IT314: Software Engineering (Project Report)

Prof. Saurabh Tiwari

Project Title: Crowd-Powered Smart Complaint Management System

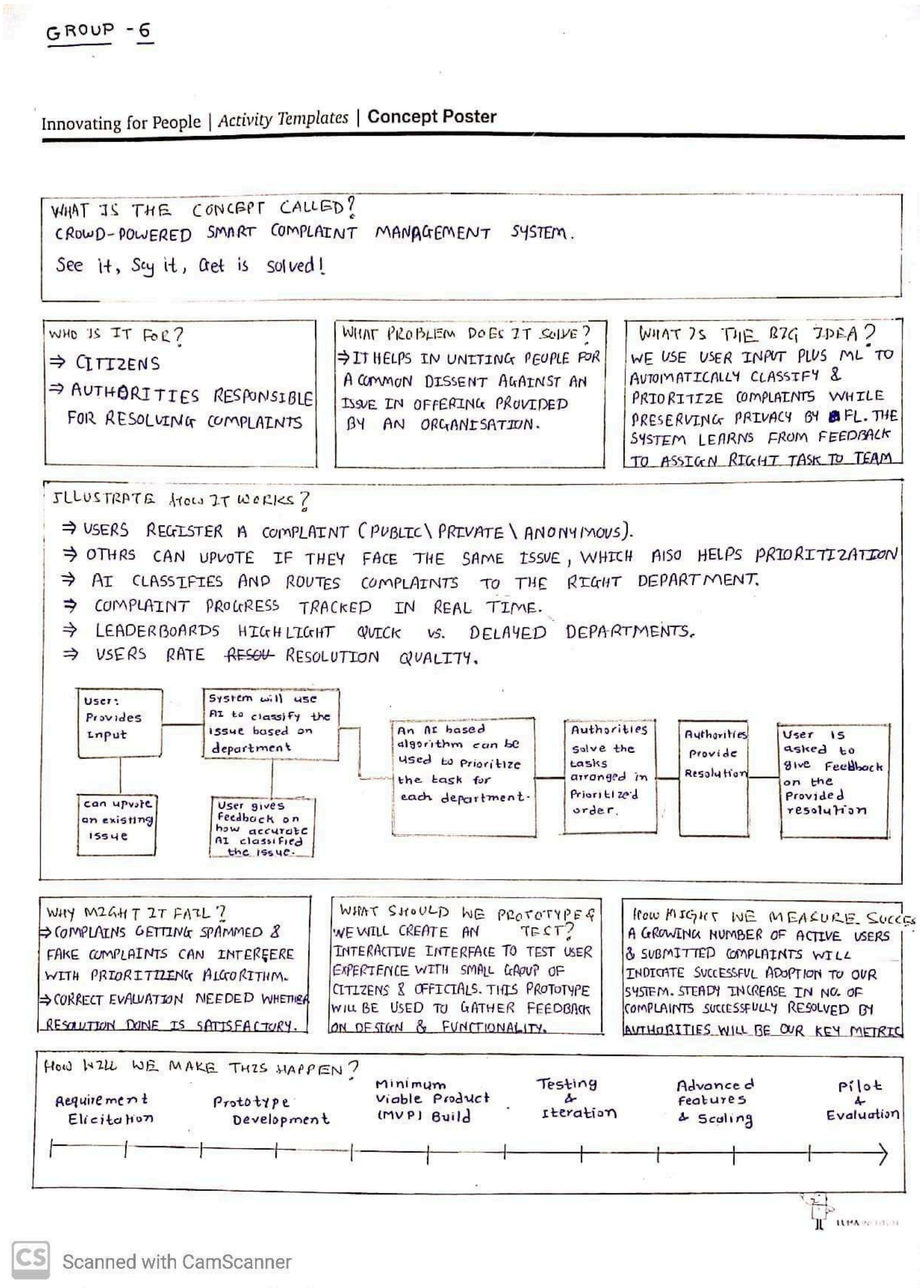
Group : 6

GitHub Repository:

https://github.com/DhruvilMehta07/Crowd-Powered-Smart-Complaint-Management-System

Name	**Student ID**
JAINIL SHAILESH JAGTAP	202301032
SHAMIT GANDHI	202301041
MEHTA DHRUVIL VIMALKUMAR (LEADER)	202301061
BHATT PARTH BHASKARBHAI	202301022
OM KANTILAL SANTOKI	202301019
NEEV VEGADA	202301031
TIRTH KORADIYA	202301018
KARAN MAKASANA	202301053
RASHA PARMAR	202301012
SAMARTH AGARWAL	202301040

Concept Poster:



GROUP -6

Innovating for People | *Activity Templates* | **Concept Poster**

WHAT IS THE CONCEPT CALLED?

CROWD-POWERED SMART COMPLAINT MANAGEMENT SYSTEM.

See it, Say it, Get is solved!

WHO IS IT FOR?

⇒ CITIZENS

⇒ AUTHORITIES RESPONSIBLE FOR RESOLVING COMPLAINTS

WHAT PROBLEM DOES IT SOLVE?

⇒ IT HELPS IN UNITING PEOPLE FOR A COMMON DISSENT AGAINST AN ISSUE IN OFFERING PROVIDED BY AN ORGANISATION.

WHAT IS THE BIG IDEA?

WE USE USER INPUT PLUS ML TO AUTOMATICALLY CLASSIFY & PRIORITIZE COMPLAINTS WHILE PRESERVING PRIVACY BY FL. THE SYSTEM LEARNS FROM FEEDBACK TO ASSIGN RIGHT TASK TO TEAM

ILLUSTRATE HOW IT WORKS?

⇒ USERS REGISTER A COMPLAINT (PUBLIC\ PRIVATE \ ANONYMOUS).

