A PROJECT REPORT

on

Mero Samasya: A Local Issue Reporting and Resolution Platform

Submitted To:

Department of Information Technology

Central Campus of Technology

Hattisar, Dharan

In partial fulfilment of the requirements for the degree of Bachelors of Information Technology (BIT)

Submitted By:

Aayush Dhungel [BIT 614/078]

Jeevan Poudel [BIT 619/078]

Sanjeev Shrestha [BIT 626/078]

[November, 2025]

SUPERVISOR RECOMMENDATION

This is to recommend that **Mr. Aayush Dhungel**, Symbol No. BIT 614/078, **Mr. Jeevan Poudel**, Symbol No. BIT 619/078 and **Mr. Sanjeev Shrestha**, Symbol No. BIT 626/078, has carried out Project Work (BIT404) on the topic of "Mero Samasya: A Local Issue Reporting and Resolution Platform" in partial fullment of the requirements for the degree of Bachelors of Information Technology (BIT) under my supervision in the Department of Information Technology, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal. To the best of my knowledge, this work has not been submitted for any other degree. They have fullled all the requirements introduced by Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the submission of the project work for the partial fulllment of Bachelor of Information Technology.

Mr. Bikash Soni
Supervisor
Department of Information Technology
Central Campus of Technology
Dharan-14, Sunsari

DECLARATION

This project work entitled "**Mero Samasya: A Local Issue Reporting and Resolution Platform**" is being submitted to the Department of Information Technology, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the partial fulfillment of the requirement to the project work in Bachelor of Information Technology (BIT). This project work is carried out by me under the supervision of Mr. Bikash Soni T.U., Department of Information Technology, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal.

This work is original and has not been submitted earlier in part or full in this or any other form to any university or institute, here or elsewhere, for the award of any degree.

Aayush Dhungel [BIT 614/078]
Jeevan Poudel [BIT 619/078]
Sanjeev Shrestha [BIT 626/078]
Department of Information Technology
Central Campus of Technology
Dharan-14, Sunsari

CERTIFICATE OF APPROVAL

This project work entitled "**Mero Samasya: A Local Issue Reporting and Resolution Platform**" by Mr. Aayush Dhungel [BIT 614/078], Mr. Jeevan Poudel [BIT 619/078] and Mr. Sanjeev Shrestha [BIT 626/078] under the supervision of Mr. Bikash Soni in the Department of Information Technology, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), is hereby submitted for the partial fulllment of the Bachelor of Information Technology (BIT) degree. This report has been accepted and forwarded to the Exam Section, Office of the Dean, Institute of Science and Technology, Tribhuvan University, Nepal for the legal procedure.

Mr. Bikash Soni
Supervisor
Department of IT
Central Campus of Technology
IOST, Tribhuvan University

Mr. Balabhadra Bhandari
Program Director/HOD
Department of IT
Central Campus of Technology
IOST, Tribhuvan University

External Examiner
Mr. Pradip Khatiwada
Mechi Multiple Campus
IOST, Tribhuvan University

Internal Examiner
Name :
Central Campus of Technology

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our supervisor, Mr. Bikash Soni, for his invaluable guidance, unwavering support, and remarkable patience throughout our project work. His commitment to fostering our academic and personal growth has been truly inspiring, and the insights we have gained under his mentorship will remain with us always.

We extend our sincere appreciation to the faculty members of the Department of Information Technology at CCT, Dharan for their insightful feedback, constructive suggestions, and encouragement from the outset of this project. We are equally grateful to our peers and colleagues, whose intellectual discussions and critical input have greatly enriched our work. The serene and vibrant campus, nestled amidst lush greenery at the foothills of Dharan sub-metropolitan city, has provided an inspiring backdrop for our collaborative endeavors.

We are deeply thankful to our families and close friends, who have supported us through every challenge and consistently encouraged us to pursue our goals with dedication and passion.

Finally, we are profoundly honored by the support and contributions of everyone involved in this journey. Your guidance, encouragement, and belief in our potential have been instrumental to the success of this project, and for that, we are eternally grateful.

We always had persistent reminiscence.

Aayush Dhungel [BIT 614/078]
Jeevan Poudel [BIT 619/078]
Sanjeev Shrestha [BIT 626/078]

ABSTRACT

This report outlines the development and implementation of Mero Samasya, a digital platform designed to streamline local issue reporting and resolution in communities. The system bridges the gap between citizens and local authorities by integrating a web and mobile application, leveraging a microservices architecture for scalability and modularity. The citizen interface enables users to report issues (e.g., potholes, water leakages) with media uploads and GPS-based location capture using Leaflet and OpenStreetMap data and provide feedback on resolutions. The authority dashboard facilitates issue tracking, status updates, and direct communication with citizens. Built using an Iterative Waterfall approach, the platform employs React for the frontend, Node.js for backend microservices, Flutter for mobile application, MongoDB for data storage and secure algorithms like JWT for authentication, bcrypt for password hashing and Reverse Geocoding Algorithm for fetching possible human-readable addresses near the coordinates. Mero Samasya aims to enhance community engagement, improve governance transparency, and optimize issue resolution efficiency, offering a modern, user-friendly solution for local governance.

Keywords: Microservices, Next.js, Node.js, Flutter, MongoDB, Leaflet, Tailwind CSS, OpenStreetMap, JWT, bcrypt, Iterative Waterfall, Community Engagement

CONTENTS

LIST OF ACRONYMS AND ABBREVIATIONS

ACM Association for Computing Machinery

API Application Programming Interface

CSCW Computer Supported Cooperative Work

CSS Cascading Style Sheets

DFD Data Flow Diagram

ER Entity Relationship

GPS Global Positioning System

HMAC Hash-based Message Authentication Code

ICT Information and Communication Technology

IEEE Institute of Electrical and Electronics Engineers

JWT JSON Web Token

KPI Key Performance Indicator

NPR Nepalese Rupees

REST Representational State Transfer

SDK Software Development Kit

SHA Secure Hash Algorithm

SSRN Social Science Research Network

TOCHI Transactions on Computer-Human Interaction

UI User Interface

UX User Experience

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction

Traditional issue reporting mechanisms, such as in-person complaints or unstructured communication channels, often result in delayed responses, lack of transparency, and growing citizen disillusionment[1]. These methods typically result in significantly delayed responses, primarily due to bureaucratic hurdles, limited accessibility, and the absence of systematic tracking and follow-up. Citizens often have to physically visit offices multiple times for complaints to be heard and resolved, incurring travel and opportunity costs that can be prohibitive, especially for marginalized communities. For example, in formal complaint systems like those studied in rural India, complainants reported extensive travel, food expenses, and time loss just to attend hearings, with repeated visits needed for a single complaint[2]. Addressing these inefficiencies, **Mero Samasya** is a digital platform designed to facilitate the reporting and resolution of local issues, thereby fostering a more direct and transparent relationship between citizens and their local authorities. In an increasingly digital world, the platform leverages technology to empower citizens to take an active role in the development and maintenance of their communities. By providing a user-friendly interface for reporting problems such as infrastructure damage, sanitation issues, and public service disruptions, "Mero Samasya" aims to streamline communication and enhance the efficiency of local governance. The platform is built on the principle of community engagement, recognizing that active citizen participation is crucial for the effective functioning of local government.

At its heart, the core functionality of "Mero Samasya" revolves around a simple yet powerful workflow that encourages collaborative problem-solving: citizens can quickly report issues via the app or website, complete with photos, location data, and descriptions for clarity. Local authorities, in turn, access a dedicated dashboard that allows them to track, assign, manage, and resolve the reported problems efficiently, with real-time updates shared back to the reporters and community.

This transparent, end-to-end process not only ensures that pressing issues are addressed in a timely manner but also promotes greater accountability, as users can monitor progress and provide feedback on resolutions. Ultimately, by democratizing the issue-reporting process and integrating community input, "Mero Samasya" strengthens civic bonds, reduces bureaucratic delays, and contributes to more responsive and inclusive governance aligning with the Digital Nepal framework[3].

1.2 Problem Statement

Local issue reporting in many communities, especially within Nepali municipalities, is notoriously plagued by severe inefficiencies rooted in outdated, manual bureaucratic processes. These archaic systems not only delay the response time to pressing community concerns but also obscure the entire complaint resolution process behind a veil of opacity that frustrates citizens and erodes their faith in local governance.

Specifically in Nepal, local governance faces endemic systemic failures that aggregate these inefficiencies into three critical dimensions:

- **Information Asymmetry**: Citizens rarely know what happens after submitting complaints. With no clear feedback system, issues often disappear into bureaucracy, fostering mistrust and discouraging engagement.
- **Geographic Fragmentation**: The absence of standardized location data makes it hard for authorities to map and prioritize problems. This leads to delays, duplication, and overlooked areas, a serious hurdle in Nepals diverse geography.
- **Accountability Gaps**: A decentralized, paper-based system lacks centralized auditing to monitor response times and resolution quality. As a result, inefficiency, negligence, and corruption often go unchecked.

Under these conditions, local governments grapple with constrained resources, political instability, and capacity limitations, exacerbating the dysfunctional interplay between citizens' expectations and service delivery. Public participation mechanisms such as citizen forums or public hearings exist on paper but are frequently underutilized, underpowered, or overshadowed by bureaucratic inertia and informal political interests.

1.3 Objectives

The primary objective of our proposed system/application is to empower residents by providing a transparent, real-time, and reliable platform for reporting and accessing critical information about local community issues. Along with it some of the main objectives of our application are mentioned below:

- To provide a simple interface for citizens to report issues with photos and GPS in under 2 minutes.
- To offer real-time dashboards for authorities to track issue status (pending, in progress, resolved).
- To allow users to set severity levels (Low to Critical) to help prioritize resources.

- To ensure cross-device accessibility, with an emphasis on mobile phones used by most citizens.

1.4 Scope and Limitation

The "Mero Samasya" platform aims to streamline local issue reporting and resolution, fostering effective communication between citizens and local authorities. Developed as a final-year project, it integrates a web and mobile application using a microservices architecture to ensure scalability and modularity. The initiative focuses on the following key areas:

1. **Citizen Reporting Interface:** Allowing users to submit reports of community issues such as broken roads, waste management problems, or waterlogging, with support for text descriptions, photos, and location details.

2. **Issue Tracking and Transparency:** Providing authorities with a dashboard to track reported issues, update their status (e.g., pending, in-progress, resolved), and maintain accountability.

3. **Prioritization of Issues:** Enabling citizens to assign severity levels (Low, Medium, High, Critical) to assist authorities in resource allocation and prioritization.

4. **Public Engagement and Communication:** Enhancing communication between citizens and local authorities by providing real-time updates and status visibility on issues.

5. **Data Analytics for Authorities:** Offering insights into trends, frequently reported issues and response times to help improve decision-making and service delivery.

The system is a partial study developed to fulfill the requirements of a Bachelor's degree, constrained by a limited time period. Despite significant efforts to design a robust and user-friendly platform, certain limitations have impacted the study, outlined as follows:

1. **Geographical Scope Limitation:** The study is designed for a generic community context and may not account for specific geographical or infrastructural variations across different wards or municipalities, limiting its adaptability to diverse urban or rural settings.

