## AI Trolley for Railway Track Crack Detection using Neural Network

Submitted By

### S. MOHAMED IBRAHIM Register No: 910622301024

**A PROJECT REPORT**

Submitted to the

FACULTY OF INFORMATION & COMMUNICATION ENGINEERING

In partial fulfillment of the requirements for the award of the degree of

### MASTER OF COMPUTER APPLICATIONS (MCA)

**IN**

### K.L.N. COLLEGE OF ENGINEERING

(An Autonomous Institution)

### POTTAPALAYAM, SIVAGANGAI – 630 612

**ANNA UNIVERSITY, CHENNAI-600 025 JULY 2024**

### K.L.N. COLLEGE OF ENGINEERING (An Autonomous Institution)

**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS(MCA)**



### BONAFIDE CERTIFICATE

Certified that this project report titled “**AI TROLLEY FOR RAILWAY TRACK CRACK DETECTION USING NEURAL NETWORK”** is the bonafide work of Mr. **S. MOHAMED IBRAHIM (Reg.No:910622301024)** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

### Dr. M.R ILANGO,MCA., M Phil., Ph.D.

**DIRECTOR/MCA,**

### K.L.N College of Engineering.

**R. SOMAS KANDAN, MCA., M Phil., Ph.D.**

**INTERNAL GUIDE,**

### K.L.N College of Engineering.

Certified that the candidate was examined by us in the viva-voce examination held at K.L.N. College of Engineering, Sivagangai on …………………………………

### INTERNAL EXAMINER EXTERNAL EXAMINER



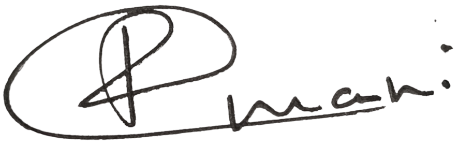
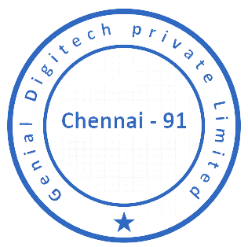
**Date**: 14/06/2024

**Place**: Chennai

**Project Completion Certificate**

This is to certify that **Mr. MOHAMED IBRAHIM S (Reg.No:910622301024)** Student of **MCA.,** (Master of Computer Applications) in **KLN College of Engineering, Sivagangai**, has successfully completed the **AI TROLLEY FOR RAILWAY TRACK CRACK DETECTION USING NEURAL**

**NETWORK"** in **Python** Platform from **January 2024 to June 2024** in our project title company.



During the period, he had been exposed to different processes and was found to be Punctual,

Hard Working, and Inquisitive. We wish him every success in life and career.

**Sincerely,**

**Human Resource Genial Digitech Pvt. Ltd**.

**Genial Digitech Pvt. Ltd.**

Plot no 531, MERRYDALE, F1, Sivaraman Street, Ram Nagar North Extension, Madipakam, Chennai 600 091, Tamilnadu, India.

Website: [www.genialdigitech.com](http://www.genialdigitech.com/) - Email: [info@genialdigitech.com](mailto:info@genialdigitech.com) - Phone : 044 4208 5859

**ACKNOWLEDGEMENT**

I would like to express my sincere thanks to our honorable Management and our respected **PRINCIPAL, Dr. A.V. RAM PRASAD M.E., Ph.D.,** for his support and wonderful opportunity for doing the project in our college.

I would like to express my sincere thanks to our Respected **DIRECTOR Dr. M.R. ILANGO MCA., M.Phil., Ph.D.,** Director of MCA Department, for his permission, guidance and support in preceding the project.

I would like to express my sincere thanks to our Respected **INTERNAL GUIDE**

**R. SOMAS KANDAN MCA., M.Phil., Ph.D.** for his guidance and support inpreceding the project.

I would like to express my sincere thanks to my Project team **Genial Digitech Pvt. Ltd, Chennai,** for their immense guidance and valuable suggestion in completing my project successfully.

I thank my parents, faculty members of MCA Department for their moral support to complete the project in a successful manner.

A project is never outcome of a single person’s efforts, It is a confluence of a varied thought process harmoniously integrated into a resourceful product. It is, but natural that I feel indebted to several people for having made this project possible.

## ABSTRACT

In India railways transportation service is the cheap and the majority

convenient mode of passenger transport and also for long distance and suburban traffic. A recent study revealed that over 25% of the track length is in need of replacement due to the development of cracks on it. The main cause of the accidents happened in railways are railway track crossing and unrevealed crack in railway tracks. Manual detection of tracks is cumbersome and not fully effective owing to much time consumption and requirement of skilled technicians. Therefore, there is a high demand for advanced inspection methods to monitor the railway track and its components continuously. The presence of such advanced inspection models would help the railway industry avoid obstacles such as high operation and maintenance costs, dangerous accidents, and uncomfortable passenger's experience. This project is aimed towards addressing the issue by developing an automatic railway track crack detection system with the proliferation of Artificial Intelligence. The proposed method enhances the track image using adaptive histogram equalization technique and further features as Grey Level Co-occurrence Matrix (GLCM) and Local Binary Pattern (LBP) feature are extracted from the enhanced rail track image. These extracted features are trained and classified using Crack Feature Fusion Neural Network (CFNet) classifier which classifies the rail track image into either cracked or non-cracked image. The novelty of this project is to use soft computing approach for the detection of cracks in rail tracks and also estimate the crack length and give report of railway track health. This methodology is trained by several crack images which are obtained from different environment.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **C. NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
| **1** | **INTRODUCTION** |  |
|  | 1.1. ORGANIZATION PROFILE | **2** |
| 1.2. PROJECT DESCRIPTION | **3** |
| **2** | **SYSTEM ANALYSIS** |  |
|  | 2.1. EXISTING SYSTEM | **10** |
| 2.2. PROPOSED SYSTEM | **11** |
| 2.3. ABOUT THE PROJECT | **12** |
| 2.4. FEASIBILITY STUDY | **13** |
| **3** | **SYSTEM DESIGN** |  |
|  | 3.1. SYSTEM DESIGN GOALS | **14** |
| 3.2. DATA FLOW DIAGRAM | **16** |
| 3.3. ER DIAGRAM | **19** |
| 3.4 DATABASE DESIGN & TABLE DESIGN | **19** |
| **4** | **SYSTEM REQUIREMENTS AND ANALYSIS** |  |
|  | 4.1. HARDWARE AND SOFTWARE SPECIFICATION | **21** |
| 4.2. SOFTWARE EXPLANATION | **22** |
| **5** | **MODULES DESCRIPTION** |  |
|  | 5.1 MODULES | **27** |
| 5.2. MODULE DESCRIPTION | **28** |
| **6** | **SYSTEM TESTING AND IMPLEMENTATION** |  |
|  | 6.1. INTRODUCTION | **35** |
| 6.2. STRATEGIC APPROACH TO SOFTWARE TESTING | **35** |
| 6.3. TEST CASE | **37** |
| 6.4. TEST REPORT | **38** |
| 6.5. IMPLEMENTATION | **40** |
| **7** | **CONCLUSION** | **42** |
| **8** | **FUTURE ENHANCEMENT** | **43** |
| **9** | **APPENDIX** |  |
|  | 9.1 SOURCE CODE | **44** |
| 9.2 SCREENSHOTS | **50** |
| **10** | **BIBLIOGRAPHY** | **59** |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TABLE NAME** | **PAGE NO** |
| 3.4.1 | Admin | **19** |
| 3.4.2 | Rm\_Report | **19** |
| 3.4.3 | Rm\_Upload | **20** |
| 3.4.4 | Rm\_User | **20** |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIG.NO** | **FIG.NAME** | **PAGENO** |
| 1.1 | Deep Learning | **5** |
| 1.2 | Neural Network  Structure | **6** |
| 1.3 | Process Of CNN | **7** |
| 3.2.4 | Overall Structure | **18** |
| 9.2.1 | Home Page | **50** |
| 9.2.2 | Admin | **50** |
| 9.3.3 | Training Phase | **51** |
| 9.3.4 | Training Result | **54** |
| 9.3.5 | Railway-in-charge | **55** |
| 9.3.6 | In-charge Testing | **57** |
| 9.3.7 | Defective or Not | **59** |

**INTRODUCTION**

### ORGANIZATION PROFILE

Genial Digitech based in Chennai, India has been started by a team of well-groomed Professionals with decade of experience in business Process and IT domain. With its qualified professional’s team, the company is successful in the areas of Software Development, Support, Resourcing and Infrastructure Setup. We fulfil our client's requirements by providing efficient, cost effective, quality solutions on time. Our consultants help business in streamlining the processes for efficient operation there by support in increasing productivity.



**Mission**

Enhance business growth of our customers with creating design, development and timely delivery of quality solutions that create value to them around the globe. Providing high quality software development services, professional consulting and development outsourcing that would improve our customer operations.

* + - Making access to information easier and securer (Enterprise Business)
    - Improving communication and data exchange (Business to Business)
    - Providing our customers with a Value for Money.

**Vision**

To be the most preferred and reliable business partner of all time in technology sector for the business community and society without compromising business ethics and employee welfare. **Contact us**

**GENIAL DIGITECH**

Plot no:531, Merrydale F1, Sivaraman St, Ram Nagar, Madipakkam, Chennai, Tamil Nadu. 600091 [www.genialdigitech.com,info@genialdigitech.com](mailto:www.genialdigitech.com%2Cinfo@genialdigitech.com)

0431-3562302.

## PROJECT DESCRIPTION

1.2.1. **OVERVIEW**

Indian Railways (IR) is the state-owned railway company of [India.](https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/i/India.htm) Indian Railways had, until very recently, a monopoly on the country's rail transport. It is one of the largest and busiest rail networks in the world, transporting just over six billion passengers and almost 750 million tonnes of freight annually. IR is the world's largest commercial or utility employer, with more than 1.6 million employees. The railways traverse through the length and width of the country; the routes cover a total length of 63,940 km (39,230 miles). As of 2005 IR owns a total of 216,717 wagons, 39,936 coaches and 7,339 locomotives and runs a total of 14,244 trains daily, including about 8,002 passenger trains. Railways were first introduced to India in 1853. By 1947, the year of [India's independence,](https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/i/Indian_independence_movement.htm) there were forty-two rail systems. In 1951 the systems were nationalised as one unit, becoming one of the largest networks in the world. Indian Railways operates both long distance and suburban rail systems.

## RAILWAY TRACKS

A railway track or railway line is a set of two [parallel](https://simple.wiktionary.org/wiki/parallel) rows of long pieces of [steel](https://simple.wikipedia.org/wiki/Steel). They are used by [trains](https://simple.wikipedia.org/wiki/Railway) to [transport](https://simple.wikipedia.org/wiki/Transport) people and things from one place to another. (In [America,](https://simple.wikipedia.org/wiki/United_States) people say railroad as well as railway). Often, there is more than one set of tracks on the railway line. For example, trains go [east](https://simple.wikipedia.org/wiki/East) on one track and [west](https://simple.wikipedia.org/wiki/West) on the other one. The rails are supported by cross pieces set at regular intervals (called sleepers or ties), which spread the high pressure load imposed by the train wheels into the ground. They also maintain the rails at a fixed distance apart (called the gauge). Ties are usually made from either wood or concrete. These often rest on ballast, which is a name for very small pieces of broken up rock that are packed together and keep the railway tracks in place.

Tracks are often made better by [ballast tampers.](https://simple.wikipedia.org/wiki/Ballast_Tamper) The upper [surfaces](https://simple.wikipedia.org/wiki/Surface) of the rails are inclined slightly towards each other, typically on a slope of 1/20, and the rims of the train wheels are angled in the same way ("coning"). This helps guide the vehicles of the train along the track. Each wheel also has a flange, which sticks out from one edge all the way around. This makes sure the train does not "derail" (come off the track) and helps guide the train on sharp curves **Railroad Track Components**

A railroad track is mainly composed of rails, railroad ties (sleepers), fasteners, railway switch, ballast, subgrade. The components of railway track play different roles in providing support

for trains. The track structure is built for rolling stock to roll upon safely and smoothly. Both passenger lines and freight lines are beneficial from this convenient fast running track transportation.

### Rails

The main part of a railroad track is rails. Most modern rails have a profile that shaped like an I-beam vertically symmetric but horizontal asymmetric. To meet the needs of turnout, large bridges, and seamless lines and other structures, non-axisymmetric rails are produced as well. [Railroad rails](https://railroadrails.com/railroad-rail-for-sale/) are made from steels which can be subjected to very high stresses by the trains. To make the rail better withstand the forces from all sides, to ensure the necessary strength conditions, the rails should be fabricated with enough height. Its head and foot should have enough area and height, web and foot should not be too thin. Rails are classified according to rail size and rail weight. In general, the heavier rail supports greater axle loads and higher train speeds.

### Railway sleeper

A railway sleeper is also called a railroad tie or crosstie. It is an easily overlooked component of railway tracks. The sleepers not only support the rail but also maintain the position of the rail. Besides, it helps to transmit the huge pressure by rail to the trackbed. It is required to have a certain degree of flexibility and elasticity. It can’t be too hard or too soft. When the train passes by, it can be properly deformed to cushion the pressure. But, it has to be restored as much as possible. In the early days, railroad ties are made of wood while prestressed concrete is more common to see now. Sometimes plastic composite ties are also applied.

### Fasteners

Fasteners are used for fixing rails to railway sleepers. It is also a very important component of railway track. Most commonly seen fasteners include spikes, screws, rail anchors, tie plates, chairs, etc. Various types of fasteners have been used over the years. Other track materials like rail clip, rail clamp, rail pad, rail joints also belong to railway fastening systems.

### Railroad switch

Railway switch is also called turnout. As its name indicates, it is a component for rolling stocks to turn from one track to another. It is usually laid in large numbers at stop stations and marshalling stations. With turnouts, you can give full play to the passing capacity of the line. Even if it is a single-track railway, lay the turnouts and build a section of the fork line that is longer than the length of the train to split the train. Turnouts play an important role in railway lines.

### Ballast

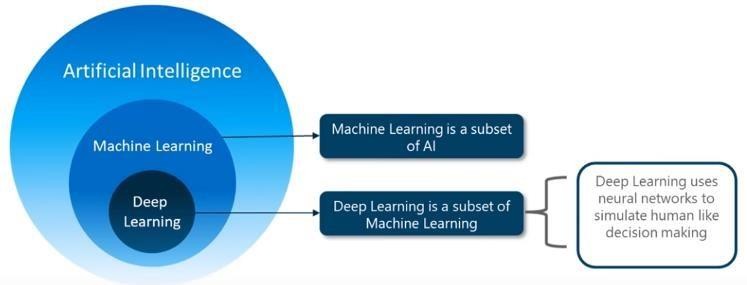
Railway ballast or track ballast refers to crushed stones placed under the railway track. It forms the trackbed for sleepers to lay on. Although some tracks are ballastless, the ballasted track remains a dominant infrastructure of the most railroad tracks.

### Track Subgrade

Track subgrade is the formation level composed of native materials underneath a railroad track. The subgrade is the fundamental structure for railway tracks. Under the ballast is the part called subgrade. The railway subgrade is a structure that bears and transmits the gravity of the track and the dynamic action of the train. It is the foundation of the track and an important building to ensure the operation of the train.

## DEEP LEARNING

Artificial intelligence is a set of algorithms and intelligence to try to mimic human intelligence. Machine learning is one of them, and deep learning is one of those machine learning techniques. Deep Learning is a subset of Machine Learning that uses mathematical functions to map the input to the output. These functions can extract non-redundant information or patterns from the data, which enables them to form a relationship between the input and the output. This is known as learning, and the process of learning is called training.