⇒ OTHRS CAN UPVOTE IF THEY FACE THE SAME ISSUE, WHICH AlSO HELPS PRIORITIZATION

⇒ AI CLASSIFIES AND ROUTES COMPLAINTS TO THE RIGHT DEPARTMENT.

⇒ COMPLAINT PROGRESS TRACKED IN REAL TIME.

⇒ LEADERBOARDS HIGHLIGHT QUICK VS. DELAYED DEPARTMENTS.

⇒ USERS RATE ~~RESOU~~ RESOLUTION QUALITY.

User: Provides Input

can upvote an existing issue

System will use AI to classify the issue based on department

User gives feedback on how accurate AI classified the issue.

An AI based algorithm can be used to prioritize the task for each department.

Authorities solve the tasks arranged in Prioritized order.

Authorities Provide Resolution

User is asked to give Feedback on the Provided resolution

WHY MIGHT IT FAIL?

⇒ COMPLAINS GETTING SPAMMED & FAKE COMPLAINTS CAN INTERFERE WITH PRIORITIZING ALGORITHM.

⇒ CORRECT EVALUATION NEEDED WHETHER RESOLUTION DONE IS SATISFACTORY.

WHAT SHOULD WE PROTOTYPE & TEST?

WE WILL CREATE AN INTERACTIVE INTERFACE TO TEST USER EXPERIENCE WITH SMALL GROUP OF CITIZENS & OFFICIALS. THIS PROTOTYPE WILL BE USED TO GATHER FEEDBACK ON DESIGN & FUNCTIONALITY.

HOW MIGHT WE MEASURE SUCCESS

A GROWING NUMBER OF ACTIVE USERS & SUBMITTED COMPLAINTS WILL INDICATE SUCCESSFUL ADOPTION TO OUR SYSTEM. STEADY INCREASE IN NO. OF COMPLAINTS SUCCESSFULLY RESOLVED BY AUTHORITIES WILL BE OUR KEY METRIC

HOW WILL WE MAKE THIS HAPPEN?

Requirement Elicitation

Prototype Development

Minimum Viable Product (MVP) Build

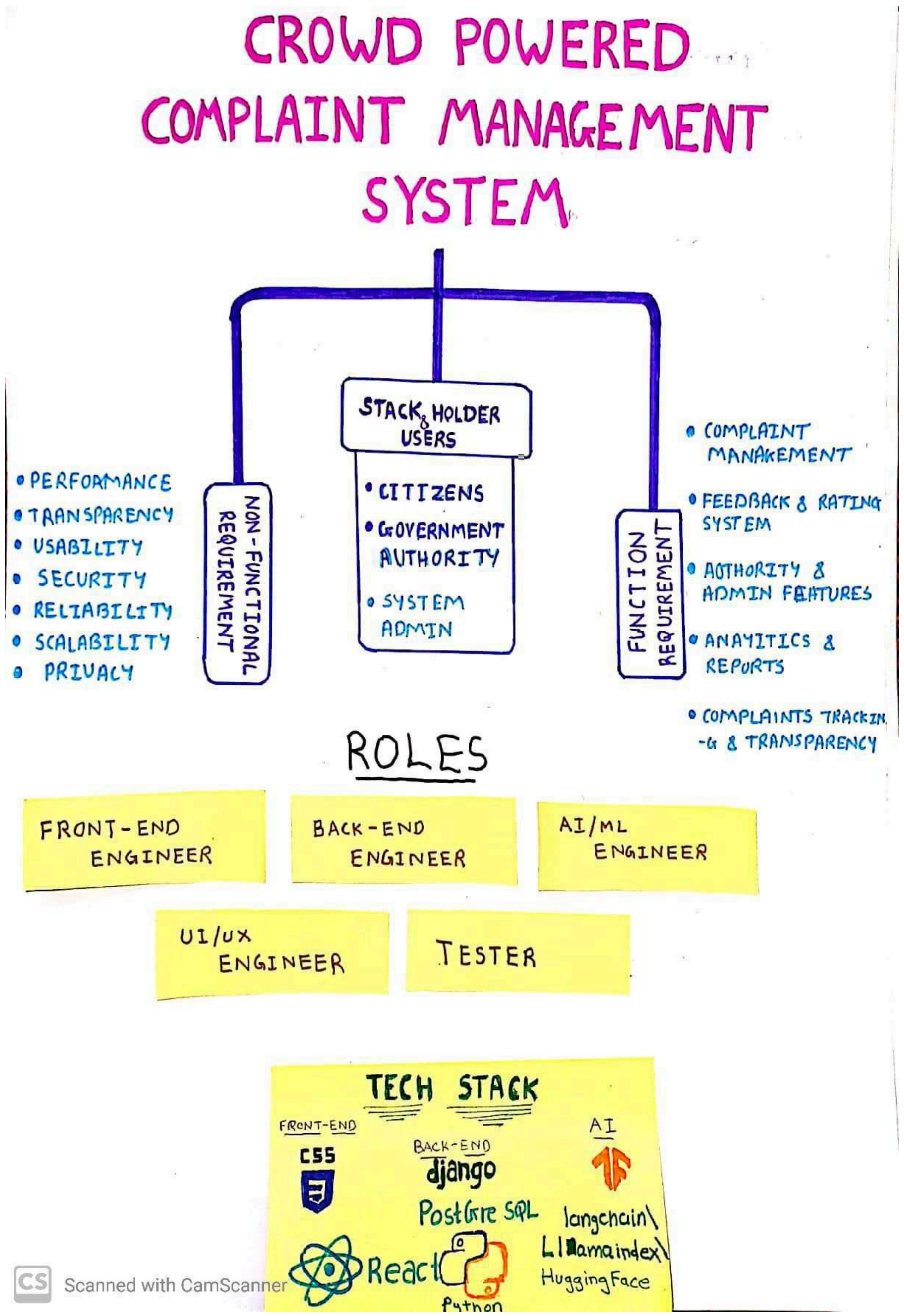
Testing & Iteration

Advanced Features & Scaling

Pilot & Evaluation

Scanned with CamScanner

Design Chart Paper:



Identification of Stakeholders and End Users:

➜ Stakeholders:

1. Citizens:

- Primary users who submit complaints (road damage, garbage, streetlight, etc.).
- Require easy submission, live tracking, and transparency.

