CIVIC CONNECT

A Report

Submitted in partialfulfillment of the

Requirements for the completion of

THEME BASED PROJECT

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

P.LAHARI	**1602-22-737-147**
V.MADHURASRI	**1602-22-737-148**
T.SANJAY	**1602-22-737-172**

Under the guidance of

Sathyadevi Maranganti,

Radha

ASSISTANT PROFESSOR

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE.

(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

2025

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

Department of Information Technology

DECLARATION BY CANDIDATES

We, **LAHARI, MADHURASRI, SANJAY,** bearing hall ticket numbers, **1602-22-737-147, 1602-22-737-148, 1602-22-737-172,** hereby declare that the project report entitled **"CIVIC CONNECT"** under the guidance of **Ms. M. SATHYA DEVI, Assistant Professor**, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfillment of the requirement for the completion of Theme-based project, VI semester, Bachelor of Engineering in Information Technology.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other institutes.

P.LAHARI
1602-22-737-147

V.MADHURASRI
1602-22-737-148

T.SANJAY
1602-22-737-172

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled "**CIVIC CONNECT**" being submitted by **,P.LAHARI, V.MADHURASRI, T.SANJAY** bearing hallticket numbers, **1602-22-737-147,1602-22-737-148,1602-22-737-172,** in partial fulfillment of the requirements for the completion of Theme-based project of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

Ms. M.Sathya Devi **Assistant Professor**	**External Examiner**	**Dr. K. Ram Mohan Rao** **Professor, HOD IT**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

It is with immense pleasure that we would like to take the opportunity to express our humble gratitude to **Ms.M.SathyaDevi, Assistant Professor, Information Technology** under whom we executed this project· Her constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depth and helped us to complete the assigned tasks.

We are very much thankful to **Dr.K.Ram Mohan Rao Professor, HOD IT and Ms. M.Sathya Devi,Information Technology,** for his kind support and for providing the necessary facilities to carry out the work.

We wish to convey our special thanks to Dr. S. V. Ramana**, Principal** of **Vasavi College of Engineering** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and our friends who had directly or indirectly helped and supported me in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities. Finally, we wish to convey our gratitude to our family who fostered all the facilities that we need.

ABSTRACT

CivicConnect is a comprehensive digital platform developed to enhance communication and collaboration between citizens and local governments by streamlining civic engagement and issue resolution. The platform addresses persistent challenges in civic administration—such as delays, inefficiencies, and lack of citizen participation—by offering a centralized system where users can easily report local issues with location data and media attachments.CivicConnect features real-time issue tracking, enabling citizens to monitor the progress of their reports through every step of the resolution process. Community forums allow residents to engage in meaningful discussions, suggest improvements, and vote on pressing concerns, promoting collective problem-solving and awareness. The platform integrates AI-powered prioritization, which analyzes reported data to help authorities identify and respond to the most urgent or widespread problems effectively.To further enhance public involvement, CivicConnect includes gamification elements such as points, badges, and leaderboards, motivating users to participate actively. Additionally, the platform offers e-governance integration, providing streamlined access to public services like permits, taxes, and electoral information.Through a user-friendly interface and intelligent technology, CivicConnect strengthens civic responsibility, enables data-driven governance, and ultimately contributes to better community outcomes.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1 Overview

1.2 Problem Statement

1.3 Motivation of theme & title

2. LITERATURE SURVEY 4

3. EXISTING SYSTEM 6

4. PROPOSED SOLUTION 7

4.1. System Design

4.1.1 Architecture Diagram

4.1.2 Use-Case, Sequence and Activity Diagrams

4.1.2.1 Use-case descriptions

4.2.1 Screenshots & Pseudocode

5. EXPERIMENTAL SETUP & IMPLEMENTATION 66

5.1 System Specifications

5.1.1 Hardware Requirements

5.1.2 Software Requirements

5.2 Methodology/Algorithm

6. RESULTS 68

7. CONCLUSION & FUTURE SCOPE 69

8. REFERENCES 70

S.No.	FigNumber	Figure Name	
1	Fig 4.1.1	Architectural Design of CivicConnect Platform	8
2	Fig 4.1.2	Use-Case Diagram	9
3	Fig 4.1.3	Sequence and Activity Diagram	15
4	Fig 4.2.1.1	Dashboard	17
5	Fig 4.2.1.2	Citizen Registration Page	18
6	Fig 4.2.1.3	Home Page of CivicConnect(Citizen View)	18
7	Fig 4.2.1.4	Issue Reporting Form with Location and Image Upload	20
8	Fig 4.2.1.5	Notification Page Showing Issue Status Update	21
9	Fig 4.2.1.6	Authority Login Page	22
10	Fig 4.2.1.7	Authority Dashboard for Managing Issues	23
11	Fig 4.2.1.8	AI-based Severity Prediction Output	24
12	Fig 4.2.1.9	Trending Issues	24
13	Fig 4.2.1.10	Workers Login Page	25
14	Fig 4.2.1.11	Officers Dashboard Overview	25
15	Fig 4.2.1.12	Issue Status Update	27

LIST OF TABLES

1) 4.1.2.1.1 - Use Case 1 10
2) 4.1.2.1.2 - Use Case 2 10
3) 4.1.2.1.3 - Use Case 3 11
4) 4.1.2.1.4 - Use Case 4 11
5) 4.1.2.1.5 - Use Case 5 11
6) 4.1.2.1.6 - Use Case 6 12
7) 4.1.2.1.7 - Use Case 7 12
8) 4.1.2.1.8 - Use Case 8 12
9) 4.1.2.1.9 - Use Case 9 13
10) 4.1.2.1.10 - Use Case 10 13
11) 4.1.2.1.11- Use Case 11 13
12) 4.1.2.1.12 – Use Case 12 14

LIST OF ACRONYMS:

Acronym	Full Form
AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
CSRF	Cross-Site Request Forgery
URL	Uniform Resource Locator
WSGI	Web Server Gateway Interface

1. INTRODUCTION

1.1 Overview:

Civic Connect is a modern civic engagement platform that bridges the gap between citizens and local authorities by providing tools for efficient issue reporting, public dialogue, and digital access to government services.

The platform allows users to quickly report civic problems—such as road damage, water supply issues, or waste management concerns—by submitting descriptions, photos, and location data. These issues are then tracked in real time, ensuring citizens are kept informed throughout the resolution process.

Key features include:

AI-powered prioritization to help authorities manage resources based on urgency and frequency of issues.

Gamification to reward active users and foster sustained participation through badges, points, and leaderboards.

1.2 Problem statement:

Civic engagement in many communities remains fragmented, with citizens often facing significant barriers when attempting to communicate local issues to the appropriate authorities. Traditional reporting mechanisms—such as phone calls, paper forms, or disconnected digital channels—frequently result in delayed responses, miscommunication, and a general lack of transparency. This inefficiency undermines public trust and satisfaction, while also placing strain on government resources.

Furthermore, the absence of a centralized platform for reporting, prioritizing, and tracking civic issues prevents both citizens and local governments from operating in a structured, data-driven manner. Without real-time updates or intelligent prioritization, critical problems may go unresolved, and minor concerns may consume unnecessary attention. This not only slows down the issue resolution process but also limits the ability of authorities to allocate resources effectively and respond proactively to community needs.

There is a clear need for an integrated digital solution that streamlines communication between citizens and local authorities, facilitates real-time tracking of civic issues, and empowers both parties with the tools to manage and resolve concerns in a transparent, organized, and engaging way.

1.3 Motivation to use Civic connect:

Traditional methods for reporting civic issues are often outdated and inefficient, resulting in delays in responses and problem resolution. This lack of responsiveness leads to citizen frustration, as people are left without real-time updates or visibility into how their concerns are being addressed, making them feel disconnected from local authorities. Additionally, rapid urbanization has intensified civic challenges, demanding faster, smarter, and more scalable solutions. In this context, there is a growing need for transparency and accountability, with citizens increasingly expecting clear insights into the resolution process. These challenges present a significant opportunity for technology to step in—digital platforms can streamline issue reporting, enhance communication, improve response times, and provide real-time updates, ultimately bridging the gap between citizens and governing bodies.

1.2 LITERATURE SURVEY

Various methods and technologies have been proposed to improve civic engagement and issue resolution through digital platforms. The integration of real-time tracking, AI-driven prioritization, and gamification has significantly contributed to more responsive and citizen-centered governance. Some of the major contributions are as follows:

In 2012, Linders et al. [1] introduced the concept of "citizen-sourcing," emphasizing the potential of digital platforms to involve citizens directly in public service delivery. This laid the foundation for participatory governance, where citizens are not just service recipients but active contributors to issue reporting and feedback.

In 2014, Goldsmith and Crawford [2] examined early implementations of digital civic tools like *SeeClickFix* and *FixMyStreet*. These platforms enabled citizens to report local issues through mobile and web interfaces. While impactful in raising civic awareness, they lacked mechanisms for prioritization or intelligent analysis of reported data.

In 2016, Janssen and Kuk [3] explored the use of Artificial Intelligence (AI) in public sector decision-making. They emphasized how AI algorithms could process large volumes of civic complaints and identify critical trends or urgent problems. This opened pathways for data-driven prioritization in digital civic platforms.

In 2018, Kumar and Patel [4] developed an integrated web portal for urban service complaints, built using relational databases and basic categorization logic. Although the platform streamlined issue submission and department routing, it lacked dynamic features like real-time updates or citizen engagement analytics.

In 2019, Gupta et al. [5] proposed an AI-based civic issue classifier that used natural language processing (NLP) to analyze user-submitted complaints. Their model categorized issues into departments and flagged high-frequency or critical complaints. The study demonstrated how machine learning could reduce administrative workload and improve response speed.

In 2021, Martins et al. [6] introduced gamification techniques in a civic participation platform to increase user retention and motivation. Users earned points and badges for reporting issues and participating in community discussions. The approach significantly boosted user engagement and encouraged long-term civic involvement.

In 2023, Thomas and Iyengar [7] conducted a comparative analysis of municipal issue-reporting platforms in developing countries. They concluded that while many platforms offered reporting functionality, most lacked intelligent features such as automated tracking, prioritization, and user interaction tools. They called for more holistic systems that combine accessibility with smart analytics.

In 2024, Tanisha Roy et al. [8] presented a real-time civic dashboard model that visualized trends in citizen complaints using heatmaps and charts. Their system helped authorities identify problem-prone areas and optimize field operations accordingly. Though effective for city-level analysis, the system did not include user-facing components for engagement or feedback.

1.3 EXISTING SYSTEM

In the realm of civic engagement and issue reporting, several digital platforms have been introduced by municipal governments and private developers to improve communication between citizens and local authorities. Notable examples include **SeeClickFix**, **FixMyStreet**, and **Swachhata App (India)**. These applications allow users to report civic issues such as potholes, garbage collection problems, or streetlight malfunctions via mobile or web interfaces. The reports are typically routed to the appropriate municipal department for further action.

While these platforms have made significant strides in enabling citizens to voice their concerns more easily, they also exhibit several limitations:

Fragmented Systems: Many existing platforms are localized or department-specific, lacking a unified structure to handle all types of civic issues across departments.

No Real-Time Updates: Most systems do not offer real-time tracking of the issue status, leaving users unaware of progress or resolution timelines.

Manual Prioritization: Issues are often reviewed and prioritized manually by officials, leading to delays in addressing urgent problems.

Limited Citizen Engagement: Existing platforms often do not include community discussion forums, voting features, or feedback loops, reducing citizen participation and sense of impact.

Lack of Data Intelligence: There is little to no use of AI or data analytics to detect patterns in civic complaints, which could otherwise help authorities allocate resources more efficiently.

Due to these limitations, there is a pressing need for a more integrated, intelligent, and engaging civic issue management platform. **Civic Connect** aims to address these gaps by offering features such as real-time tracking, AI-based prioritization, and a user-centric interface that encourages active citizen participation and data-driven governance.

1.4 PROPOSED SOLUTION

To address the shortcomings of traditional civic issue reporting systems, **CivicConnect** proposes a smart, centralized platform that enables efficient communication between citizens and local authorities. The platform focuses on fast, transparent issue reporting and intelligent incident management to ensure timely resolution and improved civic infrastructure. Key components of the proposed solution include:

Real-Time Issue Reporting and Tracking: Users can report civic issues by submitting descriptions, photos, and location data via a simple user interface. Once submitted, the status of each report is updated in real-time, allowing citizens to stay informed throughout the resolution process.

AI-Powered Incident Prioritization: The platform incorporates AI/ML models to automatically analyze and categorize reported issues. It identifies high-priority problems based on severity, frequency, and location, enabling authorities to respond to the most urgent cases first.

Trending Issues Dashboard: CivicConnect features a dynamic dashboard that highlights frequently reported issues. This helps administrators identify problem areas, monitor patterns, and allocate resources more effectively.

Geolocation Integration: Using tools like Google Maps API or OpenStreetMap, the platform accurately maps the location of reported issues. This supports faster field response and eliminates ambiguity in locating complaints.

By eliminating the delays and inefficiencies of manual and fragmented systems, **CivicConnect** ensures a data-driven, responsive approach to civic management. The platform improves transparency, enhances accountability, and fosters public trust by making civic participation accessible and action-oriented.