2. **Limited Policy Integration:** The platform does not comprehensively address local governance policies or regulations regarding issue reporting and resolution, focusing primarily on technical functionality rather than legal or administrative frameworks.

3. **Prototype Constraints:** As a student project, the system is a prototype with limited scalability testing. The microservices architecture, while designed for scalability, has not been tested under high user loads or across multiple municipalities, potentially affecting performance in real-world deployment.

1.5 Development Methodology

For the development of "Mero Samasya", we adopted the Iterative Waterfall model to balance structure with flexibility, suitable for a final-year student project constrained by a 10-week timeline. This model combines sequential phases with iterative cycles, allowing refinement of features such as issue reporting, GPS integration, and authority dashboards based on feedback from faculty and mock users. The iterative approach was chosen to accommodate evolving requirements and ensure delivery of a functional prototype that enhances community engagement and governance transparency within a short timeframe.

The Iterative Waterfall model involves the following phases, illustrated in Figure 1.1:

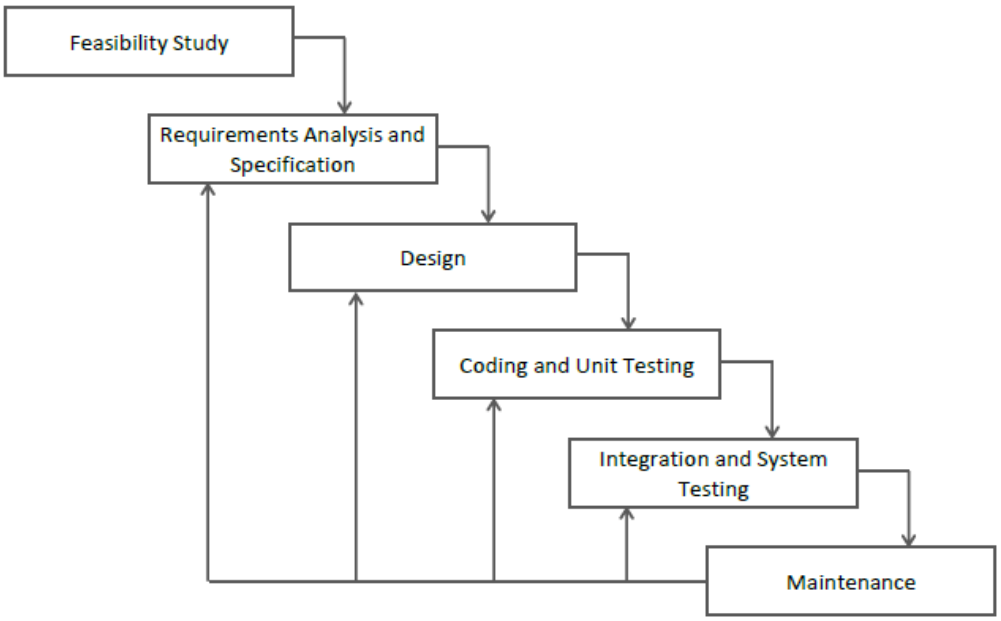


Figure 1.1: Iterative Waterfall Model

1. **Feasibility Study** (1 week): Assess technical, operational, and economic viability, identifying tools (e.g., Next.js, Node.js, MongoDB) and constraints for a microservices-based platform.

2. **Requirements Analysis and Specification** (1 week): Gather and refine user needs (e.g., issue reporting) through stakeholder consultations, iterated to finalize requirements.

3. **Design** (2 weeks): Develop the microservices architecture (e.g., User, Issue Services) and user interfaces, with iterative refinements to ensure scalability and usability.

4. **Coding and Unit Testing** (4 weeks): Implement microservices and frontend components, with unit testing (e.g., Jest) to ensure functionality, conducted in short iterative cycles.

5. **Integration and System Testing** (1.5 weeks): Integrate microservices via RESTful APIs and conduct system testing to validate features like GPS mapping.

6. **Maintenance** (0.5 week): Address post-deployment issues and refine the prototype based on user feedback, ensuring stability for the pilot deployment.

This approach enabled rapid, incremental development within 10 weeks, with each iteration refining components like the Leaflet-based GPS interface and delivering a reliable and user-friendly system tailored to community needs.

1.6 Report Organization

The report is structured into several chapters to systematically present the study. The organization is as follows:

- Chapter 1: Introduction : It provides an overview of the project, problem statement, objectives, scope and limitation and development methodology.
- Chapter 2: Background Study and Literature Review : It discusses the fundamental theories, and terminologies of the project. It also discusses about the relevant studies, systems related to the project.
- Chapter 3: System Analysis : It explains the requirements, architecture, and the conducts the feasibility study for the project.
- Chapter 4: System Design : It explains the blueprint of the system, database design, system flow, and describes the algorithms used in the project.
- Chapter 5: Implementation and Testing : It covers the implementation of the modules, technologies used, conducts unit and system testing and analyzes the result.
- Chapter 6: Conclusion and Future Recommendation : It summarizes key findings, discusses challenges and suggests future improvements.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Civic technology encompasses digital tools that promote public good by facilitating citizen-government interactions through e-governance and ICT frameworks. This field focuses on developing platforms that empower citizens to engage effectively with government and participate in public life[4].

Central to civic tech are web-based issue tracking systems that manage citizen reports through databases and user interfaces. These systems integrate GPS technology for precise location accuracy and media upload capabilities for evidence documentation. The technical foundation draws from software engineering practices, particularly issue tracking methodologies adapted from tools like Jira for civic applications.

Development of these platforms requires consideration of functional requirements (use cases, workflows) and non-functional requirements (scalability, accessibility). Digital platforms for citizen engagement have become increasingly prevalent, offering functionalities from participatory budgeting to policy discussion forums. These platforms leverage widespread mobile adoption and internet connectivity to bridge the digital divide and foster inclusive civic participation[5].

The "Mero Samasya" project is positioned within this civic technology landscape, harnessing these capabilities to address local governance challenges and enhance citizen-government communication through digital solutions.

2.2 Literature Review

The literature on civic technology highlights the growing importance of digital platforms for community issue reporting, emphasizing their role in fostering social innovation and citizen participation in governance. A systematic review of existing civic technology solutions identifies web and mobile applications as the dominant tools for urban planning and policy-making processes, demonstrating significant benefits in terms of increased civic engagement while also revealing persistent challenges related to long-term sustainability and accessibility[6].

Several existing platforms and research projects have informed the theoretical foundation

and practical development of "Mero Samasya." Internationally, platforms such as SeeClickFix,[7] and FixMyStreet[8] have established successful models for citizen government interaction. SeeClickFix allows comprehensive issue reporting, tracking, and feedback mechanisms, effectively bridging the communication gap between citizens and governments through transparent workflows. Similarly, FixMyStreet enables UK users to report municipal problems like potholes and street maintenance issues, integrating directly with local councils to facilitate efficient resolutions.

In the United States, platforms like SeeClickFix and PublicStuff[9] have successfully demonstrated the viability of centralized systems for reporting non-emergency local issues, showing that simple and accessible reporting mechanisms can significantly improve municipal service delivery and increase citizen satisfaction. Research in the field of smart cities has highlighted the critical importance of citizen-centric technologies in modern urban governance[10]. The smart city concept emphasizes leveraging data and technology to enhance residents' quality of life, positioning citizen reporting platforms as essential components of this technological ecosystem.

However, the literature also reveals significant challenges facing civic technology initiatives. Research indicates that while these platforms effectively enhance civic skills and awareness among users, they frequently encounter accessibility issues that may exclude certain demographic groups[11]. Studies consistently recommend employing participatory design methodologies to ensure inclusive platform development, emphasizing the importance of involving diverse community stakeholders in the design process. These findings directly align with Mero Samasya's goals of creating an accessible, inclusive, and effective civic engagement platform tailored to local governance needs.

CHAPTER 3: SYSTEM ANALYSIS

3.1 System Analysis

3.1.1 Requirement Analysis

i. **Functional Requirements**

a. User Registration and Authentication

- Separate registration forms for citizens and local authority officials with secure login/logout functionality.
- Role-based access control to distinguish between citizen and authority user privileges.

b. Issue Reporting and Management

- Citizens can submit new issue reports with comprehensive details including:
  - Issue title and detailed description
  - Category classification
  - Automated geolocation (GPS) integration
  - Media attachments (photos)
- Real-time submission and validation of reported issues.

c. Authority Dashboard and Issue Management

- Comprehensive dashboard for local authority officials to view and filter reported issues.
- Issue assignment capabilities to appropriate departments or personnel.
- Advanced filtering options by category, status, priority and date.

d. Status Tracking and Updates

- Authorities can update issue status through defined workflow stages (e.g., Pending, In Progress, Resolved).
- Real-time status tracking visible to both reporters and community members.

e. Notification and Communication System

- Automated real-time notifications to citizens regarding updates to their reported issues.

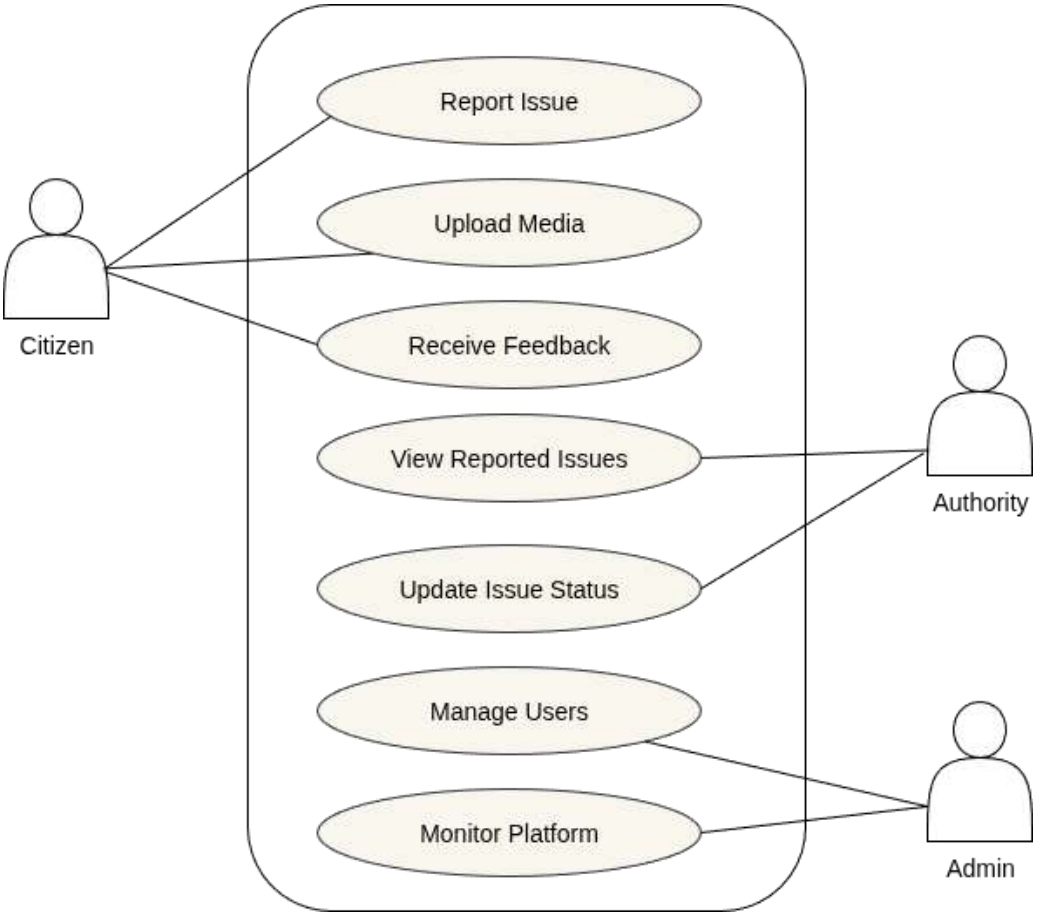


Figure 3.1: Use Case Diagram

ii. **Non-Functional Requirements**

a. Usability

- Highly intuitive and user-friendly interfaces designed to accommodate diverse user groups, ensuring smooth navigation and minimal learning curve.

b. Reliability

- Robust and dependable platform performance with consistently high availability and negligible downtime, fostering user trust and confidence.

c. Security

- Strong data protection mechanisms with secure handling, encryption, and rigorous user authentication to safeguard sensitive information.

d. Scalability

- Flexible and future-ready architecture capable of efficiently managing a growing number of users, data, and reports without compromising performance.

