# CHAPTER 1. INTRODUCTION

## PROJECT DESCRIPTION

### PROBLEM DEFINITION

Residents frequently encounter delays and inefficiencies when reporting and resolving municipal issues such as water pipe damage, blocked drainage, road damage, and streetlight failures. The current manual complaint management system is outdated, slow, and lacks transparency, leading to poor issue tracking and frequent misallocation of technicians. As a result, many problems remain unresolved or face significant delays, creating frustration among residents and lowering trust in civic authorities. One of the major drawbacks of the existing system is the absence of a mechanism to prioritize critical public complaints, which limits the overall effectiveness of municipal services. Without a clear way to differentiate between urgent and non-urgent issues, resources are often wasted, and critical problems go unattended. A smarter, automated system could significantly improve efficiency by streamlining complaint registration, assigning tasks to the right technicians, and providing real-time updates to residents. Such a system could also allow public voting on common issues, ensuring that the most pressing community concerns are addressed first. By incorporating data-driven insights and automation, municipalities could enhance service delivery, improve accountability, and foster stronger community engagement. This transformation would lead to faster resolutions, better resource management, and a more responsive civic infrastructure, ultimately improving the quality of life for all residents.

### PROPOSED SOLUTION

The **Smart Complaint Management System (SCMS)** is designed to simplify and streamline the process of reporting and resolving municipal complaints. It allows residents to easily submit complaints with detailed descriptions, images, and location data, ensuring clear communication of issues. The system features efficient task assignment automatically directing personal complaints to technicians based on their expertise and workload, which guarantees timely and quality resolutions. For public complaints, residents can use a user-friendly web interface to report issues, complete with geolocation and media attachments. An intuitive admin dashboard provides centralized monitoring and analytics, helping authorities manage resources and make informed decisions. Technicians have a dedicated portal to track tasks, update progress, and ensure expertise-driven problem-solving. To enhance convenience, the platform includes a secure payment gateway for premium or urgent services, offering flexibility for critical complaint resolutions. This integrated approach not only reduces delays but also increases transparency and accountability in municipal services. With real-time updates, smart notifications, and efficient resource management, SCMS transforms civic issue resolution into a smooth, engaging, and reliable experience for both residents and authorities.

### PURPOSE

The **Smart Complaint Management System (SCMS)** aims to revolutionize the way municipal complaints are reported and resolved. It simplifies and accelerates the process by offering a user-friendly interface for submitting detailed complaints with images and location data. Personal issues are efficiently allocated to technicians based on their expertise, ensuring quick and expert-driven resolutions. For public complaints, the system encourages community involvement through voting mechanisms, allowing authorities to prioritize the most critical issues. Real-time tracking enhances transparency and accountability, keeping residents informed about the status of their complaints. An intuitive admin dashboard helps optimize resource management and decision-making processes. Additionally, a secure payment gateway offers flexibility for specialized or urgent services, improving both scalability and convenience. By streamlining communication and automating task assignments, SCMS transforms civic issue resolution into a more effective and engaging experience

### SCOPE

The **Smart Complaint Management System (SCMS)** offers a seamless and efficient platform for addressing municipal issues. Residents can easily submit and track both personal and public complaints through user-friendly web and mobile applications. The system ensures expert-driven resolutions by automatically assigning personal complaints to technicians based on their specific expertise. Real-time complaint tracking allows residents and administrators to monitor progress via interactive dashboards, with timely updates sent through SMS, email, or app notifications. Technicians have their own portal to manage assigned tasks, update statuses, and provide resolution proof, promoting accountability and operational efficiency. A secure payment gateway supports premium or urgent services, offering multiple payment modes like UPI, credit/debit cards, and net banking for added convenience. Administrative oversight is enhanced through a powerful dashboard that monitors system performance, generates insightful analytics, and optimizes resource allocation. By integrating transparent tracking, efficient task allocation, and flexible payment options, SCMS transforms complaint management into a streamlined, responsive, and community-focused solution

# CHAPTER 2. LITERATURE SURVEY

## DOMAIN SURVEY

**2.1.1 Terminologies**

### Complaint Management System (CMS): A system designed to receive, manage, track, and resolve

### complaints or service requests. In this context, it handles both personal (private issues) and public complaints (community issues).

### Personal Complaints: Complaints related to private residential issues such as water pipe damage, plumbing issues blocked drainage, or electrical faults. These complaints are generally assigned to specific technicians based on their expertise.

### Public Complaints: Complaints that affect the public or community, such as road damage, streetlight failures, and environmental concerns. These complaints often require public authorities or municipal

### bodies to take action.

### Technicians: Trained personnel assigned to resolve personal complaints based on their expertise.

### They are dispatched by the system to the complaint's location.

* + 1. **Associated Concepts**

1. **Urban Service Delivery:** The provision of public services (e.g. waste management, road repairs, utility maintenance) to residents in urban areas. The CMS acts as an intermediary to ensure that service delivery is prompt and effective.
2. **Public Service Management:** The management of services that cater to the needs of the general public, such as road maintenance, streetlight repair, or public sanitation. This is a key area for integrating technology with service provision.
3. **Public Safety:** The protection of citizens through services that enhance safety, such as the maintenance of streetlights, roads, and public spaces. The CMS contributes to public safety by ensuring that complaints related to hazards are quickly addressed.
4. **Infrastructure Management:** The monitoring and maintenance of physical infrastructure like roads, bridges, drainage systems, and utility lines. Complaints related to infrastructure are a key part of the public complaints category in the system.

