detections, unreliable performance, and reduced applicability in practical railway monitoring scenarios.

To address these challenges, there is a need for a defect detection system that explicitly isolates railway track regions from complex backgrounds, incorporates learning strategies capable of handling imbalanced data, and remains suitable for deployment under realistic computational and operational constraints.

Scope of the Project

The scope of this project is focused on the design and implementation of a computer vision–based system for detecting **surface-level defects on railway tracks** using image data captured from real railway environments. The proposed system aims to improve defect detection reliability through structured rail-aware

preprocessing and robust learning strategies while maintaining deployment feasibility.

The scope of the work includes:

- Development of a rail-aware preprocessing pipeline that isolates railway track regions from background elements such as ballast and stones using binary segmentation masking.
- Implementation of deep learning–based models for detecting commonly occurring surface-level railway track defects.
- Integration of a few-shot learning approach to enhance detection performance for rare defect classes with limited labeled training data.
- Design of a client–server architecture in which image acquisition is performed by a lightweight edge device and computational processing is carried out on a centralized server.
- Visualization of defect detection results through a simple and interpretable dashboard interface to assist inspection and maintenance personnel.

The scope of the project excludes:

- Detection of subsurface or internal railway defects using ultrasonic or other non-destructive testing techniques.
- Large-scale field deployment and long-term operational validation on active railway networks.
- Prediction of defect progression, remaining useful life estimation, or automated maintenance decision-making.

The defined scope ensures that the project remains focused on addressing clearly identified challenges while providing a practical and extensible foundation for future enhancements. By limiting the scope to surface-level defect detection and deployment-oriented system design, the project achieves a balanced combination of technical depth, practical feasibility, and academic rigor.

1.5 Assumptions And Constraints

Every engineering system is developed under a set of assumptions and operates within practical constraints. Clearly identifying these assumptions and constraints helps define the operating conditions of the proposed system and sets realistic expectations regarding its performance, applicability, and limitations.

Assumptions

The development and evaluation of the proposed railway track defect detection system are based on the following assumptions:

- It is assumed that the images captured for inspection provide a reasonably clear view of the railway track surface. While variations in lighting conditions and minor motion blur are expected in real-world scenarios, extremely poor image quality may adversely affect defect detection performance.
- The camera used for image acquisition is assumed to be properly positioned and aligned such that the railway track remains within the field of view during the capture process.
- It is assumed that a stable network connection is available for communication between the client device and the central server, as image data and associated metadata are transmitted using REST-based APIs.
- The system assumes that the defect categories encountered during deployment are consistent with those included in the training dataset. Defect types that are entirely unseen during training are not explicitly handled in the current implementation.
- It is assumed that maintenance personnel interacting with the system possess basic familiarity with digital interfaces and are able to interpret visual outputs and inspection summaries provided through the dashboard.

Constraints

The proposed system operates under the following constraints, which influence its design choices and performance characteristics:

- **Dataset Availability Constraint:**

  The availability of labeled railway defect images is limited, particularly for rare but safety-critical defect classes. This constraint affects supervised model training and motivates the adoption of few-shot learning techniques to improve detection performance under limited data conditions.

- **Environmental Constraint:**

  Real-world railway environments introduce challenges such as varying illumination, shadows, dust, and partial occlusions. Although rail-aware preprocessing reduces background interference, extreme environmental conditions may still impact detection reliability.

- **Computational Constraint:**

  The edge device used for image acquisition has limited computational resources. Consequently, computationally intensive processing is offloaded to a centralized server, and real-time inference on the edge device is constrained.

- **Network Dependency Constraint:**

  The client–server architecture depends on network connectivity for transmitting image data and receiving detection results. Network latency or connectivity interruptions may affect response time or system availability.

- **Implementation Constraint:**

  The current implementation does not include a fallback mechanism in scenarios where rail segmentation confidence is insufficient. In such cases, the system proceeds based on the generated segmentation output without alternative processing paths.

- **Scope Constraint:**

  The system is constrained to detecting surface-level railway track defects only. Detection of subsurface or internal defects using ultrasonic or other non-destructive testing techniques is outside the scope of the present work.

- **Scalability Constraint:**

  Although the system is designed to be modular, large-scale deployment across extensive railway networks would require further optimization, infrastructure support, and comprehensive field validation.

1.6 Standards

The design and implementation of the proposed railway track defect detection system follow relevant technical, software, and ethical standards to ensure reliability, interoperability, and responsible system development. Adhering to established standards helps improve system quality and supports future integration and scalability.

From a software engineering perspective, the system follows standard practices for modular design and client–server architecture. RESTful API principles are adopted for communication between the image acquisition client and the central processing server. These APIs use standard HTTP methods and JSON-based data exchange, ensuring platform independence and ease of integration with other systems or dashboards.

In terms of computer vision and machine learning, the project follows widely accepted practices such as dataset splitting into training and testing sets, use of transfer learning for efficient model training, and performance evaluation using standard metrics like accuracy, precision, and recall. Image preprocessing steps such as normalization and resizing are carried out in accordance with common deep learning input standards to ensure consistent model behavior.

For hardware and deployment, the use of a Raspberry Pi as an edge device aligns with standard embedded system practices for low-power image acquisition and lightweight processing. Server-side deployment adheres to general cloud and web service standards, enabling scalability and maintainability.

The project also considers safety and ethical standards relevant to automated inspection systems. The system is designed as a decision-support tool rather than a fully autonomous decision-making system, ensuring that final maintenance actions remain under human supervision. Additionally, only railway track images are processed, and no personal or sensitive user data is collected, supporting responsible data usage practices.

Overall, the project aligns with commonly accepted standards in software development, machine learning experimentation, and system deployment, ensuring that the proposed solution is reliable, extensible, and suitable for practical implementation.

1.7 Objectives

The primary objective of this project is to design and implement an automated, computer vision–based system for detecting surface-level defects in railway tracks

under realistic operating conditions. To achieve this overarching goal, the following specific objectives have been defined:

- **To study and analyze existing railway track inspection techniques** and identify their limitations with respect to automation, reliability, and practical deployment.

- **To develop a rail-aware preprocessing mechanism** that isolates railway track regions from complex backgrounds containing ballast and surrounding elements using segmentation-based masking.
- **To design and implement a deep learning–based defect detection pipeline** for identifying commonly occurring surface-level railway track defects from image data.
- **To address severe class imbalance present in the dataset consisting of five surface defect classes** by incorporating a few-shot learning approach to improve detection performance for rare but safety-critical defect types.
- **To evaluate the effect of rail region isolation on defect detection performance**, using appropriate performance metrics and comparative analysis.
- **To design a deployment-oriented system architecture** based on a client–server model that separates image acquisition from computationally intensive processing.
- **To implement a simple and intuitive visualization interface** for presenting defect detection results in a form that can be easily interpreted by maintenance personnel.
- **To validate the proposed system under realistic operational constraints**, focusing on detection reliability and deployment feasibility for railway inspection scenarios.

1.8 Methodology Used

The methodology adopted in this project follows a structured and implementation-oriented approach to address the challenges associated with automated railway track defect detection in real-world environments. The proposed methodology is designed to reduce background interference, handle limited defect data, and ensure

deployment feasibility through a modular system architecture. The overall approach is divided into sequential stages, beginning with image acquisition and preprocessing, followed by defect detection and result visualization.

At a high level, the methodology is based on a **two-stage processing pipeline**. In the first stage, railway track regions are isolated from the background using segmentation-based masking. In the second stage, defect detection is performed on the extracted rail region using deep learning models, including a supervised classifier for common defects and a few-shot learning model for rare defect classes.

Stage 1: Image Acquisition and Data Transmission

Image acquisition is performed using a camera connected to a Raspberry Pi, which acts as the client device in the system architecture. The Raspberry Pi captures images of the railway track surface during inspection. In addition to image data, location information is obtained using a GPS module connected to the client device. The captured image and associated GPS data are encoded and transmitted to a centralized server through REST-based APIs.

Due to the limited computational resources available on the edge device, no intensive image processing or inference is performed at the client side. Instead, the Raspberry Pi is responsible only for reliable data acquisition and transmission, ensuring low power consumption and deployment simplicity.

Stage 2: Rail Region Isolation Using Binary Segmentation Masking

Once the image is received by the server, preprocessing is performed to isolate the railway track region from the surrounding background. An instance segmentation model is used to generate pixel-wise confidence values indicating the likelihood of each pixel belonging to the railway track.

These confidence values are converted into a **binary segmentation mask** by applying a fixed threshold. Pixels classified as rail regions are retained, while all non-rail pixels corresponding to ballast, stones, and background elements are completely removed. The resulting binary mask is applied to the original image to extract a clean representation of the railway track surface. This rail-aware preprocessing step significantly reduces background interference and ensures that subsequent defect detection focuses only on relevant regions.

Stage 3: Defect Detection Using Deep Learning Models

Defect detection is performed on the extracted rail region using two complementary learning approaches to address different data availability scenarios.

For defect classes with sufficient labeled training data, a **supervised convolutional neural network (CNN)** is used to identify commonly occurring surface-level defects. The CNN learns discriminative visual features from the rail region images and provides reliable classification for majority defect categories.

To handle rare but safety-critical defect types with limited training samples, a **few-shot learning approach based on prototypical networks** is incorporated. This model learns an embedding space in which defect classes are represented by prototype vectors. Classification is performed by measuring the distance between the extracted features and class prototypes, enabling improved recognition of rare defect classes even with very few examples.

Stage 4: Result Aggregation and Visualization

The outputs generated by the defect detection models, along with associated metadata such as location information, are aggregated at the server. The final inspection results are then presented through a dashboard interface. The visualization interface provides a clear and interpretable representation of detected defects, enabling maintenance personnel to quickly understand inspection outcomes and support informed decision-making.

1.9 Project Outcomes and Deliverables

The successful completion of this project has resulted in the development of a functional and deployment-oriented railway track defect detection system. The outcomes of the project demonstrate the practical feasibility of applying computer vision and deep learning techniques for automated railway inspection under realistic constraints. The key outcomes and deliverables of the project are summarized below.

Project Outcomes

- A two-stage computer vision–based defect detection pipeline capable of isolating railway track regions from complex backgrounds and detecting surface-level defects from image data.
- An instance segmentation–based rail isolation mechanism that generates binary segmentation masks, effectively removing ballast and background elements prior to defect analysis.
- A supervised deep learning model for reliable detection of commonly occurring surface-level railway track defects.
- A few-shot learning–based defect detection approach using prototypical networks to improve recognition of rare but safety-critical defect classes with limited training samples.
- A client–server system architecture that separates image acquisition from computational processing, enabling lightweight edge-device operation and centralized inference.
- Integration of image data with associated metadata for structured inspection output and improved interpretability.
- A visualization interface that presents defect detection results in a clear and interpretable manner to assist maintenance personnel in inspection and decision-making processes.

Project Deliverables

- A trained instance segmentation model for railway track region isolation and background removal.
- Trained deep learning models for defect detection, including both supervised and few-shot learning–based classifiers.
- A client-side image acquisition module implemented on a Raspberry Pi for capturing and transmitting inspection data.
- A server-side processing pipeline implemented using REST-based APIs for preprocessing, inference, and result aggregation.
- A dashboard interface for visualizing defect detection results and inspection summaries.
- Project documentation detailing system design, methodology, implementation, and experimental evaluation.

1.10 Novelty of Work

The novelty of this project lies in its practical and deployment-oriented approach to railway track defect detection, where real-world challenges are addressed through a structured and technically grounded solution rather than purely theoretical modeling. Unlike many existing studies that focus primarily on improving detection accuracy under controlled conditions, this work emphasizes robustness, data limitations, and feasibility of implementation in realistic railway environments.

A key novel aspect of the proposed system is the explicit use of rail-aware preprocessing through binary segmentation masking. Instead of performing defect detection directly on raw railway images, the system first isolates the railway track region using instance segmentation and converts the output into a binary mask. This strict separation of rail and non-rail regions significantly reduces background interference caused by ballast and surrounding elements, leading to more reliable defect analysis.

Another important contribution of this work is the integration of few-shot learning for handling rare defect classes within a railway inspection context. While most existing systems rely solely on conventional supervised learning, this project incorporates a prototypical network–based few-shot learning approach to improve detection performance for safety-critical defects with limited training samples. This dual-learning strategy allows the system to handle both common and rare defect scenarios more effectively.

The project also demonstrates novelty through its deployment-oriented system design. The adoption of a client–server architecture, where image acquisition is performed on a lightweight edge device and computationally intensive processing is handled on a centralized server, makes the system more practical for real-world use. This design choice balances performance and resource constraints while supporting scalability.

Additionally, the inclusion of an intuitive visualization interface for presenting inspection results enhances the practical usability of the system. Rather than

providing only raw predictions, the system presents outputs in a form that supports interpretation and decision-making by maintenance personnel.

Overall, the novelty of this work lies in the combination of rail-aware preprocessing, imbalance-resilient learning strategies, and deployment-focused system architecture, resulting in a comprehensive and practical solution for automated railway track defect detection.

2. Requirement Analysis

Requirement analysis is a crucial phase in the development of any engineering system, as it defines what the system is expected to achieve and the conditions under which it must operate. In the context of railway track inspection, requirement analysis helps translate real-world safety, operational, and infrastructural challenges into clearly defined technical and functional system requirements.

2.1 Literature Survey

2.1.1 Related Work

Railway track inspection has traditionally relied on manual visual inspections and non-destructive testing techniques such as ultrasonic testing, eddy current methods, and fiber-optic sensing. These techniques are effective for detecting internal or subsurface defects; however, they require specialized equipment, trained operators, and often interrupt regular railway operations. Due to these limitations, traditional methods are difficult to scale for frequent and continuous monitoring of large railway networks.

