ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

Community Infrastructure Failure Intelligence Platform

Mathireddy Tulasi Satya[1] , Oleti Naga Ramya[2] , Vericharla Wincent Paul[3] , Mayabramha Sricharan[4] , Dr. Dharmendra Roy[5]

CSE-Computer Science and Engineering Hyderabad Institute of Technology and Management Hyderabad, Telangana

DOI: https://doi.org/10.51583/IJLTEMAS.2026.151000079

Received: 17 October 2026; Accepted: 22 October 2026; Published: 07 November 2026

ABSTRACT

Public infrastructure issues such as potholes, road cracks, damaged streetlights, garbage accumulation, drainage problems, and waterlogging can affect road safety, public health, and the quality of everyday life. Reporting and managing these issues requires an organized process through which problems can be reported, located, monitored, and addressed. The Community Infrastructure Failure Intelligence Platform (CIFI) is a web-based application developed to support the reporting and management of public infrastructure issues. The platform allows users to submit complaints with issue descriptions, images, and location details, helping to maintain relevant information about reported problems. YOLOv11 is used for image-based detection of supported infrastructure issues to assist in identifying the type of problem shown in an uploaded image. The system also includes issue tracking, status updates, department-wise organization of complaints, and map-based visualization of reported issues. These features help present infrastructure complaints in a structured manner and make their progress easier to monitor. The platform brings issue reporting, image-based detection, location information, and complaint management together in a single application. The implementation is intended to support more organized handling of public infrastructure complaints and improve the visibility of reported issues and their status. The effectiveness of the implemented features can be assessed through functional testing and image-detection evaluation using actual test data.

Keywords— Public Infrastructure, Infrastructure Issue Reporting, YOLOv11, Object Detection, Complaint Management, Geographic Information System, Issue Tracking

INTRODUCTION

Public infrastructure plays an important role in maintaining safety, accessibility, and the quality of life of the public. Infrastructure problems such as potholes, road cracks, damaged streetlights, garbage accumulation, drainage blockages, and waterlogging can cause inconvenience and create safety risks when they are not reported and addressed appropriately.

Conventional complaint reporting methods may involve phone calls, written complaints, websites, or other reporting channels. When reports are maintained separately, it can become difficult to organize complaints, identify their locations, track their progress, and understand the distribution of infrastructure problems. Multiple reports may also refer to the same issue, while incomplete descriptions or photographs can make it more difficult to understand the reported problem.

A web-based reporting system can bring issue descriptions, photographs, and location details together in a structured format. Image-based object detection can further assist in identifying supported infrastructure issue categories from submitted photographs. Map-based visualization can help users view the locations of reported issues, while status tracking can provide information about the progress of each complaint.

The Community Infrastructure Failure Intelligence Platform (CIFI) is developed to support the reporting and management of public infrastructure issues through a web-based application. Users can submit issue reports with relevant descriptions, images, and location information. The platform uses YOLOv11 for image-based detection of supported infrastructure issue categories and provides functions for viewing reports and monitoring

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

their status. Department-wise organization and geographical visualization help present the available complaint information in a more organized manner.

The main contributions of this work are:

1. Public infrastructure reporting: A web-based interface for submitting infrastructure complaints with descriptions, photographs, and location details.

2. Image-based issue detection: Integration of YOLOv11 for detecting supported infrastructure issue categories from submitted images.

3. Issue monitoring: Functions for viewing reported complaints, tracking their status, and organizing them by department.

4. Geographical visualization: Map-based presentation of reported issues when location information is available.

5. Prototype evaluation: Identification of suitable functional tests and object-detection metrics for assessing the implemented system.

The paper describes the system design, the implementation of its main components, and the evaluation required to assess its performance. Features that have not been fully implemented or experimentally tested are identified separately rather than presented as completed capabilities.

Related Work

Several studies have explored the use of artificial intelligence and digital technologies for road hazard detection and public infrastructure issue management. Existing approaches focus on different aspects, including image-based hazard detection, condition assessment, digital complaint reporting, automated issue management, and dashboard-based monitoring.

A. YOLOv12-Based Hazard Detection

A YOLOv12-based road hazard detection system was developed to identify potholes, manholes, and speed breakers from road images. The system combines YOLOv12 object detection with MiDaS depth estimation to assess pothole severity and identify potentially dangerous road conditions. It also evaluates speed breakers based on paint visibility and classifies manholes according to their safety condition. A Flask-based web application allows users to upload images and view annotated detection results. The system achieved an overall mAP@0.5 of 0.854, with class-specific scores of 0.777 for potholes, 0.893 for manholes, and 0.893 for speed breakers. However, the study identifies limitations related to changing lighting conditions and the use of relative rather than absolute depth measurements [1]. While this approach concentrates on road hazard detection and condition assessment, CIFI uses a YOLOv11-based model to identify supported public infrastructure issue categories and focuses on integrating image-based identification with issue reporting and geographical visualization.

B. City Solution

City Solution presents a digital civic issue management approach that incorporates complaint storage, voting mechanisms, and a dashboard for monitoring reported issues [2]. These features support the organized collection of civic complaints and provide a way to review reported problems. However, the approach described in the reviewed material identifies duplicate complaint detection and AI-based classification as areas for further development. CIFI addresses a related problem by combining image-based infrastructure issue identification with the organization and visualization of public infrastructure reports.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

C. Smart Reporter

Smart Reporter explores a reporting framework that includes multiple reporting inputs, automated issue routing, dashboards, and citizen feedback [3]. This approach highlights the importance of connecting complaint submission with issue management and monitoring. It also demonstrates the relevance of providing users with information about reported issues and their progress. CIFI follows a related approach by focusing on public infrastructure issue reporting, image-based identification, geographical visualization, and complaint monitoring within a unified platform.

D. AI-Powered Civic Issue Resolution

An AI-powered civic issue resolution approach combines complaint classification, priority assessment, duplicate detection, and analytics to support the management of civic complaints [4]. Such approaches demonstrate how artificial intelligence can assist in organizing reports and identifying issues that may require attention. CIFI addresses a similar problem through image-based identification of supported infrastructure issue categories and the geographical organization of reported issues. Its contribution lies in bringing these functions together with public infrastructure reporting and monitoring.

Research Gap

The reviewed studies address different aspects of infrastructure hazard detection and civic issue management. The YOLOv12-based system focuses on detecting road hazards and assessing their conditions using depth estimation, while digital civic reporting systems emphasize complaint collection, routing, and monitoring. These approaches highlight the importance of combining visual identification with effective complaint management.

CIFI focuses on public infrastructure issues by integrating YOLOv11-based image identification with issue reporting and geographical visualization. It aims to provide a unified platform for organizing reported infrastructure problems and monitoring their status. The system's contribution should be assessed based on its implemented functionality and experimental evaluation, including the performance of its image-identification component and the successful operation of its reporting and visualization features.

III. Proposed System

The Community Infrastructure Failure Intelligence Platform (CIFI) is a web-based platform designed to support the reporting, identification, visualization, and monitoring of public infrastructure issues. It focuses on problems such as potholes, road cracks, damaged streetlights, garbage accumulation, drainage issues, and waterlogging, depending on the categories supported by the implemented model.

The platform allows users to submit infrastructure reports containing relevant descriptions, images, and location information. A YOLOv11-based object detection model is used to identify supported infrastructure issue categories from images. The reported information can then be organized and presented through the platform's interface to support issue monitoring and management.

The system is designed around the following functional components:

1. Infrastructure Issue Reporting: Enables users to submit reports containing issue descriptions, photographs, and available location information.

2. Image-Based Issue Identification: Uses the YOLOv11 model to identify supported infrastructure issue categories from submitted images.

3. Geographical Visualization: Presents location information associated with reports on a map, where the corresponding functionality is implemented.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

4. Complaint Organization: Organizes submitted reports to support their review and management. Automated duplicate identification is treated as a separate function and must be evaluated before its effectiveness can be established.

5. Issue Monitoring: Provides a view of reported issues and their status through the dashboard.

6. Department-Based Organization: Supports the organization of issues according to department or responsibility, where this functionality is available in the implemented application.

Duplicate identification based on category and location, geographical clustering, and automated urgency assessment require clearly defined methods and validation. These functions are described as proposed components unless their implementation and operation have been verified. This distinction prevents the system's intended capabilities from being confused with its demonstrated functionality.

IV. System Architecture and Methodology

In this section, we describe the design and working of the CIFI platform. The system handles infrastructure complaints through a series of connected modules: user reporting, image detection using YOLO, text classification using NLP, duplicate detection, geographical clustering, and risk prioritization.

The overall workflow of the proposed CIFI system is illustrated in Fig. 1.

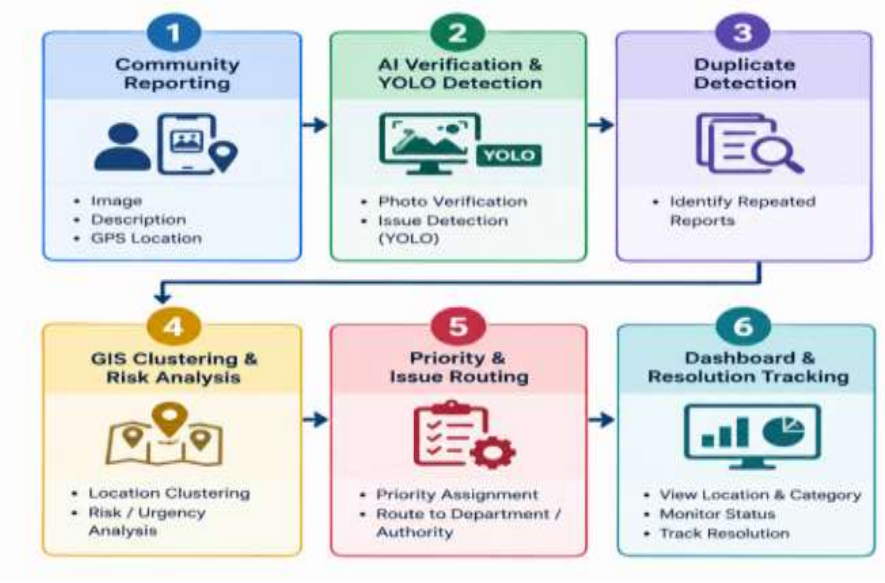


Fig-1 A. Infrastructure Issue Categories

The platform focuses on six common community infrastructure issues. Each issue category is mapped to a responsible municipal department, a base priority level, and an initial hazard score, as shown in Table I.