2. Government Organizations:

- Officials responsible for resolving complaints.
- Need a centralized dashboard, prioritization, and status update system.

3. System Administrators:

- Manage the application backend.
- Responsible for security, uptime, user roles, and escalations.

4. Technical Support Team:

- Provide user support. They assist citizens and government users with technical issues, account problems, and how-to questions.

5. Field Workers (gov't employees or contractors):

- Execute the physical work. They receive tasks, navigate to locations, log work, and provide proof of resolution.

6. Elected Officials & Policy Makers:

- They analyze dashboards showing metrics, trends, and ROI to inform public policy and budgets.

➔ End Users:

Citizens:

- Can directly post their complaints (e.g., potholes, broken streetlights, illegal dumping) via a mobile app or web portal.
- They can upload photos, pin locations on a map, and provide detailed descriptions.
- After submitting, they can track the real-time status of their report (e.g., "Received," "In Progress," "Resolved"), receive notifications, and see official updates.
- They can also upvote or comment on existing complaints posted by others to indicate widespread issues and provide additional feedback.

Government Officials:

- Use a centralized dashboard to receive, manage, and resolve citizen complaints.
- They can view a prioritized queue of issues, automatically sorted by the AI based on urgency, location, and public feedback.
- They assign tasks to specific departments or Field Workers, update the status of complaints, and post public responses.
- They also generate reports to analyze performance metrics like resolution times and common problem areas.

Field Workers:

- Receive assigned tasks directly on a mobile device.
- The app provides them with the complaint details, photos, and precise location for navigation.
- They use the app to update their status (e.g., "En Route," "Work Started"), log completed work, and upload photographic proof of resolution before closing out the ticket, all from the field.

System Administrators:

- Manage the backend infrastructure of the entire platform.
- They provision user accounts and roles for government employees, monitor server health and application uptime, perform security audits, manage databases, and ensure the overall stability and security of the system.

Technical Support Team:

- Assist all other end users when they encounter problems.
- They help citizens with account issues and app functionality, aid government officials with dashboard features, and support Field Workers with their mobile devices.

Elicitation Techniques:

Stack Holder/End User	Elicitation Techniques	Reason
Citizen	Surveys, Group discussion	Since citizens form a large and diverse group, well-crafted surveys can be an effective way to capture general expectations and opinions from a broad population. Surveys provide structured insights and help identify common priorities. In addition, group discussions allow for deeper engagement by giving citizens a platform to voice their perspectives, share experiences, and debate ideas in detail. This helps uncover nuanced viewpoints that may not surface in surveys and fosters

		collective understanding.
Government Authorities	Interview	Interviews with government authorities provide insights into existing complaint-handling mechanisms, challenges, and priorities. They help understand how issues are currently addressed and identify opportunities to improve responsiveness and efficiency.
System Admin	Prototype	Engaging system administrators through prototypes allows hands-on evaluation of technical feasibility, usability, and potential system limitations. Prototyping provides a practical way to gather feedback on design choices and identify areas for improvement before full-scale implementation.
Field Workers	Interview	Interviews with Field Workers provide practical insights into operational challenges, workflow efficiency, and resource requirements. They help identify common issues faced during maintenance and gather suggestions for improving procedures.
Tech support team	Workshops, Brainstorming	Workshops with the tech support team provide an interactive platform to explore system functionalities, troubleshoot potential issues, and discuss best practices. They help gather practical insights from

		the team who directly manage user queries and technical problems. In addition, brainstorming sessions encourage creative problem-solving, generate innovative ideas for improving processes, and identify new approaches to enhance system efficiency and user support.

Functional Requirements

Functional Requirements for Citizens

1. User Authentication & Account Management

- Citizens can easily create an account by providing essential details like name, email, phone number.
- Citizens can log in securely using their email/phone and password.
- Citizens can reset their password via a secure link sent to their registered email or an OTP sent to their phone.

2. Complaint Submission

- Submit complaints with a photo, description, and category.
- Option to submit complaints anonymously.
- Automatic generation of a unique complaint ID for each submission.

3. Complaint Tracking

- View the status of complaints through stages: **Received → In Progress → Resolved**.

- Receive notifications/alerts when the complaint status changes.

4. Community Engagement

- Upvote or support existing complaints in their area to indicate priority or urgency.
- Comment on complaints to provide additional information or feedback.

5. Search and Filtering

- Search complaints by keywords, category, or location.
- Filter complaints based on status, category, or timeframe.

6. Additional Features (Suggested Enhancements)

- Ability to edit or update a submitted complaint before it is resolved.
- Access a history of all submitted complaints.
- Integration of a map view to see complaints geographically (optional but useful).

7. Feedback and Ratings

- Provide feedback or ratings for resolved complaints to evaluate satisfaction and service quality.

Functional Requirements for Government Authorities

1. User Authentication & Account Management

- Sign-Up (Admin-Approval Based): Government officials can request an account by providing details including name, official email, and, crucially, their department/jurisdiction.
- Officials can log in securely using their credentials.

2. Dashboard and Complaint Management

- View all complaints on a centralized dashboard.
- Complaints are automatically categorized and prioritized using AI/ML algorithms.
- Update complaint status through stages: **Received → In Progress → Resolved**.
- Assign complaints to relevant departments or field teams for resolution.

