# Software Requirements Specification

# PRJ566 – Winter 2025

# PRJ566 – Team No: 3

# Name of Project: Community Service App for the Government of Ontario

# Project Leader: **Sanskar Parakhlal Pardesi**

**Last updated: 01-25-2025**

**Team Members:**

**1. Vrundaben Vijaykumar Patel**

**2. Sanskar Parakhlal Pardesi**

**3. Nadi Aung Lin**

**4. Abhi Mansukhbhai Chakrani**

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. **Document Authors**
   2. **Revision History**
   3. **Document Conventions**
   4. **Document Purpose**
   5. **Intended Audience**
   6. **Group Agreement**
2. **Project Overview**
   1. **Project Proposal**
   2. **Stakeholders and Users**
   3. **Project Scope**
   4. **Functional Requirements**
   5. **Nonfunctional Requirements**
   6. **System risks**
   7. **Operating Environment**
   8. **UI/UXD Interface Mockups**
3. **Process & Data Modeling**
   1. **UML Modeling: DFDs & Activity Diagrams**
   2. **Use Case Specification** 
      1. **Business Rules**
      2. **System Use Case Diagrams**
      3. **Use Case Description Tables**
4. **Domain Class Diagram**
5. **Database (Select either 5.1 or 5.2)**
   1. **RDBMS Artifacts**
      1. Scripts to create, populate, delete tables
      2. Data Dictionary
   2. **NoSQL Artifacts**
6. **Work breakdown Structure (WBS)**
7. **Milestones & Acceptance Criteria**
8. **Implementation Schedule (Agile/Waterfall)**
9. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## Document Authors

1. Rudra Vishalkumar Barot
2. Vrundaben Vijaykumar Patel
3. Sanskar Parakhlal Pardesi
4. Nadi Aung Lin
5. Abhi Mansukhbhai Chakrani

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | Sections of this document that were completed/updated this week, example: completed/updated  1. Introduction/Overview  1.1 Document Authors  1.2 Revision History (ongoing)  1.4 Document Purpose  1.5 Intended Audience  1.6 Group Agreement  2. Project Overview  2.1 Project Proposal |
| Week 04 | 2.2 Stakeholders and Users 2.5 Project Scope 2.6 System Risks  2.7 Operating Environment |
| Week 05 | 2.3 Functional Requirements  2.4 Non-Functional Requirements  3.1 UML/DFD Modeling and Data Modeling |
| Week 06 | 2.8 UI/UXD Interface Mockups  3.1 UML/DFD Modelling and Data Modelling: Activity Diagram |
| Week 07 | **3.2.1 Business Rules**  **3.2.2 System Use Case Diagram**  **3.2.3 Use Case Description Tables** |
| Week 08 | **3.3 Use Case Specifications with corresponding interface mockups** |
| Week 09 | 4. Domain Class Diagram |
| Week 10 | **5. Database**  **NoSQL Artifacts** |
| Week 11 | **6. Work breakdown Structure (WBS)**  **7. Milestones & Acceptance Criteria** |
| Final | 8. Implementation Schedule |

## 1.3 Document Conventions

For example:

Any text in red indicates an exception or error.

Any text in blue is in-progress.

Any text highlighted in yellow is an important point.

Any text in green was recently added.

Any text *italicized* represents definitions.

Any text with ~~strike-through~~ is deleted.

## 1.4 Document Purpose

The primary purpose of this document is to define the technical and functional requirements for the development of the **Community Service App** for the Government of Ontario. It provides:

* A clear roadmap for developers, testers, and stakeholders to understand the objectives and expectations of the system.
* Detailed requirements to guide the design, development, and deployment of the application.
* A reference for stakeholders to validate the system's alignment with their goals and expectations.

This SRS ensures that all team members and stakeholders are aligned on the project objectives, scope, and deliverables.

## 1.5 Intended Audience

This document is intended for the following audiences:

* **Project Team Members:** To ensure a shared understanding of technical and functional requirements during development.
* **Government Sponsors:** To engage specific government departments responsible for funding and supporting technology projects, ensuring alignment with governmental objectives, and securing essential resources.
* **Regulatory Authorities:** To ensure the application meets all legal and regulatory requirements, facilitating approvals and compliance checks.
* **Faculty Advisors and Evaluators:** To assess the feasibility, comprehensiveness, and progress of the project from an academic and practical perspective.
* **Developers and Testers:** To use as a reference for system design, coding, and quality assurance, facilitating accurate implementation and thorough testing.

## 1.6 Group Agreement

**TEAM AGREEMENT**

**Team #: 3**

**Project Title:** Community Service App for the Government of Ontario

**Project Time Frame:** 6 Months

**Team Members:**

1. Project Manager - **Rudra Vishalkumar Barot**
2. Technical Lead - **Vrundaben Vijaykumar Patel**
3. Software Developer - **Sanskar Parakhlal Pardesi**
4. Quality Assurance Specialist - **Nadi Aung Lin**
5. UX/UI Designer - **Abhi Mansukhbhai Chakrani**

**Team Leadership: Rudra Vishalkumar Barot**

**Team Functions:**

* *Share information and collaborate using Microsoft Teams, OneDrive and email.*
* *Regularly update project progress and share responsibilities for research, development, and documentation.*
* *Resolve conflicts or issues collaboratively, escalating to the project leader if needed.*

**Team Meetings:**

*Team meetings are scheduled weekly on* ***Wednesdays at 11:00 AM*** *and typically last for* ***two hours****. These meetings are essential for reviewing project deliverables, discussing progress, and resolving any issues that arise. Additional meetings may be convened as necessary to address urgent matters or ensure continuous alignment with project goals.*

**Team Problems:**

*Any technical, logistical, or interpersonal issues will be discussed during meetings or flagged via communication platforms for prompt resolution.*

**Team Commitment**

***The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.***

**Name Date**

|  |  |
| --- | --- |
| Vrundaben Vijaykumar Patel | 01-21-2025 |
| Rudra Vishalkumar Barot | 01-21-2025 |
| Sanskar Parakhlal Pardesi | 01-22-2025 |
| Nadi Aung Lin | 01-22-2025 |
| Abhi Mansukhbhai Chakrani | 01-23-2025 |

# 2 - Project Overview

## 2.1 Project Proposal

Project Background

The Government of Ontario is currently hampered by a fragmented and inefficient system for handling resident service requests and complaints. This system, largely dependent on manual processes, struggles with unresolved issues such as garbage collection delays, potholes, illegal dumping, and malfunctioning streetlights. The proposed Community Service App aims to revolutionize this by consolidating all service requests and complaint processes into a single, easy-to-use platform. This centralization is designed to streamline operations and significantly improve response times.

The necessity for the Community Service App stems from a critical need to enhance public satisfaction and rebuild trust in government services. By improving operational efficiency and transparency, the app will provide residents with a reliable and effective means of communicating their concerns. This, in turn, facilitates quicker resolutions, better management of city resources, and a more engaged citizenry. The Community Service App represents a pivotal advancement in how municipal services interact with the community, ensuring that the everyday challenges of residents are addressed promptly and efficiently.

**Problem Statement**

|  |  |
| --- | --- |
| The Problem of: | The Government of Ontario currently faces fragmented and inefficient processes for reporting and addressing resident complaints. Residents must navigate multiple disconnected channels, such as phone calls, emails, and in-person visits, to report issues like garbage collection delays, potholes, streetlight malfunctions, and more. These fragmented systems hinder coordination, delay response times, and cause frustration among residents. |
| Affects: | This issue affects both Ontario residents and the government departments responsible for managing public services. Residents struggle with limited visibility into the status of their complaints, while government departments are burdened by manual processes that slow down resolution and reduce efficiency. |
| The impact of which is: | Delays in addressing complaints, leading to frustration, lack of transparency regarding issue resolution, miscommunication between departments, resulting in errors, growing dissatisfaction among residents, eroding trust in government services. |
| A successful solution would: | Streamlines the reporting process for residents, enables faster resolution of issues, improves communication between residents and government departments, increases transparency, improving overall satisfaction and trust. |

**Product Vision**

|  |  |
| --- | --- |
| For | Ontario residents who need a reliable, easy-to-use way to report and track public service issues, eliminating the challenges of dealing with multiple disconnected channels. |
| Who | Face difficulties in accessing multiple reporting channels for common service issues like:   * Garbage collection delays * Potholes * Streetlight malfunctions * Illegal dumping |
| The Product Name | **Community Service App** for the Government of Ontario: a digital platform (mobile and web) designed to centralize service request management. |
| That | Offers a unified system to report issues, track resolution progress, and enhance transparency between government departments and Residents. |
| Unlike | Current systems that are inefficient, fragmented, and lack transparency, the Community Service App offers a centralized, streamlined solution that enhances communication, accountability, and satisfaction. |
| Our product | A user-friendly platform that enables residents to report issues, track resolutions in real time, and stay informed. For government departments, it provides tools for efficiently managing complaints and improving public service delivery. |

## 2.2 Stakeholders and Users

|  |  |  |
| --- | --- | --- |
| **Stakeholder Name/Identifier** | **Category** | **Description**/**Role** |
| **Ontario Residents (Citizens)** | Users | Use the app to submit public service requests, track the status of reported issues, and provide feedback to help improve the overall user experience. |
| **Government Agencies** | Admin | Provide project funding and strategic oversight, ensuring that the app aligns with public policy and improves service delivery across government departments. |
| **Department Managers** | Admin | Oversee incoming service requests, assign tasks to field officers, and monitor performance metrics to ensure efficient handling and resolution of issues. |
| **Clerks/Field Officers** | Admin | Act as the frontline personnel who process and resolve reported issues, update statuses in the system, and communicate with residents for additional information when needed. |
| **Project Team (PM, Developers, UI/UX, QA)** | Development | Design, develop, and test the app to meet project requirements, ensuring functionality, usability, and overall quality before deployment. |
| **IT Support Team** | Maintenance | Provide technical support and ongoing maintenance to ensure system stability, troubleshoot issues, and manage regular updates and security patches. |
| **Change Management Head** | Maintenance | Lead training initiatives and communication strategies to facilitate smooth adoption of the app among all stakeholders, ensuring minimal disruption during implementation. |
| **External Vendors/Partners** | Maintenance | Deliver essential third-party services such as cloud hosting, mapping solutions, and system integrations to support the app’s performance and scalability. |
| **Regulatory Authorities** | Admin | Monitor and enforce legal, ethical, and data protection standards to ensure that the app complies with all relevant regulations and guidelines. |

## 2.3 Functional Requirements

## 1. **User Registration and Authentication:**

## o Users must be able to securely register with their details (e.g., name, email, phone number, and address) and authenticate via a login system.

## o Support for two-factor authentication (2FA) for added security.

## o Separate registration and login processes for residents, clerks, and administrators.

## 2. **Issue Reporting Module:**

## o Users can report municipal service issues by selecting from predefined categories such as garbage collection, illegal dumping, potholes, or broken streetlights.

## o The issue report must include fields for location (integrated with GPS or address search), a brief description, and optional photo attachments.

## o Automated categorization and assignment of tickets to the relevant department based on the type and location of the issue.

## 3. **Real-Time Complaint Tracking:**

## o Users can track the progress of their complaints, including updates on the current status (e.g., "In Progress," "Resolved").

## o Notifications (via app, email, or SMS) are sent to users for significant updates, such as ticket assignment or resolution.

## 4. **Administrative Dashboards:**

## o Department heads and clerks can access dashboards displaying ticket details categorized by priority, location, and type.

## o Supervisors can monitor ticket resolution times and generate analytics reports on high-complaint areas.

## 5. **Multilingual Support:**

## o The app must provide language options to cater to Ontario’s diverse population (e.g., English, French, and other commonly spoken languages).

## 6. **Feedback and Ratings:**

## o Users can rate their satisfaction with the resolution process and provide feedback to help improve services.

## 7. **Analytics Tools:**

## o Administrators can generate insights from complaint trends to identify areas requiring resource allocation or operational improvement.

## 8. **User Support and Resources:**

## o A Frequently Asked Questions (FAQ) section to assist users with common queries.

## o Contact support channels for reporting app-related issues or seeking guidance.

## 

## 2.4 Non-Functional Requirements

**Operational Requirements**

1. **Availability:**
   * The app must maintain 99.9% uptime to ensure uninterrupted service for users and administrators.
   * Scheduled maintenance windows must be communicated in advance to avoid service disruptions.
2. **Compatibility:**
   * The app must be compatible desktop browsers.
   * Support for various device screen sizes and resolutions.
3. **Integration:**
   * Seamless integration with existing government systems, such as resource allocation tools and municipal service databases.
4. **Scalability:**
   * The app infrastructure must accommodate up to 1 million users initially, with the ability to scale as the user base grows.

**Performance Requirements**

1. **Response Time:**
   1. The app must load all pages and functionalities within 2 seconds under normal traffic conditions.
   2. Tickets and updates should be processed in real-time, with minimal latency.
2. **Concurrent Users:**
   1. The app must support at least 10,000 concurrent users without performance degradation.
3. **Data Storage:**
   1. The system must store and retrieve ticket and user data efficiently, ensuring a smooth user experience

**Security Requirements**

1. **Data Protection:**
   1. All user data must be encrypted both at rest and in transit using modern encryption standards.
2. **Access Control:**
   1. Implement role-based access control (RBAC) to restrict access based on user roles (e.g., residents, clerks, administrators).
3. **Regular Audits:**
   1. Conduct monthly security audits to identify vulnerabilities and ensure compliance with government privacy policies.
4. **Incident Response:**
   1. Develop a robust incident response plan to quickly identify and address potential breaches or security issues.
5. **Data Privacy Compliance:**
   1. The app must comply with privacy regulations such as Ontario's Freedom of Information and Protection of Privacy Act (FIPPA).
6. **Backup and Recovery:**
   1. Regular backups must be performed, and a disaster recovery plan must be in place to restore operations in case of data loss.

## 2.5 Project Scope

### 1. Objectives

* Develop a web application for Ontario residents to report municipal issues such as garbage collection delays, illegal dumping, potholes, and non-functional streetlights.
* Enhance communication between residents and municipal departments through real-time updates, notifications, and a transparent ticketing system.
* Improve efficiency in complaint resolution via automated categorization, prioritization, and resource allocation.
* Provide actionable insights to government departments using analytics tools to identify high-complaint areas and improve municipal services.