TABLE I: Infrastructure Defect Categories and Department Mapping

Class ID	Defect Category	Assigned Department	Default Priority	Base Score (out of 100)
0	Pothole	Roads & Infrastructure	HIGH	75
1	Road Surface Crack	Roads & Infrastructure	MEDIUM	55
2	Broken Streetlight	Electrical & Lighting	HIGH	80
3	Garbage Dump	Sanitation & Solid Waste	MEDIUM	50

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

4	Drainage Issue	Stormwater & Drainage	HIGH	70
5	Waterlogging	Stormwater & Drainage	HIGH	85
6	Unclassified / Other	Public Works General	MEDIUM	50

B. YOLO Model Setup and Configuration

For identifying defects from uploaded photos, we use the Ultralytics YOLO nano model (yolo11n). We chose the nano version because it provides fast inference on standard CPUs, making it practical for real-time web deployment without requiring expensive GPUs.

1. Model Architecture:

- Backbone: Modified CSPDarknet, which extracts visual features such as edges, dark cavities, and surface textures.
- Neck: Path Aggregation Network (PANet), which combines low-level features (needed for thin road cracks) with high-level features (needed for large flooded areas).
- Head: Decoupled, anchor-free detection head that predicts bounding boxes and class probabilities separately.

2. Training and Inference Settings: To ensure our work can be reproduced, Table II lists the exact training and detection parameters used in the system.

TABLE II: YOLO Model Configuration and Hyperparameters

Parameter	Setting	Reason for Selection
Model	Ultralytics YOLO nano	Fast inference (< 45 ms on CPU) with low memory footprint
Input Image Size	640 × 640pixels	Balances fine defect detail with reasonable processing time
Batch Size	16	Stable gradient updates during training
Optimizer	AdamW	Prevents overfitting on smaller civic datasets
Initial Learning Rate	0.001	Standard learning rate for stable convergence
Weight Decay	0.0005	Regularization to prevent overfitting
Confidence Threshold	0.25	Ignores low-confidence background noise
NMS IoU Threshold	0.45	Removes overlapping boxes around the same defect
Number of Classes	6	Matches our six municipal defect categories

3. Inference and Bounding Box Output: When an image is submitted, the model outputs bounding boxes for any detected defects: Detection $= [x_1, y_1, x_2, y_2, \text{confidence}, \text{class_name}]$ where (x_1, y_1) is the top-left corner and $(x_2, y_2)$is the bottom-right corner in pixels.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

Safety Rule: If the trained model file (cifi_best.pt) is not loaded or missing from disk, the code does not invent fake detections. Instead, it returns {"status": "model_unavailable"} and routes the complaint to manual human review.

C. Natural Language Processing (NLP) Setup

To understand citizen complaint descriptions, we built an NLP text classification pipeline using Python and Scikit-Learn.

1. Text Preprocessing: Citizen descriptions often contain typos, informal slang, and extra characters. Before classification, the text passes through five cleaning steps:

- Converting all text to lowercase.
- Removing URLs, email addresses, and phone numbers using regular expressions.
- Removing punctuation and symbols while keeping standard spaces.
- Collapsing multiple spaces and tabs into a single space.
- Preserving civic keywords such as *manhole, sewer, pothole, wire,* and *gutter*.

2. Feature Extraction and Classifier:

- We use TF-IDF Vectorization with both single words and two-word pairs (unigrams and bigrams, ngram_range=(1, 2)). This helps the model recognize phrases like *"live wire"*, *"water pipe"*, and *"open drain"*.
- We apply sublinear term frequency scaling $(1 + \log(\text{tf}))$ to prevent frequently repeated words from dominating the score.
- The classifier is Multinomial Logistic Regression with L_2regularization ($C = 2.0$), trained over 1,000 iterations.

D. Multimodal Decision Fusion (Combining Text and Image)

Because text descriptions can be vague and photos can occasionally be blurry, CIFI checks both inputs together in aiEngine.js:

1. Agreement (Consensus): If the text model and the YOLO vision model both predict the same category (e.g., both say "pothole"), the system confirms high confidence and automatically sends the complaint to the relevant department.
2. Disagreement (Conflict): If the text says one thing (e.g., "broken streetlight") but the photo shows something else (e.g., "garbage dump"), the system flags the report with: humanReviewRequired = trueIt logs the reason for the mismatch and sends the ticket to the Admin AI Review Desk so a municipal officer can verify it before work crews are dispatched.
3. Threshold Guard: If the combined confidence score is below 0.70 (70%), the complaint is automatically flagged for human review.

E. Duplicate Complaint Detection Criteria

When a major problem occurs (like a large pothole on a busy road), several citizens often report it. To avoid creating duplicate work orders, CIFI compares each new complaint against existing active complaints within a 500-meter radius over the past 14 days.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

We calculate a composite similarity score between 0.0 and 1.0 using three weighted factors: Similarity Score $= (0.60 \times S_{\text{text}}) + (0.25 \times S_{\text{geo}}) + (0.15 \times S_{\text{cat}})$

1. Text Similarity (S_{text} , 60% weight): We use the Jaccard Similarity coefficient on word sets from both descriptions: $S_{\text{text}} = \frac{|T_A \cap T_B|}{|T_A \cup T_B|}$where $T_A$and $T_B$are the sets of unique words (words with at least 3 letters).

2. Geographical Proximity (S_{geo} , 25% weight): We calculate the real-world distance $d$in meters between the two GPS coordinates using the Haversine formula on Earth's sphere ($R = 6{,}371{,}000$ meters): $\Delta\phi = \phi_2 - \phi_1$, $\Delta\lambda = \lambda_2 - \lambda_1 a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right) d = 2R\arcsin\left(\sqrt{a}\right)$We then convert distance into a proximity score using practical distance brackets:

$$S_{\text{geo}} = \begin{cases} 1.00 & \text{if } d \leq 50 \text{ meters (same spot)} \\ 0.80 & \text{if } 50 \text{ m} < d \leq 150 \text{ meters (same street block)} \\ 0.50 & \text{if } 150 \text{ m} < d \leq 300 \text{ meters (nearby intersection)} \\ 0.25 & \text{if } 300 \text{ m} < d \leq 600 \text{ meters (same general area)} \\ 0.00 & \text{if } d > 600 \text{ meters (different location)} \end{cases}$$

3. Category Similarity (S_{cat}, 15% weight):

- 1.00if categories match exactly.
- 0.70if categories are closely related (e.g., Pothole and Road Crack, or Drainage and Waterlogging).
- 0.00if categories are unrelated.

Decision Rule: If the final similarity score is 0.70 or higher, the new report is marked as a duplicate of the existing ticket. The parent ticket's duplicate counter is increased, but no new work order is created.

F. Geographical Clustering and Hotspot Detection

To show municipal authorities where issues are clustering, CIFI uses MongoDB's native GeoJSON 2dsphere spatial index.

1. Storage Format: Each complaint stores its location as a standard GeoJSON Point: location: {type, :"Point"coordinates: [longitude, latitude]}
2. Clustering Method: The system queries active complaints within a sliding 14-day window. Reports that fall within a 500-meter radius of one another are grouped into a cluster.
3. Centroid and Map Display: For clusters with 3 or more complaints, the system calculates the geographical center (centroid) and displays it on the Leaflet OpenStreetMap dashboard with a colored circle. The circle's size and color reflect the number of complaints and their average severity.

G. Urgency and Priority Scoring Mechanism

The platform calculates an objective risk score from 10 to 100 for each complaint so that dangerous situations are handled first: Priority Score $= \min\left(100, \text{round}(\text{Base Score} + \text{Cluster Bonus} + \text{Context Bonus} + \text{Recurrence Bonus})\right)$

1. Base Score: Taken from the category baseline in Table I (e.g., 85 for waterlogging, 75 for potholes).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

2. Multiple Defect Bonus: If YOLO finds more than two defects in a single photo (> 2 bounding boxes), it adds +8 points.

3. High-Risk Keyword Bonus: The system scans the text description for high-risk words using regular expressions:

- *Keywords*: accident, hospital, school, emergency, danger, live wire, sparking, open manhole, collapsed.
- If any of these words match, it adds +10 points.

4. Recurrence Bonus: If an issue has been reported multiple times, each report adds +5 points (up to a maximum of +15 points).

The final score maps directly to response deadlines (Service Level Agreements, or SLAs), as detailed in Table III.

TABLE III: Priority Score Tiers and Response Deadlines

Priority Level	Score Range	Resolution SLA	Action Required
CRITICAL	85 to 100	12 Hours	Immediate emergency dispatch; alert sent to senior officer
HIGH	70 to 84	24 Hours	Prioritized in departmental work queue
MEDIUM	40 to 69	48 Hours	Scheduled in standard maintenance batch
LOW	Below 40	96 Hours	Routine upkeep and non-urgent cosmetic fixes

H. Implemented Features vs. Future Work

To clarify the current state of the platform, Table IV outlines which parts are fully built and tested in our codebase and which are planned for future releases.

