# **Overview** This SRS outlines the system description, functional and non-functional requirements, system architecture, data handling, and interface specifications of the NLC-CMS.

# The system is designed to manage citizen complaints related to urban infrastructure, including street lighting, with full integration into the latest cloud-ready, Dockerized CMS architecture.

# **Purpose**

This document outlines the Software Requirements Specification (SRS) for a CCMS Street Lighting Complaint Management System, a crucial component of the Cochin Smart City project. A dedicated application provides a centralized platform for citizens to report issues related to the smart lighting system.

This application will help for quick response of complaints, assign to help relevant teams for immediate attention, ensure timely resolution.

# **Scope**

# The CMS will enable:

# Citizens, administrators, and maintenance teams to log, track, and resolve complaints.

# Multilingual support (Malayalam, Hindi, English) for wider accessibility.

# Real-time notifications for status updates.

# Role-based dashboards and analytics.

# Containerized deployment supporting both development and production environments.

# Secure authentication and authorization, including JWT-based login and OTP verification.

# System configuration management, including complaint types, wards, and other system settings.

# **Overall Description**

1. Product Perspective  
   The CMS is part of the broader **Urban Smart Infrastructure Project**. It connects citizens with administrative and maintenance workflows and integrates with system-level monitoring, logging, and security frameworks.

The platform is **cloud-ready** with Docker deployment, automated builds, and scalable infrastructure management via PM2 and reverse proxy support (Nginx).

1. Product Functions

 **Complaint Registration** via web and mobile platforms.

 **Complaint Status Tracking** with real-time notifications.

 **Assignment of complaints** to departments or maintenance teams by Ward Officers.

 **Admin Dashboard** for monitoring and analytics.

 **Role-Based Access Control** for secure user management.

 **Reporting & Analytics** with heatmaps, trends, and exportable reports.

 **System Config Management**: Manage complaint types, wards, zones, and subzones.

1. Language Capabilities

* Malayalam
* Hindi
* English

1. User Characteristics

 **Citizens**: Lodge and track complaints using Email/OTP verification; view history.

 **Ward Officers**: Assign complaints, review status, and create complaints if required.

 **Maintenance Teams**: Update status to In Progress/Resolved; can create complaints if required.

 **Administrators**: Full system oversight; manage users, complaint types, zones, wards; reopen/close complaints.

# **Specific Requirements**

* 1. Functional Requirements

## 5.1.1 User Registration and Login

* Secure login for citizens, staff, and admins.
* OTP-based verification for citizens using Email.
* JWT authentication with token expiration for secure session management.

## 5.1.2 Complaint Submission

* Select complaint type (configured by Admin).
* Description field with optional photo uploads.
* Automatic **geolocation capture**.
* Option for “Other” complaint types outside predefined categories.

## 5.1.3 Complaint Tracking

* Statuses: Registered → Assigned → In Progress → Resolved → Closed.
* Citizens can view history and status of complaints logged.
* Admins and Ward Officers can track all complaints in their jurisdictio.

## 5.1.4 Complaint Management

* View, assign, update, reopen, or close complaints.
* Role-based access for editing and status updates.
* Admins can configure complaint types, wards, zones, and subzones.

## 5.1.5 Reporting and Analytics

* Complaint density **heatmaps**.
* Daily, weekly, and monthly complaint trend analysis.
* Exportable reports in CSV/Excel.
* Monitoring of system performance and logs.

## 5.1.6 Notification

* Email notifications on status changes

## 5.1.7 User Profile Management

* View and edit profile
* Change password and preferences

# **System Architecture**

## 6.1 Overall Architecture

* **Frontend**: React + TypeScript + Vite
* **Backend**: Node.js + Express + TypeScript
* **Database**: PostgreSQL with Prisma ORM
* **Process Management**: PM2 (cluster mode)
* **Security**: JWT authentication, bcrypt password hashing, Helmet.js headers
* **Deployment**: Dockerized with multi-stage builds for production

Responsive web application with microservices-style modular backend components. Scalable and cloud-ready deployment using Docker Compose.

6.2 Key Components

* **Frontend**: Component library, state management (Redux + RTK Query), routing (React Router v6).
* **Backend**: RESTful API, authentication/authorization middleware, complaint management logic, file management.
* **Database**: Users, Complaints, Departments, Status Logs, Notifications, Wards, Complaint Types.
* **Infrastructure**: Logging (Winston), monitoring (health checks), security middleware (CORS, rate limiting, input validation)

6.3 Database Design

* Tables: Users, Complaints, Departments, StatusLogs, Notifications, Wards, ComplaintTypes.
* Relationships:
  + **One-to-many**: Users → Complaints
  + **Many-to-one**: Complaint → StatusLogs
  + **One-to-many**: Wards → Complaints