A MAJOR PROJECT MID TERM REPORT ON

AI-Enhanced: Smart Community Complaint App

Submitted by:

Aashutosh Sapkota	**BEC [2077-01]**
Abhishek Sharma	**BEC [2077-02]**
Hom Bdr. Pathak Kshetri	**BEC [2077-16]**
Rajeev Paudel	**BEC [2077-24]**

United Technical College

Faculty of Science and Technology

Pokhara University, Nepal

June 22, 2025

Abstract

The Smart Community Complaint App is a mobile application that helps citizens report issues like potholes, cracks, open manholes, animals, traffic lights, and waste containers. Users can upload images, use location services, and get updates on their complaints. The app uses AI to automatically detect and classify problems in the uploaded images, making it easier for authorities to respond quickly. A custom-trained YOLOv5 model is used to detect problems such as potholes, cracks, and open manholes etc. When users submit images, the model highlights and classifies the issues, linking them to the complaint system. This helps in faster maintenance and better issue tracking. The app is built using Flutter, Dart, Firebase, and Python and is expected to be completed in 10 weeks.

KEYWORDS: *Smart Complaint, YOLOv5, Object Detection, Potholes, Cracks, Manholes, Flutter, Firebase, Python., Artificial Intelligence, etc.*

TABLE OF CONTENTS

List of Figures

List of Table

ACRONYMS/ABBREVIATIONS

AI	:	Artificial Intelligence
CUDA	:	Compute Unified Device Architecture
cuDNN	:	CUDA deep neural network
ERD	:	Entity Relationship Diagram
GDP	:	Gross Domestic Product
GPS	:	Global Positioning System
GPU	:	Graphics Processing Unit
HTTP	:	Hypertext Transfer Protocol
IoU	:	Intersection Over Union
mAP	:	Mean Average Precision
ML	:	Machine Learning
MS COCO	:	Microsoft Common Objects in Context
NMS	:	Non-Max Suppression
OID	:	Open Image Dataset
RAM	:	Random Access Memory
RDBMS	:	Relational Database Maintenance System
SDK	:	Software Development Kit
VS Code	:	Visual Studio Code
YOLO	:	You Only Look Once

Chapter 1: INTRODUCTION

1.1. Background

Nepal's road infrastructure faces severe challenges, with 77% of national highways and 82% of feeder roads in poor condition, as per the Economic Survey of 2018/19. This deterioration increases vehicle operating costs, journey times, and fuel consumption. Moreover, delays in major projects like the Postal Highway Project, where 78% of work has been delayed due to design mistakes and weak contract management, highlight systemic inefficiencies. Such issues emphasize the need for a streamlined reporting mechanism to address road-related problems effectively (Maharjan, 2019).

Urban areas in Nepal, particularly the Kathmandu Valley with over 4 million residents, suffer from severe traffic congestion. Insufficient public transport and uncoordinated planning among federal, provincial, and municipal agencies make the problem worse. Roads are often dug up multiple times a year for maintenance by different agencies without coordination, worsening traffic conditions. Addressing these issues requires citizen engagement to report traffic bottlenecks and advocate for better management (Thapa, 2024).

Access to clean drinking water remains a critical issue. In Kathmandu Valley alone, the daily water demand is 400 million liters, but supply falls far short of this requirement. Projects like the Melamchi Drinking Water Project have faced prolonged delays due to bureaucratic hurdles and poor governance. Citizens frequently face water shortages and poor-quality supply, necessitating a platform to report such grievances (Purbey, 2023).

Poor drainage systems in Nepal lead to frequent flooding during heavy rains, particularly in urban areas like Kathmandu, where blocked or inadequate drainage causes regular waterlogging and health risks. Additionally, the country experiences frequent power outages due to aging infrastructure and natural disasters, with landslides damaging electricity towers and prolonging outages in rural regions.

Waste management is another significant challenge, with delays in garbage collection and illegal dumping disrupting sanitation efforts, especially in Bharatpur due to landfill disputes. Stray animals also pose safety concerns for pedestrians and can lead to accidents on the roads. Moreover, various other community issues, such as broken streetlights and environmental hazards, often go unreported due to the lack of accessible platforms (Thapa, 2024).

To tackle the diverse challenges faced by communities, the Smart Community Complaint App provides a platform for citizens to report issues related to potholes, cracks, open manholes, traffic lights, waste containers, and animal concerns. This application empowers users to actively participate in improving their surroundings by directly communicating with relevant authorities (Purbey, 2023).

The main objective of the Smart Community Complaint App is to foster transparent collaboration between citizens and government organizations. By leveraging citizen engagement and advanced technologies like artificial intelligence for image recognition, the system streamlines the reporting and resolution of problems such as potholes, traffic congestion, water shortages, and power outages (Thapa, 2024).

This app offers a promising solution to enhance community infrastructure and services. Promoting active participation and efficient collaboration has the potential to improve living conditions significantly, leading to safer roads and better overall community well-being.

1.2. Problem statement

Nepal faces critical infrastructure challenges across roads, traffic, and waste management. Poor road maintenance, traffic congestion in urban areas like Kathmandu Valley, inadequate drainage causing monsoon flooding, unreliable electricity supply, and delays in waste collection due to landfill disputes are common issues. Additionally, access to clean drinking water remains insufficient, and stray animals on roads pose safety risks (Lamichhane, 2024).

A major issue is the lack of a platform for citizens to report these problems, leaving concerns like potholes, drainage blockages, and power outages unaddressed. This gap undermines public trust and perpetuates inefficiencies. This lack of responsiveness undermines public trust in local governance (Adhikari, 2024).

The Smart Community Complaint App aims to address these challenges by providing a user-friendly platform for citizens to report issues in real-time across various categories roads, traffic, drinking water, drainage, electricity, waste management, animal concerns, and more. By leveraging technology for efficient complaint resolution processes, the app seeks to improve infrastructure quality and enhance public confidence in governance (Acharya, 2023).

1.3. Objectives

The objectives of the proposed system are as follows:

1. To develop a mobile application using Flutter for reporting various community issues, including roads, traffic, drinking water, drainage, electricity, waste management, and animal concerns.
2. To integrate features like artificial intelligence for image recognition to streamline issue identification and prioritization, along with an offline submission option for areas with limited internet connectivity.
3. To empower citizens with a platform that encourages active participation in community improvement and fosters collaboration with local authorities.

1.4. Motivation and Significance

Nepal's existing municipal complaint systems often lack real-time tracking and efficient response mechanisms. Citizens frequently face challenges reporting issues through traditional channels like phone calls or physical visits, leading to delayed responses and inadequate resolutions. Existing mechanisms for lodging grievances are inefficient, time-consuming, and often lack transparency. Many citizens are unaware of how to contact the concerned authorities, and the process often involves

bureaucratic delays. The rapid development of mobile technology and AI provides an opportunity to bridge this gap by creating a platform that simplifies issue reporting and ensures prompt action. The motivation to choose this topic stems from the need to empower citizens, enhance government accountability, and foster smarter urban management.

Significance:

The system introduces AI-powered categorization and prioritization to enhance the efficiency of complaint management. By incorporating real-time location tracking and photo evidence submission, it ensures that each complaint is grounded in reliable and verifiable data. The automatic routing mechanism directs issues to the relevant government departments, reducing delays and manual effort. Additionally, integrated smart analytics help identify recurring problems and urban patterns that may require policy-level attention.

Practical Impact:

The platform contributes to improved transparency in handling public grievances by ensuring that complaints are systematically documented, tracked, and responded to. It facilitates data driven insights that can support urban planning and resource allocation. Furthermore, the system strengthens the communication bridge between citizens and local authorities, promoting quicker resolutions and accountability in service delivery.

Technical Contribution:

The solution is developed using Flutter, allowing seamless deployment across multiple platforms with a single codebase. Machine learning models are embedded to automatically classify complaints based on categories, urgency, and departmental relevance. A real-time database architecture supports the end-to-end tracking of complaints, enabling timely updates, status monitoring, and efficient data handling.

1.5. Application

The proposed system will enable citizens in Nepal to report various civic issues through an AI-driven mobile application, covering the following areas:

1. Waste Management: Notify authorities about improper garbage disposal and collection issues.
2. Traffic Issues: Report congestion, accidents, or roadblocks affecting traffic flow.
3. Animal Issues: Report stray animals causing safety concerns.
4. Potholes: Report dangerous potholes that could damage vehicles or cause accidents.
5. Cracks: Notify authorities about road surface cracks that may lead to structural failure or accidents.
6. Open Manholes: Report uncovered manholes that pose serious risks to pedestrians and vehicles.

1.6. Limitation

1. **Data Reliability**: The accuracy and reliability of user-generated data may vary, requiring the implementation of validation and verification mechanisms to ensure credible reporting.
2. **Connectivity and Access**: Effective use of the system depends on a stable internet connection and access to smartphones or internet-enabled devices, which may limit its functionality in areas with connectivity challenges.

Chapter 2: LITERATURE REVIEW

The use of mobile-based solutions for managing community issues, such as roads, traffic, drinking water, drainage, electricity, waste management, and public complaints, has become increasingly relevant and effective in addressing these challenges.

2.1. Case study

2.1.1. Civic Social Complaint Android App

This app aims to modernize the process of grievance reporting by offering a digital solution leveraging the "Digital India" initiative. This application enables users to register complaints related to public infrastructure, sanitation, and safety. Key functionalities include real-time location tracking, complaint prioritization based on user engagement, and the use of Firebase's real-time database for immediate updates. By promoting transparency and fostering community engagement, the system bridges the gap between citizens and government authorities, ensuring efficient resolution of civic issues (Danish Khan, 2023).

The app introduces several innovative features, such as:

1. Real-Time Updates & Location Tracking: Utilizes GPS for precise complaint location, improving response efficiency.
2. Community Engagement: Users can like and comment on complaints, fostering collaboration for prioritization.
3. Complaint Prioritization: Higher-liked complaints are addressed first, ensuring urgent issues are prioritized.
4. Statistical Reporting: Offers detailed reports on complaint statuses (pending, in-progress, resolved) for performance monitoring and problem identification.

While the app introduces significant advancements, it lacks certain critical aspects relevant to Nepal, such as:

1. **Limited Offline Functionality:** The app relies heavily on real-time updates, which might be less practical in Nepal's rural areas with poor internet connectivity.
2. **Focus on Specific Complaint Types:** The existing solution primarily caters to general civic issues but does not extend to emergencies like road accidents or waste management challenges unique to Nepal.
3. **AI Integration & Cross-Platform:** While innovative, the app does not incorporate machine learning for automated complaint categorization, the cross-platform solution with enhanced scalability and analysis, which could significantly enhance resource allocation (Dishant Banga, 2023).

In Nepal, where civic reporting mechanisms are often slow and disconnected, a system incorporating offline support, AI-driven prioritization, and integration of various grievance categories would address these limitations effectively. The proposed system for Nepal will adapt the core features of this app while adding these capabilities to meet local needs.

2.1.2. E-Complaint Mobile Application

The "E-Complaint" application leverages crowdsourcing to address community grievances efficiently within the Philippine barangay system. It automates the traditional complaint-handling process, enabling residents to file, track, and analyze grievances related to environmental and human-to-human conflicts. With cloud-based Software-as-a-Service (SaaS) infrastructure, it provides a scalable solution accessible on major mobile platforms, bridging the gap between citizens and local authorities (Mary Jane C. Samonte, 2019).

Key features of the application include:

1. **User-Friendly Interface:** Residents can register complaints with photos and location data.
2. **Crowdsourcing for Community Engagement:** Enables collective participation through complaint categorization and prioritization.

3. **Real-Time Tracking and Notifications:** Users receive acknowledgment and updates about their complaints, enhancing transparency.
4. **Automated Analysis:** Generates data-driven insights, including statistics on complaint types, frequencies, and resolutions, to assist in future decision-making.
5. **Role-Based Access Control:** Administrators manage case details, notifications, and system updates through web and mobile interfaces.