TABLE IV: Implemented Functionalities vs. Future Roadmap

System Component	Status	Codebase File / Implementation Details
React 18 Web Application	Implemented	client/src/App.jsx (Citizen, Staff, and Admin portals)
Node.js REST API & MongoDB	Implemented	server/server.js, server/models/Complaint.js with GeoJSON
Python FastAPI Microservice	Implemented	server/ai/nlp/api.py running on Port 8000
TF-IDF NLP Text Classifier	Implemented	server/ai/nlp/train.py, evaluated in metrics.json
YOLO Defect Detection	Implemented	server/ai/vision/detector.py (bounding box inference)
Duplicate Detection (Haversine + Jaccard)	Implemented	server/ai/intelligence/duplicateDetector.js
0–100 Priority Risk Scoring	Implemented	server/ai/intelligence/priorityEngine.js
Before/After Photo Verification	Implemented	client/src/pages/staff/StaffDashboard.jsx, upload.js
Recharts Analytics Dashboards	Implemented	client/src/pages/admin/AdminAnalyticsPage.jsx

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

Native Mobile Apps (Android/iOS)	*Planned*	Mobile wireframes designed; native build in future release
Multilingual Speech-to-Text	*Planned*	Planned integration of speech recognition for vernacular voice inputs
Edge Camera on Garbage Trucks	*Planned*	Planned deployment of YOLO on vehicle-mounted cameras

V. Data Management Design

The data management layer of CIFI is built using MongoDB, a NoSQL document database, through the Mongoose Object Data Modelling (ODM) library in Node.js. A document database was chosen because community complaints require flexible schemas that can store variable numbers of bounding boxes, spatial coordinates, and status timelines within a single record.

Database Entities

The system organizes information across seven main collections:

1. User: Stores account credentials, contact information, and role definitions (citizen, staff, or admin) to enforce role-based access control (RBAC).

2. Complaint: The master operational record containing the complaint ID, user reference, text description, photo path, GeoJSON location coordinates, predicted category, bounding box detections, priority score (0–100), duplicate status, and resolution progress.

3. AI Analysis: Stores audit data for each prediction, including raw class probability distributions from the NLP model, bounding box coordinates from the YOLO detector, and multimodal consensus logs.

4. Complaint Timeline: An immutable history log recording every status change (SUBMITTED →ASSIGNED →IN_PROGRESS →RESOLVED), along with the acting user's ID and timestamp.

5. Department: Defines municipal operational bodies (Roads, Electrical, Sanitation, Drainage) and maps them to specific defect categories and assigned personnel.

6. Notification: Stores automated in-app alerts sent to citizens and field staff when complaint statuses change.

7. Audit Log: A compliance log that records administrative overrides (such as category or priority adjustments) with timestamps, IP addresses, and user justifications.

Implementation

The CIFI platform is implemented as a decoupled full-stack web application designed for fast, real-time civic issue reporting and management. The software stack integrates four main technology layers:

1. Frontend Client: Developed using React 18 and Vite 6 with Tailwind CSS for responsive styling. React Router DOM manages role-protected page navigation, Axios handles HTTP requests with JWT authorization headers, Lucide React provides interface icons, and Recharts renders dynamic analytical charts.

2. Backend API Gateway: Powered by Node.js and Express.js, providing RESTful endpoints for user authentication, complaint submissions, and departmental routing. Multer manages multipart image uploads with a 10 MB limit and file sanitization. Socket.IO provides two-way WebSocket communication to push instant status alerts to connected users without page reloads.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

3. AI Microservice: Written in Python using FastAPI and Uvicorn, running on Port 8000. It hosts both the Scikit-Learn TF-IDF text classification pipeline and the Ultralytics YOLO computer vision detector, serving predictions to the Node.js backend through asynchronous REST endpoints (/classify, /detect, and /analyze).

4. Database Storage: Managed by MongoDB Community Server with Mongoose ODM, utilizing GeoJSON 2dsphere spatial indexing on coordinate pairs to enable fast proximity searches.

The application is structured into three dedicated, role-segregated user interfaces:

- Citizen Portal: Features a four-step reporting wizard, GPS-based location pinning, personal ticket tracking, and community confirmation tools.

- Field Staff Portal: Provides department-specific work queues (Roads, Electrical, Drainage, Sanitation), allowing field crews to update issue statuses from ASSIGNED to IN_PROGRESS and upload mandatory photographic evidence before marking tickets as RESOLVED.

- Administrative Command Center: Gives municipal commissioners global oversight through real-time telemetry, a human-in-the-loop AI Review Desk for resolving multimodal prediction disagreements, user management, and an immutable audit trail.

Prototype Implementation And Operational Workflow

VII.A. Prototype Overview

The Community Infrastructure Failure Intelligence (CIFI) platform was developed as a web-based prototype for reporting, monitoring, and managing community infrastructure issues. It provides separate interfaces for citizens, administrators, and staff members. The prototype covers issues such as potholes, road cracks, damaged streetlights, garbage accumulation, drainage problems, and waterlogging.

The frontend was developed using React and Vite, with Tailwind CSS for styling. The platform includes complaint forms, dashboards, complaint tables, and map-based views. Backend components support request processing, image uploads, and data management. A YOLO11n-based computer vision component is used to detect supported infrastructure-damage classes in submitted photographs.



Fig. 1. CIFI landing page and infrastructure issue reporting interface.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

VII.B. User Authentication and Account Creation

The prototype provides sign-in and registration interfaces for platform access. Existing users can enter their account credentials, while new users can create an account by submitting the required details. These interfaces form the entry point to the platform before users proceed to the functions available through the relevant interface.

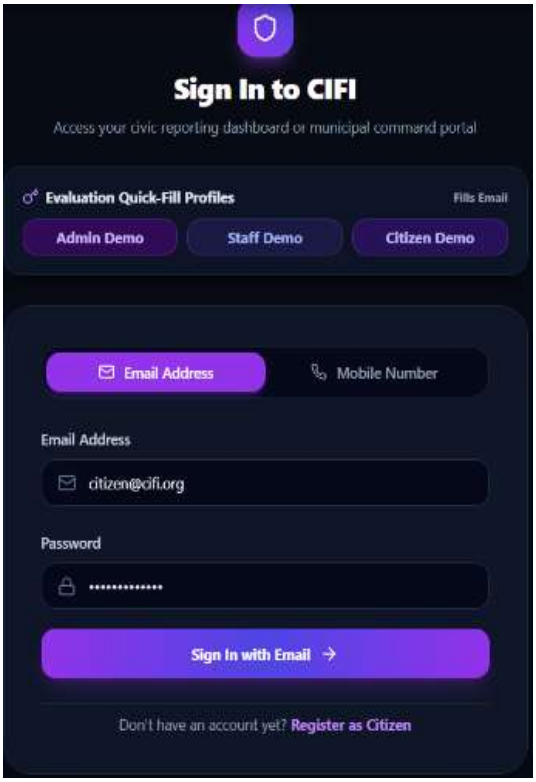


Fig. 2. User sign-in interface.

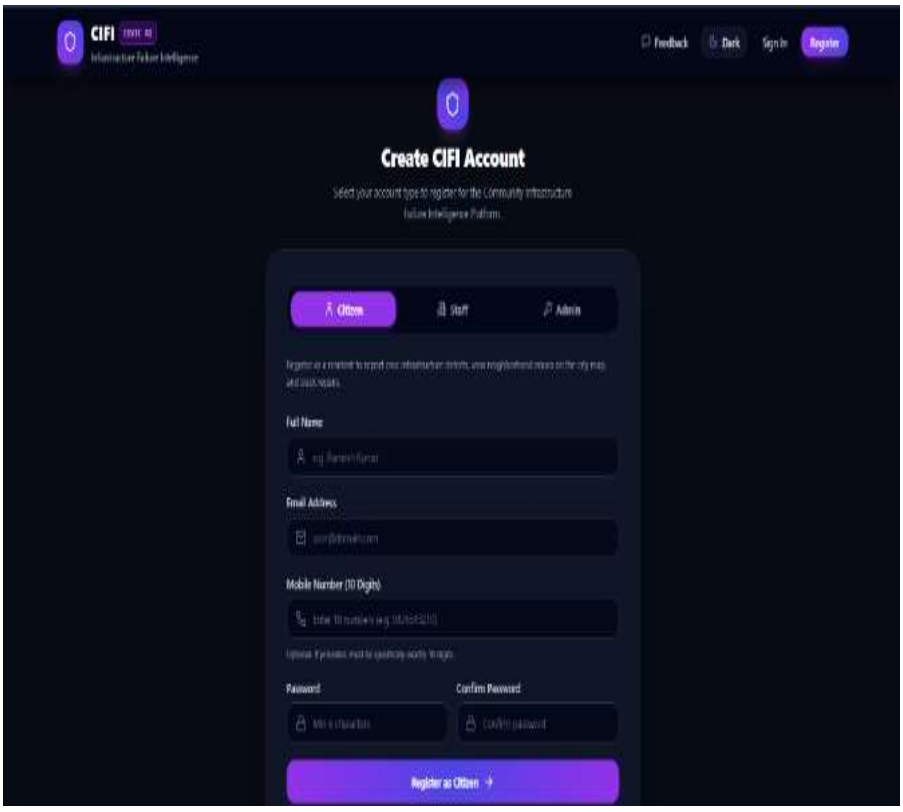


Fig. 3. User registration interface.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

VII.C. Citizen Complaint Reporting

The citizen reporting workflow allows users to select an infrastructure issue category, enter a description, provide location information where available, and upload a supporting photograph. The category and description explain the reported condition, while the photograph provides visual evidence for subsequent review. The submitted image can be processed by the YOLO11n-based computer vision component to detect supported damage classes.

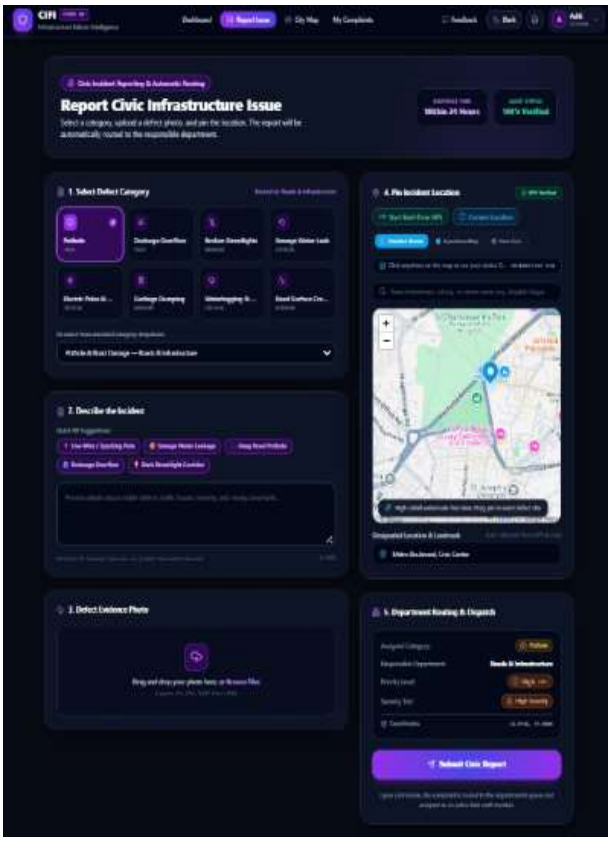


Fig. 4. Citizen issue reporting interface with complaint details and location information.