4.1. System Design:

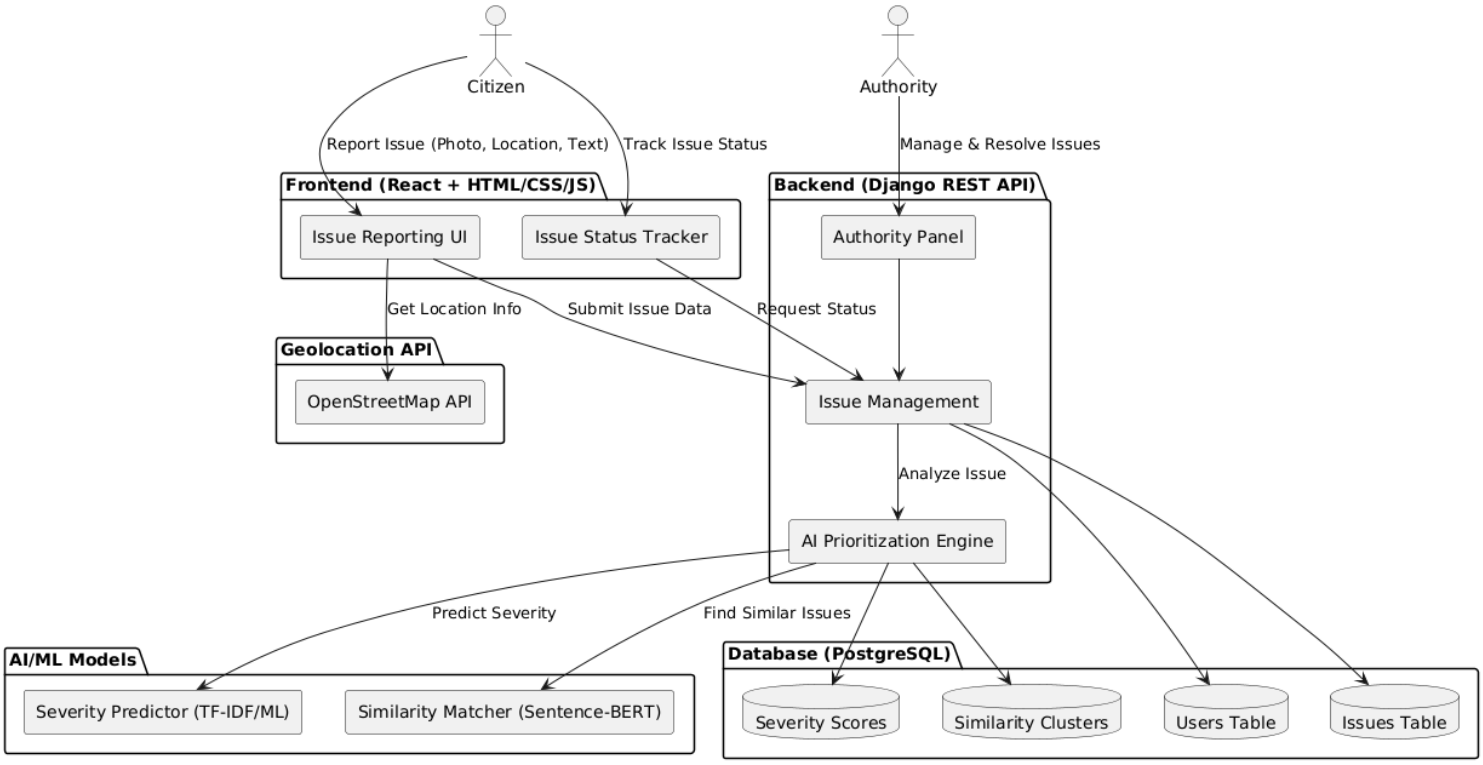


Fig 4.1.1. Architectural Design of Civic Connect

4.1.2 Use-Case Diagrams:

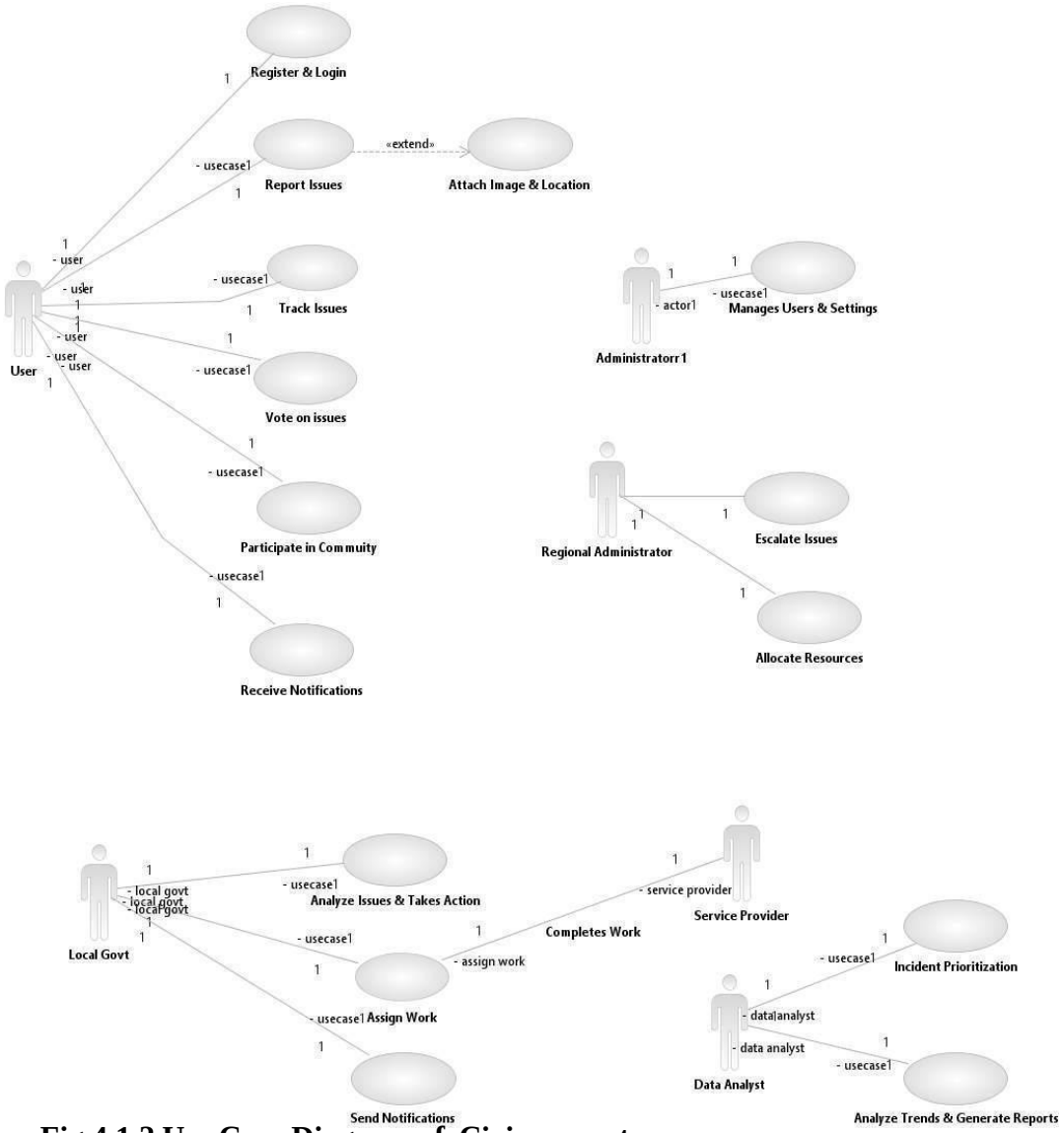


Fig 4.1.2 Use Case Diagram of Civicconnect

4.1.2.1 Use-Case Descriptions:

Use Case ID: UC01

Name: Register & Login
Actors: User
Description: The user registers on the platform and logs in with credentials to access personalized features.

User Actions	System Actions
1. Enters registration details.	1.Stores user information in the database
2. Creates credentials.	2. Authenticates and redirects to dashboard
3. Logs in using credentials.	

Table 4.1.2.1.1

Use Case ID: UC02

Name: Report Issues
Actors: User
Description: The user reports a civic issue by submitting a description, location, and image.

User Actions	System Actions
1. Opens the report issue form.	1.Stores user information in the database with metadata
2. Enters issue description	
3. Attaches image and location.	

Table 4.1.2.1.2

Use Case ID: UC03

Name: Track Issues
Actors: User
Description: Users can track the status of the issues they reported.

User Actions	System Actions
1. Opens the track issues page	1. Displays current status from the database.
2. Selects an issue.	

Table 4.1.2.1.3

Use Case ID: UC04

Name Receive Notifications
Actors: User
Description: Users receive real-time notifications about their reported issues.

User Actions	System Actions
	Sends notifications when issue status changes

Table 4.1.2.1.4

Use Case ID: UC05

Name: Manage Users & Settings
Actors: Administrator
Description: Administrator manages user roles, accounts, and system settings.

User Actions	System Actions
1. Opens Admin Panel.	1 Displays user and system configuration settings.
2. Updates user data/settings	2.Saves changes in the database.

Table 4.1.2.1.5

Use Case ID: UC06

Name :Escalate Issues
Actors: Regional Administrator
Description: Escalates unresolved or urgent issues to higher authorities.

User Actions	System Actions
1. Reviews pending issues.	1 Updates issue status in the database
2. Marks as escalated.	

Table 4.1.2.1.6

Use Case ID: UC07

Name : Allocate Resources
Actors: Regional Administrator
Description: Assigns resources or teams for specific issues based on urgency.

User Actions	System Actions
1. Selects issue needing resources	Logs assignment in the system.
2.Assigns field team.	

Table 4.1.2.1.7

Use Case ID: UC08

Name : Analyze Issues & Take Action
Actors : Local Government
Description: Reviews issues and initiates appropriate actions

User Actions	System Actions
Views issues.	Displays reported issue list.

Table 4.1.2.1.8

Use Case ID: UC09

Name : Assign Work
Actors: Local Government
Description: Assigns issues to service providers for resolution.

User Actions	System Actions
1. Selects issue.	Creates assignment record in the database .
2. Assigns service provider.	

Table 4.1.2.1.9

Use Case ID: UC10

Name : Complete Work
Actors: Service Provider
Description: Completes assigned tasks and updates issue status.

User Actions	System Actions
1 Views assigned task.	Updates status in the system.
2. Marks task as complete.	

Table 4.1.2.1.10

Use Case ID: UC11

Name : Incident Prioritization
Actors: Data Analyst
Description Uses AI-based tools to prioritize incidents based on various parameters.

User Actions	System Actions
1 Runs prioritization tool.	Analyzes data using AI model.
2. Views sorted issue list.	Outputs prioritized list.

Table 4.1.2.1.11

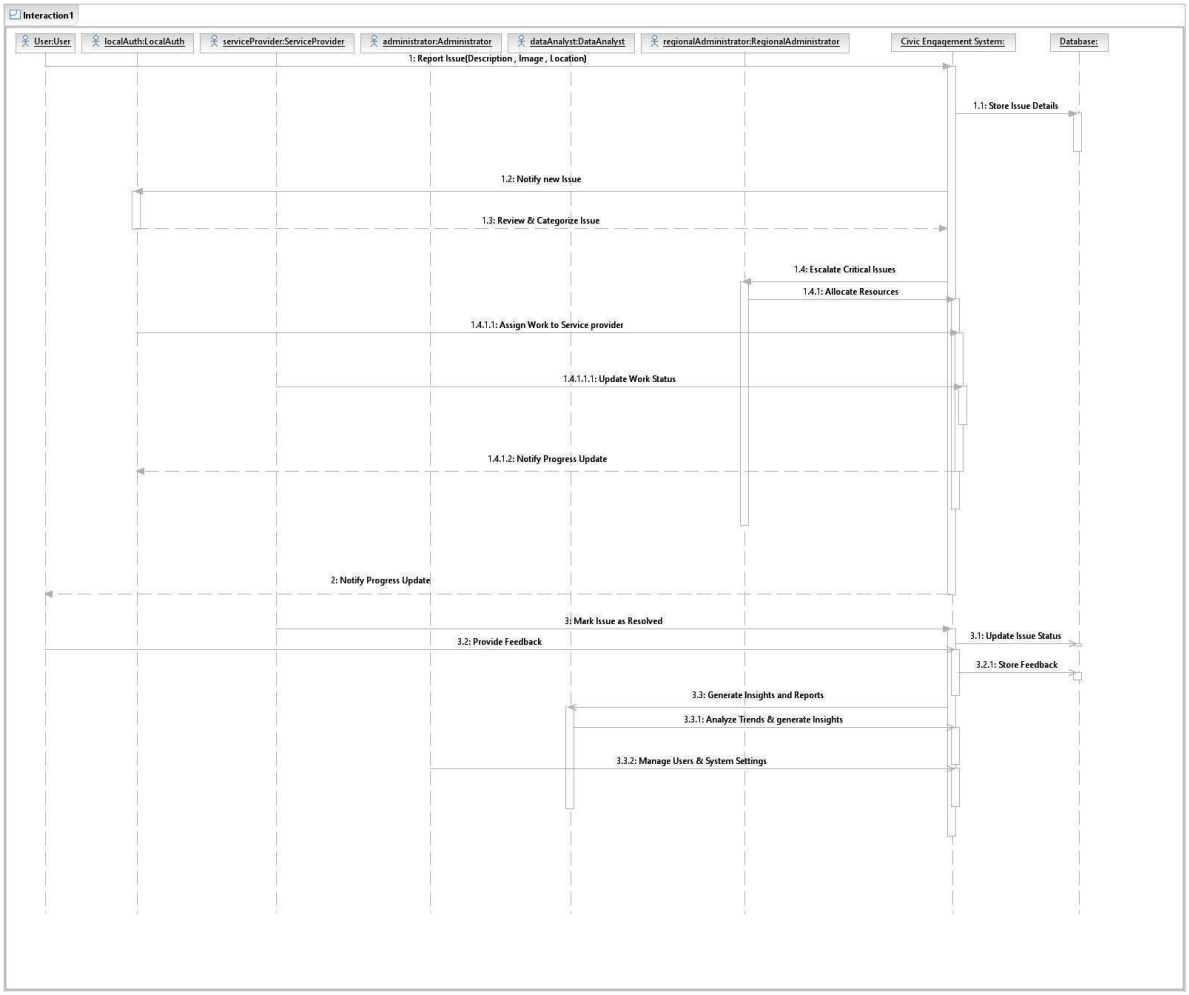
Use Case ID: UC12

Name : Analyze Trends & Generate Reports
Actors: Data Analyst
Description: Analyzes civic data to detect trends and create performance reports.

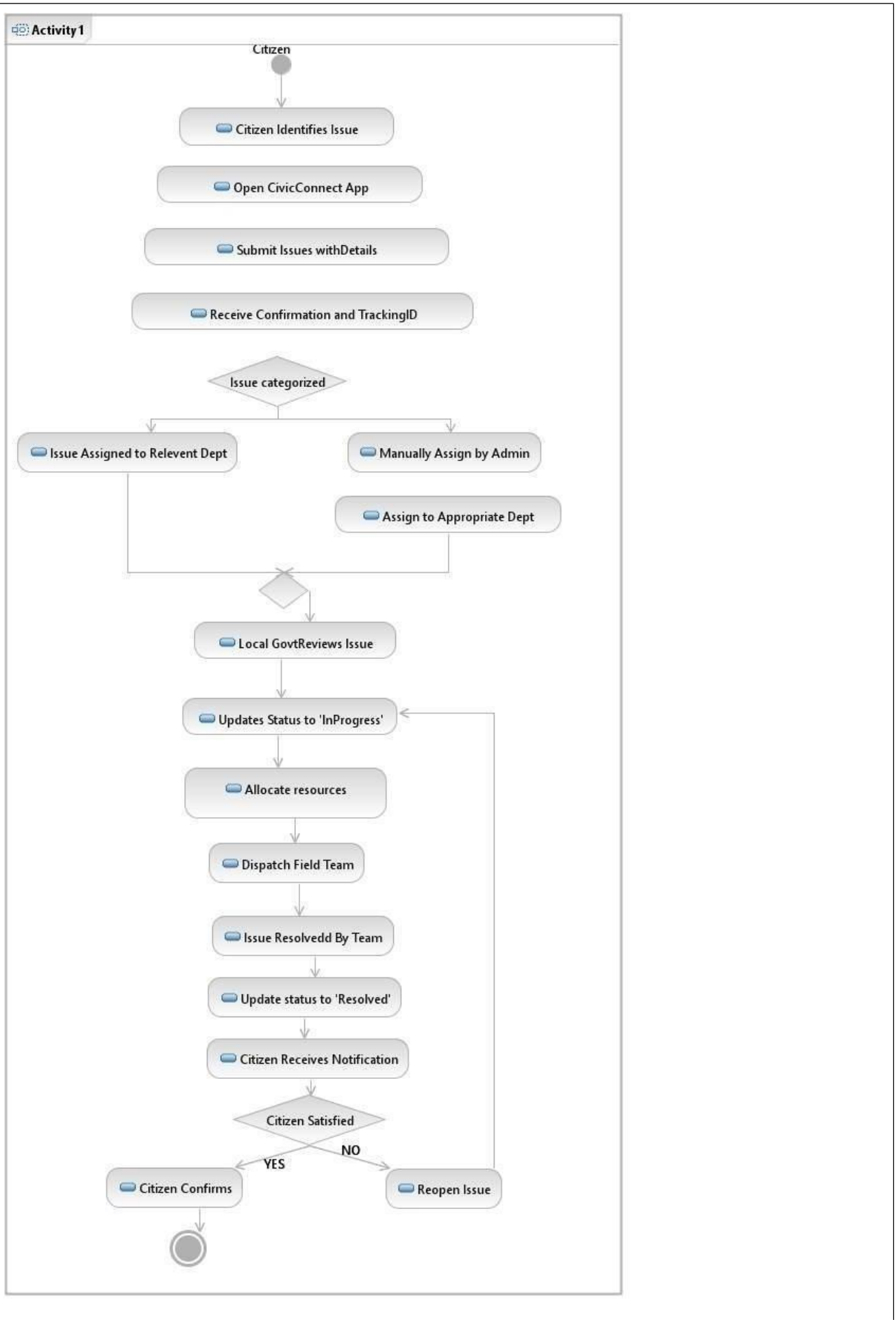
User Actions	System Actions
1.Runs analysis tool	Generates charts, reports from the database.
2. Downloads report.	Provides data export.

Table 4.1.2.1.12

4.1.3 Sequence Diagram



4.1.3 **Activity Diagram**



4.2 Functional Modules:

4.2.1. Screenshots and Code:

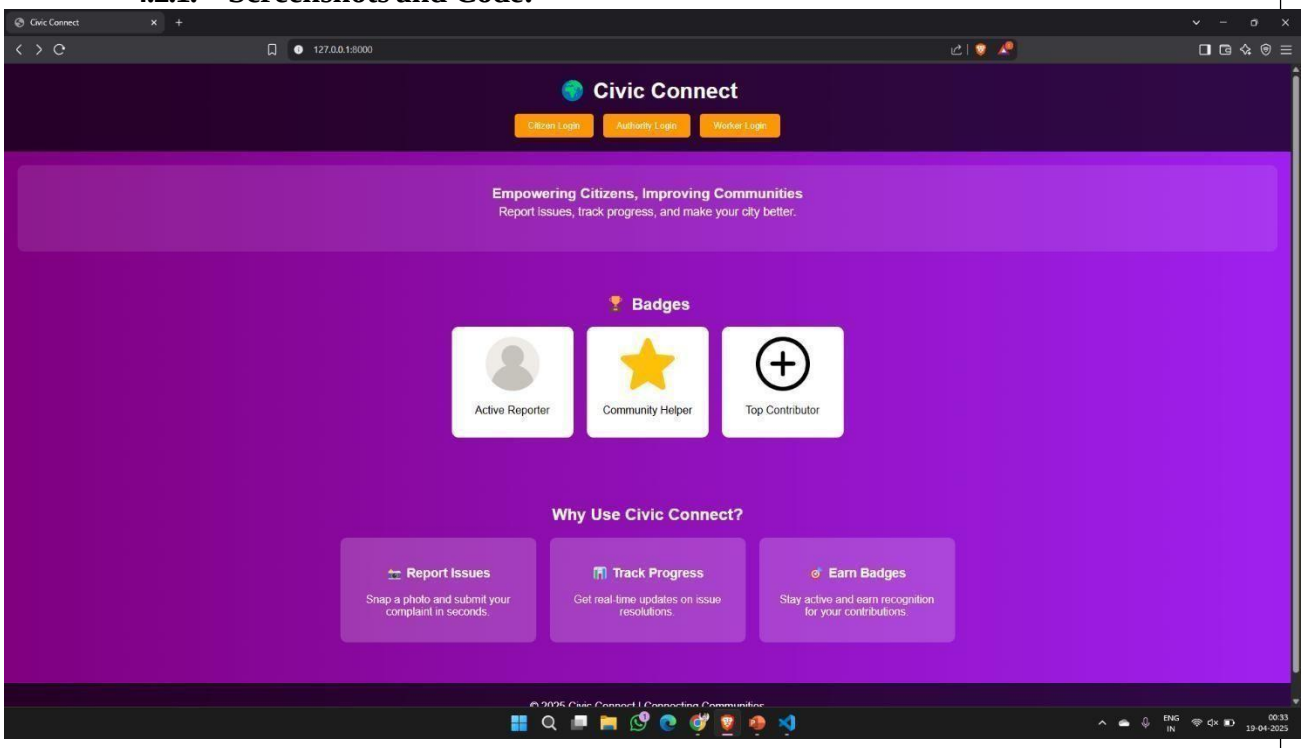
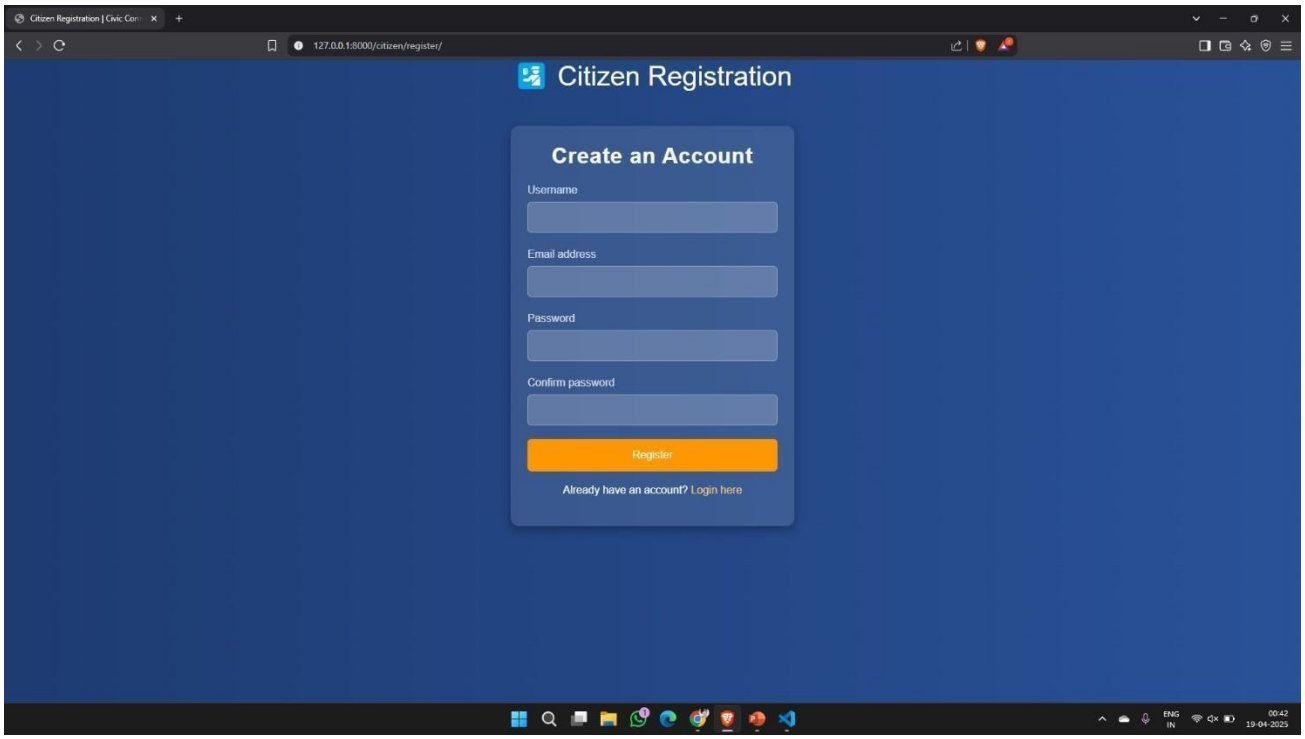