Despite the paper's thorough analysis, several key research gaps remain. One major issue is the focus on a limited range of complaint categories, which restricts the system's applicability and effectiveness. This narrow scope may prevent the system from addressing a wider array of user concerns (Mary Jane C. Samonte, 2019).

Additionally, the absence of artificial intelligence integration is another critical gap. Without AI, the system lacks the potential for enhanced efficiency and responsiveness, which could significantly improve user experience and complaint resolution.

2.1.3. Nagarik App

The Nagarik App, launched by the Government of Nepal in 2021, aims to digitize public services and enhance citizen engagement. It allows users to access a wide range of government services and register complaints related to community infrastructure, public safety, and municipal services. It streamlines services like tax payments, vehicle and license verification, and police clearance certificates. The app promotes transparency by enabling users to track complaints and receive real-time updates (Nagarik App, 2025).

Despite its success in improving service delivery and empowering citizens, challenges such as the digital divide, low user adoption in rural areas, and concerns over data privacy persist. The app has seen significant engagement, resolved thousands of complaints and simplified public service access (Nagarik App, 2025).

Key Features

1. **Complaint Registration:** Users can register complaints related to community infrastructure, municipal services, and public safety concerns. Complaints can be tracked through the app, ensuring accountability.
2. **Access to Government Services:** The app provides access to essential services such as vehicle registration, tax payments, police clearance certificates, and health insurance information.
3. **User-Friendly Interface:** Designed for ease of use, the app allows users to navigate through its features seamlessly. It requires basic information for registration, such as a valid mobile number and identification documents.
4. **Real-Time Notifications:** Users receive updates on the status of their complaints and service requests, fostering better communication between citizens and government officials.

Since its launch, the Nagarik App has seen significant user engagement, with thousands of complaints registered and resolved. The app has improved transparency in public service delivery and empowered citizens to take an active role in addressing community issues. Feedback from users indicates a positive reception, particularly regarding the convenience of accessing government services.

Chapter 3: METHODOLOGY

The methodology for the Smart Community Complaint App provides a structured framework for its design, development, and deployment. This approach ensures efficient execution and successful implementation, with each stage involving specific tasks contributing to achieving project objectives.

The methodology consists of four key stages, each enhanced with AI to improve functionality and user experience. In the first stage, Project Flow, AI tools optimize workflows, resource allocation, and risk management, streamlining the planning process. Predictive analytics can be used to anticipate potential delays or resource constraints. The second stage, System Design, focuses on defining the app's architecture and interface, with AI enhancing usability through personalized recommendations, adaptive interfaces, and automated feedback loops.

The third stage identifies Software, Hardware, and AI Requirements to ensure strong implementation. AI tools like PyTorch, combined with cloud platforms such as Google Colab, enable advanced features like real-time image recognition for complaint verification, sentiment analysis of user feedback, and automatic prioritization of complaints based on severity and location. The final stage, Testing and Maintenance, employs AI-driven testing to simulate diverse user interactions, real-time monitoring to detect anomalies, and predictive maintenance to address issues proactively.

3.1. Smart Community Complaint App

Smart Community Complaint App is a mobile application that reads and processes images of community-based issues as input and categorizes them into classes. The figure below shows an overview of the SCCA system.

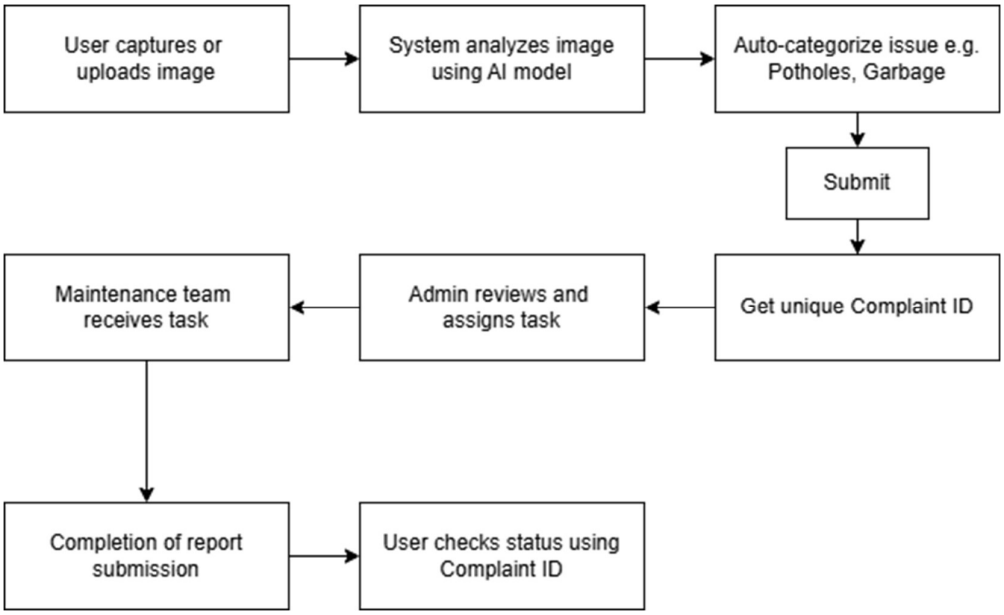


Figure 3.1: Generalized Workflow of System (Group Study, 2025)

3.1.1. Dataset Collection

An image dataset was compiled to train and evaluate an object detection model for urban and environmental monitoring. The data was aggregated from two primary sources: the Open Images Dataset and Kaggle.

A subset of six classes was selected for their relevance to the application domain: “Animal,” “Traffic light,” “Waste container,” “Potholes,” “Cracks,” and “Open Manhole.” The final collection consists of approximately 500 images per class, which were acquired using Bukkster's OIDv6_ToolKit. All images include label files with bounding box annotations in the standard OID format.

3.1.2. Dataset Splitting

In this project, the dataset was manually collected and prepared for training a YOLOv5 object detection model focused on solving real-world urban problems

such as potholes, cracks, open manholes, animals, traffic lights, and waste containers.

The data was processed separately for each category and split into training and validation sets using an 80:20 ratio. A dedicated test set was not created; instead, the model's performance was evaluated using validation metrics and visual inspection of predictions during training.

For instance, the dataset consisted of approximately 270 images per class for potholes, cracks, and open manholes, resulting in a total of around 823 images.

These were divided as follows:

Training Set: ~658 images (80%)

Validation Set: ~165 images (20%)

Similarly, for the other classes such as animals, traffic lights, and waste containers, approximately 600 images per class were collected, summing to around 1800 images.

Training Set: ~1440 images (80%)

Validation Set: ~360 images (20%)

This dataset split allowed for efficient training of the YOLOv5 model while ensuring proper validation and minimizing the risk of overfitting.

3.1.3. Preprocessing and Augmentation

In this phase, several preprocessing tasks were performed to ensure consistency and improve model robustness. All images were resized to 640×640 pixels, as required by the YOLOv5 model input format. Additionally, automatic orientation adjustment (Auto-Orient) was applied where necessary to maintain a consistent image alignment.

To enhance generalization and prevent overfitting, data augmentation was applied during training. This was handled internally by the YOLOv5 framework and included:

a. Random horizontal flips
b. Scaling and zooming
c. Color adjustments (hue, saturation, brightness)
d. Image rotation and shearing

These augmentations were applied dynamically during training, meaning the model was exposed to a wider variety of scenarios without physically increasing the dataset size. This helped the model learn to detect objects more accurately in different lighting, angle, and size conditions.

3.1.4. Training Setup and Configuration

For model training, high computational resources were required. Instead of setting up a local environment, **Google Colab** was used, which provides free access to **NVIDIA GPUs** with **CUDA** and **cuDNN** pre-installed. This eliminated the need for manual driver installation and environment configuration. The training was performed in a GPU runtime, ensuring faster processing and efficient model convergence.

3.1.5. Model Training

The YOLOv5 object detection model, pre-trained on the COCO dataset, was utilized as the base model for training. The model was fine-tuned using custom dataset consisting of six classes: Pothole, Cracks, Open Manholes, Animal, Traffic Light, and Waste Container. The training process was carried out on Google Colab with GPU support, using the following configuration:

- **Image size:** 640×640
- **Batch size:** 16
- **Epochs:** 100
- **Model weights:** yolov5s.pt (pre-trained)
- **Training-validation split:** 80-20
- **Loss and performance curves:** monitored during training

Yolo Architecture

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.

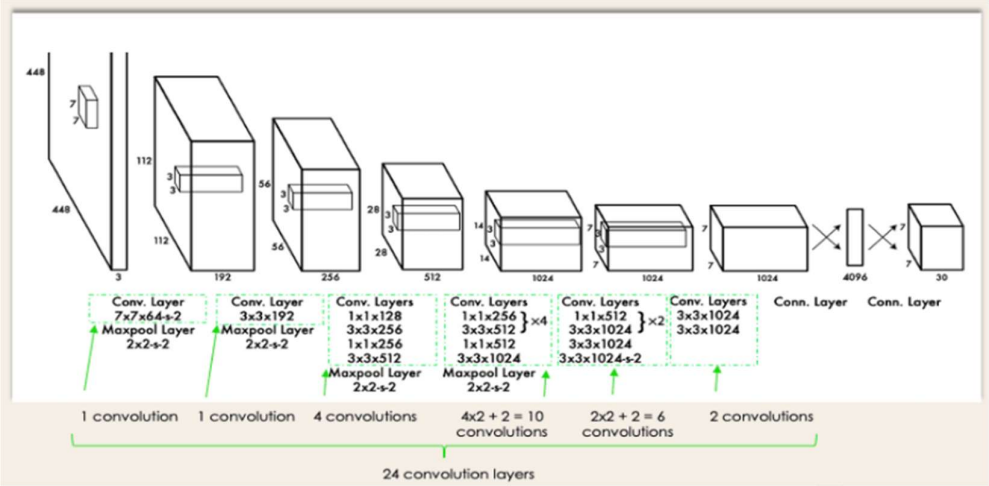


Figure 3.2: Architecture of YOLO Algorithm (Kundu, 2023)

The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates (Kundu, 2023).

YOLO divides an input image into an S × S grid. If an object's center falls within a grid cell, that cell is responsible for detecting the object. Each grid cell predicts 'B' bounding boxes and confidence scores, reflecting the model's confidence in the box containing an object and its accuracy. The bounding box attributes are predicted using a single regression module, forming a vector Y = [pc, bx, by, bh, bw, c1, c2],

where pc is the probability of an object being in the grid, bx and by are the center coordinates, bh and bw are the height and width, and c1, c2 are class probabilities.

To address multiple bounding boxes for a single object, YOLO uses the Intersection over Union (IoU) metric and Non-Max Suppression (NMS) to retain only the most relevant boxes, reducing noise (Subramanyam, 2021) (Everingham, 2010).

B1 and B2 are two bounding boxes and the formula of IoU is:

$$IoU = \frac{Area\ of\ the\ intersection\ between\ B1\ and\ B2}{Area\ of\ the\ union\ between\ B1\ and\ B2}$$

Figure 3.3: Formula equation of IoU

About YOLOv5

Released in June 2020 by Ultralytics, YOLOv5 is one of the most widely used and flexible versions of the YOLO (You Only Look Once) series. It supports key vision tasks like object detection and classification with excellent speed and accuracy. YOLOv5 is implemented entirely in PyTorch, making it easy to train and deploy across platforms. It uses a CSPDarknet53 backbone, PANet neck, and a YOLO head for efficient multi-scale detection. YOLOv5 offers multiple model sizes (n, s, m, l, x) to balance between speed and accuracy. It also supports auto-learning of anchor boxes, mosaic augmentation, and label smoothing. Evaluated on the MS COCO dataset, YOLOv5x achieves 50.1% AP at 640 pixels and runs at over 140 FPS on an NVIDIA V100. The figure below illustrates the YOLOv5 architecture in detail.

