A UNIFIED APPROACH FOR SMART CITY SERVICE MANAGEMENT USING BLOCKCHAIN

A PROJECT REPORT SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI &ML)

By

Batch – B17

M. Yasaswini (21JG1A4229)

B. Pujitha (21JG1A4205)

G. Sireesha (21JG1A4216)

P. Bhavitha(22JG5A4204)

Under the esteemed guidance of

Dr. P. Muralidhara Rao

Assistant Professor

Department of CSE, GVPCEW

Department of Computer Science and Engineering (AI&ML)

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation (NBA) for B. Tech CSE, ECE, IT- valid from 2019-22 and 2022-25]

[Accredited by National Assessment and Accreditation Council (NAAC) with A Grade- Valid from 2022-27]

Kommadi, Madhurawada, Visakhapatnam – 53004

2021 – 2025

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CERTIFICATE

This is to certify that the project report titled **"A UNIFIED APPROACH FOR SMART CITY SERVICE MANAGEMENT USING BLOCKCHAIN"** is a bonafide work of following IV B.Tech. II Sem. students in the Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2024-25, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology of this university.

M. Yasaswini (21JG1A4229)

G. Sireesha (21JG1A4216)

B. Pujitha (21JG1A4205)

P. Bhavitha(22JG5A4204)

Signature of the Guide

Dr. P. Muralidhara Rao

Assistant Professor

Dept. of CSE (AI&ML)

Signature of the HOD

Dr. Dwiti Krishna Bebarta

Professor,

Dept. of CSE(AI&ML)

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to **Dr. P. Muralidhara Rao**, Assistant Professor for encouragement all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We would like to take this opportunity to extend our gratitude to Project Coordinator, **Dr. K. Purushottam Naidu**, Associate Professor of Computer Science and Engineering - Artificial Intelligence & Machine Learning for making us to follow a defined path and for advising us to get better improvements in the project.

We express our deep sense of gratitude and thanks to **Dr. Dwiti Krishna Bebarta**, Professor and Head of the Department of Computer Science and Engineering - Artificial Intelligence & Machine Learning for his guidance and for expressing her valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We would like to take this opportunity to express our profound sense of gratitude to **Dr. R. K. Goswami,** Principal and **Dr. G. Sudheer,** Vice Principal for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We are also thankful to both teaching and non-teaching faculty of the Department of Computer Science and Engineering for giving valuable suggestions from our project.

M. Yasaswini (21JG1A4229)

B. Pujitha (21JG1A4205)

G. Sireesha(21JG1A4216)

P.Bhavitha (22JG5A4204)

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

ABSTRACT

The Smart City Portal is an advanced blockchain-integrated platform designed to revolutionize urban infrastructure management and citizen services. As cities grow, the need for secure, transparent, and efficient service delivery becomes crucial. This project leverages Ethereum blockchain technology, smart contracts, and modern web technologies to create a decentralized and tamper- proof system for managing various urban services. By implementing a decentralized architecture, the system ensures data integrity and eliminates reliance on a single central authority. All transactions are securely recorded on the blockchain, providing an immutable and transparent ledger that enhances trust among citizens and authorities. The modular design of the platform allows for the seamless integration of diverse urban services such as parking management, EV charging stations, digital payments, healthcare services, and toll collection. One of the key features of the system is its real-time service tracking and automated digital transactions, ensuring high efficiency and reduced operational costs. With smart contracts, transactions become self- executing and enforceable, eliminating intermediaries and reducing fraudulent activities. The use of Web3 technologies further enables a user-friendly interface for seamless citizen interaction with the platform. The Smart City Portal prioritizes security, scalability, and reliability through end-to-end encryption, role-based access control, and smart contract auditing. Additionally, integration with IPFS (InterPlanetary File System) provides decentralized data storage, ensuring availability even in case of network failures. By adopting Ethereum-based blockchain solutions, the project paves the way for a secure, scalable, and future-proof smart city infrastructure, fostering efficient governance, digital inclusion, and sustainable urban development.

Keywords Smart City, Blockchain, Ethereum, Smart Contracts, Urban Infrastructure, Healthcare Management, Digital Payments, Web3.

LIST OF FIGURES

LIST OF TABLES

LIST OF SCREENS

SYMBOLS & ABBREVATIONS

IPFS	InterPlanetary File System
EV	Electric Vehicle
Web3	World Wide Web - 3 (Decentralized Web)
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
JS	JavaScript
UML	Unified Modeling Language

1. INTRODUCTION

1.1 MOTIVATION

In today's rapidly urbanizing world, cities are facing unprecedented challenges in managing and delivering a wide range of public services efficiently, securely, and transparently. The increasing complexity of urban infrastructure demands the adoption of innovative technologies that can streamline operations, reduce manual intervention, and improve citizen satisfaction. Traditional systems often operate in silos, leading to inefficiencies, data fragmentation, and a lack of transparency. The motivation behind the Smart City Portal stems from the urgent need to address these limitations through a unified, decentralized platform. With the rise of electric vehicles, there is a growing necessity for accessible and reliable charging infrastructure integrated within city ecosystems. Similarly, the expansion of digital healthcare systems calls for secure and seamless management of patient data and medical services. The inefficiencies in toll collection and traffic monitoring further underscore the need for automated systems that can operate in real-time with minimal human intervention. Furthermore, public trust in digital transactions has been challenged by security concerns and lack of traceability. Blockchain technology, with its decentralized, tamper-proof, and transparent nature, presents a powerful solution to these issues. By leveraging Ethereum blockchain, smart contracts, and Web3 technologies, the Smart City Portal aims to create a robust, scalable, and citizen-centric platform that not only simplifies urban service access but also enhances accountability, reduces operational costs, and fosters digital inclusion. This project is driven by the vision to build smarter, safer, and more sustainable cities that are equipped to meet the demands of the future.

1.2 PROBLEM DEFINITION

Modern urban environments often suffer from a fragmented approach to service delivery, where various public services operate in isolation without any centralized integration. This disjointed system leads to inefficiencies in service booking and management, resulting in delays and inconvenience for citizens. Traditional service booking methods are largely manual, time-consuming, and prone to errors, making them unsuitable for the fast-paced demands of modern cities. Additionally, the lack of transparency in public service transactions fosters mistrust among citizens and increases the risk of corruption and data manipulation. Digital payment

systems, while more convenient, often face security vulnerabilities that threaten user data and transaction integrity. In the healthcare sector, the management of patient records and medical services is frequently inefficient and uncoordinated, impacting the quality and timeliness of care. Moreover, traffic and toll management systems are complex and lack automation, leading to congestion, revenue leakage, and poor user experience. These challenges highlight the need for a comprehensive, secure, and integrated solution to transform urban service delivery.

1.3 OBJECTIVE OF THE PROJECT

- To implement a blockchain-based integrated urban service platform for streamlined and unified city management.
- To develop secure smart contracts for automated and reliable service execution with minimal human intervention.
- To create user-friendly interfaces for seamless access and management of urban services by citizens and authorities.
- To establish efficient payment systems using blockchain technology to ensure secure, transparent, and fast transactions.
- To enable real-time monitoring and management of city services for improved responsiveness and operational efficiency.
- To provide transparent and immutable transaction records that enhance trust, accountability, and data integrity.

1.4 ADVANTAGES OF THE PROJECT

- Blockchain Security Ensures secure and tamper-proof.
- Integrated Services Single platform for multiple urban.
- Automated Management Smart contracts for automatic service execution.
- Transparent Operations All transactions are traceable and verifiable.
- Real-time Monitoring Live tracking of service usage and availability.
- Digital Payments Secure and efficient payment processing.
- Data Analytics Comprehensive reporting and analysis capabilities.

1.5 LIMITATIONS OF THE PROJECT

- Blockchain Scalability Limited by Ethereum network capacity.
- Initial Setup Complexity Requires significant infrastructure setup.
- User Adoption Learning curve for blockchain technology.
- Network Dependency Relies on stable internet connectivity.
- Gas Fees Transaction costs on Ethereum network.
- Technical Requirements Needs specific hardware/software setup.

1.6 ORGANIZATION OF PROJECT

Concise overview of rest of the documentation work is explained below

Chapter 1 Introduction

This chapter explains the motivation, domain, and problem statement of the project.

Chapter 2 Literature Survey

This chapter gives an overview of existing systems and ideas related to the project.

Chapter 3 Analysis

This chapter includes the system analysis, functional and non-functional requirements.

Chapter 4 Design

This chapter presents the design of the project using diagrams and models.

Chapter 5 Implementation

This chapter explains how the project was developed with steps and output screenshots.

Chapter 6 Testing and Validation

This chapter describes the testing process and results to check the correctness of the project.

Chapter 7 Conclusion

This chapter provides a summary of the project and its outcomes.

Chapter 8 References

This chapter lists the papers and materials referred to during the project.

2. LITERATURE SURVEY

2.1 INTRODUCTION

Modern cities face significant challenges in managing and integrating various urban services efficiently while ensuring transparency and security. The Smart City Portal addresses these challenges by providing a unified platform that leverages blockchain technology to streamline urban services and enhance citizen experience.

In [1] Eiman Al Nuaimi proposed “Applications of Big Data to Smart Cities”. This paper reviews the role of big data in smart city services such as healthcare, transportation, and energy management. It highlights both the opportunities and challenges associated with big data analytics and cloud computing in urban development. The study shows that big data can enhance sustainability, resilience, and quality of life. However, it also points out unresolved issues such as governance, resource management, and the absence of universal definitions for smart cities and big data. Dataset used is a combination of data from various smart city services.

In [2] Dr. Miguel de Simón-Martín proposed “Edge AI and Blockchain for Smart Sustainable Cities Promise and Potential”. This paper examines how edge AI and blockchain technologies can transform urban infrastructure, particularly in smart mobility, energy management, and environmental monitoring. It discusses various applications, including secure communication and traffic monitoring, while addressing key challenges such as data privacy, high energy consumption, and lack of standardization. Dataset used includes Smart City Sensor Data, Blockchain Transaction Data, AI Model Training Data, Urban Sustainability Data, and Environmental Monitoring Data.

In [3] Sudeep Tanwar proposed “Machine Learning Adoption in Blockchain- Based Smart Applications The Challenges, and a Way Forward”. The paper explores the integration of ML models such as SVM, CNN, and LSTM within blockchain-based smart applications, including smart grids and UAVs. It highlights the improvements in data security, efficiency, and resilience against cyber threats. However, it also acknowledges challenges such as the difficulty of integrating with existing legacy systems. Dataset used includes data from various smart applications such as UAVs and Smart Grids.

In [4] R. Wolniak proposed “Artificial Intelligence in Smart Cities Applications, Barriers, and Future Directions A Review”. This paper provides a comprehensive review of AI applications in six dimensions of smart cities smart mobility, smart environment, smart governance, smart living, smart economy, and smart people. It identifies key challenges such as ethical concerns, privacy risks, and regulatory barriers. Additionally, it explores future research directions like predictive policing, real-time monitoring, and sustainable urban planning. However, the study relies on literature-based analysis rather than experimental validation. Dataset used consists of simulated data from IoT devices, including threshold values for fire detection in smart homes.

In [5] S. A. Khan et al. proposed “Blockchain and Smart Cities for Inclusive and Sustainable Communities”. This paper conducts a systematic literature review and bibliometric analysis to examine blockchain applications in smart cities. It introduces a conceptual framework for blockchain adoption to enhance transparency, security, and efficiency in urban governance. However, the paper lacks practical or experimental validation of the proposed framework. Dataset used includes real-time data from IoT devices, social media, and public safety data sources.

In [6] Zhiqi Qian proposed “The Integration of Blockchain and Artificial Intelligence for a Smart City”. This paper reviews security issues in blockchain- based smart city systems and discusses AI-driven solutions for enhancing data integrity, privacy, and scalability. It proposes strategies for blockchain-AI integration and explores future directions for sustainable smart city technologies. However, the research is largely theoretical and lacks experimental validation. Dataset used includes open data from public platforms, satellite images, social media, and government repositories.

Table 1 : Table for Literature Survey

S.No.	Title and Author	Published Journal and year	Dataset Used	Approach	Outcome
1.	Applications of Big Data to Smart Cities **Author** Eiman Al Nuaimi	2023, Journal of Internet Services and Applications	Big-Data from various smart-city services like healthcare, transportation etc.	Reviews big data applications in smart cities, highlighting challenges and opportunity, leveraging cloud computing and big data analytics.	Identified opportunities for big data in improving sustainability, resilience, and quality of life but highlighted unresolved issues like governance and resource management.
2.	Edge AI and Blockchain for Smart Sustainable Cities Promise and Potential **Author** Dr. Miguel de Simón-Martín	2022, MDPI	Smart City Sensor Data, Blockchain Transaction Data, AI Model Training Data, Urban Sustainability Data, Environmental Monitoring Data.	Analyses the transformative potential of edge AI and blockchain technologies in smart mobility and smart energy sectors within smart cities.	Highlights applications such as traffic monitoring, energy management, and secure communications, emphasizing the benefits of integrating these technologies.
3.	Machine Learning Adoption in Blockchain-Based Smart Applications The Challenges, and a Way Forward **Author** Sudeep Tanwar	2020, IEEE Access	Data from various smart applications (e.g., UAVs, Smart Grids)	Integrated ML techniques (e.g., SVM, CNN, LSTM) with blockchain enhancing.	Improved data security, resilience against attacks, and efficiency in smart applications

Table 2 : Table for Literature Survey

S.No.	Title and Author	Published Journal and year	Dataset Used	Approach	Outcome
4.	Artificial Intelligence in Smart Cities—Applications, Barriers, and Future Directions A Review **Author** R.Wolniak	2024, Smart Cities (MDPI Journal)	Simulated data with IoT devices, including threshold values for fire detection in smart homes.	Narrative literature review focusing on six dimensions of smart cities smart mobility, smart environment, smart governance, smart living, smart economy, and smart people.	Identified in six smart city areas, highlighted challenges like privacy, ethics, and regulations, and proposed future directions, including predictive policing, personalized services, sustainable urban planning, and real-time monitoring.
5.	Blockchain and Smart Cities for Inclusive and Sustainable Communities **Author** S. A. Khan et al.	2024, Sustainability	Real-time data from IoT devices, social media, and public safety data sources.	Systematic literature review and bibliometric analysis.	Proposed a conceptual framework for blockchain adoption in smart cities
6.	The Integration of Blockchain and Artificial Intelligence for a Smart City **Author** Zhiqi Qian	2021, Academic Journal of Computing & Information Science	Open Data from public platforms, satellite images, social media, government data, and other repositories.	Comprehensive literature review of security issues in blockchain systems for smart cities. Discusses solutions for enhancing security and proposes strategies for integrating blockchain and AI.	Identified security challenges, such as data integrity, privacy, and scalability, and proposed strategies for blockchain-AI integration. Explored future directions for sustainable smart city technologies.

2.2 EXISTING SYSTEM

The Existing Smart City System operates largely on manual and paper-based processes, creating several inefficiencies for both citizens and city staff. In this system, citizens are required to physically visit service centers for various needs, such as submitting forms, seeking support, or making payments. These manual service centers often experience long queues and delays due to the sheer volume of physical visits and the lack of automation in handling requests.Citizens typically interact with the system through one of three main methods filling out paper forms, calling phone support, or visiting physical payment counters. For any financial transactions, citizens must visit physical payment counters. These counters process payments manually, requiring staff intervention for verification and recording.Moreover, relying on paper-based records reduces the system's efficiency and responsiveness, resulting in delayed services. These limitations highlight the need for a modernized, digital smart city system that can offer faster, more accessible, and more reliable services to citizens.