With the advancement of image acquisition systems and increased availability of visual data, researchers have explored computer vision–based approaches for detecting surface-level railway defects. Early vision-based methods primarily relied on classical image processing techniques, including edge detection, thresholding, texture analysis, and handcrafted feature extraction. While these methods showed initial promise, they were highly sensitive to lighting variations, background noise, and changes in rail surface appearance, which limited their robustness in real-world environments.

The introduction of deep learning, particularly convolutional neural networks (CNNs), marked a significant improvement in automated railway inspection. CNN-based models demonstrated superior performance by learning hierarchical visual features directly from image data, enabling more accurate classification of surface defects such as cracks, spalling, and wear. Several studies reported improved detection accuracy compared to traditional image processing methods, especially under controlled data collection conditions.

More recent research has focused on object detection and segmentation-based approaches to improve defect localization. Models such as YOLO and Mask R-CNN have been applied to railway inspection tasks to detect and localize defects more precisely. These approaches allow for region-level and pixel-level analysis, which enhances spatial understanding of defects. However, many existing studies apply these models directly to raw railway images without explicitly isolating the railway track from surrounding elements such as ballast and sleepers. As a result, background interference remains a major challenge, often leading to false detections.

Another key limitation identified in existing literature is the reliance on large and balanced datasets. In real-world railway environments, defect datasets are highly imbalanced, with rare but safety-critical defects having very limited labeled samples. Conventional supervised learning approaches tend to perform poorly under such conditions. Although few-shot learning and metric learning techniques have shown success in other industrial inspection domains, their application to railway track defect detection remains relatively limited.

In addition to detection performance, recent studies have highlighted the importance of system-level considerations such as deployment feasibility, computational efficiency, and result interpretation. While many research works report high accuracy in laboratory settings, fewer studies address practical deployment aspects, including edge-device constraints, network-based processing, and usability of inspection outputs for maintenance personnel.

Overall, existing research demonstrates the potential of computer vision and deep learning for railway track defect detection but also reveals persistent challenges

related to background interference, data imbalance, and real-world deployment. These limitations motivate the need for a rail-aware, data-efficient, and deployment-oriented inspection system.

2.1.2 Research Gaps of Existing Literature

A critical analysis of existing literature on railway track defect detection reveals several unresolved research gaps that limit the practical effectiveness of current approaches, particularly in real-world deployment scenarios.

1. Insufficient Handling of Background Interference:

Many vision-based inspection systems perform defect detection directly on raw railway images without explicitly isolating the rail surface. Background elements such as ballast, sleepers, and surrounding infrastructure often introduce visual patterns similar to rail defects, leading to false detections and reduced reliability. Although region-based methods are occasionally used, strict rail isolation through segmentation-based masking is still underexplored in practical systems.

2. Dependence on Imbalanced and Limited Datasets:

Real-world railway defect datasets are inherently imbalanced, with common defects dominating available samples while rare but safety-critical defects remain underrepresented. Most existing studies rely on conventional supervised learning models that assume balanced data distribution, resulting in biased predictions and poor generalization for rare defect classes.

3. Over-Reliance on Conventional Supervised Learning:

The majority of prior work adopts standard supervised deep learning techniques, which require large volumes of labeled data to achieve robust performance. Few-shot and metric learning approaches, which are better suited for learning from limited samples, have seen limited application in railway track defect detection despite their success in other industrial inspection domains.

4. Limited Focus on Deployment Feasibility:

Several studies report high detection accuracy under controlled laboratory

conditions but provide minimal discussion on deployment-related constraints. Factors such as edge-device limitations, network dependency, processing latency, and scalability are often overlooked, making many solutions difficult to implement in operational railway environments.

5. Lack of Integration Between Detection and Practical Usability:
While detection accuracy is commonly emphasized, fewer studies address how inspection results are presented to end users. The absence of intuitive visualization interfaces and actionable outputs limits the usefulness of automated inspection systems for maintenance personnel and decision-making processes.

These gaps indicate the need for a comprehensive approach that combines rail-aware preprocessing, data-efficient learning strategies, and deployment-oriented system design. Addressing these gaps forms the foundation of the proposed work.

2.1.3 Detailed Problem Analysis

Based on the literature survey and identified research gaps, the problem of automated railway track defect detection can be analyzed in detail by examining the key technical and operational challenges that affect system performance in real-world environments.

Background Interference and Visual Complexity:
Railway track images captured in operational settings contain complex visual backgrounds, including ballast, sleepers, fasteners, and surrounding infrastructure. These elements often share similar textures and color patterns with rail surfaces and defects. When defect detection models are applied directly to raw images, they may incorrectly learn features from non-rail regions, leading to false positives and inconsistent predictions. This issue highlights the need for explicit rail-aware preprocessing that ensures defect analysis is performed only on relevant rail surface regions.

Data Imbalance and Limited Defect Samples:
Another major challenge arises from the inherent imbalance in railway defect datasets. In practical scenarios, commonly occurring defects account for a large

portion of the data, while rare but safety-critical defects, such as surface cracks, have very limited representation. Conventional supervised learning models trained on such datasets tend to favor majority classes, resulting in poor detection performance for rare defects. This imbalance reduces the reliability of automated inspection systems, particularly in safety-sensitive railway applications where missing rare defects can have serious consequences.

Generalization Under Real-World Conditions:

Many defect detection models demonstrate strong performance under controlled conditions but struggle to generalize when exposed to variations in lighting, shadows, dust, and partial occlusions commonly found in railway environments. Differences in rail appearance due to wear, environmental exposure, and imaging conditions further increase variability in input data. These factors make it difficult for models trained on limited datasets to maintain consistent performance across different operational scenarios.

Deployment and System-Level Constraints:

From a system perspective, practical deployment introduces additional constraints related to computation, communication, and usability. Edge devices used for image acquisition typically have limited processing capabilities, making it impractical to perform complex inference locally. Network-based data transmission introduces latency and dependency on connectivity, which must be considered during system design. Furthermore, inspection outputs must be presented in a form that is easily interpretable by maintenance personnel to support timely and informed decision-making.

Problem Consolidation:

Considering the above challenges, the core problem can be summarized as the need to develop a railway track defect detection system that can reliably operate under real-world conditions by reducing background interference, addressing data imbalance, maintaining generalization across varying environments, and supporting practical deployment constraints. Addressing these aspects collectively is essential for transitioning automated inspection systems from research prototypes to usable field applications.

2.1.4 Survey of Tools and Technologies Used

The development of an automated railway track defect detection system requires the integration of multiple tools and technologies spanning computer vision, machine learning, embedded systems, and system deployment. Based on the literature survey and practical requirements of real-world railway inspection, the following tools and technologies are commonly used and form the foundation of the proposed system.

Computer Vision Techniques:

Computer vision is the core technology used for analyzing railway track images and identifying defect patterns. Vision-based inspection enables non-contact monitoring and supports frequent data collection without disrupting railway operations. Image preprocessing, feature extraction, and segmentation techniques are widely used to prepare raw images for defect analysis.

Deep Learning and Convolutional Neural Networks (CNNs):

Convolutional Neural Networks are extensively used for image-based defect detection due to their ability to automatically learn discriminative features from visual data. CNNs have demonstrated strong performance in classifying surface-level defects such as cracks, spalling, and wear, especially when trained on sufficient labeled data.

Instance Segmentation and Object Detection Models:

Instance segmentation models enable pixel-level identification of objects within an image, making them suitable for isolating railway track regions from complex backgrounds. Object detection frameworks such as YOLO are frequently used due to their balance between accuracy and computational efficiency. These models support precise localization and region-based analysis of defects.

Few-Shot Learning Techniques:

Few-shot learning approaches, including metric learning methods such as prototypical networks, are increasingly explored to address scenarios where labeled training data is limited. These techniques enable models to recognize rare

defect classes using only a small number of examples, making them suitable for real-world railway datasets that exhibit severe class imbalance.

Embedded Systems for Image Acquisition:

Embedded platforms such as Raspberry Pi are commonly used for image acquisition in automated inspection systems. These devices offer low power consumption, compact size, and sufficient flexibility for integrating cameras and sensors, making them suitable for deployment in railway environments.

Client–Server Architecture and Web Technologies:

Client–server architectures are widely adopted to separate data acquisition from computationally intensive processing. In such systems, images captured at the client side are transmitted to a centralized server for preprocessing and inference. Communication is typically handled through REST-based APIs, ensuring scalability, modularity, and ease of integration.

Visualization and User Interface Tools:

Visualization tools and dashboard interfaces play a crucial role in presenting inspection results to end users. Clear and interpretable visual outputs enable maintenance personnel to quickly assess defect locations and severity, supporting effective decision-making.

2.1.5 Summary

The literature survey highlights substantial progress in railway track defect detection through the use of computer vision and deep learning techniques. Existing research demonstrates the effectiveness of convolutional neural networks, object detection models, and segmentation approaches in identifying surface-level railway defects under controlled conditions. However, the survey also reveals persistent limitations related to background interference, data imbalance, deployment feasibility, and practical usability.

The proposed work builds upon prior research by adopting proven deep learning techniques while addressing key shortcomings identified in existing literature. Unlike many previous approaches that perform defect detection directly on raw

railway images, this project introduces rail-aware preprocessing through segmentation-based masking to explicitly isolate railway track regions and reduce background noise. Furthermore, the integration of a few-shot learning approach extends conventional supervised models by improving detection performance for rare but safety-critical defect classes with limited training data.

In addition to methodological improvements, this work differs from many existing studies by emphasizing deployment-oriented system design. The use of a client–server architecture and an intuitive visualization interface ensures that the proposed system is not only technically effective but also practical for real-world railway inspection scenarios. By combining rail-aware preprocessing, data-efficient learning strategies, and deployment feasibility, the proposed system offers a more comprehensive and realistic solution compared to existing approaches.

2.3 Software Requirements Specification

2.3.1 Introduction

The Software Requirements Specification (SRS) defines the functional and non-functional requirements of the proposed computer vision–based railway track defect detection system. This document provides a clear and structured description of the system's intended behavior, operational constraints, and software components required to support automated inspection of railway tracks.

The proposed system integrates rail-aware image preprocessing, deep learning–based defect detection, and GPS-enabled location tagging within a client–server architecture. The SRS serves as a reference for system development, evaluation, and future enhancement by clearly outlining what the system is designed to do and the conditions under which it operates.

2.3.1.1 Purpose

The purpose of this document is to specify the software requirements of the railway track defect detection system developed in this project. It defines how the system processes railway track images using segmentation-based masking, applies

supervised and few-shot learning models for surface defect detection, and associates inspection results with location information.

This SRS is intended to provide clarity regarding system functionality, design expectations, and operational limitations, thereby serving as a reference for developers, evaluators, and end users involved in the project.

2.3.1.2 Intended Audience and Reading Suggestions

This document is intended for the following audiences:

- **Project Developers:**
  To understand software modules, data flow, and integration requirements for implementing the defect detection pipeline and client–server communication.
- **Mentors and Panel Evaluators:**
  To assess the technical feasibility, completeness, and correctness of the proposed system in relation to project objectives.
- **Maintenance Personnel (End Users):**
  To gain an understanding of how the system supports railway track inspection through automated defect detection and visual reporting.
- **Future Researchers:**
  To use the documented system as a foundation for extending the work with advanced models, additional sensors, or large-scale deployment strategies.

2.3.1.3 Project Scope

The scope of the software system defined in this SRS includes the design and implementation of a prototype railway track defect detection system with the following capabilities:

- Use of YOLO-based instance segmentation to isolate railway track regions from background elements such as ballast and stones.
- Application of supervised convolutional neural networks for detecting commonly occurring surface-level railway track defects.

- Integration of few-shot learning techniques to improve detection of rare defect classes with limited training samples.
- Incorporation of GPS-based location data using a Neo-7M module to associate detected defects with geographic coordinates.
- Visualization of inspection results through a dashboard interface to support maintenance and inspection activities.

The scope of this project is limited to **surface-level defect detection using computer vision** and **prototype-level development and testing**. Subsurface defect detection, ultrasonic testing, predictive maintenance, and large-scale field deployment are considered outside the scope of the present work but may be explored in future extensions.

2.3.2 Overall Description

2.3.2.1 Product Perspective

The proposed system is a **computer vision–based railway track defect detection solution** designed to support automated surface-level inspection under real-world conditions. The system is intended to **complement existing railway inspection practices** by providing a portable, modular, and data-driven inspection tool rather than replacing conventional inspection vehicles or non-destructive testing methods.

From a system architecture perspective, the solution follows a **client–server model** in which image acquisition and data transmission are performed at the client side, while computationally intensive processing is handled at a centralized server. This separation allows efficient use of hardware resources and supports scalable deployment.

The system can be logically viewed as a **three-layer architecture**:

1. **Data Acquisition Layer:**
   Consists of a Raspberry Pi integrated with a camera module and a Neo-7M GPS module. This layer is responsible for capturing railway track images and associated location data.

2. **Processing Layer:**
   Located at the server side, this layer performs rail region isolation using YOLO-based instance segmentation and applies deep learning models, including supervised CNNs and few-shot learning models, for surface defect detection.
3. **Application Layer:**
   Provides a dashboard-based interface for visualizing defect detection results along with associated metadata such as defect type and location. This layer supports inspection review and decision-making by maintenance personnel.