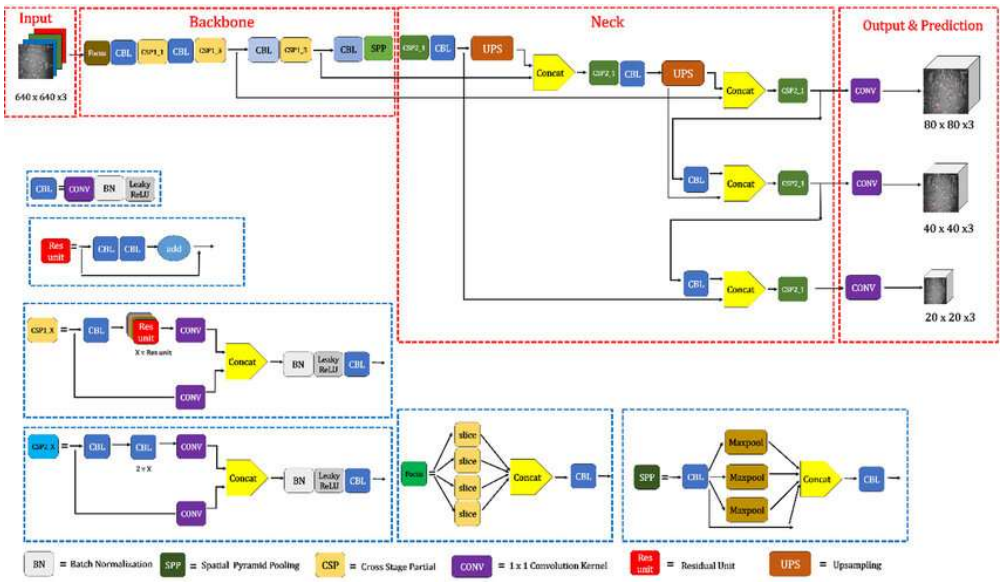


Figure 3.4: YOLOv5 Architecture

3.1.6. Model Evaluation

After training the YOLOv5 model on the road hazard detection dataset, various evaluation techniques were applied to measure its performance. Training and validation metrics were used to monitor the model's learning behavior, such as loss values and accuracy over epochs. The validation phase helped confirm the model's ability to generalize to unseen data.

Key performance indicators included:

- **Precision**: The accuracy of positive predictions (e.g., how many predicted potholes were actual potholes).
- **Recall**: The model's ability to detect all relevant objects (e.g., how many actual cracks the model was able to detect).
- **mAP@50 and mAP@50-95**: Standard object detection metrics showing the average precision across IoU thresholds.

Additional visual tools used:

- **Confusion Matrix**: Helped visualize misclassifications between the six classes.
- **PR Curves**: Showed the trade-off between precision and recall.
- **Labeled Image Predictions**: Output images with bounding boxes were analyzed to verify the model's accuracy visually.

Model Performance Analysis:

In this project, a multi-model approach was adopted to effectively address the detection of varied object categories. Specifically, Model 1, which focuses on detecting Potholes, Cracks, and Open Manholes, while Model 2 targets the detection of Animals, Traffic Lights, and Waste Containers. The performance of these models was analyzed over 100 epochs using training and validation metrics. These metrics provide crucial insights into the model's learning progression and its ability to generalize across unseen images.

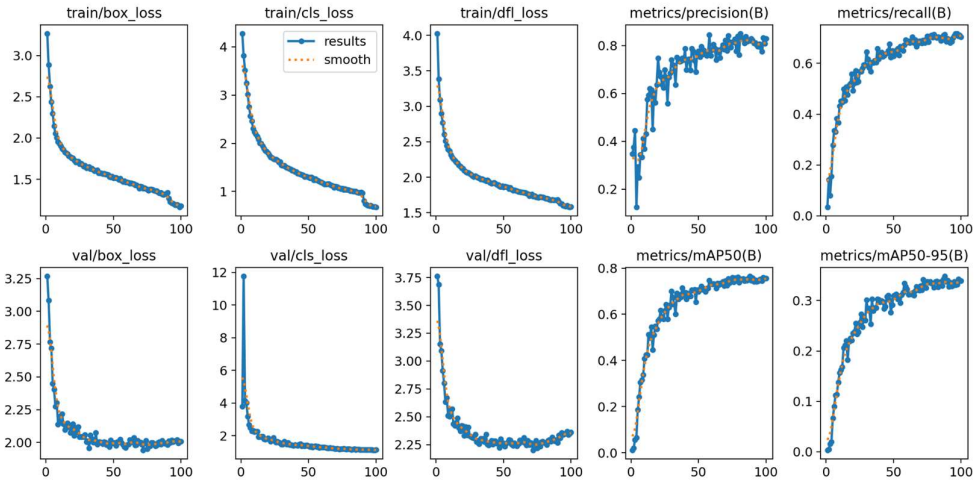


Figure 3.5: Model 1 Performance Analysis

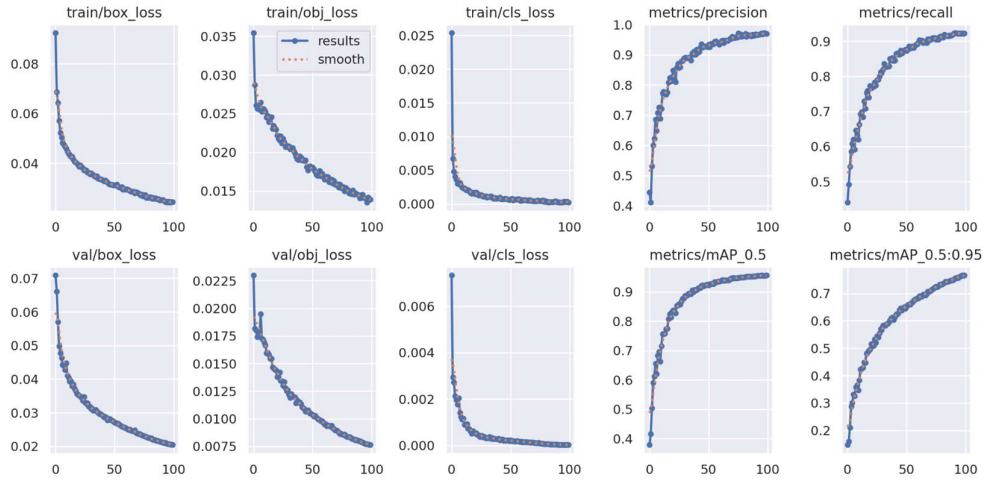


Figure 3.6: Model 2 Performance Analysis

Analysis of Model 1

Model 1 demonstrated a clear, but ultimately limited, learning trajectory. During initial training, the model showed rapid improvement, with all loss metrics dropping sharply. However, progress soon slowed, and a noticeable gap emerged between the training and validation losses. This divergence, where validation losses flattened at a higher level than training losses, points to a degree of overfitting or challenges within the validation dataset.

This behavior is reflected in the final performance metrics. At its final epoch, the model achieved a respectable precision of approximately 0.8 and a recall of 0.7, resulting in an mAP@0.5 of around 0.75. However, its performance on the stricter mAP@0.5:0.95 metric was significantly lower at 0.3. Notably, the majority of these gains were realized within the first 30 to 50 epochs, after which performance plateaued.

In summary, while Model 1 converged, its performance is moderate. Further improvements would likely require addressing the overfitting, perhaps through higher-quality data, stronger augmentations, or architectural adjustments. Future

runs could benefit from implementing early stopping based on the validation mAP to prevent training past the point of diminishing returns.

Analysis of Model 2

Model 2 exhibited a much stronger and more stable training process. All loss components, box, objectness, and classification declined smoothly and steadily throughout training. Crucially, the training and validation losses tracked each other very closely, indicating excellent generalization with minimal signs of overfitting. An interesting observation was that the classification loss approached zero very early, suggesting the task of identifying object classes was relatively straightforward for the model.

This robust training translated directly into exceptional performance metrics. By the final epoch, the model achieved a high precision of 0.96 and a recall of 0.92, culminating in an impressive mAP@0.5 of 0.95 and a strong mAP@0.5:0.95 of 0.75. The metrics improved steadily alongside the losses, reinforcing the conclusion that the model was learning effectively without just memorizing the training data.

The key takeaway is that Model 2 demonstrates strong convergence and robust generalization. With the classification task largely saturated, any further optimization could focus on improving fine-grained localization to boost the mAP@0.5:0.95 score. Given its stability, implementing an early stopping mechanism would be highly effective for optimizing computational resources without sacrificing performance. A final validation on a completely held-out test set is recommended to confirm its real-world strength.

Confusion Matrix:

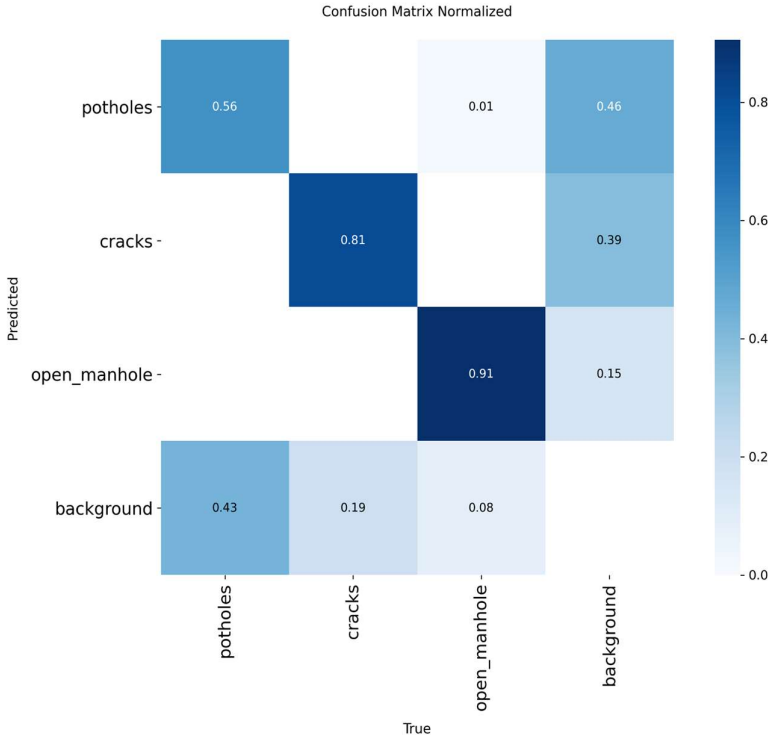


Figure 3.7: Confusion matrix (Model 1)

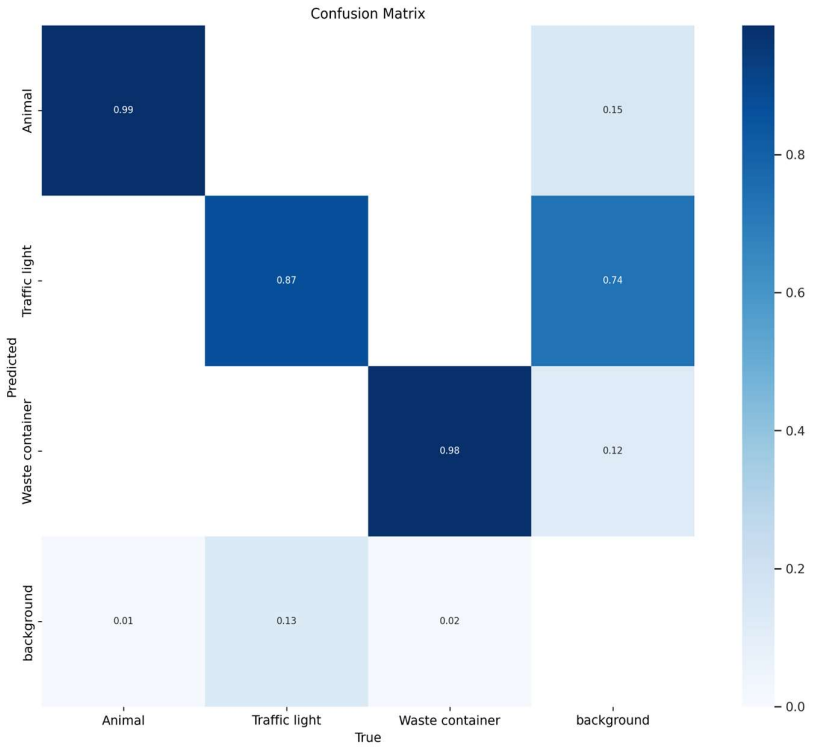


Figure 3.8: Confusion matrix (Model 2)