### 2. Key Deliverables

##### Frontend (Web Applications)

#### Resident Dashboard

* Secure user registration and login (email, phone, social media options).
* Simple issue reporting with geolocation, photo uploads, and predefined issue categories.
* Real-time status tracking of reported complaints.
* Notifications and updates on ticket progress.
* Multilingual support to serve Ontario’s diverse population.

#### Clerk Dashboard

* View assigned complaints with issue details and priority status.
* Update ticket status and record resolution actions.
* Communicate with residents for additional information if required.
* Track performance jmetrics and workload.
* Monitor system-wide complaint statuses and key performance metrics.
* Manage and assign clerks based on workload, location, and urgency.
* Automated and manual ticket assignment capabilities.
* Advanced analytics to identify trends and high-complaint areas.
* Role-based access control for supervisors and department heads.

#### Backend & Database

##### API Services

* User authentication and profile management.
* Ticket submission, categorization, assignment, and tracking.
* Notifications and communication between residents, clerks, and administrators.

##### Database Management

* Secure storage of user profiles (residents, clerks, and administrators).
* Complaint records, including status, history, and geolocation data.
* Analytics data to track service efficiency and generate reports.

#### Other Deliverables

* **Testing Artifacts:** Functional, performance, and security test plans and reports.
* **Documentation:** API documentation, system architecture, user guides, and onboarding materials for municipal staff.

### 3. Core Features

#### For Residents

* User-friendly issue reporting with categorization and location tagging.
* Real-time tracking and notifications for complaint progress.
* Access to FAQs and resources on common municipal concerns.
* Feedback and rating system for resolved complaints.

#### For Clerks

* Automated ticket assignment based on issue type, location, and priority.
* Role-based access control for different levels of staff.
* Performance tracking and reporting tools.

#### For Administrators

* Real-time dashboards for ticket status monitoring.
* Tools for workload distribution, priority adjustments, and escalations.
* Heatmaps and analytics to track trends and improve response efficiency.

### 4. Out of Scope

* Integration with third-party municipal systems (e.g., garbage collection tracking software).
* Real-time chat functionality between residents and clerks.
* Support for non-municipal service requests (e.g., event bookings, permits).

### 5. Technical Requirements

#### Operational

* Compliance with Ontario’s privacy laws (Freedom of Information and Protection of Privacy Act).
* Web-based platform supporting major desktop browsers.

#### Performance

* Support up to 100,000 simultaneous users.
* Complaint submission and dashboard load times under 2 seconds.

#### Security

* End-to-end encryption for data transfers.
* Regular security audits and penetration testing.
* Role-based access control for data security.

### 6. Assumptions

* Government agencies will collaborate for platform integration.
* High adoption rates due to ease of use and multilingual support.
* Adequate funding and technical resources for implementation and maintenance.

### 7. Constraints

* Completion and deployment within 8 months.
* Compliance with provincial data security and accessibility regulations.
* Phased implementation due to budget constraints.

### 8. Timeline (Example Phases)

#### Phase 1: Planning and Analysis (4 weeks)

* Requirement gathering from stakeholders.
* Identification of technical and operational constraints.

#### Phase 2: Design (6 weeks)

* UI/UX design for resident and administrator interfaces.
* Approval of wireframes and prototypes.

#### Phase 3: Development (12 weeks)

* Backend API development for ticketing and user management.
* Frontend development for residents, clerk, and admin dashboards.
* Integration with government data systems.

#### Phase 4: Testing (4 weeks)

* Functional, performance, and security testing.
* User acceptance testing and iterative improvements.

#### Phase 5: Deployment and User Onboarding (2 weeks)

* Hosting and deployment of the web application.
* Staff training and onboarding sessions.

### 9. Success Criteria

* **Performance:** Supports at least 10,000 active users daily with minimal downtime.
* **Efficiency:** 90% of reported issues successfully resolved within expected timelines.
* **Usability:** Positive feedback from residents and government staff.
* **Impact:** Measurable improvement in municipal response times within 3 months of launch.

## 2.6 System Risks

|  |  |
| --- | --- |
| **Risk** | **Response** |
| **Adoption Challenges:** Residents may resist using the app due to lack of awareness or technical familiarity. | Launch a government-led awareness campaign via social media, local centers, and public announcements. Offer workshops and tutorials to teach residents app usage. Provide incentives like faster resolution for early adopters. |
| |  | | --- | | **Resistance from Government Staff:** Staff may resist the new system due to workflow changes or lack of technical skills. | | Conduct change management workshops emphasizing the app’s benefits. Provide comprehensive training and involve staff in development for better ownership. Offer technical support during the initial rollout. |
| **Integration with Legacy Systems:** Existing government systems may not be compatible with the app, causing delays. | |  | | --- | | Perform a thorough audit of legacy systems to identify compatibility issues. Use middleware to bridge gaps. Allocate resources for hiring experienced integration consultants. | |
| **Cybersecurity Threats:** Handling sensitive resident data increases the risk of breaches. | Use end-to-end encryption for data transfer and storage. Conduct regular vulnerability assessments. Develop a robust incident response plan for potential breaches. |
| **High Workloads on Departments:** Increased complaint volumes may overwhelm departments. | Implement automated ticket prioritization based on urgency. Use analytics to identify high-demand areas and allocate more resources to those departments. |
| **Scalability Limitations:** The app may struggle to handle future expansions or added features. | Design a modular system with cloud-based infrastructure for easy scalability. Conduct regular performance evaluations and prepare for increased user loads. |
| **Inexperience with Development Tools:** Some team members may lack experience with tools like Android Studio or cloud platforms. | Organize team training sessions and workshops for necessary tools. Create shared resources and schedule weekly team meetings for collaborative problem-solving and skill development. |
| **Tight Project Timeline:** Completing a fully functional app within six months is ambitious. | Establish a clear work breakdown structure (WBS) with detailed milestones. Use Agile methodologies to prioritize key features and conduct regular sprints to stay on track. |

## 2.7 Operating Environment

The operating environment for the **Community Service App** is designed to ensure a streamlined, efficient, and scalable development process. The team will use modern tools and technologies to deliver a high-quality web application that meets the needs of **Ontario residents and government departments**. Below is the detailed operating environment:

## 1. Development Environment

### Frontend Development

* **Frameworks/Libraries:**
  + **React.js** for building the user interface.
  + **Next.js** for server-side rendering (SSR) and backend functionality.
* **Development Tools:**
  + **Visual Studio Code (VS Code)** as the primary code editor.
  + **Figma** for UI/UX design, wireframing, and prototyping.

### Backend Development

* **Backend Framework:** **Next.js** for API routes and server-side logic.
* **API Development:** **RESTful APIs** for communication between the frontend and backend.
* **Programming Language:** **JavaScript** (consistent across frontend and backend).

### Database

* **Database System:** **MongoDB** for storing structured and unstructured data (e.g., user profiles, complaint records, geolocation data).
* **Database Tools:** **Mongoose** for object data modeling (ODM) and database management.

### Version Control

* **Version Control System:** **Git** for code management.
* **Repository Hosting:** **GitHub** for collaboration, code reviews, and version control.

## 2. Testing Environment

### Functional Testing

* **Browser Testing:** **Selenium with Python** for cross-browser testing (**Chrome, Firefox, Safari, Edge**).
* **Device Testing:** **Selenium** for testing responsiveness on different devices (**desktop, tablet, mobile**).

### Performance Testing

* **Performance Monitoring:** **Lighthouse** for performance audits and optimization.

### Security Testing

* **Vulnerability Scanning:** **Manual and automated security testing** to ensure compliance with **Ontario’s privacy laws**.

### Staging Environment

* A **staging environment** that mimics the production environment for **pre-launch testing and validation**.

## 3. Production Environment

### Hosting

* **Cloud Hosting:** **Vercel** for seamless deployment and hosting of the **Next.js** application.
* **Scalability:** **Vercel’s serverless architecture** for automatic scaling based on traffic.

### Database

* **Database Hosting:** **MongoDB Atlas** for cloud-based database management and scalability.
* **Backup and Recovery:** **Automated backups** and **disaster recovery plans** to ensure data integrity.

### Geolocation and Mapping

* **Mapping API:** **Google Maps API** for **geolocation tagging, mapping, and geofencing**.

### Security

* **Authentication:** **OAuth 2.0** for **secure user authentication and authorization**.
* **Data Encryption:** **HTTPS** for **secure data transfer** and **encryption for sensitive data storage**.

### Payment Gateway

* **Payment Integration:** **Stripe** for potential **future payment integrations** (e.g., **fines, fees**).

## 4. External Integrations

### Third-Party APIs

* **Geolocation:** **Google Maps API** for **mapping and location-based services**.
* **Analytics:** **Google Analytics** or similar tools for **user behavior tracking**.

### Continuous Integration/Continuous Deployment (CI/CD)

* **CI/CD Pipeline:** **GitHub Actions** for **automated builds, testing, and deployments**.

## 5. Documentation and Collaboration

### Collaboration Tools

* **Communication:** **Microsoft Teams** for **team communication and collaboration**.
* **Project Management:** **GitHub Projects** or **Microsoft Teams** for **task tracking and milestone management**.

### Documentation

* **API Documentation:** **Postman** for **API documentation and testing**.
* **Diagrams and Architecture:** **Visual Paradigm** for creating **system architecture diagrams, flowcharts, and technical documentation**.
* **User Guides:** **Onboarding materials and user manuals** for **residents and government staff**.

## 6. Infrastructure for Scalability

* **Modular Architecture:** A **modular design** to ensure **scalability and flexibility** for future features.
* **Serverless Infrastructure:** **Vercel’s serverless architecture** for **automatic scaling and high availability**.
* **Content Delivery Network (CDN):** **Vercel’s built-in CDN** for **serving static assets efficiently**.