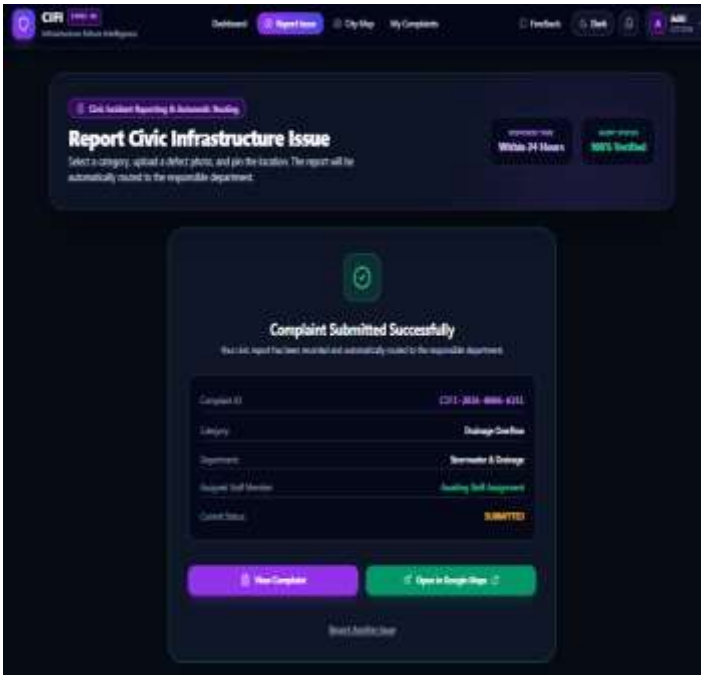


Fig. 5. Confirmation displayed after successful complaint submission.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

The confirmation view provides feedback that the reporting operation has completed successfully. It also presents the recorded complaint information, allowing the user to proceed from submission to the available tracking functions.

VII.D. Geographical Visualization and Complaint Details

CIFI includes a map-based view that displays reported issue locations. Map markers provide geographical context for complaints and help users or administrators examine where issues have been reported. The map complements the textual complaint records by making location information easier to review.

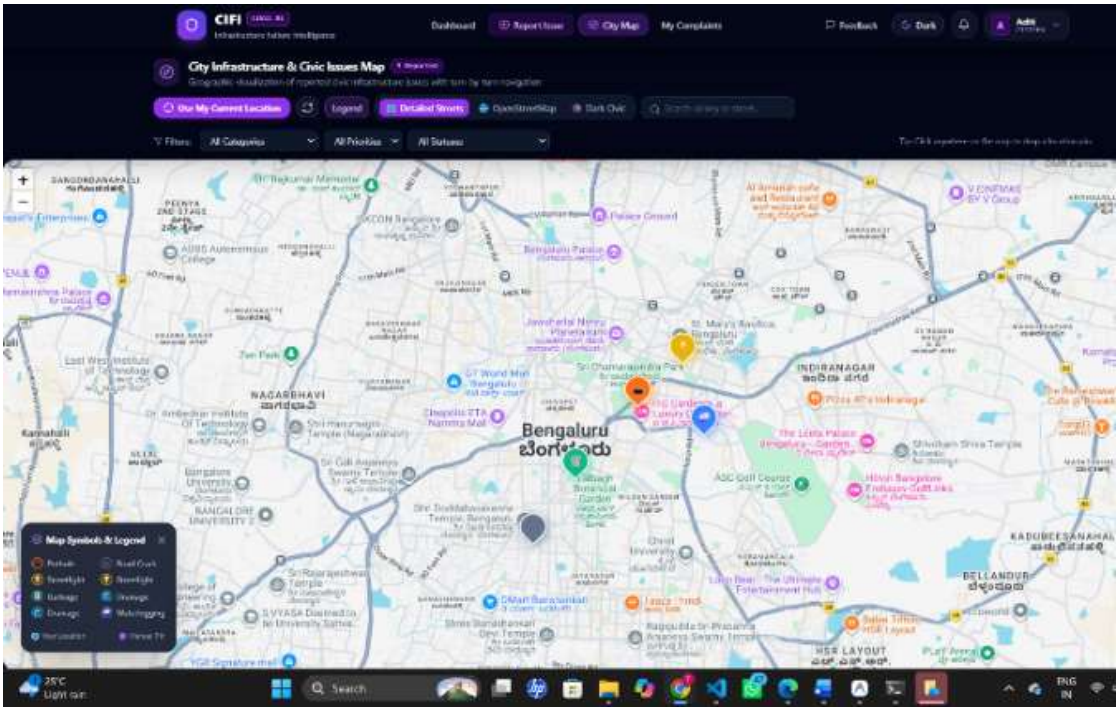


Fig. 6. Map-based visualization of reported infrastructure issue locations.

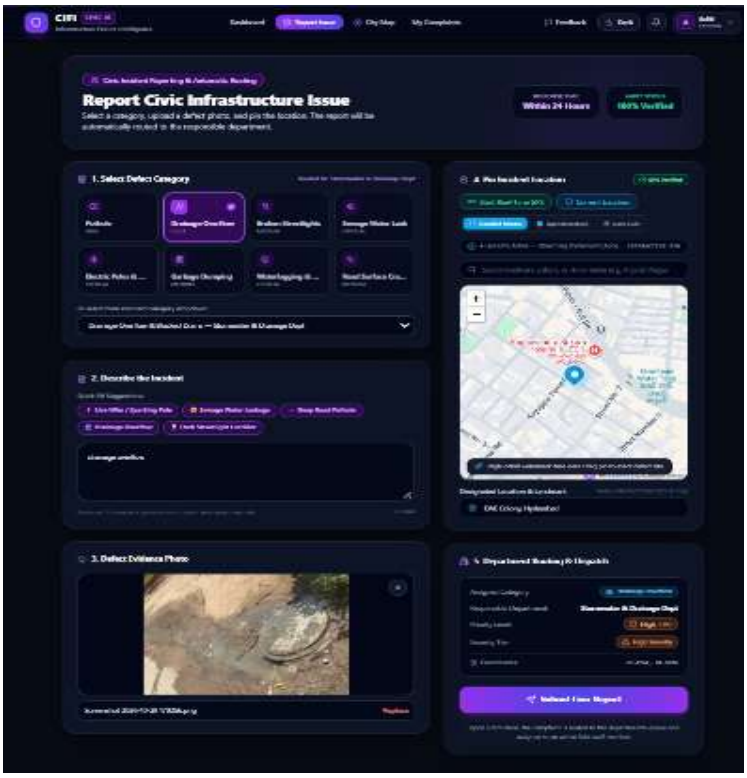


Fig. 7. Complaint details and supporting image interface.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

The complaint-details interface presents the available category, description, photograph, location, and status information for an individual report. This allows the reported condition to be reviewed after submission. Map visualization supports geographical examination, but proximity between map markers alone does not establish that complaints are duplicates.

VII.E. Administrative Complaint Management

The administrative interface provides a consolidated view of submitted complaints. Administrators can review complaint records, examine available details, and monitor the status information stored in the system. The table format supports reviewing multiple records without opening each complaint individually.

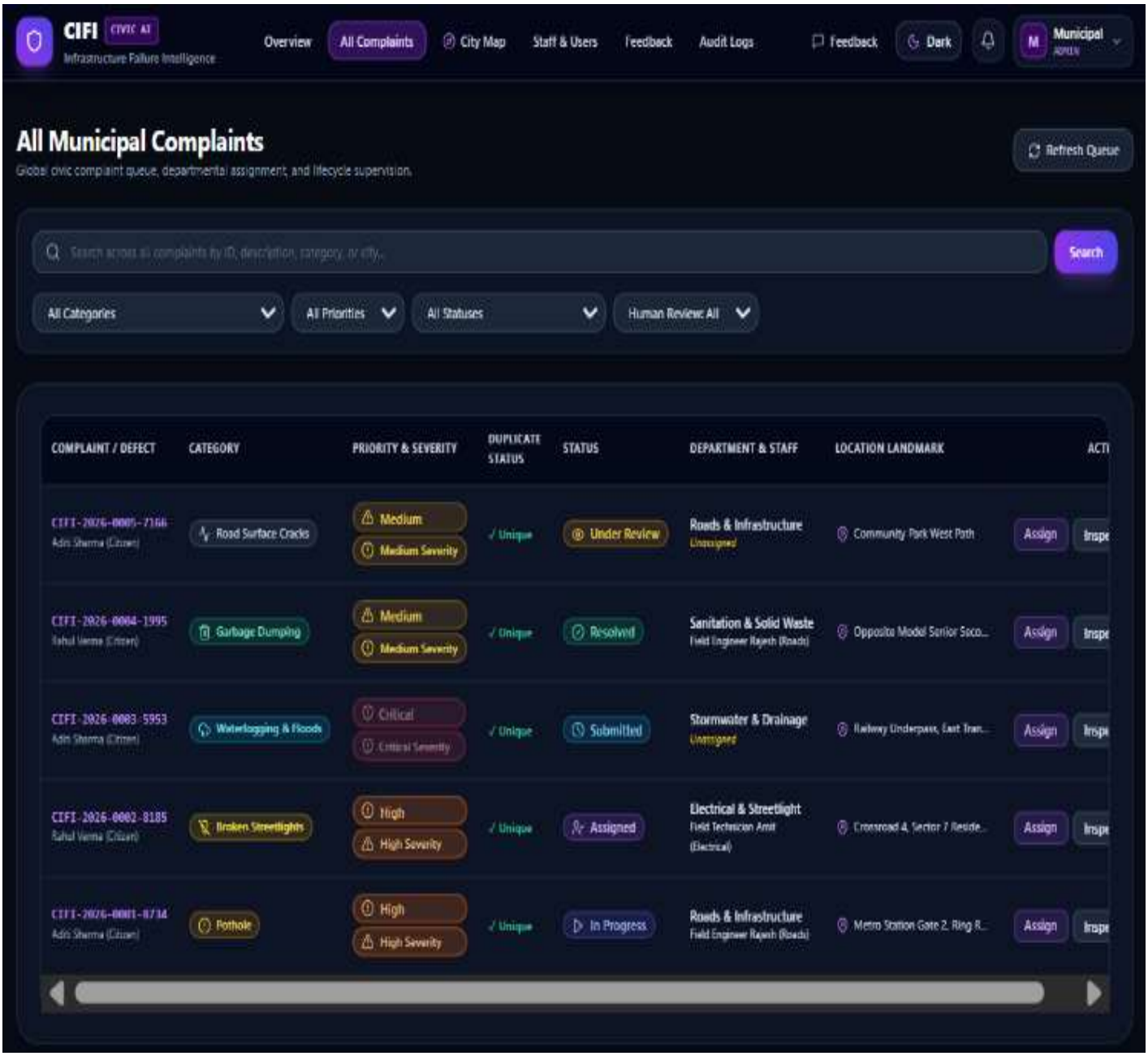


Fig. 8. Administrative table listing reported infrastructure complaints.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