2.3.2.2 Product Features

The key features of the proposed system are as follows:

- **YOLO-Based Rail Segmentation:**
  Uses instance segmentation to accurately isolate railway track regions from complex backgrounds such as ballast and surrounding infrastructure, followed by binary masking for precise region extraction.
- **Surface Defect Detection:**
  Applies supervised convolutional neural networks to detect commonly occurring surface-level railway track defects such as cracks, flaking, spalling, squats, and shelling.
- **Few-Shot Learning for Rare Defects:**
  Incorporates a few-shot learning approach based on prototypical networks to improve detection performance for rare but safety-critical defect classes with limited training data.
- **Client–Server Processing Architecture:**
  Separates data acquisition from computation by performing image capture on a lightweight edge device and executing inference on a centralized server using REST-based communication.
- **GPS-Based Defect Localization:**
  Integrates location data obtained from a Neo-7M GPS module to associate detected defects with geographic coordinates, supporting traceability and inspection analysis.

- **Visualization Dashboard:**
  Presents defect detection results in a clear and interpretable format, enabling maintenance personnel to review inspection outcomes efficiently.

2.3.3 External Interface Requirements

This section describes the external interfaces through which users, hardware components, and software modules interact with the proposed railway track defect detection system.

2.3.3.1 User Interfaces

The system provides the following user interfaces to support configuration, monitoring, and result interpretation:

- **Web-Based Dashboard:**
  A web-based dashboard interface is provided to display defect detection results, including defect type and associated GPS location. The dashboard enables users to visually review inspection outcomes in an organized and interpretable manner.
- **Client-Side Configuration Interface:**
  A basic configuration interface on the Raspberry Pi allows users to manage camera settings, initiate image capture, and monitor data transmission status.

The system is designed for inspection support and decision-making; it does not include automated alerting mechanisms such as mobile notifications or email alerts in the current implementation.

2.3.3.1 Hardware Interfaces

The proposed system interfaces with the following hardware components:

- **Raspberry Pi:**
  Acts as the client device responsible for image acquisition, GPS data collection, and communication with the server through REST-based APIs.

- **Camera Module:**
  Connected to the Raspberry Pi for capturing high-resolution images of the railway track surface during inspection.
- **Neo-7M GPS Module:**
  Interfaces with the Raspberry Pi to provide geographic location data corresponding to each captured image.

No ultrasonic sensors, phased-array devices, or additional microcontrollers are used in the current system implementation.

2.3.3.2 Software Interfaces

The system interacts with the following software modules and interfaces:

- **YOLO-Based Segmentation Module:**
  Executes on the server to perform instance segmentation and generate binary masks for isolating railway track regions from background elements.
- **Defect Detection Modules:**
  Includes supervised CNN models for common defect detection and few-shot learning models for rare defect classification, executed on the server.
- **REST-Based Communication Interface:**
  Facilitates data exchange between the client (Raspberry Pi) and server, including image transmission and result retrieval.
- **Data Storage Interface:**
  Stores captured images, defect detection results, and associated GPS data for analysis and visualization.
- **Dashboard Application:**
  Provides visualization of defect detection outputs and inspection summaries for user review.

2.3.4 Other Non-functional Requirements

Non-functional requirements define the quality attributes and operational constraints of the proposed railway track defect detection system. These requirements ensure that the system performs reliably, safely, and securely under

realistic operating conditions.

2.3.4.1 Performance Requirements

The performance requirements of the system are defined to support practical inspection workflows while considering computational and network constraints:

- **Image Processing and Inference Latency:**
  The system is designed such that the average end-to-end processing latency, including image transmission, preprocessing, and defect detection at the server, remains within acceptable limits for inspection use. Experimental evaluation indicates an average processing latency of approximately **200–250 ms per image** under normal operating conditions.
- **Defect Detection Accuracy:**
  The supervised and few-shot learning models should provide reliable classification performance across the defined surface-level defect categories, including cracks, flaking, squats, spalling, and shelling, subject to dataset quality and environmental conditions.
- **Throughput Requirement:**
  The system should be capable of processing sequential image inputs without significant backlog when operating under standard inspection scenarios.
- **Dashboard Responsiveness:**
  The visualization dashboard should update inspection results and defect logs within a few seconds after receiving processed data from the server.

2.3.4.2 Safety Requirements

As the system is intended for use in railway inspection environments, the following safety considerations are applied:

- **Operational Safety:**
  The system is designed as a non-intrusive inspection aid that does not interfere with railway operations or infrastructure during image capture and testing.
- **Electrical Safety:**
  All hardware components, including the Raspberry Pi, camera module, and GPS unit, operate on regulated power supplies to minimize electrical hazards during operation.

- **Environmental Considerations:**
  The system is expected to operate in environments subject to dust, vibration, and variable lighting conditions. While industrial-grade protection is not implemented in the prototype, reasonable care is taken to ensure safe operation during controlled field testing.
- **User Safety:**
  Clear operational guidelines are assumed for users handling the device to avoid accidental damage to equipment or exposure to unsafe conditions.

2.3.4.3 Security Requirements

The system processes visual inspection data and GPS-based location information; therefore, basic security requirements are considered:

- **Data Access Control:**
  Access to the dashboard and system interfaces is restricted to authorized users to prevent unauthorized viewing or modification of inspection data.
- **Secure Communication:**
  Communication between the client device and the server is performed using structured API-based data exchange. Secure transmission mechanisms can be incorporated to protect data during transfer.
- **Data Integrity:**
  The system ensures that captured images and associated metadata are transmitted and stored without unintended modification during normal operation.
- **Privacy Considerations:**
  The collected image and GPS data are used strictly for inspection and academic evaluation purposes. No personal or sensitive user data is processed by the system.

2.4 Cost Analysis

Cost analysis is performed to estimate the approximate expenditure involved in developing the proposed railway track defect detection system. The objective is to

ensure that the system remains cost-effective while meeting functional and performance requirements suitable for a prototype-level implementation.

Since the proposed system focuses on computer vision–based surface defect detection and does not include ultrasonic or additional sensing hardware, the cost estimation is limited to essential components required for image acquisition, location tagging, and system support.

Estimated Cost Breakdown

Component	Quantity	Unit Cost (INR)	Total Cost (INR)
Raspberry Pi 4 (4 GB / 8 GB)	**1**	**8,000**	**8,000**
Pi Camera Module (v2.1 / HQ Camera)	**1**	**400**	**400**
Neo-7M GPS Module	**1**	**650**	**650**
Power Supply / Battery Pack	**1**	**700**	**700**
Supporting Electronics (wires, connectors, casing)	**—**	**2,500**	**2,500**
Miscellaneous (mounts, testing accessories)	**—**	**2,000**	**2,000**

Table 1: Cost Analysis of Components

Estimated Total Cost: ≈ ₹14,250 /-

This cost estimation reflects a prototype-scale implementation intended for academic evaluation and controlled testing. The system is designed to be modular, allowing future enhancements or scaling with minimal additional cost.

2.5 Risk Analysis

Risk analysis is an essential step in evaluating potential challenges that may affect the successful development, deployment, and performance of the railway track defect detection system. As the proposed system integrates hardware components, computer vision models, and network-based data processing, certain technical and operational risks must be considered.

The identified risks, their potential impact, and mitigation strategies are discussed below.

4. **Technical Risks**

Risk: Limited detection accuracy for rare defect classes due to dataset imbalance.
Impact: Rare but safety-critical defects may be missed during inspection.
Mitigation: Adoption of few-shot learning techniques and periodic model retraining using newly collected data.

Risk: Reduced model performance under challenging environmental conditions such as poor lighting or shadows.
Impact: Inconsistent detection results in real-world scenarios.
Mitigation: Use of rail-aware preprocessing and controlled data collection to improve robustness.

5. **Data and Communication Risks**

Risk: Network latency or temporary connectivity issues during data transmission between the client device and server.
Impact: Delay in receiving inspection results.
Mitigation: Design of the system to support asynchronous data transmission and inspection review rather than strict real-time dependency.

Risk: Loss or corruption of captured image data during transfer or storage.
Impact: Incomplete inspection records.
Mitigation: Structured data handling, validation checks, and secure storage practices.

6. **Operational Risks**

Risk: Improper camera positioning or image capture during inspection.
Impact: Poor image quality leading to inaccurate defect detection.
Mitigation: Clear operational guidelines for image acquisition and basic configuration checks on the client device.

Risk: Limited familiarity of users with AI-based inspection tools.
Impact: Inefficient interpretation of system outputs.
Mitigation: Use of a simple and intuitive dashboard interface with clearly presented results.

7. **Financial Risks**

Risk: Increase in component costs due to availability or supply constraints.
Impact: Slight increase in prototype development cost.
Mitigation: Use of commonly available and low-cost hardware components.

Risk: Additional expenses for system refinement or testing.
Impact: Budget extension during later stages of the project.
Mitigation: Modular system design that allows incremental improvements.

8. **Environmental Risks**

Risk: Exposure of hardware components to dust, vibration, or weather variations during field testing.
Impact: Reduced hardware reliability.
Mitigation: Controlled testing environments and protective casing for electronic components.

6. Project Management Risks

Risk: Delays in dataset preparation or model training.

Impact: Extension of project timeline.

Mitigation: Structured planning, parallel task execution, and early testing of system modules.

3 Methodology Adopted

3.1 Investigative Techniques

To develop a reliable computer vision–based railway track defect detection system, a combination of descriptive, comparative, and experimental investigative techniques was adopted. These techniques supported systematic problem understanding, model selection, and system validation under practical constraints.

9. Descriptive Technique

Purpose:

To understand existing railway inspection practices and identify limitations in automated defect detection systems.

Application:

A detailed review of research papers, technical reports, and industry practices related to railway inspection, computer vision–based defect detection, and deep learning was conducted. The study focused on surface defect detection methods, segmentation approaches, and dataset challenges.

Key Observations:

- Manual inspections are time-consuming and dependent on human judgment.
- Vision-based methods are effective for surface defects but suffer from background interference.
- Most systems rely on supervised learning and struggle with rare defect classes.

Outcome:

This analysis helped identify key gaps related to background noise removal, class imbalance, and deployment feasibility, which directly influenced the proposed system design.

10. Comparative Technique

Purpose:

To evaluate alternative machine learning approaches and system architectures suitable for railway defect detection.

Application and Findings:

- **Rail Region Extraction:** Segmentation-based rail isolation was preferred over direct classification on raw images to reduce background interference.
- **Defect Detection Models:** Conventional CNNs were found effective for common defects, while few-shot learning techniques showed better performance for rare defect classes.
- **Deployment Strategy:** A client–server architecture was chosen over full edge inference due to computational limitations of edge devices.

Outcome:

This investigation resulted in the selection of a rail-aware preprocessing pipeline combined with supervised and few-shot learning models, implemented within a client–server framework.

11. Experimental Technique

Purpose:

To implement, test, and validate the proposed defect detection pipeline.

Application:

- Image datasets were prepared and used to train rail segmentation and defect detection models.
- Binary segmentation masks were applied to isolate railway track regions.

- Defect detection models were evaluated on extracted rail images under realistic conditions.
- End-to-end data flow between the client device and server was tested.

Outcome:

The experimental evaluation confirmed that rail-aware preprocessing improves detection reliability and that few-shot learning enhances recognition of rare defect classes.

3.2 Proposed Solution

The proposed solution is a **computer vision–based railway track defect detection system** designed to support automated inspection of surface-level defects under real-world conditions. The system aims to reduce reliance on manual inspection by providing a structured, data-driven inspection pipeline.

System Overview

- **Rail Region Isolation:**
  YOLO-based instance segmentation is used to identify railway track regions. The segmentation output is converted into a binary mask to extract only rail surfaces for further analysis.
- **Surface Defect Detection:**
  Extracted rail images are analyzed using deep learning models. A supervised CNN detects commonly occurring defects, while a few-shot learning model improves detection of rare defect classes.
- **Localization:**
  GPS data obtained from a Neo-7M module is associated with each inspection image to provide location context.
- **System Architecture:**
  Image acquisition and GPS data collection are handled by a Raspberry Pi at the client side. Computationally intensive processing is performed on a centralized server.

- **Visualization:**
  Detection results are presented through a dashboard interface to assist inspection review and maintenance planning.

This approach provides a practical balance between detection accuracy, computational efficiency, and deployment feasibility.

3.3 Work Breakdown Structure

The project was executed in a structured manner through the following phases:

Phase I – Requirement Analysis and Dataset Preparation

- Literature survey and problem analysis
- Collection and preprocessing of railway track images
- Dataset preparation and annotation

Deliverable: Prepared dataset and defined system requirements

Phase II – Model Development

- Training of rail segmentation model
- Development of CNN-based defect detection model
- Implementation of few-shot learning for rare defects

Deliverable: Trained segmentation and defect detection models

Phase III – System Integration

- Implementation of client–server communication
- Integration of GPS data with inspection images
- Application of binary masking and inference pipeline

Deliverable: Integrated defect detection pipeline

Phase IV – Testing and Evaluation

- Functional testing of the end-to-end system

- Performance evaluation under realistic constraints
- Visualization of inspection results

Deliverable: Validated prototype system

3.4 Tools and Technologies Used

The tools and technologies selected for this project were chosen to ensure accuracy, simplicity, and feasibility within academic and practical constraints.

Hardware Components

- **Raspberry Pi:** Image acquisition and data transmission
- **Camera Module:** Capturing railway track images
- **Neo-7M GPS Module:** Location tagging of inspection data

Software Tools

- **Python:** Core programming language
- **OpenCV:** Image preprocessing
- **Roboflow:** Dataset annotation and preparation
- **FastAPI / Flask:** Server-side API implementation
- **NumPy / Pandas:** Data handling and analysis

Machine Learning Frameworks

- **YOLO (Segmentation):** Rail region extraction
- **CNN Models:** Surface defect classification
- **Prototypical Networks:** Few-shot learning for rare defects
- **Scikit-learn:** Evaluation and metrics

Visualization Tools

- **Web-Based Dashboard:** Display of defect detection results and location data.