Confusion Metrics:

Metric	**Model 1**	**Model 2**
Overall Accuracy	Moderate (esp. for potholes)	High (near perfect for all)
Background Noise	High confusion with the background	Minor confusion with the background
Best Class	Open Manhole (91%)	Animal (99%), Waste Container (98%)
Weakest Class	Potholes (56% accurate)	Background (but not severe)

Labelled Diagram:

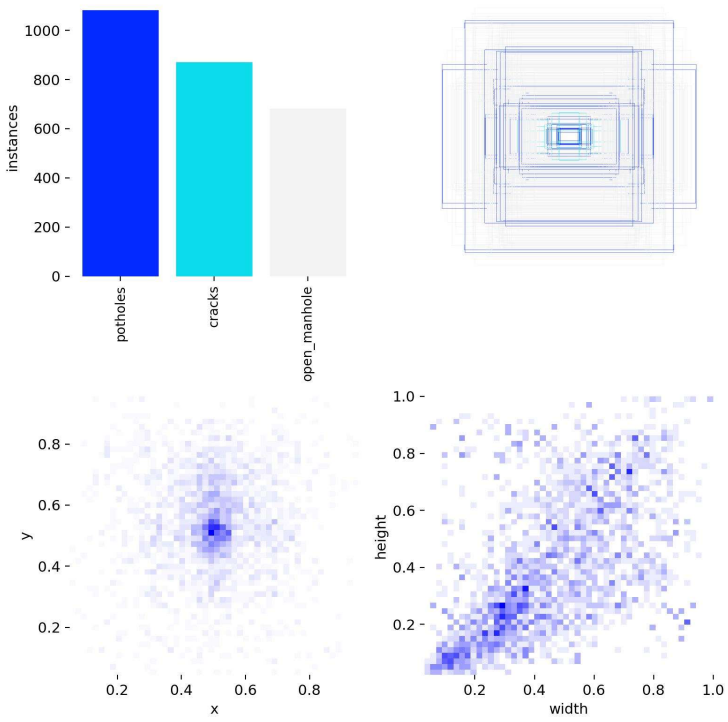


Figure 3.9: Labelled Diagram of Model 1

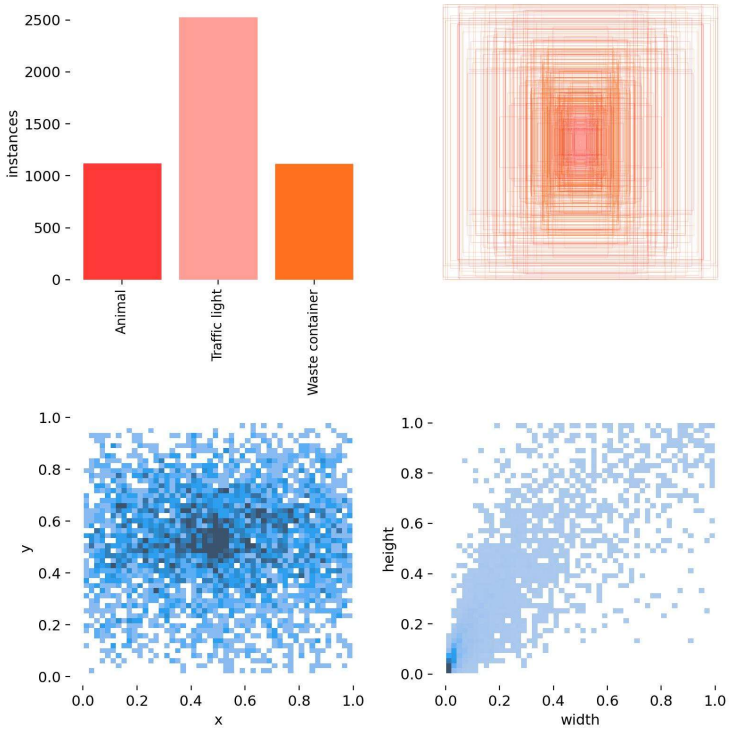


Figure 3.10: Labelled Diagram of Model 2

Model 1: Potholes, Cracks, Open Manholes

Instances Distribution (Top-left)

a. Potholes: ~**1050 instances** – most frequent class.
b. Cracks: ~**850 instances**.
c. Open Manholes: ~**680 instances** – lowest in this model.
   The dataset is relatively balanced, but with a moderate class imbalance.

Bounding Boxes (Top-right)

a. Bounding boxes are centralized and fairly consistent in size.
b. Most annotations appear concentrated in the center, indicating limited variation in object position.

Object Centers (Bottom-left)

a. Densely clustered around the center **(x≈0.5, y≈0.5).**
b. Suggests most objects in images are located near the image center.

Object Size Distribution (Bottom-right)

a. Width and height show a spread but are biased toward smaller to medium-sized objects.
b. Consistent sizing helps with more stable model training.

Model 2: Animals, Traffic Lights, Waste Containers

Instances Distribution (Top-left)

a. Traffic Lights: **~2600 instances** – dominant class.

b. Animal & Waste Container**: ~1100 - 1200** instances each. There is a visible class imbalance, which could affect model performance unless addressed through weighting or data augmentation.

Bounding Boxes (Top-right)

a. Dense overlay of bounding boxes, mostly centered, but with a wider variety in box dimensions.
b. More visual complexity than **Model 1**.

Object Centers (Bottom-left)

a. More evenly spread across the image, unlike **Model 1**.
b. This shows that the detection model had to learn to identify objects in various positions.

Object Size Distribution (Bottom-right)

a. Objects vary more in size, with a wider range from small to large dimensions.
b. This diversity in object scale helps build a more generalized model.

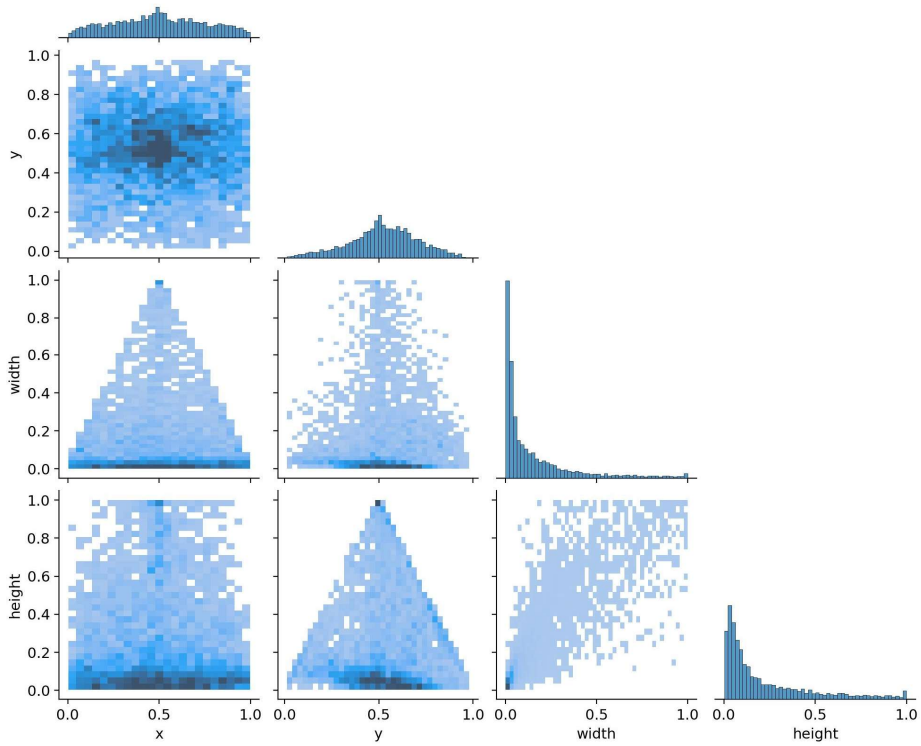


Figure 3.11: Pair plot

This image is a pair plot, which is often used to visualize the relationships between multiple variables (x, y, height, width).

Post-Processing:

In this project, post-processing was applied to refine the raw predictions produced by the YOLOv5 object detection model. After each inference, non-maximum suppression (NMS) was automatically applied by the YOLO framework to eliminate redundant and overlapping bounding boxes, retaining only the most confident predictions. Additionally, confidence thresholds were set to filter out weak detections. This helped reduce false positives and ensured more stable and accurate detection results during real-time implementation and validation.

Optimization and Hyperparameter Tuning:

During training, several hyperparameters were configured and adjusted to improve the model's learning efficiency and detection accuracy. Key parameters such as the batch size (16), image size (640), and number of epochs (100) were selected based on experimentation and computational feasibility on Google Colab's GPU. The optimizer used (SGD or AdamW as default by YOLOv5), along with learning rate scheduling, allowed the model to converge effectively. These tuning efforts resulted in high precision and recall scores, especially in Model 2, where the mAP50 reached 0.994 for the “Animal” class.

Object Detection:

The core functionality of this system was centered on object detection using YOLOv5. The model was trained to detect six distinct object classes relevant to community infrastructure and safety: potholes, cracks, open manholes, animals, traffic lights, and waste containers. Using labeled datasets and YOLOv5’s efficient one-stage detection architecture, the system could identify objects in real-time with high accuracy. This detection capability forms the backbone of the Smart Complaint Management System, allowing for automated issue identification from captured or uploaded images.

Learning Rate:

The learning rate is a crucial hyperparameter in training deep learning models, as it determines how quickly a model updates its weights during optimization. A well-tuned learning rate leads to faster convergence and better performance, while an incorrect value may cause the model to underfit or overfit. In this project, the default learning rate provided by YOLOv5 is “0.01”. The default training rate was utilized without explicit modification, as it is typically appropriate for the majority of small to medium-sized datasets.

3.2. Development and Planning

3.2.1. Concept and Initiation

Before starting any project, it's important to collect and check if the requirements make sense. The project can continue if the requirements are feasible. During this phase, everyone involved in the project, like citizens and government officials, works together to gather all the necessary information needed to create and build the project. It's like putting together all the puzzle pieces to make sure we have everything we need. This way, the developers and designers can understand what needs to be done and create a system that solves road-related problems.

3.2.2. Definition and planning

In this Smart Community Complaint App project, the aim is to develop a mobile application that streamlines the reporting and management of community issues. Key components of the system will include features for user complaint submission, services for addressing these complaints, efficient administration and maintenance of the application, and comprehensive tracking and resolution of reported issues across various categories.

3.2.3. Technology Used

1. **Flutter & Dart**: Flutter, powered by the Dart programming language, is an open-source UI toolkit by Google for building cross platform applications. It allows developers to write once and deploy across Android, iOS, desktop, and web. Key features include Hot Reload for rapid UI updates, customizable widgets for expressive designs, and high performance on both platforms. Dart’s strong typing and object-oriented structure make it ideal for scalable app development. In this project, Flutter was used to build the frontend of the mobile application for real-time detection of street level issues like potholes and waste containers (Google, Flutter, 2025) (Team D. , 2025).

2. **Visual Studio Code**: VS Code was used as the primary development environment for Python scripting. Its features like syntax highlighting, integrated terminal, and Git support made it easy to handle both machine learning scripts and dataset conversions efficiently (Microsoft, 2025).