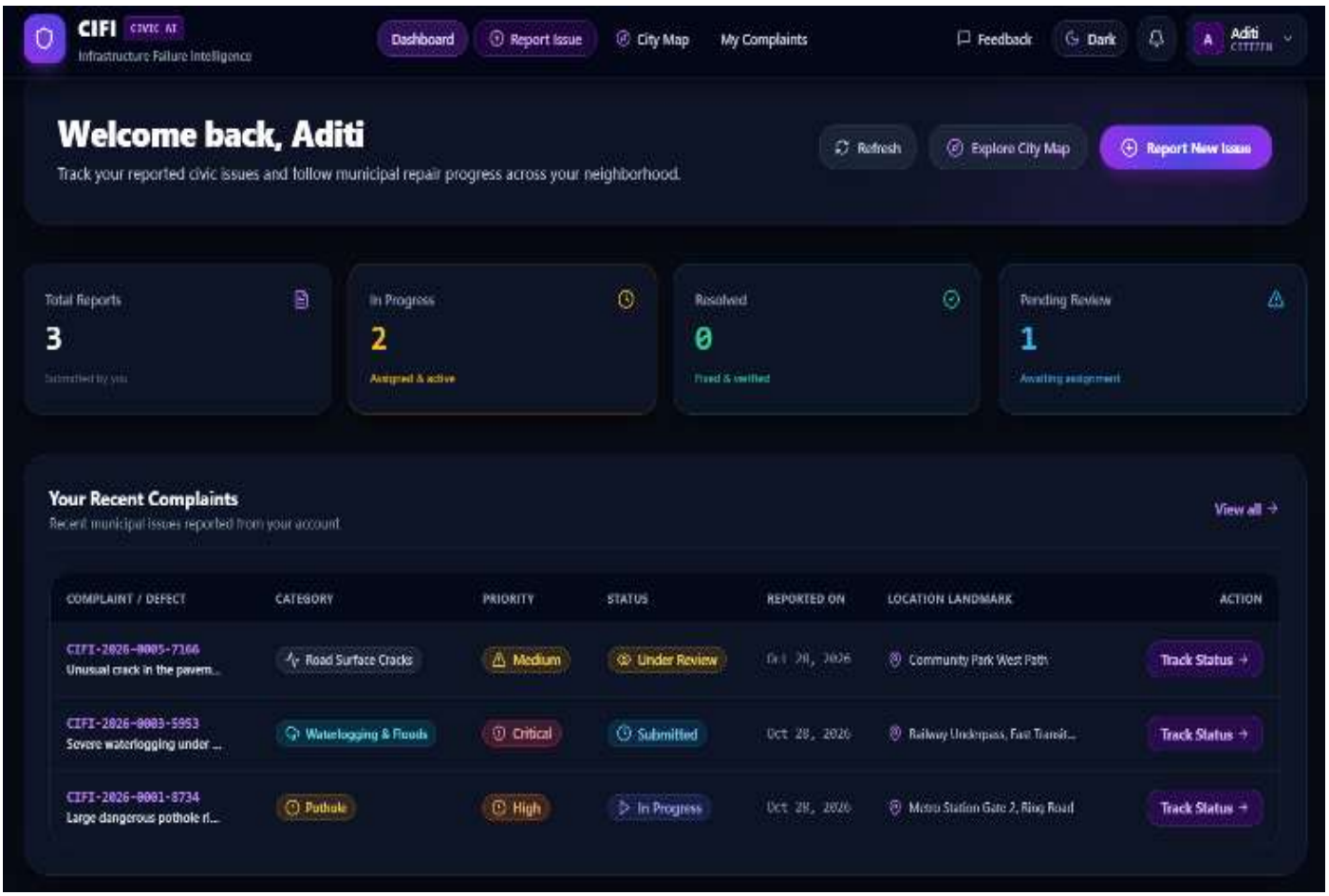


Fig. 9. Administrative dashboard with complaint summaries and status information.

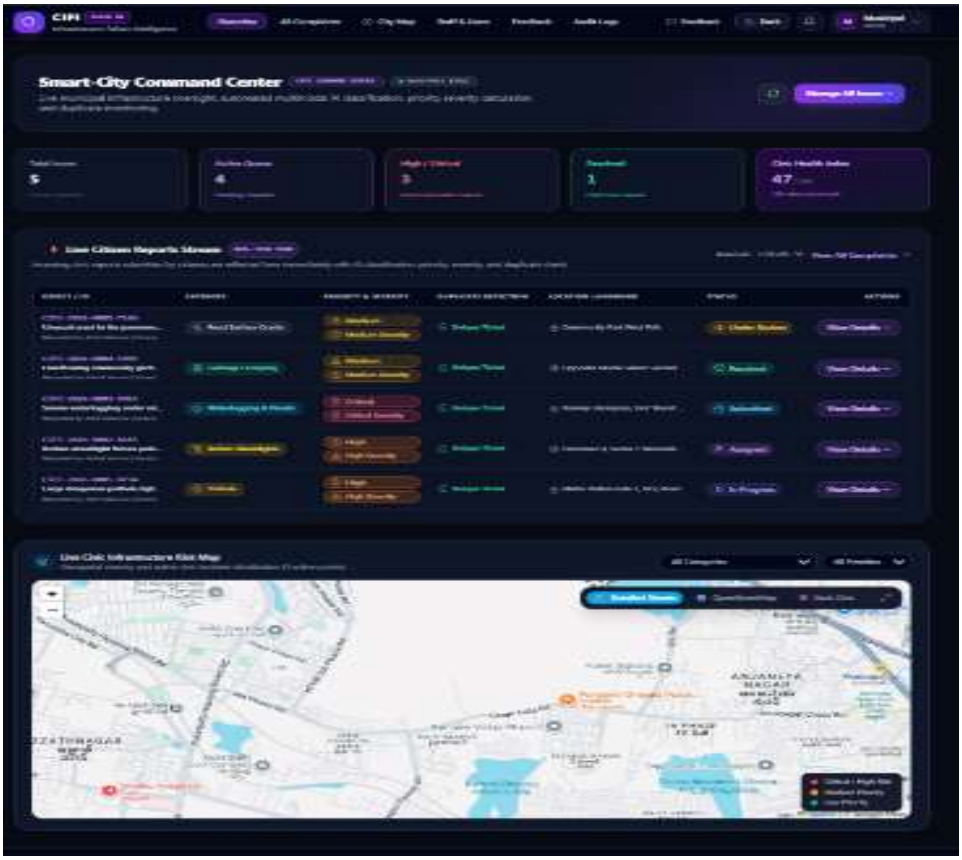


Fig. 10. Complaint management view combining records and geographical information.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

The dashboard summarizes the complaint information available in the prototype, while the complaint table supports examination of individual records. Department-related information can help administrators review how complaints are organized for handling. These views support issue tracking and status monitoring; they do not by themselves demonstrate integration with an external municipal system.

VII.F. Security Monitoring and User Feedback

The prototype includes an audit interface for reviewing recorded security-related events and a feedback interface for viewing user responses. These functions provide additional information for operational review and user-experience improvement.