4 -Design Specifications

4.1 System Architecture

- System Block Diagram

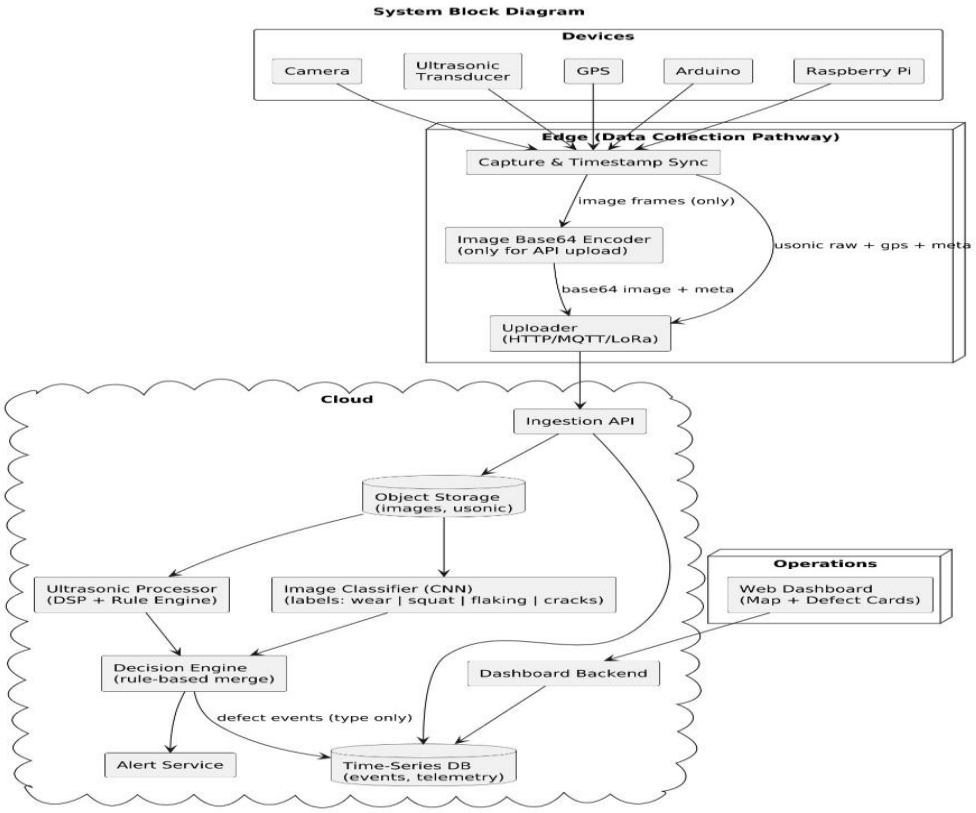


Fig 1: System Block Diagram

- Technology Stack

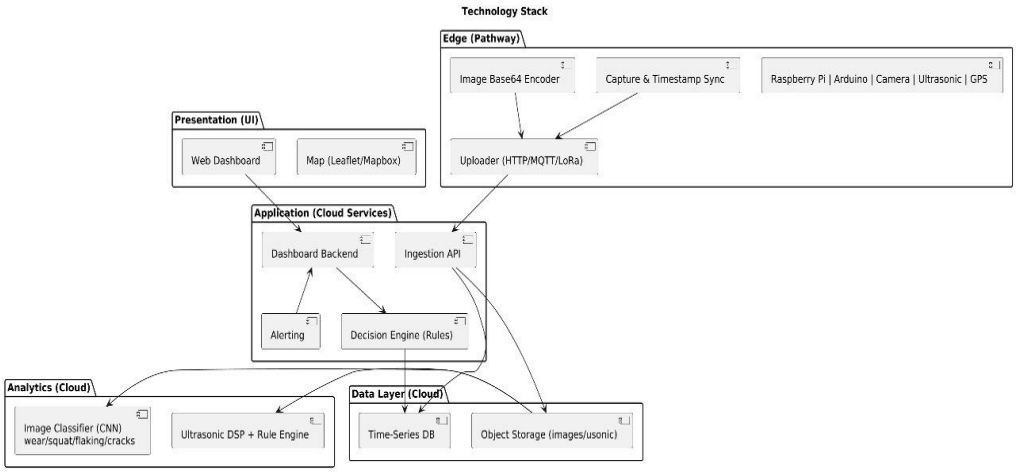


Fig 2: Technology Stack

- MVC/Tier Architecture

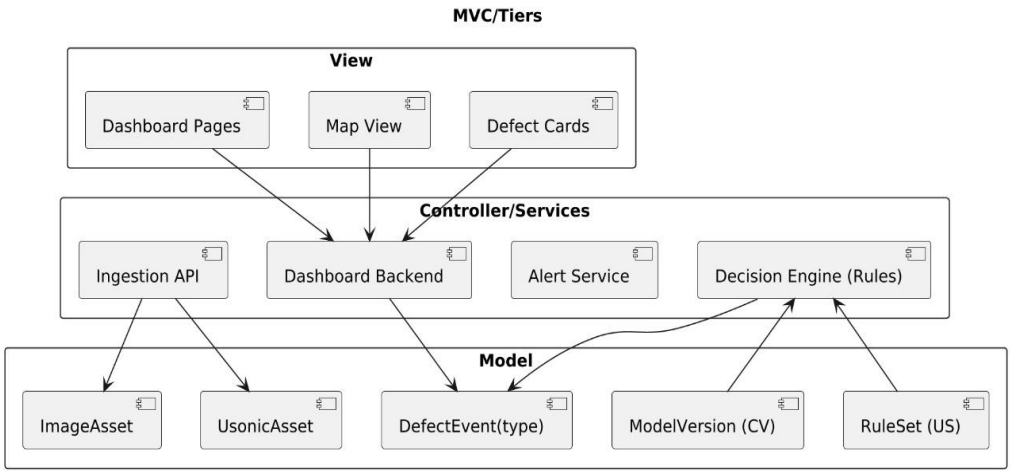


Fig 3: MVC/Tier Architecture

4.2 Design Level Diagrams

- Deployment Diagram

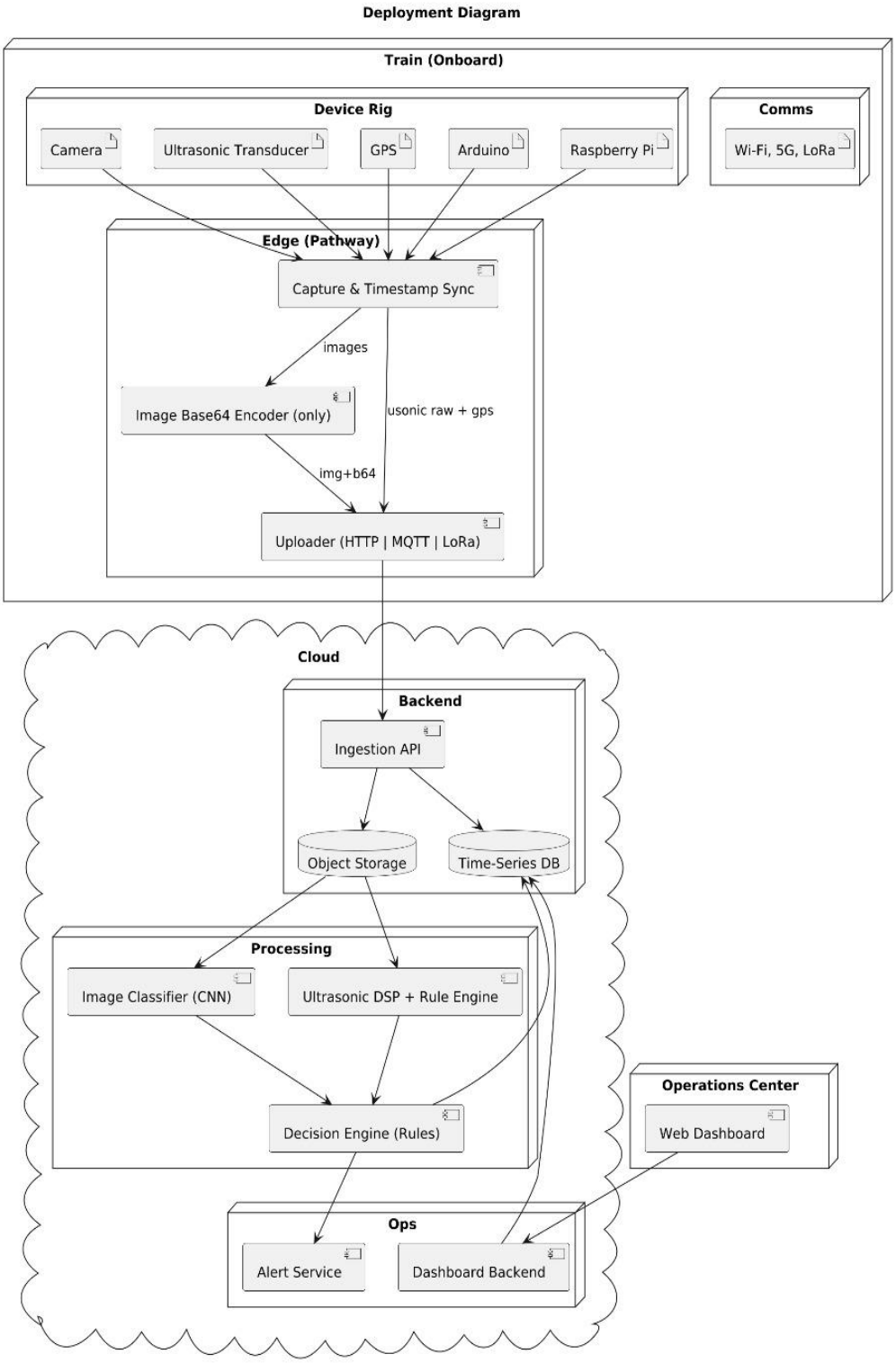


Fig 4: Deployment Diagram

- Sequence Diagram

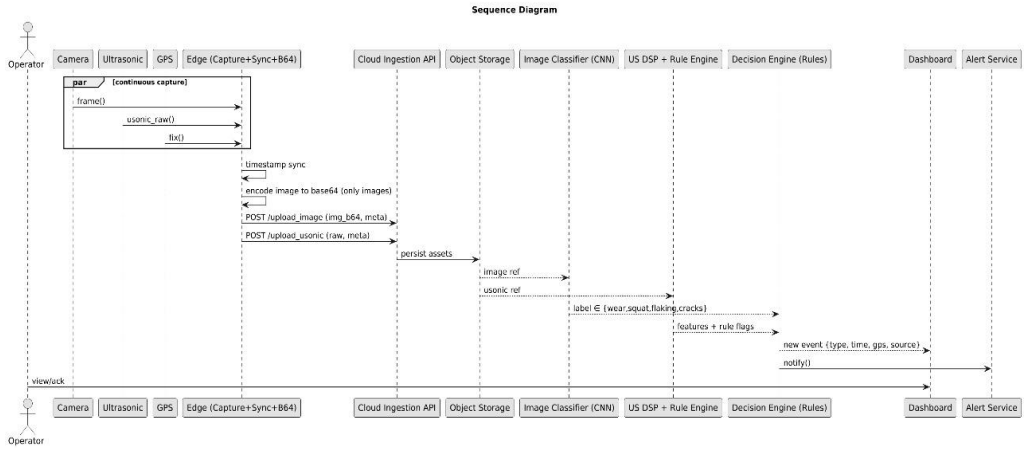


Fig 5: Sequence Diagram

- Component Diagram

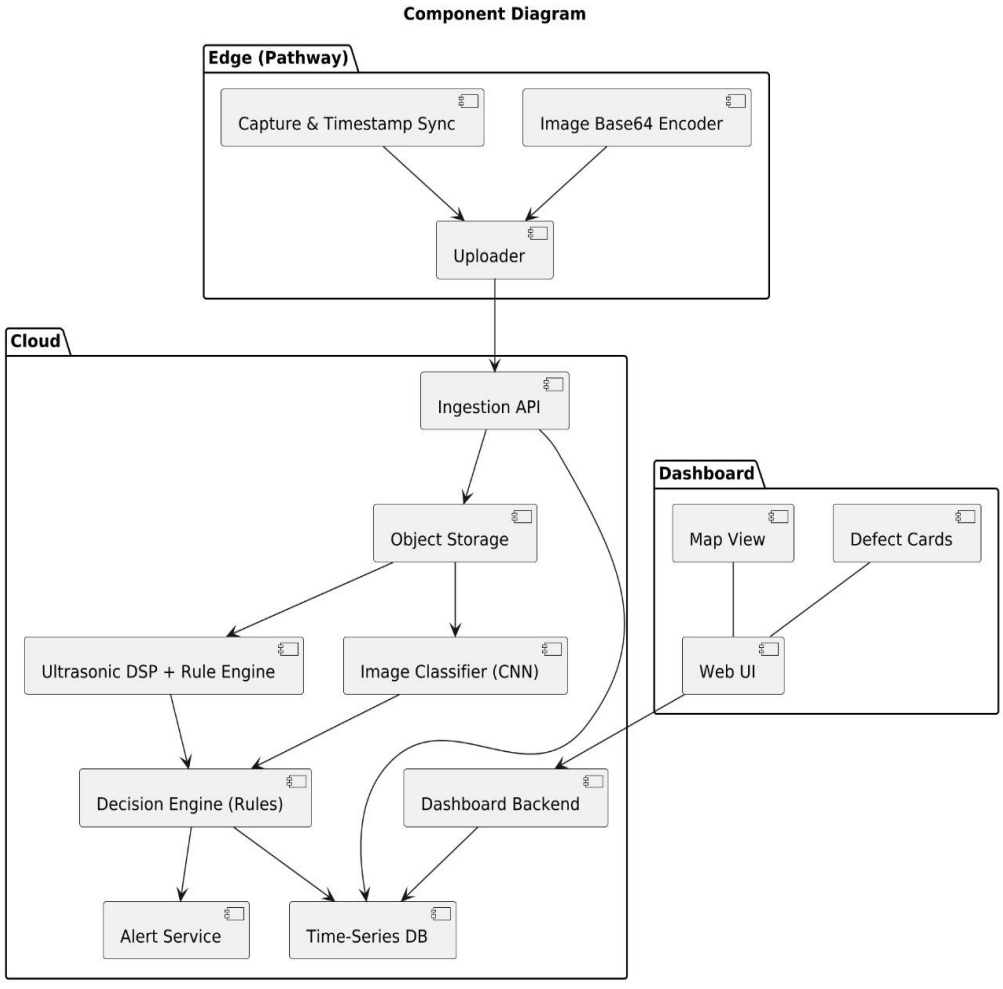


Fig 6: Component Diagram

- Activity Diagram

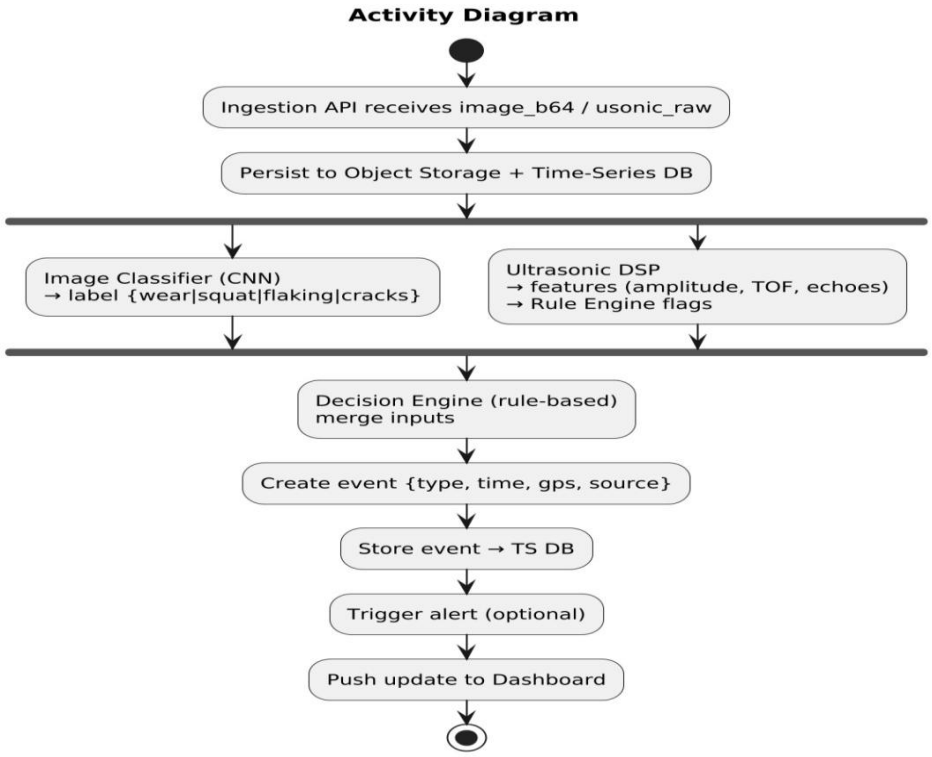


Fig 7: Activity Diagram

- Use-Case Diagram

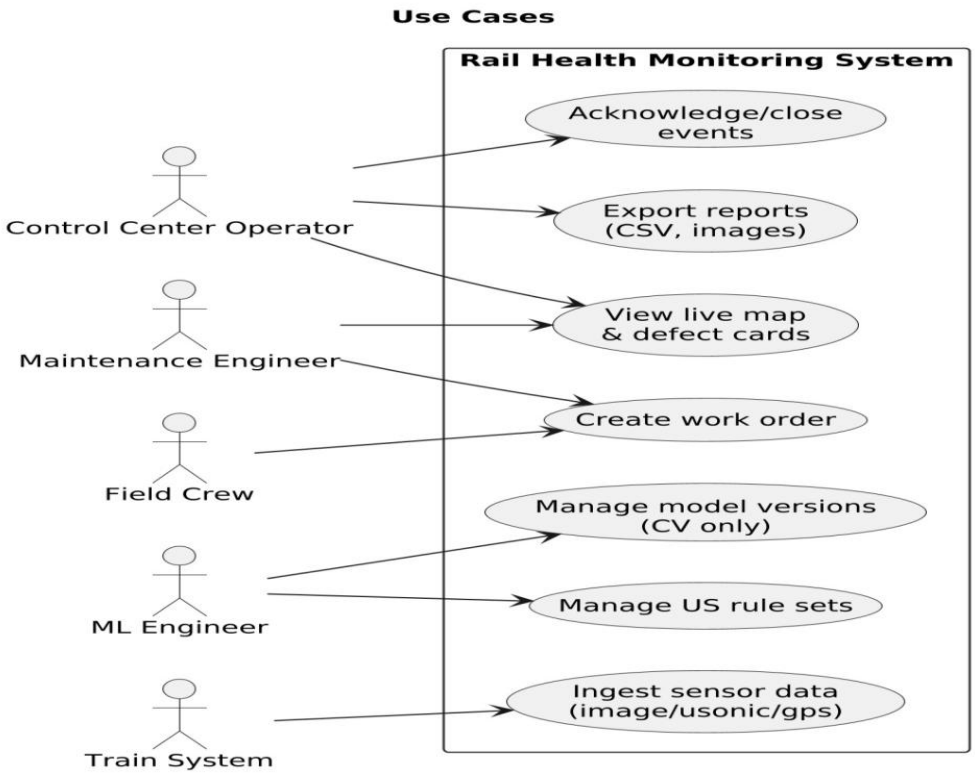


Fig 8: Use-Case Diagram

- Class Diagram

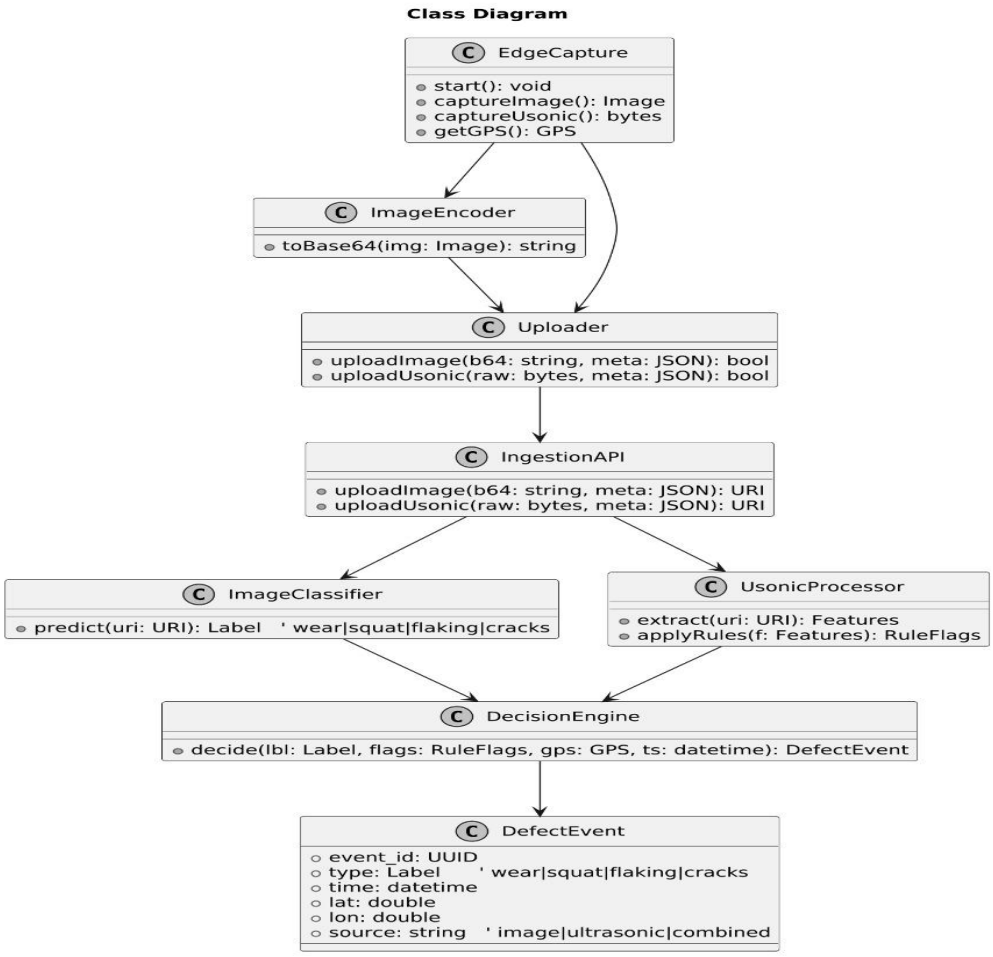


Fig 9: Class Diagram

- ER Diagram

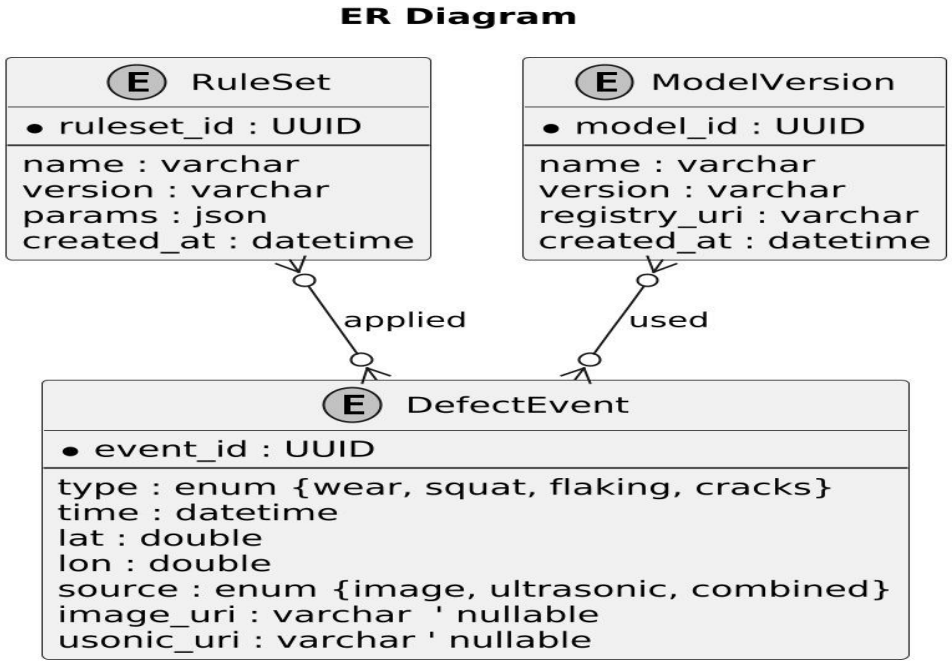


Fig 10: ER Diagram

- DFD Level-0

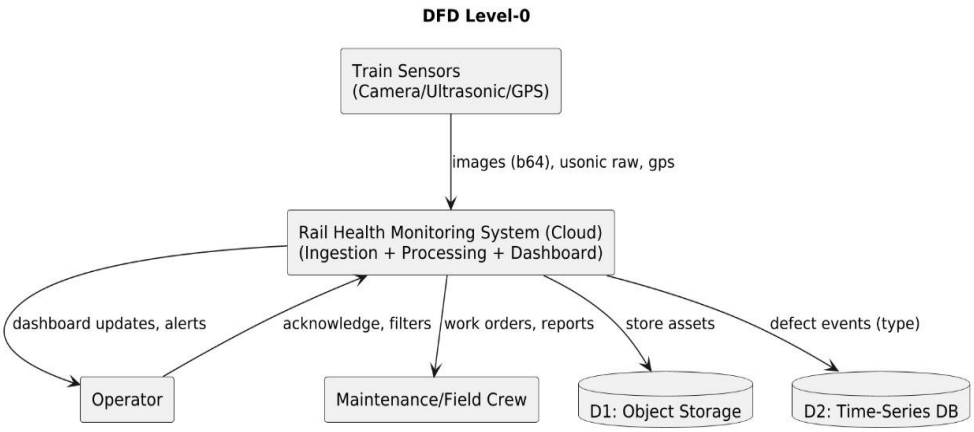


Fig 11: DFD Level-0

4.3 User Interface Diagrams

- Dashboard Wireframe

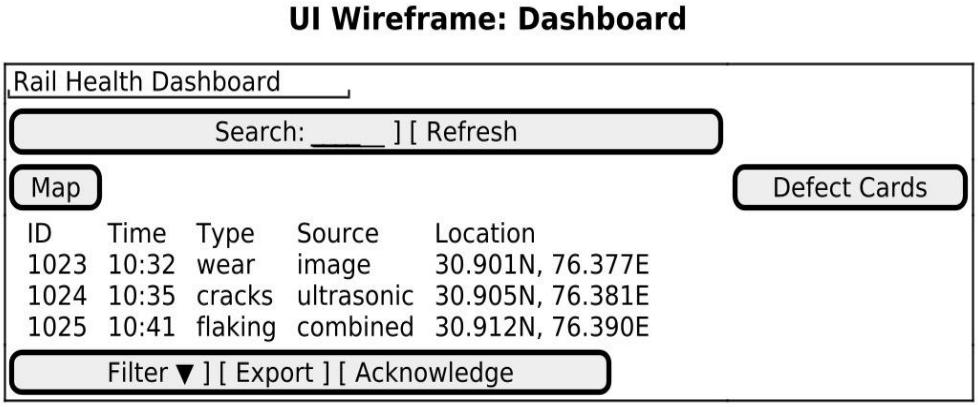


Fig 12: Dashboard Wireframe

- Defect Detail Wireframe

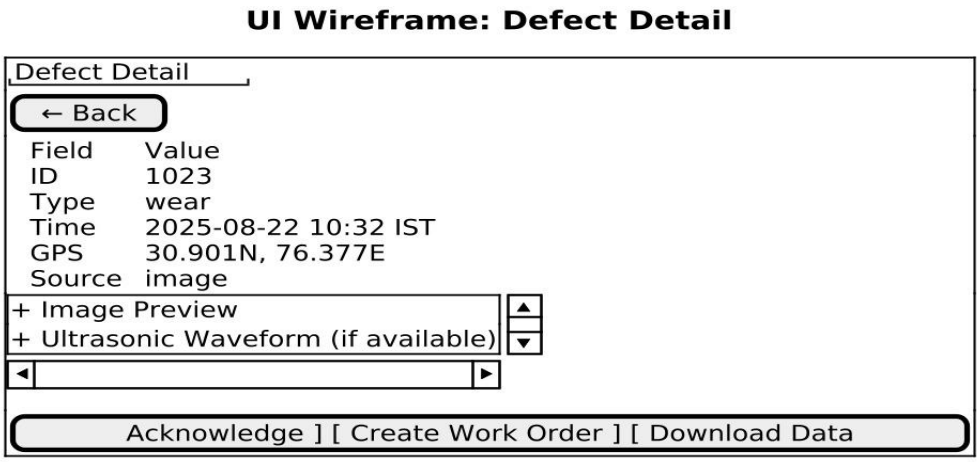


Fig 13: Defect Detail Wireframe

5 Implementation And Experimental Results

5.1 Experimental Setup

The experimental setup for this project was designed to evaluate the effectiveness and practicality of the proposed computer vision–based railway track defect detection system under realistic conditions. The setup focuses on validating the performance of the rail segmentation and defect detection pipeline while considering hardware and deployment constraints.

The system follows a **client–server architecture**. Image acquisition was performed using a Raspberry Pi equipped with a camera module, which acted as the client device. The Raspberry Pi was responsible for capturing railway track images and collecting GPS data using a Neo-7M GPS module. Due to limited computational resources on the edge device, no deep learning inference was performed locally.

Captured images and associated metadata were encoded and transmitted to a centralized server using REST-based APIs. The server environment was configured with Python-based deep learning frameworks and sufficient computational resources to execute segmentation and defect detection models efficiently.

On the server side, a YOLO-based instance segmentation model was used to generate pixel-level confidence maps for railway track regions. These confidence maps were converted into **binary segmentation masks** using a fixed threshold, ensuring that only rail surface regions were retained for further analysis. The extracted rail images were then passed through defect detection models for classification.

The experimental results were visualized through a web-based dashboard, which displayed detected defect types along with associated location information. All experiments were conducted in a controlled testing environment using collected image datasets to ensure consistency and repeatability of results.

5.2 Experimental Analysis

The experimental analysis evaluates the performance of the proposed system in terms of data handling, defect detection accuracy, and overall system responsiveness. The analysis focuses on both the effectiveness of rail-aware preprocessing and the performance of the defect classification models.

5.2.1Data

Data Sources:

The dataset used in this project consists of railway track surface images collected from real railway environments and curated research sources. The dataset includes images representing five surface-level defect categories: cracks, flaking, spalling, squats, and shelling. The dataset reflects real-world conditions, including background clutter, lighting variations, and class imbalance.

Data Cleaning:

Raw images were reviewed to remove corrupted, blurred, or irrelevant samples that could negatively impact model training and evaluation. Images with extreme noise or insufficient visibility of the rail surface were excluded to maintain dataset quality.

Data Pruning:

To avoid redundancy and bias, visually similar images were pruned where necessary. This step helped ensure diversity in the dataset and reduced overfitting during model training.

Feature Extraction Workflow:

Rather than relying on handcrafted features, the system uses deep learning models to automatically extract discriminative features from input images. Before feature extraction, rail-aware preprocessing is applied. YOLO-based segmentation isolates the rail region, and binary masking removes all background elements. The extracted rail images are then fed into the defect detection models, which learn relevant visual features internally through convolutional layers.

This workflow ensures that the models focus on defect-relevant information while minimizing the influence of background noise.

5.2.2 Performance Parameters

The performance of the proposed system was evaluated using appropriate accuracy-based and system-level parameters relevant to image-based defect detection.

Accuracy Measures:

- **Classification Accuracy:** Used to evaluate the overall correctness of defect classification across the defined defect categories.
- **Precision:** Measures the proportion of correctly identified defect samples among all samples predicted as defective.
- **Recall:** Evaluates the system's ability to correctly detect actual defect samples, particularly important for safety-critical defects.
- **F1-Score:** Provides a balanced measure by combining precision and recall, especially useful in the presence of class imbalance.

System Performance Parameters:

- **Processing Latency:** The average time required to process an image from server receipt to defect classification was measured. Experimental observations indicate an average latency of approximately **200–250 ms per image**, which is suitable for inspection support applications.
- **Dashboard Response Time:** The time taken for results to appear on the visualization interface after processing was observed to be within a few seconds, ensuring usability for inspection review.
- **Reliability of Rail Segmentation:** The effectiveness of binary rail masking was evaluated qualitatively by observing the reduction in false detections caused by background elements.

The experimental results demonstrate that rail-aware preprocessing significantly improves defect detection reliability and that the combination of supervised and few-shot learning models effectively handles both common and rare defect classes.

5.3 Working of the project

This section explains the operational working of the proposed railway track defect detection system, detailing how data flows through the system, how algorithms are applied, and how the system is deployed and visualized in practice. The explanation focuses on the end-to-end functioning of the project from image acquisition to result presentation.

5.3.1 Procedural Workflow

Step 1: Image Acquisition and Location Capture

The process begins at the client side, where a Raspberry Pi equipped with a camera module captures images of the railway track surface. Along with each image, geographic location data is obtained using a Neo-7M GPS module. This ensures that every inspection image can be associated with a precise location for later analysis and reporting.

The captured image and GPS data together form a single inspection record.

Step 2: Data Encoding and Transmission

Due to limited computational resources on the client device, no deep learning inference is performed locally. Instead, the captured image is encoded (for example, using Base64 encoding) and transmitted to a centralized server using REST-based APIs. GPS coordinates and timestamp information are sent along with the image data.