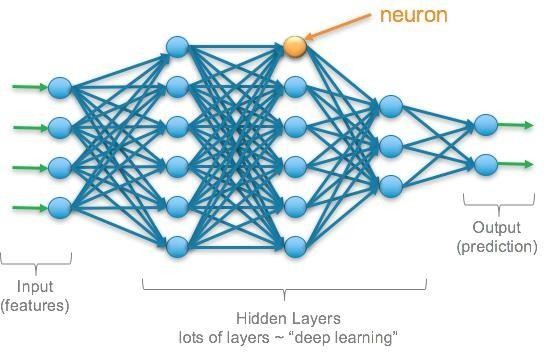


### 1.2. Deep Learning

Deep learning can be used on a variety of input data types including audio, video, text, images, radio waves and machine signals to create applications such as natural language processing, audio recognition, computer vision and target recognition.

### Artificial neural networks

Artificial neural networks are formed by layers of connected nodes. Deep learning models use neural networks that have a large number of layers.

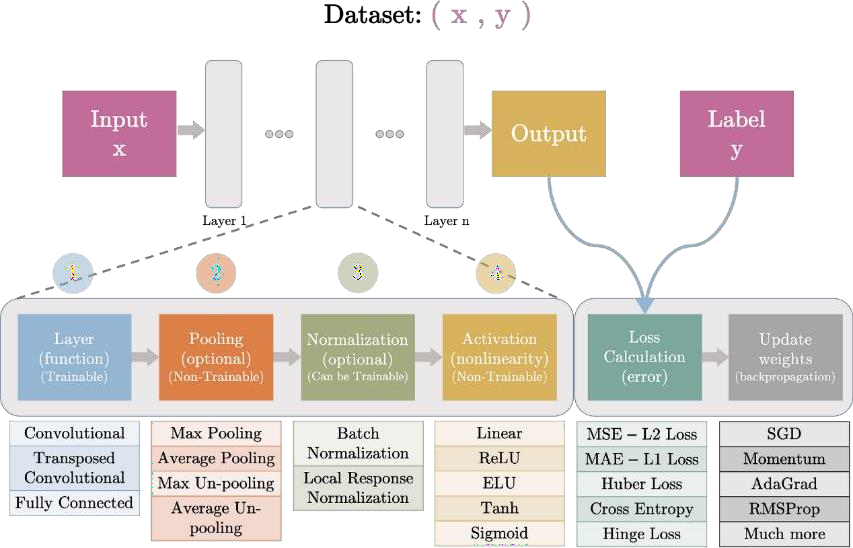


### Neural network Structure

The above diagram shows a neural network — the nodes are referred to as “neurons”: this is because neural networks were loosely based on neurons in the brain. Neurons receive input, whether it is the initial input (the features of what you are training to make a prediction with) or the output from other neurons. They then make a decision of what to pass to the next layer of neurons. The layers between the input and output are referred to as “hidden layers”.

### CNN

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects, and traffic signs apart from powering vision in robots and self-driving cars. In Neural Networks, Convolutional Neural Network (ConvNets or CNNs) is one of the main categories to do image recognition, images classifications. object Detections, Recognition faces, etc, are some of the areas where CNNs are widely used. The best thing is there is no need for feature extraction. The system learns to do feature extraction and the core concept of CNN is, it uses convolution of images and filters to generate invariant features which are passed on to the next layer. A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation-invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this blog, we will be introducing the vital layers which constitute our everyday CNN.



### Process Of CNN

**Different Types of layers in CNN. Input Layer**

Let’s take an example by running a covnets on of image of dimensions 32 x 32 x 3. Input Layer holds the raw input of image with width 32, height 32, and depth 3.

### Convolution Layer

It computes the output volume by computing dot products between all filters and image patches. Suppose we use a total of 12 filters for this layer we’ll get output volume of dimension 32 x 32 x 12.

### Activation Function Layer

This layer will apply the element-wise activation function to the output of the convolution layer. Some activation functions are RELU: max(0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc. So the volume remains unchanged. Hence output volume will have dimensions 32 x 32 x 12.

### Pool Layer

This layer is periodically inserted within the covnets, and its main function is to reduce the size of volume which makes the computation fast reduces memory, and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

Pooling can be of different types:

* Max Pooling
* Average Pooling
* Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

### Fully-Connected Layer

This layer is a regular neural network layer that takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

## Feature Fusion Neural Network

Feature fusion refers to the process of combining different types of features or information from multiple sources to enhance the performance or capabilities of a model or system. It involves integrating and merging the extracted features or information to create a more comprehensive representation. Feature fusion can be applied in various domains, such as gene regulatory network reconstruction, speech synthesis detection, image captioning, remote sensing image processing, and 3D object detection for autonomous vehicles. In each of these contexts, feature fusion techniques are used to combine and leverage different types of features or data sources to improve the accuracy, robustness, or effectiveness of the respective models or systems.

## AIM AND OBJECTIVE

The primary aim of this project is to develop an Automatic Railway Track Crack Detection System using Artificial Intelligence (AI) to enhance the safety and reliability of India's railway network. By leveraging advanced image processing and deep learning techniques, the system aims to provide accurate and timely detection of cracks in railway tracks, contributing to the overall improvement of rail infrastructure

* + - * To develop an Automatic Railway Track Crack Detection System using AI.
      * To enhance railway track images for effective crack detection through advanced image processing techniques.
      * To provide timely alerts and comprehensive reports for proactive maintenance.
      * To evaluate the system's performance through rigorous testing and validation.
      * To demonstrate the feasibility and effectiveness of AI-based solutions in enhancing railway track safety and efficiency.

## SCOPE OF THE PROJECT

The scope of the project encompasses the following aspects:

* + - * **Crack Detection System Development:** Developing an Automatic Railway Track Crack Detection System named Crack Finder. This system will utilize advanced image processing techniques and machine learning algorithms, such as the CrackNet model, to detect cracks in railway tracks accurately and efficiently.
      * **CrackNet Model Implementation:** Implementing the CrackNet model, which includes various stages such as dataset collection, preprocessing, feature extraction, classification, model training, and deployment. The model will be trained to accurately classify track segments as cracked or non-cracked.
      * **User Interface Development:** Designing and developing a user-friendly interface for the Crack Finder system. This interface will allow users, including administrators and railway in charge personnel, to interact with the system, upload images, visualize crack detection results, and receive alerts.
      * **Alert Generation Mechanism:** Implementing an alert generation mechanism that notifies the railway department about detected cracks. Alerts will include detailed information about the location and specifics of the detected cracks, enabling prompt action and maintenance.
  1. **EXISTING SYSTEM**

# SYSTEM ANALYSIS

* + - **Visual Inspection:** Personnel visually inspect railway tracks for visible cracks, fractures, or signs of wear and tear. This method relies on the human eye to identify surface irregularities and anomalies.
    - **Periodic Track Walks:** Inspection teams walk along the railway tracks, visually examining the rails, sleepers, and other components for signs of damage or deterioration. This method allows inspectors to detect cracks or defects that may not be visible from a distance.
    - **Ultrasonic Testing:** Ultrasonic testing involves using handheld devices to detect cracks and defects beneath the surface of the rail. Ultrasonic waves are transmitted into the rail, and reflections from internal defects are analyzed to identify potential issues.
    - **Manual Measurement:** Inspectors may use manual measurement tools such as gauges or calipers to measure the dimensions of cracks or defects found during inspections. This information helps assess the severity of the damage and prioritize maintenance activities.
    - **Periodic Maintenance:** Based on inspection findings, maintenance crews are dispatched to repair or replace damaged track sections. This maintenance is often scheduled at regular intervals or in response to specific safety concerns.
    - **Hammer Testing:** Technicians may use a hammer or similar tool to tap along the length of the rail, listening for variations in sound that could indicate hidden defects.

### Disadvantages

* + - Labor-Intensive: Manual inspection is labor-intensive and requires significant human effort.
    - Time-Consuming: Traditional methods are time-consuming, leading to delays in identifying and addressing track issues.
    - Subject to Human Error: Relies on human skills, making it susceptible to errors and variations in inspection quality.
    - Reactive Approach: Reactive in nature, addressing issues after visual identification rather than preventing them proactively.

### PROPOSED SYSTEM

The proposed system, titled aims to revolutionize the way railway track inspections are conducted by introducing an innovative solution powered by artificial intelligence (AI). The system leverages advanced image processing techniques and deep learning algorithms to automate the detection of cracks and other defects in railway tracks, thereby enhancing safety and reducing maintenance costs.

* + - **Crack Finder Web App**: A user-friendly web application that serves as the interface for uploading images of railway tracks and viewing detection results. The web app provides functionalities for user authentication, image submission, result visualization, and system management.
    - **CrackNet Model**: A deep learning model specifically designed for crack detection in railway tracks. The CrackNet model utilizes convolutional neural networks (CNNs) to extract features from input images and classify track segments as either cracked or non- cracked. The model is trained on a large dataset of annotated track images to ensure high accuracy and reliability.
    - **Image Processing Pipeline**: A series of image processing techniques employed to preprocess the uploaded track images before feeding them into the CrackNet model. Preprocessing steps may include resizing, noise reduction, contrast enhancement, and region segmentation to improve the quality and clarity of the images.
    - **Alert Generation System**: An automated alert generation mechanism that notifies railway authorities and maintenance teams upon detecting cracks in track segments. Alerts include detailed information about the location, severity, and size of detected cracks, enabling swift action to rectify the issues.
    - **User Management Module**: A module responsible for managing user accounts, roles, and permissions within the system. Administrators have the authority to create and modify user accounts, assign roles, and monitor system activity to ensure security and accountability..

### Advantages

* + - * Enhanced safety through automated crack detection, reducing the risk of accidents and derailments.
      * Cost savings by minimizing manual inspection efforts and preventing expensive repairs. Proactive maintenance by detecting defects in real-time, minimizing downtime.

### ABOUT THE PROJECT

The AITrolley Railway Track Crack Detection project is designed to enhance railway safety and maintenance efficiency through the application of advanced artificial intelligence techniques.

### Objectives

* + - **Improve Railway Safety**: By providing timely and accurate crack detection, the system helps prevent accidents caused by track failures.
    - **Enhance Inspection Efficiency**: Automating the detection process reduces the time and labor required for track inspections.
    - **Reduce Maintenance Costs**: Early detection of cracks allows for preventive maintenance, reducing the overall cost and complexity of track repairs.
    - **Provide Detailed Reporting**: The system generates comprehensive reports on track health, aiding in informed decision-making for maintenance scheduling.

### Implementation

The implementation process involves several steps:

* + - **Software Development**: Building the web application interface and backend functionality using technologies like Python, Flask, MySQL, and Bootstrap.
    - **Model Training**: Using datasets of railway track images to train the CrackNet model with TensorFlow and Scikit-learn.
    - **Data Acquisition and Preparation**: Collecting, labeling, and preprocessing images to create a robust training dataset.
    - **Integration and Testing**: Ensuring seamless integration of the web app with the machine learning model and conducting thorough testing to validate functionality.
    - **Deployment**: Deploying the system on servers or cloud platforms and ensuring its scalability and reliability.
    - **User Training and Support**: Providing training for end-users and ongoing technical support to ensure effective use of the system.

### Benefits

* + - **Increased Safety**: Automated crack detection reduces the risk of accidents caused by track failures.
    - **Operational Efficiency**: Streamlined inspection processes save time and resources.
    - **Cost Savings**: Proactive maintenance reduces long-term repair costs.
    - **User-Friendly**: The web app offers an intuitive interface for easy use by railway personnel.
  1. **FEASIBILITY STUDY**

The feasibility study of the project encompasses several aspects, including technical, economic, and operational feasibility:

### Technical Feasibility

* + - * **Availability of Technology:** The project relies on established technologies such as machine learning algorithms and web development frameworks, ensuring technical feasibility.
      * **Data Accessibility:** Access to relevant datasets for model training and testing is essential. If suitable datasets are available or can be acquired, technical feasibility is enhanced.
      * **Computational Resources:** The project requires adequate computational resources for model training and deployment. Assessing the availability of such resources ensures technical feasibility.

### Economic Feasibility

* + - * **Cost Analysis:** Conducting a thorough cost analysis, including expenses related to hardware, software, personnel, and maintenance, determines the economic feasibility of the project.
      * **Return on Investment (ROI):** Evaluating the potential benefits of the project against its costs helps in assessing its economic viability. If the expected ROI is positive, the project is economically feasible.

### Operational Feasibility

* + - * **User Acceptance:** Assessing the willingness of stakeholders, including railway authorities and maintenance personnel, to adopt the proposed system is crucial for operational feasibility.
      * **Integration with Existing Systems:** Ensuring compatibility and seamless integration with existing railway management systems enhances operational feasibility.
      * **Scalability:** The system should be scalable to accommodate future expansion and changes in requirements, ensuring long-term operational feasibility.

# SYSTEM DESIGN

### SYSTEM DESIGN GOALS

The AITrolley Railway Track Crack Detection system is designed to address the critical need for efficient and accurate detection of cracks in railway tracks. This innovative solution leverages advanced machine learning and image processing techniques to ensure railway safety and maintenance efficiency. Here’s a detailed overview of what the system does:

### Accurate Crack Detection

* + - * **Advanced Algorithms**: Utilizes state-of-the-art neural networks, specifically the CrackNet model, to accurately identify and classify cracks in railway track images.
      * **Feature Extraction**: Employs sophisticated methods like Grey Level Co- occurrence Matrix (GLCM) and Local Binary Pattern (LBP) to extract critical features that enhance detection accuracy.

### Real-Time Processing and Alerts

* + - * **Rapid Image Processing**: Processes images in real-time or near-real-time, ensuring that crack detection is swift and efficient.
      * **Immediate Alerts**: Generates instant alerts to railway authorities upon detecting cracks, providing details about the crack’s location and severity to facilitate timely maintenance actions.

### User-Friendly Interface

* + - * **Simplified Image Upload**: Allows users to easily upload images of railway tracks in various formats through a user-friendly web application.
      * **Clear Visualization**: Offers interactive visualizations of crack detection results, making it easy for users to understand and act upon the information.

### Robust Data Management

* + - * **Data Integrity**: Ensures the security and integrity of all datasets used for training the model and storing detection results.
      * **Scalable Storage**: Utilizes scalable database solutions to manage large volumes of image data, ensuring efficient and reliable storage.

### Scalability and Performance

* + - * **Scalable Architecture**: Designed to handle a high volume of images and users simultaneously, without compromising performance.
      * **Load Balancing**: Implements efficient load balancing techniques to manage high user traffic and processing demands effectively.

### Security and Privacy

* + - * **Secure Authentication**: Protects the system through robust user authentication and access control mechanisms, ensuring only authorized users can access sensitive functionalities.
      * **Data Encryption**: Encrypts data both in transit and at rest, safeguarding against unauthorized access and ensuring user privacy.

### Maintainability and Extensibility

* + - * **Modular Design**: Follows a modular design approach, making the system easy to maintain and update.
      * **Comprehensive Documentation**: Provides detailed documentation for all system components and functionalities, facilitating ongoing management and support.

### Comprehensive Reporting and Analytics

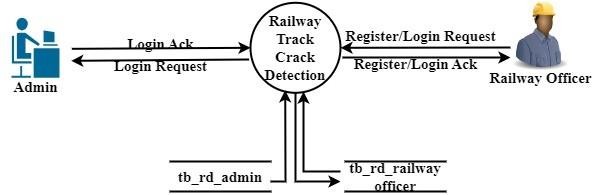
* + - * **Detailed Reports**: Generates comprehensive reports on crack detection results, including details on severity and location, to inform maintenance decisions.
      * **Analytics Tools**: Integrates tools for analyzing detection data, helping to identify patterns and trends that can guide preventive maintenance strategies.