3.1.2 Feasibility Analysis

i. **Technical:** Following technologies have been chosen for the project:

   a. Next.js - For the development of the front-end user interface and client-side functionality, the React-based framework Next.js has been utilized, providing server-side rendering capabilities and optimized performance for web applications.

   b. Node.js (Express.js) - For the development of necessary Application Programming Interface (APIs) and other backend related works, the JavaScript runtime Node.js with the Express.js framework has been used to handle server-side logic and API endpoints.

   c. MongoDB - To record the necessary information, data or records the system is intended to store, the NoSQL database technology MongoDB has been implemented for this project, providing flexible document-based storage capabilities.

   d. Flutter - The project includes a cross-platform mobile application for users to interact with the complaint system. For the development of the mobile application, Flutter framework has been utilized with its comprehensive Software Development Kit (SDK) which provides all the necessary tools and APIs to build both Android and iOS applications.

   e. Leaflet.js with OpenStreetMap - For GPS integration and location-based services, Leaflet.js library combined with OpenStreetMap data has been implemented to provide accurate mapping and geolocation functionality within the application.

   f. Cloudinary API - For secure media storage and transcoding capabilities, Cloudinary's cloud-based service has been integrated to handle image and video uploads, processing, and delivery with optimized performance.

   g. GitHub - For the project, GitHub technology has been used to manage and for the version control of the application, ensuring collaborative development and code repository management.

   h. VS Code - Primary editor for coding and development tasks across all project components.

ii. **Operational:** The platforms operational success is closely tied to both citizen adoption and integration into local government operations. If effectively promoted and supported, Mero Samasya has the potential to be fully operational and impactful in cities like Dharan, where active citizen participation and responsive governance are increasingly in demand. With proper backing from municipal authorities, the platform could evolve into a cornerstone of digital civic engagement, streamlining com-

munication between citizens and local government offices.

To ensure smooth adoption, comprehensive training programs for government staff will be necessary so they can efficiently handle and respond to citizen reports. At the same time, widespread awareness campaigns will empower citizens to use the system confidently, creating a culture of accountability and transparency. With these measures in place, adoption rates are expected to rise significantly, enabling Mero Samasya to become a transformative governance tool. Over time, its successful application in Dharan could set a national benchmark for how technology-driven solutions can strengthen trust, responsiveness, and efficiency in local governance.

iii. **Economic:** As a final-year project, the development of Mero Samasya demands only minimal financial costs, since it leverages open-source tools, student effort, and community resources. For a 10-week development cycle, the estimated expenses in Nepalese Rupees (NPR) are largely limited to development time, cloud hosting, and introductory training for users. This makes the platform not only affordable to build but also practical to pilot in a city like Dharan, where resource-efficient solutions are essential.

The potential economic benefits, however, far exceed the modest investment. By improving efficiency for local authorities, reducing manual complaint handling, and introducing the possibility of a municipal subscription model, the system ensures long-term sustainability. Furthermore, with its ability to scale across other municipalities, Mero Samasya could evolve into a cost-effective governance tool for Nepal, generating value through enhanced transparency, citizen trust, and operational savings.

iv. **Schedule:** The project lasted for 10 weeks, starting from the 3rd week of May and concluding in the last week of July. The complete schedule of the systems development is presented in Figure 3.2, illustrated with the help of a Gantt chart.

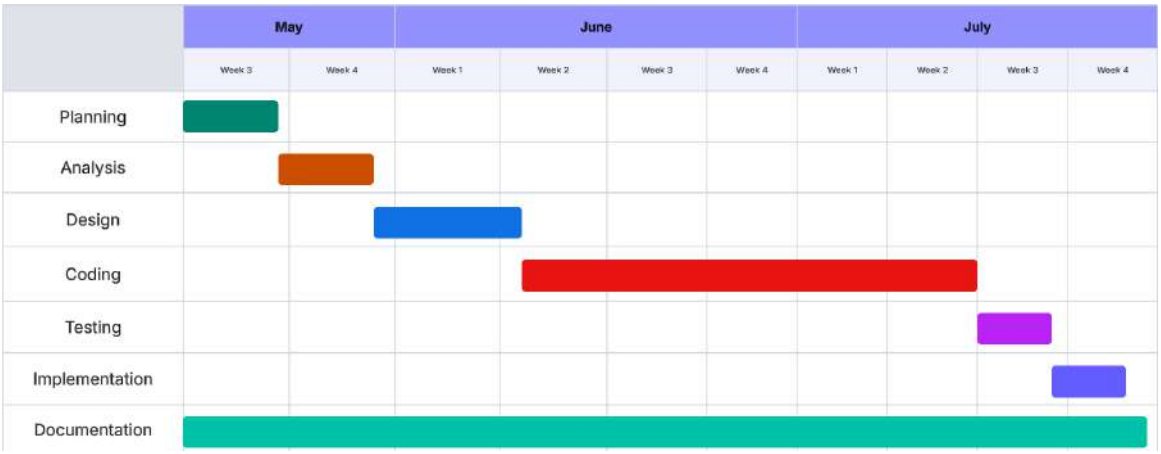


Figure 3.2: Gantt Chart demonstrating Schedule Feasibility

3.1.3 Analysis (Structured)

The analysis of "Mero Samasya" presents the system's overall structure and workflow using modeling tools. Data Modeling with ER Diagrams defines the entities, relationships, and database structure, while Process Modeling with Data Flow Diagrams (DFDs) illustrates how information flows between users, processes, and data stores. These models together provide a clear foundation for the systems design and implementation.

i. **Data Modelling:** The Entity-Relationship (ER) Diagram of Mero Samasya represents the core entities involved in the system, such as citizens, authorities, admin and reported issues, along with their relationships. It provides a clear structure of how data is stored, managed, and interconnected within the database. This diagram ensures that the systems data requirements are well-organized and supports efficient retrieval and scalability for future expansion.

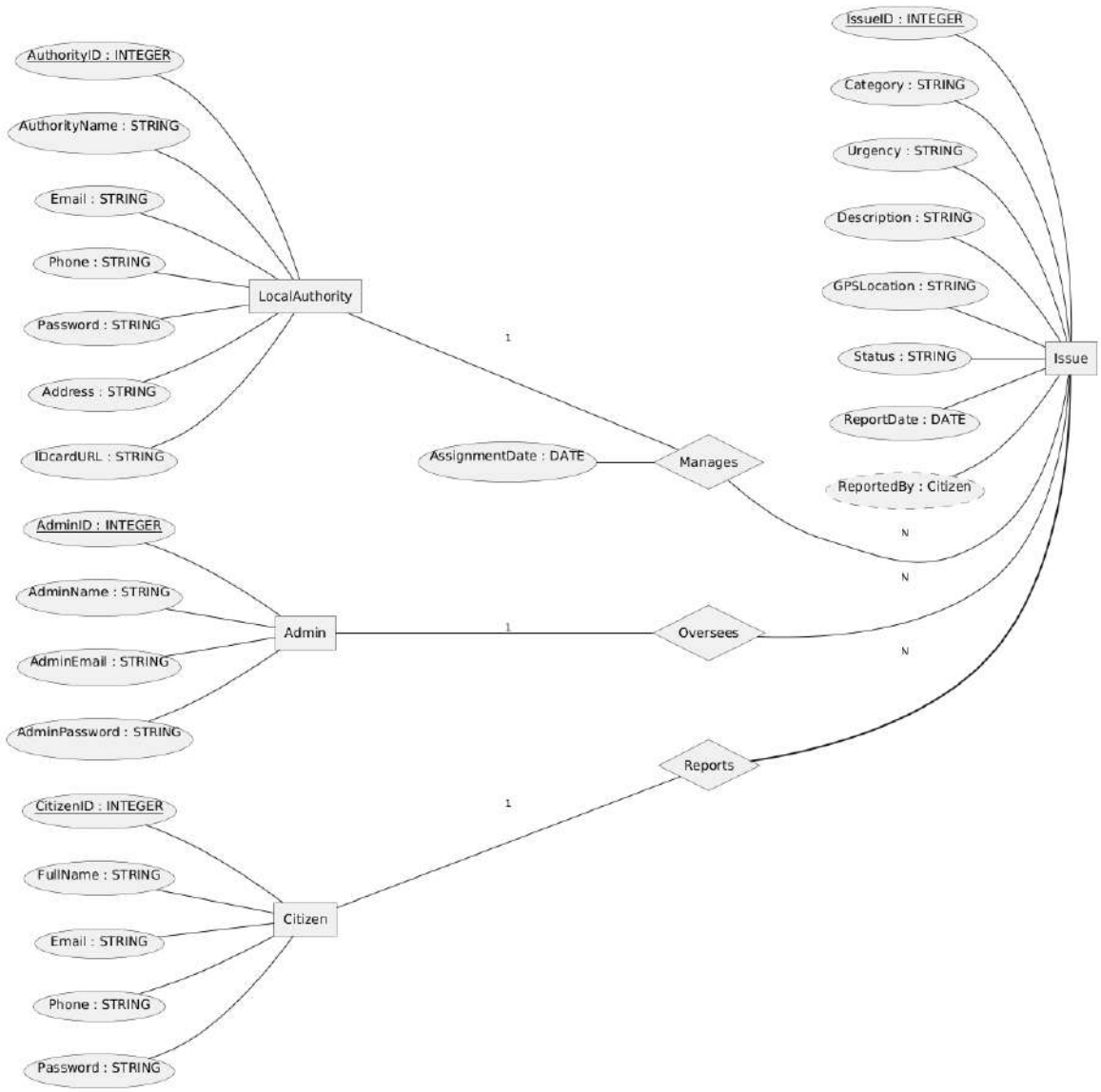


Figure 3.3: Data modelling using ER Diagram

ii. **Process modelling:** The Data Flow Diagram (DFD) of Mero Samasya illustrates the movement of data between external users, internal processes, and data stores. It highlights how issues are reported, processed by authorities, and resolved through structured workflows. By modeling these interactions, the DFD provides a transparent view of the system's functionality and helps identify areas for optimization and efficiency improvements.