This client–server separation reduces edge-device load and enables scalable processing.

Step 3: Rail Region Isolation

Upon receiving the data, the server initiates the preprocessing stage. A YOLO-based instance segmentation model processes the input image and generates pixel-level confidence values indicating the probability of each pixel belonging to the railway track.

A fixed confidence threshold is applied to convert these probabilities into a **binary segmentation mask**. Pixels classified as rail regions are retained, while all background pixels (ballast, sleepers, surrounding objects) are masked out. This step ensures that subsequent analysis focuses only on relevant rail surfaces.

Step 4: Rail Image Extraction

The binary mask is applied to the original image to extract a clean rail-only image. This extracted rail region significantly reduces background interference and improves the reliability of defect detection models.

Step 5: Defect Detection and Classification

The extracted rail image is passed to the defect detection stage. Two complementary learning approaches are used:

- A **supervised CNN model** detects commonly occurring surface-level defects.
- A **few-shot learning model (prototypical network)** improves recognition of rare defect classes with limited training data.

The system selects the appropriate classification output based on the trained models and produces a final defect label.

Step 6: Result Aggregation and Visualization

The detected defect type, along with GPS coordinates and timestamp, is stored on the server. The results are then displayed through a web-based dashboard interface. This allows users to visually review inspection outcomes and understand defect locations clearly.

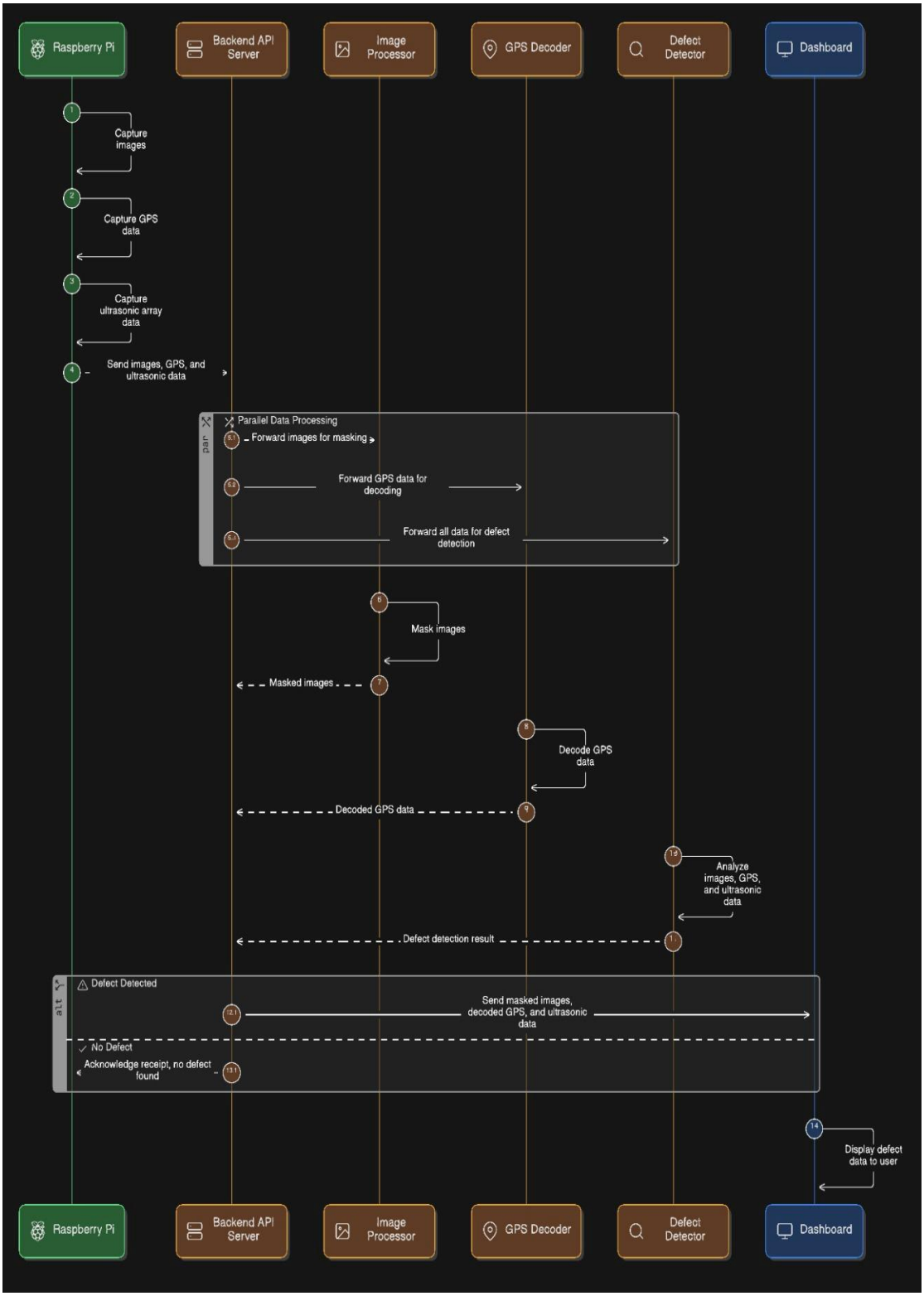


Fig 14: Workflow of the Project

5.3.2 Algorithmic Approaches Used

This project employs multiple algorithmic approaches to achieve accurate and robust defect detection. The major algorithms used are summarized below.

Algorithm 1: Rail Region Segmentation and Masking

Purpose:

To isolate railway track regions from complex backgrounds before defect detection.

Approach:

A YOLO-based instance segmentation model generates pixel-wise confidence values. These values are thresholded to create a binary mask, which is then applied to the original image.

Pseudocode:

Input: Original railway image I
Output: Extracted rail-only image R

1. seg_mask ← YOLO_Segmentation(I)
2. binary_mask ← seg_mask > threshold
3. R ← I × binary_mask
4. Return R

Explanation:

This algorithm ensures that only pixels belonging to the rail surface are retained, reducing background-induced false detections.

Algorithm 2: Supervised Defect Classification

Purpose:

To detect commonly occurring surface-level railway defects.

Approach:

A convolutional neural network processes the extracted rail image and predicts the defect class based on learned visual features.

Pseudocode:

Input: Rail-only image R
Output: Defect class C

1. features ← CNN_FeatureExtractor(R)
2. C ← Softmax_Classifier(features)
3. Return C

Algorithm 3: Few-Shot Learning for Rare Defects

Purpose:
To improve detection of rare defect classes with limited training samples.

Approach:
A prototypical network learns class prototypes in an embedding space and classifies new samples based on distance to these prototypes.

Pseudocode:

Input: Rail image R, class prototypes P
Output: Defect class C

1. embedding ← Encoder(R)
2. C ← argmin(distance(embedding, P))
3. Return C

Explanation:
This approach allows the system to generalize better when only a small number of examples are available for certain defect types.

5.3.3 Project Deployment

The proposed system is deployed using a **client–server architecture**, which supports modularity and scalability.