## RELATED WORK

### 2.2.1 RESEARCH PAPER-1

**Title:** “Complaint Registration System Through Web Services”

**Authors:** Dr. D. Venkata Subramanian, Dr. K.G.S. Venkatesan, M. Praveena, U. Prasad, N. Narasimha Rao

**Published Date:** December 2021

**Publication:** International Journal of Computational Intelligence in Control

**Summary:** This paper focuses on improving municipal complaint management through web-based services. It incorporates IT technologies like Google APIs for geolocation and Firebase for real-time notifications, streamlining the complaint registration process and enhancing user experience. The system facilitates smoother interactions between residents and local authorities, leading to faster resolutions and improved transparency. By leveraging modern technologies, this approach fosters greater civic engagement and boosts the overall efficiency of municipal services**.**

### RESEARCH PAPER-2

**Title**: “An e-Government Portal for Citizens’ Complaints in Government Services Using Web Crawling”

**Authors**: Shirley D. Moraga, Albert A. Vinluan  
**Published Date**: November 2018  
**Publication**: Asia Pacific Journal of Multidisciplinary Research

**Summary**: The "e-Reklamo" portal uses web crawling technology to collect citizen complaints from government websites and social media platforms. Its key features include tag cloud visualization and topic modelling to facilitate effective decision-making by categorizing complaints. This system enhances transparency and boosts service delivery efficiency by consolidating citizen feedback into a centralized platform, allowing for swift issue resolution by authorities.

### 2.2.3 RESEARCH PAPER-3

**Title**: “Public Services and Citizen Complaints: Economic, Social, and Political Determinants of Citizen Complaints”

**Authors**: Marianna Sebő and Germà Bel  
**Published Date**: November 2023  
**Publication**: International Public Management Journal

**Summary**: This study examines the influence of socioeconomic and political factors on public service complaints. Utilizing data from 2014–2019, it identifies correlations between political participation and complaint filing, providing valuable insights into citizen satisfaction and the role of e-governance in enhancing public service delivery.

### 2.2.4 RESEARCH PAPER-4

**Title**: “m-Government Solution for Complaint Management and Management-Designing a Solution for Government 2.0”

**Authors**: Carl-Mikael Lönn  
**Published Date**: 2014  
**Publication**: DSV Report Series, Stockholm University

**Summary**: The paper introduces "Munizapp" an m-government platform designed for Swedish municipalities, enabling real-time complaint reporting through mobile devices. It emphasizes transparency and responsiveness while also addressing organizational barriers in the adoption of mobile governance solutions.

### 2.2.5 RESEARCH PAPER-5

**Title:** “From Bulletin Boards: The Origins and Evolution of Public Complaint Website in China”

**Authors:** Christian Göbel and Jie Li

**Published Date:**2021,

**Publication:** Journal of Current Chinese Affairs,

**Summary:** This paper traces the development of complaint platforms in China, focusing on the adoption of big data analytics to improve transparency. It examines how policy innovation and governance practices have influenced public service delivery, offering a historical perspective on the evolution of e-governance.

### 2.2.6 RESEARCH PAPER-6

**Title: “**A Protrait of E-Government Implementation in Public Complaint Services**”**

**Authors:** Arif Wibowo and Dian Novita Lestari

**PublishedDate**:2023  
**Publication**: Journal of Public Administration Studies

**Summary**: Focuses on digital platforms enhancing citizen engagement and government accountability in Indonesia. It discusses challenges and successes in adopting e-government solutions, emphasizing improved responsiveness and citizen satisfaction.

## EXISTING SYSTEM

### 2.2.1 Personal and Public Complaints: The system allows residents to report personal complaints, such as water pipe damage and blocked drainage, as well as public complaints like road damage and streetlight failures, streamlining the reporting process.