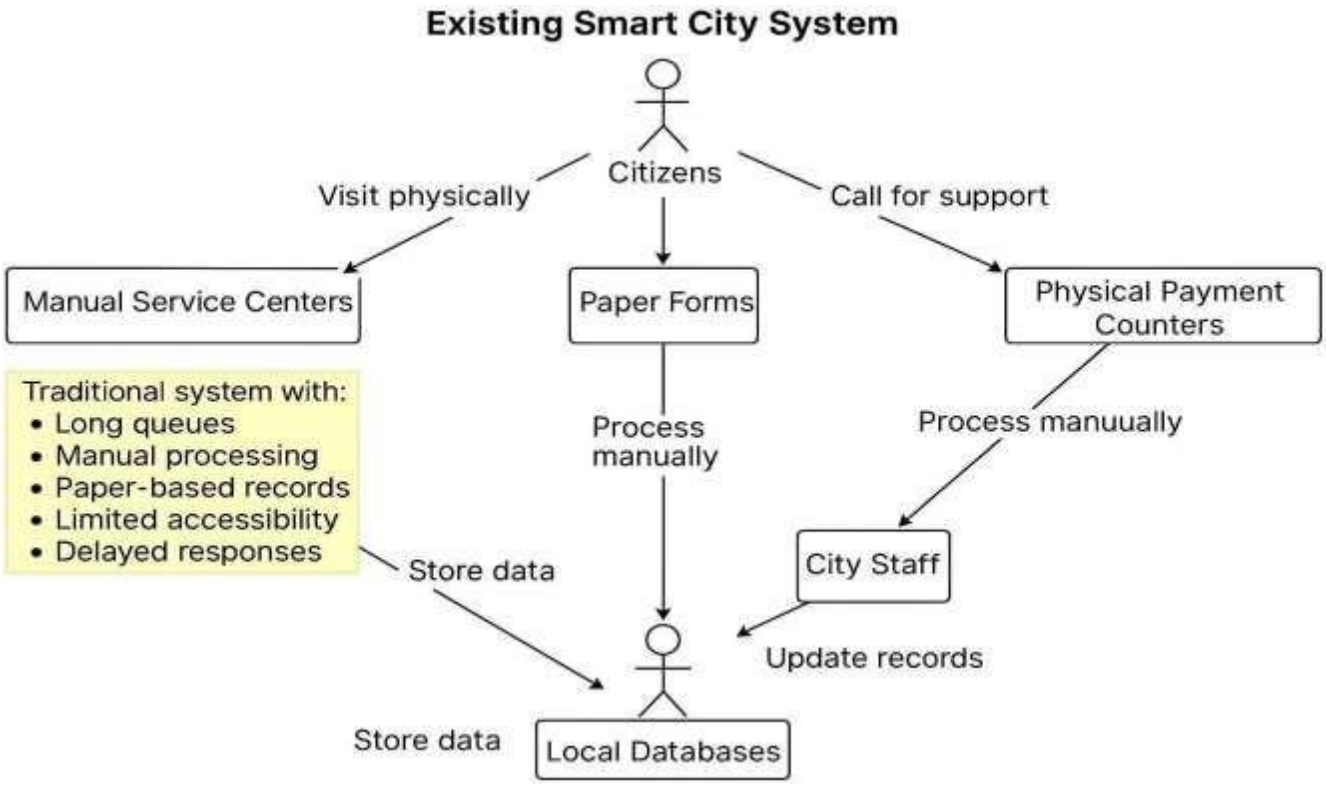


Fig. 2.3 Existing System

2.3 DISADVANTAGES OF EXISTING SYSTEM

- Services are not easily accessible to people living in remote areas or those who are unable to travel to service centers. There is also a lack of 24/7 access to services, making it inconvenient for many citizens.
- Due to the lack of automation and real-time processing, responses to citizen requests are often delayed. This slows down the overall administrative process and affects citizen satisfaction
- Maintaining manual infrastructure, physical counters, and handling paper documentation require significant manpower and recurring costs, making the system less cost-effective in the long run.

2.4 PROPOSED SYSTEM

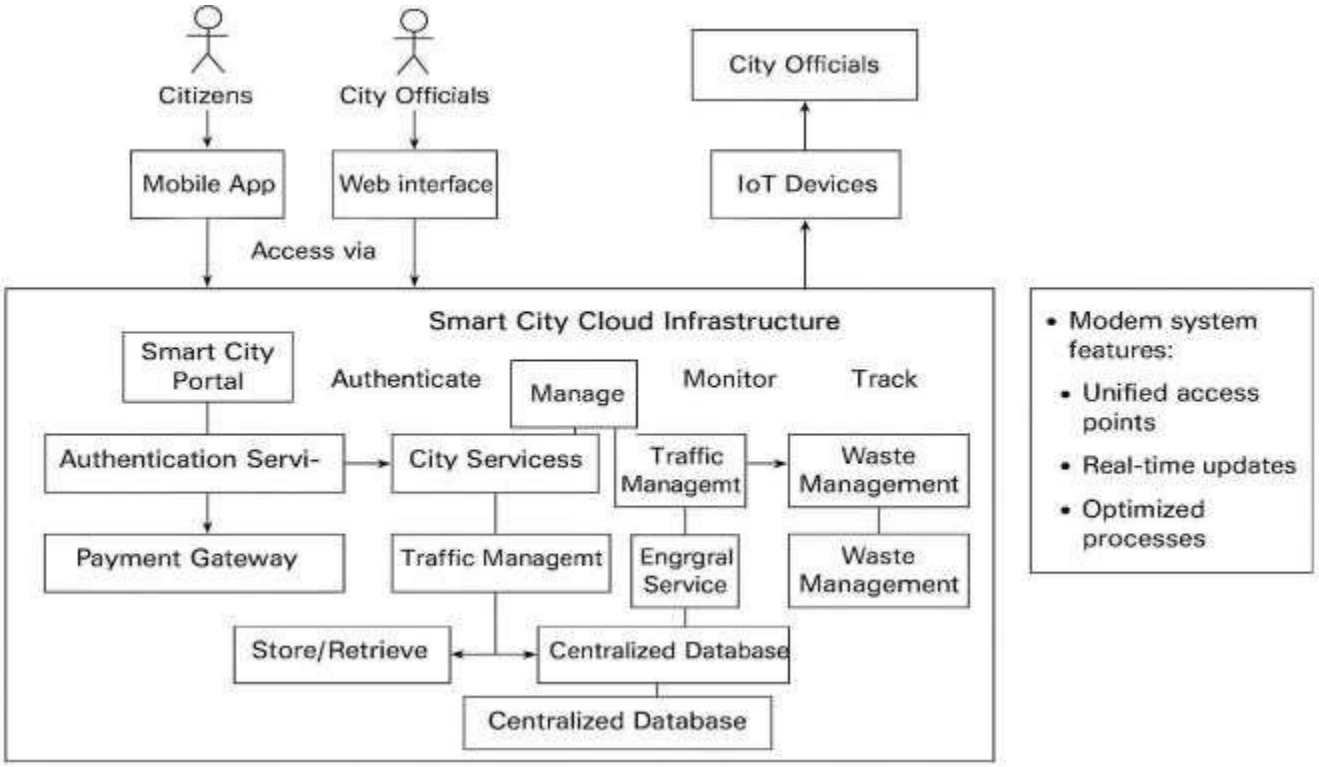


Fig 2.4 Proposed System

Proposed System is shown in Fig 2.4 . The Proposed Smart City System is a modern, cloud-based solution designed to overcome the limitations of traditional manual systems. It provides a digital platform where both citizens and city officials can access services easily through multiple channels. Citizens interact with the system using mobile apps and web interfaces, while city officials can manage services via web portals or IoT devices. These devices and applications communicate with the central cloud infrastructure using API calls, ensuring smooth and real-time data exchange.

At the heart of the system is the Smart City Cloud Infrastructure, which hosts various Core Services and City Services. The Smart City Portal acts as the main entry point, handling user authentication and payment processing through integrated services like the Authentication Service and Payment Gateway. These components ensure secure access and enable seamless digital transactions, eliminating the need for physical presence.

The City Services layer includes essential urban services such as Traffic Management, Healthcare Services, Waste Management, and Energy Management. These services are managed and monitored automatically with real-time updates and integration with IoT devices. For example, traffic can be managed more efficiently, healthcare services can be updated digitally, waste can be tracked and collected on schedule, and energy consumption can be monitored for better efficiency.

All data generated by users and services is stored, retrieved, and updated in a Centralized Database, ensuring accurate and unified record-keeping. This database supports real-time processing, instant access to services, and automated workflows that reduce human error and speed up response times.

The system is built with modern features such as 24/7 accessibility, real-time updates, digital payments, automated processing, mobile access. These enhancements make city services more efficient, transparent, and user-friendly, offering a smart and sustainable approach to urban governance.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- By integrating IPFS (Inter Planetary File System), the system ensures secure, decentralized storage of documents and digital records, protecting data integrity and enabling faster access.

- MongoDB, a NoSQL database, is used to handle large volumes of unstructured and structured data efficiently, offering scalability, fast queries, and high performance.
- The use of a centralized digital dashboard for administrators helps in monitoring, managing, and responding to city service requests more effectively
- Real-time response handling allows citizens to receive instant updates on service requests and payment status, improving transparency and user experience.
- Overall, the system improves efficiency, scalability, and reliability, ensuring smoother city service delivery and higher citizen satisfaction.

2.6 CONCLUSION

The Smart City Portal is a comprehensive platform designed to streamline urban services like traffic, healthcare, waste, and energy management. It uses MongoDB for backend storage and IPFS for decentralized file handling, ensuring both scalability and security. The system avoids IoT integration, relying solely on backend technologies. Citizens can register, log in, access services, and make payments easily through a centralized dashboard. Authorities can monitor data, manage services, and interact with smart contracts. The portal includes secure modules for authentication and encryption, complying with GDPR guidelines. Notifications and updates are handled via integrated email services. Each service is modular and connected through API gateways for smooth data flow. The architecture supports real-time updates and efficient processing. Overall, the portal enhances city management by offering a secure, accessible, and experience.

3. REQUIREMENT ANALYSIS

3.1 INTRODUCTION

Requirement analysis defines the essential aspects needed for the system to function effectively. It includes functional requirements like user management, parking management, EV charging, healthcare services, and FAS Tag services. Hardware and software requirements specify the necessary infrastructure, including servers, blockchain nodes, and development frameworks. Non- functional requirements ensure performance, security, scalability, and reliability for seamless operation. This analysis helps build a robust, efficient, and user- friendly system. friendly system.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

The software requirement specification (SRS) for the project titled “A Unified Approach for Smart City Service Management Using Blockchain” outlines the essential functional and non-functional requirements needed to develop a robust, secure, and scalable service management platform. The system aims to enable citizens, administrators, and service providers to interact efficiently within a unified framework that manages various city services such as waste management, electricity, water supply, and public transportation. Key functionalities include user authentication, service request tracking, real-time monitoring, and automated service processing using smart contracts. Blockchain integration plays a critical role in ensuring transparency, tamper-proof data storage, and trustworthy service records, enabling stakeholders to verify service transactions independently.

The platform should be designed with a modular architecture to facilitate future enhancements and the addition of new services.. Role-based dashboards should offer customized views and functionalities for citizens, administrators, and service staff. The system should log every activity to the blockchain to ensure auditability. Lastly, scalability testing and performance optimization must be a part of the development lifecycle to prepare for large-scale real-world deployment.

3.2.1. User Requirements

Table 3 : User Requirements

User Requirement	Description
User Authentication and Access Control	Users (citizens, administrators, service providers) should be able to securely log in using unique credentials. Role-based access control must be implemented to ensure appropriate access levels.
Service Request and Management	Citizens should be able to raise, view, and track service requests (e.g., water, electricity, waste management) through an intuitive interface.
Blockchain Integration for Transparency	All transactions and service logs must be recorded on a blockchain ledger to ensure transparency, immutability, and traceability.
Smart Contract Execution	Smart contracts should automate service-related tasks (e.g., assigning service providers, initiating payments) based on predefined rules.
Decentralized Data Storage	Store service data in a decentralized way using blockchain to ensure data integrity and avoid single points of failure.
Multi-Service Module Support	Support integration and management of multiple smart city services (e.g., transportation, waste, energy) within the same platform.
Real-time Monitoring and Alerts	Provide real-time dashboards for administrators to monitor service performance and alerts for any unusual activities or service failures.
Analytics and Reporting	Generate insights and reports related to service usage, performance, user satisfaction, and blockchain transaction logs.
Compliance and Data Security	Ensure system follows relevant data protection regulations and incorporates strong encryption and identity protection mechanisms.
Interoperability with Legacy Systems	Enable integration with existing city service systems and third-party providers via secure APIs or middleware.
Scalability and Performance	System should support a growing number of users and services with minimal latency and high performance.
User Training and Support	Provide help documentation, training modules, and technical support for different types of users to ensure smooth onboarding and usage.

3.2.2 Software Requirements

Table 3: Blockchain Software Requirements

Requirement Specification	Software Requirement
Language	Python, JavaScript
Operating System	Windows 11 / Linux
Interpreter/Runtime	Python 3.x, Node.js
Libraries/Frameworks	Flask, Web3.py, PyMongo,
Blockchain Platform	Ethereum (Ganache)
Smart Contract Language	Solidity
IDE	Visual Studio Code, Jupyter
Database	MongoDB

3.2.3.Hardware Requirements

Table 4 : Hardware Requirements

Requirement Specification	Hardware Requirement
System Type	x64-based PC
Processor	11th Gen Intel® Core™ i5-1135G7 CPU or higher
RAM	Minimum 8 GB
Hard Disk	1 TB HDD / 512 GB SSD (Recommended for faster access)
Graphics	Integrated Intel Iris® Xe Graphics or equivalent
Display	14” or larger, Full HD resolution
Network	Stable internet connection with minimum 10 Mbps speed

3.2.3. NON-FUNCTIONAL REQUIREMENTS

- **Performance** The Smart City Portal ensures a response time of less than 3 seconds and supports over 1000 concurrent users. It maintains 99.9% uptime, providing reliable and uninterrupted access to services.
- **Scalability** The system is designed with scalability in mind, featuring horizontal scaling capability to handle increasing demand efficiently. Its modular architecture allows easy integration of new services, while load balancing ensures optimal performance under high user loads.
- **Reliability** The platform ensures high reliability through regular data backup and recovery mechanisms to prevent data loss. Robust error handling systems maintain smooth operation, and transaction consistency guarantees accurate and dependable service execution across all modules..
- **Security** The system prioritizes security with end-to-end encryption to protect data during transmission. It employs secure authentication mechanisms to prevent unauthorized acces and conducts regular smart contract auditing to identify and eliminate potential vulnerabilities

3.3 CONTENT DIAGRAM OF PROJECT

The proposed framework for integrating blockchain in smart city applications is designed to ensure transparency, decentralization, and security across various urban services. The system begins with a frontend login interface, which allows both users and authorities to access the platform. Authentication is performed through Web3 login integrated with MetaMask, ensuring secure, decentralized identity management. Once authenticated, users can access a range of smart city service modules. These include traffic services, waste management, energy monitoring, and healthcare services. These services are powered by blockchain technology to enhance data integrity, accessibility, and reliability across departments.

To manage and control access to these services, the system utilizes smart contracts developed using Solidity and Truffle Ganache. These contracts are responsible for role-based access control, governance mechanisms, on-chain data storage, and event logging. These smart contracts run on a local blockchain network using Ganache, which simulates Ethereum block chain behavior for development and testing purposes.