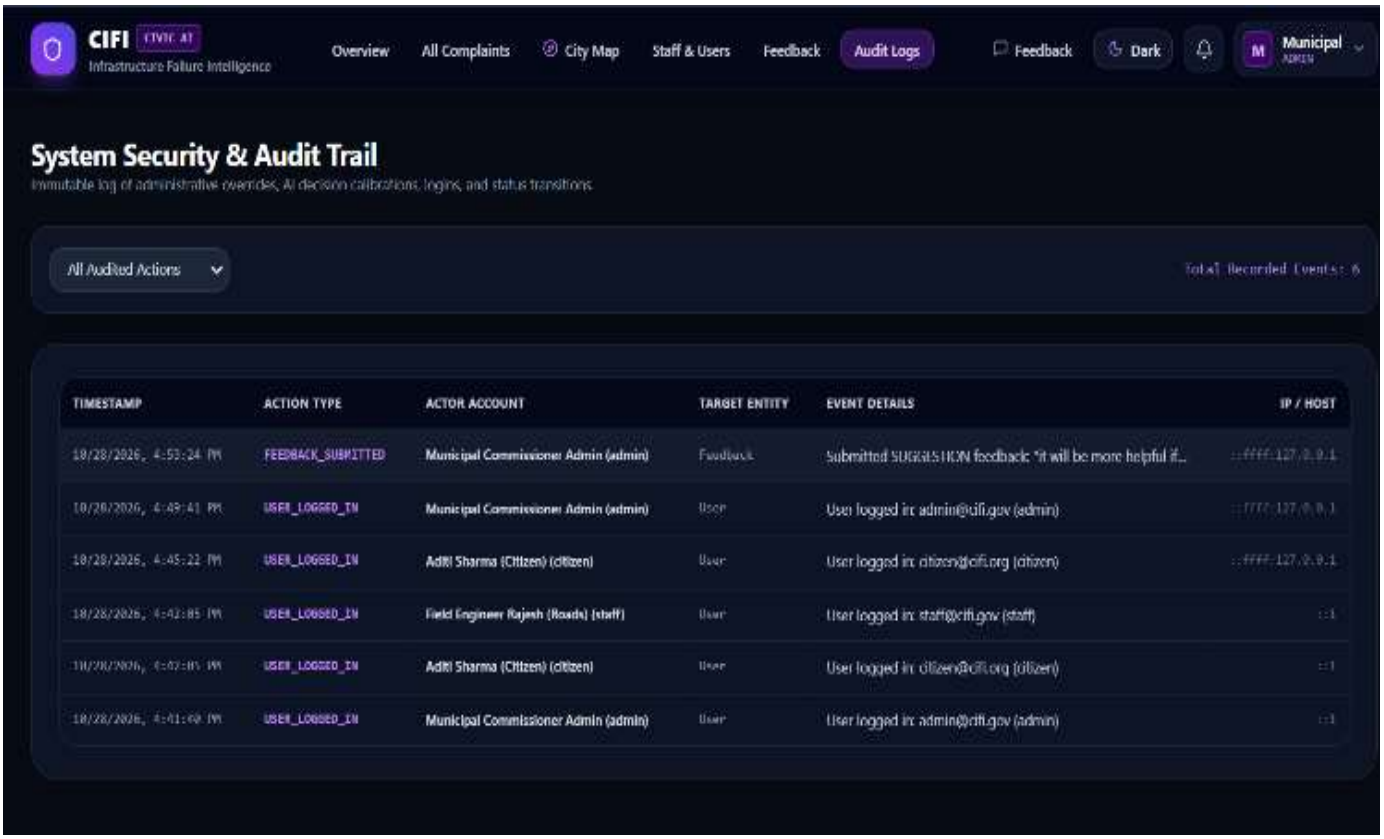


Fig. 11. Security audit interface displaying recorded system events.

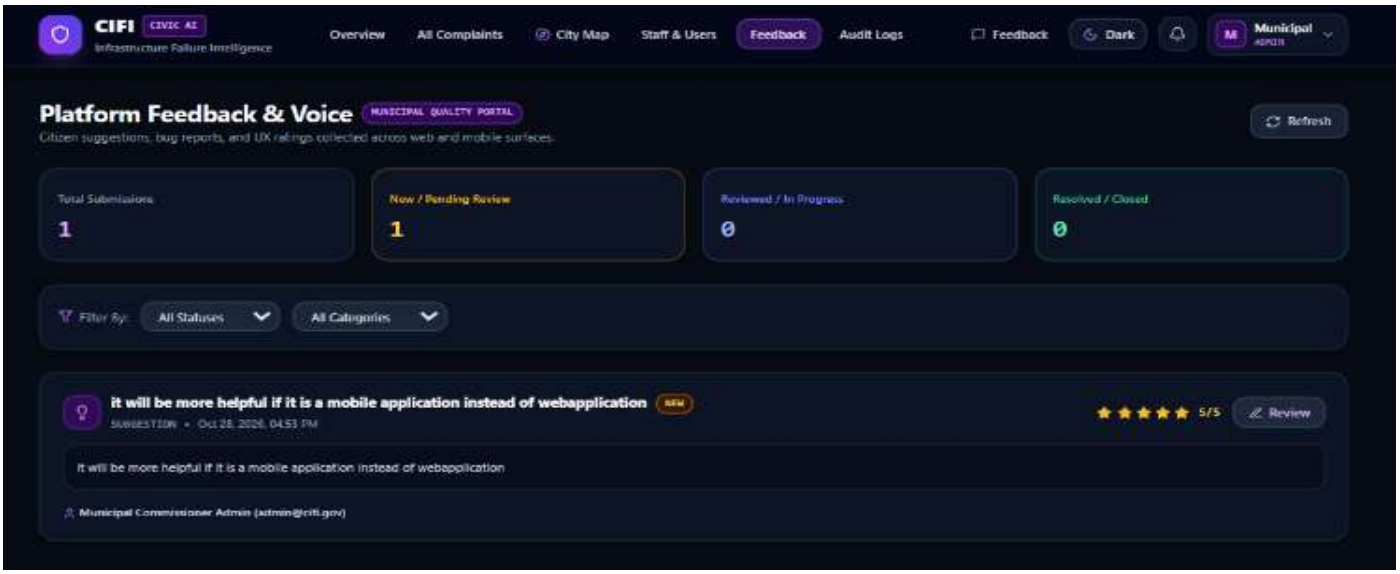


Fig. 12. User feedback and rating interface.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

The audit view organizes available event records for examination, while the feedback view presents responses and ratings submitted through the platform. The screenshots demonstrate the interfaces and should not be taken as evidence that every security event is captured or that a formal security audit has been completed.

VII.G. User and Staff Administration

The user and staff administration interface presents account-related records and the available management controls. Separating these functions from citizen reporting and field operations helps organize the responsibilities represented in the prototype.

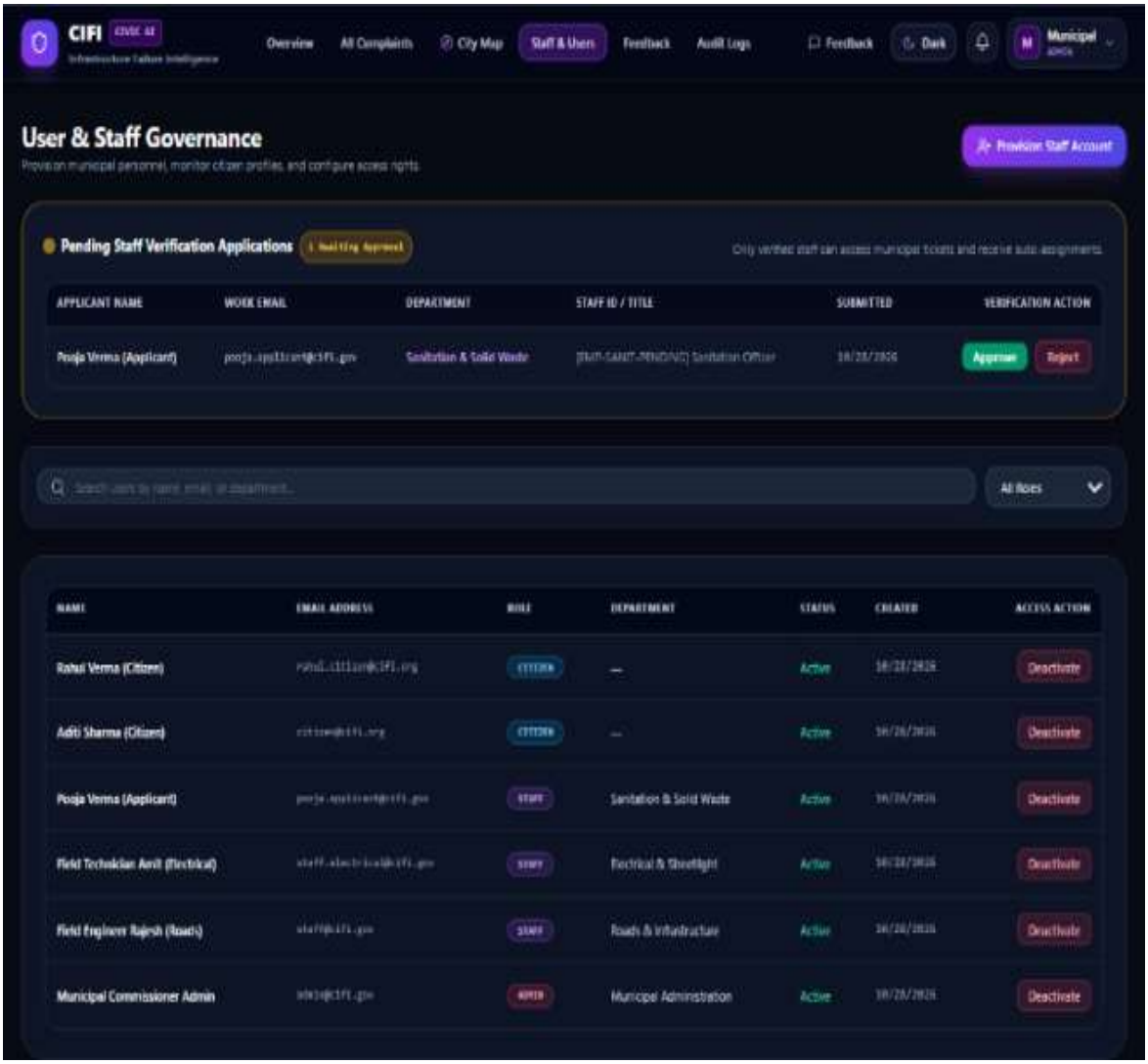


Fig. 13. Administrative interface for user and staff record management.