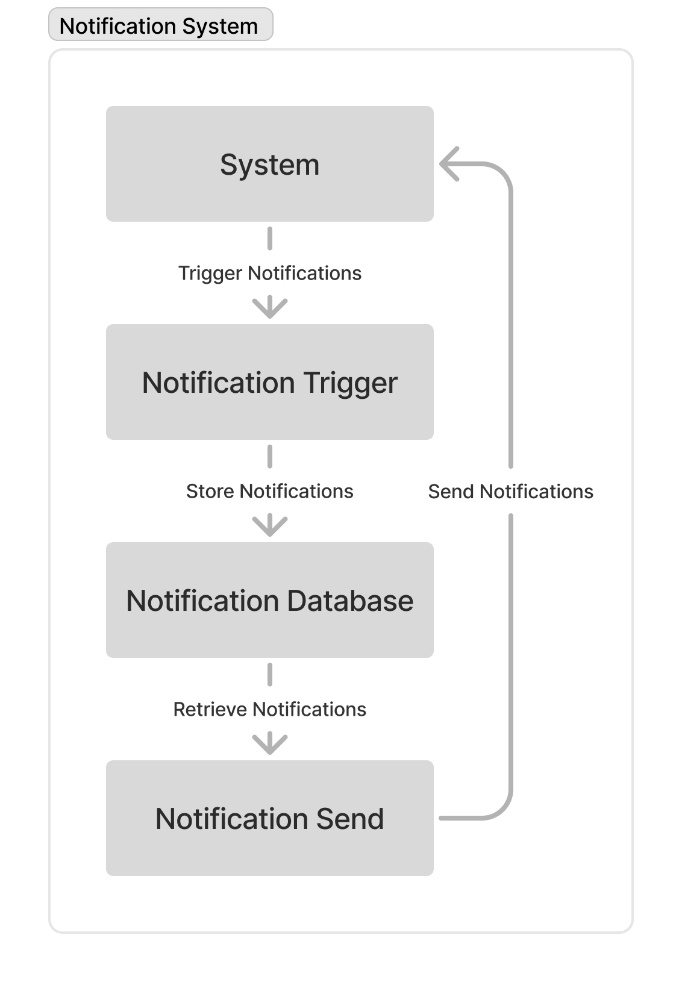
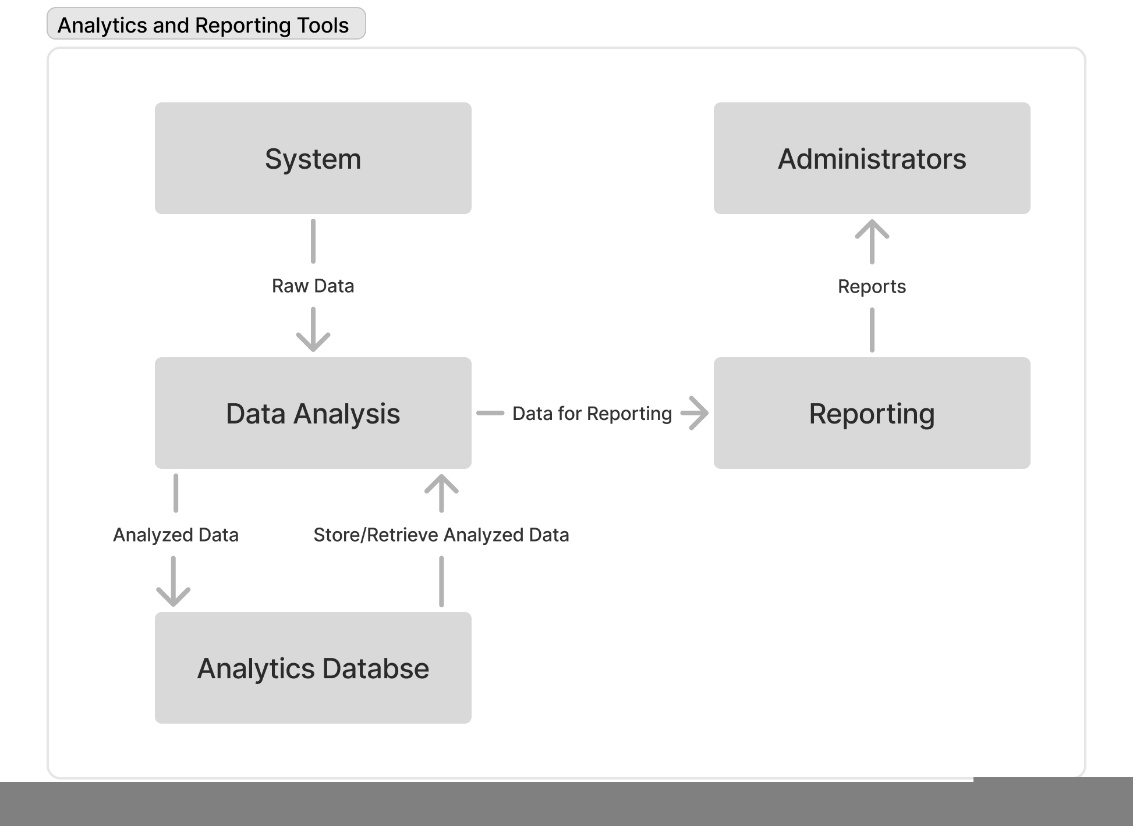
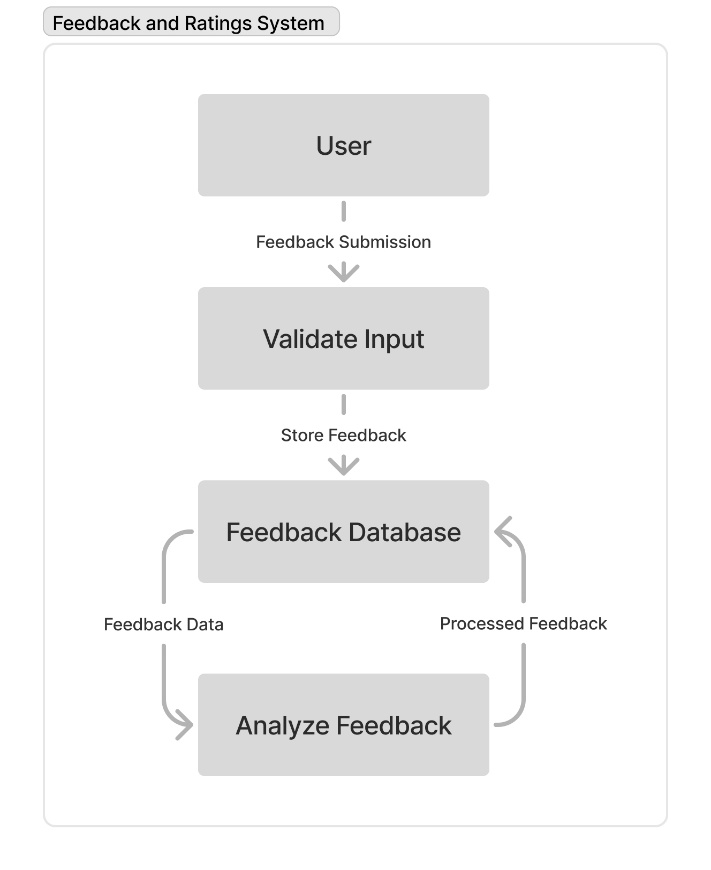
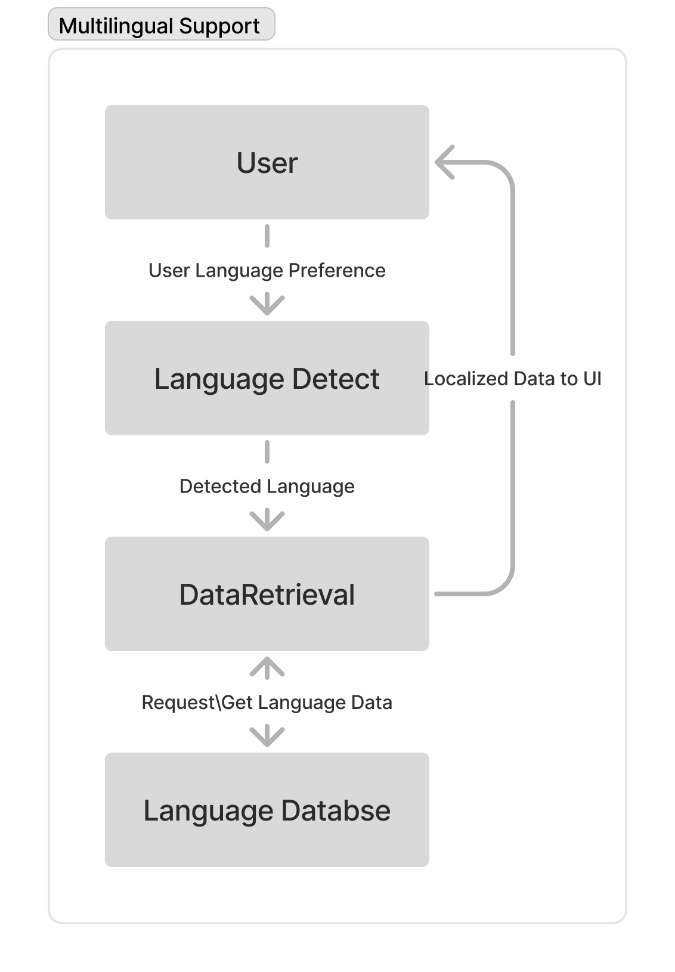
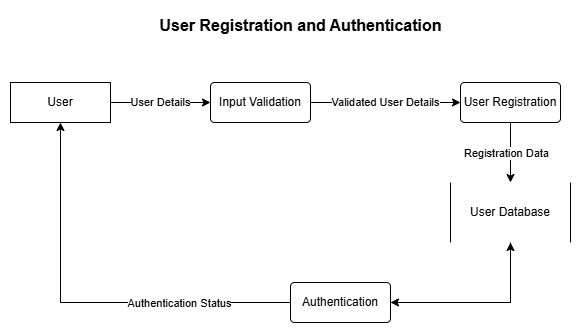
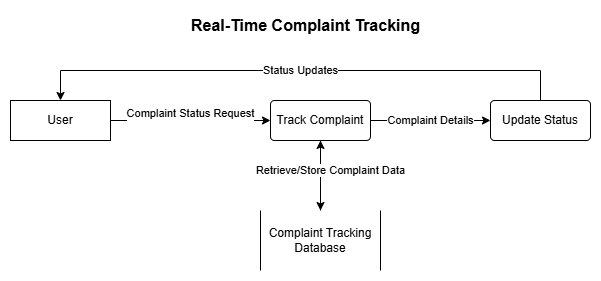
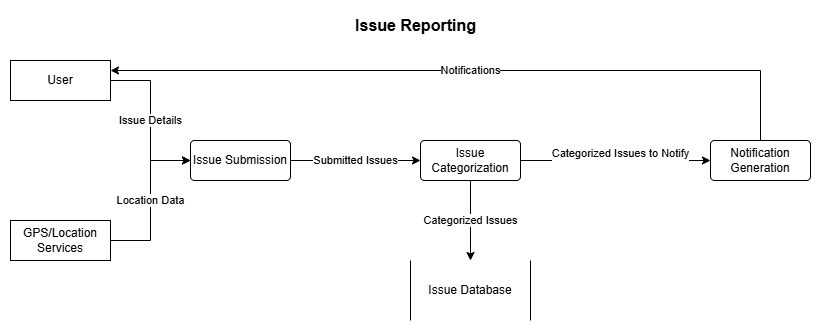
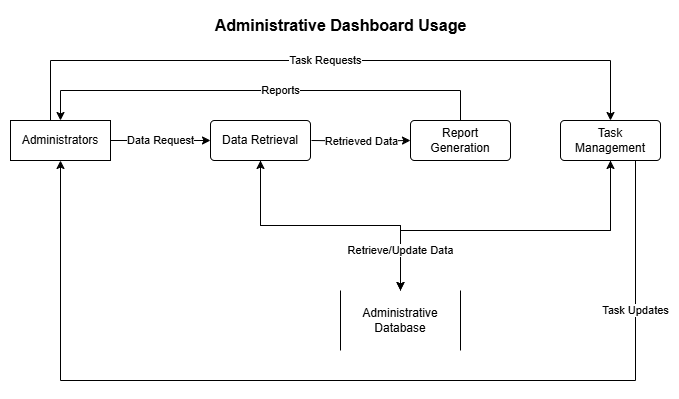
## 2.8 UI/UXD Interface Mockups

## 

# Process and Data Modeling

## **3.1 UML/DFD Modeling and Data Modeling**

### Activity Diagrams and Data Flow diagram



Activity Diagrams **A diagram of a process

AI-generated content may be incorrect.**

**A diagram of a flowchart

AI-generated content may be incorrect.**

**A diagram of a system

AI-generated content may be incorrect.**

**A diagram of a process

AI-generated content may be incorrect.**

**A screenshot of a diagram

AI-generated content may be incorrect.**

**A diagram of a data flow

AI-generated content may be incorrect.**

**A diagram of a process

AI-generated content may be incorrect.**

**A diagram of a language

AI-generated content may be incorrect.**

**A diagram of a company

AI-generated content may be incorrect.**

## **3.2 Business Rules**

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| BR01 | Users must provide a valid username, email, and password that meets specified security criteria to register. | UC01 |
| BR02 | Issue reports must be limited to 500 characters and include location data to ensure clarity and facilitate quick response. | UC02 |
| BR03 | Clerks must acknowledge new issue reports within 4 hours during operational hours to ensure timely responses. | UC03 |
| BR04 | The app must provide interfaces in English and French, defaulting to English when a user's preferred language is not supported. | UC04 |
| BR05 | High priority issues must be escalated to senior clerks or department heads within 1 hour of identification. | UC05 |
| BR06 | Reports and dashboards must reflect real-time data with a maximum delay of 5 minutes for updates. | UC06 |
| BR07 | Feedback collected through the app must be processed anonymously to protect user privacy. | UC07 |
| BR08 | Users must receive notifications of any updates or changes to their reported issues within 15 minutes of status change. | UC08 |
| BR09 | Administrators have exclusive access to system configurations, user role management, and critical data operations. | UC09 |
| BR10 | The user interface must adapt to the role of the logged-in user, displaying only the information and actions relevant to their responsibilities. | UC10 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## **3.3 Use Case Specifications with corresponding interface mockups:**

**Each use case needs to have the following:**

1- **Business Rules.**

1. **BR01 - User Registration Validation**
   * **Rule Description**: Users must provide a valid username, email, and password that meets specified security criteria to register.
   * **Related Use Case**: UC01 (User Registration)
2. **BR02 - Issue Reporting Constraints**
   * **Rule Description**: Issue reports must be limited to 500 characters and include location data to ensure clarity and facilitate quick response.
   * **Related Use Case**: UC02 (Issue Reporting)
3. **BR03 - Acknowledgment Timeframe for Reports**
   * **Rule Description**: Clerks must acknowledge new issue reports within 4 hours during operational hours to ensure timely responses.
   * **Related Use Case**: UC03 (Issue Acknowledgment)
4. **BR04 - Multilingual Interface Support**
   * **Rule Description**: The app must provide interfaces in English and French, defaulting to English when a user's preferred language is not supported.
   * **Related Use Case**: UC04 (Multilingual Support)
5. **BR05 - High Priority Issue Escalation**
   * **Rule Description**: High priority issues must be escalated to senior clerks or department heads within 1 hour of identification.
   * **Related Use Case**: UC05 (High Priority Escalation)
6. **BR06 - Real-time Reporting Accuracy**
   * **Rule Description**: Reports and dashboards must reflect real-time data with a maximum delay of 5 minutes for updates.
   * **Related Use Case**: UC06 (Real-time Reporting)
7. **BR07 - Anonymous Feedback Processing**
   * **Rule Description**: Feedback collected through the app must be processed anonymously to protect user privacy.
   * **Related Use Case**: UC07 (Feedback Processing)
8. **BR08 - Notification Timeliness**
   * **Rule Description**: Users must receive notifications of any updates or changes to their reported issues within 15 minutes of status change.
   * **Related Use Case**: UC08 (Notification System)
9. **BR09 - Administrator Role Access**
   * **Rule Description**: Administrators have exclusive access to system configurations, user role management, and critical data operations.
   * **Related Use Case**: UC09 (Administrator Access)
10. **BR10 - User Interface Customization**
    * **Rule Description**: The user interface must adapt to the role of the logged-in user, displaying only the information and actions relevant to their responsibilities.
    * **Related Use Case**: UC10 (Interface Customization)

**2- System Use Case Diagrams.**

**UC01 - User Registration Validation**

**A diagram of a system

AI-generated content may be incorrect.**

**UC02 - Issue Reporting**

**A diagram of a system

AI-generated content may be incorrect.**

**UC03 - Issue Acknowledgment**

**A diagram of a person with text

AI-generated content may be incorrect.**

**UC04 - Multilingual Support**

**A diagram of a system

AI-generated content may be incorrect.**

**UC05 - High Priority Escalation**

**A diagram of a system

AI-generated content may be incorrect.**

**UC** **06: Real-time Reporting Accuracy**

**A diagram of a data reporting system

AI-generated content may be incorrect.**

**UC07 - Anonymous Feedback Processing**

**A diagram of a person

AI-generated content may be incorrect.**

**UC08 - Notification System**

**A diagram of a system

AI-generated content may be incorrect.**

**UC09 - Administrator Access**

**A diagram of a system

AI-generated content may be incorrect.**

**UC10 -User Interface Customization**

**A diagram of a diagram

AI-generated content may be incorrect.**

**3- Use Case Descriptions.**

**UC01 - User Registration Validation**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | User Registration Validation | | |
| Triggering Event | A new user initiates the account registration process. | | |
| Brief Description | This use case allows Ontario residents to create an account in the Community Service App by providing their credentials. The system ensures data validation and security compliance before account creation. | | |
| Actors | - Resident (Primary)  - System (Secondary) | | |
| Related Use Cases | - UC10: Notification System (to send a confirmation email)  - UC09: Administrator Access (to manage user accounts) | | |
| Preconditions | - The user must have a valid email address.  - The system must be online and available. | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The user account is successfully created and stored in the database.  - A confirmation email is sent to the user. | | | |
| Flow of activities | Actor | | System |
|  | 1. | User selects the "Sign Up" option. | Displays the registration form. |
|  | 2. | User enters full name, email, password, and optionally a phone number. | Validates the email format and password strength. |
|  | 3. | User submits the registration form | Checks if the email is already registered. |
|  | 1. 4. | If valid, the system generates a unique user ID. | Stores user credentials securely in the database |
|  | 5. | System sends a confirmation email with an activation link. | Notifies the user to verify their account. |
|  | 6. | User clicks on the activation link. | Verifies the account and updates status to "Active." |
|  | 7. | User is redirected to the login page. | Displays a success message. |
| Exception Conditions | * User provides an email that is already associated with an existing account. * User enters a password that does not meet security criteria. * User enters an invalid email format. * User fails to verify their email within the required timeframe. * The system encounters an error during data processing or storage. * The activation link is expired or invalid. | | |