### 2.2.2 Efficient Task Assignment: Personal complaints are routed to technicians based on their expertise, ensuring that tasks are handled by the most qualified individuals, resulting in faster and more effective resolutions

* + 1. **Payment and Resolution:** Technicians can resolve personal complaints and collect payments for their services directly through the platform, ensuring a seamless process for both service providers.
    2. **Automated Notifications**: The system sends automatic updates to users and technicians about the status of complaints, ensuring that all parties are informed of progress and next steps, which enhances communication and satisfaction**.**
    3. **User-Friendly Interface**: The system offers an intuitive interface for both residents and technicians, making it easy for them to submit complaints, track progress, and resolve issues efficiently, regardless of technical expertise.
    4. **Citizen Engagement**: Public complaints are sent to registered users, allowing the community to participate in the problem-solving process and vote on issues that require attention, fostering greater civic engagement

# CHAPTER 3. HARDWARE AND SOFTWARE REQUIREMENTS

## HARDWARE REQUIREMENTS

### COMPUTER

**RAM**: 8GB and above

**ROM**: 16GB

**Processor**: Intel i5 or above

**Processor speed**: 2.3GHz

For the effective performance of the Smart Complaint Management System, the hardware requirements are

optimized for handling multiple users and real-time data processing. A computer with at least 8GB of RAM

ensures smooth multitasking, while 16GB of ROM provides sufficient storage for the application’s files, user

data, and system logs. The Intel i5 processor with a speed of 2.3GHz or above enables the system to handle

complex tasks such as real-time complaint tracking, payment processing, and communication between the front-

end, back-end, and database efficiently. These specifications ensure that the application runs smoothly and

delivers timely responses, enhancing the overall user experience.

## SOFTWARE REQUIREMENTS

**IDE**: Visual Studio Code (Version: 1.85)

**Operating System**: Windows 10 or above

**Front end**: React.js (Version: 18.2.0)

**Server**: Express (Version: 4.18.2)

**Back end**: Node.js (Version: 18.15.0)

**Database**: MongoDB Atlas (Latest Version)

For developing and deploying the Smart Complaint Management Application, the required software stack

includes a reliable and feature-rich IDE like Visual Studio Code, which supports various extensions for seamless

coding in JavaScript and related frameworks. The system requires Windows 10 or a newer operating system to

ensure compatibility with modern development tools and libraries. The front-end is built using React.js, enabling

a dynamic user interface, while the server is powered by Express.js for efficient routing and middleware

management. The back- end is implemented with Node.js, providing scalability and speed for real-time

application demands. MongoDB Atlas is used for cloud-based data storage, offering flexibility and scalability

for managing complaint data and user interactions. These technologies work together to create a high-

performance and responsive complaint management system.

# CHAPTER 4. SOFTWARE REQUIREMENTS SPECIFICATION

## 

#### **Admin**

The admin oversees the complete operation of the Smart Complaint Management System, ensuring its smooth functionality. Admins have access to a dashboard that allows them to monitor all complaints, view user data, and track complaint statuses. Admins can download detailed complaint data in CSV format for reporting and analysis. They have the authority to manage and assign complaints to technicians based on expertise, ensuring efficient resolution. Admins also handle payment confirmations, enabling the processing of payments once complaints are resolved. By maintaining the integrity of the system and ensuring that complaints are addressed promptly, admins ensure transparency and accountability across the platform.

#### **Applicants(user)**

Residents are individuals who report issues through the platform, ranging from personal complaints (e.g., water pipe damage, blocked drainage) to public complaints (e.g., streetlight failures, road damage). The system allows residents to submit complaints along with images of the issues for better clarity. They can track the progress of their complaints and make payments once a complaint is resolved. The platform provides a user-friendly interface, allowing citizens to easily upload images, monitor updates, and engage in real-time communication with technicians or administrators for quick resolution.

#### **Technicians**

Technicians are responsible for resolving personal complaints that require specific expertise. They receive notifications for complaints assigned to them by the admin, and the system allows them to view and prioritize their tasks. Technicians can access detailed complaint descriptions, including images submitted by residents, to understand the issue better. Once they complete the tasks, technicians can mark complaints as resolved and update the status accordingly. They can also manage payment acceptance for their services through the platform, ensuring that the payment process is integrated into the resolution workflow. Technicians play a key role in ensuring that complaints are addressed efficiently, and their feedback and task progress are essential for maintaining system transparency and performance.

## FUNCTIONAL REQUIREMENTS

#### **Registration and Login**

Residents, technicians, and administrators can register and log in with their respective roles to access the system securely.

#### **Complaint Submission:**

Residents can submit both personal and public complaints by providing a detailed description, address, and uploading supporting images for better clarity.

#### **Complaint Tracking:**

Residents, technicians, and administrators can track the real-time status of complaints, ensuring transparency and timely resolution at all stages.

#### **Resume Score**

This score will be generated by assessing factors like relevance of experience, education, skills, certifications and other indicators of a candidate. By analyzing the content of the resume using NLP, the application predicts or suggests potential job roles or positions that align with the candidate's skills and experience. This aids the user in tailoring their job applications more effectively.

#### **Public Complaint Voting**

Registered users can vote on public complaints to help prioritize the most urgent issues, ensuring that critical concerns are addressed first.

#### **Admin Dashboard**

Administrators can manage complaints, monitor technician performance, and generate reports for data analysis, ensuring smooth operations and effective management of the system.

#### **Technician Management**

Technicians can manage their profiles, view assigned tasks, and mark complaints as resolved once completed, maintaining an organized workflow.

#### **Feedback and Rating**

After a complaint is resolved, residents can provide feedback and rate technicians based on their work quality, ensuring continuous service improvement.