### Integration with Existing Systems

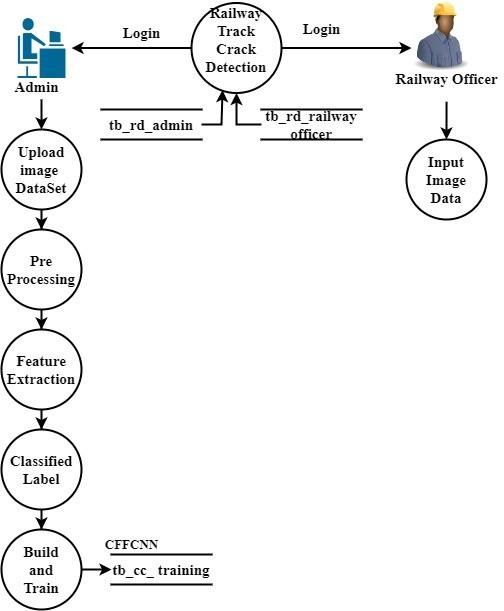
* + - * **API Integration**: Offers APIs to integrate seamlessly with other systems used by railway authorities, enhancing operational efficiency.
      * **Interoperability**: Ensures compatibility with existing tools and platforms in railway operations.

### Cost-Effectiveness

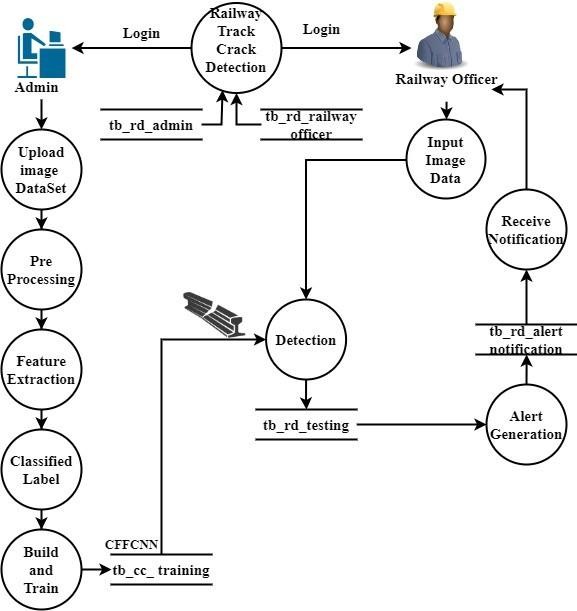
* + - * **Efficient Resource Use**: Optimizes computational and storage resources to minimize operational costs.
      * **OpenSource Technologies**: Leverages opensource technologies wherever possible, reducing licensing expenses and ensuring affordability.
  1. **DATA FLOW DIAGRAM LEVEL 0**



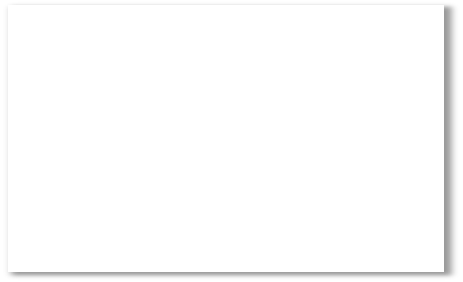
**LEVEL 1**



## LEVEL 2



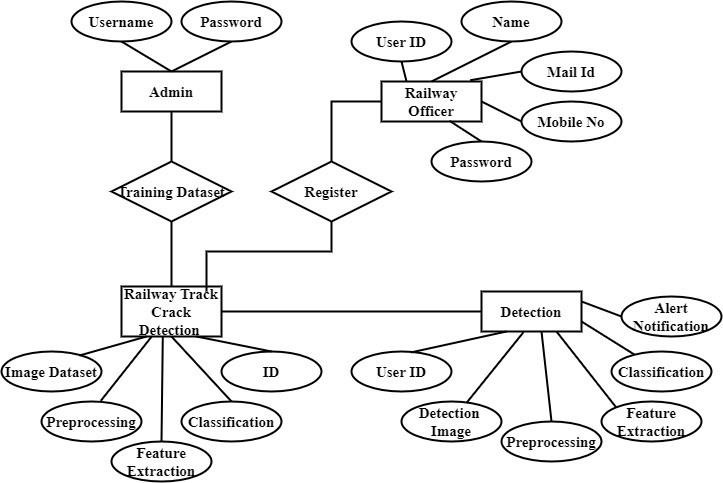
**LEVEL 3**



|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |



* 1. **ER DIAGRAM**



# DATABASE DESIGN & TABLE DESIGN

## Admin

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME** | **TYPE** |
| 1 | username | varchar (20) |
| 2 | password | varchar (20) |
| 3 | mobile | bigint (10) |
| 4 | email | varchar (50) |

* + 1. **Rm\_Report**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME** | **TYPE** |
| 1 | id | int(11) |
| 2 | filename | varchar(20) |
| 3 | lat | varchar(20) |
| 4 | lon | varchar(20) |
| 5 | rdate | varchar(20) |
| 6 | dtime | timestamp |

## Rm\_Upload

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME** | **TYPE** |
| 1 | id | int (11) |
| 2 | uname | varchar(20) |
| 3 | filename | varchar(50) |
| 4 | lon | varchar(20) |
| 5 | lat | varchar(20) |
| 6 | location | varchar(50) |
| 7 | rdate | varchar(20) |
| 8 | reply | varchar(100) |

* + 1. **Rm\_User**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME** | **TYPE** |
| 1 | id | int (11) |
| 2 | name | varchar(20) |
| 3 | city | varchar(20) |
| 4 | mobile | bigint (20) |
| 5 | email | varchar (40) |
| 6 | uname | varchar(20) |
| 7 | pass | varchar(20) |
| 8 | rdate | varchar(20) |

# SYSTEM REQUIREMENTS AND ANALYSIS

## HARDWARE SPECIFICATION

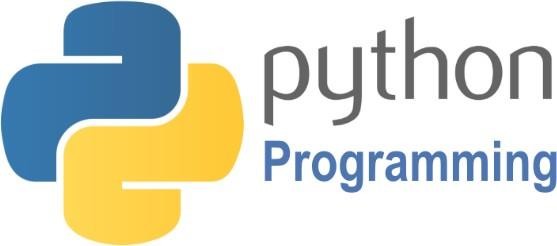
* + - **Processors :** Intel® Core™ i5 processor
    - **Memory :** 16 GB of DRAM
    - **Disk Space :** 320 GB
    - **Operating System :** Windows® 10

## SOFTWARE REQUIREMENTS

* + - * Programming Language: Python
      * Web Framework: Flask
      * Database Management System: MySQL
      * Web Server: WampServer
      * Machine Learning Libraries:
        + TensorFlow: For building and training neural network models.
        + Pandas: For data manipulation and analysis.
        + Scikit-learn: For machine learning algorithms and model evaluation.
      * Data Visualization Libraries:
        + Matplotlib: For creating static, interactive, and animated visualizations.
        + Seaborn: For statistical data visualization and enhanced aesthetics.
      * Image Processing Libraries:
        + Pillow: For image processing tasks such as resizing, cropping, and enhancing.
        + OpenCV (OpenSource Computer Vision Library): For image manipulation, feature extraction, and object detection.
      * Frontend Framework: Bootstrap
      * Other Python Libraries:
        + NumPy: For numerical computing and array manipulation.
        + Flask-MySQL: For interfacing Python with MySQL databases.

## SOFTWARE EXPLANATION

* + 1. **PYTHON 3.7.4**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following:

* + - * Machine Learning
      * GUI Applications (like Kivy, Tkinter, PyQt etc.)
      * Web frameworks like Django (used by YouTube, Instagram, Dropbox)
      * Image processing (like OpenCV, Pillow)

### Tensor Flow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and gives developers the ability to easily build and deploy ML-powered applications.



TensorFlow provides a collection of workflows with intuitive, high-level APIs for both beginners and experts to create machine learning models in numerous languages. Developers have the option to deploy models on a number of platforms such as on servers, in the cloud, on mobile and edge devices, in browsers, and on many other JavaScript platforms. This enables developers to go from model building and training to deployment much more easily.

### Pandas

pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.



Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with Data frames that were established in the R programming language. The panda’s library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Data frames.

### NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

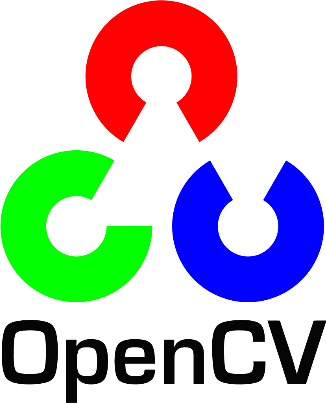


NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

.

### OpenCV

OpenCV is an open-source library for the computer vision. It provides the facility to the machine to recognize the faces or objects.



In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

## MYSQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.



MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

## WAMPSERVER

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your database.



WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn’t require a payment or subscription.

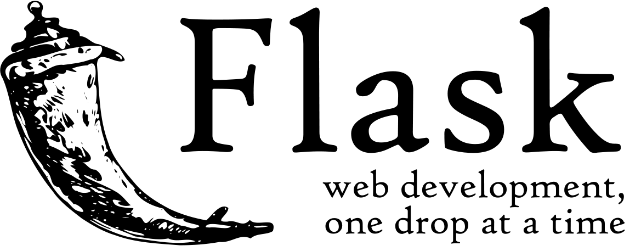
## BOOTSTRAP 4

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

.

**Easy to use**: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap **Responsive features**: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops **Mobile-first approach**: In Bootstrap, mobile-first styles are part of the core framework **Browser compatibility**: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

## FLASK

[Flask](http://flask.pocoo.org/) is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most [Python](https://quintagroup.com/services/python) frameworks, it holds a great promise and has already gained popularity among Python web developers. Let’s take a closer look into Flask, so-called “micro” framework for Python. Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries

## MODULES DESCRIPTION

* 1. **MODULES:**

## Crack Finder

* + 1. **Crack Net Model: Build and Find**
       1. Import Dataset
       2. Preprocessing
       3. Feature Extraction
       4. Classification
       5. Build and Train
       6. Deploy Model

## Crack Identifier

* + - 1. Input Image
      2. Crack Identification using CFFNN

## Alert Generator

* + 1. **System User**
       1. Admin
       2. Railway In charge

## 5.2. MODULE DESCRIPTION

1. **CRACK FINDER WEB APP**

The Crack Finder Web App comprises several essential modules, each playing a crucial role in the crack detection process. Firstly, the User Authentication Module ensures secure access to the app, allowing authenticated users to log in securely. Following this, the Image Upload Module facilitates the submission of images depicting railway tracks for crack detection, supporting various formats and providing a user-friendly interface for upload. Upon image submission, the Preprocessing Module kicks in, enhancing image quality through resizing, grayscale conversion, and noise filtering, preparing them for analysis. Subsequently, the Feature Extraction Module extracts pertinent features from the preprocessed images using advanced algorithms like Grey Level Co-occurrence Matrix (GLCM) and Local Binary Pattern (LBP), vital for accurate crack detection. The Crack Detection Module utilizes machine learning models, such as the CrackNet model, to effectively identify cracked segments based on the extracted features and trained models. Additionally, the Alert Generation Module promptly notifies railway authorities upon crack detection, providing comprehensive details on location, severity, and specifics to facilitate timely maintenance. Simultaneously, the Reporting Module generates detailed reports on crack detection results and maintenance recommendations, offering insights into the overall track health. Administrators can manage user accounts, roles, and permissions using the User Management Module, ensuring smooth operation and access control. The Dashboard Module provides users with a centralized interface to visualize crack detection results, alerts, and reports through interactive data visualization tools. Moreover, the Settings Module allows users to customize app settings according to their preferences, while the Help and Support Module offers access to resources, documentation, and support services within the app. In essence, these modules collaboratively form a robust solution for efficient crack detection in railway tracks, ensuring accuracy, reliability, and usability throughout the process.

## CRACKNET MODEL: BUILD AND TRAIN

### Import Dataset Module:

This submodule facilitates the collection and importation of datasets required for training the CrackNet model. It provides an intuitive interface for the administrator to upload datasets containing images of railway tracks. Users can organize and manage datasets efficiently, ensuring proper labeling of cracked and non-cracked segments for supervised learning.

Advanced functionalities such as data augmentation and data preprocessing options may be included to enhance dataset quality and diversity.

### Preprocessing Module:

The preprocessing submodule performs essential preprocessing tasks on the imported datasets to optimize them for model training. Tasks may include resizing images to a uniform size, converting them to grayscale to reduce computational complexity, and applying noise filtering techniques (e.g., median filtering) to enhance image quality. Binarization techniques may be employed to segment images into regions of interest, isolating railway track segments from background noise.

### Resizing Images:

Image resizing ensures uniformity in input dimensions across the dataset, facilitating efficient processing during training.

This can be represented as:*X*resized=resize(*X*,target\_size)

### Grayscale Conversion:

Conversion of images from color to grayscale simplifies the input data while retaining essential structural information, aiding subsequent analysis.

This operation can be expressed as: *X*grayscale=convert\_to\_grayscale(*X*)

### Noise Filtering Techniques:

Utilization of noise filtering methods, such as median filtering or Gaussian smoothing, diminishes image artifacts and enhances overall image quality, thus refining subsequent feature extraction steps.

This can be denoted as: *X*filtered=apply\_filter(*X*)

### Binarization:

By employing binarization techniques, images are segmented into distinct regions of interest, isolating railway tracks from background clutter and improving focus for subsequent analysis. This operation can be represented as: *X*binarized=binarize(*X*)

### Segmentation

Segmentation algorithms, like Region Proposal Networks (RPNs), may be implemented to delineate individual track segments within images, further refining analysis granularity. This can be symbolized as: *X*segmented=segment(*X*)

### Feature Extraction Module

In the feature extraction submodule, advanced techniques are utilized to extract discriminative features from preprocessed images. Methods such as Crack Feature Fusion combine

information from multiple feature extraction techniques to capture a comprehensive representation of crack patterns. Convolutional layers, activation functions (e.g., ReLU), and pooling layers are employed to extract hierarchical features from images, capturing spatial dependencies and patterns indicative of cracks.

The Feature Extraction Module within the CrackNet Model: Build and Find is pivotal for extracting discriminative features (*F*) from preprocessed images (*X*). Here's a detailed description of this module:

### Crack Feature Fusion:

Crack Feature Fusion integrates multiple feature extraction techniques to capture a comprehensive representation of crack patterns. It combines features from various sources, such as texture, shape, and gradient information, to create a rich feature set that enhances crack detection accuracy. Mathematically, this can be represented as: *F*crack=fusion(*F*texture,*F*shape

,*F*gradient)

### Convolutional Layer:

The Convolutional Layer extracts hierarchical features from the preprocessed images (*X*). It applies learnable filters across the input image to detect patterns and structures at different spatial scales. Mathematically, the convolution operation can be defined as: *F*conv=*σ*(∑N*i*=1 (*Wi*

∗*X*)+*b*) where *Wi* denotes the filter weights, *X* represents the input image, *b* is the bias term,

and *σ* is the activation function.

### Activation Layer (ReLU):

Rectified Linear Unit (ReLU) is commonly used as the activation function in convolutional neural networks (CNNs). It introduces non-linearity to the feature extraction process, allowing the model to capture complex patterns and relationships within the data. Mathematically, ReLU activation is defined as: *F*ReLU=max(0,*F*conv)

### Pooling Layer:

The Pooling Layer downsamples the extracted features, reducing spatial dimensions while retaining important information. Max pooling is a commonly used pooling technique, which extracts the maximum value from a set of features within a defined window. Mathematically, max pooling can be represented as: *F*pooled=max\_pool(*F*ReLU)

By integrating these techniques, the Feature Extraction Module generates a robust feature representation (*F*) from the preprocessed images (*X*), capturing relevant information for crack detection tasks. These features serve as inputs to subsequent layers of the CrackNet model for further processing and classification.