**UC** **02: Issue Reporting**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Issue Reporting | | |
| Triggering Event | A user initiates the process to report a municipal issue. | | |
| Brief Description | This use case allows Ontario residents to report issues such as potholes, garbage collection delays, and streetlight malfunctions. The system categorizes and assigns the issue to the appropriate department for resolution. | | |
| Actors | - Resident (Primary)  - System (Secondary)  - Clerk (Secondary) | | |
| Related Use Cases | - UC03: Issue Acknowledgment (for clerks to review issues)  - UC08: Notification System (to send status updates) | | |
| Preconditions | - The resident must be logged into their account.  - The system must have predefined issue categories available. | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The issue is recorded and assigned a unique tracking ID.  - The complaint is routed to the appropriate department for resolution.  - The user receives a confirmation notification. | | | |
| Flow of activities | Actor | | System |
|  | 1. | User navigates to the "Report an Issue" page. | Displays a form for issue reporting. |
|  | 2. | User selects the issue category (e.g., pothole, garbage collection, streetlight malfunction). | Confirms the selected category and prompts for additional details. |
|  | 3. | User enters a brief description (max 500 characters) and selects a location using GPS or manual entry. | Validates the input and checks location accuracy. |
|  | 1. 4. | User optionally uploads an image of the issue. | Accepts the image upload if it meets the format and size requirements. |
|  | 5. | User submits the issue report | Generates a unique tracking ID and assigns the issue to the appropriate department. |
|  | 6. | System sends a confirmation notification with the tracking ID. | Displays a success message and updates the user’s dashboard with complaint status. |
| Exception Conditions | * User does not provide mandatory information (category, description, or location). * Description exceeds 500-character limit. * GPS location is unavailable, and the user does not manually enter an address. * The uploaded image is in an unsupported format or exceeds size limits. * The system encounters an error while saving the complaint. | | |

**UC** **03: Issue Acknowledgment**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Issue Acknowledgment | | |
| Triggering Event | A new issue report is received by the system and assigned to a clerk. | | |
| Brief Description | This use case allows clerks to acknowledge receipt of an issue report within the required timeframe and begin the resolution process. | | |
| Actors | - Clerk (Primary)  - System (Secondary) | | |
| Related Use Cases | - UC02: Issue Reporting (to receive reports from users)  - UC05: High-Priority Issue Escalation (if an issue requires urgent action) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The system has received and categorized an issue report.  - The issue is assigned to an available clerk.  - The clerk is logged into the system. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The issue report status is updated to "Acknowledged."  - The reporting resident is notified of acknowledgment.  - The clerk is assigned responsibility for resolving the issue. | | | |
| Flow of activities | Actor | | System |
|  | 1. | Clerk logs into the system.. | Displays a list of newly assigned issue reports. |
|  | 2. | Clerk selects a specific issue report. | Shows report details, including category, description, location, and attachments. |
|  | 3. | Clerk reviews issue details. | Provides options to acknowledge, request more information, or escalate the issue. |
|  | 1. 4. | Clerk clicks on "Acknowledge Issue." | Updates the issue status to "Acknowledged." |
|  | 5. | System sends a notification to the resident. | Confirms that the issue is acknowledged and will be addressed. |
| Exception Conditions | * The clerk does not acknowledge the issue within the required timeframe (4 hours). * The system fails to assign an issue to an available clerk. * The system encounters an error when updating the acknowledgment status. * The clerk is unable to access the issue details due to a system malfunction. * The issue report contains insufficient or unclear details, requiring further information from the resident. | | |

**UC** **04: Multilingual Support**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Multilingual Support | | |
| Triggering Event | A user selects a preferred language or accesses the system for the first time | | |
| Brief Description | This use case ensures that users can access the Community Service App in English or French. If a user does not select a language, the system defaults to English. | | |
| Actors | - Resident (Primary)  - System (Secondary) | | |
| Related Use Cases | - UC01: User Registration (language preference selection during sign-up)  - UC10: Interface Customization (adapting UI based on language settings) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The system must support both English and French.  - The user must be logged in or accessing the system for the first time. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The system displays the interface in the user’s preferred language.  - All system messages, menus, and notifications are presented in the selected language. | | | |
| Flow of activities | Actor | | System |
|  | 1. | User accesses the system for the first time or navigates to language settings. | Displays language selection options (English and French). |
|  | 2. | User selects a preferred language. | Saves the user’s language preference. |
|  | 3. | User submits the selection. | Updates the system interface to the selected language. |
|  | 1. 4. | User navigates through the system. | Displays all text, buttons, and notifications in the selected language. |
|  | 5. | If no selection is made, the system defaults to English. | Stores English as the default language setting. |
| Exception Conditions | * The system fails to retrieve or save the user’s language preference. * The selected language is unavailable due to a system error. * Some system messages or notifications do not translate correctly, displaying in the default language instead. * The user attempts to select a language other than English or French, but the system does not support it. | | |

**UC** **05: High Priority Issue Escalation**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | High Priority Issue Escalation | | |
| Triggering Event | A clerk identifies an issue as high priority based on urgency or severity. | | |
| Brief Description | This use case ensures that high-priority issues (e.g., hazardous potholes, major garbage spills, malfunctioning streetlights in critical areas) are escalated to senior clerks or department heads for immediate resolution. | | |
| Actors | - Clerk (Primary)  - System (Secondary)  - Senior Clerk/Department Head (Secondary) | | |
| Related Use Cases | - UC03: Issue Acknowledgment (Clerk reviews and acknowledges the issue)  - UC08: Notification System (Updates resident and relevant personnel about escalation) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The issue has been reported and assigned to a clerk.  - The system must allow clerks to mark and escalate issues. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | -The issue is flagged as high priority.  - The issue is reassigned to a senior clerk or department head.  - The resident is notified about the escalation. | | | |
| Flow of activities | Actor | | System |
|  | 1. | Clerk logs into the system and reviews assigned issue reports. | Displays a list of new and pending issue reports. |
|  | 2. | Clerk identifies an issue as high priority based on severity. | Provides an "Escalate" option for urgent cases. |
|  | 3. | Clerk selects the "Escalate" option. | Prompts the clerk to add a reason for escalation. |
|  | 1. 4. | Clerk enters a justification and submits the escalation request. | Updates the issue status to "Escalated." |
|  | 5. | System reassigns the issue to a senior clerk or department head. | Sends an automatic notification to the senior personnel. |
|  | 6. | System notifies the resident about the escalation. | Updates the user’s dashboard with the new issue status. |
| Exception Conditions | * The system fails to reassign the escalated issue to a senior clerk. * The assigned senior clerk is unavailable, delaying the resolution process. * The issue is mistakenly escalated due to incorrect categorization. * The notification system fails to inform the resident about the escalation. * The clerk's escalation request does not meet predefined criteria, requiring review. | | |

**UC** **06: Real-time Reporting Accuracy**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Real-time Reporting Accuracy | | |
| Triggering Event | A user, clerk, or administrator accesses the system to view complaint statuses or generate reports. | | |
| Brief Description | This use case ensures that reports and dashboards display real-time data with a maximum delay of 5 minutes, providing accurate updates on issue statuses and complaint resolutions. | | |
| Actors | - Resident (Primary)  - Clerk (Secondary)  - Administrator (Secondary)  - System (Secondary) | | |
| Related Use Cases | - UC02: Issue Reporting (To update the complaint status)  - UC08: Notification System (To inform users about status updates) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The system must have access to real-time complaint data.  - Users must have the correct permissions to view reports. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The system updates all dashboards and reports within 5 minutes of any data change.  - Users receive up-to-date information on complaint statuses. | | | |
| Flow of activities | Actor | | System |
|  | 1. | User logs into the system and navigates to the complaint status page or reporting dashboard. | Retrieves the latest complaint data from the database. |
|  | 2. | User views complaint status updates or generates a report. | Displays real-time complaint statuses, response times, and other relevant metrics. |
|  | 3. | Clerk updates an issue status (e.g., from "Pending" to "In Progress"). | Logs the update and immediately reflects the change in dashboards. |
|  | 1. 4. | Administrator generates a report on complaint trends. | Fetches the latest complaint data, ensuring real-time accuracy. |
|  | 5. | System refreshes reports and dashboards every 5 minutes. | Ensures all users see the most updated data without manual refresh. |
| Exception Conditions | * System fails to refresh the data within the required 5-minute window. * Network latency causes delays in updating complaint statuses. * An issue is resolved, but the resident does not see the updated status immediately. * A generated report displays outdated data due to a system cache issue. * The database is temporarily unavailable, preventing real-time updates. | | |

**UC** **07: Anonymous Feedback Processing**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Anonymous Feedback Processing | | |
| Triggering Event | A resident submits feedback on the service provided for their reported issue. | | |
| Brief Description | This use case ensures that user feedback is collected and stored anonymously, protecting user privacy while allowing system administrators to assess service quality. | | |
| Actors | - Resident (Primary)  - System (Secondary)  - Administrator (Secondary) | | |
| Related Use Cases | - UC08: Notification System (To notify users about feedback submission)  - UC10: Interface Customization (Ensures feedback interface is user-friendly) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The reported issue must have been acknowledged and resolved.  - The user has access to the feedback submission page. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The feedback is recorded in the system without linking it to the user’s personal information.  - The administrator can analyze feedback for service improvement. | | | |
| Flow of activities | Actor | | System |
|  | 1. | User navigates to the feedback section after their issue is resolved. | Displays a feedback form with rating and comment options. |
|  | 2. | User provides a rating and an optional comment. | Ensures feedback is anonymous by not storing user-identifiable data. |
|  | 3. | User submits feedback. | Logs the feedback in the system without linking it to the user's identity. |
|  | 1. 4. | Administrator accesses the feedback dashboard. | Displays aggregated feedback data for analysis. |
|  | 5. | System generates reports on user satisfaction trends. | Provides insights into areas needing service improvements. |
| Exception Conditions | * The user submits feedback, but the system fails to anonymize it. * The feedback form is inaccessible due to a system error. * Feedback submission fails due to network or database issues. * The administrator is unable to retrieve feedback reports due to system malfunction. * The system mistakenly links feedback to a user’s identity, violating privacy policies | | |

**UC** **08: Notification System**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Notification System | | |
| Triggering Event | A change in issue status, escalation, or important system update occurs. | | |
| Brief Description | This use case ensures that users receive notifications about their reported issues within 15 minutes of a status change, keeping them informed and engaged. Notifications can be sent via the app, email, or SMS. | | |
| Actors | - Resident (Primary)  - Clerk (Secondary)  - System (Secondary) | | |
| Related Use Cases | - UC02: Issue Reporting (To notify users when their issue is logged)  - UC05: High Priority Issue Escalation (To notify users about escalated cases) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The system must be operational and capable of sending notifications.  - The user must have opted in for notifications via their preferred channel. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - Users receive timely notifications about updates related to their issues.  - The system logs notification history for tracking purposes. | | | |
| Flow of activities | Actor | | System |
|  | 1. | Clerk updates the status of a reported issue (e.g., "In Progress," "Resolved"). | Triggers a notification event in the system. |
|  | 2. | System verifies the user's notification preferences (app, email, SMS). | Selects the appropriate notification method. |
|  | 3. | System sends a notification within 15 minutes of the status change. | Ensures the notification is successfully delivered. |
|  | 1. 4. | User receives the notification and checks the updated issue status. | Updates the user’s dashboard accordingly. |
|  | 5. | System logs the notification event for auditing. | Stores the timestamp and details of the notification. |
| Exception Conditions | * The system fails to send a notification within the 15-minute window. * The user has disabled notifications, preventing updates from being received. * The email or phone number provided is invalid, causing delivery failure. * The notification system crashes, delaying or missing updates. * The user does not receive the notification due to network issues or spam filtering. | | |