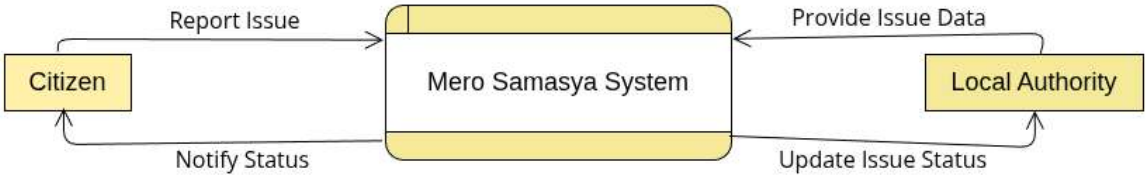


Figure 3.4: DFD Level 0 (Context) Diagram

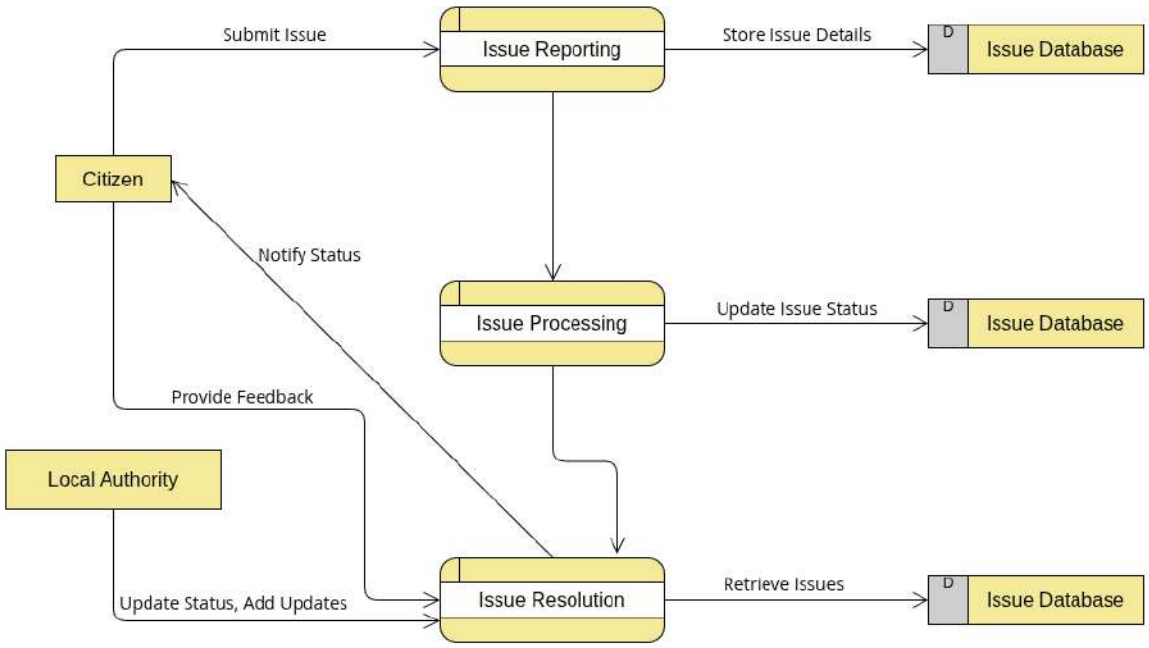


Figure 3.5: DFD Level 1 Diagram

CHAPTER 4: SYSTEM DESIGN

4.1 Design (Structured)

4.1.1 Database Design: MongoDB Documents

MongoDB Database Schema Design

The MongoDB database for the Mero Samasya platform will consist of several collections, each representing a different aspect of the system.

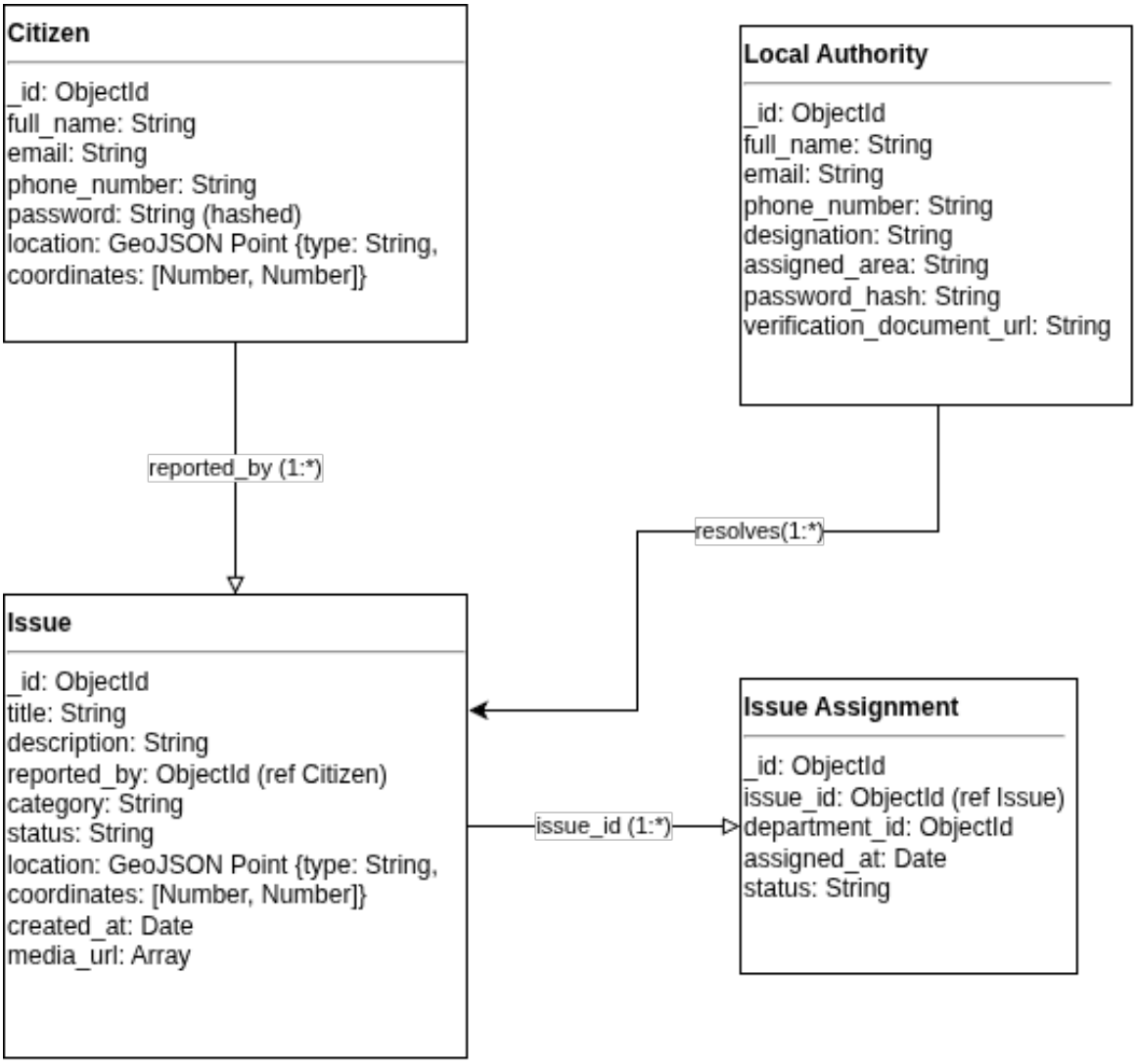


Figure 4.1: Database Schema Design

4.1.2 Forms and Report Design

This section details the design of the user interface components responsible for data input (Forms) and data output (Reports). The primary goal is to ensure a user-friendly, intuitive, and efficient experience for all user roles - Citizens, Local Authority, and Admin.

i. **Forms Design:** The forms are the primary means by which users interact with and enter data into the Mero Samasya system. Each form is designed for a specific purpose, with clear labels, logical flow, and robust input validation to ensure data integrity.

(a) **Citizen-Facing Forms**

- **Citizen Sign-Up Form**
  - **Purpose:** To allow new residents to create a personal account.
  - **Fields:**
    * Full Name: Text, Required.
    * Email Address: Text, Required, Email Format Validation.
    * Phone Number: Text, Required, Numeric/Format Validation.
    * Password: Password field, Required, Strength Validation (min. 8 characters, one number, one special character).
    * Confirm Password: Password field, Required, Must match the Password field.
  - **Mockup:**

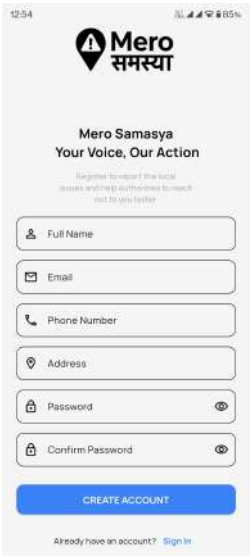


Figure 4.2: Citizen Sign-Up Form

- **Issue Reporting Form**
  - **Purpose:** To enable citizens to report a local problem. This is the core data entry form of the system.
  - **Fields:**
    * Issue Title: Text, Required, Max 100 characters.
    * Description: Text Area, Required, Detailed explanation of the problem.
    * Category: Dropdown list (e.g., Broken Roads, Garbage, Water Leakage), Required.
    * Urgency Level: Dropdown list (options: Low, Medium, High and Critical), Required.
    * Location: Auto-detected via GPS.
    * Media Upload: File Upload button, Allows multiple photos (file type and size validation).
  - **Mockup:**

Figure 4.3: Issue Reporting Form

(b) **Local Authority Forms**

- **Local Authority Sign-Up Form**
  - **Purpose:** To allow official personnel to register for an authority account, which requires verification.
  - **Fields:**
    * Full Name: Text, Required.
    * Official Email Address: Text, Required, Email format validation.

* Phone Number: Text, Required.
* Password & Confirm Password: As per citizen sign-up.
* Address: Text, Required. Relevant to the authority assigned area.
* Department Served: Dropdown or Text, Required.
* Verification Document Upload: File Upload, Required (JPG, PNG formats, size limit).

– **Mockup:**

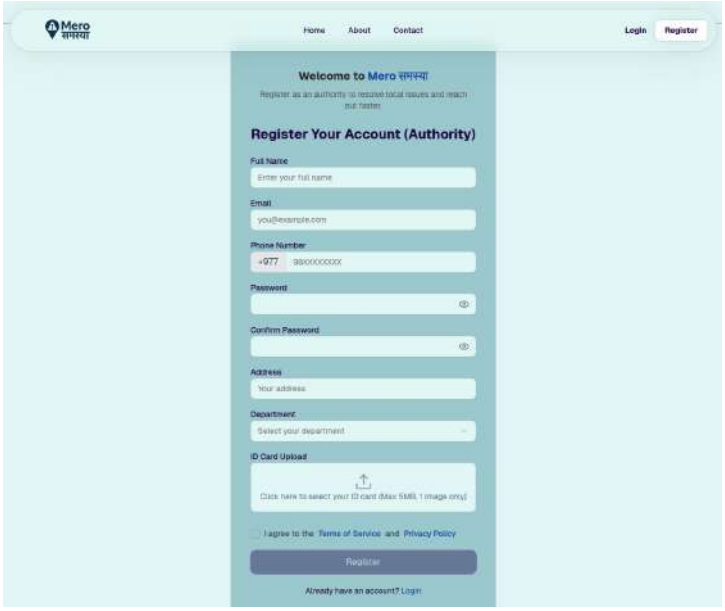


Figure 4.4: Local Authority Sign-Up Form

ii. **Report Design:** Reports are the primary output of the system, transforming raw data into structured, actionable information for analysis and decision-making.