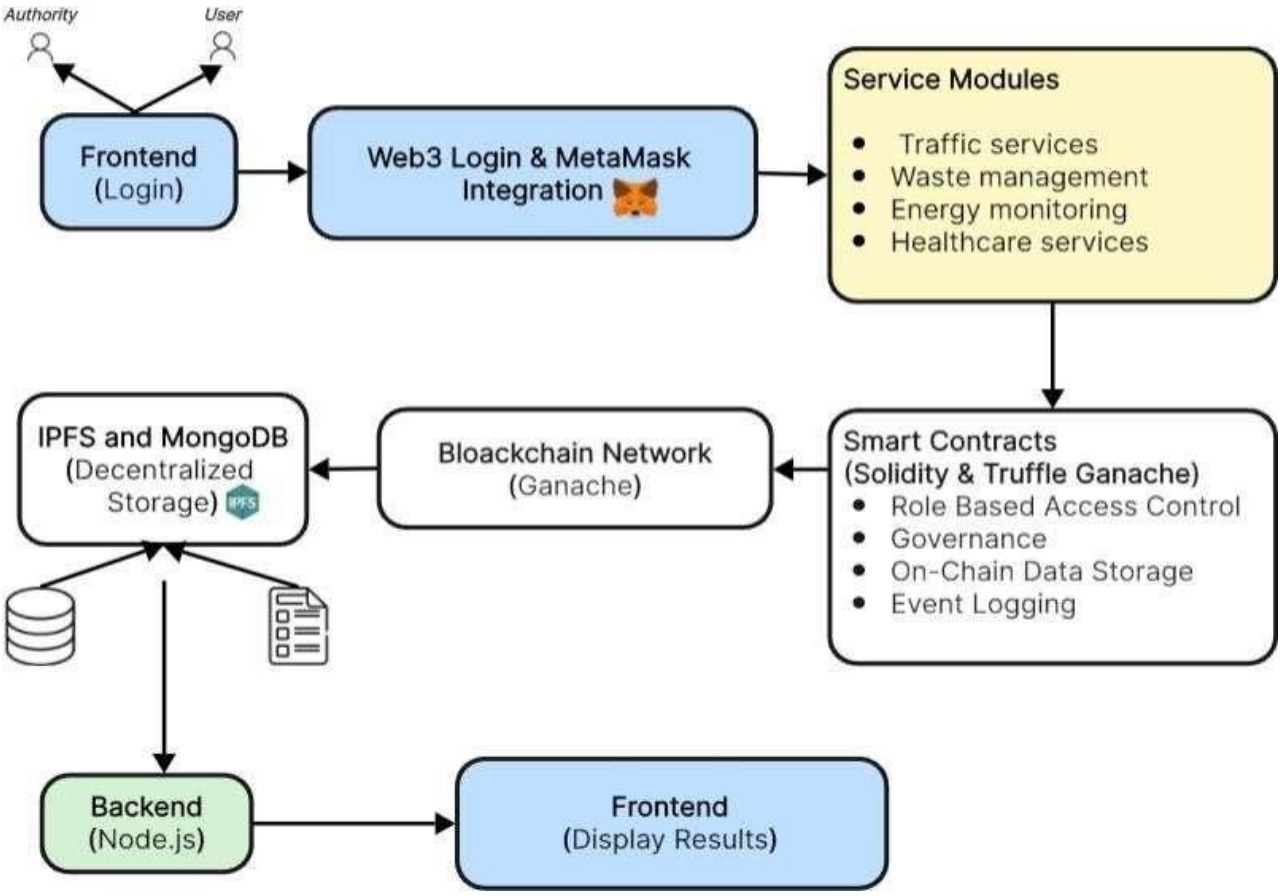


Fig 3.3. Content Diagram Of project

The data generated through service modules and smart contracts is stored using both IPFS (InterPlanetary File System) and MongoDB. IPFS provides decentralized file storage, while MongoDB handles structured off-chain data. This combination ensures efficient, scalable, and secure storage management.

On the backend, the system uses Node.js to handle server-side logic, manage communication between the blockchain network, storage systems, and the user interface. Finally, the frontend display module presents the results and service responses to the end-users in a user-friendly and interactive format.

4. DESIGN

4.1 INTRODUCTION

Unified Modeling language (UML) is used to represent the software in a graphical format, that is it provides a standard way to visualize how a system is designed. Good UML design is followed for a better and precise software output. The UML provides different constructs for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML diagrams are used to better understand the system, to maintain the document information about the system and emphasizes on the roles, actors, actions etc.

The UML diagrams considered are

- Use-case Diagram
- Class Diagram
- Sequence Diagram

RELATIONSHIPS

The relationships among different entities in the following diagram are

Association

This relationship tells how two entities react with each other.

Dependency

An entity is dependent on another if the change of that is reflecting the other.

Aggregation

It is a special kind of association which exhibits part-of relationship.

Generalization

This relationship describes the parent-child relationship among different entities.

Realization

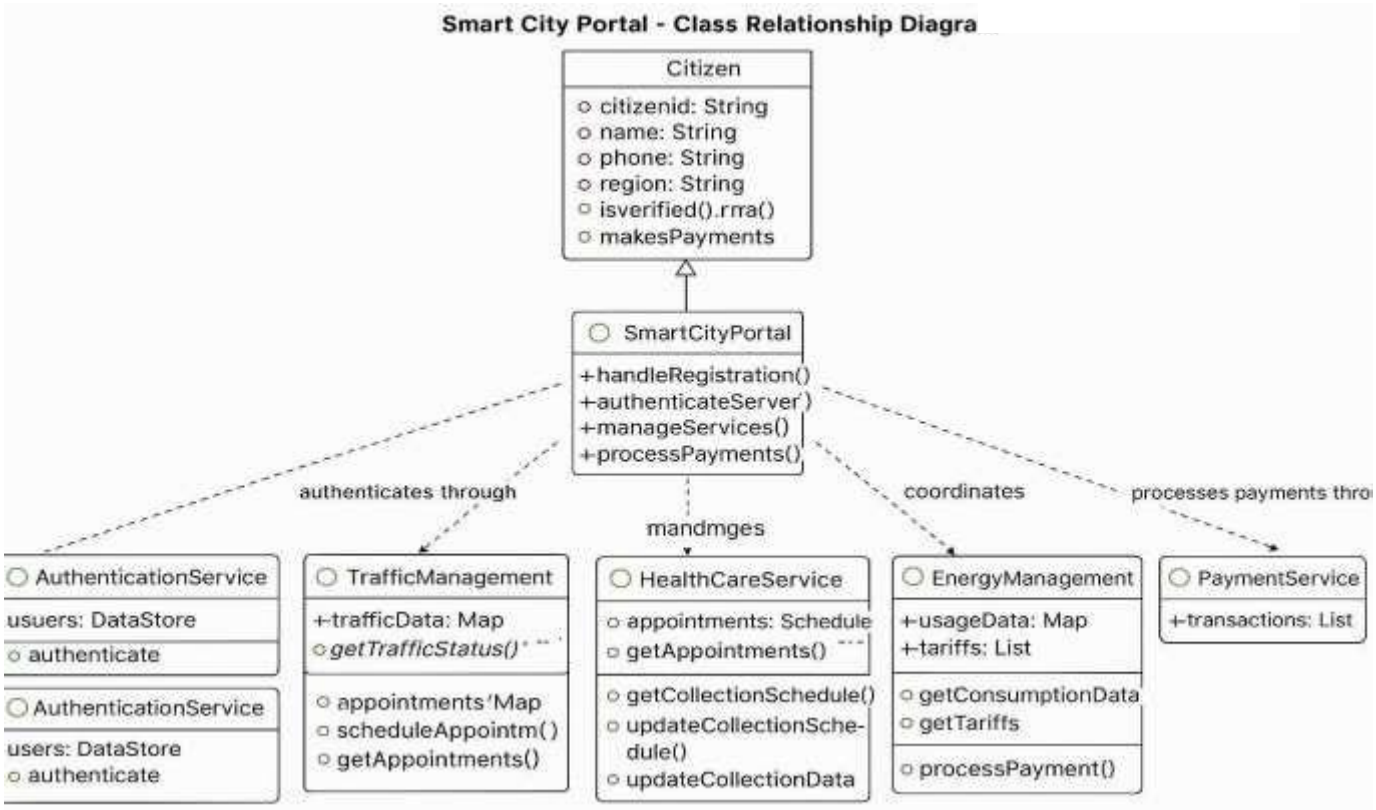


Fig 4.2 Relationships Diagram

Fig 4.1 describes the Relationships of UML Diagrams which are used to show the relationship between the entities.

4.2 UML DIAGRAMS

4.2.1 Use-Case Diagram

Use-Case diagrams capture the behavior of a system and help to emphasize the requirements of the system. It describes the high-level functionalities of the system. It consists of the following artifacts Actor, Use case.

Actor

- Initiates Register, Login, Make Payment, View Service Status

- **Authority** A user with administrative privileges to manage and monitor smart city operations.
- **Citizen** A regular user accessing city services through the portal.

2. System Smart City Portal

The system is divided into two sections

- Authority Services
- Citizen Services

3. Use Cases

Authority Services

- Login Authority logs into the system.
- Monitor Urban Data Allows the authority to access real-time data about the city, like traffic, pollution, etc.
- Manage Smart Contracts The authority can manage digital contracts related to city operations, possibly using blockchain technology.

Citizen Services

- Register Citizens can register or create an account in the portal.
- Access Traffic Management Citizens can view or interact with traffic-related services, such as traffic updates or route suggestions.
- Access Healthcare Management Citizens can use health services, view medical reports, book appointments, etc.
- Access Waste Management Citizens can report waste issues or view schedules for waste collection.
- Access Energy Management Allows access to services related to electricity, water, and other utilities.
- View Service Status Citizens can check the status of their requests or service availability.
- Make Payment Citizens can pay bills or service charges via the portal.

4. Relationships

- Each arrow shows that the actor initiates or interacts with the specific service.
- Use cases are grouped under respective service categories.

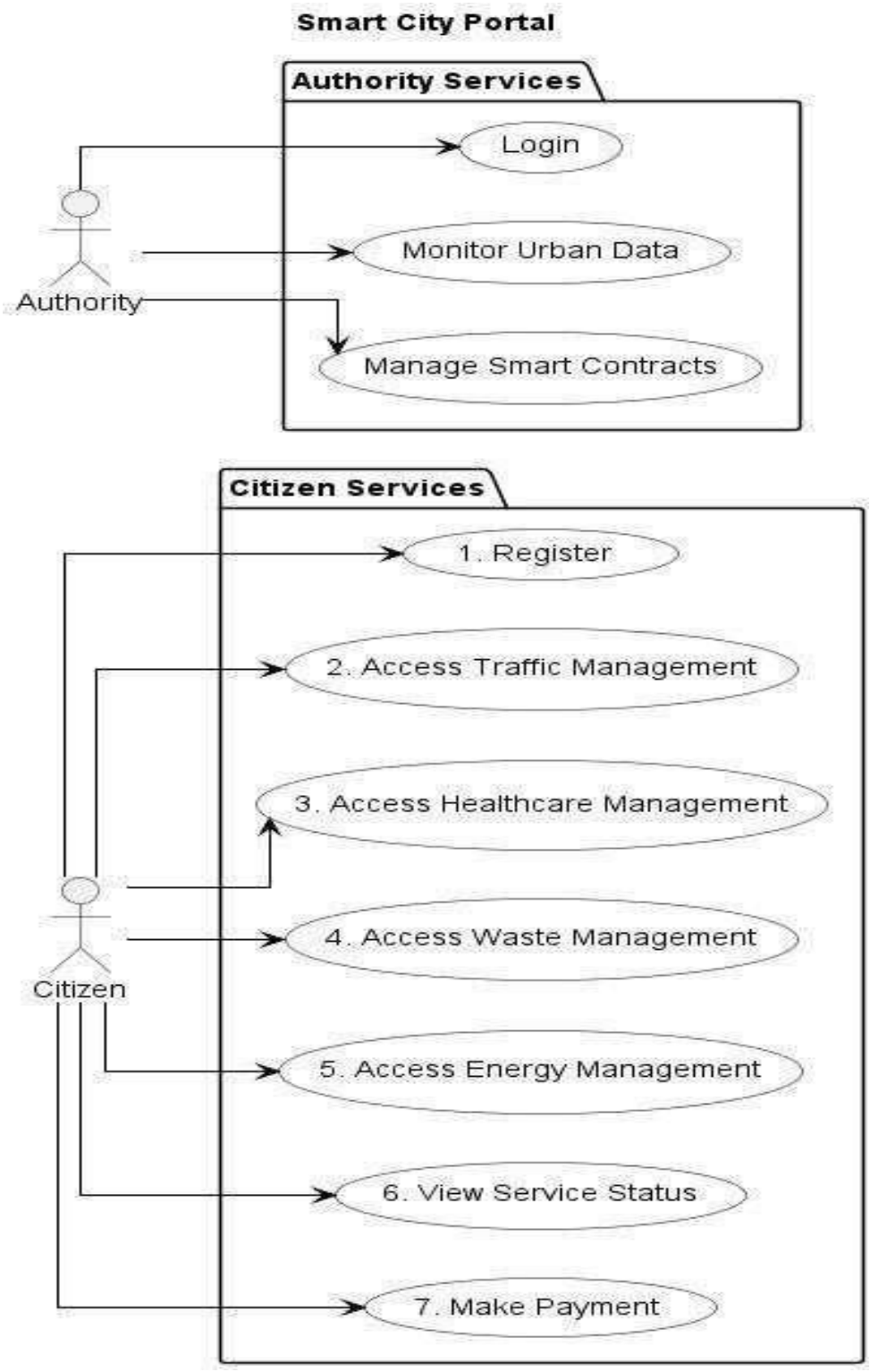


Fig 4.2.2 Use Case Diagram

Fig 4.2 Use Case Diagram shows .The Smart City Portal offers a comprehensive platform for both city authorities and citizens to engage with urban services efficiently.

4.2.3 Class Diagram

Class diagram is a static overview of the system used to highlight the important aspects as well as executables in the system. These are the only diagrams which can he mapped with the object-oriented systems. These diagrams show the responsibilities of a class and relationships among different classes in the system.

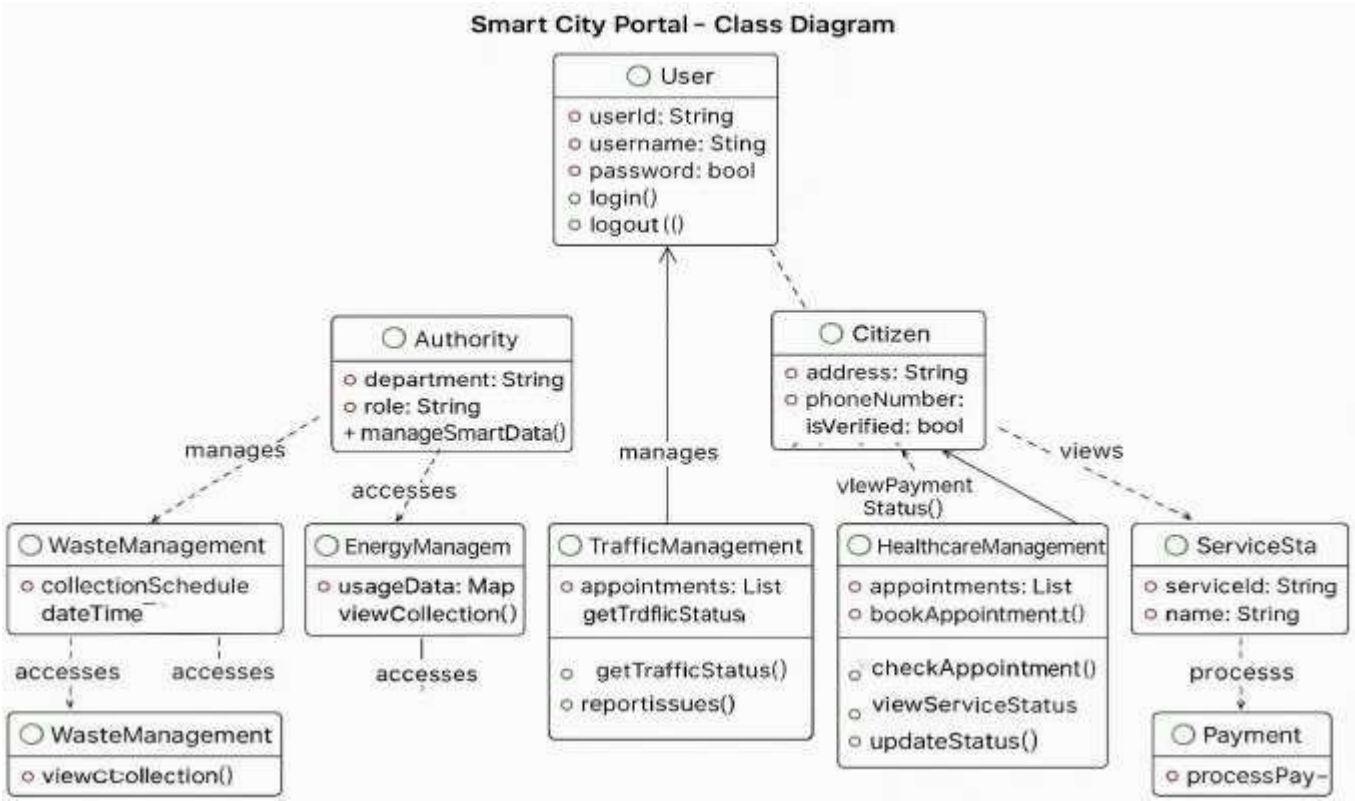


Fig 4.2.3 Class Diagram

Class Diagram is shown in Fig 4.2.3 Class diagram is a static representation of the structure of the system, illustrating how different entities interact with each other. It highlights the core components, their attributes, behaviors (methods), and the relationships among them. Class diagrams are especially useful for object-oriented design as they can be directly mapped to code structures.