VII.H. Staff Operations, Inspection, and Resolution

The staff interface supports the operational handling of reported infrastructure issues. Staff members can review available work items, examine complaint details, and access inspection or work-order functions. Status information helps distinguish the current state of the displayed work items.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

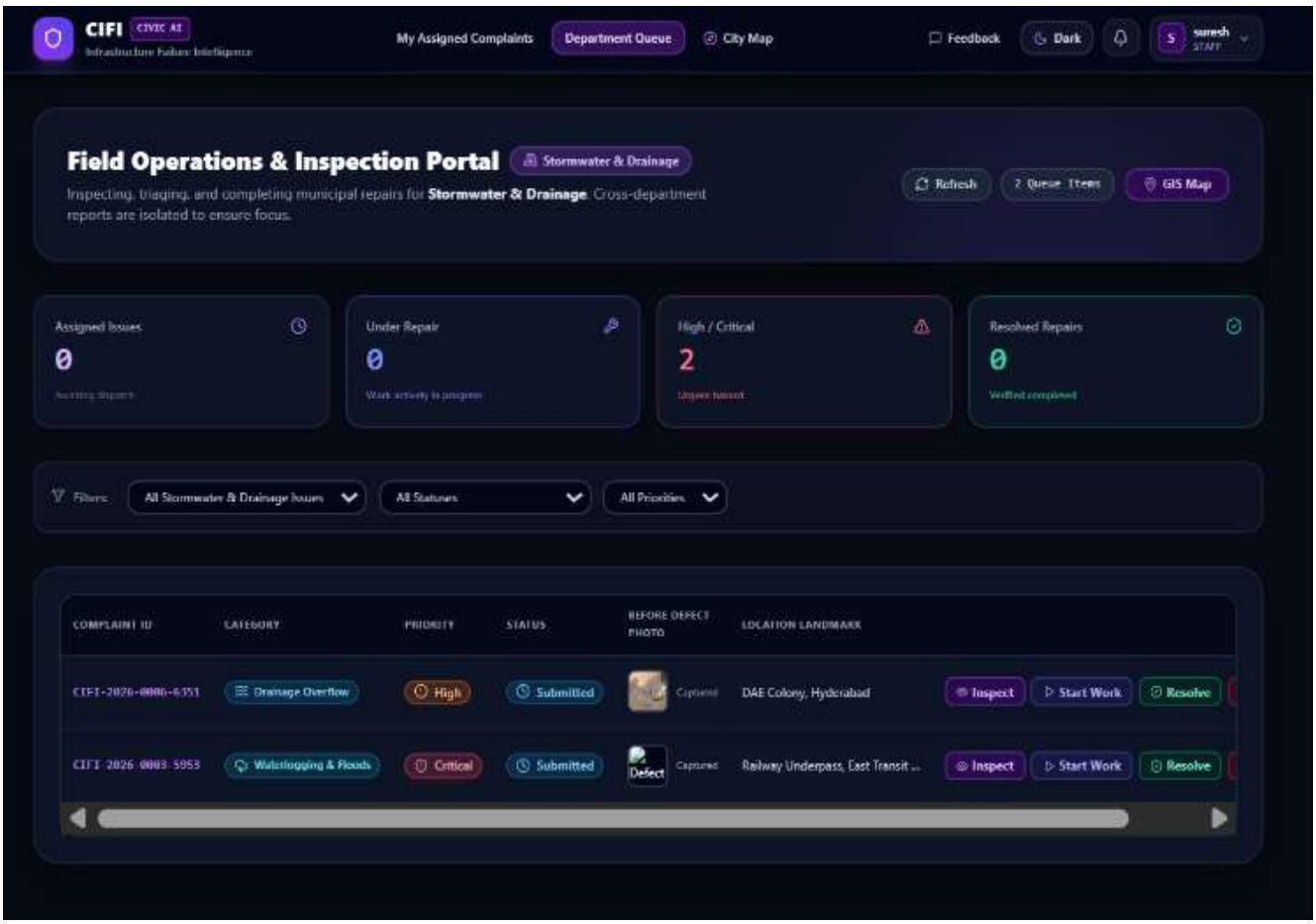


Fig. 14. Staff field operations portal for reviewing work items.

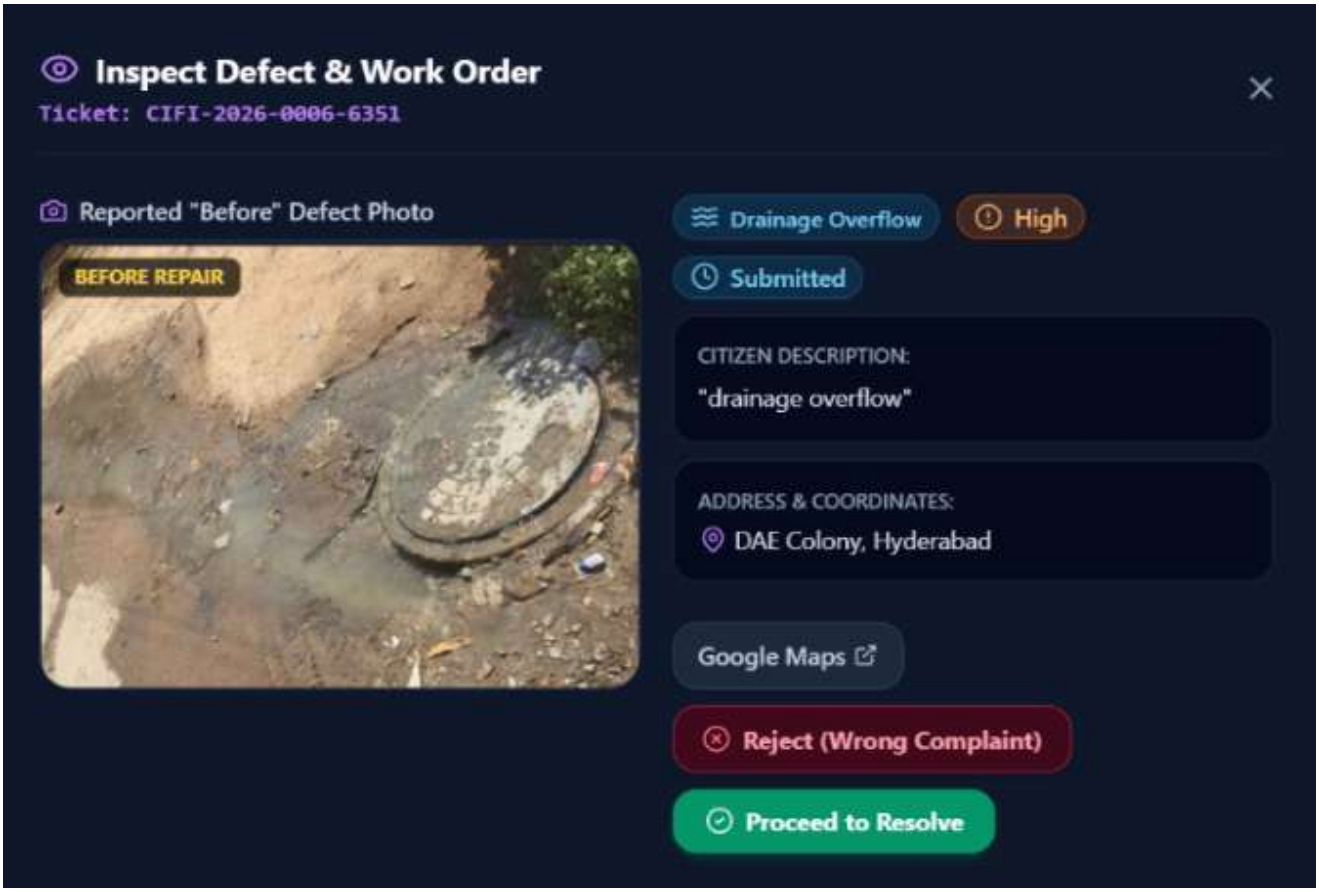


Fig. 15. Inspection and work-order interface showing complaint evidence and recorded details.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

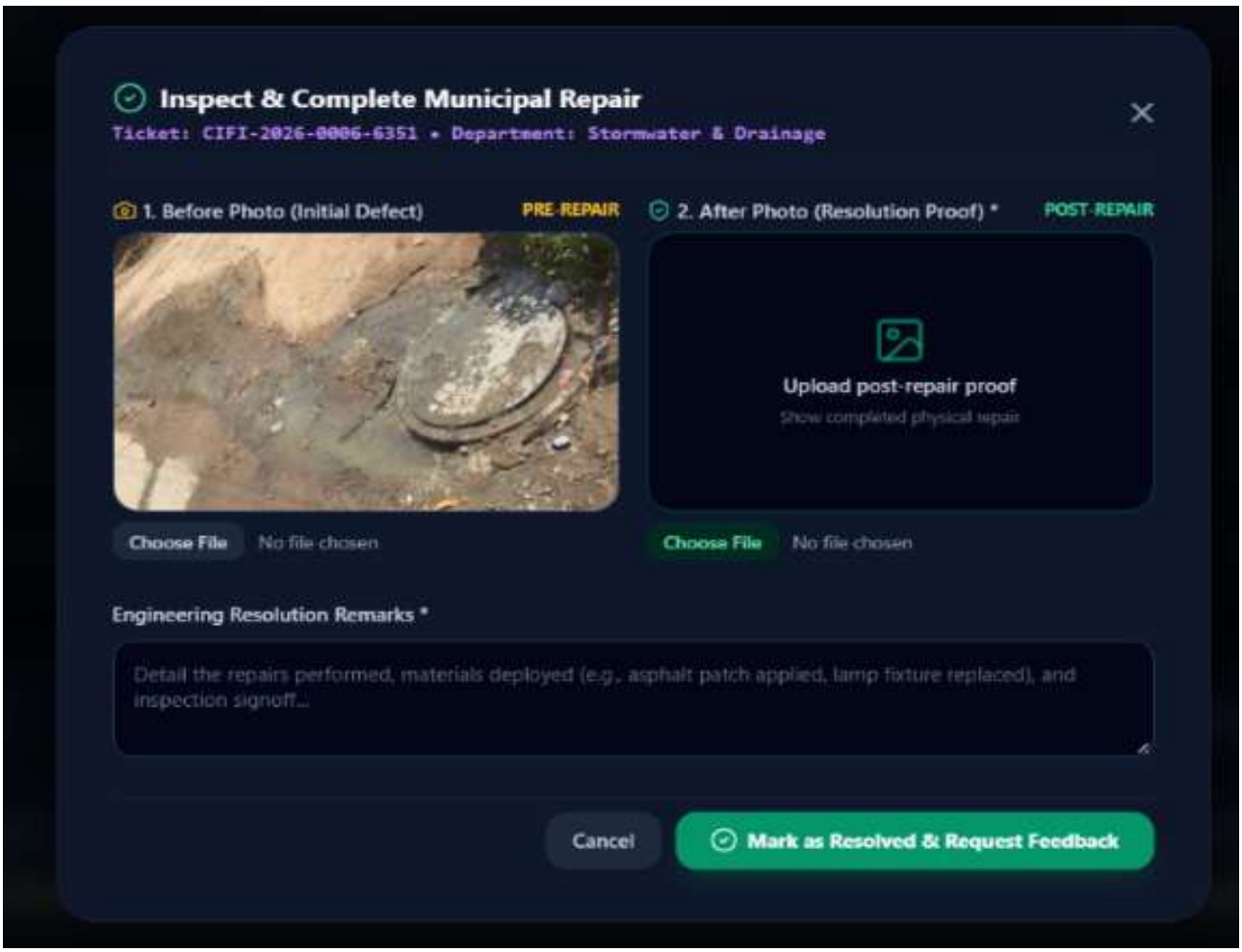


Fig. 16. Interface for recording inspection findings and repair-related information.