#### **Technician’s View**

Technicians are assigned complaints based on their expertise. Each complaint comes with a resolution deadline, and technicians are required to complete the task before this deadline.

#### **Task Status Updates**

Technicians can update the status of their tasks (e.g., In Progress, Resolved). The system automatically adjusts the deadline if the task is extended, ensuring that progress is tracked accurately.

**4.2.11** **Notifications and Alerts**:  
Both users and technicians receive instant notifications and alerts about complaint updates, task assignments, and status changes, keeping everyone informed.

**4.2.12 Routine Maintenance Reminders**:  
Technicians or residents receive reminders for scheduled maintenance tasks, helping to ensure that periodic checks and updates are not missed.

**4.2.13 Payment Integration for Personal Complaints**:  
The system integrates with popular online payment services for personal complaints. Once payment is made, residents can download a receipt for their records, ensuring a seamless payment process.

## NON-FUNCTIONAL REQUIREMENTS

#### **Usability**

The user interface is crafted with a focus on simplicity and user-friendliness. It features clear navigation menus and intuitive controls to make the tool accessible to users with varying levels of technical expertise. The process of uploading resumes, viewing analysis results, and accessing recommendations is designed to be straightforward, minimizing the learning curve for users.

#### **Reliability**

This ensures that the application is available and accessible to users for the majority of the time. Minimize downtime for maintenance or unexpected issues.

#### **Performance**

The application establishes acceptable response time benchmarks for each step of the resume analysis process. This includes the time taken for parsing, generating recommendations, and displaying results to the user.

#### **Compatibility**

The tool will be compatible with different browsers, ensuring consistent user experience.

* + 1. **Scalability**

The system should be able to scale horizontally to accommodate an increasing number of users and complaints without significant re-architecture.

* + 1. **Security**

User data must be encrypted both in transit and at rest to protect sensitive information.

# CHAPTER 5. SYSTEM DESIGN

## ARCHITECTURE DIAGRAM



Fig5.1. Architecture diagram

The architecture diagram of the **Smart Complaint Management System** consists of three main layers: the **front-end**, **server**, and **back-end**. The **front-end** (React.js) is responsible for the user interface, allowing residents, technicians, and admins to interact with the system. The **server** (Express.js) handles requests and responses, serving as the intermediary between the front-end and back-end, managing authentication and API calls. The **backend** (Node.js) processes business logic, stores data in the database (MongoDB Atlas), and manages tasks such as complaint submission, status updates, and payments. This layered architecture ensures smooth communication between components, real-time updates, and secure data handling.

## CONTECT DIAGRAM

**A diagram of a smart complaint system

AI-generated content may be incorrect.**

Fig5.2. Block diagram

The **Context Flow Diagram** of the **Smart Complaint Management System** represents the interaction between users (residents, technicians, and admins) and the system. Residents submit complaints, track progress, and make payments through the front-end (React.js), which sends requests to the backend via the server (Express.js). The backend processes these requests, updates the database (MongoDB Atlas), and manages the business logic such as complaint categorization, technician assignment, and payment tracking. Admins oversee the entire system, monitor progress, assign tasks, and generate reports, while technicians receive complaints assigned to them, update their status, and mark them as resolved. The system ensures secure communication and real-time updates between all users.

# CHAPTER 6. DETAILED DESIGN

## USE CASE DIAGRAM

## A diagram of a company AI-generated content may be incorrect.

Fig6.1 Use Case diagram

The Use Case Diagram for the Smart Complaint Management System depicts the key interactions between users and the system. Residents can submit complaints, track statuses, and make payments, while technicians can view and update assigned complaints. Admins have overarching control, including managing complaints, assigning tasks, monitoring technician performance, and generating reports. Each user’s role is represented with specific use cases that highlight their access and responsibilities within the system.