(a) **Citizen-Facing Reports**

- **Purpose:** To provide a transparent, real-time view of all reported issues in the community.
- **Format:** An interactive map view and a filterable list view.
- **Data Points:** Issue Title, Category, Location (on map), Status, Date Reported.
- **Features:** Users can filter by status, category, and date. Clicking on an issue reveals more details.

(b) **Local Authority Reports**

- **Purpose:** To serve as the main operational view for officials to track and manage all incoming issues efficiently.

- **Format:** A sortable and filterable data table displayed at the top (e.g., "New Issues Today", "Pending Issues", "Resolved Issues").
- **Data Points:** Issue ID, Title, Reporter Name, Ward/Area, Status, Date Reported, Assigned Department.
- **Features:** Advanced filtering (by status, date range), sorting by priority and search functionality.

(c) **Administrator Reports**

- **Purpose:** For the system administrator to monitor platform activity and user management.
- **Format:** Data tables and activity logs.
- **Data Points:**
  - List of pending Local Authority verification requests.
  - Number of new user sign-ups over a period.
  - Total issues reported and resolved system-wide.
  - Log of significant admin actions (e.g., account verifications, deletions).

4.1.3 Interface and Dialogue Design

i. **Interface Design:** The interface is designed to be clean, intuitive, and responsive, ensuring a seamless experience across different devices (desktop and mobile). The design language is consistent across the platform but tailored to the specific needs of each user role.

(a) **Citizen Interface (Web and Mobile App)**

- **Layout:** The primary interface for citizens is a dashboard-centric layout. On a mobile device, this features a main view with a bottom navigation bar. On the web, it uses a top navigation bar.
- **Navigation:**
  - Main Dashboard/Home: The default screen showing the map and list of local issues.
  - Report New Issue: A highly visible button that provides immediate access to the issue reporting form.
  - My Issues: A dedicated section where a user can track the status of the issues they have personally reported.
  - Profile: Allows users to view/edit their profile information and log out.
- **Key UI Elements:**

- Issue Map: Uses standard badges to show the status of issues. Badges are color-coded by status (e.g., Red for Pending, Yellow for In Progress, Green for Resolved).
- Issue Cards: In the list view, each issue is represented by a "card" containing the issue title, an image thumbnail, status and date reported.

- **Wireframe:**

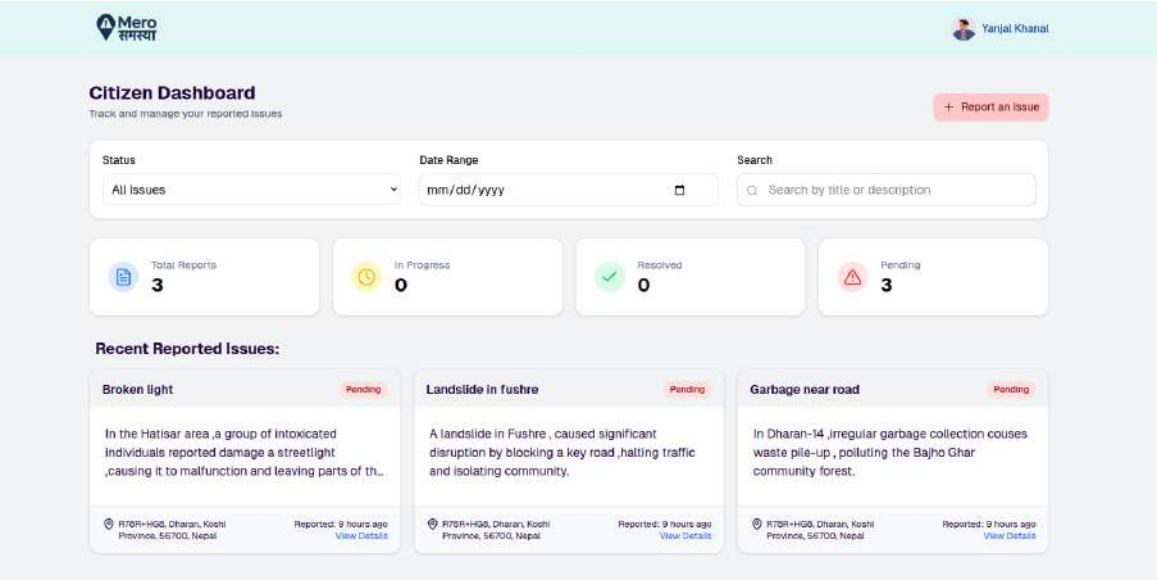


Figure 4.5: Citizen Dashboard (Web)

Figure 4.6: Citizen Dashboard (Mobile App)

(b) **Local Authority Interface (Web-based Dashboard)**

- **Layout:** A professional, data-centric dashboard layout with a fixed sidebar for navigation on the left and a large main content area on the right. The top of the main content area features key statistics (KPIs).
- **Navigation:**
  - Dashboard: An overview page with summary statistics and charts.
  - All Issues: A comprehensive table view of all issues within their jurisdiction.
  - Reports: Access to the performance and summary reports module.
  - Profile/Logout: For viewing their account details and logging out.
- **Key UI Elements:**
  - KPI Cards: At the top of the dashboard, cards display critical numbers like "New Issues Today", "Total Pending Issues" and "Average Resolution Time."
  - Total Issues Reported per month: A bar graph to display all issues reported at each month of the year.
- **Wireframe:**

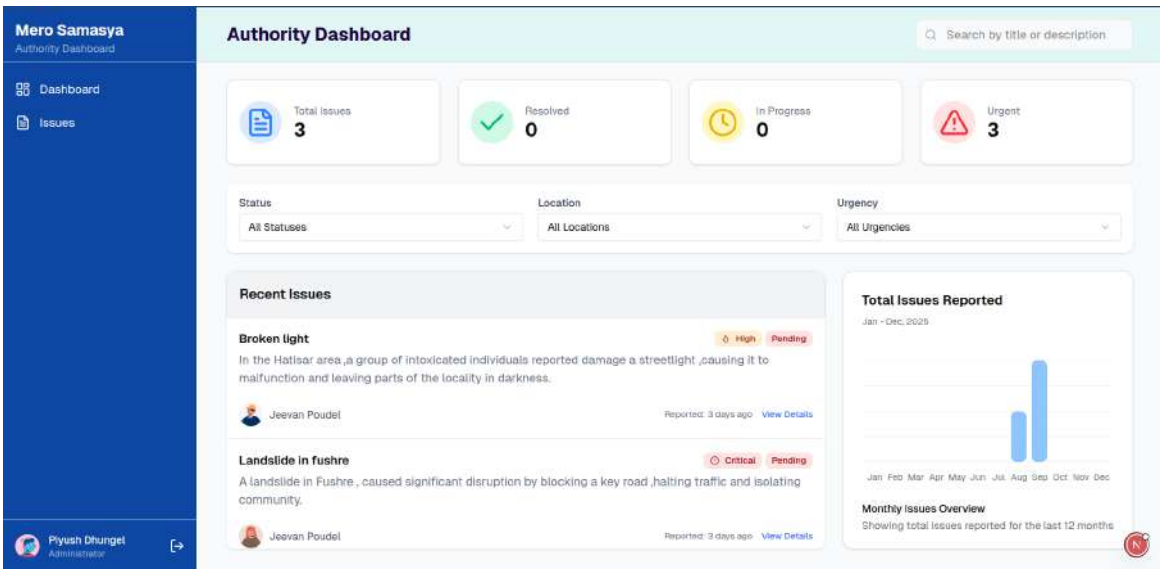


Figure 4.7: Authority Dashboard

ii. **Dialogue Design:** Dialogue design defines the flow of interaction, ensuring that users can achieve their goals logically and efficiently. This includes success paths, error handling, and system feedback.

(a) **Dialogue Flow: Citizen Reports a New Issue**

- **Step 1: Initiation**

- **User Action:** Clicks the "Report New Issue" button from the main dashboard.
- **System Response:** The system displays the "New Issue Reporting Form."

- **Step 2: Data Entry**
  - **User Action:** Fills in the issue title, description, selects a category, and uploads photos.
  - **System Response:** The system provides real-time validation. For example, if the title is empty, the field border turns red with a message "Title is required." The GPS location is auto-detected and shown on an embedded map.
- **Step 3: Submission**
  - **User Action:** Clicks the "Submit Issue" button.
  - **System Response:** The system displays a loading indicator while the data and media are uploaded.
- **Step 4: Confirmation**
  - **User Action:** (Waits for system processing).
  - **System Response:** Upon successful submission, the system displays a success message: "Success! Your issue has been reported and assigned ID #582." The user is then redirected back to their "My Issues" page, where the new issue appears at the top with the status "Pending."

(b) **Dialogue Flow: Local Authority Resolves an Issue**

- **Step 1: Login & Navigation**
  - **User Action:** A Ward Officer logs into the Local Authority portal.
  - **System Response:** The system displays the main dashboard with an overview of issues.
- **Step 2: Issue Selection**
  - **User Action:** The officer navigates to the "All Issues" table and clicks on a high-priority pending issue.
  - **System Response:** The system opens the "Issue Detail View," showing all information submitted by the citizen, including photos and the exact map location.
- **Step 3: Action & Update**
  - **User Action:** The officer assesses the issue and changes its status from "Pending" to "In Progress" using a dropdown menu. They may add an internal note.

- **System Response:** The system saves the change and automatically triggers a notification to the reporting citizen. It also displays a confirmation toast message: "Status updated to In Progress."

- **Step 4: Resolution**
  - **User Action:** After the issue is physically resolved, the officer opens the issue again, changes the status to "Resolved," and adds a public comment like, "The pothole has been filled."
  - **System Response:** The system saves the final status, updates the public dashboard, and sends a final notification to the citizen, which includes a prompt to provide feedback on the resolution.

4.2 Algorithm Details

The following algorithms are implemented to address security needs:

i. **JSON Web Token (JWT) for Authentication**:

- *Purpose*: Securely authenticates users (citizens, authorities, admins) across microservices, ensuring authorized access to features.
- *Mechanism*:
  (a) Upon login, the User Service verifies credentials and generates a JWT (Header, Payload, Signature) using HMAC-SHA256 and a secret key.
  (b) The payload includes user ID and expiration time (e.g., 1 day).
  (c) The client stores the JWT and includes it in API requests Authorization header.
  (d) Microservices verify the JWTs signature and payload to grant access.
- *Contribution*: Ensures secure, stateless authentication, supporting role-based access (e.g., citizens report issues, authorities update issue status) and enhancing user trust, aligning with transparency goals.

ii. **bcrypt for Secure Password Storage**:

- *Purpose*: Protects user passwords in MongoDB, preventing unauthorized access if the database is compromised.
- *Mechanism*:
  (a) During registration, bcrypt generates a random salt and hashes the password using a work factor (e.g., 12) to produce a 60-character hash.
  (b) The hash, including the salt, is stored in MongoDB.