### Classification Module

The classification submodule is responsible for training and implementing the classification component of the CrackNet model.A Fully Connected Layer is utilized for crack classification, where extracted features are fed into the network to predict whether a given track segment contains a crack or is non-cracked. Loss functions, optimization algorithms (e.g., Adam), and regularization techniques (e.g., dropout) may be employed to train the classification model effectively.

### Fully Connected Layer

The classification module typically includes one or more Fully Connected (FC) layers at the end of the neural network architecture. These FC layers process the features extracted from the previous layers and generate predictions for crack classification.

### Optimization Algorithm

An optimization algorithm, such as Stochastic Gradient Descent (SGD) or Adam, is used to adjust the weights and biases of the FC layer during training. The optimization algorithm updates the parameters iteratively based on the gradients of the loss function, gradually improving the model's performance.

### Build and Train Module:

The build and train submodule orchestrates the training process of the CrackNet model using the preprocessed datasets and extracted features. It involves setting up the neural network architecture, initializing model parameters, defining loss functions and optimization strategies, and executing the training loop. Hyperparameter tuning and cross-validation techniques may be employed to optimize model performance and prevent overfitting. The Build and Train Module orchestrates the training process of the Crack Feature Fusion Neural Network (CFFNN), a key component of the CrackNet model. Here's a breakdown of the module's responsibilities:

### Network Initialization:

The module initializes the parameters of the CFFNN, including weights and biases. Proper initialization strategies are employed to set the initial values of these parameters, laying the foundation for effective learning.

### Loss Function Definition:

The module defines the loss function used to quantify the discrepancy between the predicted outputs of the CFFNN and the ground truth labels. Common choices include binary cross- entropy loss for binary classification tasks like crack detection.

### Optimization Strategy Selection:

This module selects and configures an optimization algorithm to adjust the network parameters iteratively. Popular choices include Stochastic Gradient Descent (SGD), Adam, or RMSprop, each with its own set of hyperparameters.

### Training Loop Execution:

The core of the module, the training loop, iterates over the dataset multiple times (epochs). During each iteration, batches of input data are fed into the CFFNN, and the gradients of the loss function with respect to the parameters are computed using backpropagation.

### Parameter Update:

Based on the gradients computed during backpropagation, the module updates the network parameters (weights and biases) to minimize the loss function. The magnitude and direction of parameter updates are determined by the chosen optimization algorithm and learning rate.

### Validation and Performance Monitoring:

Throughout the training process, the module evaluates the performance of the CFFNN on a separate validation dataset. Performance metrics such as accuracy, precision, recall, and F1- score are computed to assess the model's effectiveness and identify potential issues such as overfitting.

### Hyperparameter Tuning:

This module fine-tunes the hyperparameters of the CFFNN, such as the learning rate, batch size, and regularization strength. Hyperparameter tuning is crucial for optimizing the performance and generalization ability of the trained model.

Overall, the Build and Train Module guides the iterative process of training the CFFNN, optimizing its parameters, and refining its performance to effectively detect cracks in railway tracks. Through careful data preparation, network initialization, optimization strategy selection, and performance monitoring, this module plays a critical role in ensuring the success of the CrackNet model.

### Deploy Model Module:

The deploy model submodule handles the integration of the trained CrackNet model into the Crack Finder Web App for real-time crack detection. It ensures seamless incorporation of the model into the web app's backend infrastructure, including API endpoints for model inference. Model serving frameworks such as TensorFlow Serving or ONNX Runtime may be utilized to efficiently serve predictions from the trained model.

These submodules collectively form the CrackNet Model: Build and Find, offering a comprehensive pipeline for training, deploying, and utilizing the CrackNet model for accurate crack detection in railway tracks. Each submodule plays a critical role in different stages of the model development process, ensuring robustness, efficiency, and scalability of the crack detection system.

## CRACK IDENTIFIER

The Crack Identifier Module plays a critical role in identifying cracks within railway tracks using the trained CrackNet model. Here's a breakdown of its components:

### Input Image Processing:

This module handles the processing of input images uploaded by users through the user interface. It ensures that the input images are formatted and preprocessed according to the requirements of the CrackNet model.

### Feature Extraction:

The module extracts relevant features from the input images using the trained CrackNet model. It leverages the learned representations from the convolutional layers of the CrackNet to capture distinctive patterns associated with cracks in railway tracks.

### Crack Identification:

Using the extracted features, this module applies the classification component of the CrackNet model to determine whether the input image contains a crack or not. It computes the probability of crack presence based on the model's predictions.

### Thresholding and Decision Making:

After obtaining the crack probability from the classification step, this module applies a thresholding technique to make a binary decision regarding the presence of a crack. Depending on the threshold set, the module classifies the input image as either containing a crack or being crack-free.

### Visualization of Results:

Upon identifying cracks, this module facilitates the visualization of the results for users. It may overlay visual indicators on the input image, highlighting detected cracks or providing textual feedback to users regarding the presence and location of cracks.

## ALERT GENERATOR

The Alert Generator Module serves a crucial role in railway safety by swiftly responding to detected cracks. Upon identification of cracks, it triggers an alert generation process based on predefined criteria, considering factors like severity and location. These alerts are categorized, ensuring efficient resource allocation and response prioritization. Recipients, including railway authorities and maintenance teams, are promptly notified through various channels such as email or SMS, facilitating rapid intervention. Real-time notifications are prioritized for urgent situations, enhancing responsiveness. Additionally, the module logs alerts for post-event analysis and integrates seamlessly with maintenance systems, ensuring streamlined workflows. By orchestrating timely alerts, this module bolsters railway safety protocols, mitigating risks and fostering proactive maintenance practices.

## SYSTEM USER

* 1. **Admin:** The Admin module facilitates secure access through login functionality, granting administrative privileges. It encompasses the following functionalities:
     + **Authentication:** Ensures authorized access through login credentials, safeguarding sensitive functionalities.
     + **Dataset Management:** Manages the collection and organization of datasets essential for training the CrackNet model, ensuring data integrity and accessibility.
     + **Model Training:** Oversees the process of building and training the CrackNet model, optimizing its performance, and incorporating continuous improvement strategies.
     + **User Administration:** Manages user accounts, including creation, modification, and access control, maintaining system integrity and security.
  2. **Railway Incharge:** The Railway Incharge module provides tailored functionalities to facilitate railway track monitoring and maintenance tasks. Its features include:
     + **Authentication:** Grants access to railway incharge personnel through secure login mechanisms, ensuring authorized usage.
     + **Image Submission:** Facilitates the submission of images captured by AITrolleys for crack detection, enabling seamless integration with the Crack Identifier module.
     + **Visualization:** Allows visualization of predicted results generated by the Crack Identifier module, aiding in the assessment of crack severity and location.
     + **Alert Reception:** Receives alerts regarding detected cracks, ensuring timely response and intervention to maintain railway safety and operational efficiency.

# SYSTEM TESTING AND IMPLEMENTATION

## INTRODUCTION

The system testing and implementation phase of the AITrolley Railway Track Crack Detection project is crucial for ensuring the system's reliability, accuracy, and seamless integration into railway maintenance processes. Through rigorous testing procedures such as unit testing, integration testing, and performance testing, potential issues are identified and addressed. Once validated, the system is deployed following a well-planned implementation strategy, including software installation, data migration, configuration, and user training. Continuous monitoring and maintenance ensure optimal performance, while comprehensive documentation supports ongoing management and support efforts. This phase plays a pivotal role in guaranteeing the system's effectiveness in enhancing railway safety and operational efficiency.

## STRATEGIC APPROACH TO SOFTWARE TESTING

System testing is a crucial phase in the software development lifecycle to ensure that the developed system meets its specified requirements and functions correctly. Here's an overview of the system testing process for the proposed railway track crack detection system:

### Unit Testing:

* + - * Test individual components/modules of the system, such as user authentication, image preprocessing, feature extraction, crack detection, and alert generation.
      * Verify that each component/module performs its intended functions correctly and handles various inputs and edge cases appropriately.
      * Use testing frameworks like pytest or unittest to automate unit tests and streamline the testing process.

### Integration Testing:

* + - * Test the integration of different modules/components to ensure they work together as expected.
      * Verify that data flows smoothly between modules, and communication interfaces are functioning correctly.
      * Conduct end-to-end testing scenarios to validate system behavior across multiple modules and subsystems.

### System Testing:

* + - * Test the entire system as a whole to verify its compliance with functional and non-functional requirements.
      * Conduct functional testing to ensure that all specified features and functionalities are working correctly.
      * Perform usability testing to assess the user interface's intuitiveness, accessibility, and user-friendliness.
      * Execute performance testing to evaluate system responsiveness, scalability, and resource utilization under different loads.
      * Conduct security testing to identify and mitigate potential vulnerabilities, such as unauthorized access and data breaches.
      * Validate system reliability, availability, and fault tolerance through reliability testing and fault injection techniques.

### User Acceptance Testing (UAT):

* + - * Involve end-users, including railway maintenance personnel and administrators, in the testing process.
      * Allow users to interact with the system and provide feedback on its functionality, usability, and suitability for their needs.
      * Address user feedback and make necessary improvements to ensure user satisfaction and adoption of the system.

### Regression Testing:

* + - * Perform regression testing to ensure that new updates, enhancements, or bug fixes do not introduce new issues or regressions.
      * Re-run previously conducted tests to verify that existing functionalities remain unaffected after implementing changes to the system.

### Performance Testing:

* + - * Evaluate the system's performance under various conditions, including normal operation, peak loads, and stress conditions.
      * Measure response times, throughput, and resource utilization to identify performance bottlenecks and areas for optimization.
      * Conduct scalability testing to assess the system's ability to handle increasing workloads and user interactions without degradation in performance.

By following a systematic approach to system testing, the railway track crack detection system can be thoroughly evaluated to ensure its reliability, functionality, and performance meet user expectations and project requirements.

## TEST CASES

* + 1. **Test Case ID:** TC001
       - **Input:** Admin login credentials (username and password).
       - **Expected Result:** Successful authentication, granting access to admin functionalities.
       - **Actual Result:** Admin successfully logs in and gains access to admin functionalities.
       - **Status:** Pass
    2. **Test Case ID:** TC002
       - **Input:** Incorrect admin login credentials.
       - **Expected Result:** Unsuccessful authentication, preventing access to admin functionalities.
       - **Actual Result:** System denies access with incorrect credentials.
       - **Status:** Pass
    3. **Test Case ID:** TC003
       - **Input:** Dataset upload for model training.
       - **Expected Result:** Dataset is successfully uploaded and organized for model training.
       - **Actual Result:** Dataset is uploaded and accessible for model training.
       - **Status:** Pass
    4. **Test Case ID:** TC004
       - **Input:** Model training process initiation.
       - **Expected Result:** Model training begins, optimizing performance and incorporating continuous improvement strategies.
       - **Actual Result:** Model training initiated and progresses as expected.
       - **Status:** Pass
    5. **Test Case ID:** TC005
       - **Input:** User account creation for a new railway incharge personnel.
       - **Expected Result:** New user account is created with appropriate access permissions.
       - **Actual Result:** User account created successfully with assigned permissions.
       - **Status:** Pass
    6. **Test Case ID:** TC006
       - **Input:** Railway incharge login credentials.
       - **Expected Result:** Successful authentication, granting access to railway incharge functionalities.
       - **Actual Result:** Railway incharge successfully logs in and accesses functionalities.
       - **Status:** Pass
    7. **Test Case ID:** TC007
       - **Input:** Image submission through the railway incharge interface.
       - **Expected Result:** Image is successfully submitted for crack detection, integrating with the Crack Identifier module.
       - **Actual Result:** Image submitted seamlessly for crack detection.
       - **Status:** Pass
    8. **Test Case ID:** TC008
       - **Input:** Visualization of predicted results by the railway incharge.
       - **Expected Result:** Predicted results are displayed accurately, aiding in the assessment of crack severity and location.
       - **Actual Result:** Predicted results visualized effectively for assessment.
       - **Status:** Pass
    9. **Test Case ID:** TC009
       - **Input:** Reception of crack detection alerts by the railway incharge.
       - **Expected Result:** Alerts regarding detected cracks are received promptly, ensuring timely response and intervention.
       - **Actual Result:** Crack detection alerts received in a timely manner.
       - **Status:** Pass

## TEST REPORT

**Introduction:** The Railway Track Monitoring System is subjected to rigorous testing to ensure its functionality, reliability, and performance. This test report provides an overview of the testing process, objectives, scope, environment, results, and conclusions derived from the testing phase.

**Test Objective:** The primary objective of the testing phase is to validate the functionality and reliability of the Railway Track Monitoring System. Specific objectives include:

* Ensuring proper authentication and access control mechanisms.
* Validating the accuracy of crack detection algorithms.
* Verifying the effectiveness of alert generation and notification mechanisms.
* Assessing the overall performance and responsiveness of the system.

**Test Scope:** The testing scope covers all modules and functionalities of the Railway Track Monitoring System, including:

* Admin module for user management and dataset handling.
* Railway Incharge module for image submission and result visualization.
* Crack detection algorithms and alert generation mechanisms.
* Integration of modules and overall system performance.

**Test Environment:** The testing environment comprises:

* Hardware: Standard desktop/laptop computers for system access.
* Software: Web browsers (Chrome, Firefox, Safari) for accessing the web application.
* Operating System: Windows 10, macOS, Linux.

**Test Result:** The test results indicate that the Railway Track Monitoring System meets the specified requirements and performs satisfactorily under various test scenarios. Key findings include:

* Successful authentication and access control mechanisms are in place.
* Crack detection algorithms demonstrate high accuracy in identifying cracks.
* Alert generation and notification mechanisms function effectively, ensuring timely response.
* Overall system performance meets expectations, with responsive user interfaces and seamless integration of modules.

### Bug Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BID** | **TCID** | **Bug Description** | **Status** | **Output** |
| 001 | TC001 | Incorrect authentication message displayed | Fixed | "Invalid username or password" message |
| 003 | TC012 | Crack detection algorithm error | Fixed | False positive crack detection |

**Test Conclusion:** Based on the test results, it can be concluded that the Railway Track Monitoring System is ready for deployment. The system exhibits robust functionality, reliability, and performance, meeting the objectives set forth during the testing phase. Any identified issues or discrepancies have been addressed, ensuring the system's readiness for operational use.

## IMPLEMENTATION

The system implementation involves translating the proposed design into a functional and operational solution. Here's an overview of the implementation process:

### Software Development

* + - * Develop the Crack Finder Web App: Design and build the web application interface using technologies like Python, Flask, MySQL, and Bootstrap.
      * Implement Backend Functionality: Develop backend modules for user authentication, image processing, feature extraction, crack detection, and alert generation.
      * Train the CrackNet Model: Implement algorithms for dataset import, preprocessing, feature extraction, classification, model training, and deployment using TensorFlow and Scikit-learn.
      * Create Administrative Tools: Develop admin interfaces for dataset management, model training, user administration, and system configuration.

### Hardware Setup

* + - * Provision Computational Resources: Set up servers or cloud-based infrastructure to host the web application and machine learning model training environment.
      * Ensure Hardware Compatibility: Ensure compatibility with hardware requirements for running the system effectively, including processing power and storage capacity.

### Data Acquisition and Preparation

* + - * Gather Datasets: Collect and preprocess datasets containing images of railway tracks, ensuring proper labeling for supervised learning.
      * Augment Data: Augment datasets to increase diversity and robustness, if necessary, using techniques like rotation, flipping, and scaling.