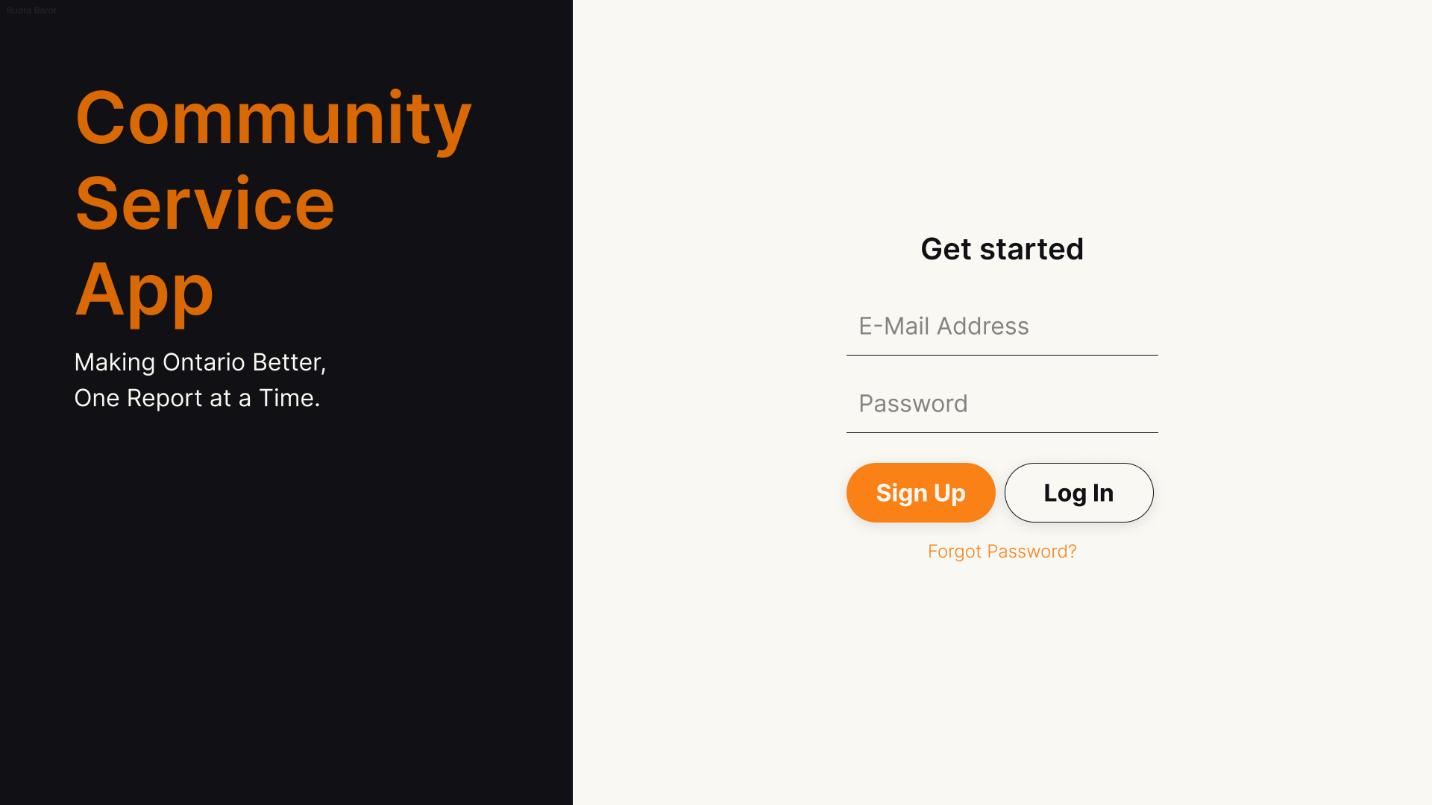
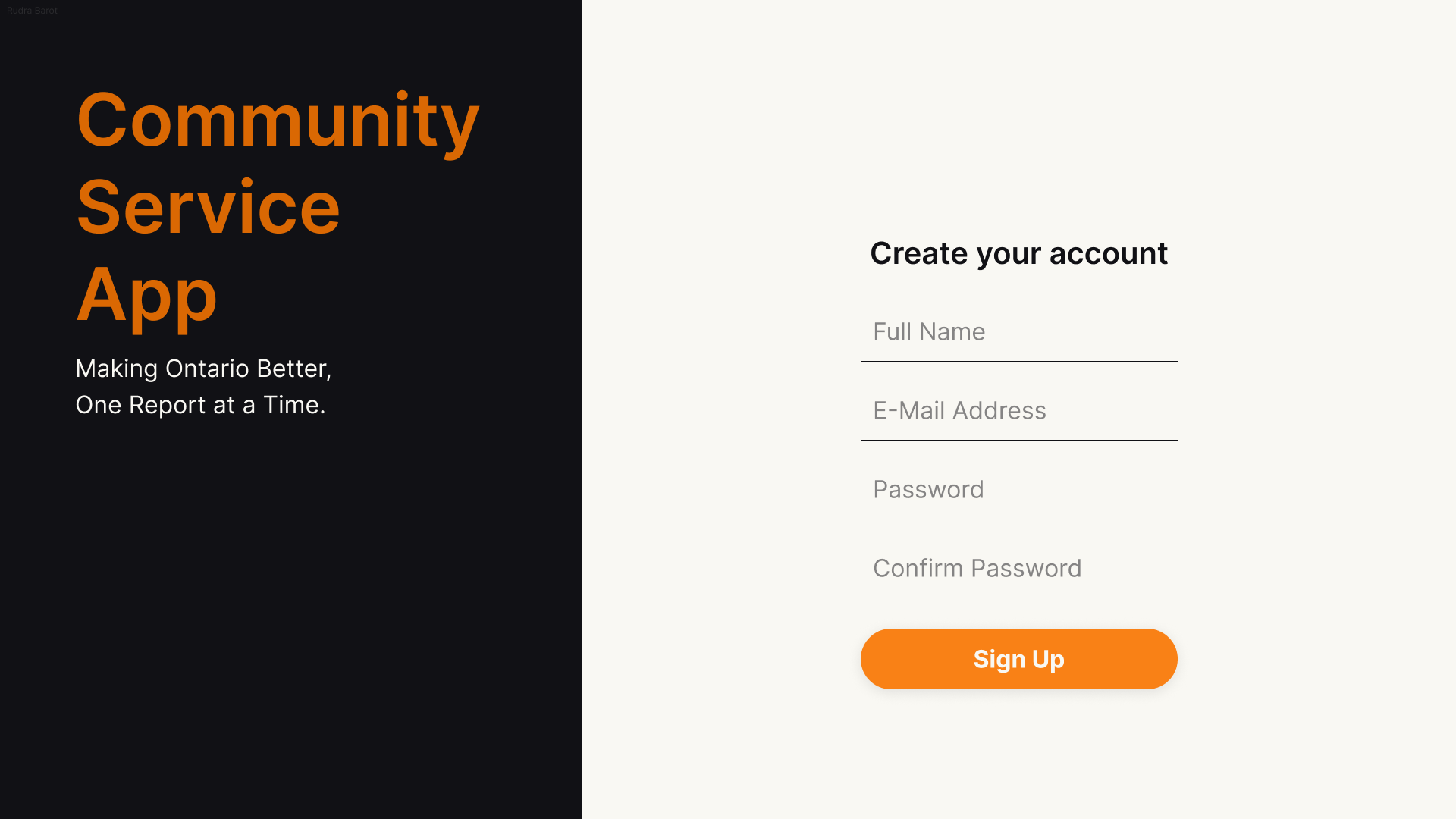
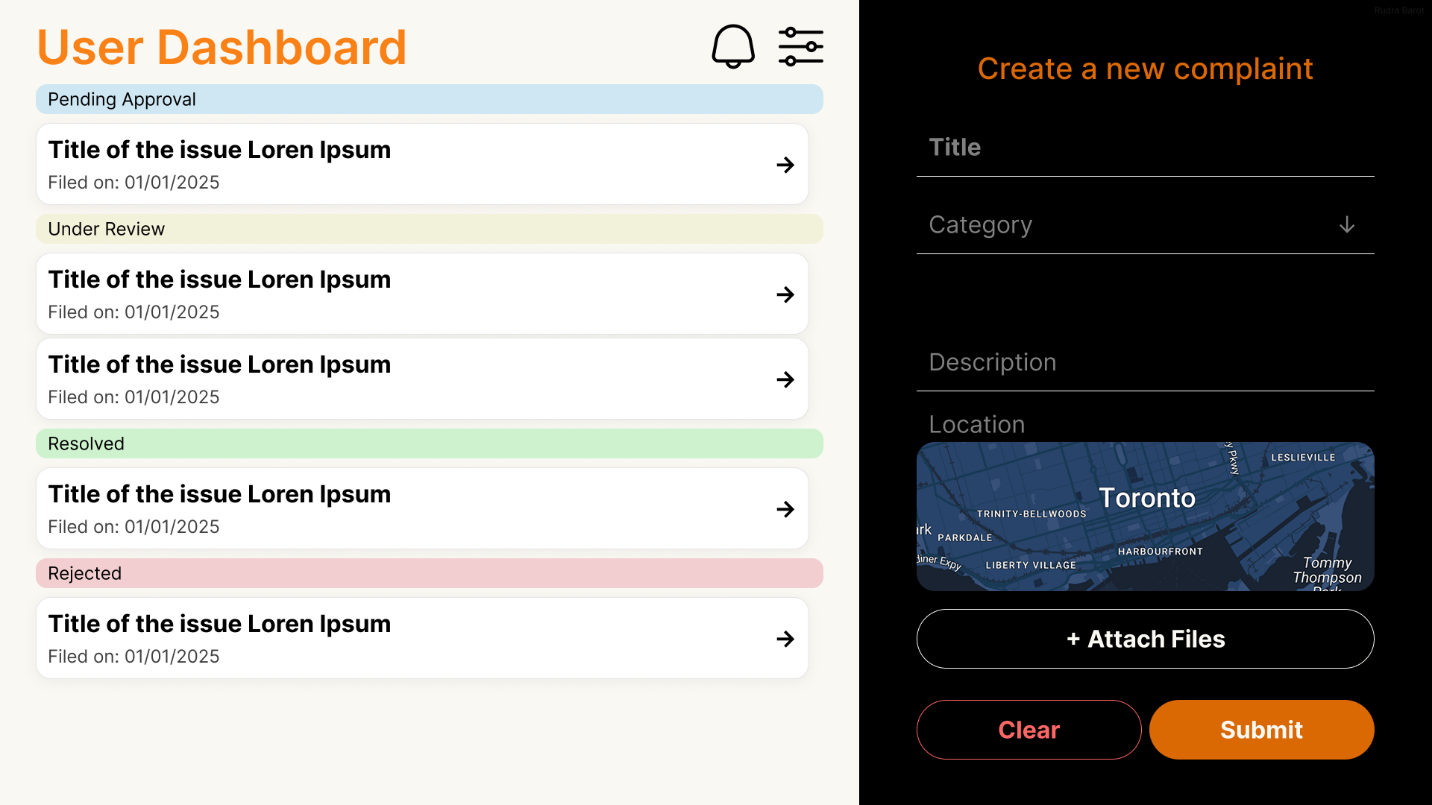
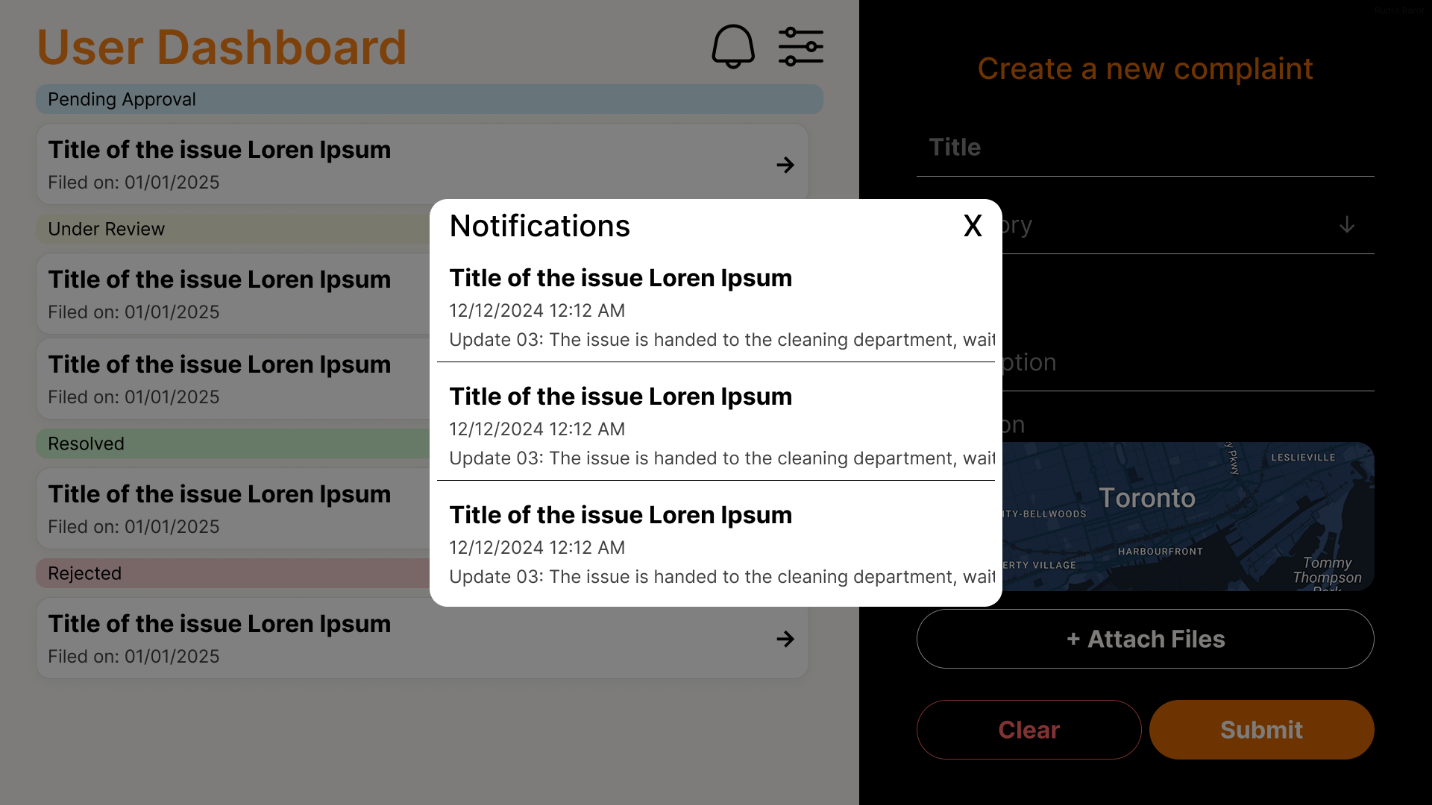
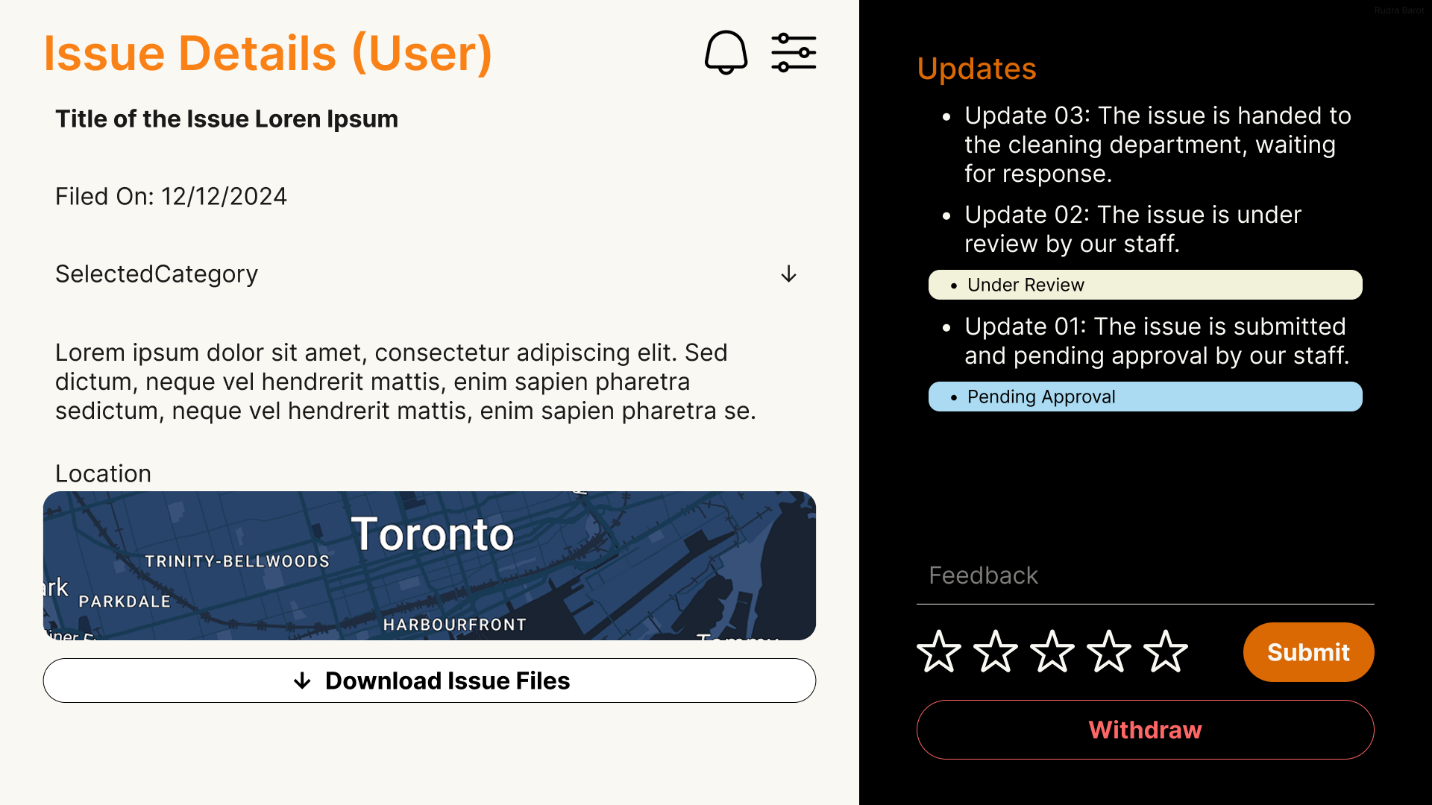
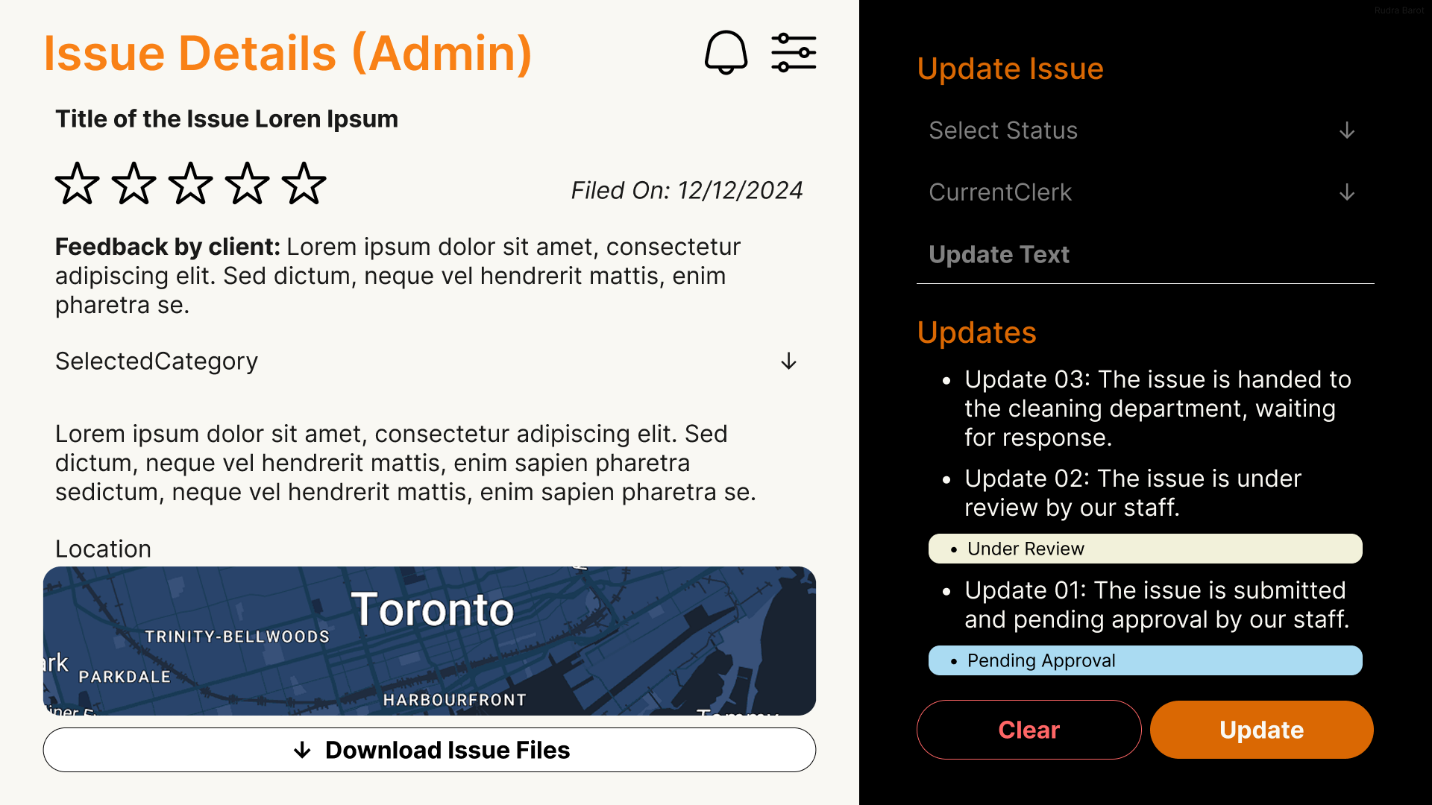
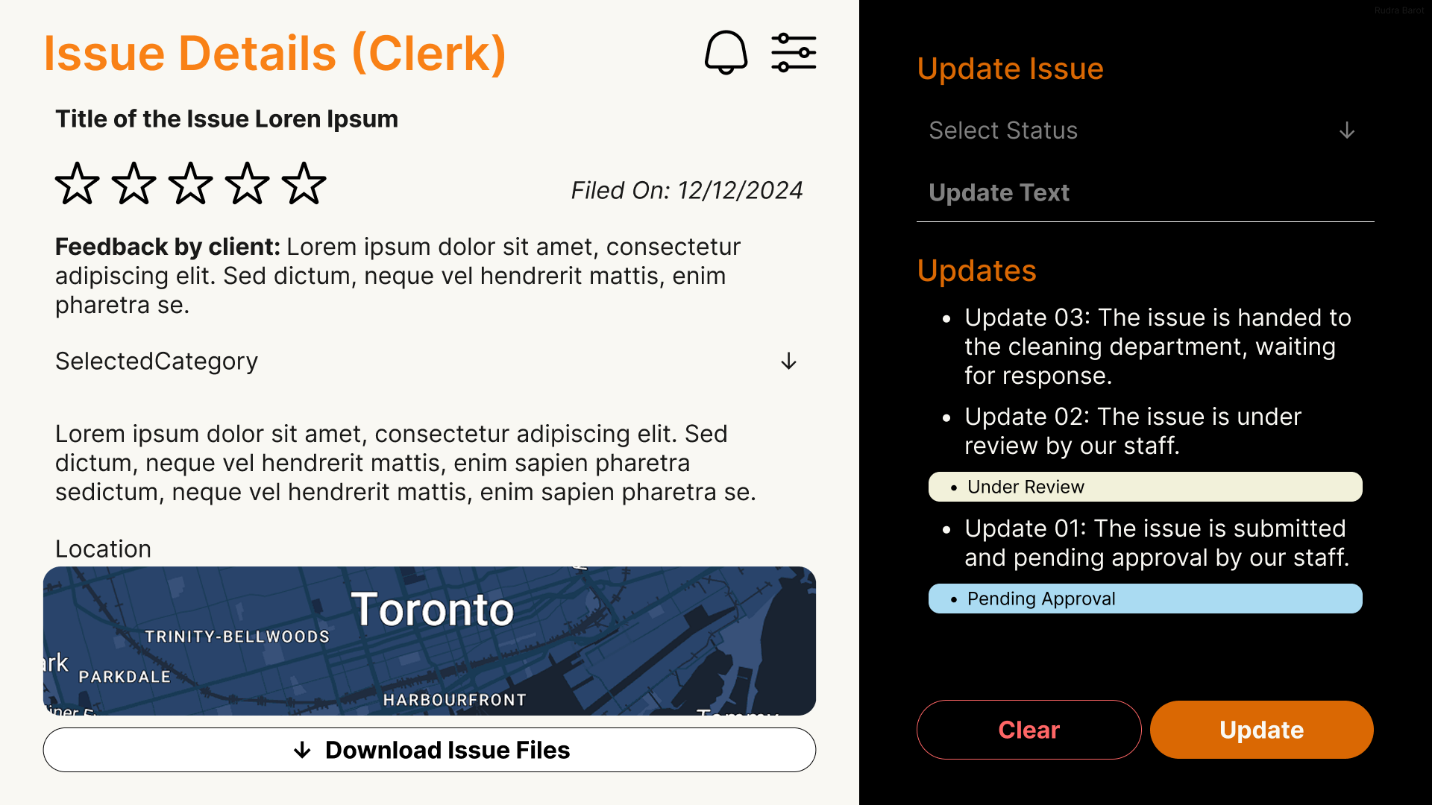
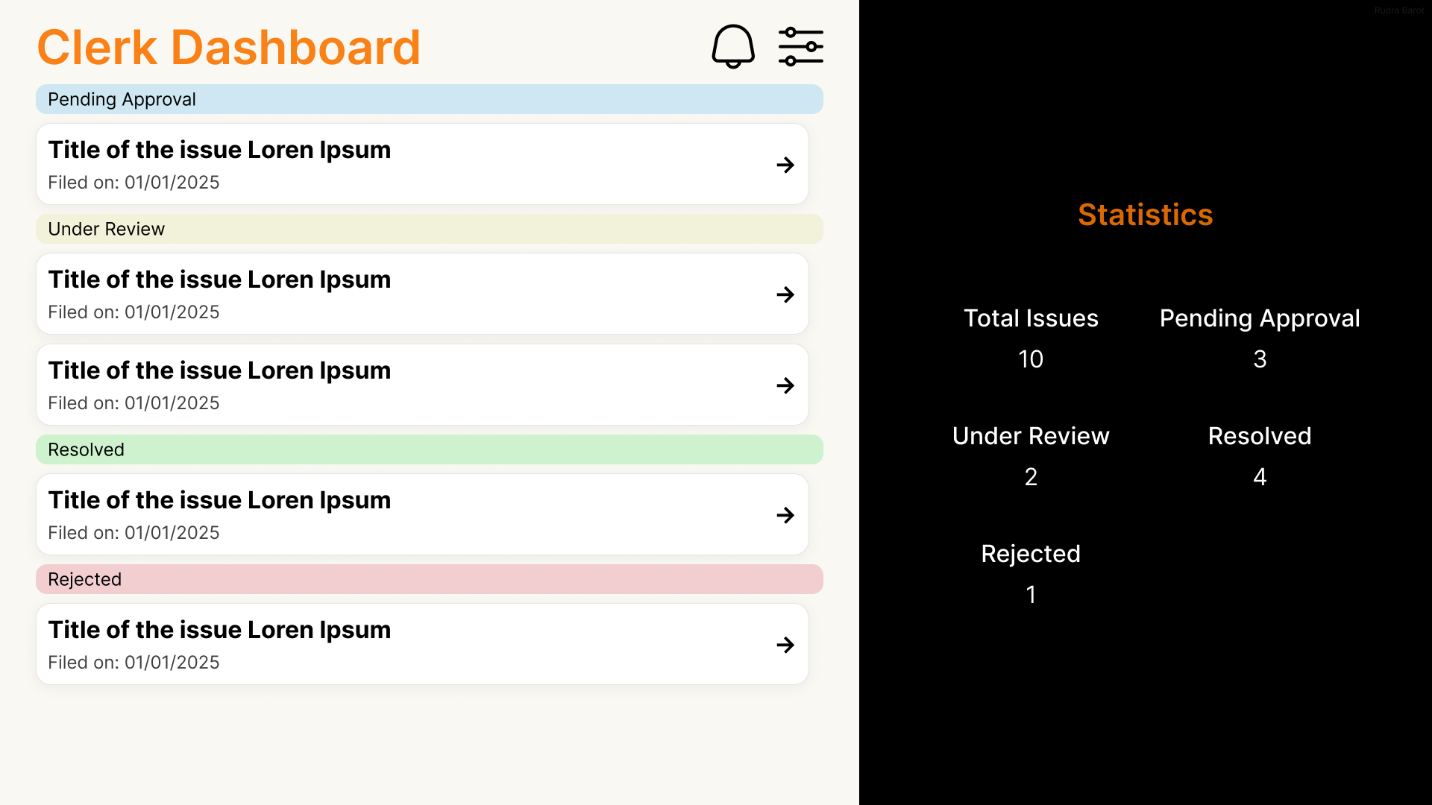
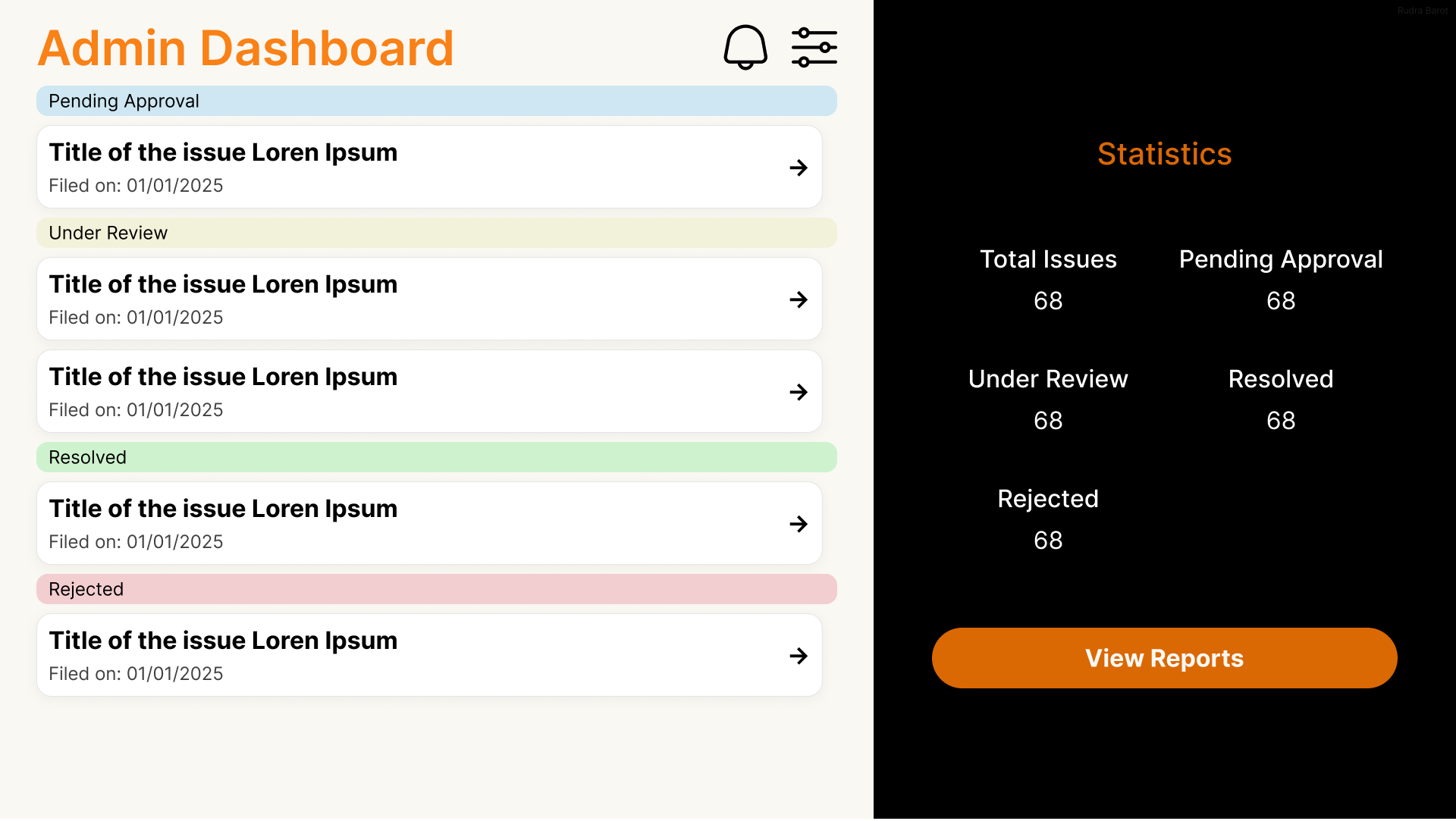
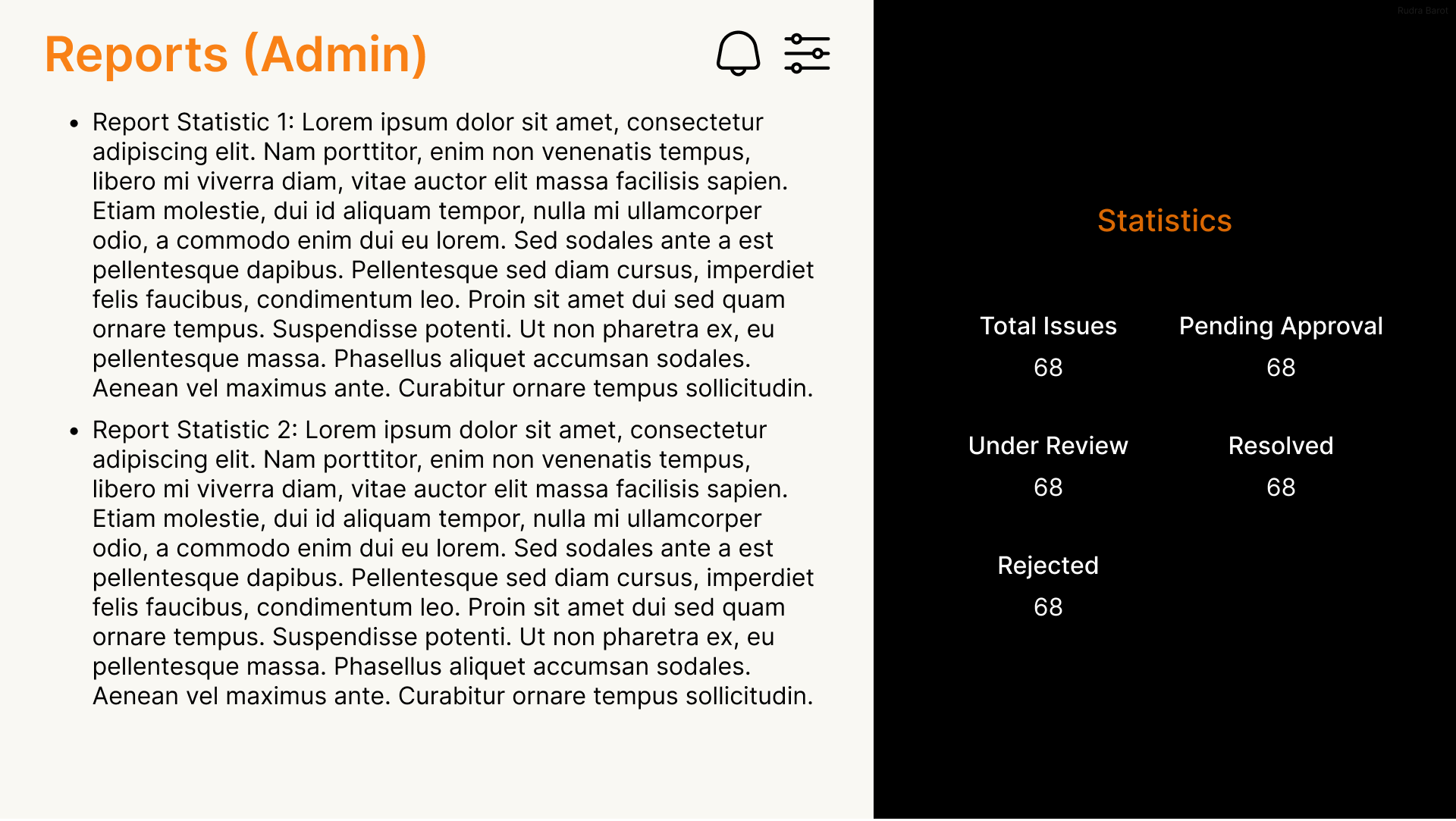
**UC** **09: Administrator Role Access**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Administrator Role Access | | |
| Triggering Event | An administrator logs into the system to manage configurations, user roles, or critical system settings. | | |
| Brief Description | This use case ensures that only administrators have access to system configurations, user management, and other critical data operations, maintaining security and role-based control. | | |
| Actors | - Administrator (Primary)  - System (Secondary) | | |
| Related Use Cases | - UC01: User Registration (Administrators can manage user accounts)  - UC06: Real-time Reporting Accuracy (Administrators can generate system-wide reports) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | -The administrator must have a valid account with the appropriate role permissions.  - The system must have security measures in place (e.g., role-based access control). | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The administrator can modify system configurations and manage users.  - All administrative actions are logged for audit purposes. | | | |
| Flow of activities | Actor | | System |
|  | 1. | Administrator navigates to the login page and enters credentials. | Verifies login credentials and role-based permissions. |
|  | 2. | Administrator accesses the system dashboard. | Displays administrative tools and settings. |
|  | 3. | Administrator manages system configurations, user roles, or service settings. | Applies changes and updates configurations. |
|  | 1. 4. | Administrator reviews reports and analytics. | Retrieves and displays system-wide data. |
|  | 5. | Administrator logs out. | Ends the session and logs actions for security tracking. |
| Exception Conditions | * An unauthorized user attempts to access administrative settings. * The administrator enters incorrect login credentials, leading to failed authentication. * The system fails to enforce role-based access control, allowing unauthorized modifications. * A configuration change causes system errors or disruptions. * Audit logs are not properly recorded, leading to compliance issues. | | |