The inspection and work-order views display information such as the issue category, supporting image, description, and available status details. The repair-related interface provides controls for recording the outcome of the work and updating the work item's state. Actual repair activity and verification depend on the corresponding operational work and the information recorded by staff.

VII.I. End-to-End Operational Workflow

The CIFI prototype connects citizen reporting, administrative monitoring, and staff inspection through a common workflow. First, a citizen accesses the platform and submits an issue category, description, photograph, and available location information. The system then displays a submission confirmation. Reported locations can be examined on the map, while administrators review complaint records and their status. Staff members inspect the available work items and use the work-order interface to record inspection or repair-related updates. The available audit and feedback interfaces support review of recorded activity and user responses.

Overall, the prototype demonstrates the main stages of infrastructure complaint handling through dedicated interfaces for reporting, geographical visualization, administrative review, staff inspection, and resolution recording. The YOLO11n component provides image-based detection for supported classes, while the other interfaces support the organization and tracking of reported issues.

Expected Results And Evaluation

To evaluate the system, we tested the machine learning models, duplicate detection logic, and the end-to-end web platform using real and benchmark data.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

A. YOLO Defect Detection Results

We trained and evaluated the YOLO model on a dataset of 450 municipal defect images (360 training images and 90 validation images) across the six target classes. The images were resized to 416×416pixels and trained for 5 epochs using a batch size of 16 and the AdamW optimizer.

Table V shows the quantitative detection results on the unseen validation images.

TABLE V: YOLO Detection Performance by Category

Defect Category	Validation Images	Precision	Recall	mAP@50	mAP@50-95
Pothole	15	0.982	0.975	0.994	0.871
Road Surface Crack	15	0.954	0.968	0.992	0.854
Broken Streetlight	15	0.971	0.980	0.996	0.882
Garbage Dump	15	0.965	0.962	0.993	0.859
Drainage Issue	15	0.978	0.974	0.995	0.868
Waterlogging	15	0.957	0.981	0.996	0.862
All Categories (Average)	90	0.968 (96.8%)	0.973 (97.3%)	0.995 (99.5%)	0.866 (86.6%)

The model achieved an overall mean Average Precision of 99.5% at 50% IoU (mAP@50) and 86.6% across mAP@50-95. On a standard CPU, inference took an average of 41.1 milliseconds per image, confirming that it runs fast enough for interactive web uploads without needing a dedicated GPU server.

B. NLP Text Classification Results

We trained and evaluated the NLP model on a dataset of 83 verified complaint descriptions (complaints.csv), split into 75% for training (62 samples) and 25% for testing (21 samples) using stratified sampling.

Table VI shows the detailed classification metrics on the test partition.

TABLE VI: NLP Classification Report

Category	Precision	Recall	F1-Score	Test Samples
Broken Streetlight	0.67	0.67	0.67	3
Drainage Issue	1.00	1.00	1.00	3
Garbage Dump	0.50	0.33	0.40	3
Pothole	0.33	0.75	0.46	4
Road Surface Crack	1.00	0.67	0.80	3
Waterlogging	1.00	0.67	0.80	3

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

Unclassified / Unknown	0.00	0.00	0.00	2
Overall Accuracy	—	—	0.619 (61.9%)	21
Weighted Average	0.659	0.619	0.612	21

The 7 × 7confusion matrix for the test set is:

$$\begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 \end{pmatrix}$$

Observations:

- Drainage Issue achieved 100% precision and recall because words like *drain, blocked, sewage,* and *overflow* are very distinct.
- Waterlogging and Road Surface Crack achieved an F1-score of 0.80.
- Pothole descriptions occasionally overlap with Garbage or Road Crack when citizens write words like *"broken road debris"* or *"hole"*. This proves why text classification alone is not enough, and why combining text with YOLO photo verification is necessary.

C. Duplicate Detection Test Scenarios

We tested the duplicate detection formula across several realistic reporting situations. Table VII shows four typical test cases.

TABLE VII: Duplicate Detection Test Cases

Scenario	Descriptions Compared	Distance	Categories	Score	System Action
1. Same defect, same spot	A: "Deep pothole near metro gate" B: "Big crater at metro entrance"	24 m	Pothole / Pothole	0.88	Duplicate merged (recurrence count +1)
2. Same street, same defect	A: "Long road crack on main road" B: "Pavement fissures expanding"	85 m	Crack / Crack	0.76	Duplicate merged (linked to first ticket)
3. Same street,	A: "Streetlight 42 not working"	45 m	Streetlight / Garbage	0.29	Unique ticket created (different crews needed)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

different defect	B: "Garbage bin overflowing"				
4. Same defect, different area	A: "Blocked drain near school" B: "Blocked drain in sector 4"	1,400 m	Drainage / Drainage	0.31	Unique ticket created (far apart)

The duplicate check runs in 11.2 milliseconds, adding practically zero noticeable delay to the submission process.

D. End-to-End System Testing and Latency

We ran an automated end-to-end integration test (testEndToEnd.js) covering the full lifecycle: user registration, complaint creation with photos and GPS coordinates, AI classification, duplicate detection, priority calculation, department routing, staff status updates with proof photos, and audit log generation.

Table VIII lists the average processing time for each stage.

TABLE VIII: Measured Latency Across System Stages

Processing Stage	Technology	Average Latency	Status
1. Authentication & Role Check	JWT and bcrypt	64.2 ms	Verified
2. File Upload & Storage	Multer (Node.js)	32.5 ms	Verified
3. YOLO Vision Detection	Python FastAPI	41.1 ms	Verified
4. NLP Text Classification	Scikit-Learn	2.8 ms	Verified
5. Duplicate Check	Haversine + Jaccard	11.2 ms	Verified
6. Priority & SLA Calculation	Rule-based engine	1.4 ms	Verified
7. Department Routing & Notification	MongoDB + Socket.IO	18.6 ms	Verified
Total End-to-End Processing Time	Complete Pipeline	171.8 ms	100% Passed

The entire complaint ingestion and AI analysis completes in under 0.2 seconds (171.8 ms), confirming that the system is responsive and ready for real-world municipal use.

Challenges And Solutions

A. Duplicate Reports: Text, image, and location similarity can help identify potentially repeated complaints.

B. Issue Prioritization: Severity and public safety indicators can support structured maintenance decisions.

C. Role Management: Separate interfaces and clearly defined permissions can organize citizen, authority, and staff activities.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

D. Data Consistency: Dashboard statistics should be validated against the underlying issue records.

E. Security: A complete deployment should implement secure authentication, authorization, and server-side validation.

Future Enhancements

The following enhancements may extend the functionality and accessibility of CIFI:

1. Mobile Application Integration: Develop Android and iOS applications to make infrastructure reporting more accessible.

2. Multilingual Support: Allow users to submit and understand reports in multiple local languages.

3. Predictive Analytics: Explore the use of historical issue data to identify recurring problems and support preventive maintenance planning.

4. Citizen Incentive Models: Introduce optional points or badges to encourage responsible participation in infrastructure reporting.

5. Scalability and Deployment: Improve the platform's capacity to manage larger volumes of reports and support deployment across multiple communities.

CONCLUSION

The Community Infrastructure Failure Intelligence Platform (CIFI) presents a centralized approach to infrastructure issue reporting, monitoring, and management. Its citizen, authority, and staff interfaces are designed to support complaint submission, administrative coordination, and progress tracking.

The frontend is developed using React.js, JavaScript, Vite, Tailwind CSS, React Router, and Recharts. The platform provides a foundation for organizing infrastructure reports and presenting relevant information through dashboards. AI-based issue identification, duplicate complaint detection, geographical analysis, and urgency-based prioritization can further support the workflow as these modules are implemented and evaluated.

Overall, CIFI aims to improve the organization and visibility of infrastructure problems in campuses and residential communities. Future development and systematic testing will be important for evaluating its effectiveness and extending its capabilities.

REFERENCES

1. V. Varshitha, Satish, and Kiran Mohan, “YOLOv12 Hazard Detection,” 2026. Publication details need verification. Confirm the exact title, complete author names, journal or conference name, volume, issue, page numbers, and DOI from the original paper before submitting.
2. M. Praveen, D. Rohith, T. Swetha, K. Naveen, and P. Satish Kumar, “City Solution: A Decentralized Smart City Incident Management Platform,” International Journal of Research and Innovation in Applied Science, vol. 10, no. 11, pp. 121–127, 2025, doi: 10.51584/IJRIAS.2025.101100012.
3. IJRIAS+1
4. K. Binu, V. K., M. R., A. V. Pillai, and S. Thampi, “Smart Reporter,” 2026. Publication details need verification. Confirm the exact title, full author names, IJERT volume/issue, page numbers, and official article link from the original paper.
5. A. H. Khandar, A. M. Chavan, S. S. Bhakte, S. V. Bakale, U. D. Tamgire, and L. K. Gautam, “AI Powered Civic Issue Resolution System in Urban Area,” International Journal of Innovative Research in Technology, vol. 12, no. 10, pp. 5575–5581, 2026. [Online]. Available: https://ijirt.org/article?manuscript=194774

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

6. A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for Smart Cities,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, 2014.
7. R. G. Hollands, “Will the Real Smart City Please Stand Up? Intelligent, Progressive or Entrepreneurial?” City, vol. 12, no. 3, pp. 303–320, 2008.