3. **Android Studio**: Although Flutter handles most of the cross-platform development, Android Studio was used for device emulation, native debugging, and performance profiling during mobile app testing (Google, Android Studio, 2025).

4. **Git & GitHub**

   Git, a decentralized version control system, was used to manage source code versions. It helped track changes, collaborate across devices, and maintain backups of training scripts and datasets. GitHub hosted the remote repository and enabled sharing across platforms (Team G. , 2025).

5. **Python & PyTorch**

   Python served as the backbone for training the custom object detection model using PyTorch, a powerful deep learning framework. YOLOv5, which is based on PyTorch, was used to train on multiple classes such as *Animal*, *Traffic Light*, *Waste Container*, *Potholes*, *Cracks*, and *Open Manhole*. Google Colab was utilized to leverage free GPU acceleration during training. Python scripts were also written to preprocess datasets, convert labels to YOLO format, and manage data augmentation (Foundation, 2025).

6. **AI/ML Integration**

   The Smart Community Complaint App incorporates AI/ML to enhance efficiency and accuracy in handling complaints. Machine Learning models will primarily be used for:

   a. **Image Recognition**: Identifying the type and severity of the issue (e.g., potholes, drainage blockages) from uploaded images.

b. **Complaint Categorization**: Automatically classifying reports into predefined categories based on keywords and image analysis.
c. **Priority Determination**: Leveraging algorithms to prioritize complaints based on severity, location density, and community impact.

7. **Google Colab**

   Google Colab was used as the cloud-based development environment for training the object detection model. It provided access to free GPU acceleration, enabling faster model training using PyTorch and YOLOv5.

8. **Open Images Dataset**

   The dataset was collected using the Open Images Dataset V6 Toolkit. Classes such as *Animal*, *Traffic Light*, and *Waste Container* were downloaded and preprocessed into the YOLO format, serving as the foundation for training the detection model (Google Apis, 2012).

9. **Kaggle**

   Kaggle is a popular platform for sharing datasets and building machine learning models. For this project, YOLO-annotated road hazards such as *Potholes, Cracks, Open Manholes* dataset from Kaggle was used, which was ready for training and easy to integrate.

3.2.4. Technological Approach

1. **PyTorch**

   The main deep learning framework used to build and train the YOLOv5 model. Provides a dynamic computational graph and GPU acceleration.

2. **YOLOv5 (by Ultralytics)**

   Object detection architecture used for training. Built on top of PyTorch. Provides utilities for training, validation, testing, and inference.

3. **OpenCV (cv2)**

   Used for image preprocessing, visualizations, and bounding box rendering. Handy for real-time video or webcam-based applications.

4. **Pillow (PIL)**

   Python Imaging Library used for loading, resizing, and processing images.

3.2.5. Launch and Execution

Once the development of the Smart Community Complaint App is complete, it will be launched for real-world usage. Users will be able to access the system, submit complaints, and interact with its features. Compatibility will be ensured, testing conducted, and valuable user feedback gathered to make necessary improvements.

3.2.6. Performance and Control

The system's speed, stability, and scalability will be continuously monitored. Regular performance tests will be conducted to ensure it meets project goals. Measures for data backup and security will also be implemented to maintain control and ensure a seamless user experience.

3.3. System Design

After gathering and analyzing the requirements, the Smart Community Complaint App was given a solid structure. During this phase, the project's architecture was designed based on the collected requirements. Creative and intuitive diagrams such as system flow, ER diagrams, system architecture, and use cases were created.
Those visual representations helped in visualizing the system's structure, data flow, and interactions. By carefully designing the architecture, the foundation was set for the successful development and implementation of Smart Community Complaint App.

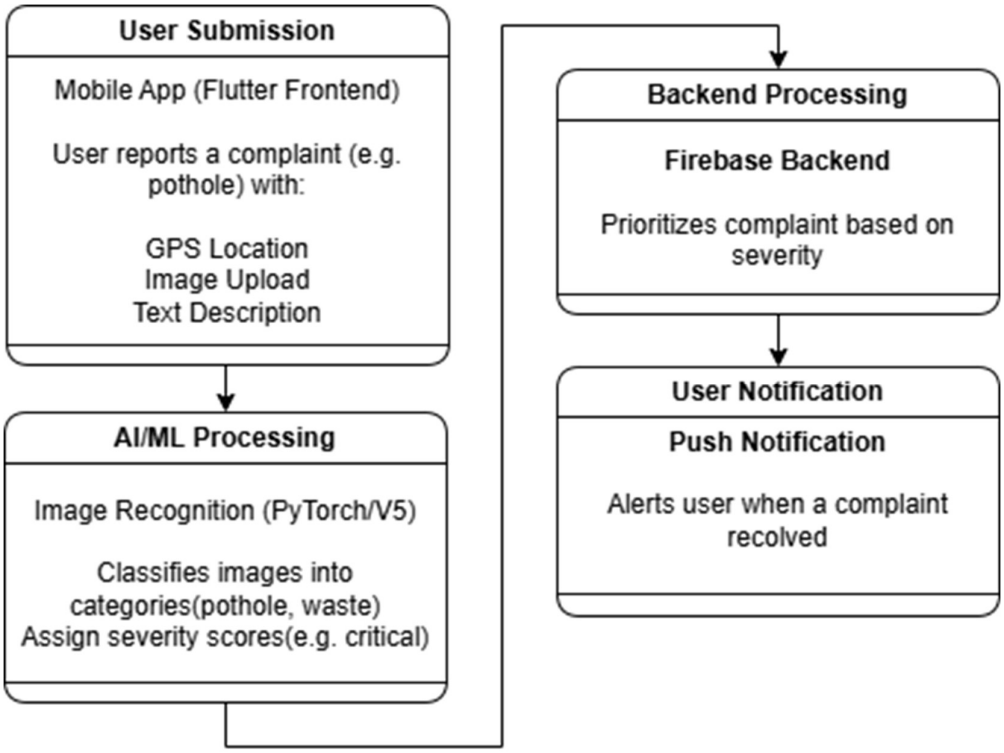


Figure 3.12: System Design of SCCA

3.3.1. System Flow Diagram

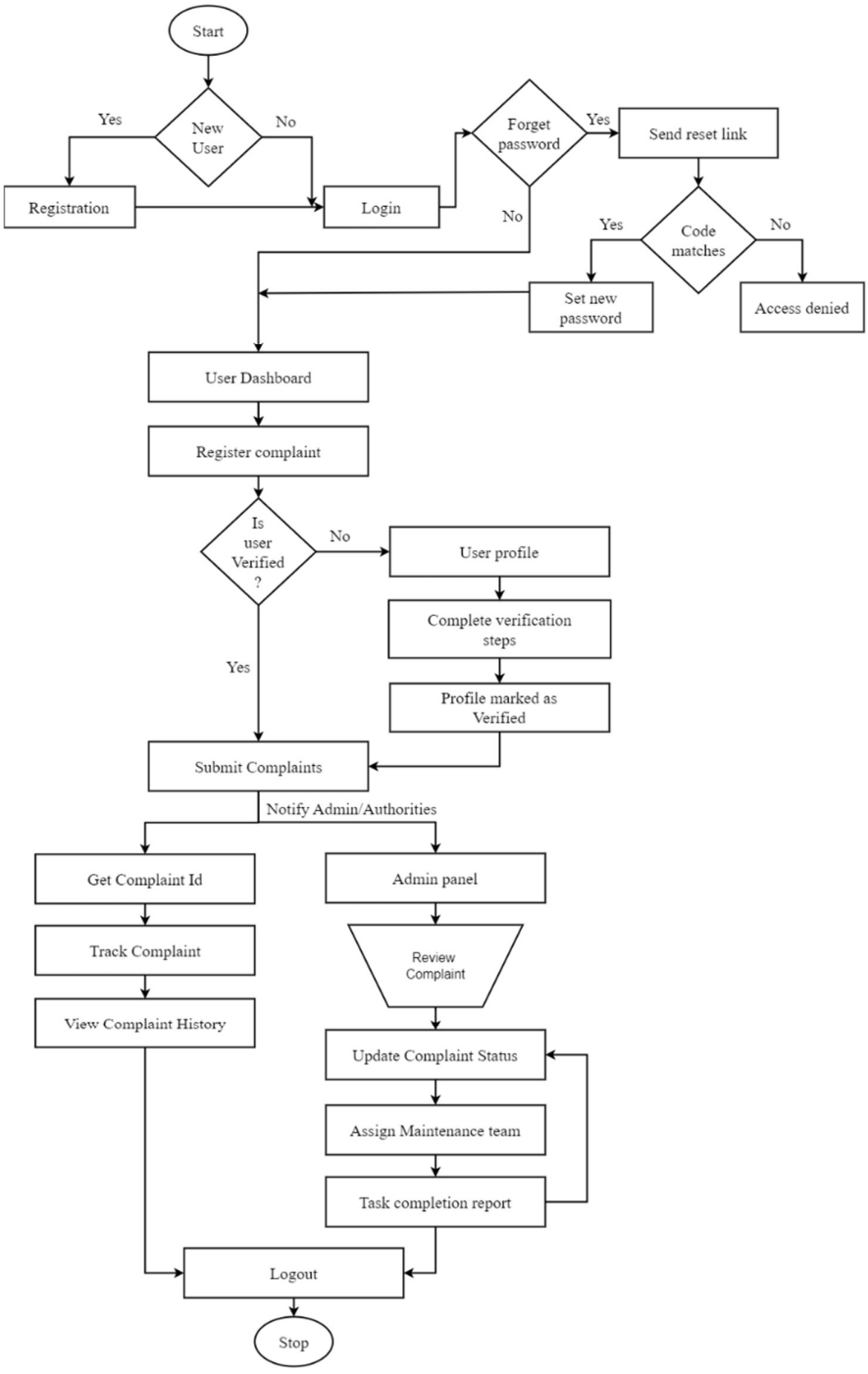


Figure 3.13 System Flow Diagram

(Group Study, 2025)

3.3.2. E-R diagram

E-R Diagram stands for Entity Relationship Diagram, also known as ERD, which is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes, and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes, and diamond shapes to represent relationships.

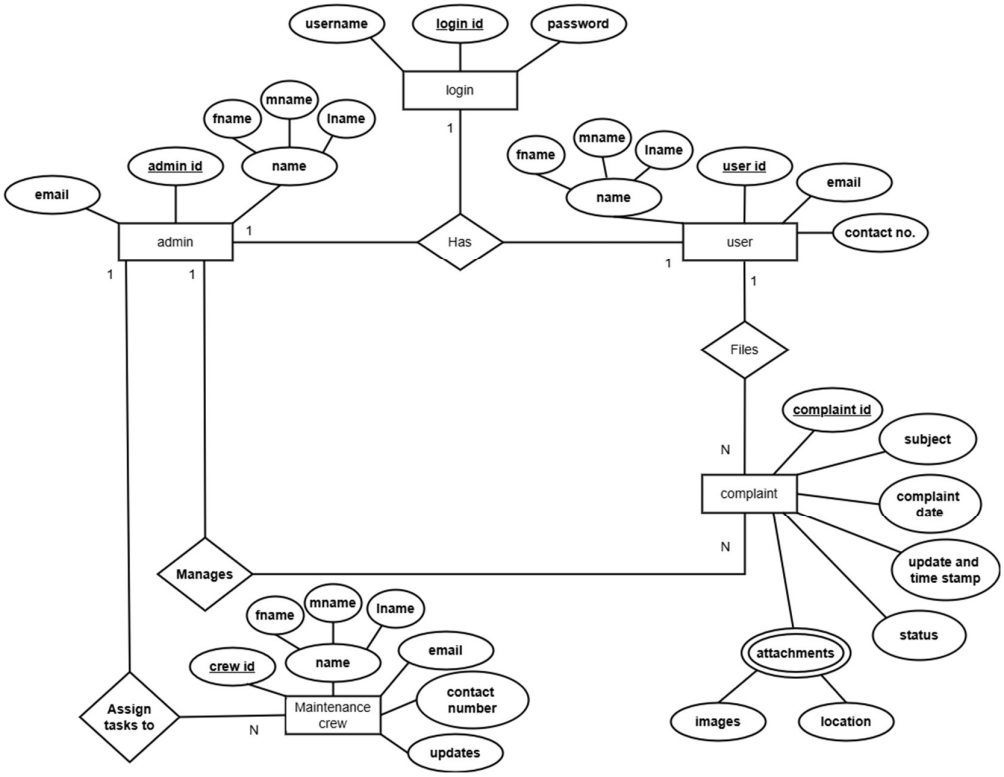


Figure 3.14: ER Diagram

(Group Study, 2025)