**UC** **10: User Interface Customization**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | User Interface Customization | | |
| Triggering Event | A user logs into the system, and the interface adapts based on their role (Resident, Clerk, Administrator). | | |
| Brief Description | This use case ensures that the system customizes the user interface (UI) dynamically, displaying only relevant information and functionalities based on the user’s role. | | |
| Actors | - Resident (Primary)  - Clerk (Primary)  - Administrator (Primary)  - System (Secondary) | | |
| Related Use Cases | - UC09: Administrator Role Access (Administrators manage UI settings)  - UC02: Issue Reporting (Residents access issue submission UI)  - UC03: Issue Acknowledgment (Clerks access assigned reports UI) | | |
| Preconditions | |  | | --- | |  |  |  | | --- | | - The user must have a registered account and be logged in.  - The system must have predefined role-based UI layouts. | | | |
| Post Conditions | |  | | --- | |  |  |  | | --- | | - The user sees an interface tailored to their role.  - Unnecessary features and information are hidden to improve usability. | | | |
| Flow of activities | Actor | | System |
|  | 1. | User logs into the system. | Verifies login credentials and identifies the user role. |
|  | 2. | System loads the appropriate UI based on the user’s role. | Displays customized interface elements relevant to the user. |
|  | 3. | Resident accesses the system. | Displays issue reporting, status tracking, and feedback sections. |
|  | 1. 4. | Clerk accesses the system. | Displays assigned complaints, escalation options, and acknowledgment tools. |
|  | 5. | Administrator accesses the system. | Displays user management, system settings, and reporting dashboards. |
|  | 6. | User interacts with the system. | Ensures only role-specific actions are available. |
| Exception Conditions | * The system fails to load the correct UI based on the user’s role. * A resident sees administrator-level features due to an access control failure. * A clerk is unable to access their assigned issues due to UI misconfiguration. * The system crashes while loading the customized interface. * UI elements do not update dynamically when switching roles or permissions. | | |