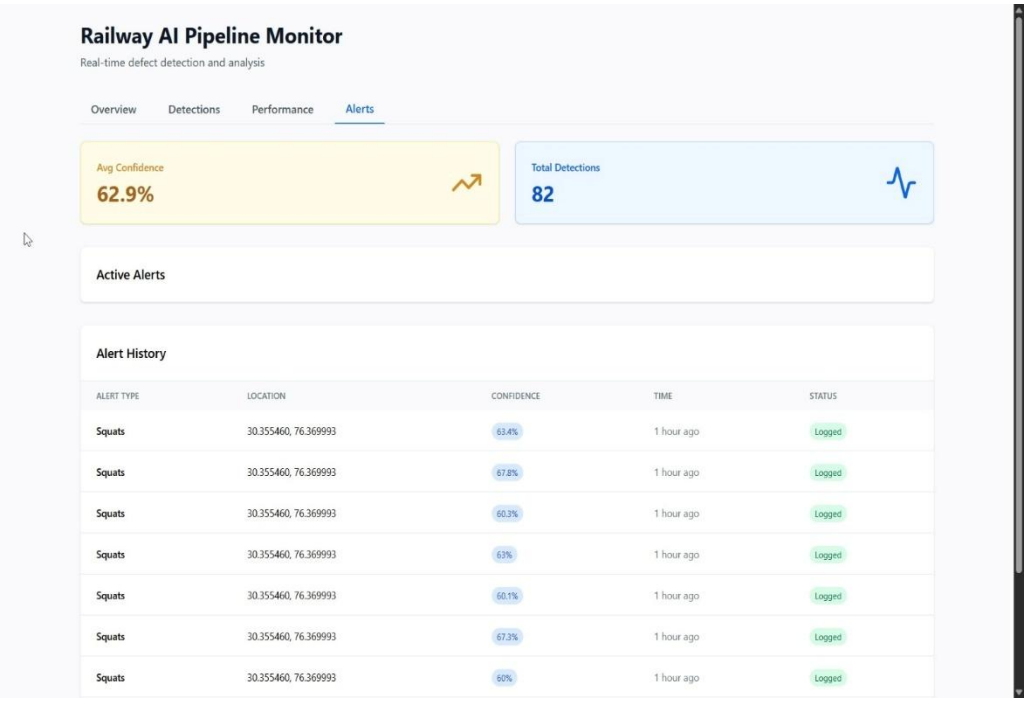
Client Side (Edge Device)

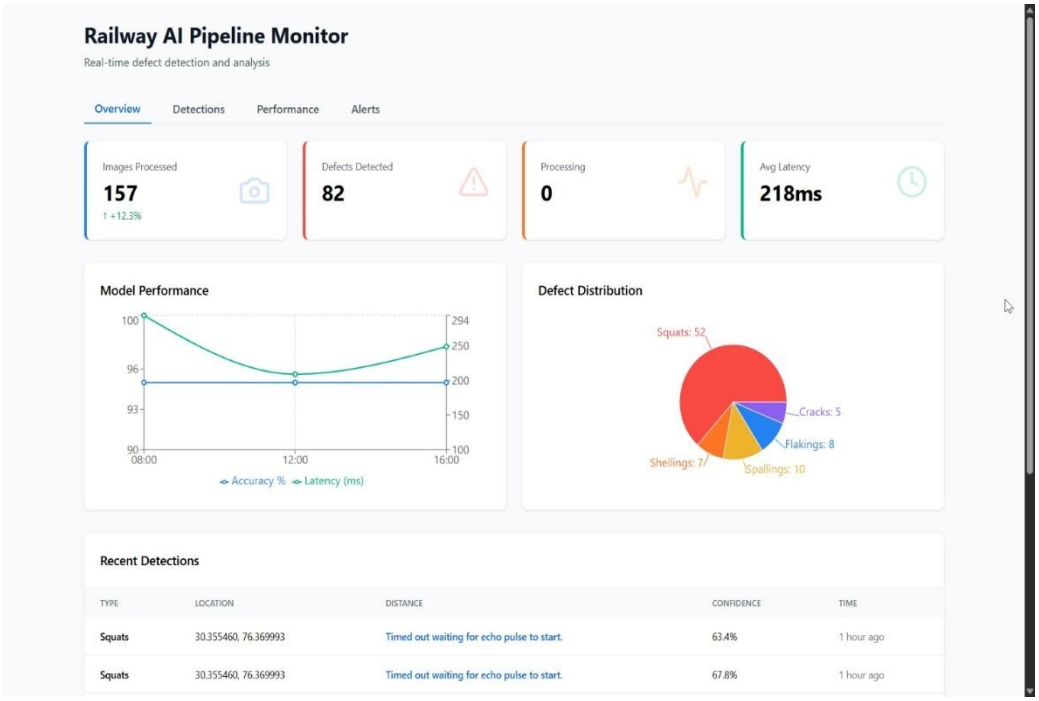
- Raspberry Pi for image acquisition
- Camera module for capturing rail images
- GPS module for location tagging
- REST API client for data transmission

Server Side

- YOLO-based segmentation model
- Binary masking and preprocessing pipeline
- CNN and few-shot learning models for defect detection
- Data storage and result management
- Web-based dashboard for visualization

5.3.4 System Screenshots





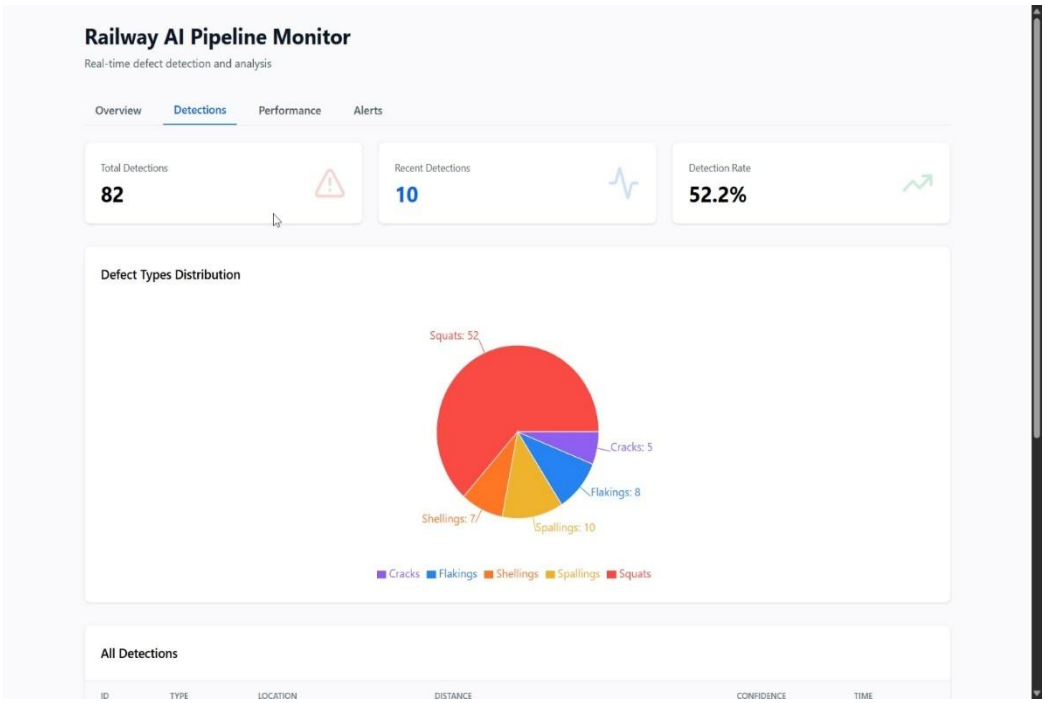


Fig 15: Dashboard Images

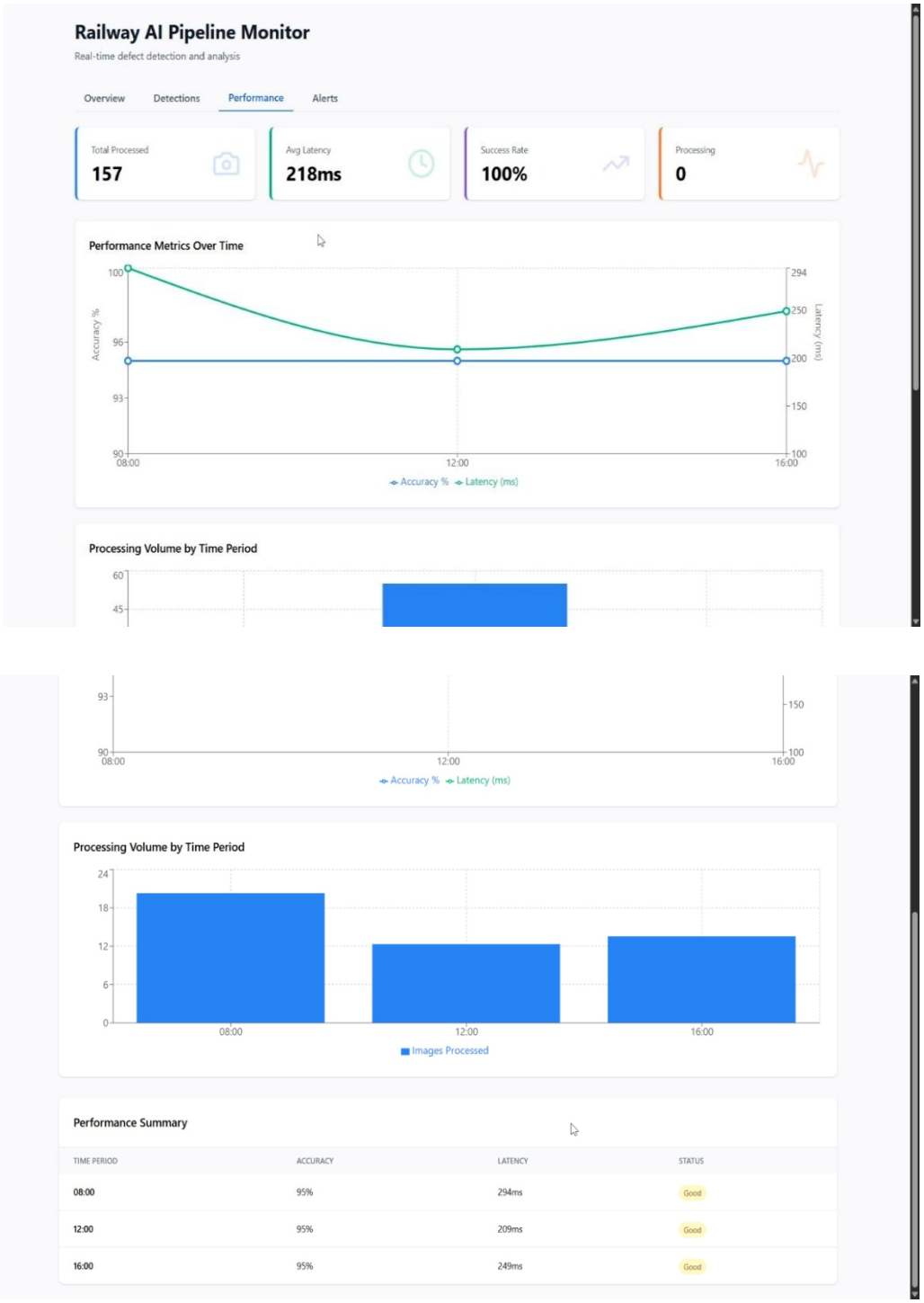


Fig 16: Dashboard Analysis

5.4 Testing Process

Testing plays a crucial role in validating the correctness, reliability, and usability of the proposed railway track defect detection system. Since the system integrates image acquisition, deep learning–based processing, server-side inference, and result visualization, testing was carried out at multiple levels to ensure that each functional component operates as intended and that the complete system performs reliably under realistic conditions.

The testing process focused on verifying functional correctness, evaluating performance, and identifying potential failure points during system operation.

5.4.1 Test Plan

Testing plays a crucial role in validating the correctness, reliability, and usability of the proposed railway track defect detection system. Since the system integrates image acquisition, deep learning–based processing, server-side inference, and result visualization, testing was carried out at multiple levels to ensure that each functional component operates as intended and that the complete system performs reliably under realistic conditions.

The testing process focused on verifying functional correctness, evaluating performance, and identifying potential failure points during system operation.

5.4.2 Features to be tested

The test plan outlines the approach adopted to systematically verify the functionality and performance of the proposed system. The primary objectives of the test plan are:

- To verify correct image acquisition and GPS data capture at the client side
- To validate successful data transmission between the client device and server
- To evaluate the accuracy and consistency of rail segmentation and defect detection
- To ensure correct visualization of results on the dashboard

Testing was conducted incrementally, starting from individual modules and progressing towards end-to-end system testing. All tests were performed using collected railway track images under controlled experimental conditions.

5.4.3 Test Strategy

A **combination of functional testing and integration testing** was adopted for this project.

- **Module-Level Testing:**
  Individual modules such as image capture, segmentation, and defect classification were tested independently to ensure correct functionality.
- **Integration Testing:**
  Interactions between the client device, server-side processing pipeline, and dashboard interface were tested to verify correct data flow.
- **System-Level Testing:**
  End-to-end testing was performed by capturing images, processing them through the complete pipeline, and validating the final outputs.

The test strategy emphasized correctness and reliability rather than stress testing, as the project is designed for prototype-level validation.

5.4.4 Test Techniques

The following testing techniques were employed:

- **Black-Box Testing:**
  Used to verify system behavior based on input-output observations without examining internal implementation details.
- **Functional Testing:**
  Ensured that each feature performed according to the defined requirements.
- **Manual Testing:**
  Manual inspection was used to validate segmentation quality, defect labels, and dashboard outputs.

- **Regression Testing:**

  Repeated testing was carried out after model updates or pipeline changes to ensure previously working features remained unaffected.