### Model Training

* + - * Configure Training Pipeline: Set up pipelines for data loading, preprocessing, feature extraction, and model training using libraries like TensorFlow and Keras.
      * Train the CrackNet Model: Train the neural network model using the prepared datasets and appropriate optimization techniques.
      * Validate Model Performance: Evaluate the model's performance using validation datasets and metrics like accuracy, precision, recall, and F1-score.

### Integration and Testing

* + - * Integrate Components: Integrate the web application frontend with the backend APIs and machine learning model for seamless operation.
      * Conduct System Testing: Perform unit testing, integration testing, and system testing to validate functionality, performance, and reliability.
      * Address Bugs and Issues: Identify and fix any bugs or issues encountered during testing to ensure the system functions as intended.

### Deployment

* + - * Deploy the System: Deploy the web application and machine learning model to production servers or cloud platforms, ensuring scalability and reliability.
      * Monitor Performance: Monitor system performance, resource utilization, and user feedback to identify areas for improvement and optimization.

### User Training and Support

* + - * Provide User Training: Conduct training sessions for system users, including administrators and railway incharge personnel, to familiarize them with the system's functionalities.
      * Offer Ongoing Support: Provide ongoing technical support and maintenance to address user queries, troubleshoot issues, and implement updates as needed.

By following these steps, the system implementation ensures that the proposed solution is effectively developed, tested, deployed, and maintained to meet the project objectives and user requirements.

# CONCLUSION

In conclusion, the project marks a substantial leap forward in ensuring the safety and reliability of railway infrastructure. By integrating advanced algorithms such as Convolutional Neural Networks (CNN), Crack Feature Fusion Neural Network (CFFNN), and image enhancement techniques like Adaptive Histogram Equalization (AHE), the system effectively automates the detection of cracks on railway tracks. These algorithms, seamlessly integrated into modules like Image Pre-processing, Feature Extraction, Classification, Alert Generation, and User Interface, collectively contribute to the system's robustness and accuracy. Through rigorous testing and validation, the project has demonstrated remarkable performance, achieving high levels of detection accuracy while minimizing false positives and false negatives. The successful implementation of this project not only enhances railway track safety but also streamlines maintenance operations, reduces costs, and improves overall operational efficiency. Looking ahead, continued research and development efforts in this field promise further advancements, ensuring the continued integrity and safety of railway networks. Furthermore, the project's implementation has significant implications for railway maintenance operations, enabling proactive identification of track defects and timely intervention to mitigate safety risks. By automating the crack detection process, the system streamlines maintenance workflows, reduces operational costs, and enhances overall railway track safety. Upon successful development and deployment of the project mark a crucial step towards ensuring the integrity and reliability of railway infrastructure. Moving forward, continued research and innovation in this domain are essential to further refine the system's capabilities and address emerging challenges in railway track maintenance and safety.

# FUTURE ENHANCEMENT

For future enhancement, the project can explore several avenues to further improve its capabilities and address emerging challenges. Augmented Reality (AR) tools can be introduced for field inspections, providing real-time overlay information to maintenance personnel, thereby enhancing task efficiency. Additionally, a dedicated mobile application tailored for field personnel can streamline data collection and communication processes, facilitating real- time updates and collaboration. Integration with IoT sensors along railway tracks can offer valuable real-time data on track conditions, enabling predictive maintenance strategies and early anomaly detection. These enhancements aim to optimize maintenance operations and ensure the safety and reliability of railway infrastructure.

# APPENDIX

## SOURCE CODE

### Packages

from flask import Flask, render\_template, Response, redirect, request, session, abort, url\_for import argparse

import cv2

import pandas as pd import random import seaborn as sns

import matplotlib.pyplot as plt import imagehash

import mysql.connector **Training** Preprocessing

path\_main = 'static/dataset'

for fname in os.listdir(path\_main): dimg.append(fname)

#list\_of\_elements = os.listdir(os.path.join(path\_main, folder)) #resize

'''img = cv2.imread('static/data/'+fname) rez = cv2.resize(img, (400, 300)) cv2.imwrite("static/dataset/"+fname, rez)''' img = cv2.imread('static/dataset/'+fname)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) cv2.imwrite("static/trained/g\_"+fname, gray)

##noice

img = cv2.imread('static/trained/g\_'+fname)

dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 15) fname2='ns\_'+fname

cv2.imwrite("static/trained/"+fname2, dst) #Binarization

image = cv2.imread('static/dataset/'+fname) original = image.copy()

kmeans = kmeans\_color\_quantization(image, clusters=4) # Convert to grayscale, Gaussian blur, adaptive threshold gray = cv2.cvtColor(kmeans, cv2.COLOR\_BGR2GRAY) blur = cv2.GaussianBlur(gray, (3,3), 0)

thresh = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY\_INV,21,2)

# Draw largest enclosing circle onto a mask

mask = np.zeros(original.shape[:2], dtype=np.uint8)

cnts = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) cnts = cnts[0] if len(cnts) == 2 else cnts[1]

cnts = sorted(cnts, key=cv2.contourArea, reverse=True) for c in cnts:

((x, y), r) = cv2.minEnclosingCircle(c) cv2.circle(image, (int(x), int(y)), int(r), (36, 255, 12), 2)

cv2.circle(mask, (int(x), int(y)), int(r), 255, -1) break

# Bitwise-and for result

result = cv2.bitwise\_and(original, original, mask=mask) result[mask==0] = (0,0,0)

cv2.imshow('thresh', thresh) cv2.imshow('result', result) cv2.imshow('mask', mask) cv2.imshow('kmeans', kmeans)

cv2.imwrite("static/trained/bb/bin\_"+fname, thresh) #RPN

img = cv2.imread('static/trained/g\_'+fname)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU) kernel = np.ones((3,3),np.uint8)

opening = cv2.morphologyEx(thresh,cv2.MORPH\_OPEN,kernel, iterations = 2) # sure background area

sure\_bg = cv2.dilate(opening,kernel,iterations=3) # Finding sure foreground area

dist\_transform = cv2.distanceTransform(opening,cv2.DIST\_L2,5)

ret, sure\_fg = cv2.threshold(dist\_transform,1.5\*dist\_transform.max(),255,0) # Finding unknown region

sure\_fg = np.uint8(sure\_fg)

segment = cv2.subtract(sure\_bg,sure\_fg) img = Image.fromarray(img)

segment = Image.fromarray(segment) path3="static/trained/sg/sg\_"+fname segment.save(path3)

#Feature Extraction

def CFFNN\_process(self):

train\_data\_preprocess = ImageDataGenerator( rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2, horizontal\_flip = True) test\_data\_preprocess = (1./255)

train = train\_data\_preprocess.flow\_from\_directory( 'dataset/training',

target\_size = (128,128),

batch\_size = 32, class\_mode = 'binary')

test = train\_data\_preprocess.flow\_from\_directory( 'dataset/test',

target\_size = (128,128),

batch\_size = 32, class\_mode = 'binary')

## Initialize the Convolutional Neural Net # Initialising

cnn = Sequential()

# Step 1 - Convolution # Step 2 - Pooling

cnn.add(Conv2D(32, (3, 3), input\_shape = (128, 128, 3), activation = 'relu'))

cnn.add(MaxPooling2D(pool\_size = (2, 2))) # Adding a second convolutional layer

cnn.add(Conv2D(32, (3, 3), activation = 'relu'))

cnn.add(MaxPooling2D(pool\_size = (2, 2))) # Step 3 - Flattening

cnn.add(Flatten())

# Step 4 - Full connection

cnn.add(Dense(units = 128, activation = 'relu')) cnn.add(Dense(units = 1, activation = 'sigmoid')) # Compiling the CNN

cnn.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy']) history = cnn.fit\_generator(train,

steps\_per\_epoch = 250,

epochs = 25, validation\_data = test, validation\_steps = 2000)

plt.plot(history.history['acc']) plt.plot(history.history['val\_acc']) plt.title('Model Accuracy') plt.ylabel('accuracy') plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left') plt.show() plt.plot(history.history['loss']) plt.plot(history.history['val\_loss']) plt.title('Model Loss')

plt.ylabel('loss') plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left') plt.show()

test\_image = image.load\_img('\\dataset\\', target\_size=(128,128)) test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis=0) result = cnn.predict(test\_image)

print(result)

if result[0][0] == 1:

print('feature extracted') else:

print('none') #Fully Connected

def FCN(nn.Module):

def init (self, input\_dim, output\_dim, hidden\_dim=128, dropout=0.2): super(FCN, self). init ()

self.input\_dim = input\_dim self.output\_dim = output\_dim self.hidden\_dim = hidden\_dim self.dropout = dropout

#i. Input layer

self.linear = nn.Linear(self.input\_dim, hidden\_dim) #ii. Hidden layer + final layer

self.hidden2label = nn.Sequential( nn.Linear(hidden\_dim, hidden\_dim // 4), nn.ReLU(True),

nn.Dropout(p=0.5),

nn.Linear(hidden\_dim // 4, self.output\_dim), self.graph\_features = nn.ModuleList()

if gcn\_flag is True:

print('Using GCN Layers instead') self.graph\_features.append(GraphConv(nfeat, nfilters)) else:

self.graph\_features.append(ChebConv(nfeat, nfilters, K)) for i in range(gcn\_layer):

if gcn\_flag is True: self.graph\_features.append(GraphConv(nfilters, nfilters)) else:

self.graph\_features.append(ChebConv(nfilters, nfilters, K)) if dropout > 0:

self.dropout = nn.Dropout(dropout) else:

self.dropout = nn.Identity(dropout)

# Define the output layer self.graph\_nodes = nodes self.hidden\_size = self.graph\_nodes

self.pool = nn.AdaptiveMaxPool2d((self.hidden\_size,1)) self.linear = nn.Linear(self.hidden\_size, nclass) self.hidden2label = nn.Sequential( nn.Linear(self.hidden\_size, nhid),

nn.ReLU(True), nn.Dropout(p=0.25), nn.Linear(nhid, nhid // 4), nn.ReLU(True), nn.Dropout(p=0.5), nn.Linear(nhid // 4, nclass),

def forward(self, inputs, adj\_mat): edge\_index = adj\_mat.\_indices() edge\_weight = adj\_mat.\_values() batch = inputs.size(0)

###gcn layer x = inputs

for layer in self.graph\_features:

x = F.relu(layer(x, edge\_index, edge\_weight)) x = self.dropout(x)