3. Feedback Handling

- Access citizen feedback and ratings on resolved complaints.
- Analyze trends in feedback to identify recurring issues or service gaps.

4. Search and Filtering

- Search complaints by ID, category, location, or status.
- Filter complaints by priority, department, or timeframe.

5. Additional Features (Suggested Enhancements)

- Generate reports and analytics for decision-making and performance evaluation.
- Receive alerts for high-priority or escalated complaints.
- Track departmental or team-level resolution efficiency.

Functional Requirements for Field Workers

1. User Authentication & Account Management

- Sign-Up (Admin-Approval Based): Field workers can be registered by an administrator or request an account with details including name, and assigned department/crew.
- Field workers can log in securely using their credentials.

2. Work Reporting

- Upload photos of completed work for verification and record-keeping.
- Update work status linked to assigned complaints (**Assigned → In Progress → Completed**).
- Add remarks or notes regarding work done, challenges faced, or special observations.

3. Task Management

- View assigned complaints/tasks with details such as location, priority, and deadline.
- Mark tasks as completed or request assistance if additional support is needed.

4. Communication and Coordination

- Communicate with government authorities or system administrators regarding progress or obstacles.
- Receive alerts or notifications for newly assigned or escalated tasks.

5. Additional Features

- Access history of completed tasks with photos and remarks.
- Provide feedback on workflow or system usability to improve operational efficiency.
- Integration of GPS/location tracking for field verification and monitoring (optional but useful).
- Leaderboard for field workers who has solved maximum problems.

Non-Functional Requirements

1. Usability

- Intuitive and easy-to-use interface suitable for users of all ages and backgrounds.
- Consistent navigation and layout across web and mobile platforms.
- Accessible design adhering to accessibility standards (e.g., WCAG) for visually or physically challenged users.

2. Performance

- AI-powered complaint categorization and prioritization should complete within 10 seconds.
- System should maintain low response times for all user interactions, even under high load.

3. Scalability

- Capable of handling thousands of concurrent complaints and user sessions.
- Flexible architecture to accommodate future growth in users, complaints, and modules.

4. Reliability

- Ensure 24/7 uptime with minimal downtime.
- Implement automatic failover and recovery mechanisms to handle system failures.
- Maintain data integrity and consistency across all modules.

5. Security and Privacy

- Data encryption for storage and transmission of sensitive information.
- Implement federated learning or other privacy-preserving techniques for AI training on sensitive data.
- Role-based access control to prevent unauthorized access.
- Compliance with local privacy regulations (e.g., GDPR, India’s IT Act).

6. Multi-Language Support

- System should support multiple languages for wider citizen accessibility.

7. Maintainability and Extensibility

- Modular design to allow easy updates and addition of new features.
- Well-documented codebase for efficient maintenance and troubleshooting.

8. Auditability and Logging

- Maintain detailed logs of user actions and system events for auditing and accountability.
- Provide reporting tools for system usage and performance monitoring.

User Stories:

User Story 1 – Sign Up/Login user story for all users

Front:
As a user (Citizen, Government Official, or Field Worker), I want to create an account and log in securely so that I can access the complaint management system according to my role.

Back (Acceptance Criteria):

➜ Citizen:

Can sign up by providing name, email, phone number, and address/location.

Login using email/phone and password.

➜ Government Official:

Can request an account by providing name, official email, and department/jurisdiction.

Account requires admin approval before access is granted.

Login using approved credentials.

➜ Field Worker:

Can be registered by admin or request an account by providing name, employee ID, and assigned department/crew.

Login using approved credentials.

Failure:

- Account creation fails if required fields are missing or invalid.
- Login fails if credentials are incorrect or account is not approved.

User Story 2 – Complaint Submission

Front:
As a citizen, I want to submit a complaint with a photo and description so that authorities can understand and resolve it.

Back (Acceptance Criteria):

- Complaint ID auto-generated.
- Photo upload and description required.
- Category selection available.

Failure:

- Complaint not saved if mandatory fields are missing.

- Upload fails due to unsupported format/size.

User Story 3 – Anonymous Complaint

Front:
As a citizen, I want the option to submit complaints anonymously so that I can report issues without fear.

Back (Acceptance Criteria):

- Checkbox/toggle for anonymity.
- The system hides identity from government officials.

Failure:

- Identity accidentally revealed in dashboard.
- Anonymous complaints not tracked properly.

User Story 4 – Complaint Tracking

Front:
As a citizen, I want to track my complaint status so that I know when it is being resolved.

Back (Acceptance Criteria):

- Complaint moves through stages: **Received → In Progress → Resolved**.
- Citizens can see status updates in real time.

Failure:

- Status not updated even after official action.
- Notifications not received.

User Story 5 – Upvote Complaint

Front:
As a citizen, I want to upvote complaints so that high-impact issues get priority.

Back (Acceptance Criteria):

- Citizens can upvote once per complaint.
- Upvote count visible to officials.

Failure:

- Multiple upvotes from same user.
- Votes not reflected in dashboard.

User Story 6 – Search Complaints

Front:
As a citizen, I want to search complaints by keywords, category, or location so that I can quickly find relevant issues.

Back (Acceptance Criteria):

- Search bar accepts keywords.
- Users can search by category (road, garbage, etc.).
- Users can search by location (city/area).

Failure:

- No results shown even if complaint exists.
- Wrong complaints returned.

User Story 7 – Leaderboard for Field Workers

Front:

As a field worker, I want a leaderboard of field workers based on the number of problems solved so that people can appreciate their performance.

Back (Acceptance Criteria):

- Leaderboard displays field workers ranked by the total number of problems solved.
- Option to filter leaderboard by time period (daily, weekly, monthly).
- Each entry shows worker name, ID, and problems solved count.
- Data auto-updates when a problem is marked as solved.

Failure:

- Leaderboard does not update when problems are solved.
- Incorrect ranking/order of workers.
- Missing or incomplete data for some workers.