(c) On login, the provided password is hashed with the stored salt and compared to the stored hash for verification.

- *Contribution*: Enhances security by safeguarding credentials, supporting community trust and secure authentication for JWT, critical for a platform handling sensitive data.

iii. **Reverse Geocoding Algorithm[12]**:

- *Purpose*: Enable citizens to pinpoint the exact location of an issue on a map during the reporting process, and allow authorities to exactly locate the points and addresses on the map based on the provided latitude and longitude coordinates.
- *Mechanism*:
  (a) Initialize a Leaflet map in the reporting form, defaulting to a city center or the users current location (via geolocation, if permitted).
  (b) Add a click event listener to the map, placing a draggable marker at the clicked location.
  (c) Update the markers coordinates dynamically as it is dragged.
  (d) Upon marker placement or dragging:
    i. Use a reverse geocoding service to fetch possible human-readable addresses near the coordinates.
    ii. If no suitable addresses are found, prompt the user to adjust the marker.
    iii. Select the most appropriate address (e.g., the one closest to the marker).
  (e) Automatically populate the location field in the form with the selected address, allowing the user to edit it if necessary.
  (f) On form submission, extract the markers latitude and longitude, and include them along with the location field data in the issue data sent to the backend via a POST request.
  (g) For authorities viewing reported issues, display a Leaflet map initialized with the stored latitude and longitude, placing a marker at the location and using reverse geocoding to confirm or display the human-readable address.
- *Contribution*: Enhances the precision and convenience of issue reporting by providing an interactive map for location selection, automatic address filling, and validation based on issue type, while also enabling authorities to accurately visualize and locate reported issues on the map for efficient resolution.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Tools Used

a. **Front-end tools**

i. React/Next.js

ii. Tailwind CSS

iii. Leaflet using OpenStreetMap[13]

b. **Backend tools**

i. Node.js (Express)[14]

c. **Mobile Application**

i. Flutter[15]

d. **Database**

i. MongoDB (Mongoose)[16]

ii. Cloudinary (media storage)

e. **Authentication and Security**

i. JSON Web Token (JWT)[17]

ii. bcrypt[18]

f. **Code-editor**

i. Visual Studio Code[19]

g. **Report editor**

i. LaTeX

h. **Version Control**

i. GitHub

i. **UI/UX Design**

i. Figma

j. **Data visualization tool**

i. ChartJS[20]

5.1.2 Implementation Details of Modules

The modules are the units and the partition of a project to make the development process easier. The partitioning of the modules also eases the debugging process, which helps in finding any bugs and errors present in the system. The modules used in this system are explained below with their implementation methods:

a. **User Module**

- **Purpose:** Handles user registration for citizens and authorities, authentication, profile management, and role-based access to ensure secure interaction with the local issue reporting platform.
- **Implementation:** Developed using Node.js and Express for backend authentication logic and API endpoints. User registration forms (with fields like name, email, phone, and verification for authorities) are implemented using React components. Profile management utilizes Mongoose for database operations with MongoDB. Authentication is handled through JWT tokens for secure API access, with bcrypt for password hashing.

b. **Issue Module**

- **Purpose:** Manages core issue reporting functionality including media upload, GPS location tagging using Leaflet and OpenStreetMap, issue description, and an upvoting system for community prioritization.
- **Implementation:** Backend implemented using Node.js microservices with file upload handling via Cloudinary for media attachments. GPS coordinates are captured through browser geolocation API and Leaflet maps, stored as GeoJSON in MongoDB. Frontend designed using React for responsive issue forms and display components.

c. **Authority Module**

- **Purpose:** Provides a dedicated dashboard for authorities to view reported issues, update status (Pending, In Progress, Resolved), and communicate directly with citizens.
- **Implementation:** Dashboard developed using Node.js backend with role-based access control. Real-time issue visualization implemented using React components integrated with Leaflet for map displays.

5.2 Testing

5.2.1 Test Cases for Unit Testing

Unit testing is a foundational practice in software engineering to ensure individual components function correctly in isolation. For the Mero Samasya project-a civic issue reporting and management platformunit testing is critical to validate core functionalities such as user authentication, issue submission, authority dashboard operations, and error handling.

a. **Authentication Module**

Table 5.1: Unit Test Cases for User Authentication

Test ID	Description	Input Data	Output	Status
1	Validate citizen sign-up with correct credentials	Name: John Doe Email: john@example.com Phone: 9876543210 Password: Secure@123 ConfirmPassword: Secure@123	User record created; email verification sent	Pass
2	Reject sign-up with duplicate email	Email:john@example.com (already registered)	Error:Email already exists.	Pass
3	Validate authority login with correct credentials	Email:ward1@municipality.gov.np Password:Admin@2024	Dashboard loaded; token valid	Pass
4	Reject login with incorrect password	Email:ward1@municipality.gov.np Password:WrongPass123	Error:Invalid credentials.	Pass
5	Test password hashing	Password:Secure@123 (hashed with BCrypt)	Hash verified successfully	Pass

b. **Issue Reporting Module**

Table 5.2: Unit Test Cases for Issue Reporting

Test ID	Description	Input Data	Output	Status
1	Submit issue with valid media (image)	Title: "Broken Road" Description: "Pothole near School" Image: "road_damage.jpg" (3MB, JPEG) GPS: "27.7000, 85.3167"	Issue #1001 created; image URL stored	Pass
2	Reject oversized media upload	Image: "leakage.jpg" (6.2MB, exceeds 5MB limit)	Error: "File size exceeds limit."	Pass
3	Test empty description field	Title: "Water Leak", Description: "", GPS: "27.7172, 85.3240"	Error: "Description required."	Pass
4	Validate GPS coordinates	GPS: "999.0, 999.0" (invalid coordinates)	Error: "Invalid location."	Pass

c. **Authority Dashboard Module**

Table 5.3: Unit Test Cases for Authority Dashboard

Test ID	Description	Input Data	Output	Status
1	Update issue status to "Resolved"	Issue ID: #1001, Status: "Resolved",	Status updated; notification sent	Pass
2	Test real-time notification	Action: Comment on issue #1001 Message: "Work in progress."	Notification received	Pass
3	Filter issues by status	Filter: "Pending" (3 pending issues in DB)	3 issues displayed	Pass
4	Test unauthorized access	Role: "Citizen", Attempt to access /authority/dashboard	Redirect to login	Pass

5.2.2 Test Cases for System Testing

The system testing phase of Mero Samasya was conducted after module integration to verify functionality, accuracy, and efficiency against project specifications. Real user data was used to simulate interactions, uncovering issues not found during earlier unit tests.

From the citizens side, testing began with account registration and login, followed by reporting issues with descriptions, media uploads, and location selection. Citizens could also track statuses to work smoothly and securely.

For authorities, testing covered registration, login, and dashboard access to manage complaints, update statuses and communicate with citizens. Overall, system testing confirmed the platform met user needs, handled real-time data effectively, and remained robust across citizen and authority use cases.

5.3 Result Analysis

Throughout the development, implementation, and refinement of Mero Samasya, comprehensive research and analysis were conducted, including a detailed review of existing systems and relevant technologies. Although numerous challenges arose during developmentsuch as integrating real-time GPS location tracking, these were successfully overcome through iterative adjustments and leveraging extensive online resources.

The project incorporated both a web application and a mobile app to serve different user roles effectively. The web app, primarily aimed at local authorities and administrators, enables comprehensive management of reported issues, status tracking, and generation of insightful reports. The admin dashboard features real-time visualization tools such as charts for monitoring issue volume, resolution timelines, and citizen feedback, supporting informed decision-making.

The mobile app focuses on citizen engagement, offering an intuitive and user-friendly interface for reporting local problems by submitting descriptions and media attachments with automatic GPS tagging. Citizens can also view ongoing issue statuses and provide feedback on resolution quality, fostering community involvement.

Authentication mechanisms ensure secure access for all user types, with role-based permissions safeguarding sensitive data and operations. Communication channels between citizens and authorities are integrated within both apps, enabling transparency and accountability.

Overall, Mero Samasya performed as expected, demonstrating robust functionality across platforms, seamless data synchronization through APIs between the web and mobile interfaces, and satisfactory user experience. The system effectively bridges the gap between citizens and local governance, facilitating faster and more transparent resolution of community issues.

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 Conclusion

"Mero Samasya" presents a significant opportunity to improve local governance and community engagement through the strategic use of technology. The platform successfully bridges the gap between citizens and local authorities, fostering transparency, accountability, and participatory governance. By providing a transparent and efficient platform for issue reporting and resolution, the project aims to empower citizens while digitizing the complaint reporting and resolution process to ensure issues are managed efficiently.

The implementation of modern mobile app frameworks, cloud services, and AI-based classification makes the system highly scalable and adaptive, enhancing the accountability of local authorities and improving service delivery and community trust. The successful deployment of this platform has the potential to serve as a model for other municipalities seeking to leverage technology for better public service delivery, ultimately contributing to the creation of more responsive and well-maintained communities through enhanced participatory governance.

6.2 Future Recommendations

The "Mero Samasya" platform represents only the beginning of what could become a revolutionary transformation in digital governance. Future enhancements present unprecedented opportunities to create a truly comprehensive and intelligent civic engagement ecosystem:

a. **Integration with Official Government Systems:** Integrating the platform with the existing databases and workflows of local government bodies to further streamline the issue resolution process.

b. **Data Analytics and Reporting:** Implementing a data analytics module to generate reports on the types of issues being reported, the time taken for resolution, and citizen satisfaction levels. This data can provide valuable insights for local authorities to improve their services.

c. **AI-Powered Issue Categorization:** Utilizing artificial intelligence and machine learning to automatically categorize reported issues and assign them to the relevant departments, reducing manual effort.

d. **Gamification Features:** Introducing gamification elements such as points and badges for active citizen participation to further encourage community engagement.

REFERENCES

[1] T. D. Beshi and R. Kaur, "Public trust in local government: Explaining the role of good governance practices," *Public Organization Review*, vol. 20, no. 2, pp. 337–350, 2020.

[2] C. Kumar and M. R. Sharan, "Complaint resolution systems: Experimental evidence from rural india," tech. rep., University of Chicago, Harris School of Public Policy, 2024. Available: https://chinmayakumar.com/research/jmp.pdf.

[3] Government of Nepal, "Digital nepal framework: Unlocking nepal's growth potential," tech. rep., Ministry of Communication and Information Technology, 2019. Available: https://drc.gov.np/storage/backend/pages/resources/others/D8lp6S0TBu0kqwXB7V90hB9aodF4v6qTLGzUvN7M.pdf.

[4] UNDP Asia-Pacific, "Civic tech for transparent, inclusive, and accountable governance," 2025. Available: https://www.undp.org/asia-pacific/civic-tech.