3.3.3. Use case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well.

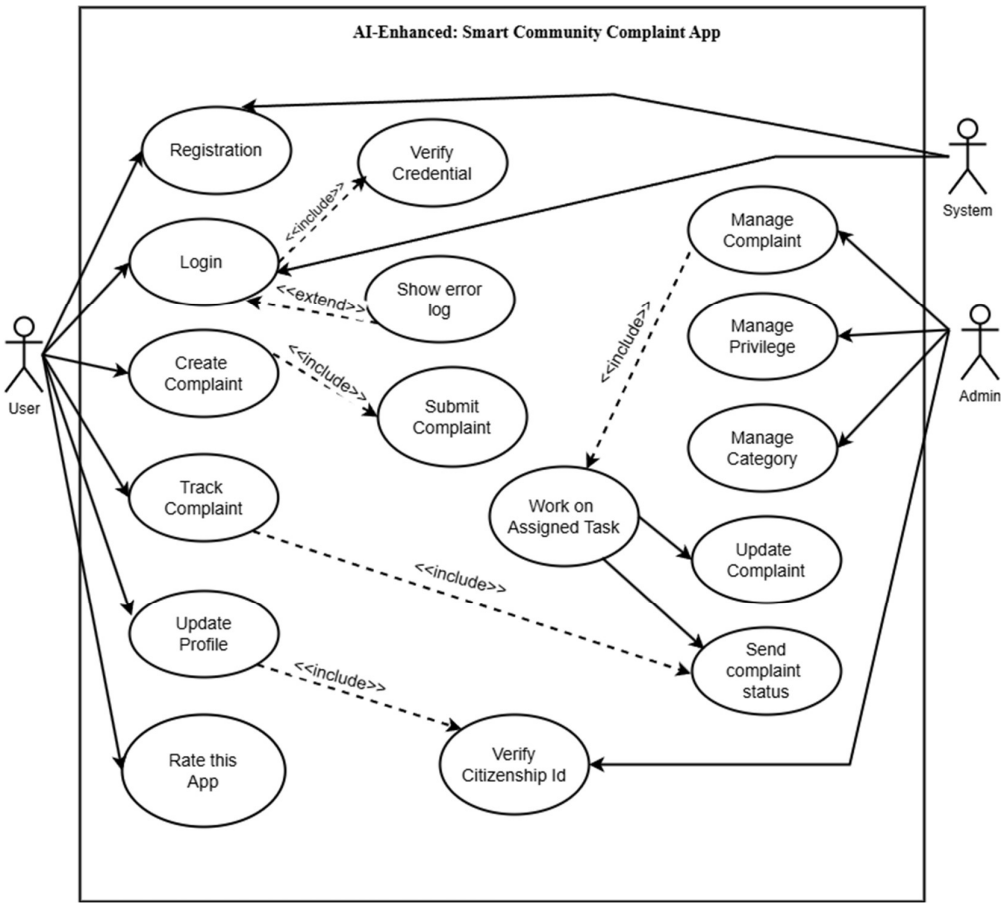


Figure 3.15 Use case Diagram

(Group Study, 2025)

3.3.4. Software Development Model

The Agile model is a flexible and iterative approach to software development that emphasizes collaboration, customer feedback, and small, incremental releases. Instead of planning the entire project upfront, Agile breaks down the work into smaller, manageable chunks called "sprints," typically lasting 1-4 weeks. Each

sprint results in a potentially shippable product increment, allowing teams to adapt to changing requirements and deliver value to customers continuously. Agile encourages close collaboration between developer teams and stakeholders, ensuring that the final product closely aligns with user needs and business (Team G. , 2024).

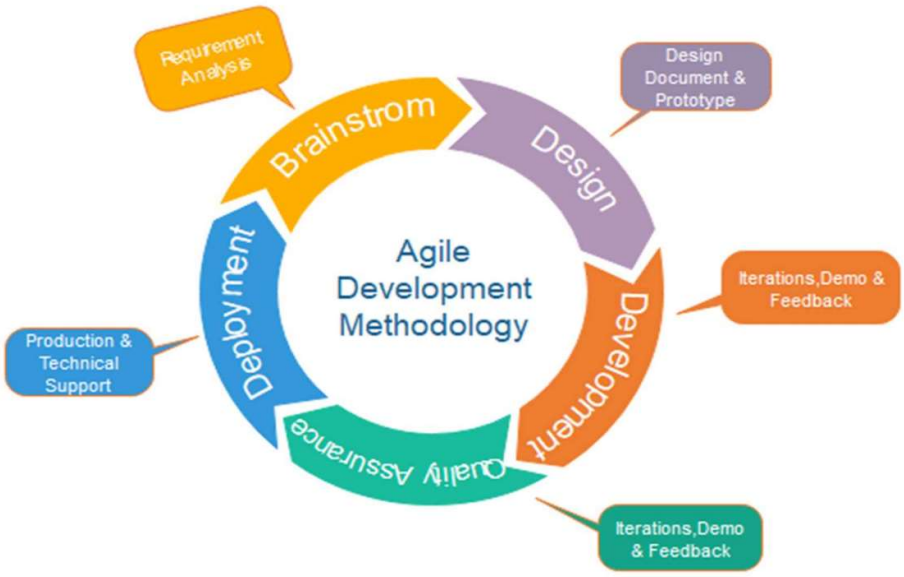


Figure 3.16: Agile Development Model

(Group Study 2025)

3.4. Software and hardware requirements

For this Smart Community Complaint App project, the hardware and software requirements are as follows:

Hardware Requirements:

1. **Mobile Devices:** Any smartphone or tablet running the supported operating system.
    - **Android:**

      Flutter often supports Android versions 4.1 (API level 16) and above.

- **iOS:**

  For iOS, Flutter typically supports iOS versions 9.0 and above.

Software Requirements:

1. **Operating System:**

   Compatible with various operating systems, including Android and iOS.
2. **Internet Connectivity**:

   Access to a stable internet connection for submitting complaints and receiving real-time updates.
3. **Camera Access:**

   Permission for the app to access the device's camera if users want to include photos when reporting issues.
4. **Location Services:**

   Permission for the app to access the device's location services for precise geolocation when reporting problems.

By ensuring compatibility with a wide range of devices, the Smart Community Complaint App can be accessed and used seamlessly by users on their smartphones.

3.5. Testing

Before launching a project, thorough testing is essential to ensure functionality across various devices like smartphones and tablets. AI/ML enhances this phase with AI-driven testing, simulating diverse user interactions to identify and resolve issues early. Thorough testing and performance evaluation were conducted to ensure the accuracy, reliability, and generalization ability of the object detection model developed for identifying community-related issues such as animals on the road, traffic lights, and waste containers.

3.5.1. Validation Process

During training, a portion of the dataset was reserved for validation. This validation set was used to assess how well the model generalizes to unseen data. The validation was conducted within the YOLOv5 training pipeline on Google Colab, which provided insights into the model's predictions on images it had not seen during training.

3.5.2. Evaluation Metrics

The model's performance was assessed using several key object detection metrics:

- **Precision:** The proportion of correctly identified objects out of all detections.
- **Recall:** The proportion of actual objects that were successfully detected.
- **mAP@0.5 (Mean Average Precision at IoU threshold 0.5):** A standard metric for evaluating object detection performance, indicating how well the bounding boxes and class predictions align with ground truth.
- **mAP@0.5:0.95:** A more rigorous version of mAP that averages over multiple IoU thresholds, providing a more comprehensive evaluation.

Chapter 4: Results and Discussion

4.1. Overview

This chapter presents the outcomes of training and evaluating the YOLOv5-based object detection model on a custom road hazard dataset comprising potholes, cracks, and open manholes, animal, traffic lights, waste-container. The goal was to accurately detect and classify these hazards in real-world road images using state-of-the-art deep learning techniques.

4.1.1. Model Performance

The object detection model was trained using YOLOv5, a deep learning architecture known for real-time performance and high accuracy. The model was evaluated using standard object detection metrics, including **Precision**, **Recall**, **mAP@0.5**, and **mAP@0.5:0.95**. The project involved the training and evaluation of two separate YOLOv5 models. Each model was designed to detect a specific set of classes relevant to smart city monitoring. The results discussed below reflect the individual performance of each model on its corresponding classes.
The final results from validation on unseen data are summarized below:

Class	**Precision**	**Recall**	**mAP50**	**mAP50-95**
Potholes	0.835	0.504	0.613	0.273
Cracks	0.775	0.751	0.786	0.310
Open Manholes	0.856	0.886	0.897	0.463

Overall (Model 1)	**0.822**	**0.713**	**0.766**	**0.349**
Animals	0.994	0.982	0.994	0.842
Traffic Lights	0.951	0.810	0.881	0.606
Waste Containers	0.986	0.982	0.993	0.850
Overall (Model 2)	**0.971**	**0.925**	**0.956**	**0.766**

Training and Evaluation Visualizations

During training, several plots were generated to visualize the model's learning behavior and evaluation metrics. These visual results not only confirm the model's quantitative performance but also reveal patterns, strengths, and potential areas for improvement

Training and Validation Batches

What it shows: Example batches from training and validation datasets with bounding boxes.