**4- Corresponding Mockups**



# Domain Class Diagram

A diagram of a software application

AI-generated content may be incorrect.

# Database

Schemas for NoSQL MongoDB:

#### 1. Users Collection

In this design, both residents and staff are stored in one collection. A discriminator field (userType) indicates whether a user is a resident or staff member. Fields such as address are only applicable to residents, while department and role are used for staff.

{

"\_id": "user123",

"username": "johnDoe",

"email": "john@example.com",

"password": "hashed\_password",

"phoneNumber": "1234567890",

"preferredLanguage": "English",

"registrationDate": ISODate("2025-01-25T10:00:00Z"),

"userType": "resident", // Possible values: "resident", "clerk", "manager"

"address": "123 Main St, Toronto", // Only for residents

"department": null, // For clerks: e.g., "Public Works"

"role": null, // For clerks: e.g., "Clerk" or “Manager”

"managedDepartment": null // reference to the department they manage (e.g., "dept001")

}

#### 2. Departments Collection

This collection stores details for each department. Staff documents can reference a department using the \_id field.

{  
 "\_id": "dept001",  
 "name": "Public Works",  
 "contactEmail": "[publicworks@gov.on.ca](mailto:publicworks@gov.on.ca)"  
}

#### 3. Issues Collection

Each issue reported by a resident is stored in this collection. The document embeds a subdocument for the geolocation and uses an array (feedbacks) to store multiple feedback entries. References are included for the reporting resident and the staff member assigned to resolve the issue.

{  
 "\_id": "issue456",  
 "category": "Pothole",  
 "description": "Large pothole on Main St near 5th Ave.",  
 "location": {  
 "latitude": 43.6532,  
 "longitude": -79.3832,  
 "address": "Main St & 5th Ave, Toronto"  
 },  
 "status": "Open",  
 "priority": "High",  
 "residentId": "user123",   
 "assignedTo": "user789",   
 "reportedDate": ISODate("2025-01-26T08:30:00Z"),  
 "resolvedDate": null,  
 "feedbacks": [  
 {  
 "feedbackId": "fb001",  
 "rating": 5,  
 "comment": "Issue resolved quickly!",  
 "submissionDate": ISODate("2025-01-27T09:00:00Z"),  
 "anonymous": true  
 },  
 {  
 "feedbackId": "fb002",  
 "rating": 4,  
 "comment": "Quick service, but communication could improve.",  
 "submissionDate": ISODate("2025-01-27T10:15:00Z"),  
 "anonymous": false  
 }  
 ]  
}

#### 4. Notifications Collection

This collection stores notifications sent to users. Each document references the user who receives the notification.

{  
 "\_id": "notif001",  
 "userId": "user123", // Reference to the recipient  
 "message": "Your issue has been acknowledged.",  
 "type": "Update",  
 "status": "Sent",  
 "sentDate": ISODate("2025-01-26T09:15:00Z")  
}

# Work Breakdown Structure (WBS)

## Work Breakdown Structure

Community Service App Dev

Community Service App Dev   
 1. Requirements Gathering   
 1.1 Stakeholder Engagement   
 - Identify Stakeholders   
 - Conduct Interviews/Workshops   
 - Document Stakeholder Needs   
 1.2 Requirements Analysis   
 - Functional Requirements   
 - Non-Functional Requirements   
 - Regulatory & Compliance Requirements   
 - Risk Assessment   
 1.3 Requirements Validation   
 - Prototyping & Feedback   
 - Requirements Sign-Off   
 1.4 Documentation   
 - SRS Document   
 - Use Case Specifications   
 2. Designing Phase   
 2.1 Architecture Design   
 - System Architecture (Component Diagram, Data Flow Diagram, Integration Points)   
 - Database Design (Data Model/ER Diagram, Schema Definition, Indexing & Optimization)   
 - API Design (REST/GraphQL Specification, Endpoint Definitions)   
 - Security Architecture (Data Encryption, Access Control & RBAC, Compliance & Audit)   
 2.2 UI/UX Design   
 - Wireframes & Mockups   
 - Interactive Prototyping   
 - Usability Testing   
 2.3 Design Documentation   
 - Technical Documentation   
 - Style Guides & Patterns   
 3. Development Phase   
 3.1 Back-End Development   
 - Database Implementation (Schema Creation, Data Migration Strategies)   
 - Server-Side Logic (Business Logic Development, External API Integration)   
 - API Development (API Endpoints Coding, API Testing & Documentation)   
 3.2 Front-End Development   
 - Web Interface Development (Responsive Design, Browser Compatibility)   
 - Mobile App Development (Native/Hybrid App Coding, Cross-Platform Testing) 3.3 Integration Phase   
 - Third-Party Services Integration   
 - Continuous Integration Setup   
 - System Integration Testing   
 3.4 Quality Assurance   
 - Unit Testing   
 - Integration Testing   
 - Performance Testing   
 - Security Testing & Penetration Tests   
 4. Deployment   
 4.1 Pre-Deployment   
 - Environment Setup (Dev/Staging/Prod)   
 - Deployment Scripts & Automation   
 - Backup & Recovery Plans   
 4.2 User Acceptance Testing (UAT)   
 - UAT Planning   
 - UAT Execution & Feedback   
 4.3 Production Deployment   
 - Go-Live Preparation   
 - Rollout & Monitoring   
 - Post-Deployment Review   
 5. Maintenance & Support   
 - Regular Updates & Bug Fixes   
 - Performance Monitoring & Optimization   
 - Security Patches & Audits   
 - Help Desk & Technical Support   
 6. Project Management   
 - Project Planning & Scheduling (Milestone Definition, Resource Allocation, Budgeting & Cost Estimation)   
 - Progress Tracking & Reporting (Status Meetings, Risk Management, Issue Tracking)   
 - Communication & Documentation (Team Collaboration, Client/Stakeholder Reporting, Change Management)   
 7. Final Documentation & Training   
 - Technical Documentation   
 - User Manuals & Guides   
 - Training Sessions & Materials



# Milestones and Acceptance Criteria

#### 7. 1 Milestone 1: Project Initiation and Requirements Finalization

**Definition:**  
 Complete the initial planning and analysis phase by engaging with all relevant stakeholders, conducting comprehensive requirement gathering sessions, and identifying potential risks. Establish a clear project plan and secure formal approvals from all key parties.  
 **Acceptance Criteria:**  
 • A fully documented and approved Software Requirements Specification (SRS) along with a detailed project plan.  
 • All stakeholder interviews, workshops, and requirement sessions are documented with actionable outcomes.  
 • A risk register with identified risks and corresponding mitigation strategies is completed and signed off by the project leadership.