FIG 4.2.1.1: Dashboard



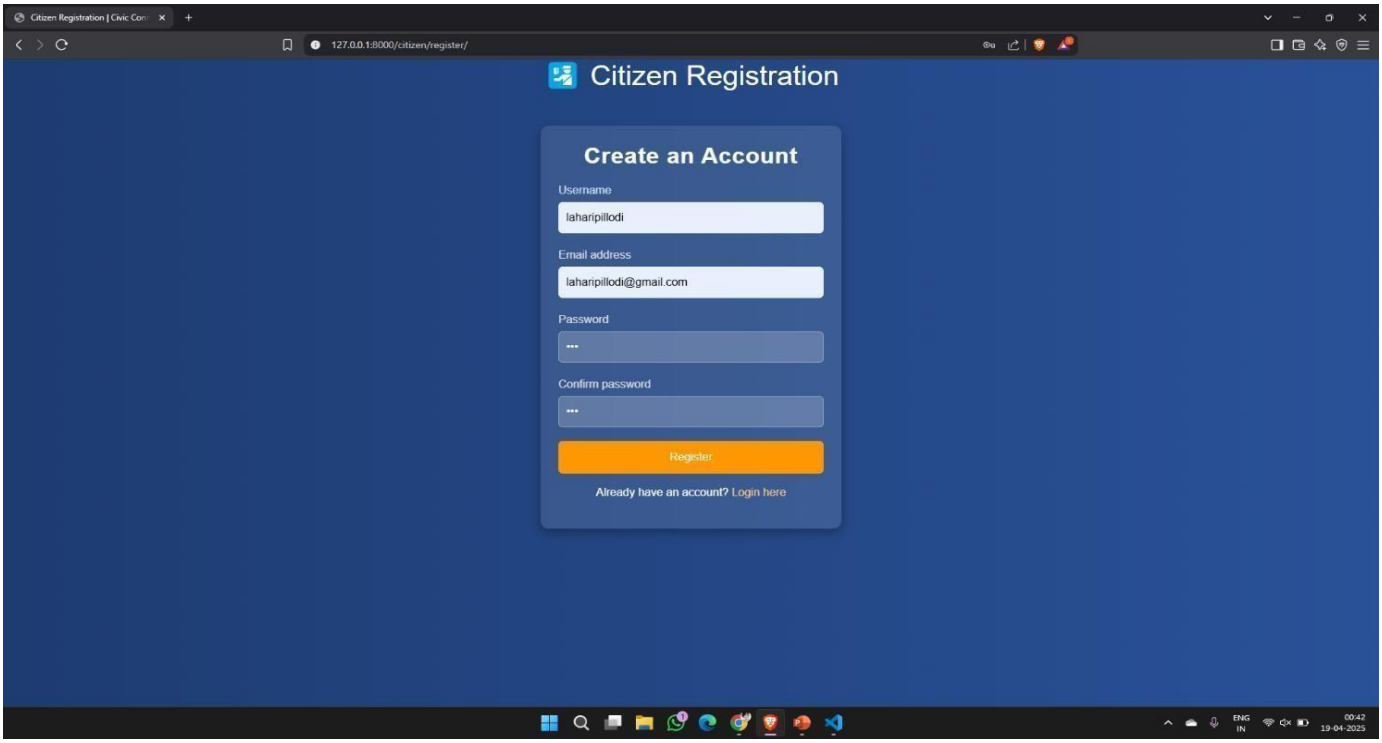


FIG 4.2.1.2: Citizen Registration Page

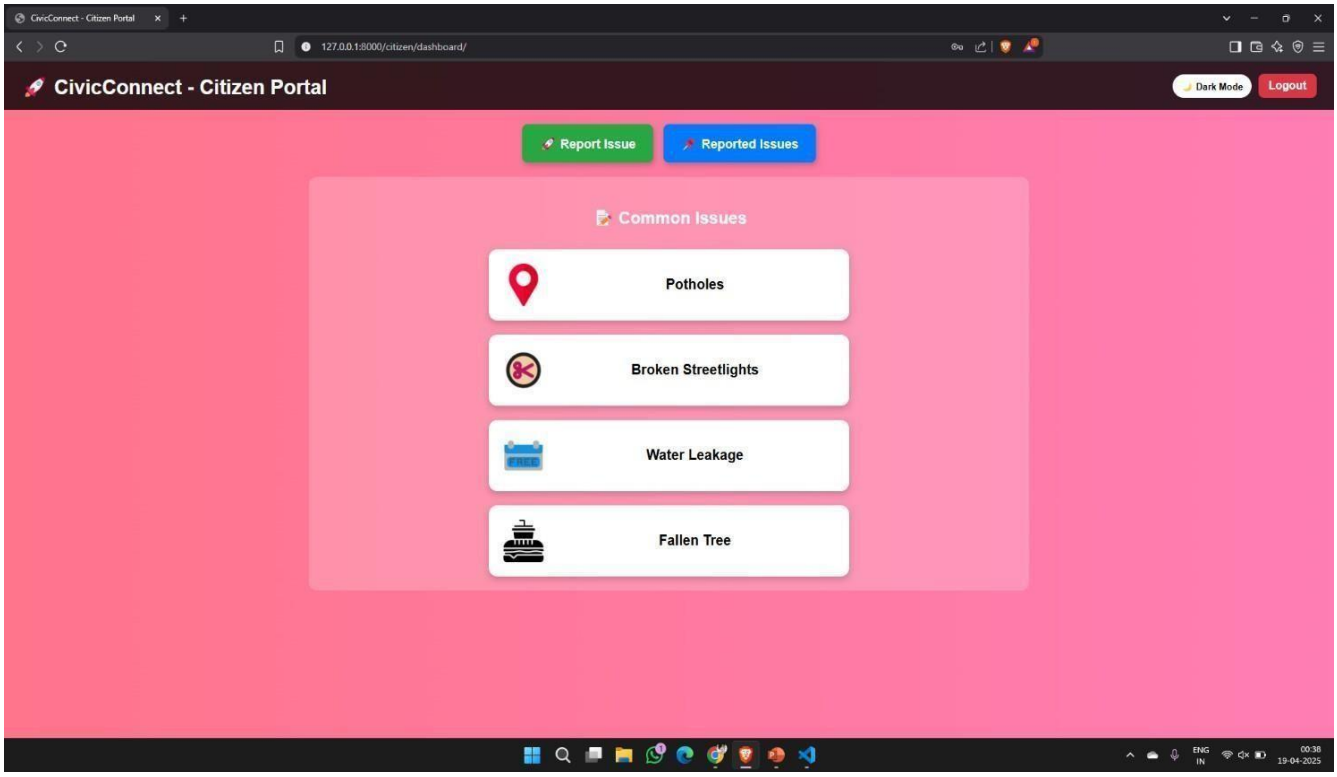
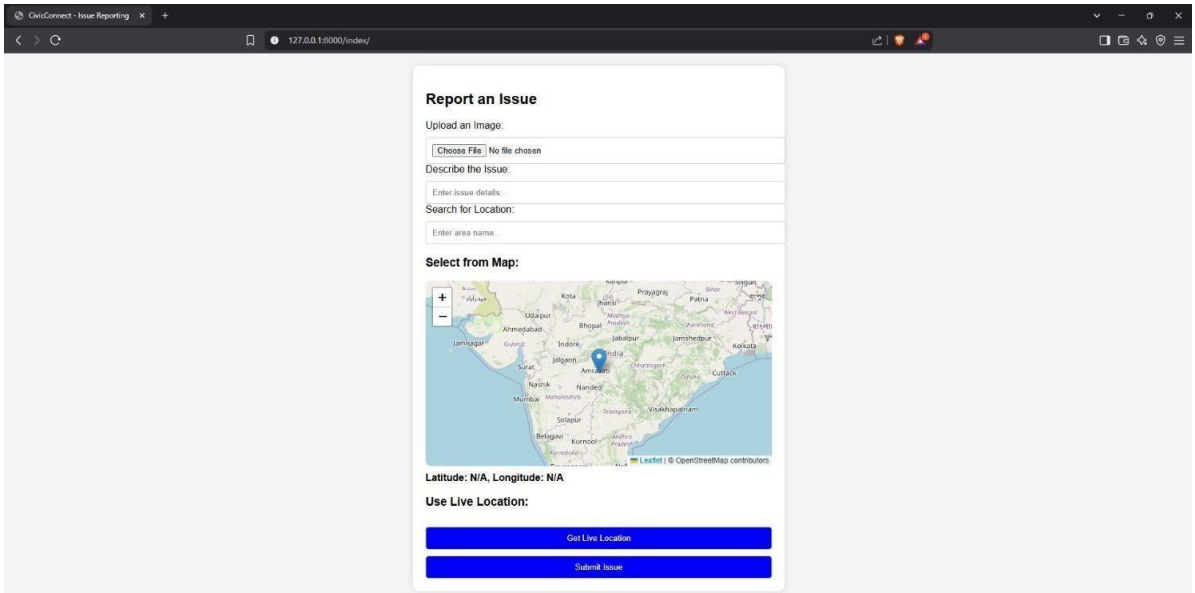
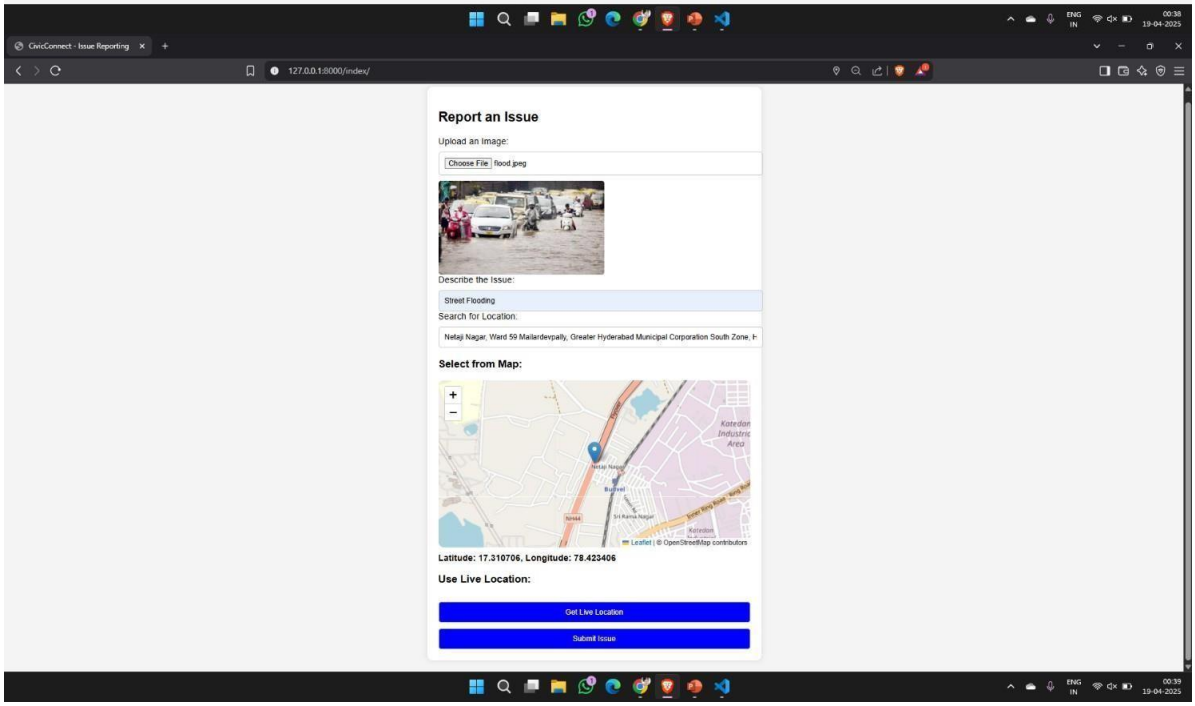
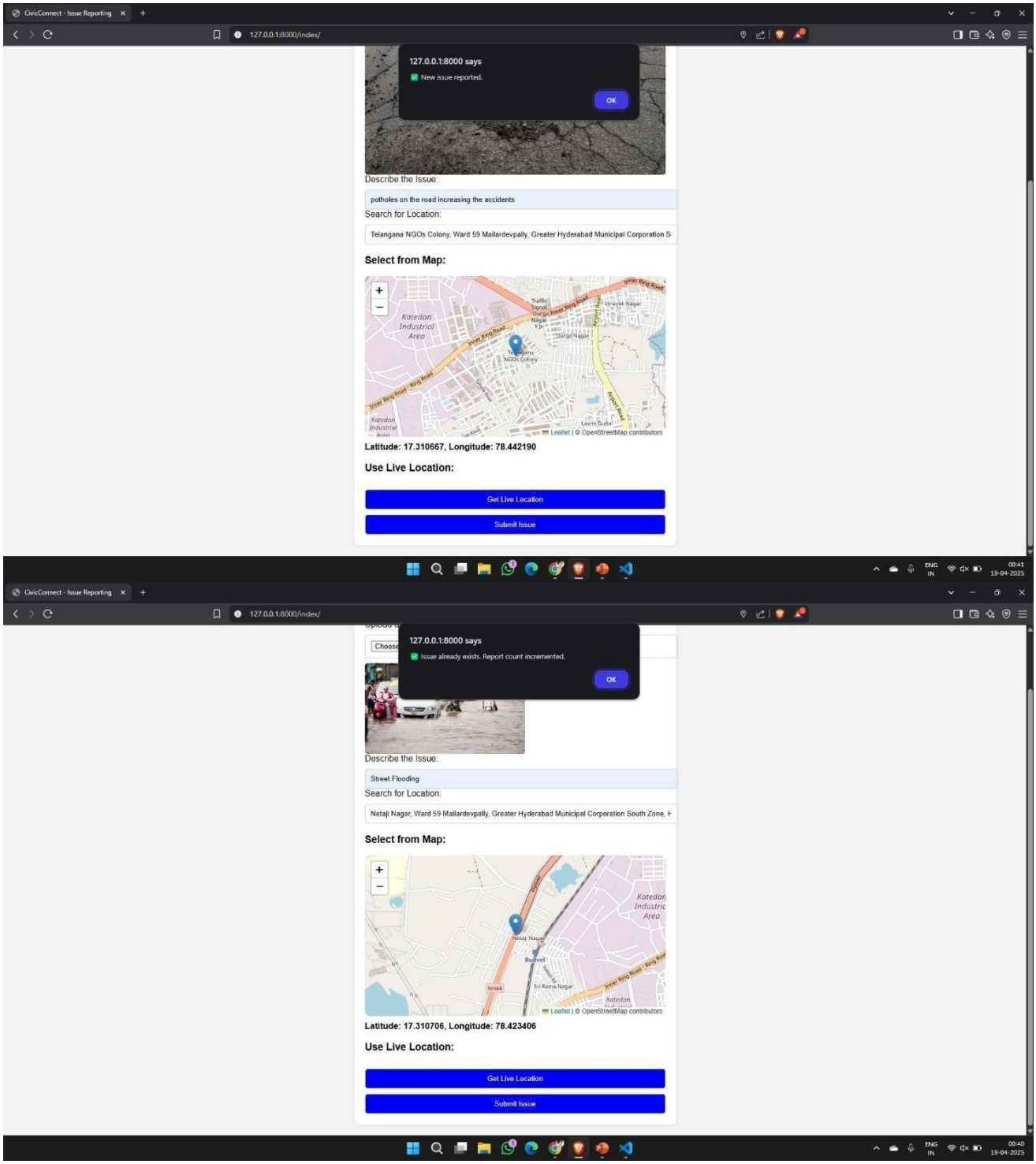