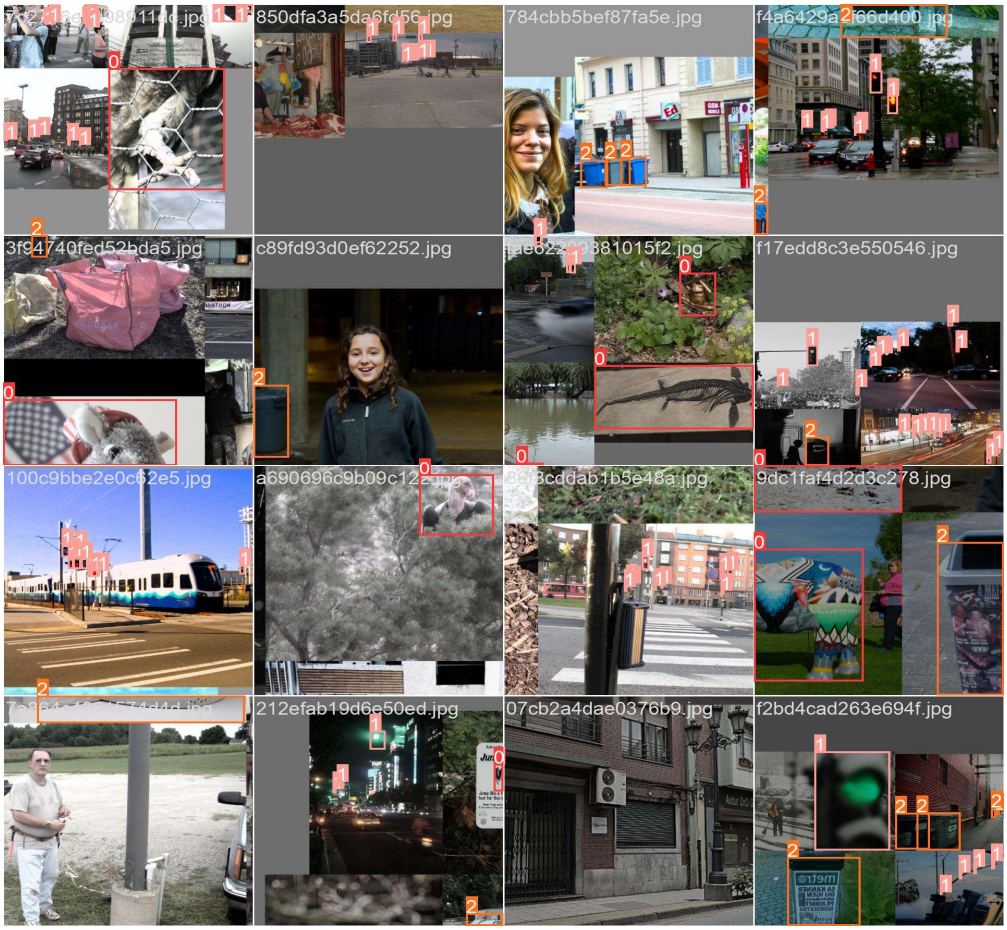


Figure 4.1: Training Batches

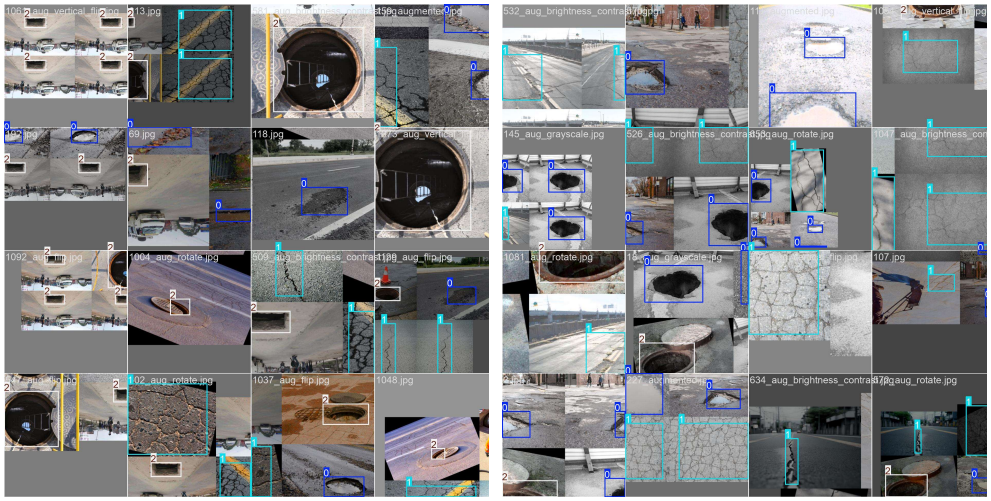


Figure 4.2: Training Batches

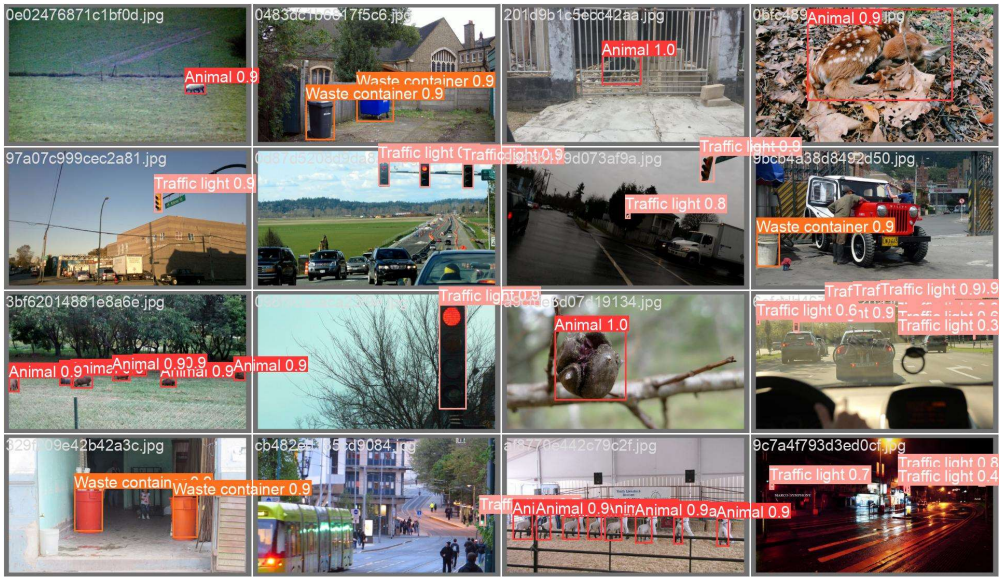


Figure 4.3: Validation Batches

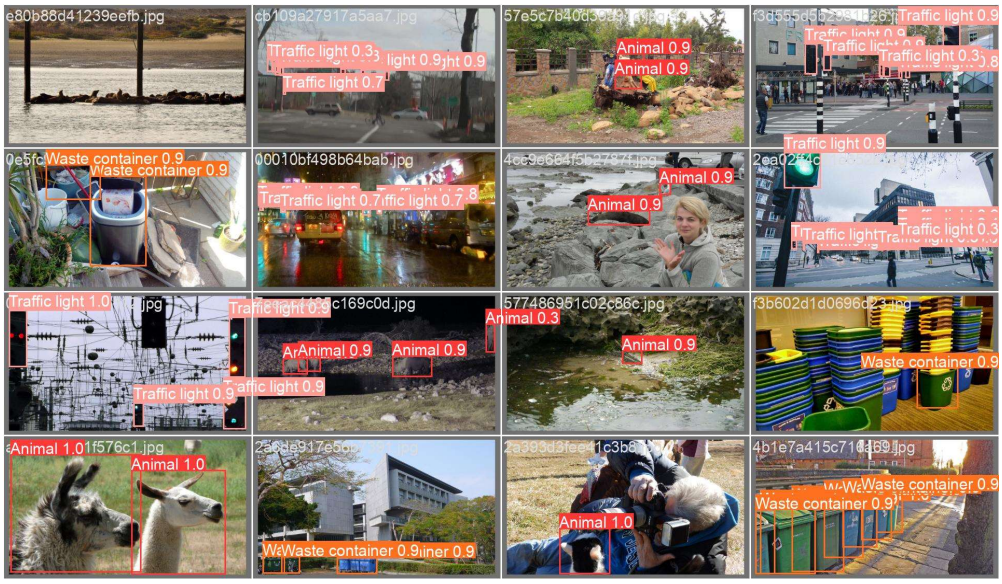


Figure 4.4: Validation Batches

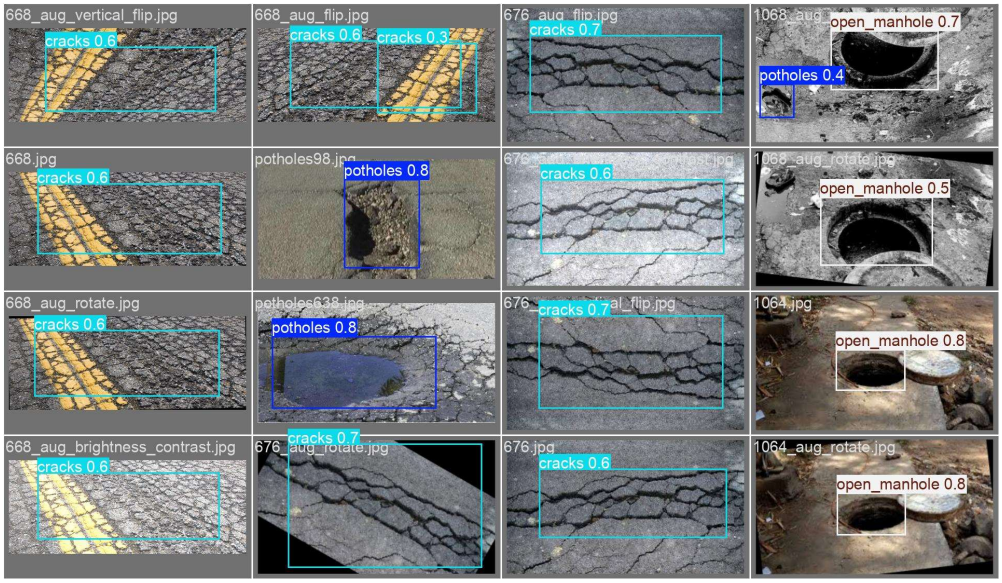


Figure 4.5: Validation Batches

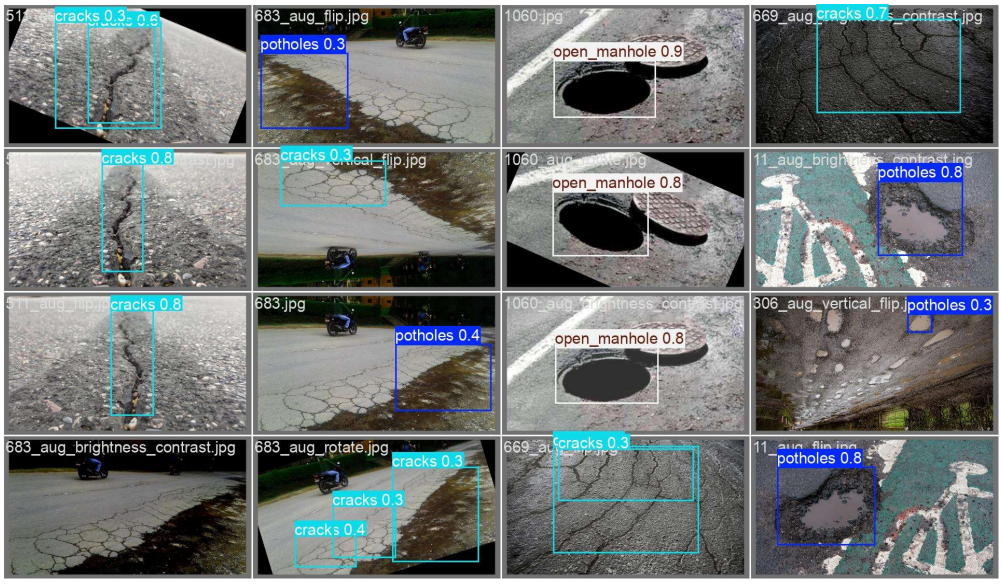


Figure 4.6: Validation Batches

4.2. Application Interface and User Experience

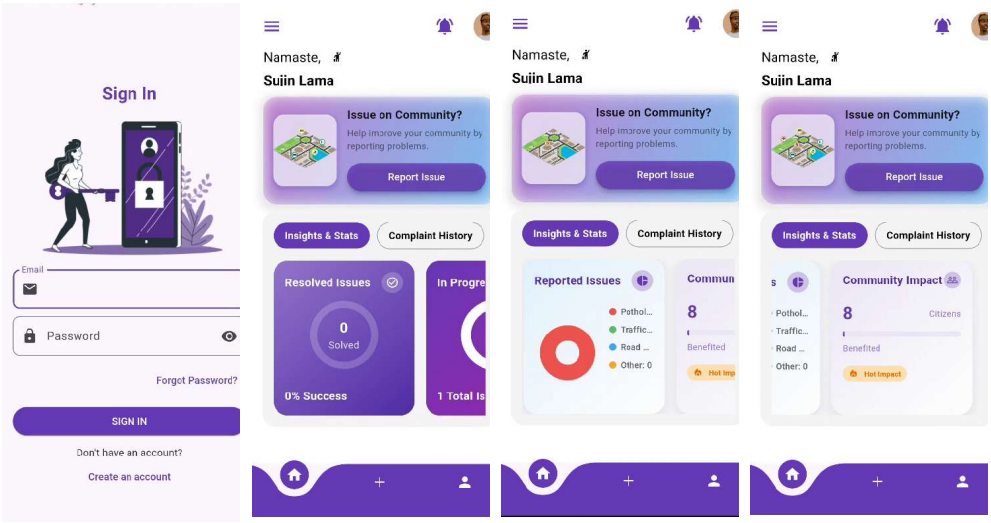


Figure 4.7: Login, Intro Interface with Scrolling View

The login screen is simple, with options to sign in or create an account. After logging in, users are greeted by name and can quickly report problems with just one tap. The dashboard shows useful stats like how many issues are resolved or in progress. Users can also view their complaint history and see how their reports are making a real impact through clear, visual insights. The clean and modern design makes the whole experience smooth and user-friendly.