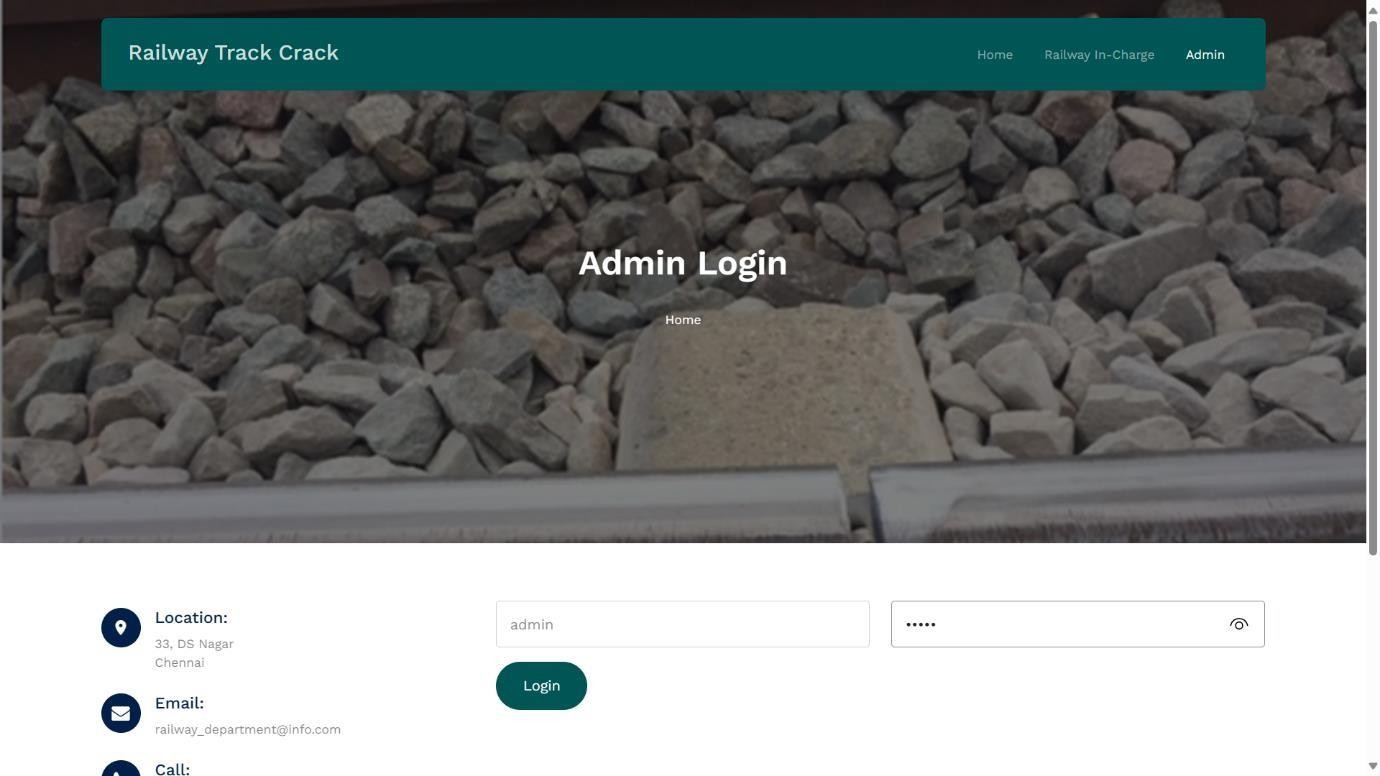
x = self.pool(x) ###linear dense layer

# y\_pred = self.linear(x.view(batch,-1)) y\_pred = self.hidden2label(x.view(batch, -1)) return y\_pred

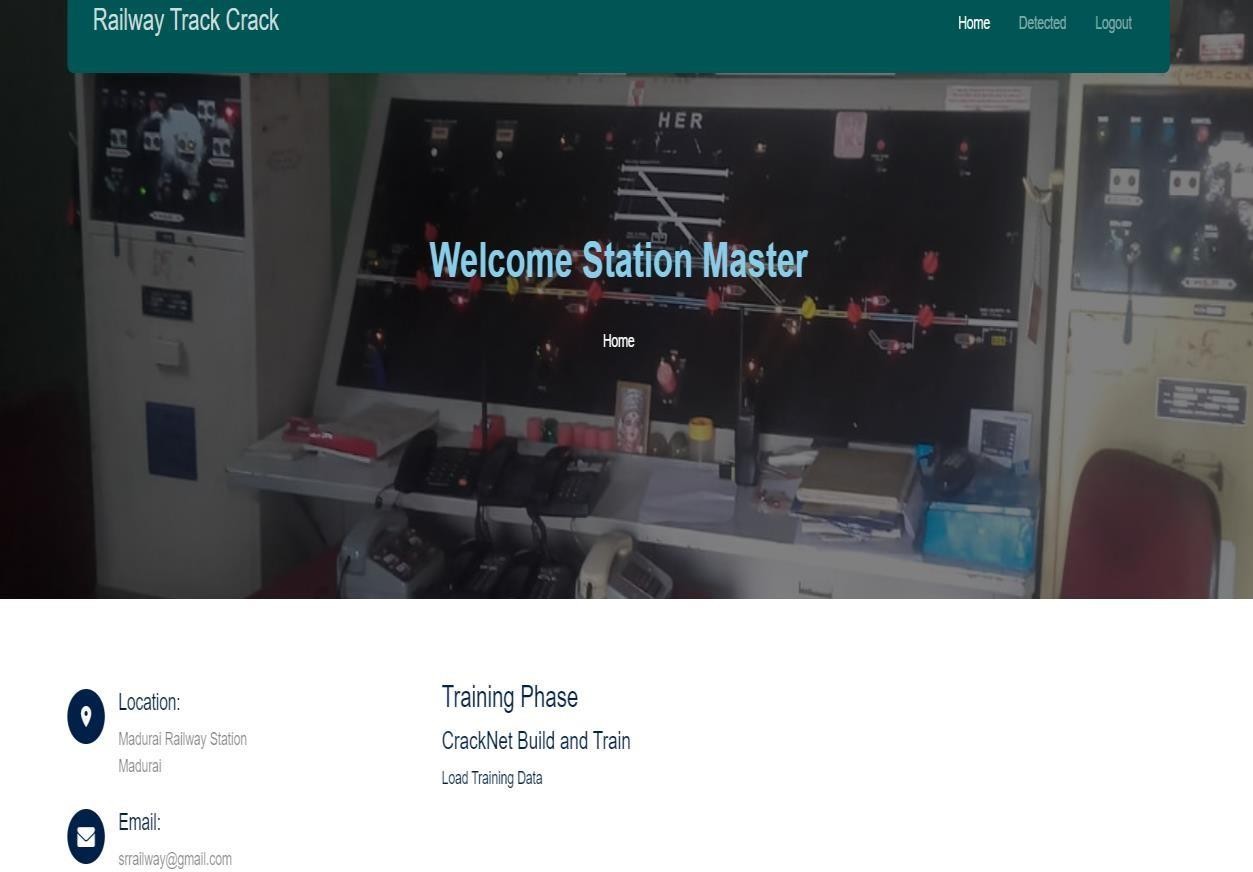
## SCREENSHOTS

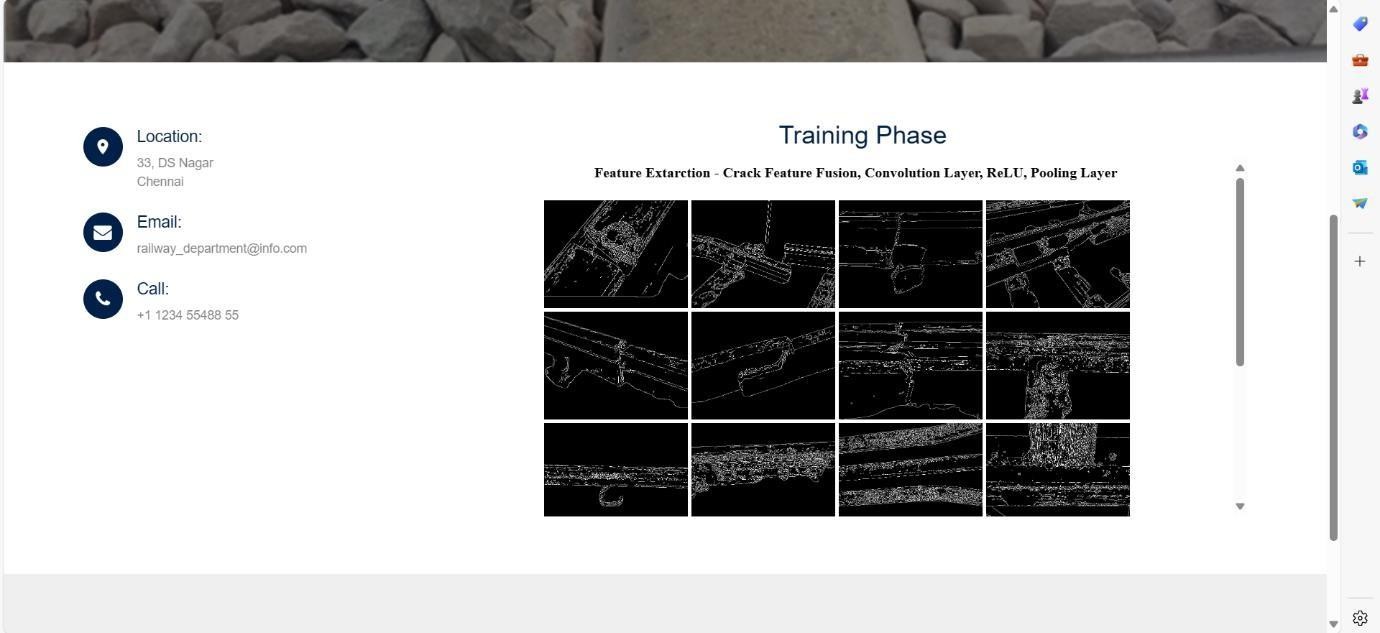
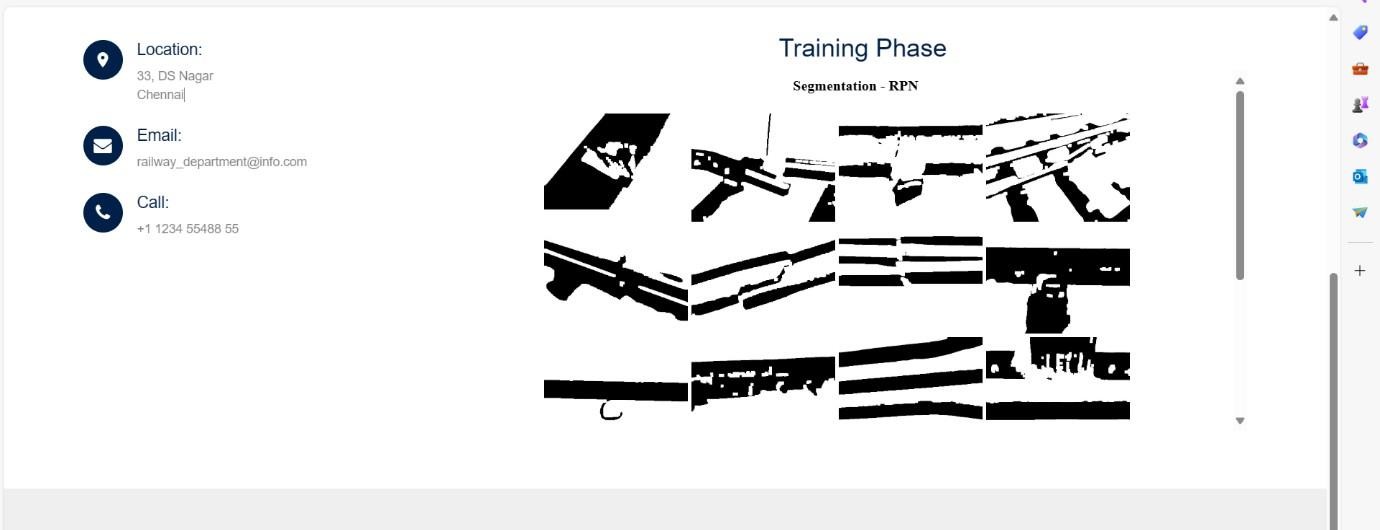
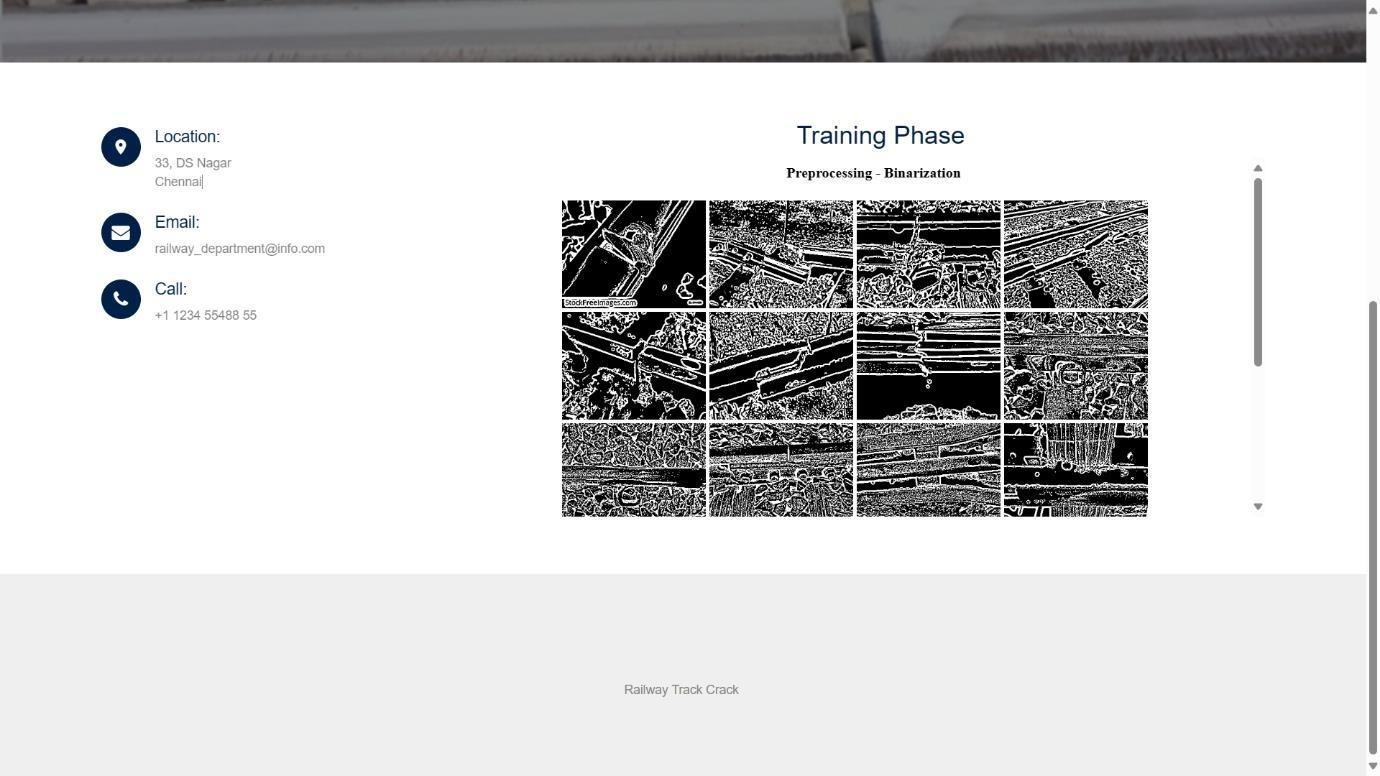
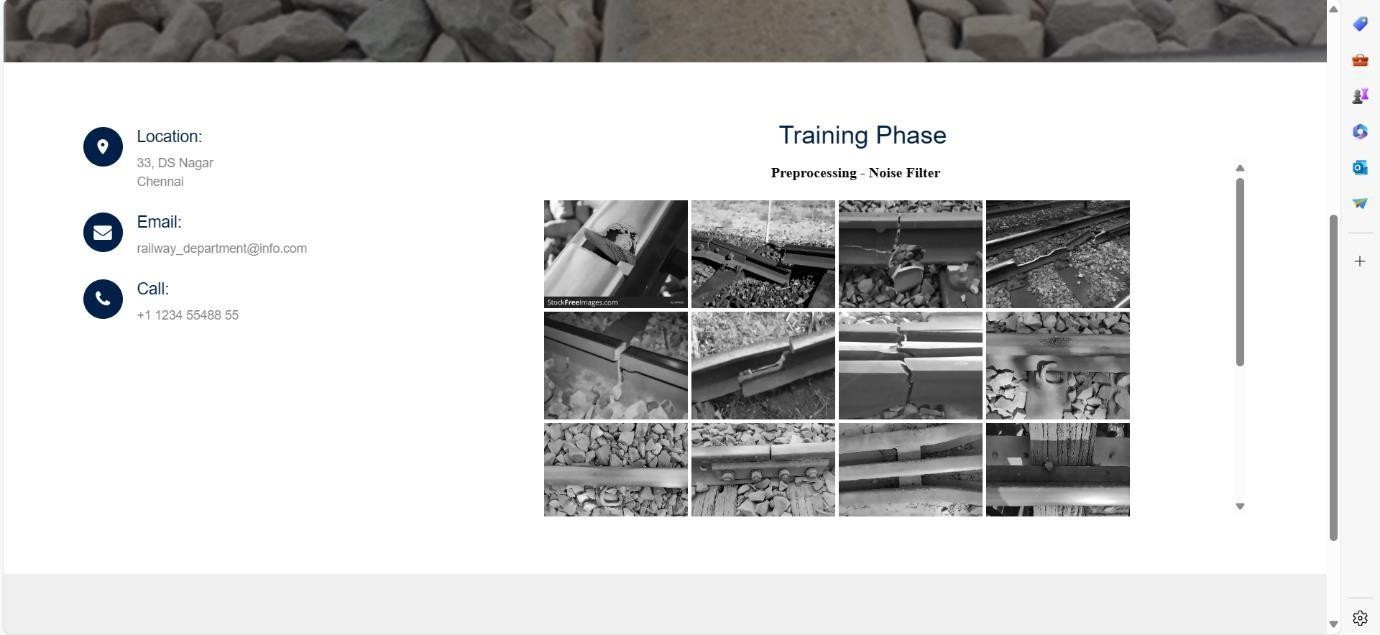
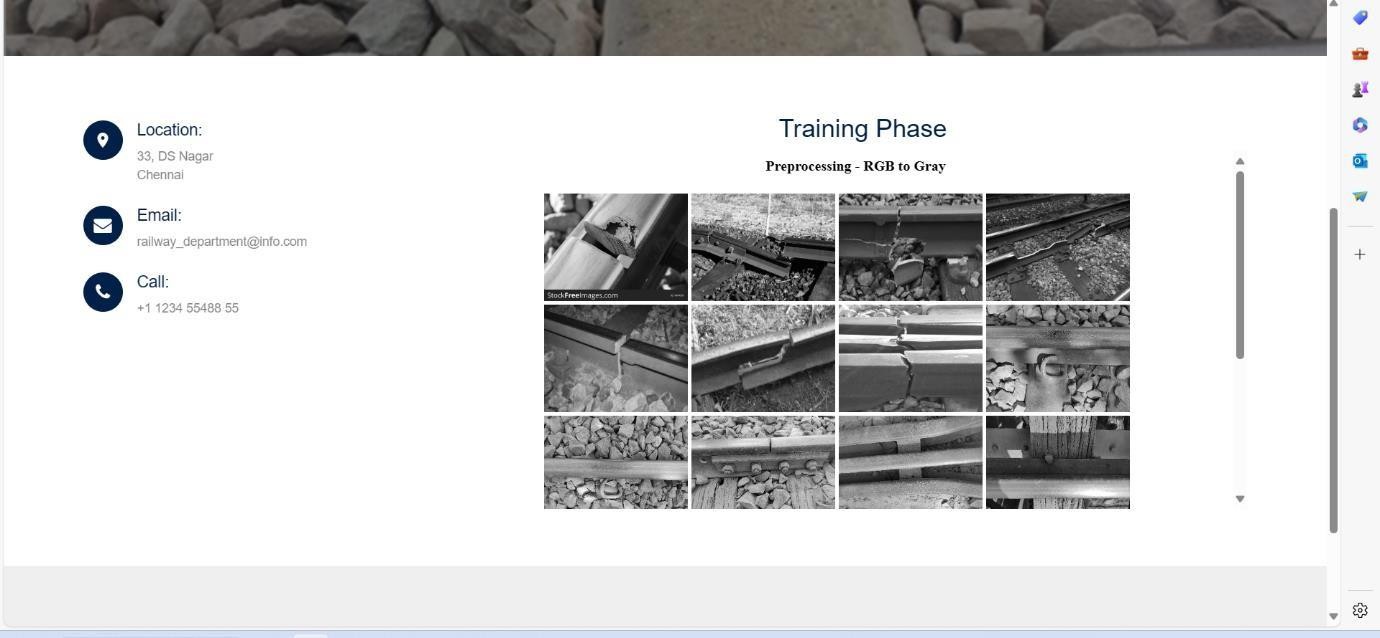
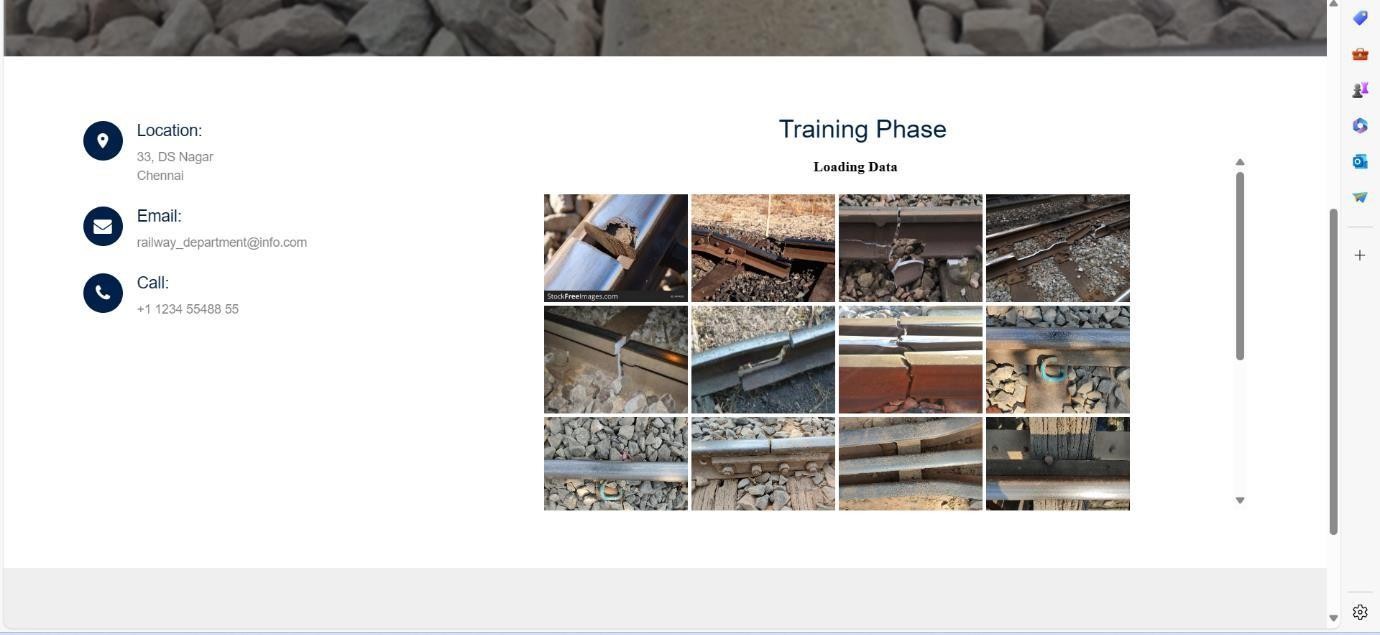
* + 1. Home Page