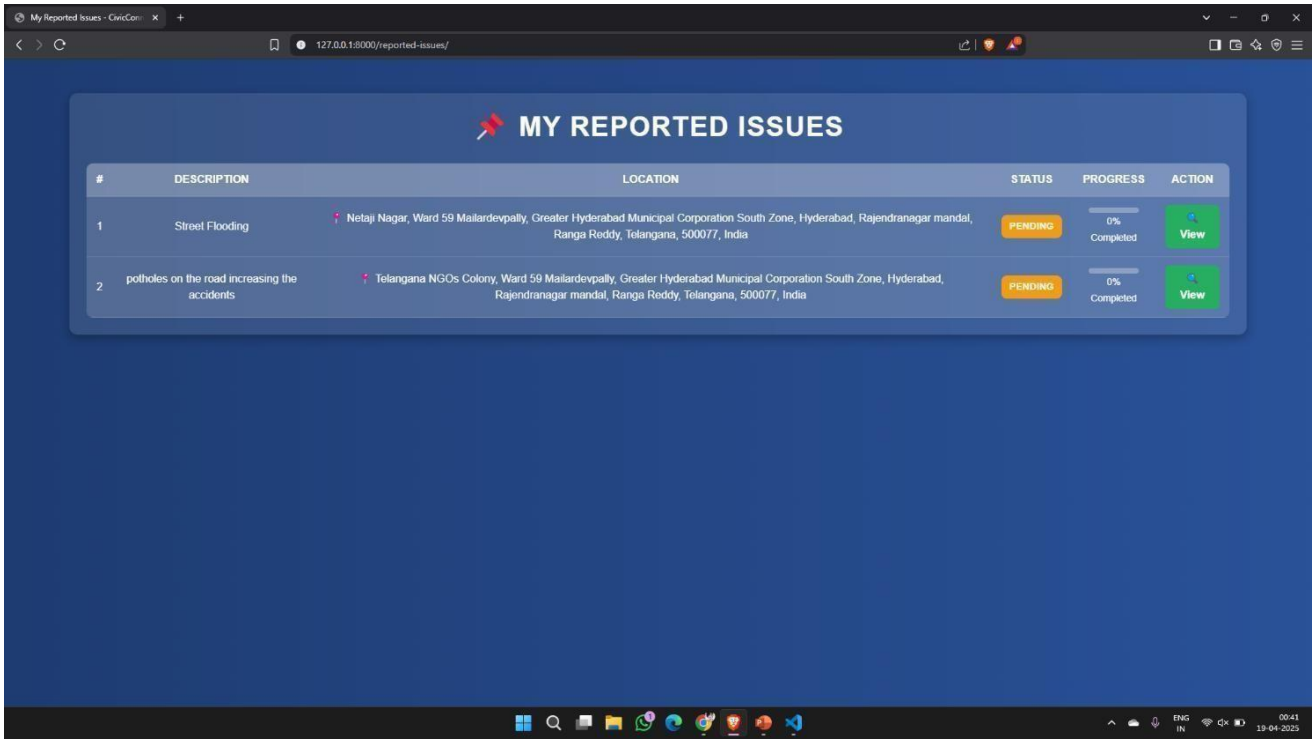
FIG 4.2.1.3: Home Page of CivicConnect (Citizen View)


CivicConnect - Issue Reporting
127.0.0.1:8000/index/
Report an Issue
Upload an Image:
Choose File No file chosen
Describe the Issue:
Enter issue details...
Search for Location:
Enter area name...
Select from Map:
Latitude: N/A, Longitude: N/A
Use Live Location:
Get Live Location
Submit Issue

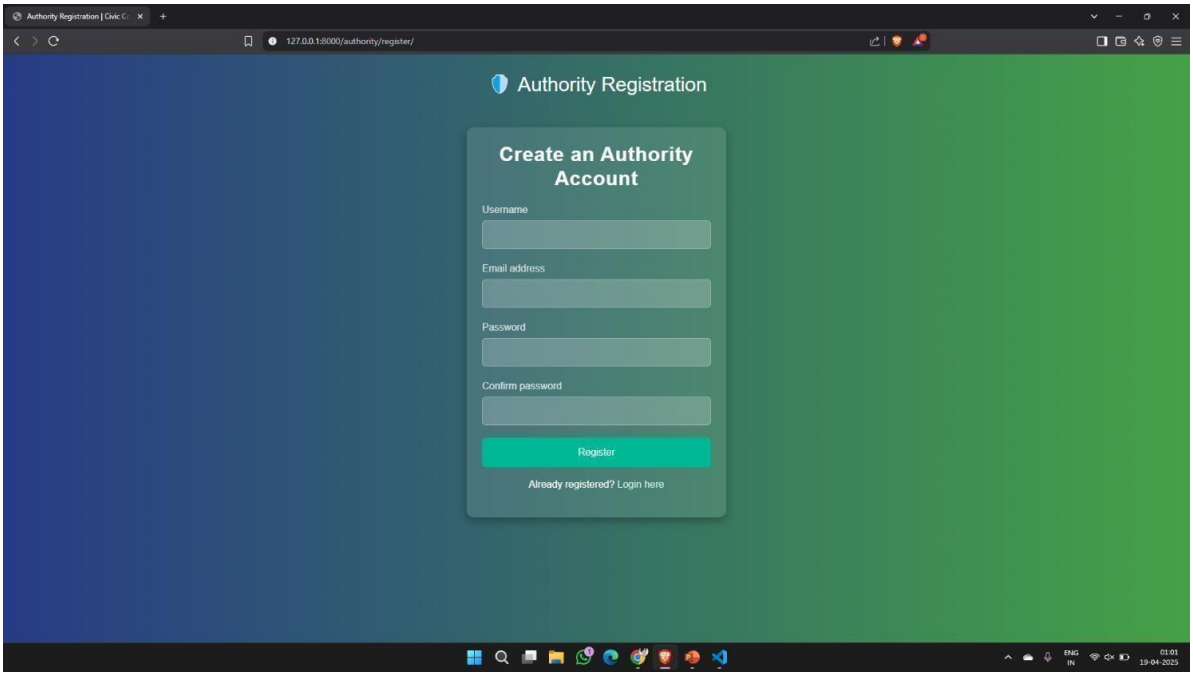

CivicConnect - Issue Reporting
127.0.0.1:8000/index/
Report an Issue
Upload an Image:
Choose File flood.jpeg
Describe the Issue:
Street Flooding
Search for Location:
Netaji Nagar, Ward 59 Mallardevpally, Greater Hyderabad Municipal Corporation South Zone, H
Select from Map:
Latitude: 17.310706, Longitude: 78.423406
Use Live Location:
Get Live Location
Submit Issue