## 

## DATABASE -SCHMEA

## A screen shot of a computer Description automatically generated

A close-up of a document

Description automatically generated

A close-up of a document

Description automatically generated A screen shot of a computer code

Description automatically generated

# CHAPTER 7. IMPLEMENTATION

# 7.1 **PSEUDOCODE**

# 

# 7.11 Login code

# FUNCTION Login ()

# // State Declarations

# DECLARE navigate = **use Navigate (**)

# DECLARE **show Password** STATE = FALSE

# DECLARE **background Index** STATE = 0

# DECLARE data STATE = **{name**: "", email: "", password: "", **confirm Password**: "" }

# DECLARE **user Data** = **use Selector (**state => state)

# DECLARE dispatch = **use Dispatch (**)

# // Background Images for Slideshow

# DECLARE **background Images** = [

# "https://cdn.pixabay.com/photo/2022/10/16/12/23/tram-7524963\_1280.jpg",

# "https://cdn.pixabay.com/photo/2019/12/05/05/50/san-francisco-4674350\_1280.jpg",

# "https://cdn.pixabay.com/photo/2023/04/04/19/48/houses-7900142\_1280.jpg",

# ]

# // Effect for Background Slideshow

# USE\_**EFFECT (**() =>

# SET\_**INTERVAL (**() =>

# SET **background Index (**(**Prev Index**) => (**Prev Index** + 1) MOD **background Images** .length)

# , 5000)

# RETURN CLEAR\_INTERVAL

# , [**background Images. length**])

# // Function to Toggle Password Visibility

# FUNCTION **handle Show Password (**)

# SET **show Password**((**Prev**) => NOT **Prev**)

# // Function to Handle Input Changes

# FUNCTION **handsearching**(event)

# DECLARE name = event.target.name

# DECLARE value = event. **target. Value**

# **Sedate**((**Prev**) => ({

# ...**Prev**,

# [name]: value,

# }))

# // Function to Handle Form Submission

# ASYNC FUNCTION **handle Submit**(event)

# event.PREVENT\_DEFAULT()

# DECLARE **{name**, email, password, **confirm Password}** = data

# IF name AND email AND password AND **confirm Password** THEN

# IF password === **confirm Password** THEN

# TRY

# DECLARE **fetch Data** = AWAIT **FETCH (**

# `${process. **overeat**\_APP\_SERVER\_DOMIN}/**Admi Login**`,

# {

# METHOD: "POST",

# HEADERS: {

# "Content-Type": "application/**Json**",

# },

# BODY: JSON.STRINGIFY(data),

# }

# )

# IF NOT fetch **Data. ok** THEN

# DECLARE **error Data** = AWAIT fetch **Data. JSON (**)

# TOAST.ERROR(error **Data. Message** OR "An error occurred")

# RETURN

# END IF

# DECLARE dataRes = AWAIT **fetchData. JSON**()

# TOAST(**dataries** .message)

# IF userData.user AND dataRes.alert THEN // This condition seems a bit off, consider if it should be `userData.user && dataRes.alert` or just `dataRes.alert` for success.

# DISPATCH(loginRedux(dataRes))

# SET\_TIMEOUT(() =>

# NAVIGATE("/admin/admindashboard")

# , 1000)

# END IF

# CATCH error

# TOAST.ERROR("An error occurred while logging in.")

# END TRY

# ELSE

# TOAST.ERROR("Password and confirm password do not match")

# END IF

# ELSE

# TOAST.ERROR("Please fill in all fields")

# END IF

# // Render Section

# RETURN (

# <DIV className="min-h-screen flex relative">

# {/\* Background Slideshow \*/}

# <DIV

# className="absolute inset-0 bg-cover bg-center bg-fixed transition-opacity duration-1000"

# style={{ backgroundImage: `url(${backgroundImages[backgroundIndex]})` }}

# >

# <DIV className="absolute inset-0 bg-black opacity-50"></DIV>

# </DIV>

# {/\* Left Content \*/}

# <DIV className="w-1/2 flex items-center justify-center relative z-10 animate-slide-down">

# <DIV className="flex flex-col items-center text-center text-white px-6 bg-gray-800 bg-opacity-50 rounded-lg p-8 shadow-xl">

# <H1 className="text-4xl font-bold mb-4 text-white-400 drop-shadow-lg">

# Welcome Back!

# </H1>

# <P className="text-lg mb-4 text-white-300 drop-shadow-md">

# access your administrative tools and enhance

# your overall experience.

# </P>

# <UL className="list-disc list-inside text-left text-white-300">

# <LI> Manage accounts with ease</LI>

# <LI> Track and monitor activity</LI>

# <LI> Customize settings for efficiency</LI>

# </UL>

# </DIV>

# </DIV>

# {/\* Login Form \*/}

# <DIV className="w-1/2 flex items-center justify-center relative z-10 animate-slide-up">

# <DIV className="w-full max-w-sm bg-gray-900 bg-opacity-80 p-6 rounded-lg shadow-2xl">

# <H2 className="text-center text-3xl text-white mb-6 drop-shadow-lg">

# Admin Login

# </H2>

# <FORM onSubmit={handleSubmit} className="flex flex-col">

# <LABEL htmlFor="name" className="text-white font-medium">

# Name

# </LABEL>

# <INPUT

# type="text"

# id="name"

# name="name"

# className="mt-1 mb-2 w-full bg-gray-700 px-3 py-2 rounded text-white focus:outline-none focus:ring-2 focus:ring-blue-500"

# value={data.name}

# onChange={handleOnChange}

# required

# />

# <LABEL htmlFor="email" className="text-white font-medium">

# Email

# </LABEL>

# <INPUT

# type="email"

# id="email"

# name="email"

# className="mt-1 mb-2 w-full bg-gray-700 px-3 py-2 rounded text-white focus:outline-none focus:ring-2 focus:ring-blue-500"