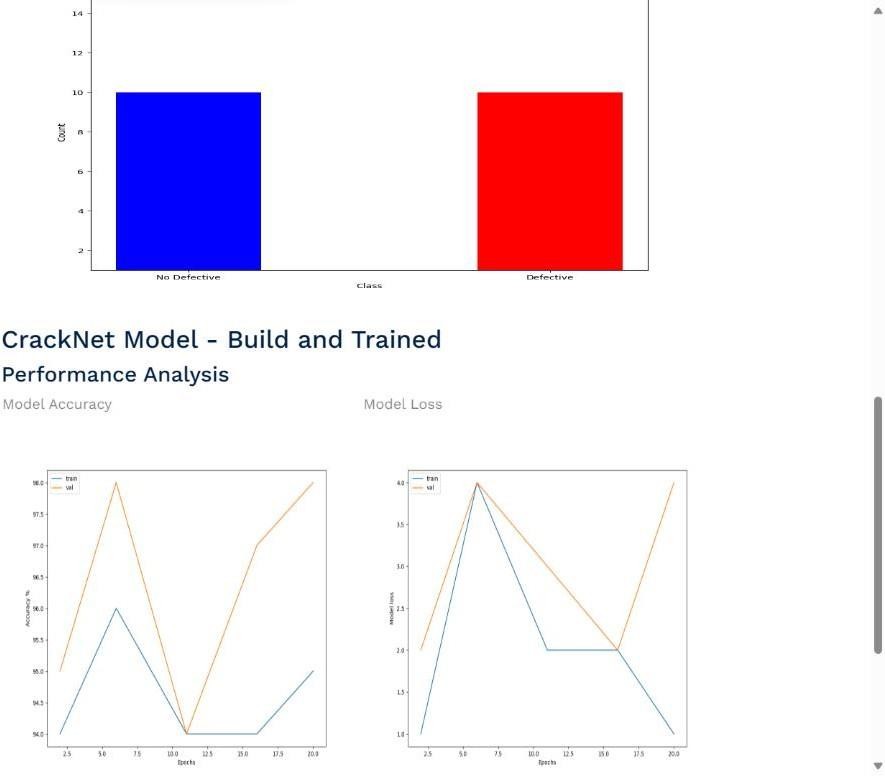
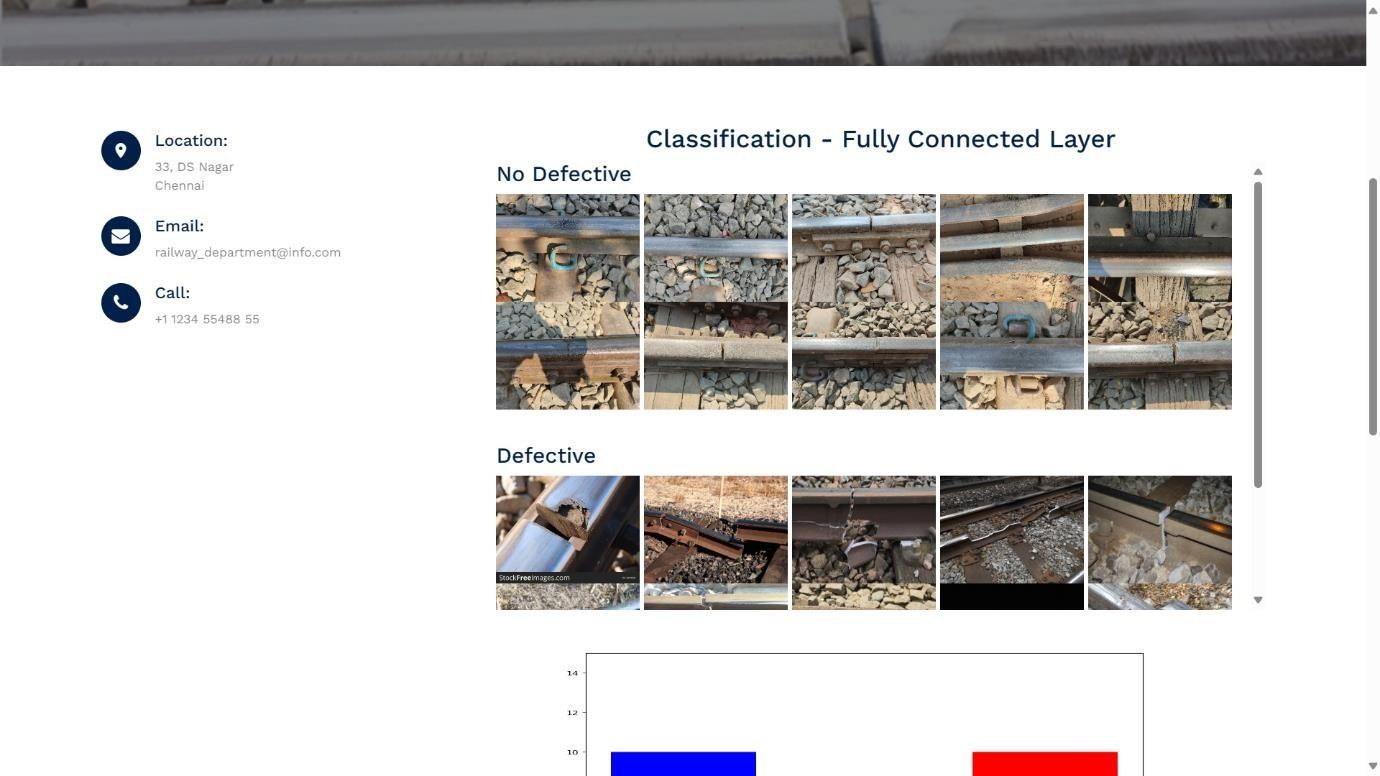