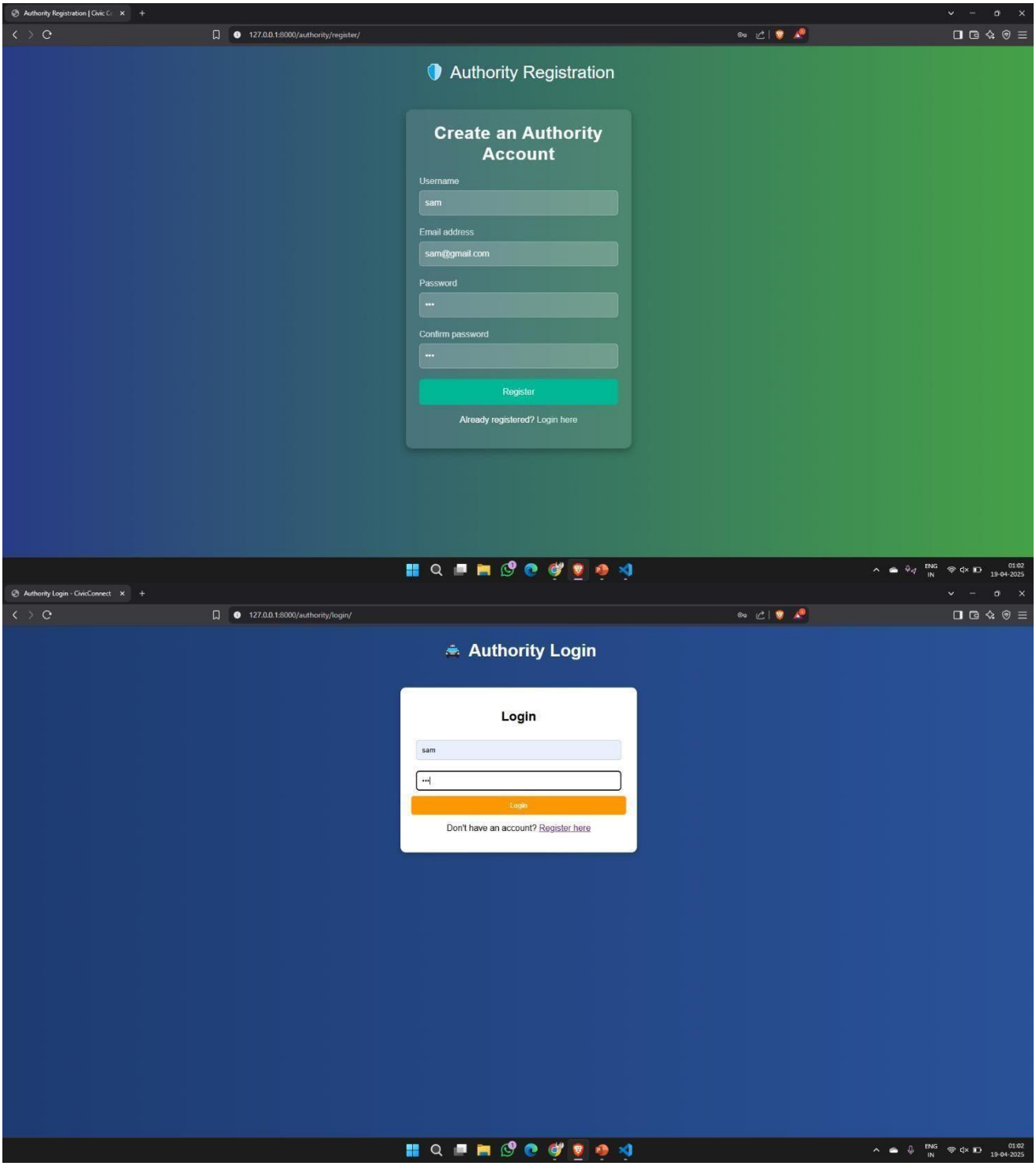


4.2.1.4 ：Issue Reporting Form with Location and Image Upload

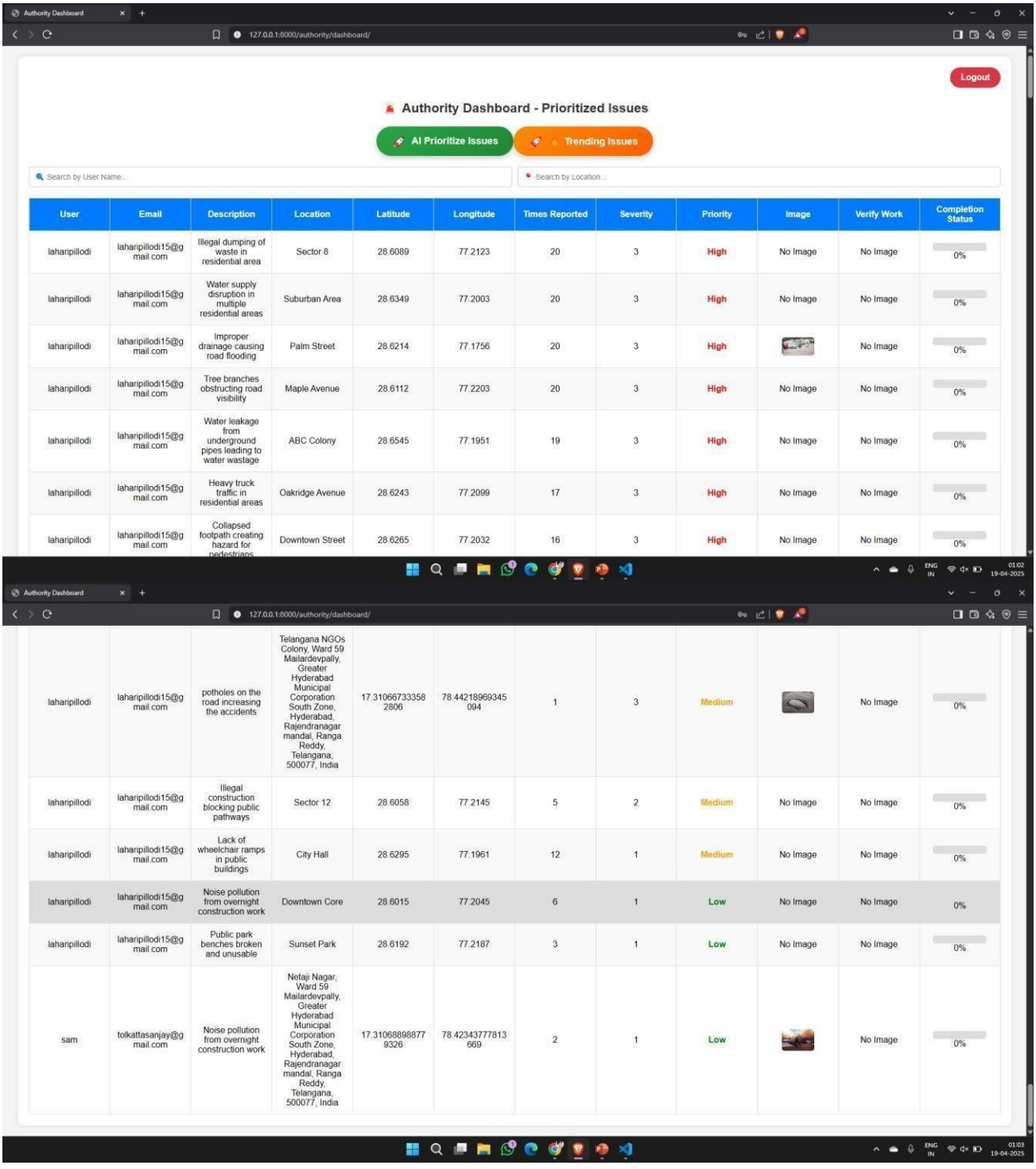


4.2.1.5 : Notification Page Showing Issue Status Update

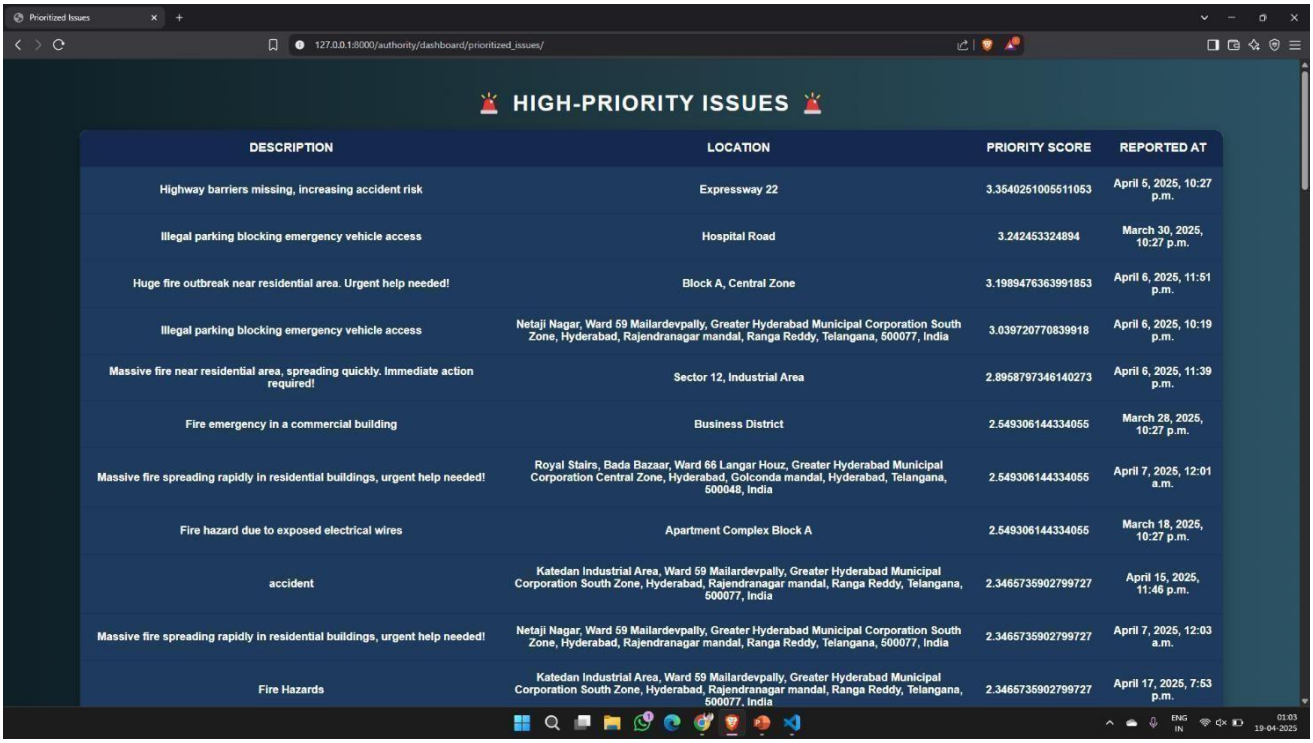




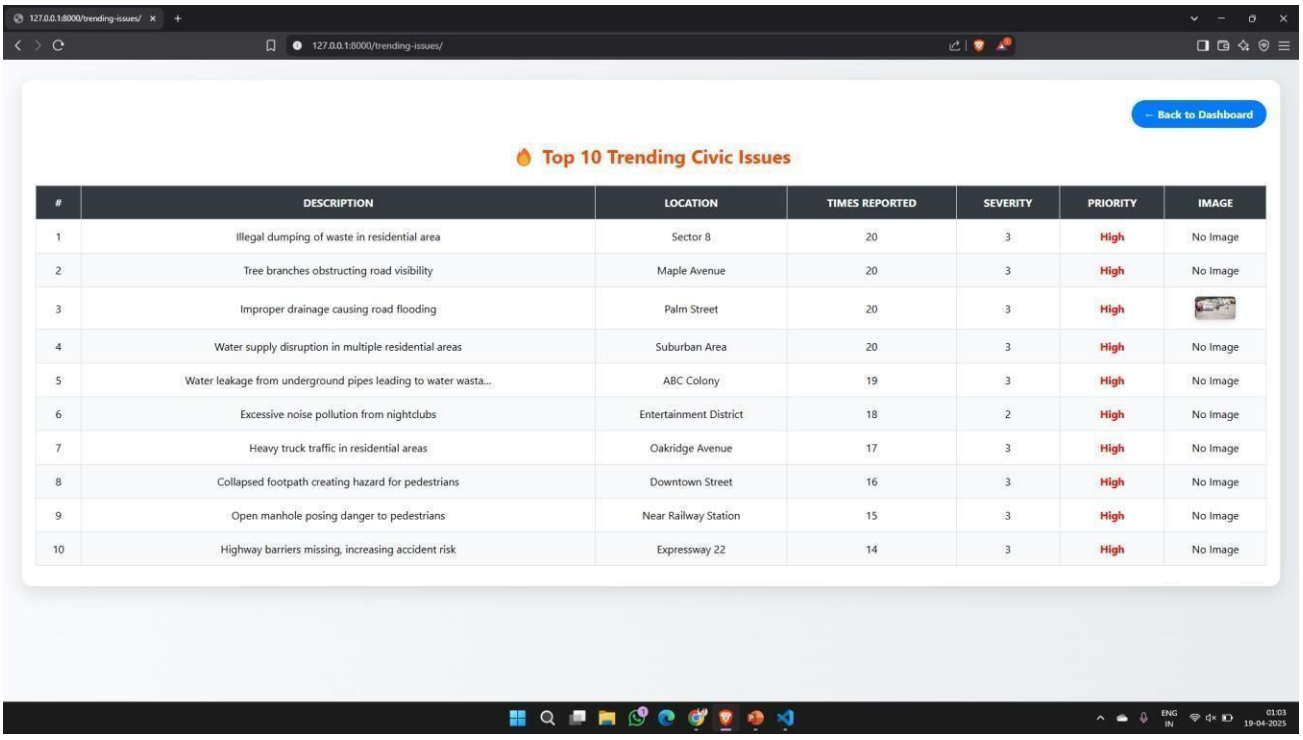
4.2.1.6 : Authority Login Page



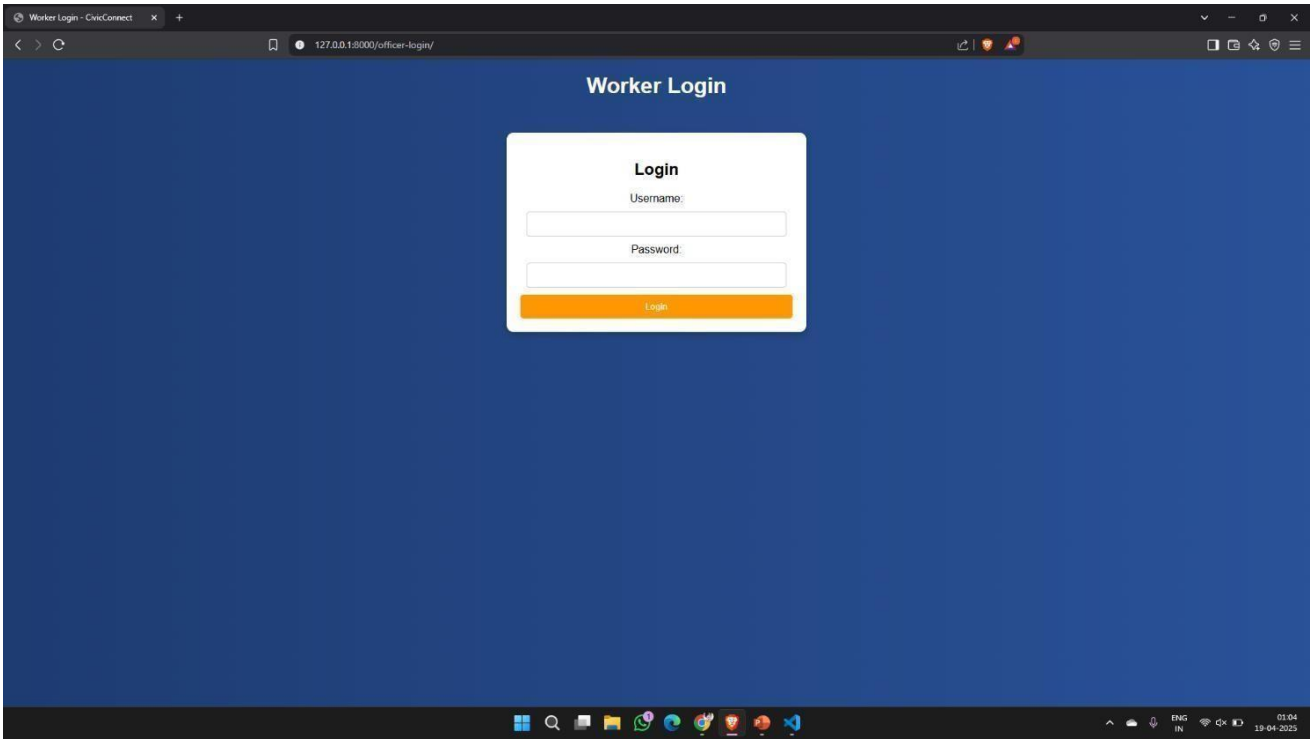
4.2.1.7 :Authority Dashboard for Managing Issues



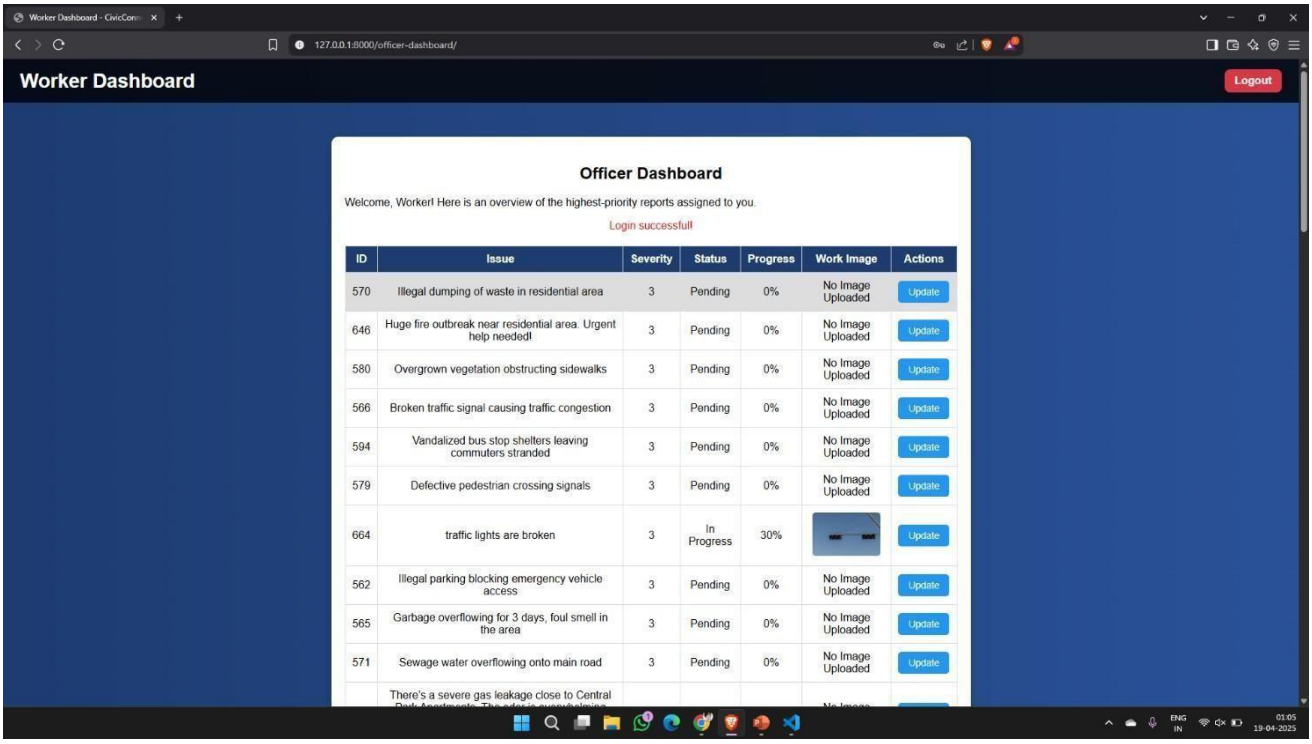
4.2.1.8 :AI-based Severity Prediction Output