Artifacts of the Class Diagram are

- **Class** A blueprint for objects, representing common properties and behaviors.
- **Name** The identifier of the class.

- **Attributes** Represent the data or properties associated with a class.
- **Responsibilities (Methods)** Define the behavior or operations the class can perform.

Description of Classes in the Smart City Portal

- **User** This is a base class representing all users in the system. It includes common attributes like userId, name, email, and password. It has methods for login(), logout(), and updateProfile().
- **Table 5 Hardware Requirements Authority** Also inherits from User and represents administrators or department heads. It contains department and role as attributes, and methods like monitor Urban Data() and manage Smart Contracts().
- **WasteManagement** Handles waste collection and scheduling. It includes attributes like area and collection Schedule, and methods schedule Pickup() and report Issue().
- **EnergyManagement** Manages energy usage and billing. It has attributes consumerId and usage Data, and supports view Consumption() and payBill().
- **TrafficManagement** Oversees traffic data and updates. Attributes include location and trafficStatus. Operations are getTrafficUpdates() and reportIssue().
- **Healthcare Management** Manages health services for citizens. It contains attributes like facility Name and service Type, and methods book Appointment() and viewMedicalRecords().
- **ServiceStatus** Tracks the status of various services requested by citizens. It contains serviceId, status, and lastUpdated, with methods checkStatus() and update Status().
- **Payment** Handles transaction-related operations. Includes payment Id, amount, date, and status. It provides functions like process Payment() and generate Receipt().
- Each class is connected based on their relationships and dependencies. Citizens access services, make payments, and view their status. Authorities manage and monitor specific services. This diagram helps structure the Smart City system into well-defined components for efficient implementation.

4.2.4 Sequence Diagram

A sequence diagram is a type of interaction diagram that shows how processes operate with one another and in what order. It is a dynamic model that depicts object interactions arranged in a time sequence. It visually represents the flow of messages and the interactions between different entities involved in the system.

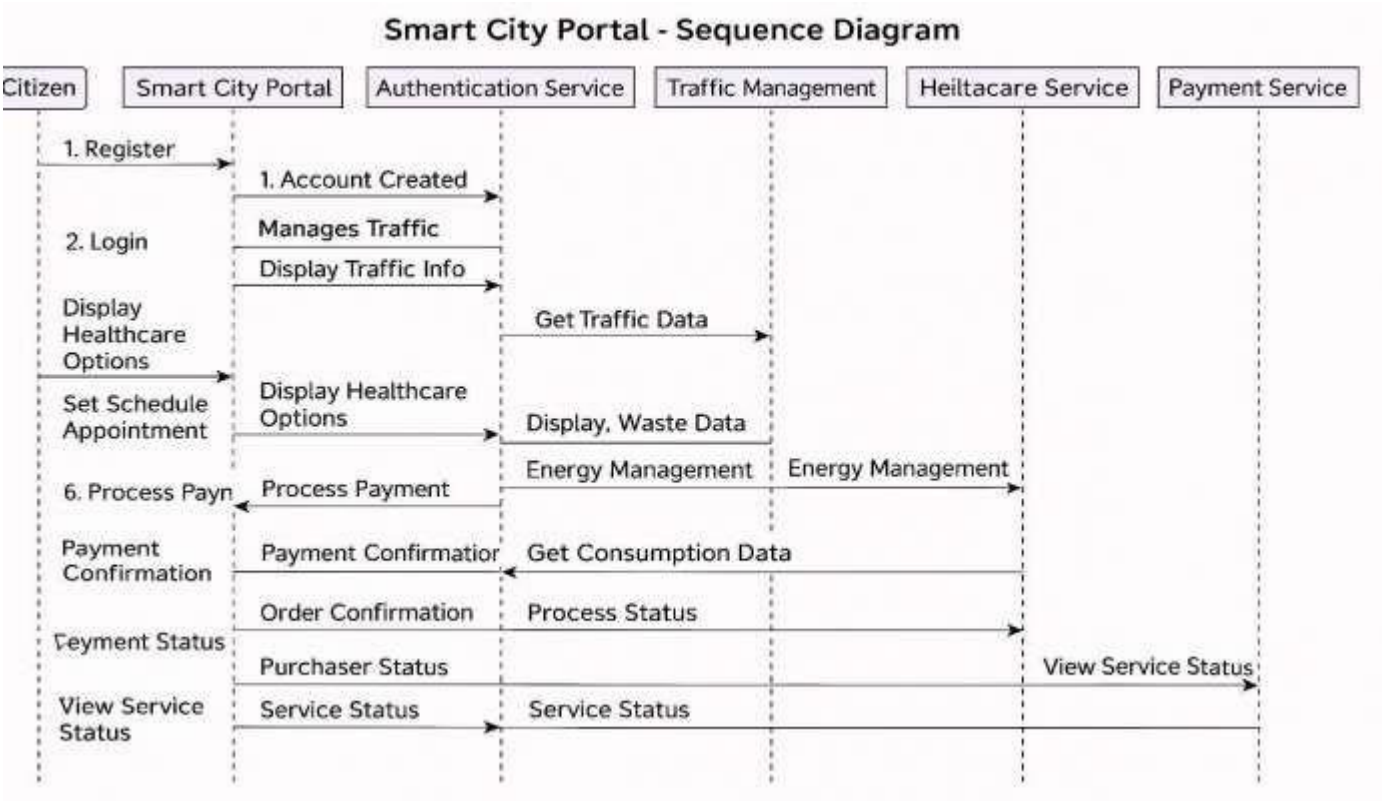


Fig 4.2.4 sequence Diagram

Sequence Diagram of the system is shown in Fig 4.4. This sequence diagram helps visualize the end-to-end service interaction flow and showcases how various backend services communicate with the front-end user requests, making the system efficient and organized.

- The interaction begins when the Citizen registers on the Smart City Portal, which then triggers account creation in the Authentication Service.
- After logging in, the citizen requests Traffic Info, and the Smart City Portal fetches data from the Traffic Management module and displays it.
- The citizen selects a healthcare option and schedules an appointment. This interaction routes to the Healthcare Service, which processes and confirms the appointment.
- Next, the citizen views waste management data. The Smart City Portal fetches and displays waste-related information from the Waste Management service.
- To manage energy usage, the portal fetches consumption data from the Energy Management module and displays it to the citizen.
- When a payment is initiated, the Smart City Portal processes the request and forwards it to the Payment Service.

- Upon processing, the payment status is confirmed and updated back to the citizen, including order and delivery status.
- Throughout the sequence, the Smart City Portal acts as a central coordinator, managing communication among all service modules and ensuring real-time responses to citizen actions.

4.3 MODULE DESIGN

The Smart City Portal is divided into several functional modules to streamline services and improve user interaction. Each module is designed to handle specific tasks and communicate with others through a centralized system. This modular approach enhances scalability, maintainability, and efficient processing of services.

4.3.1 User Module

The User Management System in the Smart City Portal is designed to manage user interactions securely and efficiently. It ensures proper registration, authentication, role-based access, and profile handling, while integrating with services like payment and email. The module also ensures data security and compliance with privacy regulations.

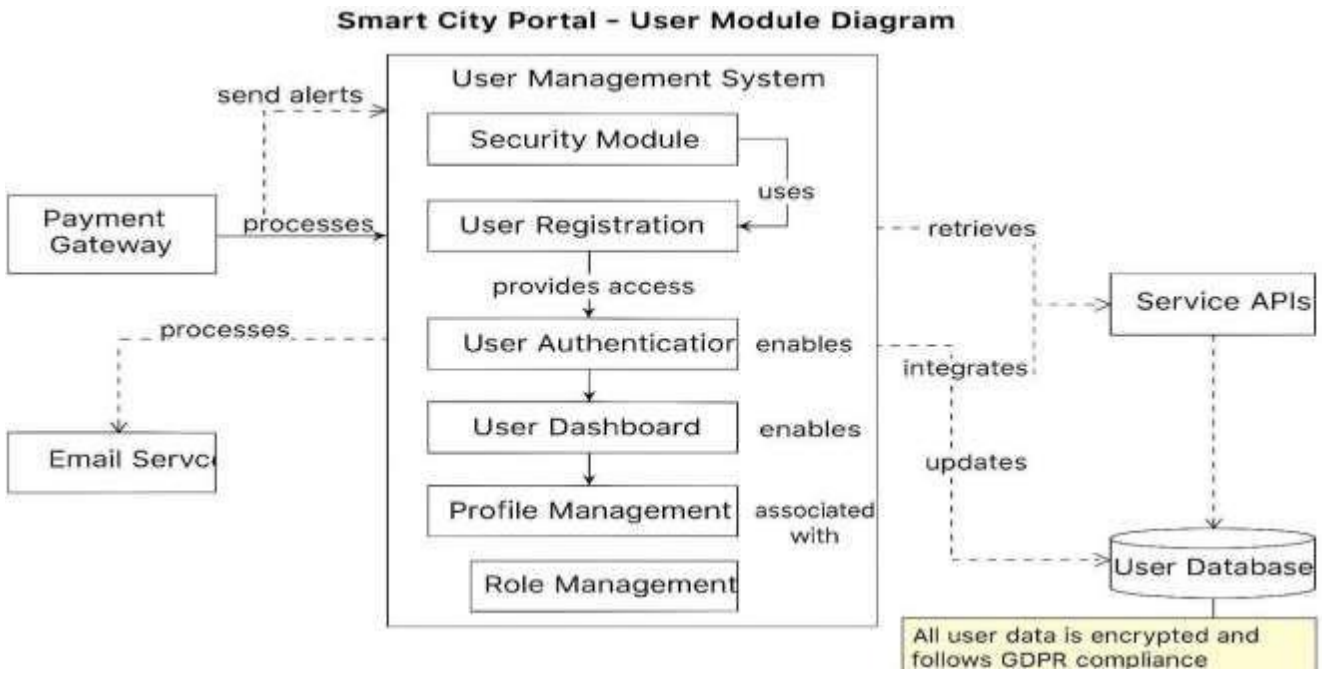


Fig 4.3.1 User Module Diagram

- User Registration This component allows new users to register by submitting necessary personal information. It is the entry point to access portal services and interacts with the authentication component for validation.

- User Authentication Validates user credentials during login. It ensures secure access by enforcing security protocols defined by the Security Module and grants access to the user dashboard.
- Security Module Enforces authentication protocols and handles alert generation. It works with the email service to send verification codes or alerts for suspicious activity.
- Email Service Sends alerts and verification messages to users. It acts as a notification bridge between the system and the users.
- User Dashboard Once authenticated, users are directed to the dashboard which displays services and options available to them, depending on their role.
- Profile Management Allows users to view and update their personal details. It is tightly linked with role management and updates the user database accordingly.
- Role Management Defines the permissions and capabilities of users (e.g., citizen or authority) and works with profile management to ensure access control.
- Service APIs Integrate the user system with other city services like traffic, energy, and healthcare. The APIs update and pull data from the user database when needed.
- User Database Stores all user-related data including login credentials, profile information, and service interactions. It is encrypted and adheres to GDPR compliance for data protection and privacy.
- Payment Gateway Handles financial transactions initiated from the user dashboard, like bill payments or service fees. It connects with the email service for payment confirmation alerts.

4.3.2 System Module

The System Module Design of the Smart City Portal outlines the interactions between different layers and components to deliver seamless smart city services. It integrates security, user handling, data processing, and service management while maintaining privacy standards and data encryption.

.

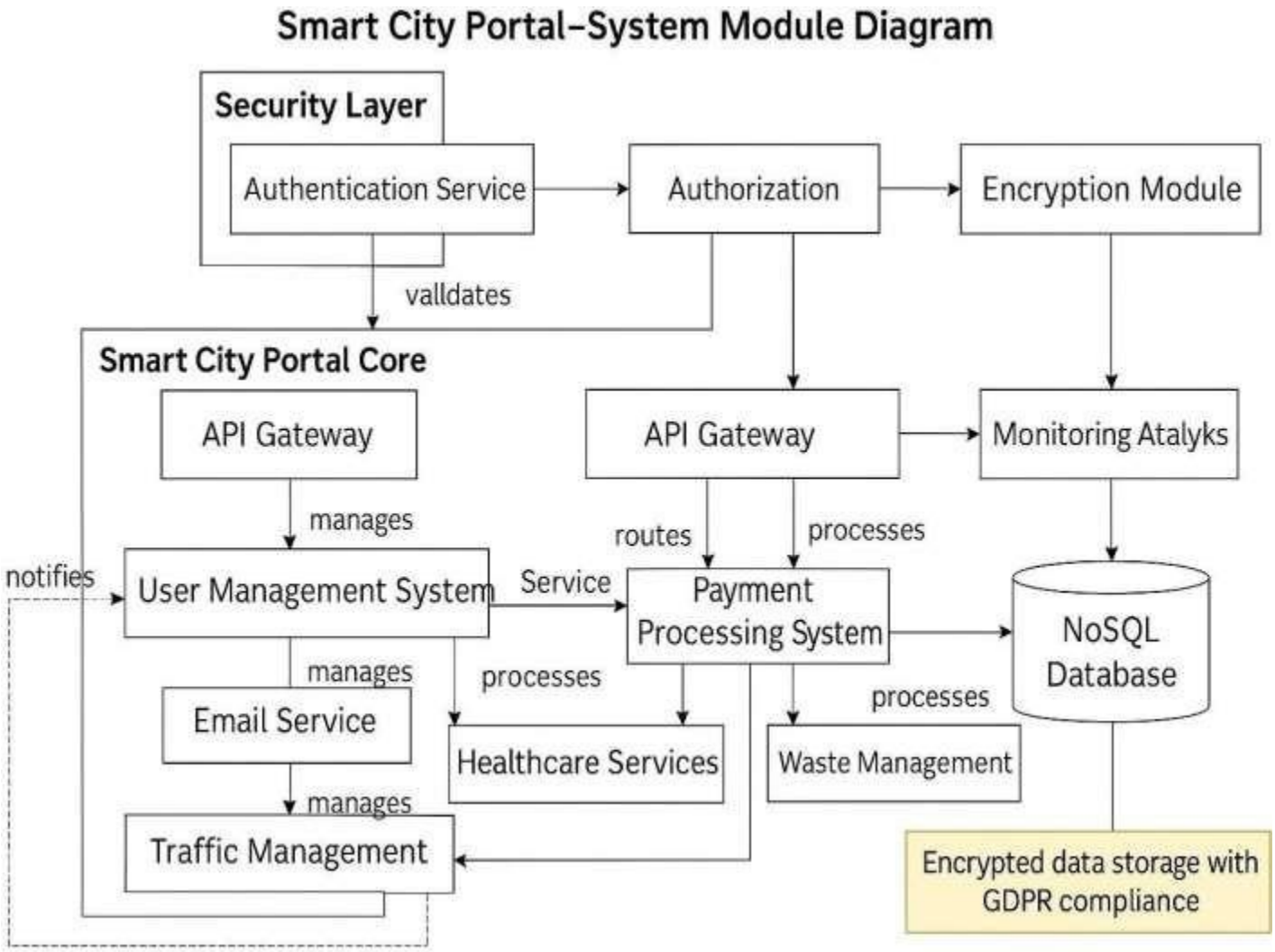


Fig. 4.3.2 System Module Architecture

4.4 CONCLUSION

The Smart City Portal is a comprehensive platform designed to streamline urban services like traffic, healthcare, waste, and energy management. It uses MongoDB for backend storage and IPFS for decentralized file handling, ensuring both scalability and security. The system avoids IoT integration, relying solely on backend technologies. Citizens can register, log in, access services, and make payments easily through a centralized dashboard. Authorities can monitor data, manage services, and interact with smart contracts. The portal includes secure modules for authentication and encryption, complying with GDPR guidelines. The backend infrastructure powered by MongoDB ensures robust handling of large-scale, unstructured data, while IPFS (InterPlanetary File System) enhances data availability and decentralized storage, ensuring higher security and accessibility. The absence of IoT components makes the system lightweight and more focused on web-based interaction, minimizing hardware dependencies.The system is designed to be modular, making it scalable and adaptable to future requirements. It emphasizes real-time data access and efficient service management. Overall, the Smart City Portal delivers a unified solution for smarter, safer, and more efficient urban governance. The architecture supports real-time updates and efficient processing. Overall, the portal enhances city management by offering a secure, accessible, and user-friendly experience.