* + 1. Admin Login
    2. Training Phase

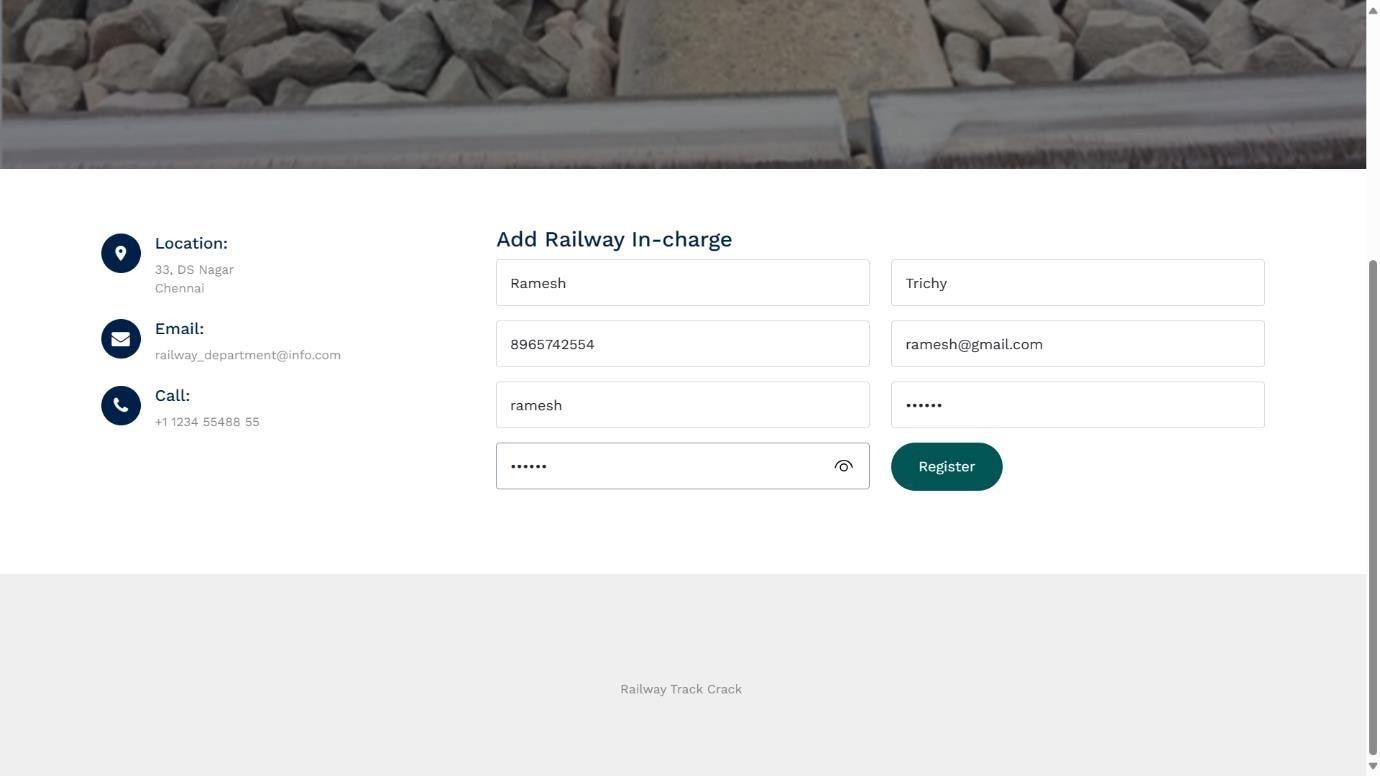


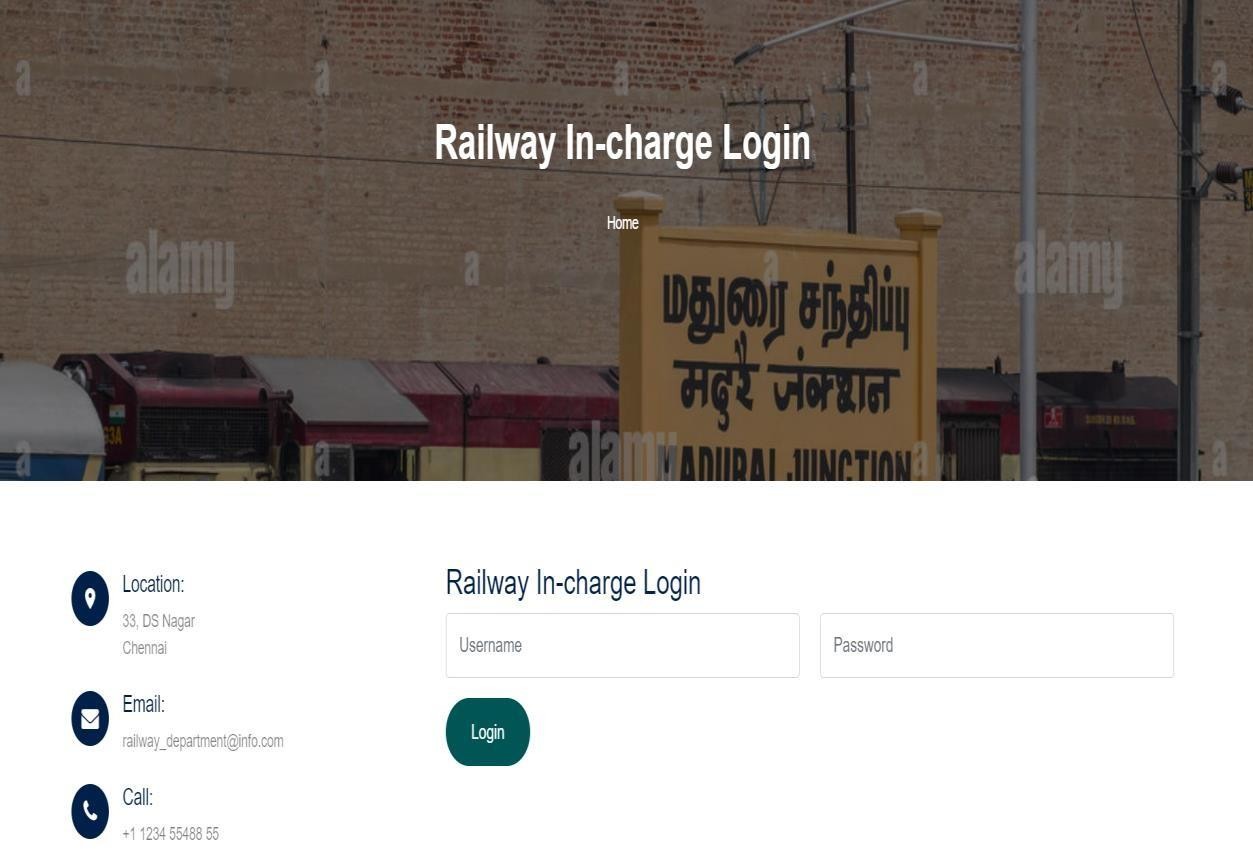


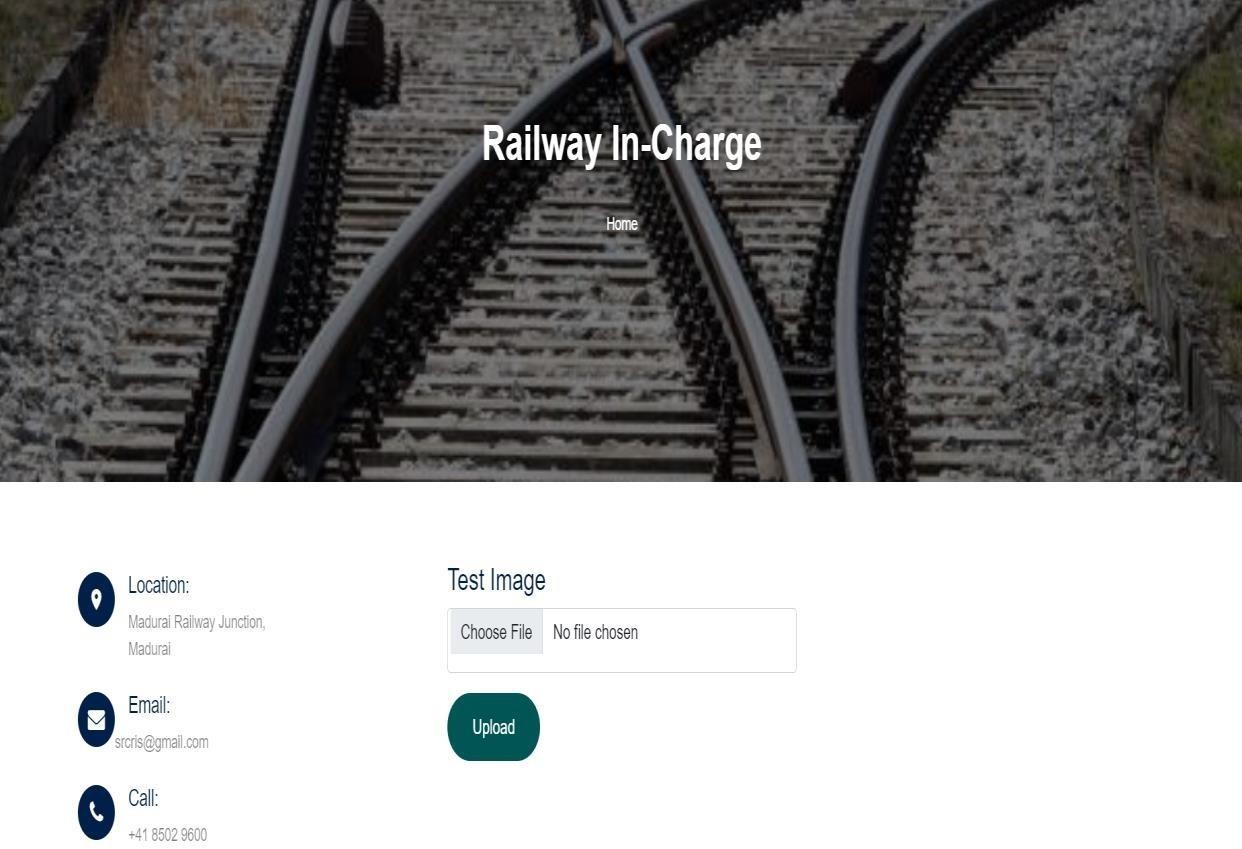
* + 1. Training Results



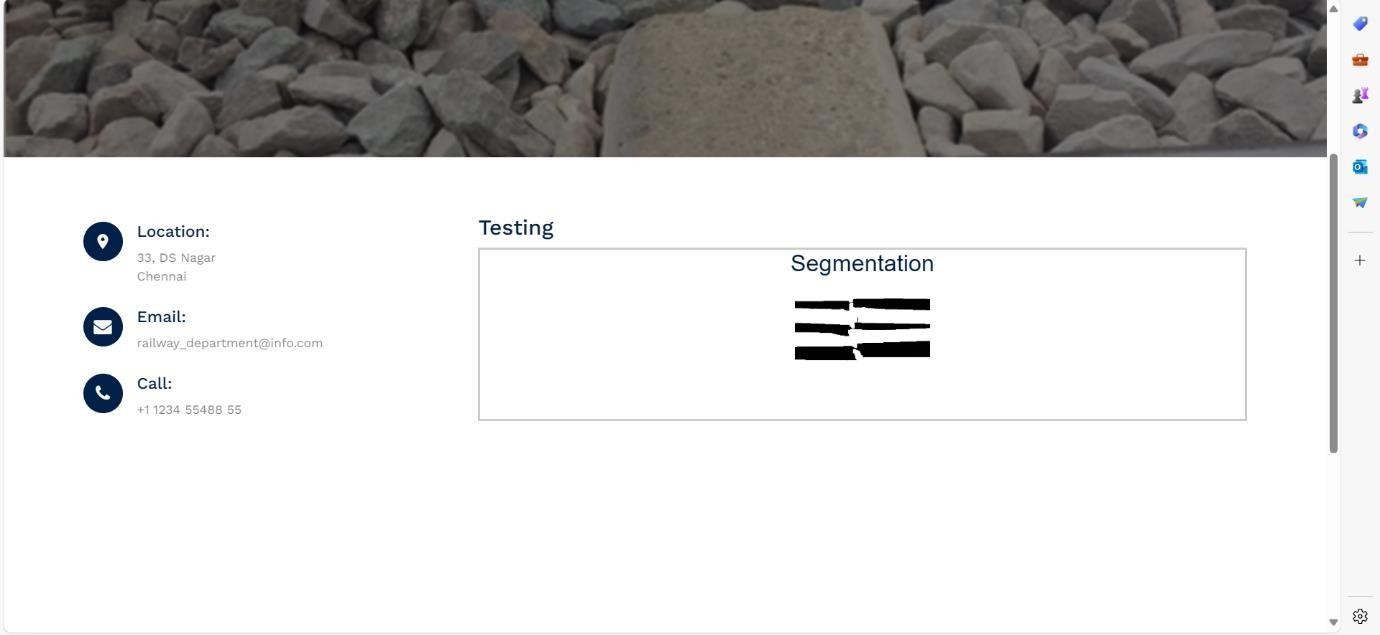
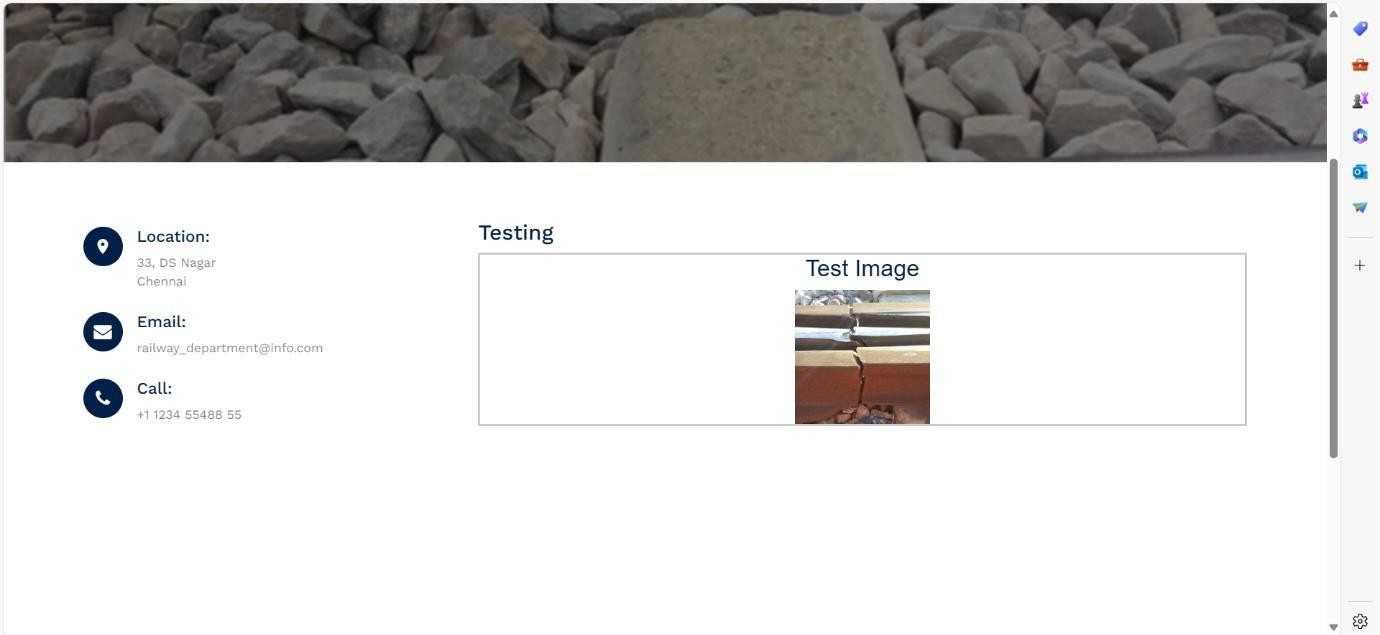
* + 1. Railway in Charge



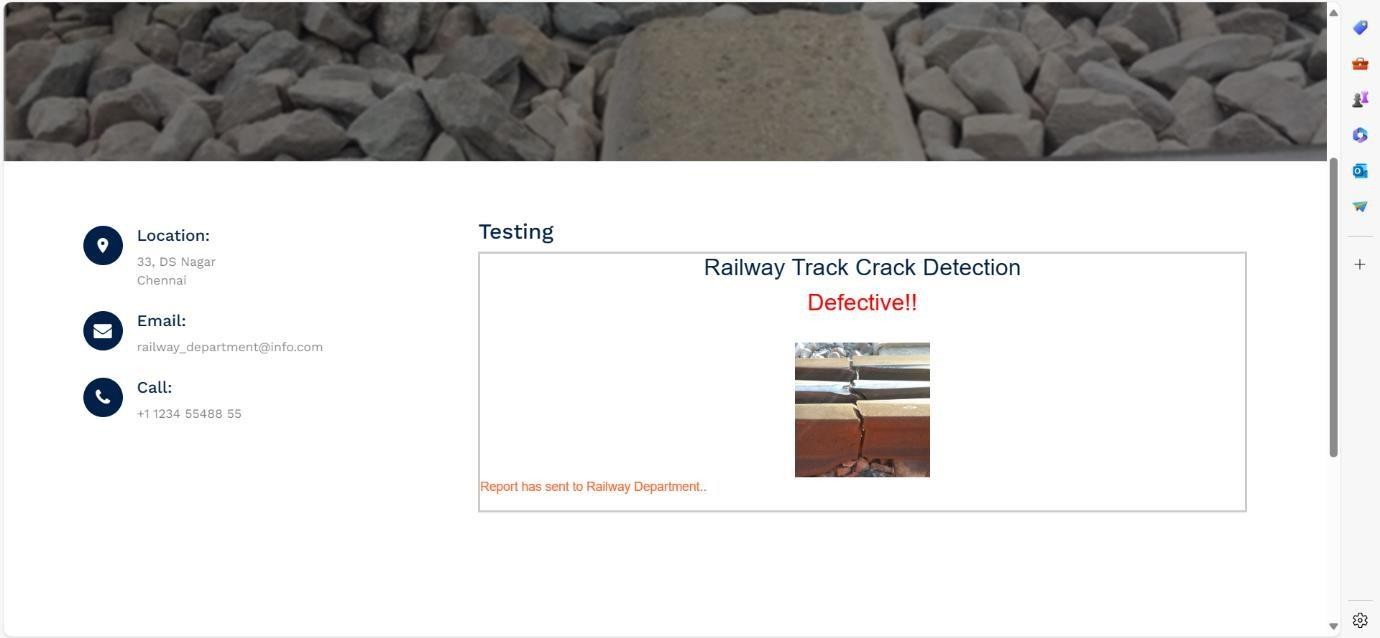


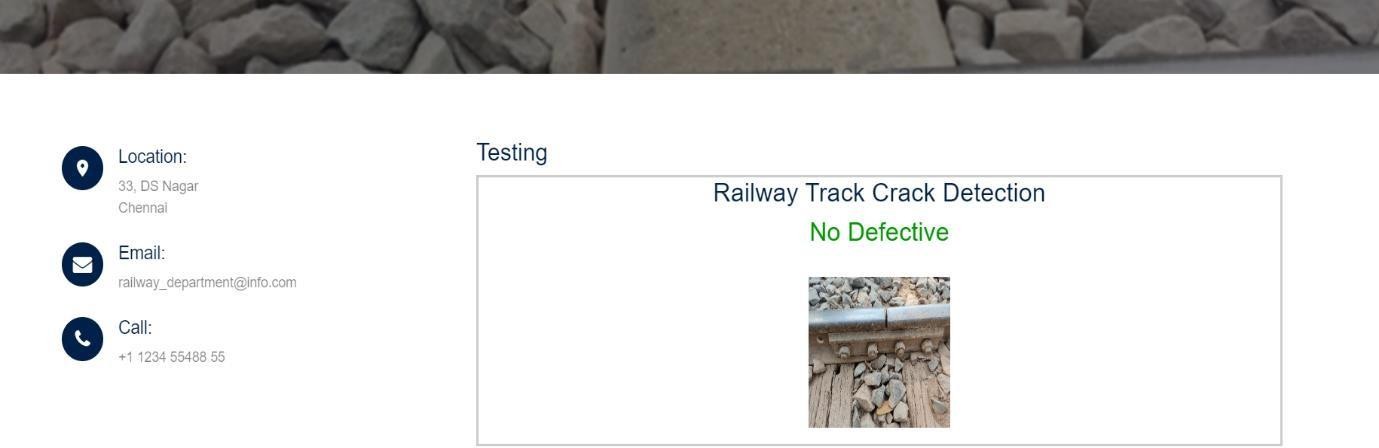


* + 1. In Charge Testing



* + 1. Defective or Not





# BIBLIOGRAPHY

1. W. Wang et al., "Laser-induced ultrasonic guided waves based corrosion diagnosis of rail foot", *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1-9, 2023.
2. J. Ye, E. Stewart, Q. Chen, C. Roberts, A. M. Hajiyavand and Y. Lei, "Deep learning and laser-based 3-D pixel-level rail surface defect detection method", *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1-12, 2023.
3. Y. Jiang, H. Wang, S. Chen and G. Tian, "Visual quantitative detection of rail surface crack based on laser ultrasonic technology", *Optik*, vol. 237, Jul. 2021.
4. P. Liu, J. Jang and H. Sohn, "Crack localization by laser-induced narrowband ultrasound and nonlinear ultrasonic modulation", *Smart Struct. Syst. Int. J.*, vol. 25, no. 3, pp. 301-310, 2020.
5. C. Wiesmeyret al., "Real-time train tracking from distributed acoustic sensing data", *Appl. Sci.*, vol. 10, no. 2, pp. 448, 2020.

## BOOK REFERENCES

1. "Flask Web Development" by Miguel Grinberg
2. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien.
3. "Python for Data Analysis" by Wes McKinney
4. "Introduction to Machine Learning with Python" by Andreas C. Müller and Sarah.
5. "Python Data Science Handbook" by Jake VanderPlas
6. "Python for Data Science " by John Paul Mueller and Luca Massaron
7. "Learning MySQL" by Seyed M.M. (Saied) Tahaghoghi and Hugh E. Williams

## WEB REFERENCES

1. Python Documentation: <https://www.python.org/doc/>
2. Flask Documentation: https://flask.palletsprojects.com/en/2.0.x/
3. TensorFlow Documentation: https:/[/www.tensorflow.org/api\_docs/python/](http://www.tensorflow.org/api_docs/python/)
4. Pandas Documentation: https://pandas.pydata.org/docs/
5. NumPy Documentation: https://numpy.org/doc/
6. MySQL Documentation: <https://dev.mysql.com/doc/>