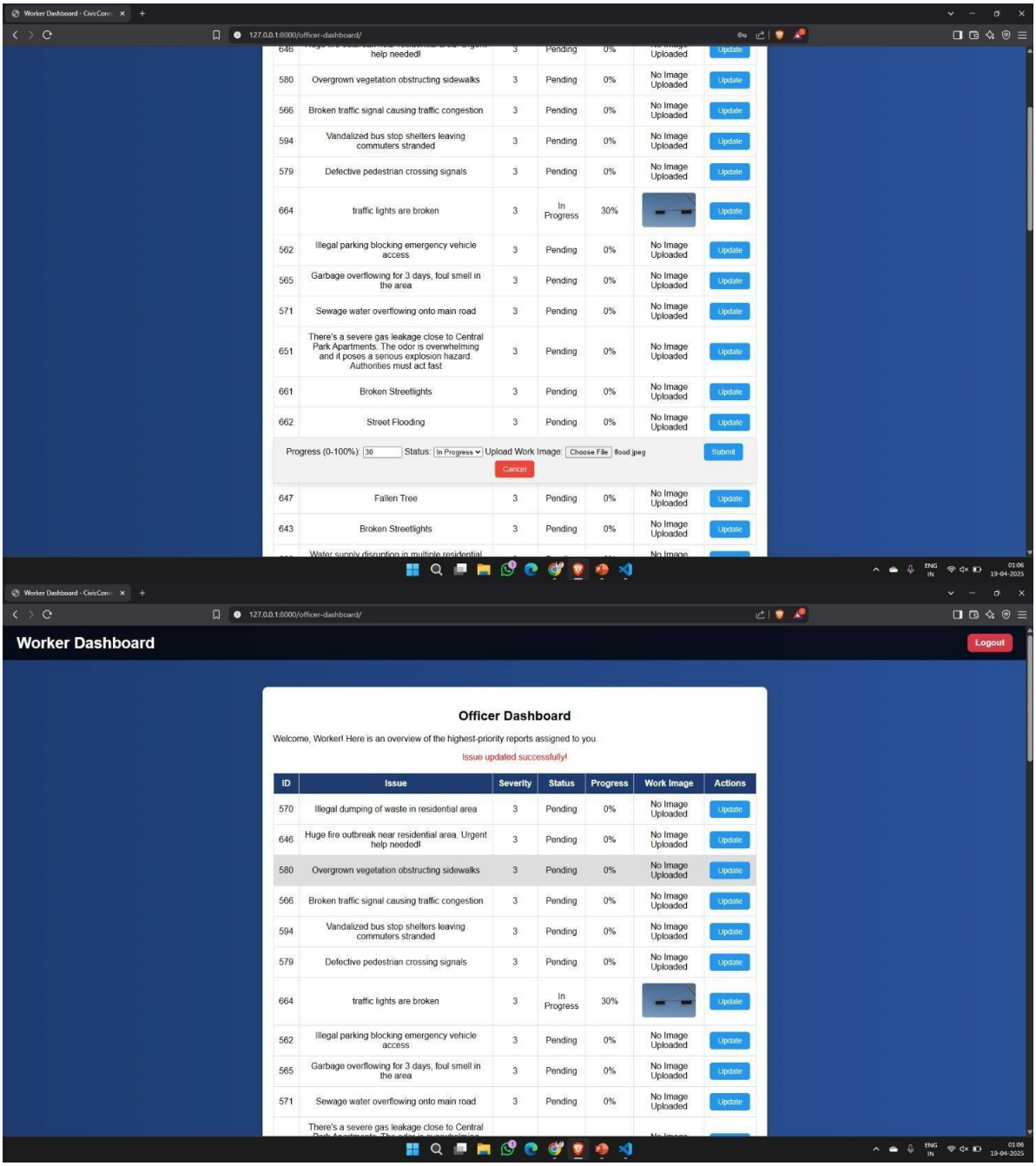
4.2.1.9 : Trending Issues

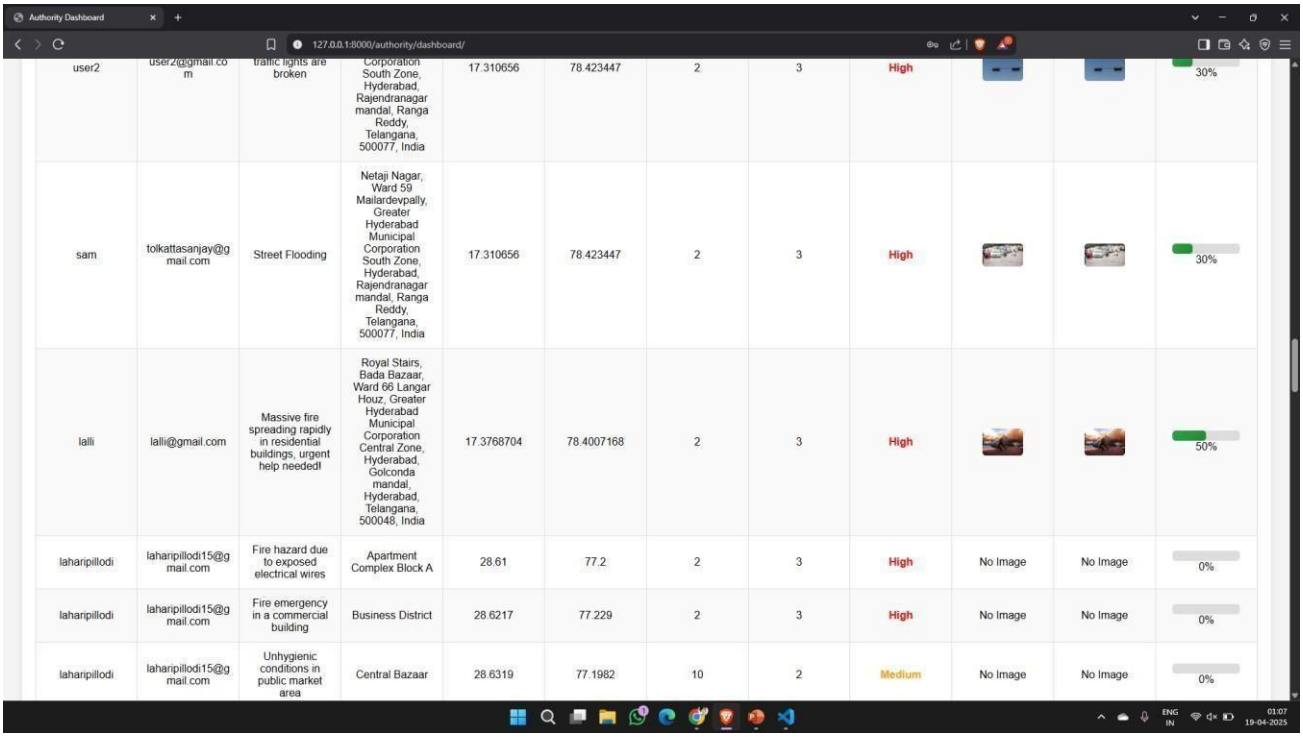


4.2.1.10 :Workers Login page

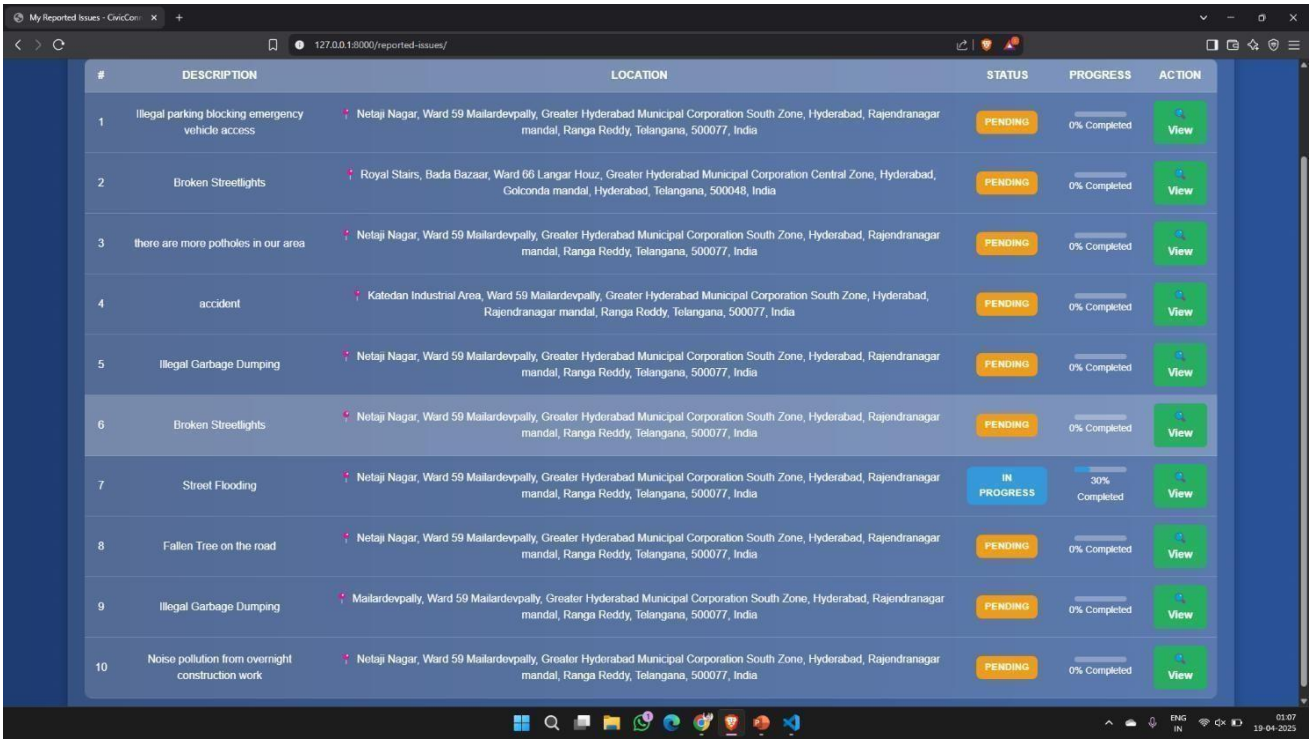


4.2.1.11 :Officers Dashboard Overview


Worker Dashboard - CivicConn
127.0.0.1:8000/officer-dashboard/
646 help needed! 3 Pending 0% No Image Uploaded Update
580 Overgrown vegetation obstructing sidewalks 3 Pending 0% No Image Uploaded Update
566 Broken traffic signal causing traffic congestion 3 Pending 0% No Image Uploaded Update
594 Vandalized bus stop shelters leaving commuters stranded 3 Pending 0% No Image Uploaded Update
579 Defective pedestrian crossing signals 3 Pending 0% No Image Uploaded Update
664 traffic lights are broken 3 In Progress 30% Update
562 Illegal parking blocking emergency vehicle access 3 Pending 0% No Image Uploaded Update
565 Garbage overflowing for 3 days, foul smell in the area 3 Pending 0% No Image Uploaded Update
571 Sewage water overflowing onto main road 3 Pending 0% No Image Uploaded Update
651 There's a severe gas leakage close to Central Park Apartments. The odor is overwhelming and it poses a serious explosion hazard. Authorities must act fast. 3 Pending 0% No Image Uploaded Update
661 Broken Streetlights 3 Pending 0% No Image Uploaded Update
662 Street Flooding 3 Pending 0% No Image Uploaded Update
Progress (0-100%): 30 Status: In Progress Upload Work Image: Choose File flood.jpeg Submit
Cancel
647 Fallen Tree 3 Pending 0% No Image Uploaded Update
643 Broken Streetlights 3 Pending 0% No Image Uploaded Update
Worker Dashboard - CivicConn
127.0.0.1:8000/officer-dashboard/
Worker Dashboard
Logout
Officer Dashboard
Welcome, Worker! Here is an overview of the highest-priority reports assigned to you.
Issue updated successfully!
ID Issue Severity Status Progress Work Image Actions
570 Illegal dumping of waste in residential area 3 Pending 0% No Image Uploaded Update
646 Huge fire outbreak near residential area. Urgent help needed! 3 Pending 0% No Image Uploaded Update
580 Overgrown vegetation obstructing sidewalks 3 Pending 0% No Image Uploaded Update
566 Broken traffic signal causing traffic congestion 3 Pending 0% No Image Uploaded Update
594 Vandalized bus stop shelters leaving commuters stranded 3 Pending 0% No Image Uploaded Update
579 Defective pedestrian crossing signals 3 Pending 0% No Image Uploaded Update
664 traffic lights are broken 3 In Progress 30% Update
562 Illegal parking blocking emergency vehicle access 3 Pending 0% No Image Uploaded Update
565 Garbage overflowing for 3 days, foul smell in the area 3 Pending 0% No Image Uploaded Update
571 Sewage water overflowing onto main road 3 Pending 0% No Image Uploaded Update



user2	user2@gmail.com	traffic lights are broken	Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	17.310656	78.423447	2	3	High			30%
sam	tolkattasanjay@gmail.com	Street Flooding	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	17.310656	78.423447	2	3	High			30%
lalli	lalli@gmail.com	Massive fire spreading rapidly in residential buildings, urgent help needed!	Royal Stairs, Bada Bazaar, Ward 66 Langar Houz, Greater Hyderabad Municipal Corporation Central Zone, Hyderabad, Golconda mandal, Hyderabad, Telangana, 500048, India	17.3768704	78.4007168	2	3	High			50%
laharipillodi	laharipillodi15@gmail.com	Fire hazard due to exposed electrical wires	Apartment Complex Block A	28.61	77.2	2	3	High	No Image	No Image	0%
laharipillodi	laharipillodi15@gmail.com	Fire emergency in a commercial building	Business District	28.6217	77.229	2	3	High	No Image	No Image	0%
laharipillodi	laharipillodi15@gmail.com	Unhygienic conditions in public market area	Central Bazaar	28.6319	77.1982	10	2	Medium	No Image	No Image	0%



#	DESCRIPTION	LOCATION	STATUS	PROGRESS	ACTION
1	Illegal parking blocking emergency vehicle access	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
2	Broken Streetlights	Royal Stairs, Bada Bazaar, Ward 66 Langar Houz, Greater Hyderabad Municipal Corporation Central Zone, Hyderabad, Golconda mandal, Hyderabad, Telangana, 500048, India	PENDING	0% Completed	View
3	there are more potholes in our area	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
4	accident	Katedan Industrial Area, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
5	Illegal Garbage Dumping	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
6	Broken Streetlights	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
7	Street Flooding	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	IN PROGRESS	30% Completed	View
8	Fallen Tree on the road	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
9	Illegal Garbage Dumping	Mailardevpally, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View
10	Noise pollution from overnight construction work	Netaji Nagar, Ward 59 Mailardevpally, Greater Hyderabad Municipal Corporation South Zone, Hyderabad, Rajendranagar mandal, Ranga Reddy, Telangana, 500077, India	PENDING	0% Completed	View

4.2.1.1.12: Issue Status Update

Base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Civic Connect{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'style.css' %}">


    <style>
        /* General Styles */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background: linear-gradient(to right, #800080, #a020f0);
            color: white;
            text-align: center;
        }

        header {
            background: rgba(0, 0, 0, 0.7);
            padding: 1em;
        }

        header h1 {
            margin: 0;
            font-size: 2em;
        }

        nav {
            margin-top: 10px;
        }

        .btn {
            background: #ff9800;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            border-radius: 5px;
            margin: 5px;
            display: inline-block;
            transition: 0.3s;
            cursor: pointer;
            border: none;
        }

        .btn:hover {
            background: #e68900;
```

```
}

.hero {
    padding: 30px 20px;
    background: rgba(255, 255, 255, 0.1);
    margin: 20px 20px 0 20px;
    border-radius: 10px;
}

.hero h2 {
    max-width: 600px;
    margin: 0 auto;
    font-size: 1.3em;
}

.hero p {
    font-size: 1.1em;
    margin-top: 5px;
}

.badges {
    padding: 40px 20px;
    margin-top: 0;
}

.badge-container {
    display: flex;
    justify-content: center;
    gap: 20px;
}

.badge {
    background: white;
    padding: 15px;
    border-radius: 10px;
    color: black;
    width: 150px;
    text-align: center;
    transition: 0.3s;
}

.badge img {
    width: 80px;
}

.badge:hover {
    transform: scale(1.1);
}

.features {
    padding: 40px 20px;
}
```

```
      .feature-list {
         display: flex;
         justify-content: center;
         gap: 20px;
      }

      .feature {
         background: rgba(255, 255, 255, 0.2);
         padding: 20px;
         border-radius: 10px;
         width: 250px;
         transition: 0.3s;
      }

      .feature:hover {
         transform: scale(1.1);
      }

      footer {
         margin-top: 20px;
         background: rgba(0, 0, 0, 0.7);
         padding: 10px;
      }
   </style>
</head>
<body>

   <header>
      <h1> Civic Connect</h1>
      <nav>
         <button class="btn" onclick="redirectToLogin()">Citizen Login</button>
         <button class="btn" onclick="redirectToAuthorityLogin()">Authority Login</button>
         <button class="btn" onclick="redirectToOfficerLogin()">Worker Login</button>



      </nav>
   </header>

   <main>
      <section class="hero">
         <h2>Empowering Citizens, Improving Communities</h2>
         <p>Report issues, track progress, and make your city better.</p>
      </section>

      <section class="badges">
         Badges</h2>
         <div class="badge-container">
            <div class="badge">
               <img src="https://cdn-icons-png.flaticon.com/512/847/847969.png" alt="Badge 1">
               <p>Active Reporter</p>
            </div>
```

```
            <div class="badge">
                <img src="https://cdn-icons-png.flaticon.com/512/1828/1828884.png" alt="Badge
2">
                <p>Community Helper</p>
            </div>
            <div class="badge">
                <img src="https://cdn-icons-png.flaticon.com/512/992/992651.png" alt="Badge 3">
                <p>Top Contributor</p>
            </div>
        </div>
    </section>

    <section class="features">
        <h2>Why Use Civic Connect?</h2>
        <div class="feature-list">
            <div class="feature">
                <h3>📢 Report Issues</h3>
                <p>Snap a photo and submit your complaint in seconds.</p>
            </div>
            <div class="feature">
                <h3>🚀 Track Progress</h3>
                <p>Get real-time updates on issue resolutions.</p>
            </div>
            <div class="feature">
                <h3>🎯 Earn Badges</h3>
                <p>Stay active and earn recognition for your contributions.</p>
            </div>
        </div>
    </section>
</main>

<footer>
    <p>&copy; 2025 Civic Connect | Connecting Communities</p>
</footer>

<script>
    function redirectToLogin() {
        window.location.href = "{% url 'citizen_login' %}";
    }

    function redirectToAuthorityLogin() {
        window.location.href = "{% url 'authority_login' %}";
    }
    function redirectToOfficerLogin() {
        window.location.href = "{% url 'officer_login' %}";
    }
</script>
</body>
</html>
```

Citizen_login.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>citizen Login - CivicConnect</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(to right, #1e3c72, #2a5298);
            color: white;
            text-align: center;
            margin: 0;
            padding: 0;
        }

        .container {
            background: white;
            color: black;
            max-width: 400px;
            margin: 50px auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
        }

        h2 {
            margin-bottom: 20px;
        }

        input {
            width: 90%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        .btn {
            background: #ff9800;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            border-radius: 5px;
            display: inline-block;
            border: none;
            cursor: pointer;
            width: 100%;
        }
```

```
        .btn:hover {
            background: #e68900;
        }
    </style>
</head>
<body>

    <h1>citizen Login</h1>

    <div class="container">
        <h2>Login</h2>
        <form method="POST">
            {% csrf_token %}
                <input type="text" id="username" name="username" placeholder="Enter Username"
required><br>
              <input type="password" id="password" name="password" placeholder="Enter Password"
required><br>
            <button type="submit" class="btn">Login</button>
            <p>Don't have an account? <a href="{% url 'citizen_register' %}">Register here</a></p>

        </form>
    </div>

</body>
</html>
```

Citizen_register.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Citizen Registration | Civic Connect</title>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

<style>
/* General Styling */
body {
font-family: 'Poppins', sans-serif;
background: linear-gradient(to right, #1e3c72, #2a5298);
color: white;
text-align: center;
margin: 0;
padding: 0;
}

/* Registration Box */
.container {
background: rgba(255, 255, 255, 0.1);
color: white;
max-width: 420px;
margin: 50px auto;
padding: 25px;
border-radius: 12px;
box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
backdrop-filter: blur(10px);
}

h2 {
font-weight: 600;
letter-spacing: 1px;
margin-bottom: 20px;
}

/* Floating Labels */
.form-group {
position: relative;
margin-bottom: 20px;
text-align: left;
}

.form-group label {
font-weight: 500;
```

```
color: rgba(255, 255, 255, 0.8);
margin-bottom: 5px;
display: block;
}

.form-group input {
width: 100%;
padding: 10px;
background: rgba(255, 255, 255, 0.2);
border: 1px solid rgba(255, 255, 255, 0.3);
border-radius: 6px;
color: white;
font-size: 1rem;
outline: none;
}

.form-group input::placeholder {
color: rgba(255, 255, 255, 0.7);
}

/* Buttons */
.btn {
background: #ff9800;
color: white;
padding: 12px;
font-size: 1rem;
border: none;
border-radius: 6px;
width: 100%;
cursor: pointer;
transition: 0.3s ease-in-out;
}

.btn:hover {
background: #e68900;
transform: scale(1.05);
}

/* Error Message */
.error {
color: #ff4d4d;
font-size: 0.9rem;
margin-top: 5px;
}

/* Responsive */
@media screen and (max-width: 480px) {
.container {
        width: 90%;
        padding: 20px;
}
}
```

```
</style>
</head>
<body>

<h1>Citizen Registration</h1>

<div class="container">
<h2>Create an Account</h2>

<!-- Show Messages Dynamically -->
{% if messages %}
{% for message in messages %}
        <div class="alert {% if message.tags == 'success' %}alert-success{% else %}alert-
        danger{% endif %} fade show" role="alert">
                {{ message }}
        </div>
{% endfor %}
{% endif %}

<form method="POST">
{% csrf_token %}

{% for field in form %}
        <div class="form-group">
                <label for="{{ field.id_for_label }}">{{ field.label }}</label>
                {{ field }}
                {% if field.errors %}
                <div class="error">
                {{ field.errors|striptags }}
                </div>
                {% endif %}
        </div>
{% endfor %}

<button type="submit" class="btn">Register</button>
</form>

<p class="mt-3">Already have an account? <a href="{% url 'citizen_login' %}" style="color:
#ffcc80; text-decoration: none;">Login here</a></p>
</div>