# value={data.email}

# onChange={handleOnChange}

# required

# />

# <LABEL htmlFor="password" className="text-white font-medium">

# Password

# </LABEL>

# <DIV className="flex items-center bg-gray-700 rounded mt-1 mb-2">

# <INPUT

# type={showPassword ? "text" : "password"}

# id="password"

# name="password"

# className="w-full bg-gray-700 border-none outline-none text-white px-3 py-2"

# value={data.password}

# onChange={handleOnChange}

# required

# />

# <SPAN

# className="flex text-xl cursor-pointer text-white p-2"

# onClick={handleShowPassword}

# >

# {showPassword ? <BiShow /> : <BiHide />}

# </SPAN>

# </DIV>

# <LABEL

# htmlFor="confirmPassword"

# className="text-white font-medium"

# >

# Confirm Password

# </LABEL>

# <DIV className="flex items-center bg-gray-700 rounded mt-1 mb-2">

# <INPUT

# type={showPassword ? "text" : "password"}

# id="confirmPassword"

# name="confirmPassword"

# className="w-full bg-gray-700 border-none outline-none text-white px-3 py-2"

# value={data.confirmPassword}

# onChange={handleOnChange}

# required

# />

# <SPAN

# className="flex text-xl cursor-pointer text-white p-2"

# onClick={handleShowPassword}

# >

# {showPassword ? <BiShow /> : <BiHide />}

# </SPAN>

# </DIV>

# <BUTTON className="w-full bg-red-500 hover:bg-red-600 text-white text-xl font-medium text-center py-2 rounded-full mt-4 transform hover:scale-105 transition duration-200">

# Login

# </BUTTON>

# </FORM>

# <P className="text-left text-sm mt-2 text-white">

# Don't have an account?{" "}

# <Link to={"/signup"} className="text-red-500 underline">

# Sign Up

# </Link>

# </P>

# </DIV>

# </DIV>

# {/\* Animation Styles (inline for clarity, but ideally in a CSS file) \*/}

# <STYLE>

# {`

# @keyframes slideDown {

# from {

# transform: translateY(-100%);

# opacity: 0;

# }

# to {

# transform: translateY(0);

# opacity: 1;

# }

# }

# @keyframes slideUp {

# from {

# transform: translateY(100%);

# opacity: 0;

# }

# to {

# transform: translateY(0);

# opacity: 1;

# }

# }

# .animate-slide-down {

# animation: slideDown 2s ease-out;

# }

# .animate-slide-up {

# animation: slideUp 5s ease-out;

# }

# `}

# </STYLE>

# </DIV>

# )

# END FUNCTION

# 7.12 Payments code

# FUNCTION TechnicianPaymentComponent

# // STATE VARIABLES

# INITIALIZE technicianId as empty string

# INITIALIZE paymentDetails as empty list

# INITIALIZE innerBgColor as dark transparent color

# // DEFINE an array of vibrant background colors

# DECLARE colors = [red, green, blue, yellow, orange, indigo, violet]

# // BACKGROUND COLOR CHANGER - every 2 seconds

# USE\_EFFECT:

# SET interval to change innerBgColor to a random color from 'colors'

# RETURN cleanup function to clear interval

# // FUNCTION: Fetch Payment Details from server

# FUNCTION handleFetchPaymentDetails:

# IF technicianId is empty

# SHOW alert: "Please enter a technician ID"

# RETURN

# TRY:

# MAKE GET request to `${server}/api/get-payment-details/{technicianId}`

# PARSE response to JSON

# 

# IF response is successful:

# SET paymentDetails to received data

# AFTER 30 seconds:

# CLEAR paymentDetails

# ELSE:

# SHOW alert with error message

# CLEAR paymentDetails

# CATCH error:

# LOG error

# SHOW alert: "Error fetching payment details"

# // JSX RETURN (UI STRUCTURE)

# RETURN:

# DIV (outer container with background image and full screen height)

# DIV (innerContainer with dynamic background color)

# HEADING: "Technician Payment Module"

# INPUT: for technicianId

# BUTTON: "Fetch Payment Details", onClick = handleFetchPaymentDetails

# IF paymentDetails is not empty:

# DISPLAY payment details in a styled list:

# FOR each payment in paymentDetails:

# SHOW complaintId, finalAmount, paymentStatus, paymentId, timestamp

# SEPARATE each item with a line

# END FUNCTION

# 7.2 SCREEN SHOTS

# A screenshot of a computer AI-generated content may be incorrect. Fig 7.2.1 USER LOGIN: Please log in by providing your name, email, password, confirm password, and captcha verification.

# A screenshot of a computer AI-generated content may be incorrect. Fig 7.2.2 ADMIN LOGIN: Please log in by providing your name, email, password, confirm password.before login first **signup**

# A screenshot of a computer AI-generated content may be incorrect.

# Fig 7.2.3 Users can submit personal or public complaints by providing a title, description, address, and uploading an image.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Fig 7.2.4 Feedback:** Users can provide feedback to the technician based on the quality of the work completed.

## A screenshot of a computer AI-generated content may be incorrect.

## **Fig 7.2.5** Admin dashboard: allows for viewing complaints, accepting them, and either assigning them to a technician or placing them on hold.