5 . IMPLEMENTATION AND RESULTS

5.1 INTRODUCTION

The implementation phase of the Smart City Portal focuses on transforming the system design into a working application. This includes developing the front-end user interface, integrating back-end functionalities, and establishing secure communication between modules. The implementation ensures the system is functional, user-friendly, and meets the objectives of delivering smart urban services through an efficient web-based platform. The results demonstrate how well each component works individually and as part of the overall system.

5.2 SOFTWARE DESIGN

The software design of the Smart City Portal is structured into modular components, each responsible for specific functionality. It follows a layered architecture, ensuring separation of concerns, better scalability, and maintainability. The system includes user interface components, service APIs, and a secure backend connected to a decentralized storage solution using IPFS and a flexible NoSQL database using MongoDB.

5.2.1 Front-End Design

The front-end is developed using modern web technologies such as HTML, CSS, and JavaScript with frameworks like ReactJS to build a responsive and interactive user interface. The design ensures ease of navigation, real-time service access, and dynamic data rendering. Features like login, dashboard, service access (traffic, healthcare, waste, energy), and payment modules are clearly organized for a smooth user experience. The UI is also designed to be mobile-friendly and accessible for a wide range of users.

5.2.2 Back-End Design

The back-end design of the Smart City Portal is responsible for handling data processing, storage, and logic behind each module. It manages user data, service records, and transactions to ensure smooth functioning of the entire system. The back-end is connected to a database

where all user and service-related information is stored securely.The system supports modules such as traffic updates, waste management, energy usage, healthcare appointments, and payments. When a user interacts with the portal through the front-end, the back-end processes the request, retrieves or stores information in the database, and sends back the result. The design ensures that all services are available quickly and work correctly based on user input. Data is managed efficiently to support multiple users at the same time. The back-end also includes functions for user login, role checking (citizen or authority), and displaying only relevant information for each type of user. It ensures consistency and accuracy across all service modules.

5.3 METHOD OF IMPLEMENTATION

5.3.1 Sample code

Front end

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract UserAuth {
    struct User {
        bool isRegistered;
        string role;
        address walletAddress;
    }

    mapping(address => User) public users;
    address public admin;

    event UserRegistered(address indexed userAddress, string role);
    event RoleUpdated(address indexed userAddress, string newRole);

    constructor() {
        admin = msg.sender;
        users[msg.sender] = User(true, "admin", msg.sender);
    }

    modifier onlyAdmin() {
        require(msg.sender == admin, "Only admin can perform this action");
        _;
    }
function registerUser(address _userAddress, string memory _role) public onlyAdmin {
```

```
        require(!users[_userAddress].isRegistered, "User already registered");
        users[_userAddress] = User(true, _role, _userAddress);
        emit UserRegistered(_userAddress, _role);
    }

    function updateUserRole(address _userAddress, string memory _newRole) public
onlyAdmin {
        require(users[_userAddress].isRegistered, "User not registered");
        users[_userAddress].role = _newRole;
        emit RoleUpdated(_userAddress, _newRole);
    }

    function getUserRole(address _userAddress) public view returns (string memory) {
        require(users[_userAddress].isRegistered, "User not registered");
        return users[_userAddress].role;
    }

    function isUserRegistered(address _userAddress) public view returns (bool) {
        return users[_userAddress].isRegistered;
    }
}

const mongoose = require('mongoose');
```

```
const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  role: {
    type: String,
    enum: ['user', 'admin', 'authority'],
    default: 'user'
  },
  walletAddress: {
    type: String
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

module.exports = mongoose.model('User', userSchema);

{
```

```
  "name": "smart-city-portal",
  "version": "1.0.0",
  "description": "Smart City Portal with Blockchain Integration",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "client": "cd client && npm start",
    "dev": "concurrently \"npm run start\" \"npm run client\""
  },
  "dependencies": {
    "@google-cloud/vision": "^5.1.0",
    "axios": "^1.8.4",
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "express-validator": "^7.2.1",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^7.8.6",
    "multer": "^1.4.5-lts.2",
    "recharts": "^2.15.1",
    "uuid": "^11.1.0"
  },
  "devDependencies": {
    "concurrently": "^8.0.1",
    "nodemon": "^2.0.22"
  }
}
```

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');
const path = require('path');

// Import all models
const { ParkingLocation, ParkingBooking } = require('./models/Parking');
const { EVStation, EVBooking } = require('./models/EVCharging');
const { TollPlaza, FASTag, Transaction } = require('./models/FASTag');
const { HealthMetric, HealthStatistic, WHOReport } = require('./models/Healthcare');

// Import routes
const parkingRoutes = require('./routes/parkingRoutes');
const evChargingRoutes = require('./routes/evChargingRoutes');
const fastagRoutes = require('./routes/fastagRoutes');
const healthcareRoutes = require('./routes/healthcareRoutes');
const userRoutes = require('./routes/userRoutes');
const navigationRoutes = require('./routes/navigationRoutes');
const challanRoutes = require('./routes/challanRoutes');

dotenv.config();

const app = express();

// Middleware
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Connect to MongoDB
```

```
console.log('Attempting to connect to MongoDB...');
mongoose.connect(process.env.MONGODB_URI
'mongodb://localhost:27017/smartportal', {
useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => {
  console.log('✓ MongoDB Connected Successfully');

  // Initialize all collections with default data
  Promise.all([
    // Initialize parking locations
    ParkingLocation.initializeDefaultLocations(),

    // Initialize EV stations
    EVStation.initializeDefaultStations(),

    // Initialize toll plazas
    TollPlaza.initializeDefaultPlazas(),

    // Initialize health metrics and WHO reports
    HealthMetric.initializeDefaultMetrics(),
    WHOReport.initializeDefaultReports()
  ]).then(() => {
    console.log('✓ All collections initialized with default data');

    console.error('Error initializing collections:', err);
  });
})
.catch(err => console.error('MongoDB connection error:', err));
```

```
handling middleware
app.use((err, req, res, next) => {
console.error('Error:', err);
  res.status(500).json({ message: 'Internal Server Error', error: err.message });
});

// Serve static files in production
if (process.env.NODE_ENV === 'production') {
  app.use(express.static(path.join(__dirname, 'client/frontend/build')));
  app.get('*', (req, res) => {
    res.sendFile(path.join(__dirname, 'client/frontend/build', 'index.html'));
  });
}

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`✓ Server running on port ${PORT}`);
  console.log(`✓ API endpoints available at http://localhost:${PORT}/api`);
});
const mongoose = require('mongoose');

const violationSchema = new mongoose.Schema({
  code: String,
  description: String,
  penalty: Number,
  points: Number // Demerit points
});

const challanSchema = new mongoose.Schema({
  challanNumber: {
    type: String,
```

```
// Error  vehicleNumber: String,
  vehicleType: String,
  ownerName: String,
  location: {
    name: String,
    coordinates: {
      latitude: Number,
      longitude: Number
    }
  },
  violations: [violationSchema],
  totalAmount: Number,
  status: String, // Pending, Paid, Disputed
  issuedBy: {
    officerId: String,
    officerName: String,
    department: String
  },
  evidenceUrls: [String], // URLs to photos/videos
  issueDate: Date,
  dueDate: Date,
  paymentDetails: {
  transactionId: String,
  paidAmount: Number,
  paidDate: Date,
  method: String
  }
}, { timestamps: true });

// Static method to initialize default violations
violationSchema.statics.initializeDefaultViolations = async function() {
const defaultViolations = [
```

```
    await this.deleteMany({});
    await this.insertMany(defaultViolations);
    console.log('✓ Default violations initialized');
  } catch (err) {
    console.error('Error initializing violations:', err);
  }
};

// Static method to initialize default challans
challanSchema.statics.initializeDefaultChallans = async function() {
const defaultChallans = [
    {
      challanNumber: "CHN-2025-001",
      vehicleNumber: "TN-01-AB-1234",
      vehicleType: "Car",
      ownerName: "John Doe",
      location: {
        name: "Anna Salai Signal",
        coordinates: { latitude: 13.0569, longitude: 80.2425 }
      },
      violations: [
        {
          code: "RLV-01",
          description: "Red Light Violation",
          penalty: 1500,
          points: 4
        }
      ],
      totalAmount: 1500,
      status: "Pending",
      issuedBy: {
```

```
officerName: "Officer Kumar",
      department: "Traffic Police"
    },
    evidenceUrls: ["traffic-cam-001.jpg"],
    issueDate: new Date(),
    dueDate: new Date(Date.now() + 15 * 24 * 60 * 60 * 1000) // 15 days from now
  },
  {
    challanNumber: "CHN-2025-002",
    vehicleNumber: "TN-02-CD-5678",
    vehicleType: "Two Wheeler",
    ownerName: "Jane Smith",
    location: {
      name: "T Nagar Signal",
      coordinates: { latitude: 13.0418, longitude: 80.2341 }
    },
    violations: [
      {
        code: "NHM-01",
        description: "Not Wearing Helmet",
        penalty: 500,
        points: 2
      }
    ],
    totalAmount: 500,
    status: "Paid",
    issuedBy: {
      officerId: "TRF-002",
      officerName: "Officer Ravi",
      department: "Traffic Police"
    },
    evidenceUrls: ["traffic-cam-002.jpg"],
```

```
    dueDate: new Date(Date.now() - 5 * 24 * 60 * 60 * 1000), // 5 days ago
    paymentDetails: {
     transactionId: "PAY-001",
     paidAmount: 500,
     paidDate: new Date(Date.now() - 10 * 24 * 60 * 60 * 1000), // 10 days ago
     method: "Online"
    }
   }
 ];

 try {
  await this.deleteMany({});
  await this.insertMany(defaultChallans);
  console.log('✓ Default challans initialized');
 } catch (err) {
  console.error('Error initializing challans:', err);
 }
};

const Violation = mongoose.model('Violation', violationSchema);
const Challan = mongoose.model('Challan', challanSchema);

// Initialize default data
Violation.initializeDefaultViolations().catch(console.error);
Challan.initializeDefaultChallans().catch(console.error);
module.exports = { Violation, Challan };
const mongoose = require('mongoose');
const chargingTypeSchema = new mongoose.Schema({
 type: String,
 power: Number,
 price: Number
});
```

```
const evStationSchema = new mongoose.Schema({
  name: String,
  location: String,
  coordinates: {
  latitude: Number,
  longitude: Number
  },
  totalPorts: Number,
  availablePorts: Number,
  chargingTypes: [chargingTypeSchema],
  amenities: {
    parking: Boolean,
    restroom: Boolean,
    cafe: Boolean
  },
  operatingHours: {
    weekday: String,
    weekend: String
  },
  status: String
}, { timestamps: true });

const evBookingSchema = new mongoose.Schema({
  stationId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'EVStation'
  },
  portNumber: Number,
  vehicleNumber: String,
  startTime: Date,
  endTime: Date,
  chargingType: String,
```

```
    amount: Number,
    status: String,
    transactionId: String
  }
}, { timestamps: true });

// Static method to initialize default stations
evStationSchema.statics.initializeDefaultStations = async function() {
const defaultStations = [
    {
      name: "T Nagar EV Charging Hub",
      location: "T Nagar, Chennai",
      coordinates: { latitude: 13.0418, longitude: 80.2341 },
      totalPorts: 10,
      availablePorts: 10,
      chargingTypes: [
        { type: "Fast DC", power: 50, price: 18 },
        { type: "AC Level 2", power: 7.4, price: 12 }
      ],
      amenities: { parking: true, restroom: true, cafe: true },
      operatingHours: { weekday: "24/7", weekend: "24/7" },
      status: "Active"
    },
    {
      name: "Anna Nagar EV Station",
      location: "Anna Nagar, Chennai",
      coordinates: { latitude: 13.0850, longitude: 80.2101 },
      totalPorts: 8,
      availablePorts: 8,
      chargingTypes: [
        { type: "Fast DC", power: 50, price: 18 },
        { type: "AC Level 2", power: 7.4, price: 12 }
```

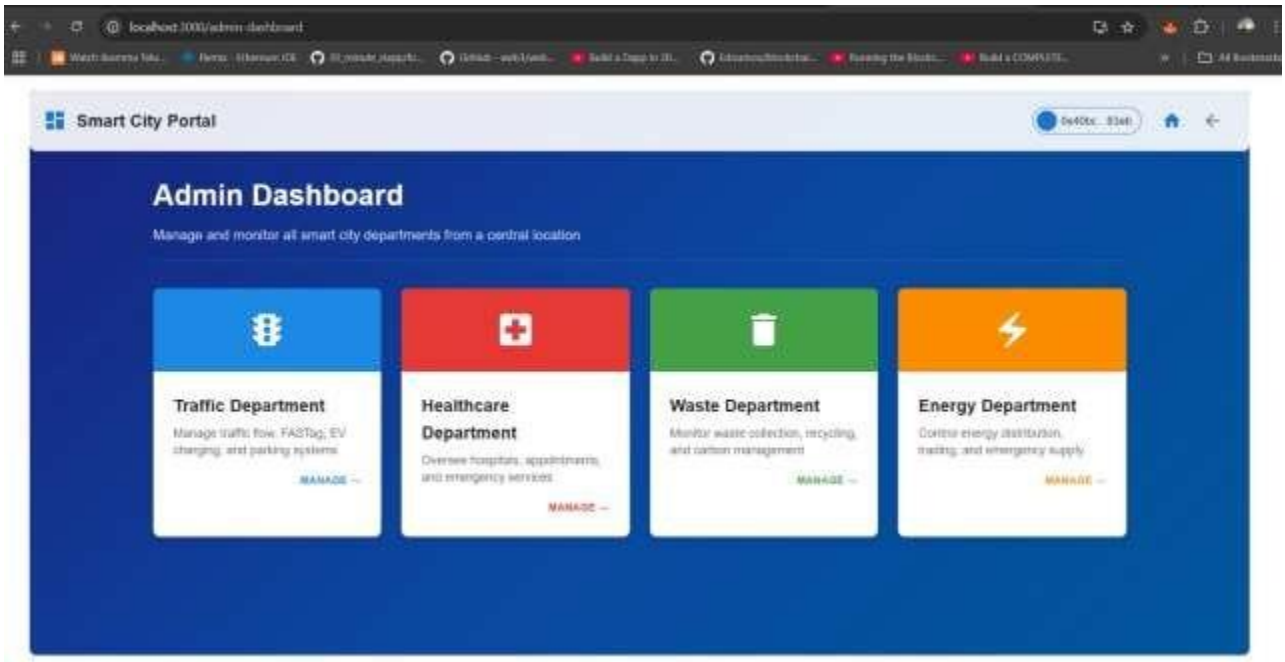
```
  },
  {
    name: 'Losartan',
    category: 'Blood Pressure & Heart',
    description: 'Angiotensin receptor blocker for blood pressure',
    brands: ['Cozaar', 'Losar', 'Zaart'],
    uses: ['High blood pressure', 'Kidney protection in diabetes', 'Heart failure'],
    sideEffects: ['Dizziness', 'High potassium', 'Cough'],
     precautions: ['Regular kidney function tests', 'Avoid in pregnancy', 'Monitor blood pressure'],
    dosageForm: ['Tablets'],
    prescriptionRequired: true
  },
  {
    name: 'Metoprolol',
    category: 'Blood Pressure & Heart',
    description: 'Beta-blocker for heart and blood pressure',
    brands: ['Lopressor', 'Metolar', 'Betaloc'],
    uses: ['High blood pressure', 'Heart rhythm problems', 'Angina'],
    sideEffects: ['Fatigue', 'Slow heart rate', 'Cold hands and feet'],
     precautions: ['No sudden stopping', 'Regular heart rate monitoring', 'Report breathing problems'],
    dosageForm: ['Tablets', 'Extended-release tablets'],
    prescriptionRequired: true
  },
  {
    name: 'Aspirin (Low Dose)',
    category: 'Blood Pressure & Heart',
    description: 'Blood thinner for heart attack prevention',
    brands: ['Ecosprin', 'Baby Aspirin', 'Cardioprin'],
    uses: ['Heart attack prevention', 'Stroke prevention', 'Blood clot prevention'],
```

```
const populateDatabase = async () => {
  try {
    await mongoose.connect('mongodb://localhost:27017/smartcity', {
      useNewUrlParser: true,
      useUnifiedTopology: true
    });

    // Clear existing medicines
    await Medicine.deleteMany({});

    // Insert new medicines
    for (const category in medicineData) {
      for (const medicine of medicineData[category]) {
        await Medicine.create(medicine);
        console.log(`Added ${medicine.name}`);
      }
    }

    console.log('Database populated successfully');
    process.exit(0);
  } catch (error) {
    console.error('Error populating database:', error);
    process.exit(1);
  }
};

populateDatabase();
```