User Story 8 – Filter Complaints

Front:
As a citizen, I want to filter complaints based on status, category, or timeframe so that I can narrow down results.

Back (Acceptance Criteria):

- Filter options include status (**Received, In Progress, Resolved**).
- Complaints filter correctly by category & date range.

Failure:

- Filters not applied correctly.
- Combining multiple filters gives wrong results.

User Story 9 – Feedback & Ratings

Front:
As a citizen, I want to provide feedback or ratings for resolved complaints so that I can evaluate service quality.

Back (Acceptance Criteria):

- Rating scale (e.g., 1–5 stars).
- Optional text feedback allowed.
- Feedback stored with complaint record.

Failure:

- Ratings allowed for unresolved complaints.
- Feedback not saved or linked properly.

User Story 10 – Dashboard for Officials

Front:
As a government official, I want a centralized dashboard so that I can monitor and resolve complaints efficiently.

Back (Acceptance Criteria):

- Dashboard lists all complaints.
- Complaints sortable by urgency, date, or location.

Failure:

- Dashboard fails to load large numbers of complaints.
- Sorting/filters not working properly.

User Story 11 – AI Prioritization

Front:
As a government official, I want AI to auto-categorize and prioritize complaints so that urgent cases are addressed first.

Back (Acceptance Criteria):

- High-priority cases appear at the top of the queue.

Failure:

- Wrong categorization leading to misrouted complaints.

User Story 12 – Review Feedback

Front:
As a government official, I want to review citizen feedback and ratings on resolved complaints so that I can assess satisfaction and service quality.

Back (Acceptance Criteria):

- Officials can view ratings and comments linked to resolved complaints.
- Feedback visible in complaint details/dashboard.

Failure:

- Feedback not displayed or mismatched with complaint.
- Unresolved complaints show feedback.

User Story 13 – Search Complaints (Officials)

Front:
As a government official, I want to search complaints by ID, category, location, or status so that I can find specific cases quickly.

Back (Acceptance Criteria):

- Search works by ID, category, location, and status.

Failure:

- Wrong or incomplete results.

User Story 14 – Field Worker Assignment

Front:
As a Field Worker, I want to receive assigned tasks with photos and maps so that I can resolve issues quickly.

Back (Acceptance Criteria):

- Task includes description, photo, location map.
- Push notification sent for new assignment.

Failure:

- Missing location/map details.
- Crew does not receive assignment in real time.

User Story 15 – Proof of Resolution

Front:
As a Field Worker, I want to upload photos after work so that I can provide proof of completion.

Back (Acceptance Criteria):

- At least one completion photo required before closing.
- Status updated to “Resolved” after upload.

Failure:

- Upload fails due to poor connectivity.
- Complaint marked resolved without proof.

User Story 16 – System Security

Front:
As a system admin, I want to enforce role-based access so that unauthorized users cannot misuse the system.

Back (Acceptance Criteria):

- Roles (Citizen, Official, Admin, Crew) clearly defined.
- Unauthorized access attempts logged.

Failure:

- Citizens gain access to the admin dashboard.
- Role permissions not applied consistently.

User Story 17 – Multi-language Support

Front:
As a citizen, I want the app in my local language so that I can use it comfortably.

Back (Acceptance Criteria):

- Minimum 2 languages supported.
- Users can switch languages easily.

Failure:

- Incomplete translations.

EPICs (Set of Sprints)

Epic 1:

As any user, I can sign up, log in, and log out so that I can access the system according to my role.

Child Stories:

- As a citizen, I can create an account and log in securely.
- As a field worker, I can log in to access assigned tasks.
- As an authority, I can log in to monitor and assign complaints.
- As an admin, I can log in and manage the whole system.

Epic 2:

As a citizen, I can raise complaints (public or private) so that issues are recorded transparently but sensitive complaints remain private.

Child Stories:

- As a citizen, I can submit a complaint with description, category, and location.
- As a citizen, I can attach media (photo/video).
- As a citizen, I can mark a complaint as private (visible only to authority, not public dashboards).
- As a citizen, I can track and withdraw a complaint before assignment.

Epic 3:

As the system, I can classify complaints into categories using AI and FL so that sensitive data remains private while helping authorities prioritize issues.

Child Stories:

- As a system, I can auto-classify complaints into categories (roads, water, electricity, etc.).
- As an authority, I can correct the AI's classification when it is wrong.
- As a system, I can prioritize urgent complaints using AI scoring.
- As a system, I can use federated learning to improve classification without exposing private complaint data.

Epic 4:

As a field worker, I can view, update, and complete assigned tasks so that authorities know the progress.

Child Stories:

- As a field worker, I can view my assigned complaints.
- As a field worker, I can accept or reject a task (with reason).
- As a field worker, I can update task status (in-progress, completed).
- As a field worker, I can upload proof of completion (photo, notes).
- As a field worker, I need a Leaderboard for field workers who has solved maximum problems.

Epic 5:

As a government authority, I can verify the resolution of complaints so that the system prevents false completions.

Child Stories:

- As an authority, I can verify field worker–completed complaints with evidence.
- As an authority, I can reopen complaints if work is unsatisfactory.
- As a citizen, I can confirm whether my issue is resolved.
- As an authority, I can close complaints after both worker and citizen verification.

Epic 6:

As a government authority, I can track complaints and resolution metrics so that governance becomes data-driven.

Child Stories:

- As an authority, I can filter complaints by urgency, location, and category.
- As an authority, I can view unresolved complaint hotspots.
- As an authority, I can generate periodic reports (weekly/monthly).
- As an authority, I can monitor average resolution time and worker efficiency.

Epic 7:

As an admin/tech support, I can maintain the system and assist users so that it runs smoothly.

