Software Requirements Specification

for the

City Grievance Redressal System

Contents

1 Introduction

This section provides an overview of the entire SRS document, including its purpose, scope, and definitions.

1.1 Purpose

The purpose of this document is to provide a detailed description of the requirements for the City Grievance Redressal System. It will define the system's intended purpose, features, and functionalities. The stakeholders for this project include the project development team (4 members), the supervising professor (acting as client and evaluator), the testers (the same team members), and the intended end users (as defined in the problem statement).

1.2 Scope

The software product, to be called the City Grievance Redressal System (CGRS), is a web-based system designed to streamline the reporting and resolution of civic issues.

The system will perform the following:

- Allow citizens to register and log in to the platform.
- Enable registered citizens to file new complaints with details, photos, concerned department and location.
- Allow citizens to track the status of their submitted complaints (pending, inProgress, resolved).
- Provide a feature for citizens to upvote existing complaints to highlight their urgency.
- Provide a dashboard for government authorities to view, manage, and update the status of assigned complaints **department-wise, with a consistent interface across departments**.

The primary goal of this project is to create a transparent and efficient communication bridge between citizens and government authorities responsible for resolving civic problems.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- CGRS: City Grievance Redressal System (The name of this software)
- Citizen: An end-user who registers to report and track grievances.
- Authority: An end-user, typically a government official, responsible for addressing and resolving grievances.
- UI: User Interface
- Grievance: A formal complaint or issue reported by a citizen.

1.4 References

1. IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications.*

1.5 Overview

This document is organized into three main sections. Section 1 provides an introduction and overview of the project and this document. Section 2 gives an overall description of the software, its context, user characteristics, and constraints. Section 3 details the specific functional and non-functional requirements of the system.

2 Overall Description

This section provides a high-level overview of the product, its users, and the operating environment, offering a background for the detailed requirements in Section 3.

2.1 Product Perspective

The CGRS is a new, self-contained, web-based system. It is designed to operate independently.

- **User Interfaces:** The system will provide a responsive web-based UI accessible via modern web browsers on desktops, tablets, and mobile phones.
- **Hardware Interfaces:** The system will require no special hardware on the client-side other than a standard internet-connected device.
- **Software Hosting:** Frontend (React + Tailwind) → will be deployed on Vercel. Backend (Node + Express) → will be deployed on Render. Database (MongoDB) → will be cloud hosted on MongoDB Atlas.
- **Software Interfaces:**
  - Frontend: Built using React.js and Tailwind CSS.
  - Backend: Built using Node.js with the Express.js framework.
  - Database: MongoDB will be used for data storage.
  - Client-Side: The system must be compatible with the latest versions of major web browsers (e.g., Chrome, Firefox, Safari, Edge).
- **Communications Interfaces:** Communication between the client and server will occur via HTTPS protocol.
  Email notifications will depend on a third-party service (e.g., Gmail SMTP, SendGrid, or Mailgun).

2.2 Product Functions

The system will provide the following summary of functions:

- User Authentication: Secure registration and login for 2 user types (Citizen, and Authority).
- Complaint Submission: Citizens can create and submit detailed complaints.
- Complaint Tracking: Citizens can view the real-time status of their submitted complaints.
- Authority Dashboard: Authorities can view a list of department wise assigned complaints sorted on the basis of number of upvotes and manage them.
- Complaint Management: Authorities can update the status and add comments to complaints.
- Community Engagement: Citizens can upvote existing complaints.

2.3 User Characteristics

- Citizens: The general public with varying levels of technical proficiency. The interface for citizens must be extremely intuitive, user-friendly, and simple to navigate.
- Authorities: Government or municipal employees who are expected to be comfortable using web applications. Their interface should be efficient and focused on task management.

2.4 Constraints

- The system must be implemented using the specified technology stack: React.js (Frontend) and Node.js with Express.js (Backend).
- The UI must be user-friendly and mobile-responsive.
- All data storage and transmission must be secure.
- The system must be scalable to handle a large number of user requests.

2.5 Assumptions and Dependencies

- **Assumptions:**
  - End-users will have a stable internet connection and a compatible device with a modern web browser.
  - The deployment environment will be available and properly configured.
- **Dependencies:**
  - The version control system will be Git/GitHub.

  - **The email notification system will rely on a third-party service (SMTP or API-based provider).**

3 Specific Requirements

This section provides detailed requirements. These requirements are organized by user class to provide clarity on the functionality available to different types of users.

3.1 External Interface Requirements

3.1.1 User Interfaces

- **Citizen Interface:**
  - Registration/Login screens.
  - A dashboard to view a list of personal submitted complaints and their statuses.
  - A form to submit a new complaint, including fields for title, description, location, concerned department and an option to upload photos.
  - A detailed view page for each complaint.
- **Authority Interface:**
  - Login screen.
  - A dashboard displaying a list of sorted complaints (no. of upvotes) assigned to them department wise.
  - A detailed view page for each complaint with options to update its status (e.g., 'In Progress', 'Resolved') and add official comments.

3.2 Functional Requirements

3.2.1 Citizen User Functions

- FR-C1: Registration: The system shall allow a citizen to register with a username, email, and password.
- FR-C2: Login: The system shall allow a registered citizen to log in using their credentials.
- FR-C3: Complaint Submission: The system shall provide a form for citizens to submit a complaint with a title, description, location, concerned department and photos.
- FR-C4: Complaint Tracking: The system shall allow a citizen to view the current status (e.g., 'Submitted', 'inProgress', 'Resolved') of all their complaints.
- FR-C5: Upvoting: The system shall allow citizens to view complaints submitted by others and upvote them.
- FR-C6: Notifications: The system shall notify citizens on their email when the status of their complaint is updated(backed by third-party service).

3.2.2 Authority User Functions

- FR-A1: Login: The system shall allow a registered authority to log in using their credentials.
- FR-A2: View Assigned Complaints: The system shall display a dashboard to the authority showing all complaints currently assigned to them department wise.
- FR-A3: Update Complaint Status: The system shall allow the authority to change the status of a complaint.

3.3 Performance Requirements

- The system's web interface shall be responsive, with page load times not exceeding 3 seconds under normal network conditions.
- The system must be capable of handling at least 100 concurrent user sessions without a noticeable degradation in performance.

3.4 Design Constraints

As stated in section 2.4, the system must be developed using React.js for the frontend and Node.js with Express.js for the backend. The database will be MongoDB.

3.5 Software System Attributes

- **Availability:** The system shall be available 24/7 with a target uptime of 99.5%, excluding scheduled maintenance periods.
- **Security:** Secure authentication is required for all users. Passwords must be stored in a hashed format. All communication between the client and server must be encrypted using HTTPS.
- **Maintainability:** The source code shall be well-documented and modular to allow for easy modifications and future enhancements.
- **Scalability:** The system architecture must be scalable to accommodate future growth in the number of users and complaints.