## A person standing on a rock with a flashlight AI-generated content may be incorrect.

## **Fig 7.2.6** Recent Activity: **A user recently performed an activity, which was completed successfully.**

A screenshot of a computer

AI-generated content may be incorrect.

Fig 7.2.7 **Personal complaints Details:** are individual issues such as water leakage, electrical faults, or drainage problems. These complaints are assigned to the respective technicians based on the complaint type and urgency.

A screenshot of a computer

AI-generated content may be incorrect.

Fig 7.2.8 **Contact Details:** For any queries or support, feel free to reach out through the contact form.Provide your name, email, and message so we can assist you promptly.

A screenshot of a computer

AI-generated content may be incorrect.

Fig 7.2.9 **Payment Gateway:** You can make payments securely through all major platforms including Card, Net Banking, and UPI. Enjoy a hassle-free transaction experience with multiple payment options integrated.

A screenshot of a computer

AI-generated content may be incorrect.

Fig 7.2.10 **Payment Successful:** Your payment was successful and has been securely processed.  
Thank you! A confirmation has been sent, and your service will be updated shortly.

**Testcases:**

**1.Authentication & User Management**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC1.1 | Enter valid email/password | Redirects to Dashboard | Redirected to Dashboard successfully | Pass |
| TC1.2 | Enter invalid password | Shows error message | Error message displayed: "Incorrect password" | Pass |
| TC1.3 | Attempt to register with existing email | Shows error message | Error message displayed: "Email already exists" | Pass |
| TC1.4 | Reset password with valid email | Sends password reset link | Password reset link sent successfully | Fail |

**2. Complaint Submission – Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC2.1 | Fill in complaint form with valid details | Complaint submitted successfully | Complaint submitted successfully | Pass |
| TC2.2 | Submit empty complaint form | Shows validation error | Validation error shown: "Fields cannot be empty" | Pass |
| TC2.3 | Attach a file larger than the limit | Shows error message | Error message displayed: "File size exceeds limit" | Pass |
| TC2.4 | Submit complaint with invalid email format | Shows validation error | Validation error shown: "Invalid email format" | Pass |

**3. Payment Gateway – Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC3.1 | Select payment option | Redirects to payment gateway | Redirected to payment gateway | Pass |
| TC3.2 | Enter valid payment details | Payment processed successfully | Payment processed successfully | Pass |
| TC3.3 | Enter invalid payment details | Shows error message | Error message displayed: "Invalid payment details" | Pass |
| TC3.4 | Cancel payment during processing | Returns to previous screen | Returned to previous screen successfully | Pass |

**4.Technician Acceptance of Complaints – Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC4.1 | Technician views assigned complaints | Displays list of assigned complaints | List of assigned complaints displayed | Pass |
| TC4.2 | Technician accepts a complaint | Complaint status updated to "Accepted" | Status updated to "Accepted" | Pass |
| TC4.3 | Technician rejects a complaint | Complaint status updated to "Rejected" | Status updated to "Rejected" | Pass |
| TC4.4 | Technician completes a complaint | Complaint status updated to "Resolved" | Status updated to "Resolved" | Pass |

**5.User Feedback – Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC5.1 | Provide feedback on complaint resolution | Feedback submitted successfully | Feedback submitted successfully | Pass |
| TC5.2 | Submit feedback without text | Shows validation error | Validation error shown: "Feedback cannot be empty" | Pass |
| TC5.3 | Rate the complaint handling | Rating submitted successfully | Rating submitted successfully | Pass |

**6.Performance & Security – Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC6.1 | Submit 100 complaints in quick succession | No app crash/slowdown | App remained responsive with no lag | Pass |
| TC6.2 | Attempt to access complaints without authentication | Access denied | Access denied as expected | Pass |
| TC6.3 | Test system response time for complaint submission | Response time < 2 seconds | Response time was 1.5 seconds | Pass |

**7.Edge Cases – Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC7.1 | Submit complaint with special characters | Complaint submitted successfully | Complaint submitted successfully | Pass |
| TC7.2 | Submit extremely long complaint | Shows error message | Error message displayed: "Complaint too long" | Pass |
| TC7.3 | Attempt to submit complaint while offline | Shows error message | Error message displayed: "No internet connection" | Pass |

**8.Database Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TC8.1 | Insert a new complaint record with valid data | Record inserted successfully | Record inserted successfully | Pass |
| TC8.2 | Retrieve a complaint record by ID | Returns the correct complaint details | Correct complaint details returned | Pass |
| TC8.3 | Update the status of a complaint record | Status updated successfully | Status updated successfully | Pass |

**MANUAL TESTING FOR PAYMENT GATEWAY – SMART COMPLAINT SYSTEM**

| **Test Case ID** | | **Test Case Description** | | |
| --- | --- | --- | --- | --- |
| TC1 | | Test the Payment Gateway Functionality of Smart Complaint System | | |
| **Created By** | **Reviewed By** | | **Version** |
| NOOTHAN KP | NOOTHAN KP | | 1.0 |

| **QA Tester’s Log** | | | |
| --- | --- | --- | --- |
| Review the Payment Gateway Functionality in the Smart Complaint System for online payments. | | | |
|  | | | |
| **Tester’s Name** | **Date Tested** | **Test Case (Pass/Fail/Not)** |
| NOOTHAN KP | May 5, 2025 | Pass |