#### 7. 2 Milestone 2: Design Phase Completion

**Definition:**  
 Finalize the design aspects of the project, including user interface (UI) prototypes, system architecture diagrams, and both relational and NoSQL data models. Ensure that the designs accommodate multilingual support and role-based interfaces.  
 **Acceptance Criteria:**  
 • High-fidelity wireframes and interactive prototypes for residents, clerks, and managers are completed, reviewed, and approved by stakeholders.  
 • Detailed system architecture diagrams and comprehensive data models (ERD and MongoDB schema) are finalized and signed off.  
 • Integration designs with external APIs (e.g., Google Maps, OAuth 2.0) and middleware are fully documented and approved.

#### 7. 3 Milestone 3: Prototype Development for Key Features

**Definition:**  
 Develop a working prototype that demonstrates core functionalities such as secure user registration, issue reporting with geolocation support, and the notification system. Use this prototype to gather early feedback and validate design assumptions.  
 **Acceptance Criteria:**  
 • A functional prototype is developed that includes user registration with two-factor authentication and secure login capabilities.  
 • A working prototype for issue reporting and tracking is available, showcasing geolocation integration, image upload capabilities, and real-time updates.  
 • The prototype triggers test notifications within 15 minutes of simulated status changes, and early user feedback is documented.

#### 7. 4 Milestone 4: Core Backend and API Implementation

**Definition:**  
 Develop and integrate the core backend functionalities, including the creation of RESTful APIs for user management, issue processing, notifications, and feedback management. Ensure that all backend services are secure and scalable.  
 **Acceptance Criteria:**  
 • All RESTful APIs for user registration, login, issue reporting, assignment, and notifications are implemented, tested, and documented.  
 • Secure data storage practices (such as encrypted password storage and role-based access control) are in place.  
 • Comprehensive API documentation is provided, and regular code reviews have addressed all major security and performance concerns.

#### 7. 5 Milestone 5: Frontend Development and Integration

**Definition:**  
 Develop the user interfaces for all user roles (residents, clerks, and managers), ensuring a responsive design that supports multiple languages. Integrate these interfaces with the backend services to provide real-time data updates and role-based functionalities.  
 **Acceptance Criteria:**  
 • Fully functional and responsive dashboards and interfaces are developed for each user type with role-specific views.  
 • Multilingual support is operational with seamless switching between English and French.  
 • Successful integration of frontend components with backend APIs is demonstrated through real-time updates and comprehensive integration tests.

#### 7. 6 Milestone 6: System Integration and End-to-End Testing

**Definition:**  
 Combine all system components—frontend, backend, database, and external integrations—into a unified solution. Conduct comprehensive end-to-end testing to ensure that all modules work together seamlessly and reliably under real-world conditions.  
 **Acceptance Criteria:**  
 • Integration tests confirm that data flows correctly between the frontend and backend, and with all external services.  
 • End-to-end testing shows that all system modules interact seamlessly with no critical integration issues remaining.  
 • Test results and documented evidence of successful module interactions are provided, along with sign-offs from testing leads.

#### 7. 7 Milestone 7: Performance, Security, and User Acceptance Testing (UAT)

**Definition:**  
 Conduct rigorous performance, security, and user acceptance testing to validate the system’s ability to handle load, safeguard sensitive data, and provide a user-friendly experience. This phase will ensure the system meets operational targets before final deployment.  
 **Acceptance Criteria:**  
 • Load testing demonstrates that the system can support at least 10,000 concurrent users with response times consistently under 2 seconds.  
 • Security assessments, including penetration testing and vulnerability scans, reveal no critical security flaws.  
 • User acceptance testing (UAT) produces positive feedback from a representative sample of residents, clerks, and managers, with formal sign-offs obtained from UAT participants.

#### 7.8 Milestone 8: Production Deployment and User Onboarding

**Definition:**  
 Deploy the fully integrated system to the production environment, ensuring that all configurations, monitoring, and backup systems are in place. Conduct thorough training and onboarding sessions for all user groups to ensure smooth adoption.  
 **Acceptance Criteria:**  
 • The application is successfully deployed on production infrastructure (e.g., Vercel for hosting, MongoDB Atlas for the database) with performance and backup systems fully operational.  
 • Comprehensive user documentation, training materials, and support resources are distributed to all user groups.  
 • Successful completion of onboarding sessions is confirmed through sign-offs and initial user performance metrics.

#### 7. 9 Milestone 9: Post-Deployment Monitoring and Optimization

**Definition:**  
 Initiate continuous monitoring of the live system, address any post-deployment issues, and optimize performance based on real-world usage and user feedback. Implement improvements to enhance the overall user experience and system efficiency.  
 **Acceptance Criteria:**  
 • Monitoring tools confirm a system uptime of 99.9% or higher and track performance metrics in real time.  
 • All critical post-deployment issues are resolved within predefined Service Level Agreements (SLAs).  
 • Detailed reports document user feedback and implemented optimization measures, with evidence of improved performance metrics over time.

#### 7.10 Milestone 10: Final Project Review, Closure, and Maintenance Handover

**Definition:**  
 Conclude the project with a final review meeting, finalize all technical and user documentation, and formally hand over the system to the maintenance and support teams. Document lessons learned and best practices for future projects.  
 **Acceptance Criteria:**  
 • A comprehensive final project review is held with all key stakeholders, and formal sign-offs are obtained indicating project completion.  
 • All project documentation, including technical manuals, user guides, and training materials, is finalized, archived, and handed over to the support team.  
 • A formal maintenance plan and a detailed lessons learned report are prepared, approved, and distributed to ensure continuous improvement in future projects.

# Implementation Schedule

Implementation Schedule using MS Project (Waterfall)

OR

Product Backlog (Agile-Scrum)

**Community Service App Product Backlog**

**1. User Registration and Authentication**

**Epic:** Secure User Authentication and Role Management

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-001 | As a resident, I want to register using email, password, and basic details. | High | - Email format validated. - Passwords are hashed and stored securely. - Verification email sent upon registration. | Email Service Integration | 5 |
| CS-002 | |  | | --- | |  |  |  | | --- | | As a resident, I want to log in with my credentials. | | High | - Valid credentials are required for login. - Session tokens are generated post-login to maintain security. | CS-001 | 3 |
| CS-003 | As a user, I want my session to be securely managed. | High | - Admins can assign and update roles. - Users can access only authorized sections of the app. | CS-002 | 3 |
| CS-004 | As a user, I want to securely reset my password if I forget it. | High | - "Forgot Password" link available on login page. - Secure email sent for password reset. | Email Service | 8 |
| CS-006 | As a clerk or admin, I want to log in through a separate flow. | High | - Clerks/admins can log in with role-specific redirection.- Visual indicators to differentiate user roles. | CS-001, CS-005 | 3 |
| |  | | --- | | CS-022 | | As an admin, I want to register or promote users as clerks/admins. | Medium | - Admin can add/upgrade users with role assignment - Role change confirmation prompt. | CS-001, CS-005 | 4 |

**2. Role-Based Access Control (RBAC)**

**Epic:** Efficient Management of Community Service Requests

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-005 | As a user, I want UI and features shown based on my role. | High | - UI elements and pages rendered based on user roles (resident, clerk, admin).- Unauthorized access restricted. | CS-002, CS-023 | 5 |
| CS-023 | As a system, I should enforce permissions based on user roles. | High | - All restricted actions validate role permissions - Prevent API misuse and unauthorized access. | CS-005, CS-022 | 8 |
| CS-025 | As a user, I want to see a customized dashboard based on my role. | High |  Render only relevant features for logged-in user’s role. - Hide unauthorized UI elements. | CS-005, CS-023 | 4 |

1. **Issue Reporting**

**Epic:** Issue the report of the complain and make in detail for administrative.

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-007 | |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | As a resident, I want to report issues with category, image, description, and location. | |  |  | | --- | |  | | High | - Residents can submit issues with category, description, image, and location. - Validation for required fields. - Submit button triggers confirmation. | CS-002, CS-009, CS-008 | 5 |
| CS-008 | As a user, I want to upload images with file type/size validation. | Medium | - Accept only JPG, PNG; limit size (e.g., 5MB). - Preview before submission. -Display error for invalid formats/sizes. | CS-007 | 3 |
| CS-009 | |  | | --- | | As a user, I want to tag location via Google Maps. |  |  | | --- | |  | | High | - Google Maps API shows and stores pin for reported location.- Autofill coordinates from device if permission granted. | CS-007 | 5 |
| CS-010 | |  | | --- | |  |  |  | | --- | | As a system, I should store complaint data in MongoDB. | | High | - Complaint details saved securely. - Includes resident ID, category, location, and initial status. | CS-007, CS-027 | 3 |
| CS-011 | As a resident, I want to view the list and status of my reported issues. | High | - Resident dashboard shows all submitted issues and current statuses. - Option to filter by status or category. | CS-010 | 3 |

1. **Issue Management (Clerk/Admin)**

**Epic:** Efficient Management of Community Service Requests

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-012 | As a clerk, I want to view assigned tickets and update their status with notes. | High | - List of assigned tickets. -Status change options (e.g., Acknowledged, In Progress). -Add internal notes. | CS-010, CS-017 | 5 |
| CS-013 | |  | | --- | |  |  |  | | --- | | As a clerk, I want to escalate high priority issues with justification. | | High | - Option for clerk to escalate critical issues. - Mandatory justification field. - Sends notification to admin. | CS-012, CS-014 | 5 |
| CS-017 | As an admin, I want to view all issues and assign them to clerks. | High | - Admin can view unassigned tickets and assign them to clerks. - Filter tickets by category or urgency. | CS-010 | 5 |
| CS-018 | |  | | --- | |  |  |  | | --- | | As an admin, I want to filter issues and generate reports. | | Medium | - Admin can filter by status, type, department. - Export to CSV or view printable reports. | CS-010, CS-017 | 5 |
| CS-019 | |  | | --- | |  |  |  | | --- | | As a user, I want the dashboard to refresh automatically with updated ticket data. | | High | - Auto-refresh dashboards every 5 mins or on action. - Manual refresh button available. | CS-011, CS-012, CS-017 | 4 |

**5. Notifications and Communications**

**Epic:** Real-Time Notifications and Communications for Efficient Information Flow

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-014 | |  | | --- | | As a resident, I want to receive email or in-app notifications when ticket status changes. |  |  | | --- | |  | | High | - Residents notified on status changes via email or in-app alerts. - Opt-out/in options for email notifications. | CS-010, CS-012, CS-011 | 5 |
| CS-015 | As a resident, I want to give feedback after issue resolution. | Medium | - Display feedback form after issue is marked resolved. -Ratings and optional comments field. | CS-014 | 5 |
| CS-016 | |  | | --- | |  |  |  | | --- | | As a system, I should store feedback anonymously but linked to the resolution. | | Medium | -Store feedback without user identity while linking to resolved issue. - Admin can view feedback but not resident identity. | CS-015 | 3 |

**6. Multilingual Support**

**Epic:** Multilingual Interface to Cater to All Residents of Ontario

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-020 | As a user, I want to use the app in English or French, based on my preference. | High | - Language toggle on UI (English/French). - All static content translated. | CS-001 | 5 |
| CS-021 | As a user, I want my language preference saved and applied on login. | Medium | - Save user language choice and reload on login. - Stored in user profile or browser. | CS-020 | 3 |

**7. Analytics & System Configuration**

**Epic:** Advanced Analytics for Decision Making and Reporting

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-024 | As an admin, I want to view analytics on issues, resolution time, and feedback. | Medium | - Admin can view charts on issue trends, resolution time. - Ratings from feedback displayed. | CS-010, CS-016 | 4 |
| CS-026 | As an admin, I want to manage roles, default language, and other settings. | Medium | - Admins can change defaults (language, escalation settings). - Save and reflect across app immediately. | CS-005, CS-020 | 5 |

**8. Security & Infrastructure**

**Epic:** Enhance security of the application from the unknow user and rigid infrastructure

| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| --- | --- | --- | --- | --- | --- |
| CS-027 | As a system, I should securely store data in MongoDB with encryption and access control. | High | - Data encrypted in MongoDB with access control. - Use connection strings via .env | CS-010, CS-028 | 5 |
| CS-028 | As a developer, I want to set up GitHub, MongoDB Atlas, CI/CD, and linting tools. | High | - GitHub repo setup with CI/CD. - Vercel hosting, ESLint, Prettier, MongoDB Atlas connected. - .env file handled securely. | None | 5 |
| CS-029 | |  | | --- | |  |  |  | | --- | | As a system, I must encrypt data and comply with FIPPA and Ontario privacy laws. | | High | - Encrypt sensitive data in transit and at rest. - FIPPA and Ontario privacy guidelines followed. | CS-027 | 6 |
| CS-030 | |  | | --- | |  |  |  | | --- | | As an admin, I want backup and recovery plans for MongoDB. | | Medium | - Regular backups configured (e.g., daily). - Recovery tested and documented. | CS-027 | 4 |

# Client / Faculty Sign-off

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

X .

Name of Client/Rep/Professor