[5] R. Abbasov, "The role of participatory budgeting in enhancing citizen engagement: A comprehensive analysis," *SSRN Electronic Journal*, 2025. Available: https://ssrn.com/abstract=5150297.

[6] J. Saldívar, C. Parra, M. Alcaraz, R. Arteta, and L. Cernuzzi, "Civic technology for social innovation: A systematic literature review," *Computer Supported Cooperative Work (CSCW)*, vol. 28, no. 2, pp. 169–207, 2019.

[7] B. Berkowitz and J.-P. Gagnon, "Seeclickfix empowers citizens by connecting them to their local governments," *Democratic Theory*, vol. 4, no. 1, pp. 121–124, 2017.

[8] "Fixmystreet." https://www.fixmystreet.com/. [Online; accessed 28-Aug-2025].

[9] Wikipedia contributors, "PublicStuff — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/wiki/PublicStuff, 2025. [Online; accessed 28 Aug, 2025].

[10] K. Townsend and W. Bank, "Amplifying Voices, Strengthening Impact: A Proposal for Civic Tech and Social Accountability." World Bank, Washington, D.C., 2025. [Online; accessed 28-Aug-2025; PDF; 42 pp.].

[11] D. Gooch, M. Barker, L. Hudson, R. Kelly, G. Kortuem, J. van der Linden, and M. Petre, "Amplifying Quiet Voices: Challenges and Opportunities for Participatory Design at an Urban Scale." Revised submission to ACM Transactions on ComputerHuman Interaction (TOCHI), 2018. [Online; accessed 28-Aug-2025; PDF; 37 pp.].

[12] Google Maps Platform Documentation, "Reverse geocoding (address lookup) request and response." https://developers.google.com/maps/documentation/geocoding/requests-reverse-geocoding, 2025. Accessed: 2025-11-01.

[13] J. Cheng, B. Schloerke, B. Karambelkar, Y. Xie, and G. Aden-Buie, *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2025. R package version 2.2.3.

[14] T. Holowaychuk and O. Foundation, "Express.js back end web application framework for node.js." Wikipedia, accessed Sept. 3, 2025, 2025.

[15] A. N. Jazi, "Low-code flutter application development solution," in *Proceedings of the 2024 IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, 2024. IEEE, DOI:10.1145/3652620.3688330.

[16] "Mongoose (mongodb) javascript library for working with mongodb." Wikipedia, accessed Sept. 3, 2025. Provides schemabased solutions for MongoDB and Node.js.

[17] M. B. Jones, J. Bradley, N. Sakimura, *et al.*, "Json web token (jwt)." IETF Proposed Standard, RFC 7519, 2015.

[18] N. Provos and D. Mazières, "bcrypt: A future-adaptable password scheme." USENIX Association, presented June 10, 1999, 1999.

[19] E. Gamma and Microsoft, "Visual studio code." Wikipedia, accessed Sept. 3, 2025, 2025.

[20] N. Downie and C. Team, "Chart.js." Wikipedia, accessed Sept. 3, 2025, 2025.

APPENDIX

Table 8.1: Log of visits to supervisor

Date	Time	Purpose of visit	Duration	Feedback	Next Step
2082/02/02	12 PM	Discussion on project idea and requirement gathering.	40 Minutes	Supervisor approved the project idea.	Prepare flowchart and algorithm for approval.
2082/02/20	2 PM	Presentation of design and ER diagram.	1 hour	Refine the system architecture.	Update the ER diagram
2082/03/14	2 PM	Review of initial Implementation.	1 Hour	Suggested testing edge cases.	Test the registration module for all possible edge cases
2082/05/05	3 PM	Review the Report.	1 Hour	Suggestions for report formatting.	Make corrections on the report and prepare for mid-defence.

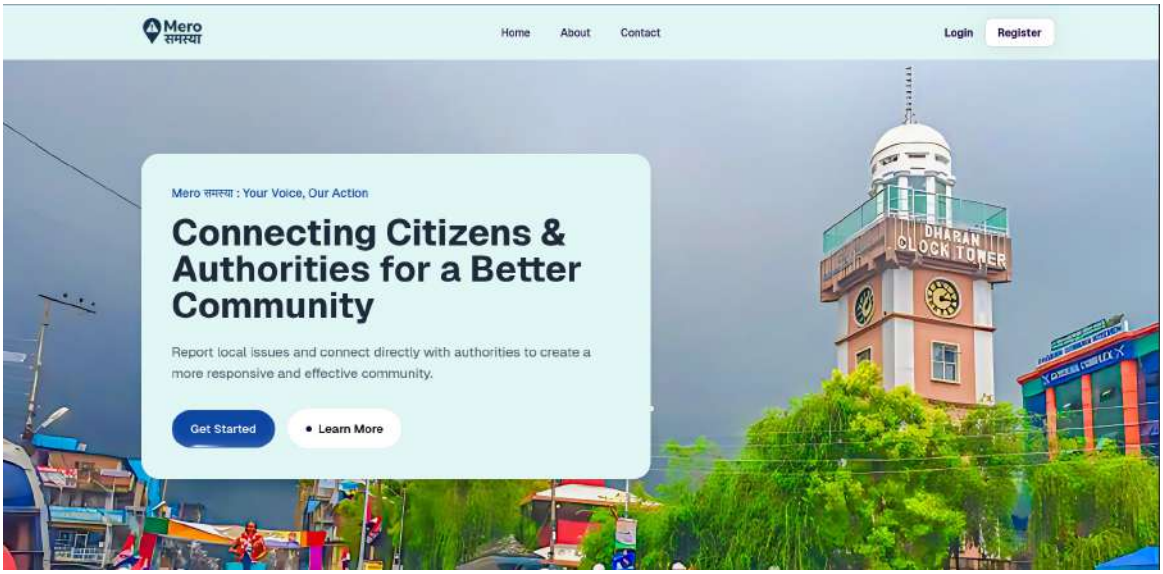


Figure 8.1: Landing Page

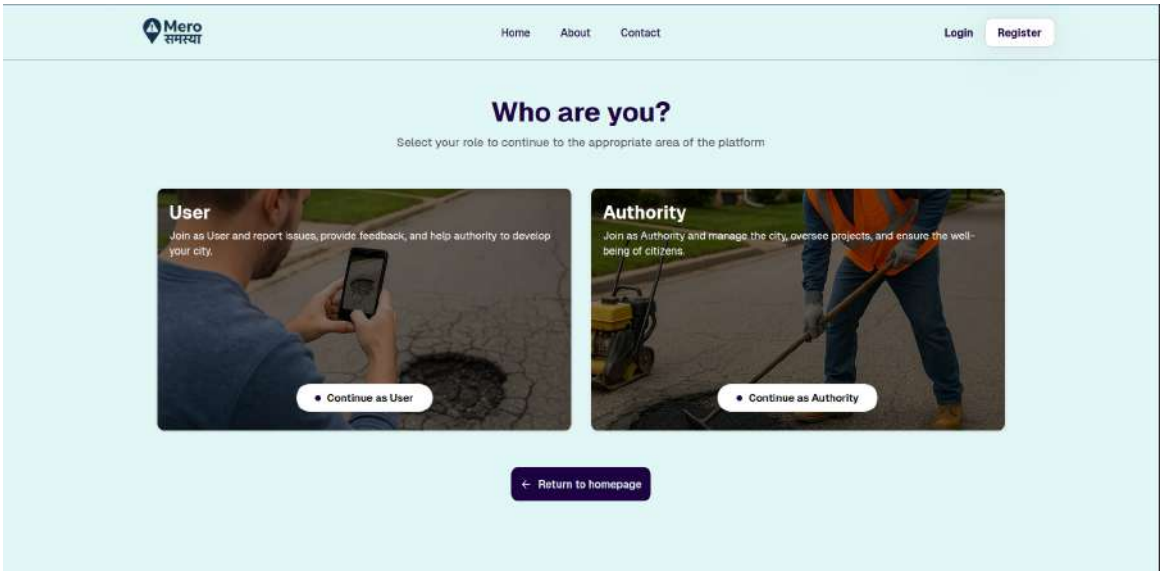


Figure 8.2: Role Selection Page

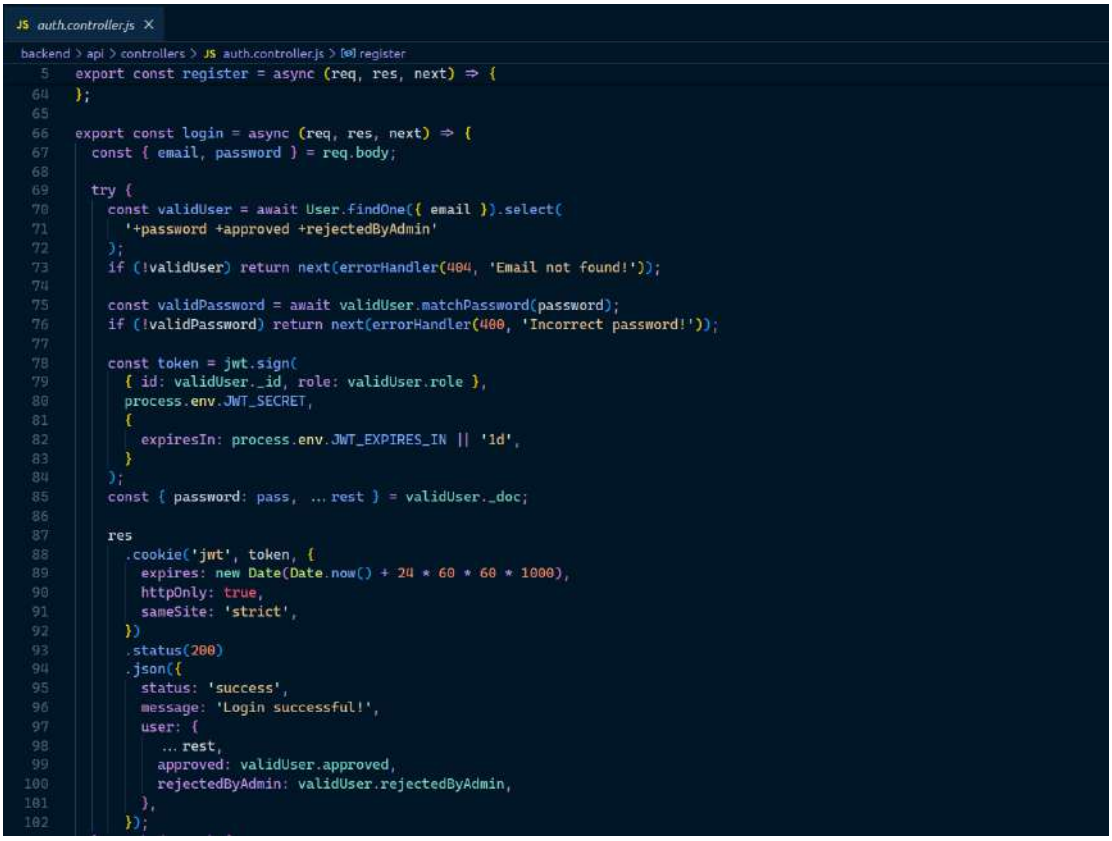
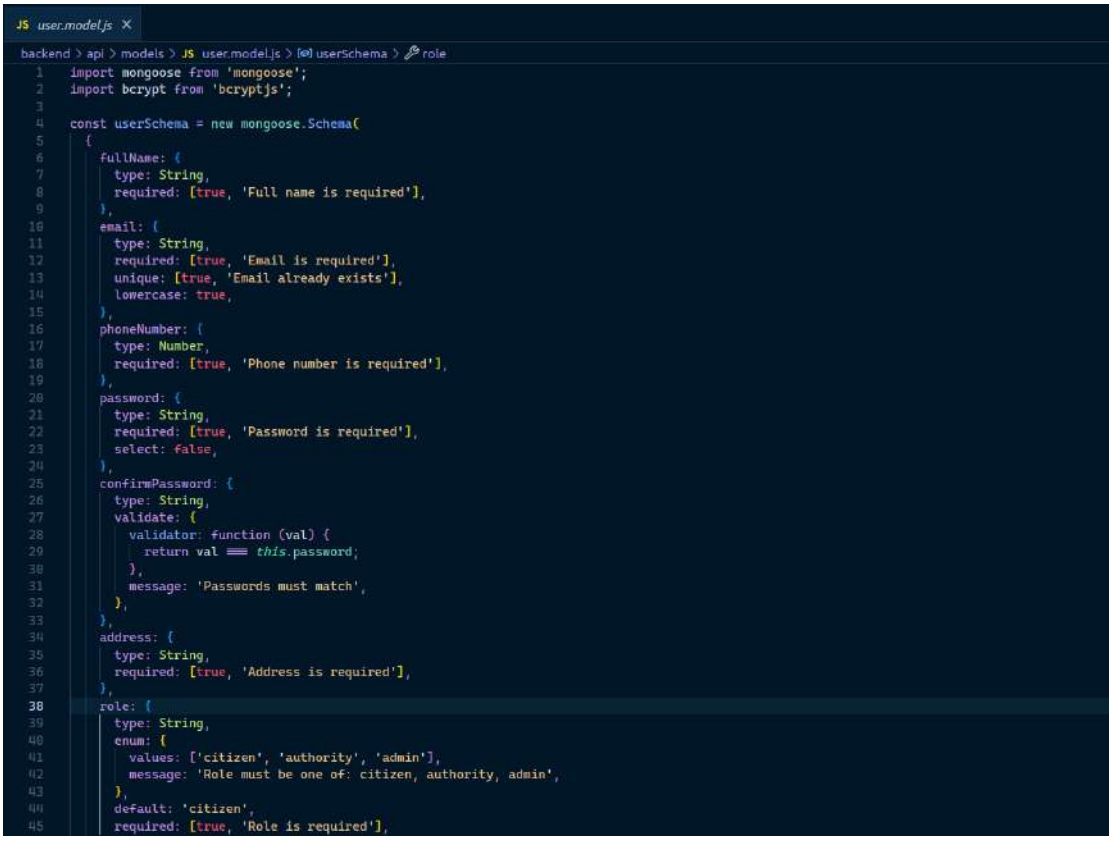