**Prerequisites**

| **S #** | **Prerequisites** |
| --- | --- |
| 1 | Access to Chrome Browser |
| 2 | Valid Complaint ID or Complaint Payment Link |
| 3 | Active Internet Connection |

**Test Data**

| **S #** | **Test Data** |
| --- | --- |
| 1 | Complaint ID = SMART\_PERS\_001 |
| 2 | User Email = user@example.com |
| 3 | Payment Amount = 250 |
| 4 | Payment Method = Debit Card |
| 5 | Valid Card Details = 4111 1111 1111 1111 |
| 6 | Invalid Card Details = 1234 5678 9012 3456 |
| 7 | Payment Gateway = razorpay\_gateway\_01 |
| 8 | Cancel Payment Option Enabled |

**Test Scenario**

Verify payment process works correctly for complaint payments, handles both valid and invalid payment attempts, and supports transaction cancellation.

**CONCLUSIONS**

The **Smart Complaint System** is a modern, digital solution designed to simplify and improve the way complaints are registered, managed, and resolved in both public and private domains. This system plays a vital role in strengthening communication between the public and governing authorities or organizations by providing a structured and transparent platform for reporting and resolving issues.

One of the standout features of this system is its ability to distinguish between **personal** and **public** complaints. **Personal complaints** relate to individual issues such as broken water pipelines, electrical faults, drainage problems, or household maintenance requests. These are submitted directly by users and assigned to appropriate **technicians** based on their area of expertise. This targeted allocation ensures that issues are addressed by skilled professionals, improving resolution speed and service quality.

In contrast, **public complaints** are concerns that affect entire communities or neighbourhoods—such as malfunctioning streetlights, potholes, garbage collection delays, or damaged public infrastructure. These complaints are made visible to other residents, who can **vote** to indicate the urgency or importance of the issue. This voting mechanism creates a **community-driven prioritization** process, allowing decision-makers to take action based on collective public interest. It ensures that resources are allocated effectively, and the most pressing issues are addressed first.

To make the process even more user-friendly and secure, the system includes **Razor pay Payment Gateway integration**. This enables users to make necessary payments for services, technician visits, or complaint-related fees directly through the platform. The use of a trusted payment gateway like Razor pay adds an additional layer of **security, speed, and convenience** to the system. Users receive instant payment confirmations, and transactions are fully traceable, promoting transparency and accountability.

Another critical component of the system is its **technician management module**. Technicians are registered with verified credentials and login access to view, update, and manage the complaints assigned to them. This system tracks technician performance, response time, and task completion, encouraging **accountability and continuous improvement**. By matching the right technician to the right task, the platform ensures high-quality service and timely resolution of issues.

Moreover, the platform can be accessed through both desktop and mobile interfaces, making it **highly accessible** to users from different backgrounds. Its intuitive design and real-time complaint tracking allow users to stay updated about the progress of their submissions. Notifications, status updates, and estimated resolution times help in building trust and user satisfaction.

**FUTURE ENHANCEMENTS**

As technology continues to evolve, there are several exciting future enhancements planned for the Smart Complaint System to make it more powerful, accessible, and user-friendly. These upgrades aim to improve user experience, ensure faster responses, and create a smarter, more connected environment for complaint management.

One of the major future goals is to develop a dedicated mobile application for both Android and iOS platforms. While the current web-based system serves its purpose, a mobile app would significantly enhance usability and reach. Users will be able to register complaints, upload photos or videos, track status, and communicate with technicians directly from their smartphones. The app will provide a smooth, responsive interface, making it easier for users of all ages and backgrounds to submit and follow up on their complaints anytime, anywhere.

Another important enhancement is the introduction of a real-time notification system. This feature will allow users to receive instant alerts via SMS, email, or in-app messages when there are updates to their complaint status—such as technician assignment, complaint resolved, or payment confirmation. Notifications will keep users informed throughout the process, reducing the need for manual follow-ups and increasing transparency and trust.

A cutting-edge feature in the pipeline is the integration of voice assistant technology. In this enhancement, users will be able to register complaints using voice commands through popular virtual assistants like Google Assistant, Alexa, or an in-app AI voice bot. This will make the system accessible to users with limited literacy or those who prefer voice input over typing. The voice assistant can also provide responses such as complaint status, expected resolution time, and technician updates—making the entire process more natural and user-centric.

Additionally, we plan to include multi-language support in both the web and mobile versions, enabling users to use the system in their regional languages. This will broaden the user base and ensure inclusivity, especially in rural or semi-urban areas.

To support local governance and improve decision-making, the future version may also include data analytics dashboards for administrators. These dashboards will present complaint trends, technician performance, area-wise issues, and user feedback, helping the authorities take preventive and informed action.