Child Stories:

- As an admin, I can create, deactivate, or assign roles to users.
- As an admin, I can monitor system health (errors, performance).
- As tech support, I can respond to user issues (e.g., login problems).
- As tech support, I can escalate system-level bugs to admins.

Conflicts Between Epics

Epic 2 (Complaint Submission & Privacy) vs Epic 3 (AI/FL Classification)

- **Conflict:** AI may misclassify private complaints since training data must remain secure.
- **Resolution:** Use federated learning so sensitive data stays local, and allow manual correction by authorities.

Epic 4 (Field Worker Tasks) vs Epic 5 (Verification & Oversight)

- **Conflict:** Field workers might mark tasks as completed prematurely.
- **Resolution:** Authority + citizen verification required before final closure.

Epic 5 (Verification & Oversight) vs Epic 6 (Authority Dashboard)

- **Conflict:** Authorities may close complaints without citizen confirmation to improve dashboard metrics.
- **Resolution:** Enforce citizen confirmation step before final resolution unless citizen is unreachable.

Epic 6 (Dashboard & Analytics) vs Epic 7 (System Admin & Tech Support)

- **Conflict:** Authorities may demand sensitive aggregated data that admins restrict for privacy.
- **Resolution:** Provide only anonymized, aggregate analytics to authorities.

Proof of Concept (POC) – Sprint 1

Objective

The goal of Sprint 1 is to establish the core user authentication system using Django’s built-in AbstractUser Model for the Crowd-Powered Smart Complaint

Management System. This sprint will validate the integration of our frontend, backend, and database within a containerized environment, ensuring a solid foundation for subsequent sprints.

The main objective of this sprint is to validate the complete authentication cycle (**sign up/register, login, logout**) and provide a foundation for secure citizen access to the system.

Scope of Sprint 1

Frontend (React + Figma)

UI/UX

- Design a minimal and clean UI for login, registration, logout input screens.
- Wireframes and mockups created in Figma before implementation.

Pages

- Registration Page: ➜ with multiple registration option

  (Name, Email, Password, Confirm Password, Phone Number)

  - Register as User
  - Register as Government Authority
  - Register as Field Worker
  - Register as System Admin
- Login Page (Email + Password) Same login page for each user

Features

- Form validations (email format, password strength, phone number format).
- Integration with backend authentication logic.

Backend (Django + PostgreSQL)

Authentication

- Use Django's built-in auth models.
- Login/Logout with Django session authentication.

Database (PostgreSQL)

- Persist users with:
    - username
    - email
    - phone_number
    - password (hashed)
    - date_joined *(subject to change)*
    - Extend User model with a **role field** (e.g., USER, GOV_AUTHORITY).

Testing (tests.py)

Unit tests for:

- User registration (duplicate email validation).
- Login (valid + invalid credentials).
- Role-based login (User vs Government Authority).

Automation:

- Automated test execution before deployment.

Containerization (Docker + Docker Compose)

- Dockerfile for Django backend, React frontend, and PostgreSQL DB.
- Docker Compose setup to run full stack with one command.
- Validate service communication (React ↔ Django ↔ PostgreSQL).