</body>
</html>
```

Citizen_dashboard.html

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CivicConnect - Citizen Portal</title>

    <style>
        /* Global Styles */
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(to right, #ff758c, #ff7eb3);
            color: white;
            text-align: center;
            margin: 0;
            padding: 0;
            transition: background 0.3s ease-in-out;
        }

        /* Header */
        header {
            background: rgba(0, 0, 0, 0.8);
            display: flex;
            align-items: center;
            justify-content: space-between;
            padding: 15px 25px;
        }

        header h1 {
            margin: 0;
            font-size: 1.8em;
        }

        /* Header Buttons (Logout & Dark Mode) */
        .header-buttons {
            display: flex;
            align-items: center;
            gap: 10px;
        }

        /* Logout Button */
        .logout-btn {
            background: #dc3545;
            color: white;
            padding: 8px 15px;
            border-radius: 8px;
            font-weight: bold;
```

```
    text-decoration: none;
    transition: 0.3s;
    border: none;
    cursor: pointer;
}

.logout-btn:hover {
    background: #c82333;
    transform: scale(1.05);
}

/* F    Dark Mode Toggle */
.dark-mode-btn {
    background: white;
    color: black;
    border: none;
    padding: 8px 12px;
    border-radius: 20px;
    font-weight: bold;
    cursor: pointer;
    transition: 0.3s;
}

.dark-mode-btn:hover {
    background: #ddd;
}

/*  Buttons  */
.button-container {
    display: flex;
    justify-content: center;
    gap: 15px;
    margin-top: 20px;
}

.btn {
    background: #28a745;
    color: white;
    padding: 15px 25px;
    border-radius: 8px;
    border: none;
    cursor: pointer;
    font-size: 18px;
    font-weight: bold;
    transition: 0.3s;
    box-shadow: 2px 4px 8px rgba(0, 0, 0, 0.2);
    text-decoration: none;
    display: inline-block;
}

.btn:hover {
```

```
    background: #218838;
    transform: scale(1.05);
}

/* Track Issue Button */
.track-btn {
    background: #007bff;
}

.track-btn:hover {
    background: #0056b3;
}

/* Reported Issues Section */
.report-container {
    background: rgba(255, 255, 255, 0.2);
    margin: 20px auto;
    padding: 20px;
    border-radius: 12px;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    align-items: stretch;
    gap: 20px;
    max-width: 1000px;
}

.report-container h2 {
    width: 100%;
    text-align: center;
    margin-bottom: 10px;
}

.issue {
    display: flex;
    align-items: center;
    justify-content: flex-start;
    background: white;
    color: black;
    padding: 20px;
    border-radius: 12px;
    width: calc(50% - 20px);
    min-width: 320px;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
    transition: 0.3s ease-in-out;
}

.issue:hover {
    transform: translateY(-5px);
    box-shadow: 0px 6px 12px rgba(0, 0, 0, 0.3);
}
```

```
        .issue img {
            width: 60px;
            height: 60px;
            margin-right: 15px;
        }

        .issue span {
            font-size: 20px;
            font-weight: bold;
            flex-grow: 1;
        }

        @media (max-width: 768px) {
            .issue {
                width: 90%;
            }
        }

        /* Dark Mode */
        .dark-mode {
            background: linear-gradient(to right, #333, #222);
            color: white;
        }

        .dark-mode .issue {
            background: #444;
            color: white;
        }

        .dark-mode .report-container {
            background: rgba(0, 0, 0, 0.3);
        }

    </style>
</head>
<body>

    <header>
        CivicConnect - Citizen Portal</h1>
        <div class="header-buttons">
            <button class="dark-mode-btn" onclick="toggleDarkMode()">Dark Mode</button>
            <a href="{% url 'logout' %}" class="logout-btn">Logout</a>
        </div>
    </header>

    <!-- Report & Track Issue Buttons -->
    <div class="button-container">
        <a href="{% url 'index' %}" Report Issue</a>
        <a href="{% url 'reported_issues' %}" class="btn track-btn">Reported Issues</a>
    </div>
```

```
    <div class="report-container" id="report-box">
      <h2>❓ Common Issues</h2>

      <div class="issue">
        <img src="https://cdn-icons-png.flaticon.com/512/684/684908.png" alt="Pothole">
        <span>Potholes</span>
      </div>

      <div class="issue">
        <img src="https://cdn-icons-png.flaticon.com/512/5785/5785093.png" alt="Streetlight">
        <span>Broken Streetlights</span>
      </div>

      <div class="issue">
        <img src="https://cdn-icons-png.flaticon.com/512/5786/5786191.png" alt="Water Leakage">
        <span>Water Leakage</span>
      </div>

      <div class="issue">
        <img src="https://cdn-icons-png.flaticon.com/512/1037/1037820.png" alt="Fallen Tree">
        <span>Fallen Tree</span>
      </div>
    </div>

    <script>
      function toggleDarkMode() {
        document.body.classList.toggle("dark-mode");
        const btn = document.querySelector(".dark-mode-btn");
        if (document.body.classList.contains("dark-mode")) {
          btn.textContent = "🌕 Light Mode";
        } else {
          btn.textContent = "🌑 Dark Mode";
        }
      }
    </script>

</body>
</html>
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CivicConnect - Issue Reporting</title>

    <!-- ■ Latest Leaflet CSS -->
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />

    <!-- ■ Load Leaflet JS -->
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>

    {% load static %}
    <link rel="icon" href="{% static 'favicon.ico' %}" type="image/x-icon">

    <style>
        body { font-family: Arial, sans-serif; max-width: 600px; margin: auto; padding: 20px;
background-color: #f4f4f4; }
        .container { background: white; padding: 20px; border-radius: 10px; box-shadow: 0 0 10px
rgba(0, 0, 0, 0.1); }
        input, button, textarea { width: 100%; margin-top: 10px; padding: 10px; border: 1px solid #ccc;
border-radius: 5px; }
        button { background: blue; color: white; cursor: pointer; }
        #map { height: 300px; border-radius: 10px; margin-top: 10px; }
        img { max-width: 100%; margin-top: 10px; border-radius: 5px; }
        #locationDisplay { margin-top: 10px; font-weight: bold; }
        .dropdown { position: relative; }
        .dropdown-content {
            display: none;
            position: absolute;
            background: white;
            width: 100%;
            border: 1px solid #ccc;
            border-radius: 5px;
            max-height: 150px;
            overflow-y: auto;
            z-index: 1000;
        }
        .dropdown-content div { padding: 10px; cursor: pointer; }
        .dropdown-content div:hover { background: #f1f1f1; }
    </style>
</head>
<body>
    <div class="container">
        <h2>Report an Issue</h2>

        <form id="issueForm" method="POST" enctype="multipart/form-data">
            {% csrf_token %}
```

```
        <label for="imageInput">Upload an Image:</label>
        <input type="file" id="imageInput" name="image" accept="image/*">
        <img id="preview" style="display: none;">

        <div class="dropdown">
            <label for="issueDescription">Describe the Issue:</label>
                <input type="text" id="issueDescription" name="description" placeholder="Enter issue
details..." oninput="filterIssues()">
            <div id="issueList" class="dropdown-content"></div>
        </div>


        <label for="locationSearch">Search for Location:</label>
            <input type="text" id="locationSearch" name="location_name" placeholder="Enter area
name...">

        <h3>Select from Map:</h3>
        <div id="map"></div>
        <div id="locationDisplay">Latitude: N/A, Longitude: N/A</div>

        <h3>Use Live Location:</h3>
        <button type="button" onclick="getLiveLocation()">Get Live Location</button>

        <input type="hidden" id="latitude" name="latitude">
        <input type="hidden" id="longitude" name="longitude">

        <button type="submit">Submit Issue</button>
    </form>
</div>

<script>
    let issueLat = null, issueLon = null;
    const locationDisplay = document.getElementById("locationDisplay");
    const locationSearch = document.getElementById("locationSearch");
    const issueInput = document.getElementById("issueDescription");
    const issueList = document.getElementById("issueList");
    const imageInput = document.getElementById("imageInput");
    const preview = document.getElementById("preview");

    // ■ Image Preview Function
    imageInput.addEventListener("change", function () {
        const file = this.files[0];
        if (file) {
            const reader = new FileReader();
            reader.onload = function (e) {
                preview.src = e.target.result;
                preview.style.display = "block";
            };
            reader.readAsDataURL(file);
        } else {
            preview.style.display = "none";
        }
```

```
    });

    // ■ Common Issues List
    const commonIssues = [
    "Potholes", "Broken Streetlights", "Water Leakage", "Fallen Tree",
    "Power Outage", "Illegal Garbage Dumping", "Fire Hazards", "Blocked Drainage",
    "Sewage Overflow", "Road Damage", "Traffic Signal Malfunction", "Illegal Parking",
    "Construction Debris", "Open Manholes", "Stray Animals", "Noise Pollution",
    "Vandalism", "Uncollected Waste", "Public Toilet Issues", "Street Flooding"
];

function filterIssues() {
    const query = issueInput.value.toLowerCase().trim();
    issueList.innerHTML = "";

    if (query === "") {
        issueList.style.display = "none";
        return;
    }

    const filteredIssues = commonIssues.filter(issue => issue.toLowerCase().includes(query));

    if (filteredIssues.length > 0) {
        issueList.style.display = "block";
        filteredIssues.forEach(issue => {
            const item = document.createElement("div");
            item.textContent = issue;
            item.onclick = () => selectIssue(issue);
            issueList.appendChild(item);
        });
    } else {
        issueList.style.display = "none";
    }
}

function selectIssue(issue) {
    issueInput.value = issue;
    issueList.style.display = "none";
}

// ■ Close dropdown when clicking outside
document.addEventListener("click", function(event) {
    if (!event.target.closest(".dropdown")) {
        issueList.style.display = "none";
    }
});
    // ■ Initialize Map
    const map = L.map('map').setView([20, 78], 5);
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '© OpenStreetMap contributors'
    }).addTo(map);
```

```
let marker = L.marker([20, 78], { draggable: true }).addTo(map);

// ■ Update marker on map click
map.on('click', function(e) {
    setMarker(e.latlng.lat, e.latlng.lng, true);
});

marker.on('dragend', function(e) {
    setMarker(e.target.getLatLng().lat, e.target.getLatLng().lng, true);
});

function getLiveLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
            (position) => {
                setMarker(position.coords.latitude, position.coords.longitude, true);
            },
            (error) => {
                alert("Geolocation error: " + error.message);
            },
            { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }
        );
    } else {
        alert("Geolocation is not supported.");
    }
}

function setMarker(lat, lon, fetchAddress = false) {
    console.log("Setting marker at:", lat, lon);
    issueLat = lat;
    issueLon = lon;
    marker.setLatLng([lat, lon]);
    map.setView([lat, lon], 15);
    locationDisplay.textContent = `Latitude: ${lat.toFixed(6)}, Longitude: ${lon.toFixed(6)}`;
    document.getElementById("latitude").value = lat;
    document.getElementById("longitude").value = lon;

    if (fetchAddress) {
      fetch(`https://nominatim.openstreetmap.org/reverse?format=json&lat=${lat}&lon=${lon}`)
            .then(response => response.json())
            .then(data => {
                locationSearch.value = data.display_name || `Lat: ${lat}, Lon: ${lon}`;
            })
            .catch(error => console.error("Error fetching address:", error));
    }
}

document.getElementById("imageInput").addEventListener("change", function(event) {
    const file = event.target.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = function(e) {
```

```
            document.getElementById("preview").src = e.target.result;
            document.getElementById("preview").style.display = "block";
        };
        reader.readAsDataURL(file);
    }
});

// ■ Fixed: Show Alert Only After Issue is Submitted
document.getElementById("issueForm").addEventListener("submit", function(event) {
    event.preventDefault();

    const formData = new FormData(this);
    formData.append("latitude", issueLat);
    formData.append("longitude", issueLon);

    fetch("{% url 'report_issue' %}", {

        method: "POST",
        body: formData,
        headers: { "X-CSRFToken": getCSRFToken() },
    })
    .then(response => response.json())
    .then(data => {
        if (data.message) {
            alert(" ■" + data.message); // ■ Show success alert after submission

            // ■ Wait for 1 second, then reload the page
            setTimeout(() => {
                location.reload();
            }, 1000);
        } else {
            alert("+ Error: " + (data.error || "Something went wrong!"));
        }
    })
    .catch(error => {
        console.error("Error:", error);
        alert("+ Failed to submit issue. Please try again.");
    });
});

function getCSRFToken() {
    return document.querySelector("[name=csrfmiddlewaretoken]").value;
}
  </script>
</body>
</html>
```

Authority_login.html

```
{% load static %}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Authority Login - CivicConnect</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(to right, #1e3c72, #2a5298);
            color: white;
            text-align: center;
            margin: 0;
            padding: 0;
        }

        .container {
            background: white;
            color: black;
            max-width: 400px;
            margin: 50px auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
        }

        h2 {
            margin-bottom: 20px;
        }

        input {
            width: 90%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        .btn {
            background: #ff9800;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            border-radius: 5px;
            display: inline-block;
            border: none;
            cursor: pointer;
            width: 100%;
        }

        .btn:hover {
            background: #e68900;
```

```
        }
    </style>
</head>
<body>

    <h1>´/ˆ-¨▲ Authority Login</h1>

    <div class="container">
        <h2>Login</h2>
        <form method="POST">
            {% csrf_token %}
                <input type="text" id="username" name="username" placeholder="Enter Username"
required><br>
              <input type="password" id="password" name="password" placeholder="Enter Password"
required><br>
            <button type="submit" class="btn">Login</button>
            <p>Don't have an account? <a href="{% url 'authority_register' %}">Register here</a></p>

        </form>
    </div>

</body>
</html>
```

Authority_register.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Authority Registration | Civic Connect</title>

    <!-- Bootstrap -->
                                                    <link                    rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

    <style>
        body {
            font-family: 'Poppins', sans-serif;
            background: linear-gradient(to right, #283c86, #45a247);
            color: white;
            text-align: center;
            margin: 0;
            padding: 0;
        }
```

```
.container {
    background: rgba(255, 255, 255, 0.1);
    color: white;
    max-width: 420px;
    margin: 50px auto;
    padding: 25px;
    border-radius: 12px;
    box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
    backdrop-filter: blur(10px);
}

h1 {
    margin-top: 30px;
    font-size: 2rem;
}

h2 {
    font-weight: 600;
    letter-spacing: 1px;
    margin-bottom: 20px;
}

.form-group {
    position: relative;
    margin-bottom: 20px;
    text-align: left;
}

.form-group label {
    font-weight: 500;
    color: rgba(255, 255, 255, 0.8);
    margin-bottom: 5px;
    display: block;
}

.form-group input {
    width: 100%;
    padding: 10px;
    background: rgba(255, 255, 255, 0.2);
    border: 1px solid rgba(255, 255, 255, 0.3);
    border-radius: 6px;
    color: white;
    font-size: 1rem;
    outline: none;
}

.form-group input::placeholder {
    color: rgba(255, 255, 255, 0.7);
}

.btn {
    background: #00b894;
```

```
            color: white;
            padding: 12px;
            font-size: 1rem;
            border: none;
            border-radius: 6px;
            width: 100%;
            cursor: pointer;
            transition: 0.3s ease-in-out;
        }

        .btn:hover {
            background: #009475;
            transform: scale(1.05);
        }

        .error {
            color: #ff4d4d;
            font-size: 0.9rem;
            margin-top: 5px;
        }

        a {
            color: #d1f2eb;
            text-decoration: none;
        }

        a:hover {
            text-decoration: underline;
        }

        @media screen and (max-width: 480px) {
            .container {
                width: 90%;
                padding: 20px;
            }
        }
    </style>
</head>
<body>

    <h1>)     Authority Registration</h1>

    <div class="container">
        <h2>Create an Authority Account</h2>

        {% if messages %}
            {% for message in messages %}
              <div class="alert {% if message.tags == 'success' %}alert-success{% else %}alert-danger{%
endif %} fade show" role="alert">
                  {{ message }}
              </div>
            {% endfor %}
```

```
        {% endif %}

        <form method="POST" novalidate>
            {% csrf_token %}
            {% for field in form %}
                <div class="form-group">
                    <label for="{{ field.id_for_label }}">{{ field.label }}</label>
                    {{ field }}
                    {% if field.errors %}
                        <div class="error">
                            {{ field.errors|striptags }}
                        </div>
                    {% endif %}
                </div>
            {% endfor %}

            <button type="submit" class="btn">Register</button>
        </form>

        <p class="mt-3">
            Alreadyregistered?
            <a href="{% url 'authority_login' %}">Login here</a>
        </p>
    </div>

</body>
</html>
```

Authority_dashboard.html

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Authority Dashboard</title>

    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 95%;
            margin: 20px auto;
```

```
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    overflow-x: auto;
    text-align: center;
}
h2 {
    text-align: center;
    color: #333;
}
.filter-box {
    margin-bottom: 15px;
    display: flex;
    justify-content: space-between;
    gap: 10px;
}
.filter-box input {
    flex: 1;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 14px;
}

.prioritize-container {
    display: flex;
    justify-content: center;
    margin-bottom: 20px;
}

.prioritize-btn {
    background: linear-gradient(135deg, #28a745, #218838);
    color: white;
    font-size: 18px;
    font-weight: bold;
    padding: 14px 28px;
    border: none;
    border-radius: 50px;
    cursor: pointer;
    transition: all 0.3s ease;
    display: inline-flex;
    align-items: center;
    gap: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    text-decoration: none;
}

.prioritize-btn:hover {
    background: linear-gradient(135deg, #218838, #1e7e34);
    box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
}
```

```
.prioritize-btn::before {
    content: 🚨
    font-size: 20px;
}

.table-container {
    margin-top: 20px;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
    table-layout: fixed;
}
th, td {
    padding: 12px;
    text-align: center;
    border: 1px solid #ddd;
    word-wrap: break-word;
}
th {
    background-color: #007BFF;
    color: white;
}
tr:nth-child(even) {
    background-color: #f9f9f9;
}
tr:hover {
    background-color: #ddd;
}
img {
    width: 60px;
    height: auto;
    border-radius: 5px;
    display: block;
    margin: 0 auto;
}
.priority-high { color: red; font-weight: bold; }
.priority-medium { color: orange; font-weight: bold; }
.priority-low { color: green; font-weight: bold; }

.progress-bar {
    width: 100px;
    height: 15px;
    background-color: #ddd;
    border-radius: 5px;
    position: relative;
    overflow: hidden;
    margin: auto;
}
```

```
        .progress-fill {
            height: 100%;
            background: linear-gradient(135deg, #28a745, #218838);
            border-radius: 5px;
            transition: width 0.5s ease-in-out;
        }

        @media (max-width: 768px) {
            .filter-box {
                flex-direction: column;
                gap: 5px;
            }
            .filter-box input {
                width: 100%;
            }
            table {
                display: block;
                overflow-x: auto;
                white-space: nowrap;
            }
            .prioritize-btn {
                width: 100%;
                justify-content: center;
            }
        }

        .logout-btn {
            background-color:  #dc3545;
            color: white;
            padding:  10px  20px;
            border-radius:  20px;
            text-decoration: none;
            font-weight: bold;
            transition: background-color 0.3s ease;
        }
        .logout-btn:hover {
            background-color: #c82333;
        }
    </style>
</head>
<body>

<div class="container">
    <div style="display: flex; justify-content: flex-end; margin-bottom: 10px;">
        <a href="{% url 'logout' %}" class="logout-btn">Logout</a>
    </div>
    Authority Dashboard - Prioritized Issues</h2>

    <div class="prioritize-container">
        <a href="{% url 'prioritized_issues' %}" class="prioritize-btn">AI Prioritize Issues</a>
            <a href="{% url 'trending_issues' %}" class="prioritize-btn" style="background:  linear-
gradient(135deg, #ff8c00, #ff5e00);">
```

```
        🟠 Trending Issues
      </a>
    </div>

    <div class="filter-box">
      <input type="text" id="searchUser" pla ce ho lder ="🔍 Search by User Name...">
      <input type="text" id="searchLocation" plac eho ld er="📍 Search by Location...">
    </div>

    <div class="table-container">
      {% if reported_issues %}
      <table id="issuesTable">
        <thead>
          <tr>
            <th>User</th>
            <th>Email</th>
            <th>Description</th>
            <th>Location</th>
            <th>Latitude</th>
            <th>Longitude</th>
            <th>Times Reported</th>
            <th>Severity</th>
            <th>Priority</th>
            <th>Image</th>
            <th>Verify Work</th>
            <th>Completion Status</th>
          </tr>
        </thead>
        <tbody>
          {% for issue in reported_issues %}
          <tr>
            <td>{{ issue.user.username }}</td>
            <td>{{ issue.user.email }}</td>
            <td>{{ issue.description }}</td>
            <td>{{ issue.location_name }}</td>
            <td>{{ issue.latitude }}</td>
            <td>{{ issue.longitude }}</td>
            <td>{{ issue.report_count }}</td>
            <td>{{ issue.severity }}</td>
           <td class="{% if issue.priority == 3 %}priority-high{% elif issue.priority == 2 %}priority-
medium{% else %}priority-low{% endif %}">
              {% if issue.priority == 3 %}
                High
              {% elif issue.priority == 2 %}
                Medium
              {% else %}
                Low
              {% endif %}
            </td>
            <td>
              {% if issue.image %}
```

```
                    <img src="{{ issue.image.url }}" alt="Issue Image">
                {% else %}
                    No Image
                {% endif %}
            </td>
            <td>
                {% if issue.work_images %}
                    <img src="{{ issue.work_images.url }}" alt="Work Image">
                {% else %}
                    No Image
                {% endif %}
            </td>
            <td>
                <div class="progress-bar">
                  <div class="progress-fill" style="width: {{ issue.progress_percentage }}%;"></div>
                </div>
                {{ issue.progress_percentage }}%
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
{% else %}
    <p>No issues reported yet.</p>
{% endif %}
</div>
</div>

<script>
    document.getElementById("searchUser").addEventListener("keyup", function() {
        filterTable(0, this.value);
    });

    document.getElementById("searchLocation").addEventListener("keyup", function() {
        filterTable(3, this.value);
    });

    function filterTable(columnIndex, filterValue) {
        let table = document.getElementById("issuesTable");
        let rows = table.getElementsByTagName("tr");

        for (let i = 1; i < rows.length; i++) {
            let cell = rows[i].getElementsByTagName("td")[columnIndex];
            if (cell) {
                let text = cell.textContent || cell.innerText;
                    rows[i].style.display = text.toLowerCase().includes(filterValue.toLowerCase()) ? "" :
"none";
            }
        }
    }
</script>
```

```
</body>
</html>
```