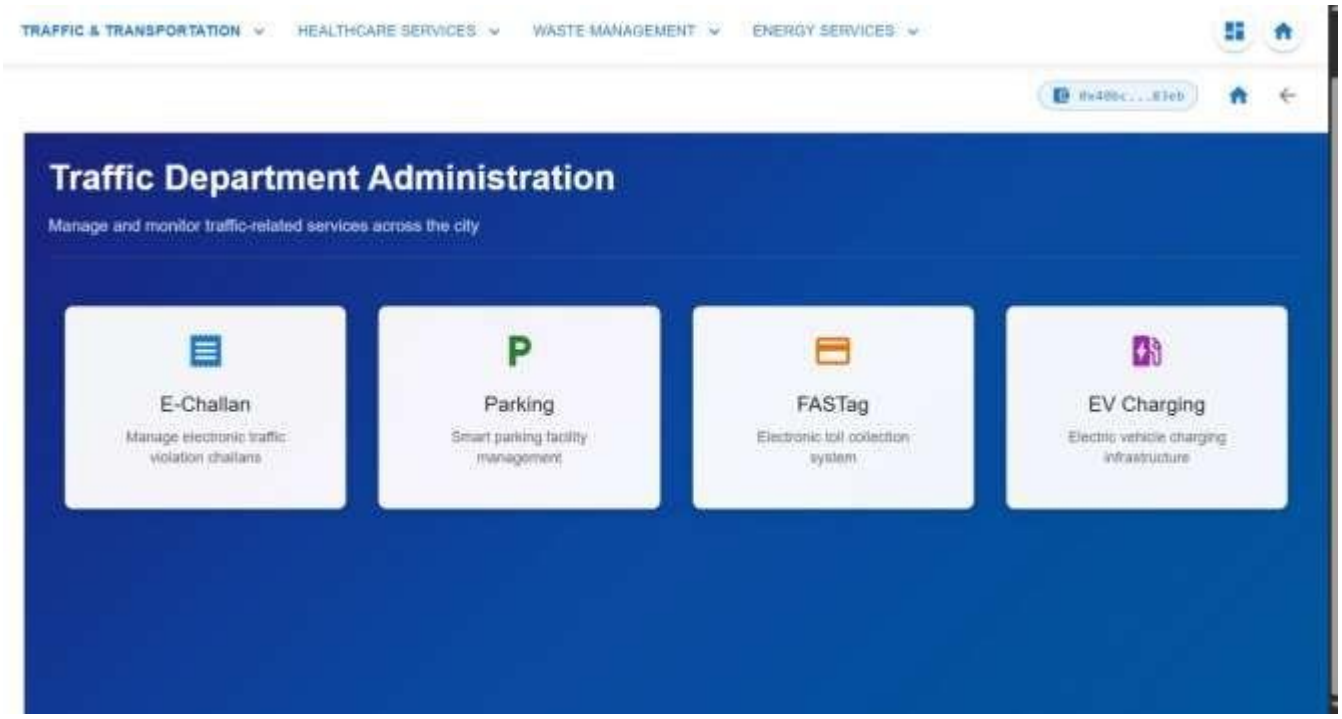
```
module.exports = {
  networks: {
  development: {
     host: "127.0.0.1",     // Localhost (default: none)
     port: 7545,            // Standard Ganache port
     network_id: "5777",    // Match Ganache network id
   },
   ganache: {
     host: "127.0.0.1",
     port: 7545,
     network_id: "5777"     // Match Ganache network id
   }
  },

  // Configure your compilers
  compilers: {
   solc: {
     version: "0.8.19",    // Fetch exact version from solc-bin
     settings: {
       optimizer: {
         enabled: true,
```