Figure 8.3: Auth Controller

Figure 8.4: Report Controller



```
JS user.model.js ×
backend > api > models > JS user.model.js > [@] userSchema > role
import mongoose from 'mongoose';
import bcrypt from 'bcryptjs';

const userSchema = new mongoose.Schema(
  {
    fullName: {
      type: String,
      required: [true, 'Full name is required'],
    },
    email: {
      type: String,
      required: [true, 'Email is required'],
      unique: [true, 'Email already exists'],
      lowercase: true,
    },
    phoneNumber: {
      type: Number,
      required: [true, 'Phone number is required'],
    },
    password: {
      type: String,
      required: [true, 'Password is required'],
      select: false,
    },
    confirmPassword: {
      type: String,
      validate: {
        validator: function (val) {
          return val === this.password;
        },
        message: 'Passwords must match',
      },
    },
    address: {
      type: String,
      required: [true, 'Address is required'],
    },
    role: {
      type: String,
      enum: {
        values: ['citizen', 'authority', 'admin'],
        message: 'Role must be one of: citizen, authority, admin',
      },
      default: 'citizen',
      required: [true, 'Role is required'],
```

Figure 8.5: User Database Schema

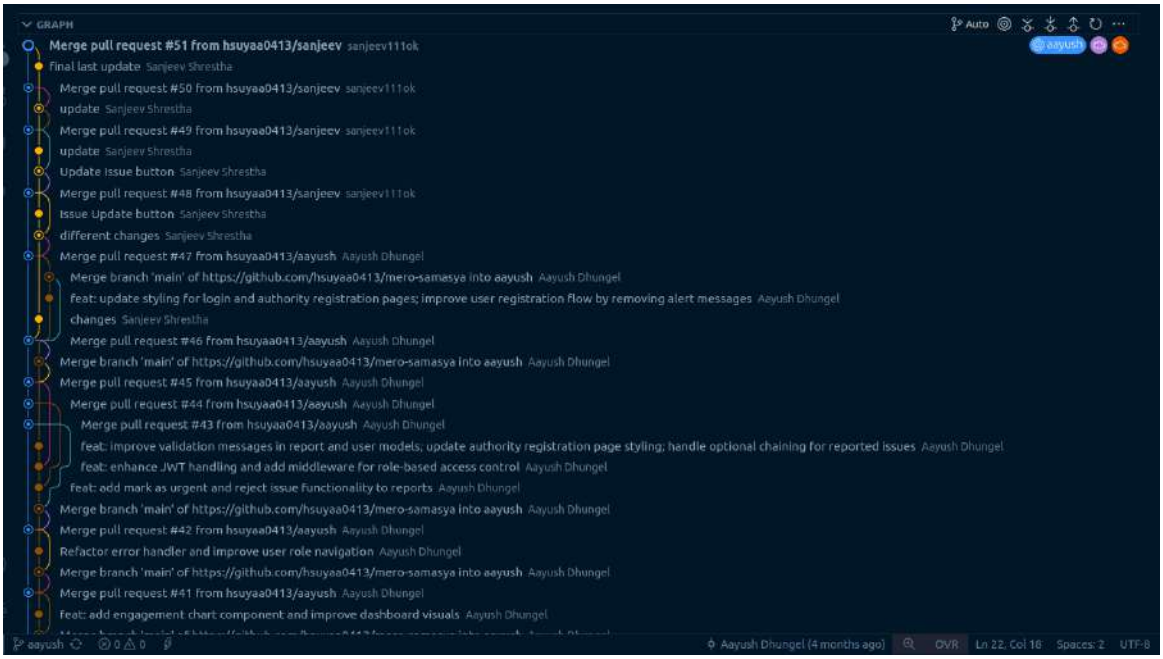


Figure 8.6: Github Version Control (Graph)

```dart

Future<void> loadReports() async {
loading = true;
notifyListeners();

try {
  final userId = await _storage.read(key: 'userId');
  if (userId == null) {
    print("No userId found in storage.");
    loading = false;
    notifyListeners();
    return;
  }

  print("Loading reports for userId: $userId");

  await _reportService.connect();

  // Fetch user reports (handles string or ObjectId)
  userReports = await _reportService.getReportsByUserId(userId);
  print("Fetched ${userReports.length} user reports.");

  // Fetch all reports
  allReports = await _reportService.getAllReports();
  print("Fetched ${allReports.length} total reports.");

```

Figure 8.7: Reports Loading Logic (Mobile App)

Figure 8.8: Reports Updating Logic (Mobile App)

```yaml
name: nero_samasyal
description: "A new Flutter project."

publish_to: 'none'


version: 1.0.0+1

environment:
  sdk: ^3.8.1


dependencies:
  flutter:
    sdk: flutter


  cupertino_icons: ^1.0.8
  # To store the data in internal memory
  shared_preferences: ^2.5.3
  # mongo dart dependence
  mongo_dart: ^0.10.3
  http: ^1.1.0
  flutter_secure_storage: ^9.2.4
  validators: ^3.0.0
  intl: ^0.18.1
  # For using the location services
  geolocator: ^14.0.2
  geocoding: ^4.0.0
  # To pick image from gallery
  image_picker: ^1.0.4
  cached_network_image: ^3.3.1
  hive: ^2.2.3
  hive_flutter: ^1.1.0


  flutter_dotenv: ^5.2.1

  location: ^8.0.1
  bcrypt: ^1.1.3
  provider: ^6.1.5
  carousel_slider: ^5.1.1
  flutter_local_notifications: ^19.4.0
  permission_handler: ^12.0.1
  # flutter_native_splash: ^2.4.6
  flutter_launcher_icons: ^0.14.4

dev_dependencies:
  hive_generator: ^2.0.0
  build_runner: ^2.4.6
  flutter_test:
    sdk: flutter


  flutter_lints: ^5.0.0

# To get the google font MANROPE
flutter:
  fonts:
    - family: Manrope
      fonts:
        - asset: assets/fonts/Manrope/static/Manrope-Regular.ttf
        - asset: assets/fonts/Manrope/static/Manrope-Bold.ttf
          weight: 700
        - asset: assets/fonts/Manrope/static/Manrope-Medium.ttf
          weight: 500
        - asset: assets/fonts/Manrope/static/Manrope-SemiBold.ttf
          weight: 600


  uses-material-design: true

# All the images gif and the external file location
  assets:
    - assets/images/
    - assets/gif/
    - .env


# For the app logo
flutter_icons:
  android: true
  ios: false
  image_path: "assets/images/logo.png"
```

Figure 8.9: Mobile App Dependencies

```dart

Future<void> _submitReport() async {
  final lang = Provider.of<LanguageProvider>(context, listen: false);

  if (!_formKey.currentState!.validate()) return;

  if (_selectedCategory == null || _selectedUrgency == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Please fill all required fields')),
    );
    return;
  }

  try {
    final storage = FlutterSecureStorage();
    String? user = await storage.read(key: 'userId');

    if (user == null || user.length != 24) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('User ID missing or invalid')),
      );
      return;
    }

    ObjectId? userId;
    try {
      userId = ObjectId.fromHexString(user);
      print('User ObjectId: $userId');
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Invalid user ID format: $e')),
      );
      return;
    }

    final String? address = (locationMessage.startsWith('Current Location'))
        ? null
        : locationMessage;

    final categoryToStore = lang.isEnglish
        ? categoryMapEng[_selectedCategory]!
        : categoryMapNep[_selectedCategory]!;

    final urgencyToStore = lang.isEnglish
        ? urgencyMapEng[_selectedUrgency]!
        : urgencyMapNep[_selectedUrgency]!;

    await ReportService.submitReport(
      title: _titleController.text.trim(),
      description: _descController.text.trim(),
      category: categoryToStore,
      urgency: urgencyToStore,
      latitude: _currentPosition?.latitude,
      longitude: _currentPosition?.longitude,
      address: address,
      imageUrls: imageUrls,
      reportedId: userId,
    );

    if (!mounted) return;

    _formKey.currentState?.reset();
    if (mounted) {
      setState(() {
        _selectedCategory = null;
        _selectedUrgency = null;
        _currentPosition = null;
        imageUrls = [];
      });
    }
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Error: ${e.toString()}')),
      );
    }
  }
}

```

Figure 8.10: Report Submitting Logic (Mobile App)