Settings.py

```
import os
import warnings

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# ■ Ignore Warnings in Development
warnings.filterwarnings("ignore", category=UserWarning)

# ■ Secret Key (Use Environment Variable)
SECRET_KEY = os.getenv('DJANGO_SECRET_KEY', 'fallback-secret-key')

# ■ Debug Mode (Disable in Production)
DEBUG = os.getenv('DJANGO_DEBUG', 'True') == 'True'

# ■ Allowed Hosts (Update for Production)
ALLOWED_HOSTS = ['127.0.0.1', 'localhost']

# Database Configuration (PostgreSQL)
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('DB_NAME', 'civicconnect_db'),
        'USER': os.getenv('DB_USER', 'postgres'),
        'PASSWORD': os.getenv('DB_PASSWORD', '555666'), # Change in environment
        'HOST': os.getenv('DB_HOST', 'localhost'),
        'PORT': os.getenv('DB_PORT', '5432'),
    }
}

# Login URL
LOGIN_URL = '/citizen_login/'

# Installed Apps
INSTALLED_APPS = [
    'django.contrib.admin',
    "django_extensions",
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'reports',
]

# Middleware
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'civicconnect.urls'

# Templates
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'reports/templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

# WSGI Application
WSGI_APPLICATION = 'civicconnect.wsgi.application'

#  Static & Media Files
STATIC_URL = "/static/"
STATICFILES_DIRS = [os.path.join(BASE_DIR, "civicconnect/static")]
STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles")

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Allow media files to be served in development
if DEBUG:
    import mimetypes
    mimetypes.add_type("image/svg+xml", ".svg", True)

# ■  Internationalization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Asia/Kolkata'
USE_I18N = True
USE_L10N = True
USE_TZ = True
```

```
# ■ Logging Configuration
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'ERROR',
            'class': 'logging.FileHandler',
            'filename': os.path.join(BASE_DIR, 'django_errors.log'),
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file'],
            'level': 'ERROR',
            'propagate': True,
        },
    },
}

import logging
import warnings

# ■ Suppress specific warnings
warnings.filterwarnings("ignore",  category=UserWarning)

# ■ Suppress Django's development server warnings
logging.getLogger("django").setLevel(logging.ERROR)
logging.getLogger("django.server").setLevel(logging.ERROR)
logging.getLogger("django.request").setLevel(logging.ERROR)
logging.getLogger("django.db.backends").setLevel(logging.ERROR)
```

views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from django.contrib.auth.models import User
from django.db.models import F, Q
from django.views.decorators.csrf import csrf_exempt
from django.db import IntegrityError
from .models import Issue, Officer, ReportedUser
from .forms import CitizenRegistrationForm, AuthorityRegistrationForm
from .ai_prioritization import compute_severity, calculate_priority

# Home Page View
def home(request):
    return render(request, "reports/base.html")
```

```
#  Citizen Login
def citizen_login(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = authenticate(request, username=username, password=password)
        if user:
            login(request, user)
            return redirect("citizen_dashboard")
        else:
            messages.error(request, "Invalid username or password.")
    return render(request, "reports/citizen_login.html")

# Citizen Dashboard
@login_required
def citizen_dashboard(request):
    citizen_issues = Issue.objects.filter(user=request.user)
    return render(request, "reports/citizen_dashboard.html", {"issues": citizen_issues})

#  Track Issue
@login_required
def track_issue(request):
    issues = Issue.objects.filter(user=request.user).order_by('-created_at')
    return render(request, 'reports/track_issue.html', {"issues": issues})

# Get Issue Status
@login_required
def get_issue_status(request, issue_id=None):
    try:
        if issue_id:
            issue = Issue.objects.get(id=issue_id, user=request.user)
        else:
            issues = Issue.objects.filter(user=request.user).values('id', 'title', 'status', 'created_at')
            return JsonResponse(list(issues), safe=False)

        data = {
            "title": issue.title,
            "description": issue.description,
            "severity": issue.severity,
            "date":  issue.created_at.strftime("%Y-%m-%d"),
            "image": issue.image.url if issue.image else None,
            "status": issue.status
        }
        return JsonResponse(data)
    except Issue.DoesNotExist:
        return JsonResponse({"error": "Issue not found"}, status=404)

@login_required
def reported_issues(request):
    # Show all issues the user has ever reported (even duplicates)
    issues = Issue.objects.filter(reported_user_entries__user=request.user).distinct()
```

```
    return render(request, 'reports/reported_issues.html', {'issues': issues})

# Report Issue with Semantic Deduplication + AI Priority
@login_required
def report_issue(request):
    if request.method == "POST":
        user = request.user
        location_name = request.POST.get("location_name", "").strip()
        description = request.POST.get("description", "").strip()
        latitude = float(request.POST.get("latitude"))
        longitude = float(request.POST.get("longitude"))
        image = request.FILES.get("image")

        if not location_name or not description:
            return JsonResponse({"error": "Location and description are required."}, status=400)

        nearby_issues = Issue.objects.filter(
            Q(latitude__range=(latitude - 0.0002, latitude + 0.0002)) &
            Q(longitude__range=(longitude - 0.0002, longitude + 0.0002))
        )

        for issue in nearby_issues:
            similarity = issue.compute_description_similarity(description)
            if similarity >= 0.75:  # 75% semantic similarity threshold
                if ReportedUser.objects.filter(issue=issue, user=user).exists():
                    return JsonResponse({"error": "You have already reported this issue."}, status=400)
                issue.report_count = F("report_count") + 1
                issue.save(update_fields=["report_count"])
                issue.refresh_from_db()
                ReportedUser.objects.create(issue=issue, user=user)
                issue.update_priority()
                return JsonResponse({
                    "message": "Issue already exists. Report count incremented.",
                    "report_count": issue.report_count,
                    "priority": issue.priority_score
                })

        new_issue = Issue.objects.create(
            user=user,
            username=user.username,
            email=user.email,
            description=description,
            location_name=location_name,
            latitude=latitude,
            longitude=longitude,
            image=image if image else None,
            report_count=1
        )
        ReportedUser.objects.create(issue=new_issue, user=user)
        new_issue.update_priority()

        return JsonResponse({"message": "New issue reported.", "issue_id": new_issue.id})
```

```
    return JsonResponse({"error": "Invalid request method."}, status=400)

# Report Issue Page
@login_required
def index(request):
    return render(request, "reports/index.html")

# Authority Login
@csrf_exempt
def authority_login(request):
    if request.method == "POST":
        username = request.POST.get("username")
        password = request.POST.get("password")
        user = authenticate(request, username=username, password=password)

        if user:
            login(request, user)
            messages.success(request, "Login successful!")
            return redirect("authority_dashboard")
        else:
            messages.error(request, "Invalid credentials.")

    return render(request, "reports/authority_login.html")

# Authority Dashboard
@login_required
def authority_dashboard(request):
    issues = Issue.objects.all().order_by('-priority_score', '-report_count')

    # Ensure priorities are freshly recalculated
    for i in issues:
        i.update_priority()

    return render(request, "reports/authority_dashboard.html", {"reported_issues": issues})

# Citizen Registration
def citizen_register(request):
    if request.method == "POST":
        form = CitizenRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.set_password(form.cleaned_data['password'])
            user.save()
            messages.success(request, "Registration successful! Please log in.")
            return redirect('citizen_login')
        else:
            messages.error(request, "Registration failed.")
    else:
        form = CitizenRegistrationForm()

    return render(request, 'reports/citizen_register.html', {'form': form})
```

```
# Authority Registration
def authority_register(request):
    if request.method == "POST":
        form = AuthorityRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.set_password(form.cleaned_data["password"])
            user.save()
            messages.success(request, "Registration successful! Please log in.")
            return redirect("authority_login")
        else:
            messages.error(request, "Registration failed.")
    else:
        form = AuthorityRegistrationForm()

    return render(request, "reports/authority_register.html", {"form": form})

# AI-Prioritized Issues List
@login_required
def prioritized_issues(request):
    calculate_priority()
    issues = Issue.objects.order_by('-priority_score')
    return render(request, 'reports/prioritized_issues.html', {'issues': issues})

# Officer Login
def officer_login(request):
    if request.method == "POST":
        username = request.POST.get("username")
        password = request.POST.get("password")
        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect("officer_dashboard")
        else:
            messages.error(request, "Invalid credentials. Please try again.")

    return render(request, "reports/officer_login.html")

# Officer Dashboard
@login_required
def officer_dashboard(request):
    officer = get_object_or_404(Officer, user=request.user)
    issues = Issue.objects.filter(assigned_officer=officer)
    return render(request, "reports/officer_dashboard.html", {"issues": issues})

# Issue Detail View
@login_required
def issue_detail(request, issue_id):
    issue = get_object_or_404(Issue, id=issue_id)
    return render(request, "reports/issue_detail.html", {"issue": issue})
```

```
# Update Progress View
@login_required
def update_progress(request, issue_id):
    issue = get_object_or_404(Issue, id=issue_id)

    if request.method == "POST":
        progress = request.POST.get("progress_percentage")
        status = request.POST.get("status")
        work_image = request.FILES.get("work_image")

        issue.progress_percentage = int(progress)
        issue.status = status

        if work_image:
            issue.work_images = work_image

        issue.save()
        messages.success(request, "Issue updated successfully!")
        return redirect("officer_dashboard")

    return redirect("officer_dashboard")

#  Logout
def logout_user(request):
    from django.contrib.auth import logout
    logout(request)
    return redirect("home")

# Trending Issues Dashboard
@login_required
def trending_issues(request):
    trending_issues = Issue.objects.order_by('-report_count')[:10]

    # Refresh priority before showing
    for issue in trending_issues:
        issue.update_priority()
```

track_issue.html

```
<h2>Track Your Issue</h2>

<form id="trackIssueForm">

    <input type="number" id="report_id" name="report_id" placeholder="Enter Report ID" required>

    <button type="submit">Track Issue</button>

</form>


<div id="issueDetails" style="display: none;">

    <h3>Issue Details</h3>
```

```
    <p><strong>Title:</strong> <span id="issueTitle"></span></p>
    <p><strong>Status:</strong> <span id="issueStatus"></span></p>
    <p><strong>Reported On:</strong> <span id="issueDate"></span></p>
</div>

<script>
document.getElementById("trackIssueForm").addEventListener("submit", function(e) {
    e.preventDefault();
    let reportID = document.getElementById("report_id").value.trim();

    fetch(`/get_issue_status/${reportID}/`)
    .then(response => response.json())
    .then(data => {
        if (data.error) {
            alert("+ Report not found! Please check the Report ID.");
        } else {
            document.getElementById("issueTitle").innerText = data.title;
            document.getElementById("issueStatus").innerText = data.status;
            document.getElementById("issueDate").innerText = data.date;
            document.getElementById("issueDetails").style.display = "block";
        }
    });
});
</script>
```

5. EXPERIMENTAL SETUP & IMPLEMENTATION

5.1 System Specifications

5.1.1 Hardware Requirements

Minimum:

- Processor: Intel Core i3 / AMD Ryzen 3
- RAM: 4 GB
- Storage: HDD/SSD with sufficient space

Recommended:

- Processor: Intel Core i5 or higher
- RAM: 8 GB or more
- Storage: SSD for better performance

5.1.2 Software Requirements

Minimum:

- OS: Windows 10 / Ubuntu 18.04+ / macOS 10.12+
- Browser: Google Chrome / Firefox
- Backend: Python 3.9+, Django 4.x • Frontend: HTML5, CSS3, JavaScript (ES6)
- Database: PostgreSQL or MySQL
- Others: Postman (API testing), VS Code

Recommended:

- OS: Windows 11 / Ubuntu 22.04 / macOS 14+
- Frontend Frameworks: TailwindCSS / Bootstrap
- Mapping: OpenStreetMap / Google Maps API
- ML Tools: TensorFlow, Scikit-learn
- AI Libraries: SentenceTransformers, XGBoost

5.2 Methodology/Algorithm:

CivicConnect is a full-stack civic issue reporting platform using Django, PostgreSQL, and AI-powered prioritization. It allows real-time issue reporting, officer assignment, and tracking via a web interface.

1. Data Modeling:

- Users: Citizens, Officers, Authorities (modeled using Django's auth system)
- Issues: Stored with attributes like description, location, severity, and priority
- Relationships: Citizens report issues; officers are auto-assigned to high-priority ones

2. Frontend Development (HTML/CSS/JS):

- Responsive UI for citizen and authority dashboards
- Citizens report issues with images, map location, and descriptions
- Dashboards show issue status: Pending → In Progress → Resolved

3. Backend Implementation (Django + PostgreSQL):

- Views handle login, registration, issue submission, officer assignment, and dashboard updates
- Issue deduplication uses semantic similarity via SentenceTransformers
- AI model (XGBoost) predicts severity from description and location

4. AI-Powered Prioritization:

- Severity is predicted using a trained ML model
- Priority score is calculated based on severity + number of unique reports

5. Issue Verification & Tracking:

- Authorities can manually verify or edit reports
- Citizens can track status updates in real-time
- Officers update progress with images and comments

6. Deduplication Logic:

- Identifies similar issues based on location proximity and semantic text similarity
- Prevents spamming and focuses efforts on real, unique civic problems

6 . RESULTS

1. Platform Implementation

CivicConnect was successfully developed and deployed, enabling real-time civic issue reporting, semantic deduplication, officer coordination, and progress tracking. It fosters data-driven decision-making and responsive governance.

2. Features & Functionality

- Citizen Dashboard: Submit issues with image, geolocation, and description.
- Officer Dashboard: View assigned issues, update progress, upload field photos.
- Authority Dashboard: View all reports, filter by severity or status, track performance.
- AI-Powered Prioritization: Uses ML to dynamically rank issues by importance.
- Deduplication Mechanism: Prevents duplicates using location + semantic similarity.
- Real-Time Status Updates: Tracks progress from submission to resolution.

3. Performance Evaluation

Parameter	Performance
Scalability	Scales with concurrent reports using PostgreSQL indexing
Security	Hashed passwords, role-based access, input validation
Reliability	Stable backend/API interactions, low downtime
AI Accuracy	Severity prediction accuracy: ~92–95%
Usability	Simple UI, map integration, and real-time feedback

7 . CONCLUSION & FUTURE SCOPE

Conclusion:

CivicConnect addresses the urban governance gap by offering a centralized, intelligent civic issue reporting and monitoring system. Through AI-based prioritization, semantic deduplication, and live dashboards, it improves transparency, accelerates response time, and enhances citizen satisfaction.

Future Scope:

1. IoT Sensor Integration: Auto-detect road, drainage, or light issues via sensors.
2. Mobile App Launch: Android/iOS support for field use and broader access.
3. Real-Time Analytics Dashboard: For city planning and heatmap visualization.
4. Multilingual Interface: Improve adoption in diverse regions.
5. Smart Notifications: Citizens get auto-updates as issue status changes.
6. Officer Performance Reports: Track workload and response efficiency.
7. Integration with e-Governance Portals: Auto-sync with existing municipal records.

.

8. REFERENCES

[1] S. J. Heikkinen, "Citizen-Centric Smart Cities: A Review of Digital Public Service Platforms," *International Journal of Smart City Applications*, vol. 12, no. 3, pp. 201–210, 2020.

[2] R. Patel and K. Shah, "AI-Powered Incident Management in Urban Governance," *Journal of Urban Technology and Innovation*, vol. 9, no. 4, pp. 85–92, 2021.

[3] V. R. Gupta and A. Sharma, "Design and Development of a Civic Issue Reporting System Using Geolocation and Image Processing," *International Journal of Computer Applications*, vol. 175, no. 15, pp. 14–20, 2020.

[4] T. Iyer, S. Mishra, and D. Nair, "AI-Driven Prioritization Models for Civic Complaint Systems," in *Proc. IEEE Conf. on Smart Governance and Urban Intelligence (SGUI)*, Hyderabad, India, 2022, pp. 98–104.

[5] M. R. Deshmukh and P. K. Jain, "A Django-Based Municipal Grievance Redressal System with Real-Time Dashboard," *International Journal of Web Engineering*, vol. 8, no. 2, pp. 47–55, 2023.

[6] A. Banerjee, S. Ghosh, and L. Fernando, "Semantic Similarity-Based Civic Issue Deduplication Using Sentence Transformers," in *Proc. 2023 IEEE Intl Conf. on AI for Social Good (AISG)*, Mumbai, India, pp. 72–79.

[7] N. S. Rao and M. B. Reddy, "Geospatially Enabled Civic Complaint Applications: A Case Study of Indian Smart Cities," *International Journal of Geoinformatics and Public Policy*, vol. 5, no. 1, pp. 60–67, 2021.