5.2.3 OUTPUT SCREENS

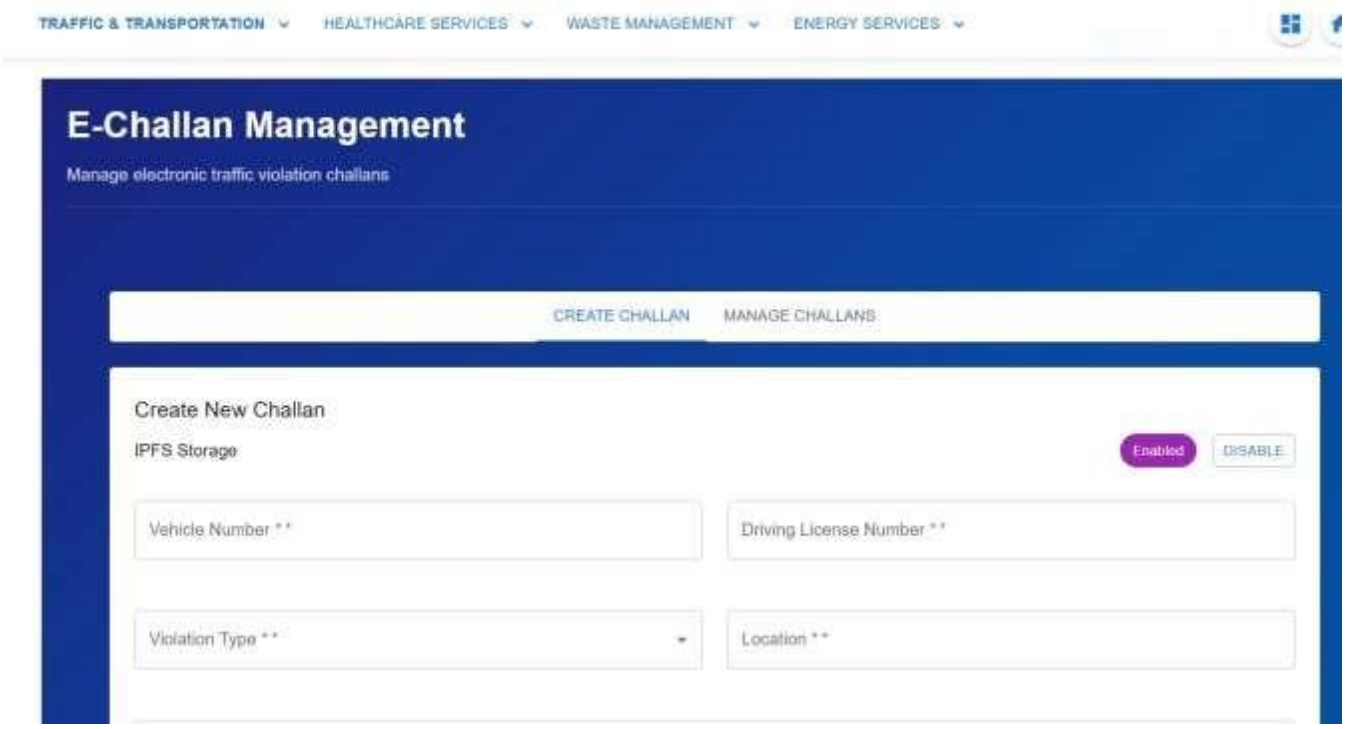
Screen 1 : Login Page



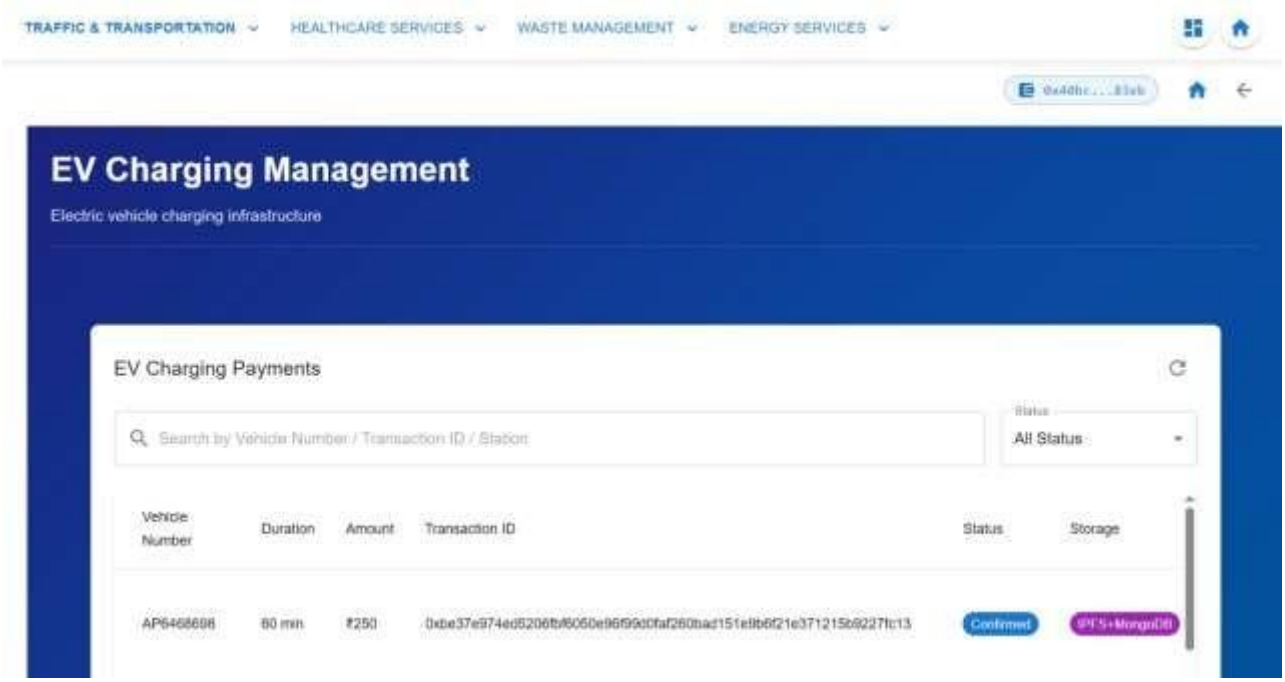
Screen 2 : Dashboard



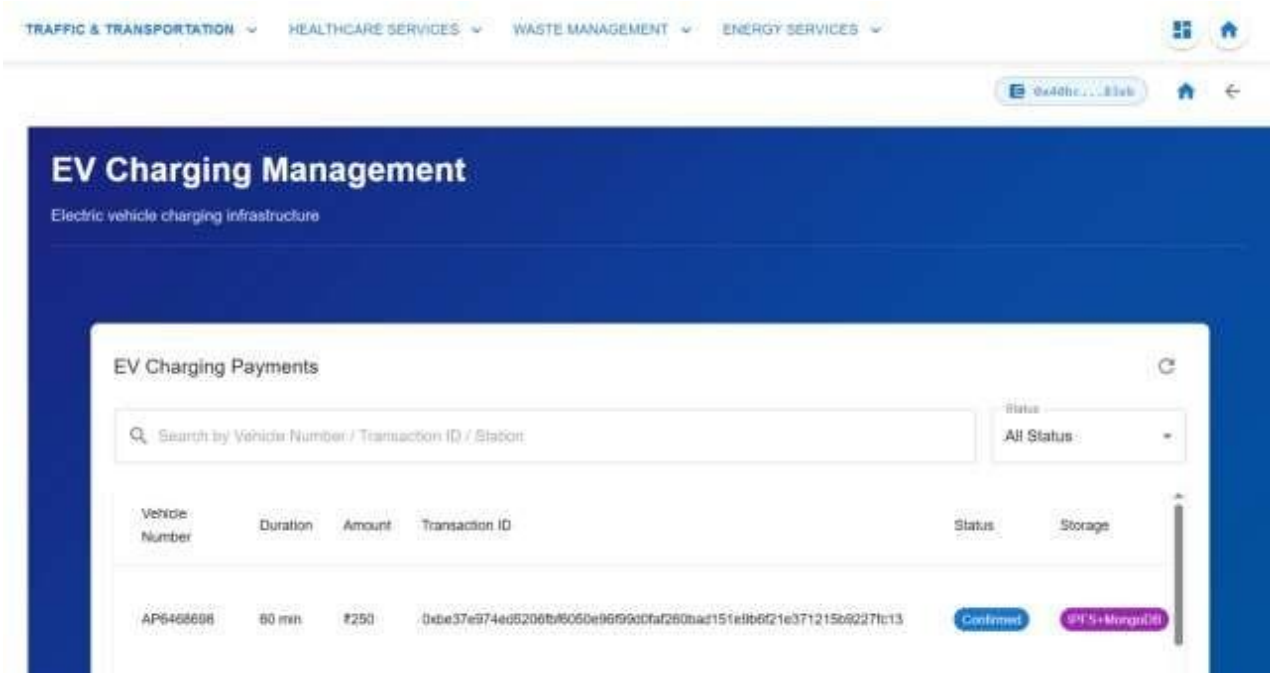
Screen 3: Traffic Management



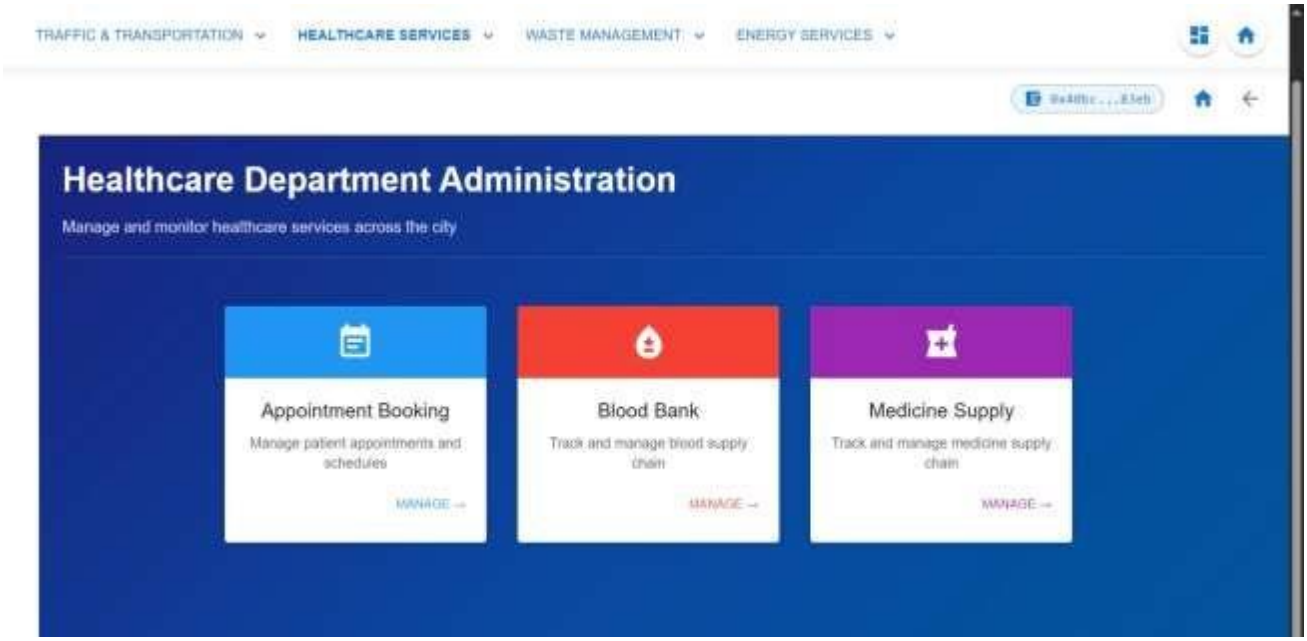
Screen 4: E-Challan Mangagement



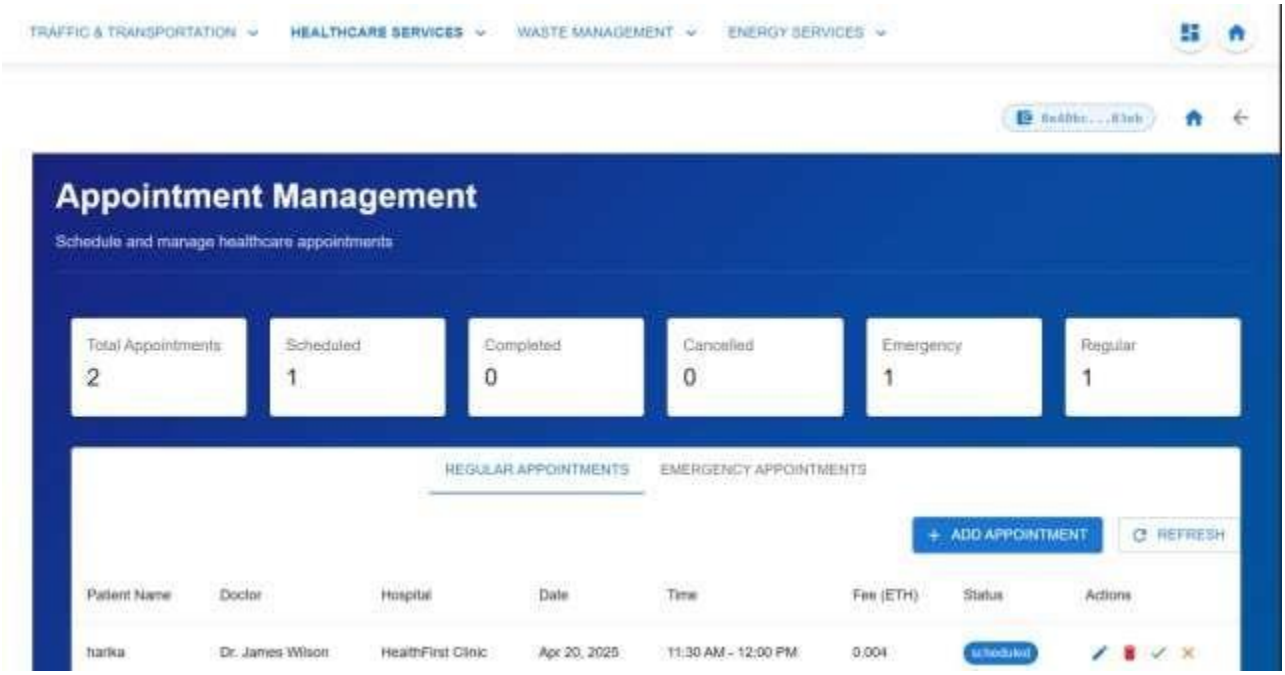
Screen 5 : EV Changing



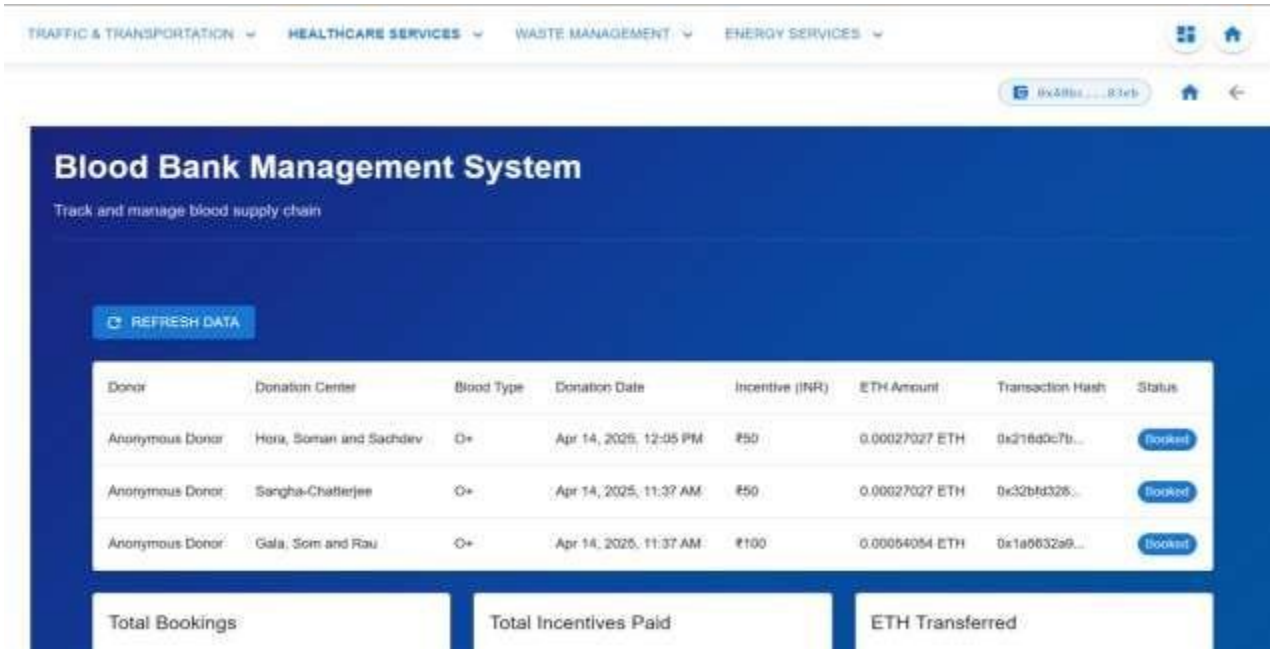
Screen 6: Manage Stations



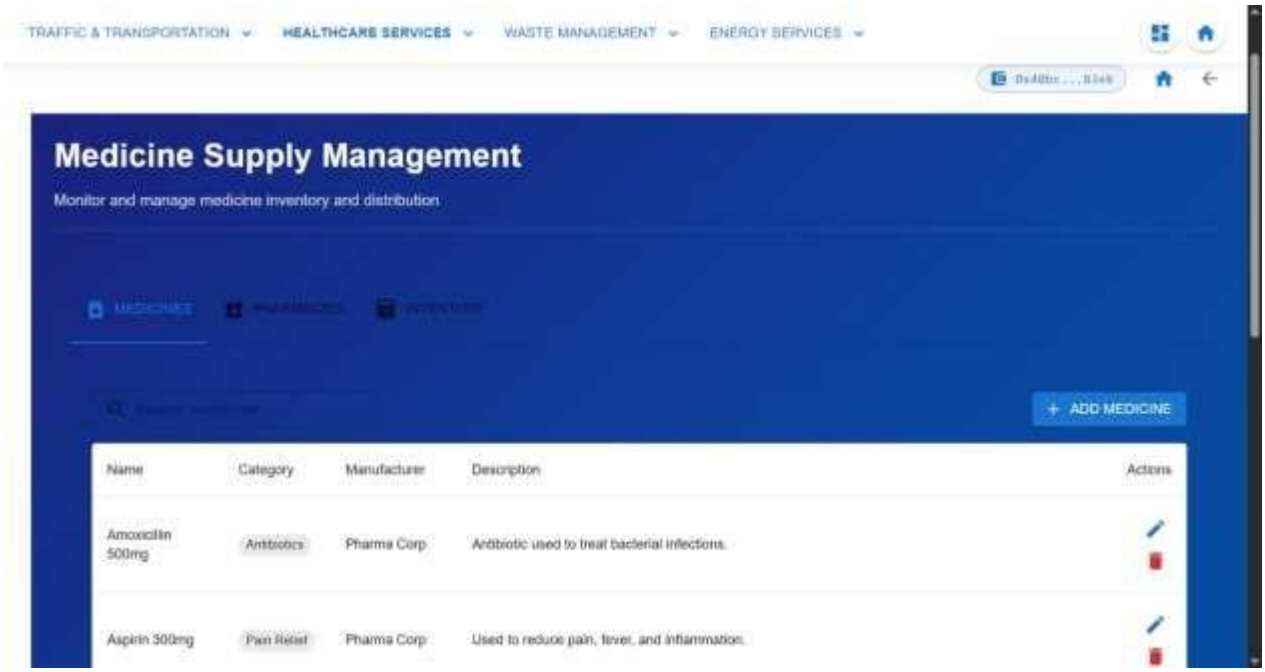
Screen 7 : Healthcare Department Administration