5.4.5Test Cases

Test Case ID	Description	Input	Expected Outcome	Result
TC-01	Image capture test	Railway track image	Image captured successfully	Pass
TC-02	GPS data capture	Image + GPS module	Correct coordinates attached	Pass
TC-03	Data transmission	Encoded image	Image received by server	Pass
TC-04	Rail segmentation	Raw railway image	Rail region isolated correctly	Pass
TC-05	Defect detection	Extracted rail image	Correct defect classification	Pass
TC-06	Dashboard display	Detection result	Output displayed correctly	Pass

Table 2: Test Cases

5.4.6 Test Results

The test results indicate that the proposed railway track defect detection system operates reliably under the defined experimental conditions. All major functional components passed their respective test cases without critical failures.

Rail-aware preprocessing using segmentation significantly reduced background interference, resulting in improved defect detection consistency. Data transmission between the client and server was stable, and the dashboard interface displayed inspection results clearly and accurately.

Minor variations in detection accuracy were observed under challenging lighting conditions; however, these did not affect overall system functionality. The testing process confirms that the system meets its functional requirements and is suitable for prototype-level deployment and academic evaluation.

5.5 Results and Discussions

This section discusses the results obtained from implementing and testing the proposed computer vision–based railway track defect detection system. The evaluation focuses on the effectiveness of rail-aware preprocessing, the performance of defect detection models, and the overall system behavior under realistic testing conditions.

The experimental results demonstrate that isolating the railway track region prior to defect detection significantly improves classification reliability. By removing background elements such as ballast and surrounding infrastructure, the system reduces false detections that commonly occur when models are applied directly to raw images.

The supervised CNN model showed consistent performance in detecting commonly occurring surface-level defects such as spalling, flaking, squats, and shelling. These defect classes were well represented in the dataset, allowing the model to learn stable visual patterns. For rare defect categories, particularly cracks with limited training samples, the few-shot learning approach improved recognition by leveraging distance-based classification in an embedding space.

From a system perspective, the client–server architecture proved effective in balancing computational load and operational feasibility. Offloading inference to the server ensured stable performance without overloading the edge device. The average processing latency observed during experiments remained within acceptable limits for inspection support applications, confirming the practicality of the chosen architecture.

The visualization dashboard successfully presented defect detection results along with corresponding GPS coordinates, enabling clear inspection review. While environmental factors such as lighting variations influenced detection confidence in some cases, the system maintained consistent functionality across test scenarios.

Overall, the results validate the proposed approach and demonstrate that rail-aware preprocessing combined with data-efficient learning strategies enhances the reliability of automated railway track defect detection.

5.6 Inferences Drawn

Based on the experimental results and system evaluation, the following key inferences can be drawn:

1. Rail-aware preprocessing using segmentation-based masking significantly reduces background interference and improves defect detection consistency.
2. Conventional supervised learning models are effective for detecting frequently occurring railway track defects when sufficient data is available.
3. Few-shot learning techniques provide a practical solution for handling rare defect classes in highly imbalanced railway datasets.
4. A client–server architecture offers a reliable and scalable approach for deploying vision-based inspection systems without overburdening edge devices.
5. Visual presentation of results through a dashboard enhances interpretability and supports inspection decision-making.
6. Environmental factors such as lighting and image quality influence detection confidence, highlighting the importance of controlled data acquisition.

These inferences confirm that addressing real-world challenges such as background noise and data imbalance is essential for practical railway inspection systems.

5.7 Validation of Objectives

The objectives defined at the beginning of the project were systematically validated through implementation and experimental evaluation, as summarized below:

- **Objective:** Study and analyze existing railway inspection techniques
  Validation: Achieved through an extensive literature survey and requirement analysis.

- **Objective:** Develop a rail-aware preprocessing mechanism
  Validation: Successfully implemented using YOLO-based segmentation and binary masking, which improved defect detection reliability.
- **Objective:** Implement deep learning–based defect detection models
  Validation: CNN-based models were trained and evaluated for surface defect classification with satisfactory performance.
- **Objective:** Address class imbalance using few-shot learning
  Validation: Few-shot learning models demonstrated improved recognition of rare defect classes.
- **Objective:** Design a deployment-oriented system architecture
  Validation: A client–server architecture was implemented and tested successfully.
- **Objective:** Provide intuitive visualization of inspection results
  Validation: A web-based dashboard was developed to display defect detection outcomes and location information.
- **Objective:** Ensure robustness under realistic constraints
  Validation: The system functioned reliably during experimental testing with acceptable processing latency.

The successful validation of these objectives confirms that the proposed system meets its intended goals and provides a solid foundation for future enhancements.

6 Conclusions And Future Directions

6.1 Conclusions

This project successfully demonstrates the application of computer vision and deep learning techniques for automated detection of surface-level railway track defects under realistic operating conditions. The work addresses key challenges commonly encountered in vision-based railway inspection, including background interference caused by ballast and surrounding structures, as well as the presence of highly imbalanced defect datasets.

A rail-aware preprocessing mechanism based on instance segmentation was implemented to isolate railway track regions prior to defect analysis. This approach proved effective in reducing background noise and improving the reliability of

defect detection. The integration of supervised convolutional neural networks for common defect classes, along with a few-shot learning approach for rare defects, enabled the system to handle diverse defect scenarios with improved generalization.

The adoption of a client–server architecture ensured that computationally intensive processing could be performed efficiently without overloading edge devices. Experimental evaluation confirmed that the system operates with acceptable latency and produces interpretable results through a visualization dashboard. Overall, the project achieves its intended objectives and establishes a practical foundation for automated, vision-based railway track inspection.

6.2 Environmental, Economic and Societal Benefits

Environmental Benefits

The proposed system contributes to more sustainable railway infrastructure management by reducing dependence on frequent manual inspections and heavy inspection vehicles. Early detection of surface defects helps prevent severe rail damage, thereby minimizing material wastage and unnecessary track replacement. This supports more efficient use of resources and reduces the environmental footprint associated with large-scale maintenance activities.

Economic Benefits

From an economic perspective, the system offers a cost-effective alternative to traditional inspection methods. The use of low-cost hardware components and open-source software frameworks reduces implementation costs. Early identification of defects enables timely maintenance, which can significantly lower long-term repair expenses and reduce service disruptions. The modular design of the system also allows for incremental scaling without major infrastructure investment.

Societal Benefits

The system enhances passenger safety by supporting timely identification of potentially hazardous railway track defects. Improved inspection reliability contributes to safer and more dependable railway operations, increasing public

confidence in rail transportation. In addition, the project promotes the adoption of modern AI-driven technologies in infrastructure maintenance, encouraging skill development and innovation in the engineering workforce.

6.3 Reflections

This project provided valuable insights into the challenges of developing real-world computer vision systems beyond controlled laboratory environments. One key learning was the importance of preprocessing and data quality in achieving reliable model performance. The project highlighted that addressing background interference and dataset imbalance is often as critical as selecting advanced learning models.

The development process also emphasized the need for realistic system design choices, particularly in terms of deployment architecture and computational constraints. Implementing a client–server model enabled practical experimentation while maintaining scalability. Overall, the project strengthened understanding of applied machine learning, system integration, and the importance of aligning technical solutions with real-world constraints.

6.4 Future Work

While the proposed system meets its current objectives, several opportunities exist for future enhancement:

1. **Dataset Expansion:**
   Collecting larger and more diverse datasets under varying environmental conditions would improve model robustness and generalization.
2. **Advanced Model Architectures:**
   Future work can explore newer deep learning architectures such as YOLOv8, vision transformers, or hybrid CNN–Transformer models to further improve segmentation and classification accuracy.
3. **Edge Inference Optimization:**
   Model compression and optimization techniques could be investigated to enable partial or full inference on edge devices.

4. **Extended Defect Categories:**
   The system can be extended to detect additional surface anomalies such as rail wear patterns or joint irregularities.
5. **Large-Scale Field Validation:**
   Long-term testing on operational railway sections would provide deeper insights into system performance and reliability.
6. **Integration with Maintenance Systems:**
   Linking inspection outputs with digital maintenance planning tools could enhance decision-making and maintenance prioritization.

7 PROJECT METRICS

This chapter presents qualitative and quantitative measures used to evaluate the learning outcomes, technical depth, teamwork, and overall effectiveness of the project. The metrics discussed here reflect both the technical execution of the railway track defect detection system and the academic learning gained during the project lifecycle.

7.1 Challenges Faced

During the execution of the project, several challenges were encountered across technical, operational, and learning dimensions:

- **Background Interference in Images:**
  Railway track images contained ballast, sleepers, and surrounding objects that visually resembled rail surfaces, making defect detection difficult. This challenge required the adoption of rail-aware preprocessing using segmentation.
- **Imbalanced Dataset:**
  Certain defect types were significantly underrepresented, which affected the performance of conventional supervised models. Addressing this imbalance required additional study and implementation of few-shot learning techniques.
- **Data Quality Variations:**
  Variations in lighting conditions, shadows, and image clarity impacted model consistency, highlighting the importance of careful data cleaning and preprocessing.

- **System Integration Complexity:**
  Integrating image acquisition, GPS data, server-side processing, and dashboard visualization into a single pipeline required careful coordination and iterative testing.
- **Time and Resource Constraints:**
  Limited time and computational resources necessitated practical design decisions, such as adopting a client–server architecture instead of full edge inference.

Overcoming these challenges contributed significantly to both the technical robustness of the system and the learning outcomes of the project.

7.2 Relevant Subjects

The project integrates knowledge and skills from multiple subjects studied during the academic program, including:

- **Machine Learning and Deep Learning:**
  Applied concepts of convolutional neural networks and metric-based learning for defect detection.
- **Computer Vision:**
  Used image preprocessing, segmentation, and feature extraction techniques.
- **Artificial Intelligence:**
  Implemented intelligent decision-making through trained models and data-driven analysis.
- **Embedded Systems:**
  Utilized Raspberry Pi for image acquisition and system interfacing.
- **Software Engineering:**
  Followed structured requirement analysis, system design, testing, and documentation practices.
- **Database and Web Technologies:**
  Supported result storage and visualization through server-side applications and dashboards.

7.3 Interdisciplinary Knowledge Sharing

The project encouraged interdisciplinary learning by combining concepts from different engineering domains:

- Integration of **AI and computer vision** with **embedded hardware systems**
- Application of **software engineering principles** to real-world infrastructure problems
- Use of **data analytics and visualization** for inspection support and decision-making

This interdisciplinary approach enhanced problem-solving skills and provided exposure to real-world system design challenges where multiple technologies interact.

7.4 Peer Assessment Matrix

The contribution of each team member was evaluated through peer assessment based on defined criteria such as technical contribution, participation, and responsibility.

Criteria	Divrose	Himanshu	Kezia	Lalit	Pranav
Technical Contribution	Excellent	Very Good	Good	Very Good	Excellent
Participation & Commitment	Excellent	Good	Good	Very Good	Very Good
Problem Solving	Very Good	Good	Very Good	Good	Good
Documentation & Reporting	Good	Very Good	Good	Very Good	Good
Overall Contribution	Excellent	Very Good	Good	Very Good	Excellent

Table 3: Peer Assessment

7.5 Role Playing and Work Schedule

Roles were clearly defined among team members to ensure efficient execution of the project:

- **Project Planning and Coordination:** Requirement analysis, scheduling, and progress tracking

- **Model Development:** Dataset preparation, segmentation, and defect detection model implementation
- **System Integration:** Client–server communication and GPS integration
- **Testing and Documentation:** Experimental evaluation, report preparation, and result analysis

The work schedule followed a phased approach, including requirement analysis, design, implementation, testing, and documentation, ensuring timely completion of project milestones.

7.6 Student Outcomes Description and Performance Indicators

The project outcomes were mapped to standard engineering student outcomes as follows:

Student Outcome	Description	Achievement Level
SO-A	Applied engineering knowledge to solve real-world problems	High
SO-B	Problem analysis and requirement identification	High
SO-C	System design and development	High
SO-D	Teamwork and collaboration	Moderate to High
SO-E	Use of modern engineering tools	High
SO-F	Professional and ethical responsibility	Moderate
SO-G	Effective communication	High
SO-H	Impact of engineering solutions on society	Moderate
SO-I	Lifelong learning	High
SO-J	Awareness of contemporary issues	Moderate
SO-K	Use of engineering techniques and skills	High

Table 4: Students Outcomes

7.7 Brief Analytical Assessment

The project metrics indicate that the proposed system successfully meets both technical and academic expectations. From a technical standpoint, the system demonstrates effective handling of real-world challenges such as background interference and dataset imbalance. From an academic perspective, the project facilitated strong learning outcomes in machine learning, system integration, and applied problem-solving.

The structured execution, interdisciplinary approach, and clear documentation reflect a mature understanding of engineering practices. Overall, the project stands as a meaningful application of theoretical knowledge to a practical infrastructure problem, fulfilling the intended objectives and learning outcomes.

APPENDIX A: REFERENCES

1. S. Sharma, A. Kumar, and R. Singh, "Vision-based railway track inspection using deep learning techniques," *International Journal of Rail Transportation*, vol. 9, no. 3, pp. 215–231, 2021.
2. M. L. Zhang and Z. H. Zhou, "A review on multi-label learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.
3. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.
4. A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
5. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
6. J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4077–4087.
7. R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
8. D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
9. OpenCV Documentation, "Open Source Computer Vision Library," [Online]. Available: https://opencv.org
10. Roboflow Documentation, "Dataset annotation and preprocessing tools," [Online]. Available: https://roboflow.com
11. Raspberry Pi Foundation, "Raspberry Pi 4 Model B Documentation," [Online]. Available: https://www.raspberrypi.com
12. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer, 2009.
13. Indian Railways Institute of Civil Engineering (IRICEN), "Permanent Way Manual," Ministry of Railways, Government of India.