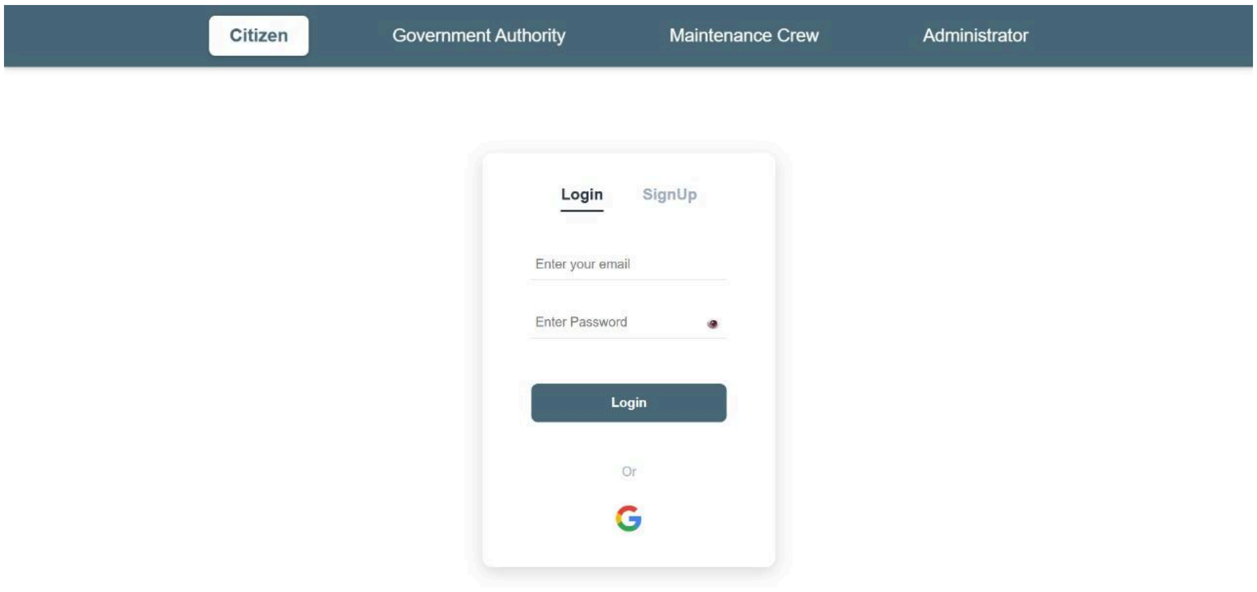
Why This Sprint is Valuable

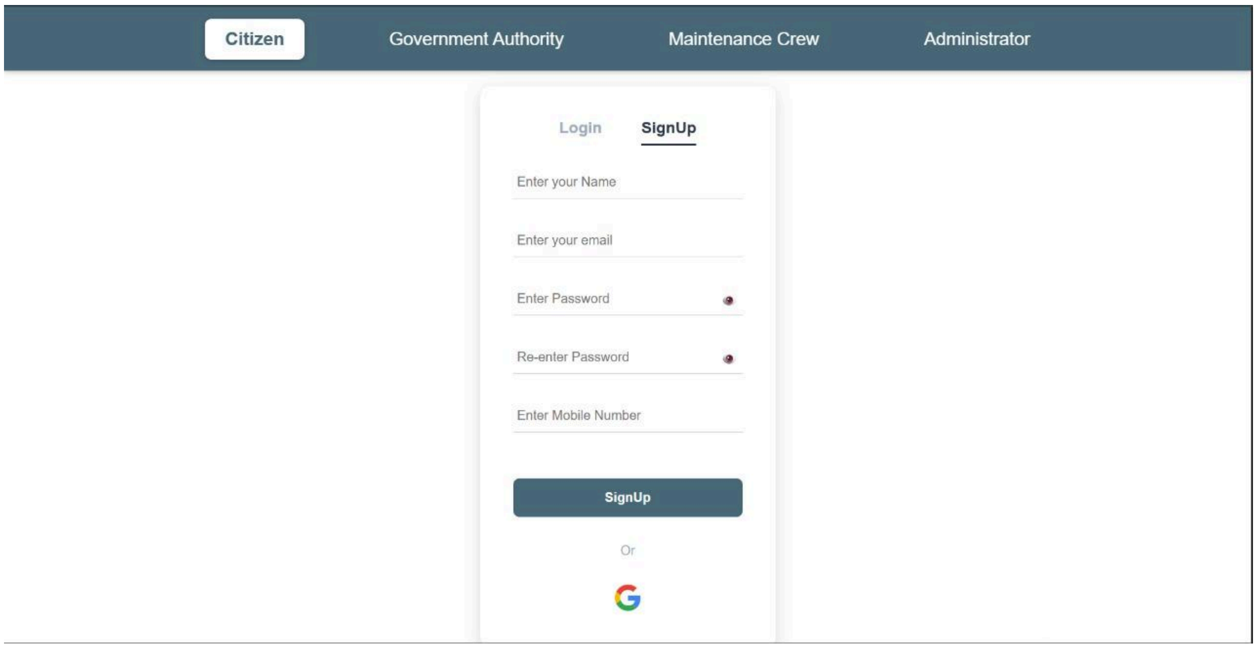
- Establishes core entry point for all users and government authority.
- Demonstrates end-to-end integration of frontend, backend, database, email, and Docker.
- Sets up testing pipeline early (**good DevOps practice**).
- Establishes a foundation for all future sprints, as authentication is a prerequisite for most other modules.

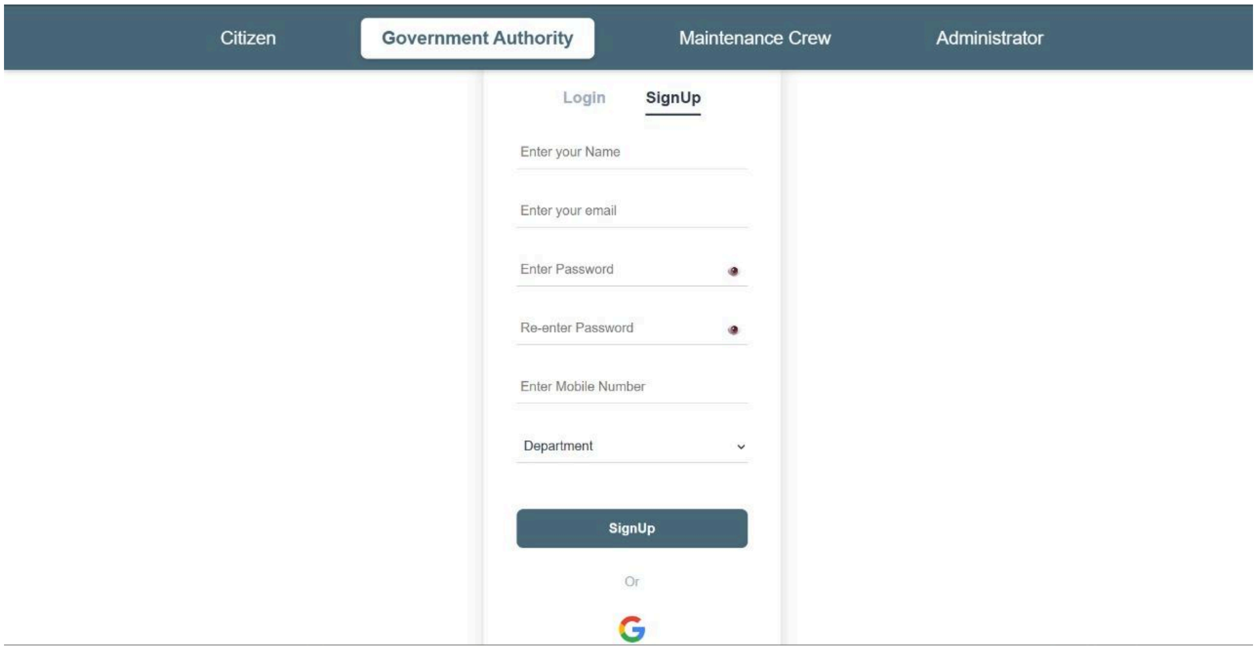
GitHub Repository:

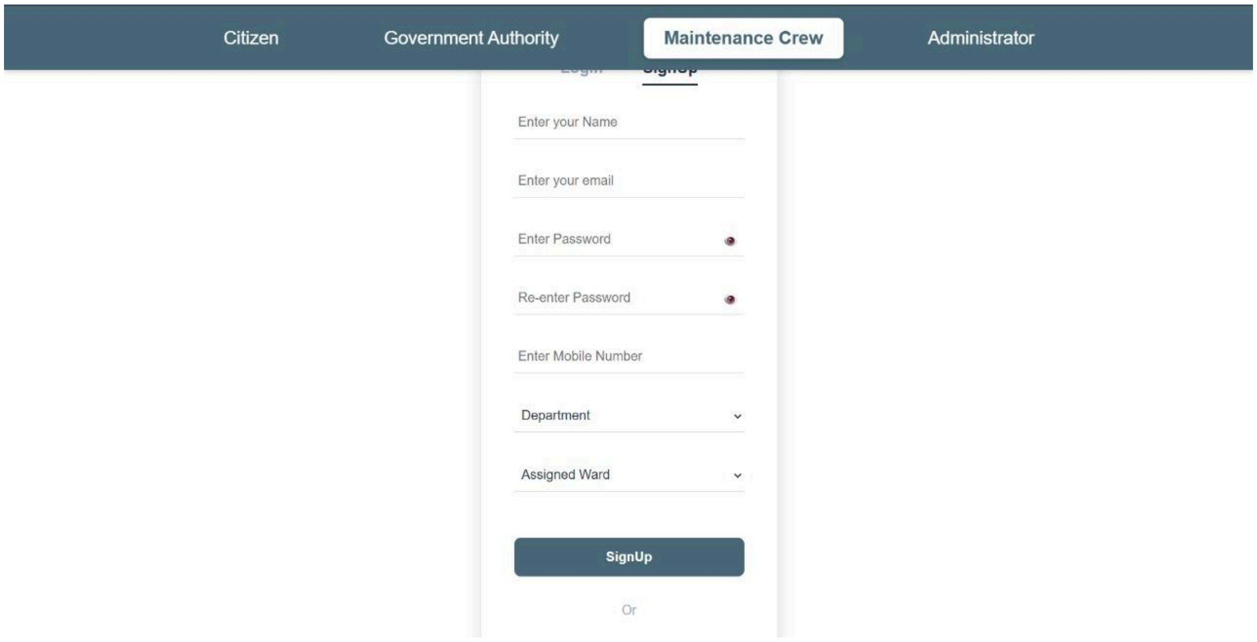
https://github.com/DhruvilMehta07/Crowd-Powered-Smart-Complaint-Management-System

User-Interface Photos: (React + Vite)






Citizen
Government Authority
Maintenance Crew
Administrator
Login
SignUp
Enter your Name
Enter your email
Enter Password
Re-enter Password
Enter Mobile Number
Department
SignUp
Or


Citizen
Government Authority
Maintenance Crew
Administrator
Enter your Name
Enter your email
Enter Password
Re-enter Password
Enter Mobile Number
Department
Assigned Ward
SignUp
Or

Contributions:

Each group member initially prepared their own individual work covering **Stakeholders, Functional Requirements (FR), Non-Functional Requirements (NFR), and User Stories**. After reviewing all individual submissions, we consolidated the most relevant and valuable points to create the **final documentation**. *(Commits and individual files are visible in the GitHub repository.)*

The final documentation includes the following components:

- Concept Poster
- Design Chart Paper
- Stakeholders and End Users - Parth
- Elicitation Techniques
    - Citizen: Shamit, Dhruvil, Parth, Rasha
    - Government Authorities: Jainil
    - Field Worker: Karan, Neev, Tirth
    - System Admin: Om
    - Tech Support: Samarth
- Functional Requirements (FR)
    - Citizen: Shamit, Dhruvil, Parth, Rasha
    - Government Authorities: Jainil
    - Field Worker: Karan, Neev, Tirth
- Non-Functional Requirements (NFR) - Neev, Om
- User Stories
    - Citizen: Shamit, Dhruvil, Parth, Rasha
    - Government Authorities: Jainil
    - Field Worker: Karan, Neev, Tirth
    - System Admin: Om
- EPICs - Shamit
- Identified Conflicts - Shamit, Dhruvil

- Proof of Concept (POC) for Sprint 1 - Jainil

Interviews Conducted:

GitHub Link:
https://github.com/DhruvilMehta07/Crowd-Powered-Smart-Complaint-Management-System/tree/main/Task1/Interview_Files

  - 2 Interviews (Field Worker, Government Authority) – *Jainil, Karan*
  - 1 Interview (Field Worker) – *Tirth, Neev*

Survey Form – *Dhruvil, Parth, Shamit, Rasha*

GitHub Link:
https://github.com/DhruvilMehta07/Crowd-Powered-Smart-Complaint-Management-System/tree/main/Task1/Survey

Backend Contributions

Members: *Shamit, Jainil, Dhruvil, Om*

- **Dhruvil:** Created forms for each entity present in models, contributed to the creation of API Views for different users, mapped respective URLs in the backend, and optimized the initial Dockerfile for easier builds.
- **Jainil:** Designed different entities in database models using OOP principles, contributed to the creation of API Views and their URLs, modified the Dockerfile to integrate Django REST framework and CORS headers for React.js integration.
- **Shamit:** Built the initial Dockerfile and Docker Compose files to install all required libraries during the first build, created initial class-based views for a template-based approach, and wrote basic tests to validate the backend.
- **Om:** Implemented React rerouting on frontend for client side URL routing.

(All backend files and commits are available in the GitHub repository, with individual contributions.)

Frontend Contributions

Members: *Parth, Neev, Karan, Tirth, Rasha, Samarth*

- **Parth:** Created the Figma design (UI).
- **Neev, Karan, Tirth:** Developed the Login Page using React.
- **Rasha, Samarth:** Developed the Home Page using React.

(All React + Vite files have been committed individually by the frontend team and are available in the GitHub repository.)