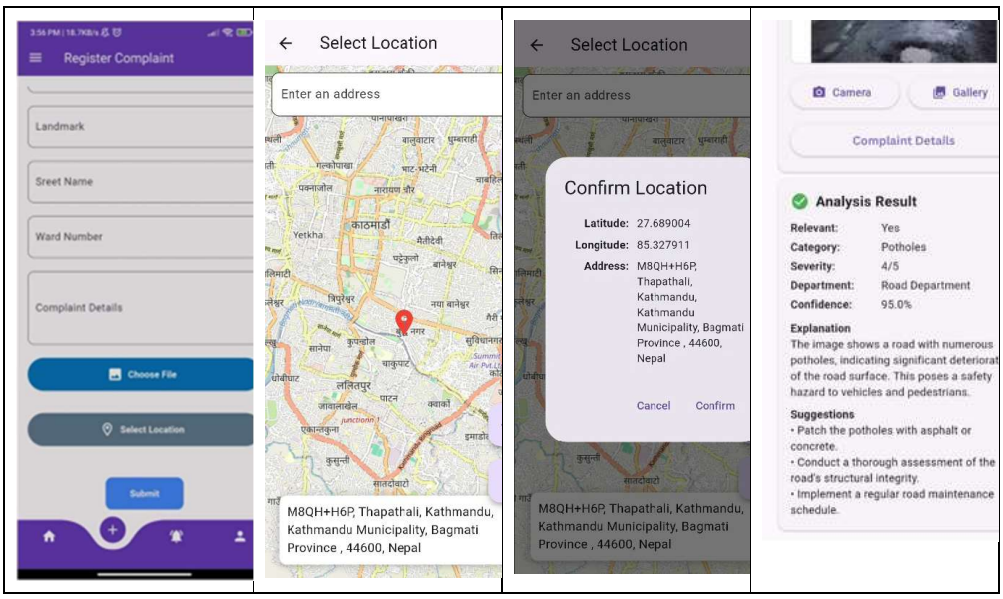


Figure 4.8: Complaint Registration and AI-Based Analysis Flow

Users can upload a photo, mark the location on a map, and get updates on their complaint. What makes the app even smarter is the built-in AI, when someone uploads an image, a custom-trained model automatically detects and highlights the problem, helping speed up the process with an AI-generated analysis result. This not only saves time for users but also helps authorities respond faster.

Chapter 5: Remaining/Future Work

While a significant portion of the Smart Community Complaint App has been completed, a few essential tasks remain, alongside opportunities for future enhancement:

5.1. Remaining Work

1. **ML-Based Complaint Classification:** Integration of machine learning models to classify complaints based on text and images is pending. This will improve prioritization and automate initial assessment.
2. **Automated Routing to Departments:** Implementation of a smart routing system that forwards complaints to relevant municipal departments based on category and location.
3. **Offline Functionality:** Offline complaint submission needs final integration to ensure seamless reporting in low-connectivity areas.
4. **Real-Time Notifications:** Enhancing the push notification system to provide timely updates on complaint status and authority responses.
5. **UI/UX Enhancements:** Improving the user interface and overall experience for easier navigation, better usability, and accessibility.

5.2. Future Work

1. **Multilingual Support:** Enabling regional language support to cater to users across diverse linguistic backgrounds.
2. **Chatbot for Complaint Submission:** Introducing a chatbot to assist users in filing complaints through guided conversation.
3. **Predictive Issue Forecasting:** Using historical data to anticipate and alert authorities about recurring or seasonal problems.
4. **Citizen Feedback System:** Allowing users to rate complaint resolutions, fostering transparency and accountability.

5. **Admin Dashboard and Analytics:** Developing a dashboard for authorities to monitor complaints, track performance, and generate data-driven insights.

6. **Voice Input for Complaint Submission:** Adding voice input functionality to allow users to submit complaints hands-free, enhancing accessibility for all.

REFERENCES

(2025). Retrieved March 26, 2025, from Nagarik App: https://nagarikapp.gov.np/

Acharya, D. S. (2023, November 29). Retrieved March 24, 2025, from nepallivetoday: https://www.nepallivetoday.com/2023/11/29/challenges-and-prospects-in-sustainable-infrastructure-development-in-nepal/

Adhikari, E. R. (2024, December 05). Retrieved March 21, 2025, from thehimalayantimes: https://thehimalayantimes.com/opinion/transportation-challenges-in-nepal-consequences-and-solutions

Ahmed Fawzy Gad, J. S. (2024, August 28). Retrieved March 28, 2025, from Digital Ocean: https://www.digitalocean.com/community/tutorials/mean-average-precision

Anglen, J. (2024). Retrieved March 25, 2025, from Rapid Innovation: https://www.rapidinnovation.io/post/top-10-open-source-computer-vision-libraries-you-need-to-know

Balduino, R. (2021, March 9). Retrieved March 24, 2025, from Medium: https://medium.com/ibm-data-ai/how-ai-can-transform-your-complaints-management-strategy-82cba16a82e

Danish Khan, S. S. (2023). CIVIC SOCIAL COMPLAINT ANDROID BASED APP. *International Research Journal of Modernization in Engineering Technology and Science*. Retrieved March 21, 2025

Dishant Banga, K. P. (2023). Artificial Intelligence for Customer Complaint. *International Journal of Computer Trends and Technology*. Retrieved March 27, 2025

Everingham, M. V. (2010). The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*. Retrieved from https://doi.org/10.1007/s11263-009-0275-4

Foundation, P. S. (2025). Retrieved March 21, 2025, from Python: https://www.python.org/

Google. (2025). Retrieved March 24, 2025, from Android Studio: https://developer.android.com/studio

Google. (2025). Retrieved March 25, 2025, from Flutter: https://docs.flutter.dev/resources/faq

Google Apis. (2012). Retrieved March 12, 2025, from googleapis: https://storage.googleapis.com/openimages/web/index.html

Hussain, M. (2023, June 23). YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. doi:https://doi.org/10.3390/machines11070677

Kundu, R. (2023, January 17). *v7labs*. Retrieved June 19, 2025, from Resources: https://www.v7labs.com/blog/yolo-object-detection

Lamichhane, R. (2024, February 11). Retrieved March 28, 2025, from The Kathmandu Post: https://kathmandupost.com/columns/2024/02/11/paving-the-path-to-prosperity

Maharjan, N. (2019). Retrieved March 26, 2025, from Nepal Economic Forum: https://nepaleconomicforum.org/revisiting-nepals-infrastructural-bottlenecks-and-progress/

Mary Jane C. Samonte, J. M. (2019, March). E-Complaint: An Analytical Crowdsourcing Mobile Application for Community Peace and Order System. Retrieved March 24, 2025

Matouq, Y. M. (2024). AI-driven approach for automated real-time pothole detection, localization, and area estimation. doi:https://doi.org/10.1177/03611981241246993

Microsoft. (2025). Retrieved March 28, 2025, from Visual Code: https://code.visualstudio.com/

NVIDIA Corporation. (2023). Retrieved March 20, 2025, from NVIDIA: https://developer.nvidia.com/about-cuda

NVIDIA Corporation. (2023). Retrieved March 20, 2025, from NVIDIA: https://developer.nvidia.com/cudnn

Purbey, A. (2023, June 14). Retrieved March 29, 2025, from nepallivetoday: https://www.nepallivetoday.com/2023/06/14/what-impedes-nepals-infrastructure-governance/

Subramanyam, V. S. (2021, January 20). Retrieved June 19, 2025, from Medium: https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536

Team, D. (2025). Retrieved March 22, 2025, from Dart: https://dart.dev/

Team, G. (2024, July 3). Retrieved March 26, 2025, from GeeksforGeeks: https://www.geeksforgeeks.org/software-engineering-agile-development-models/

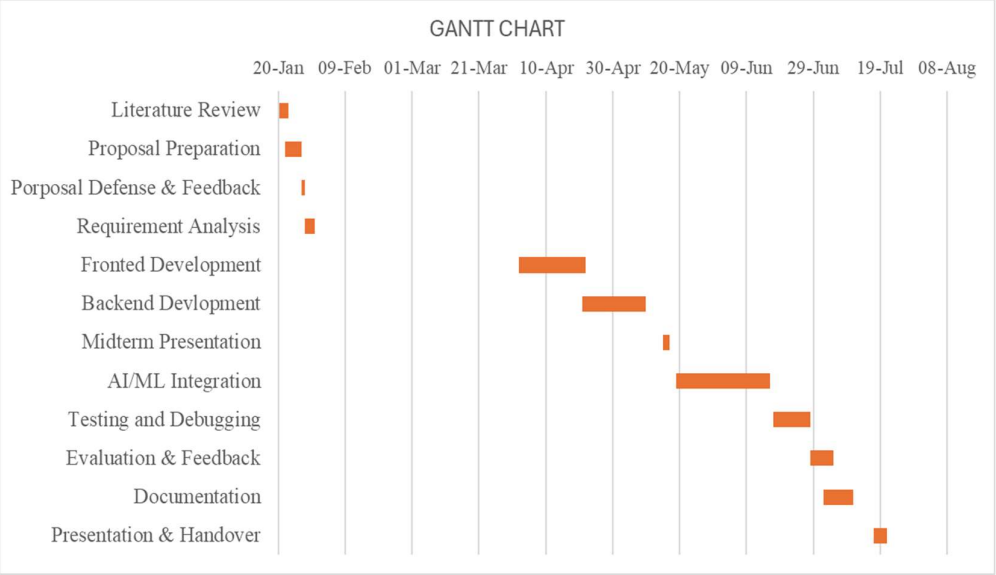
Team, G. (2025). Retrieved March 25, 2025, from Git SCM: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Thapa, E. K. (2024, August 05). Retrieved March 23, 2025, from nepalinfrastructuresummit: https://nepalinfrastructuresummit.com.np/blog/sustainable-urban-infrastructure

APPENDICES

Before starting any project, it's important to create a clear schedule that outlines the tasks and milestones for the development phase. To ensure effective time Maintenance, we have prepared a Gantt chart covering 3 months, spanning 10 weeks from the phase after the proposal defense to the final report submission and defense. This schedule acts as a roadmap, helping us stay organized, track progress, and meet deadlines. By following the allocated time frames for each task, we can ensure efficient project execution and timely completion.

Table 1: Gantt chart of the project



(Group Study, 2025)

The proposed research project is structured as a comprehensive academic initiative, strategically aligned with the academic calendar across the 7th and 8th semesters. The initial phase encompasses critical preparatory activities, beginning with a rigorous literature review to establish a robust theoretical foundation. This initial stage is immediately followed by meticulous proposal preparation and a focused

proposal defense, allowing for comprehensive requirement analysis and strategic project scoping.

The development trajectory progresses through distinct technical stages, commencing with frontend development and subsequently transitioning to backend infrastructure construction. A midterm presentation provides a critical checkpoint for progress evaluation and stakeholder engagement. The project distinguishes itself through a substantial AI/ML integration phase, representing a significant technological component that demands 28 days of dedicated implementation and refinement.

The concluding phases prioritize systematic validation and documentation, featuring comprehensive testing and debugging protocols, followed by structured evaluation and feedback mechanisms. The project culminates in a comprehensive documentation process and a formal presentation and project handover, ensuring a structured and professional conclusion to the research initiative. This methodical approach underscores a rigorous, academically-oriented project management strategy that balances technical innovation with methodical execution.