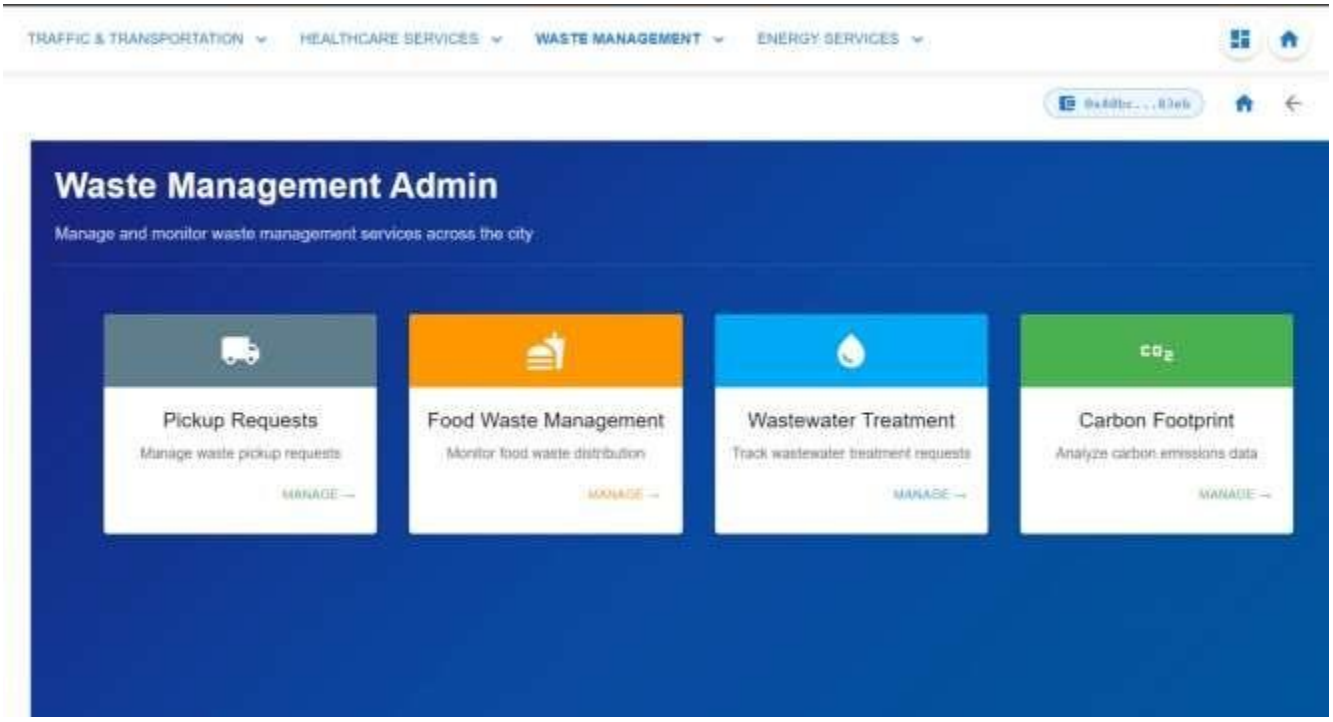
Screen 8 : Appointment management



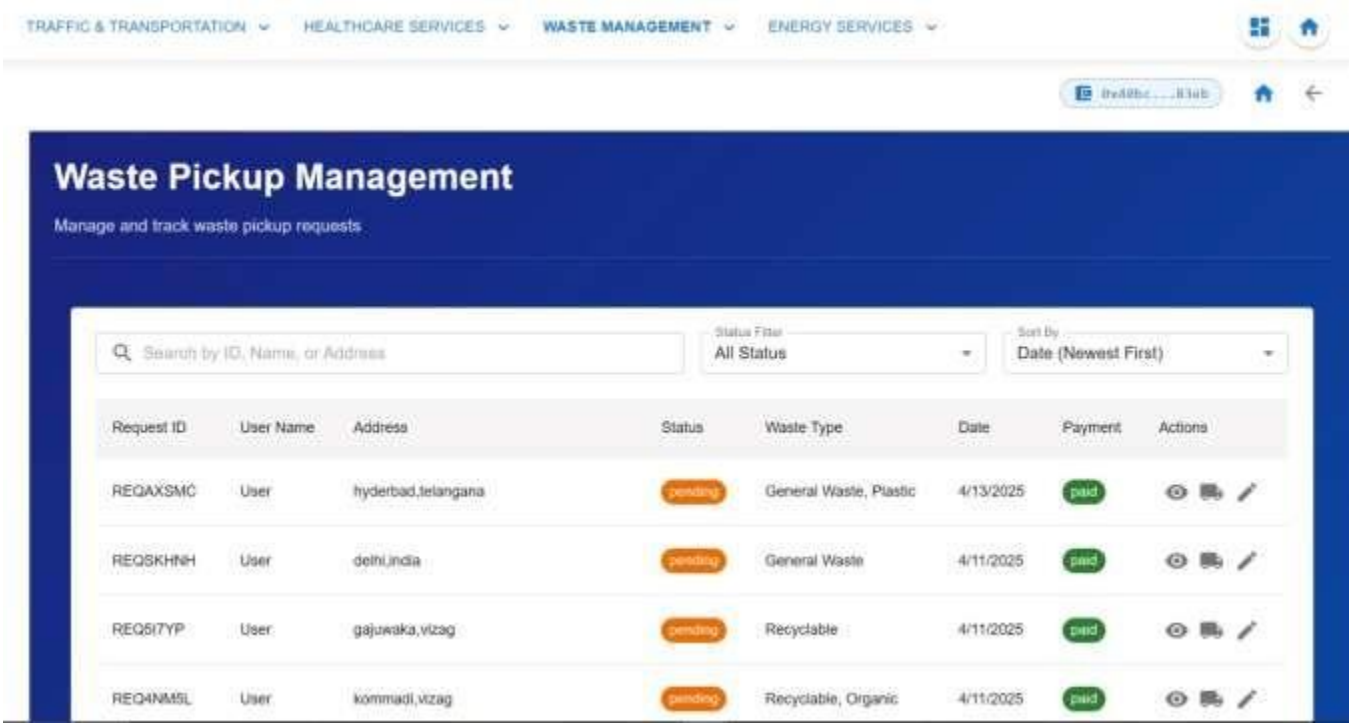
Screen 9 : Blood Bank Management System



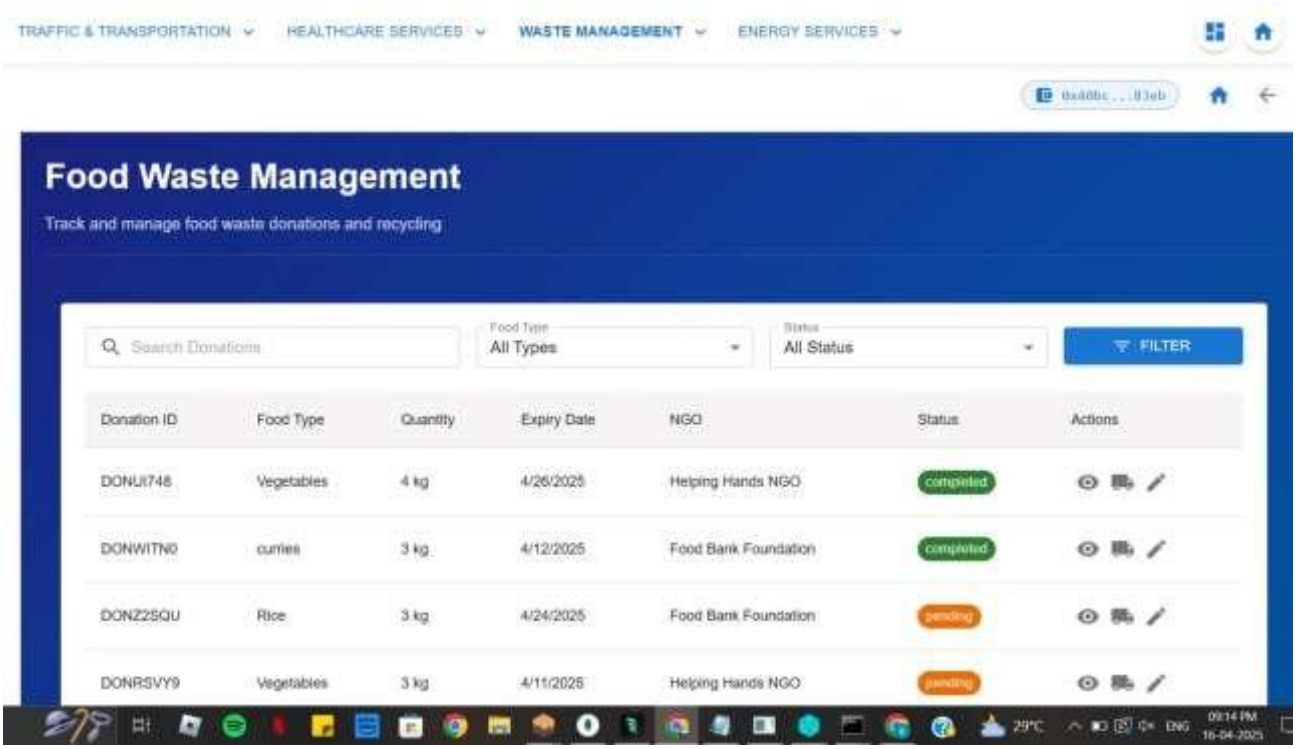
Screen 10 : Medicine Supply Management



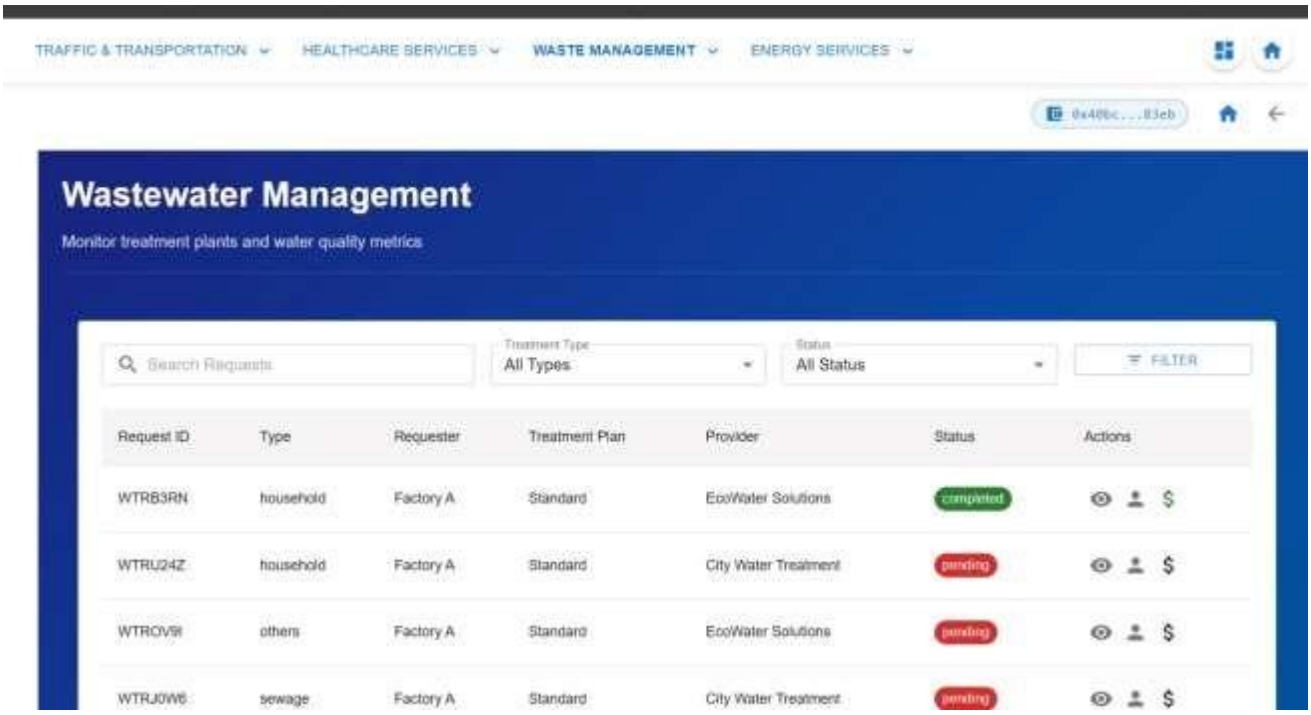
Screen 11 : Waste Management Admin



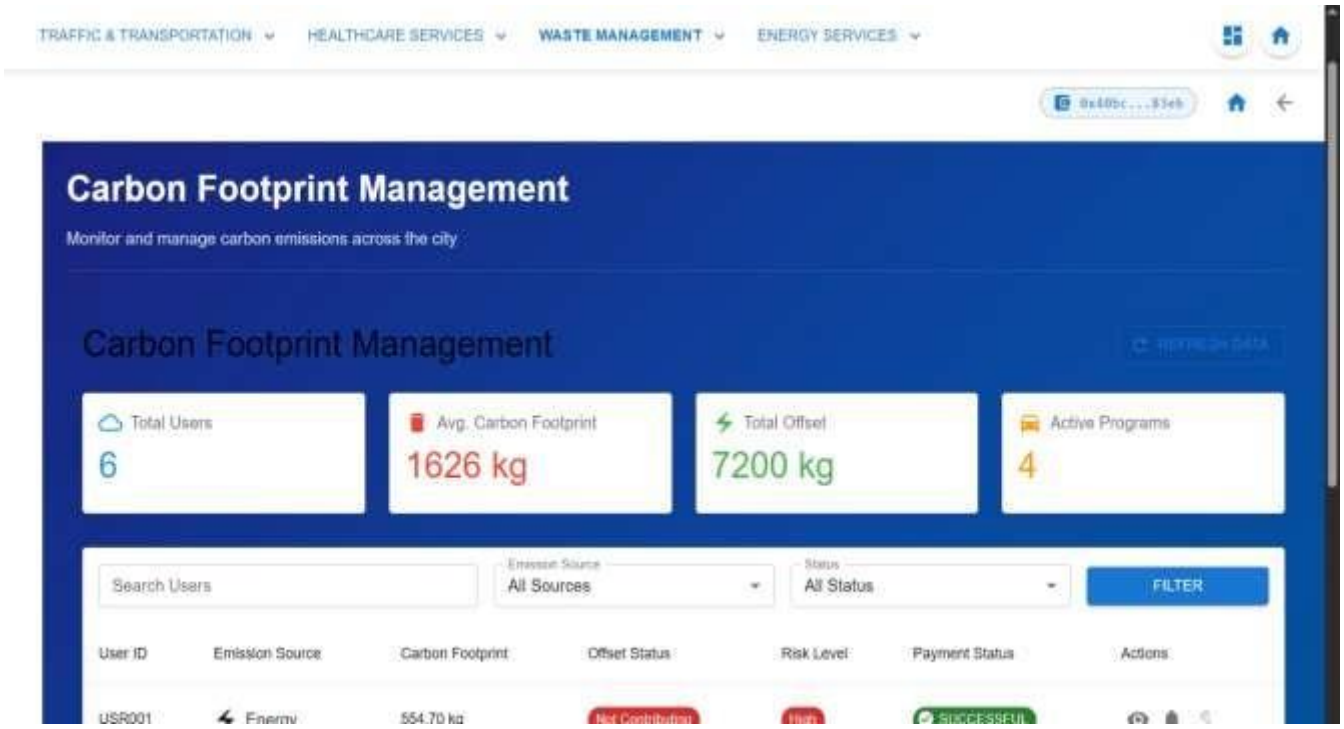
Screen 12 : Waste Pickup Management



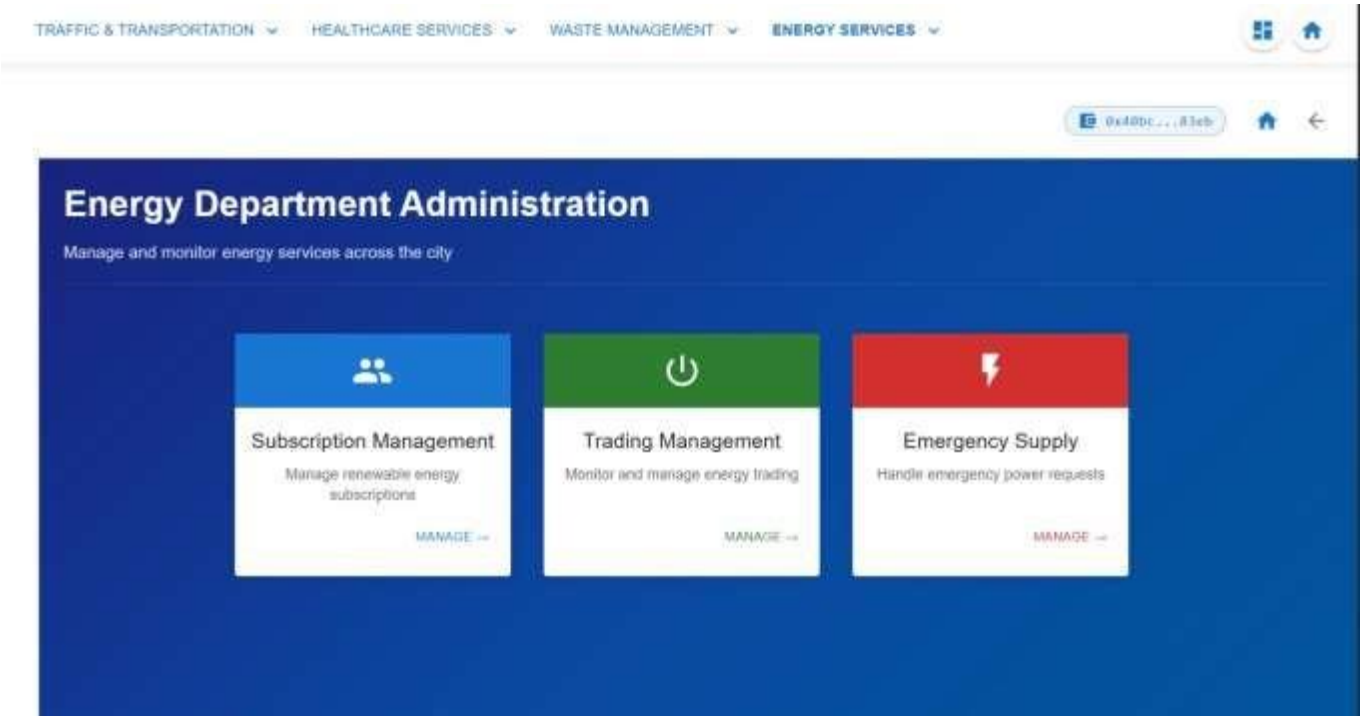
Screen 13 : Food Waste Management



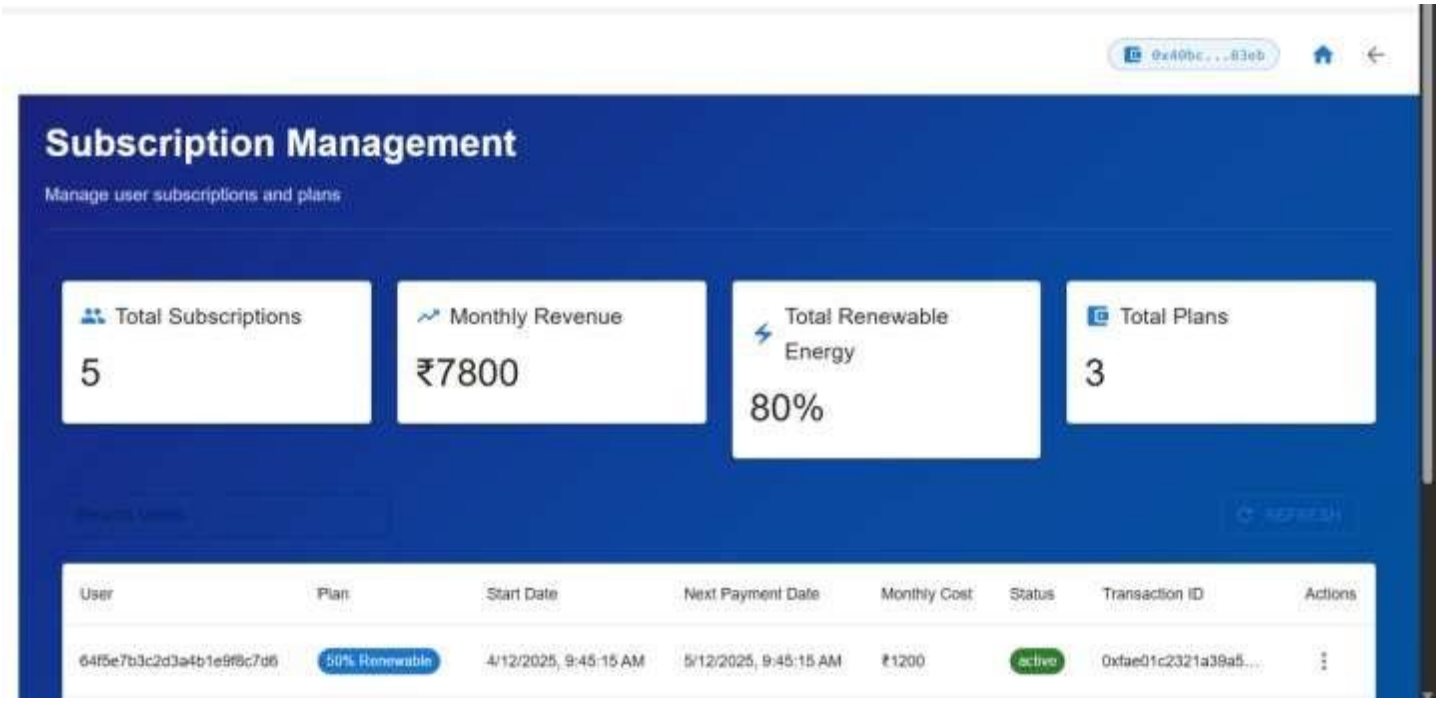
Screen 14 :Waste Water management



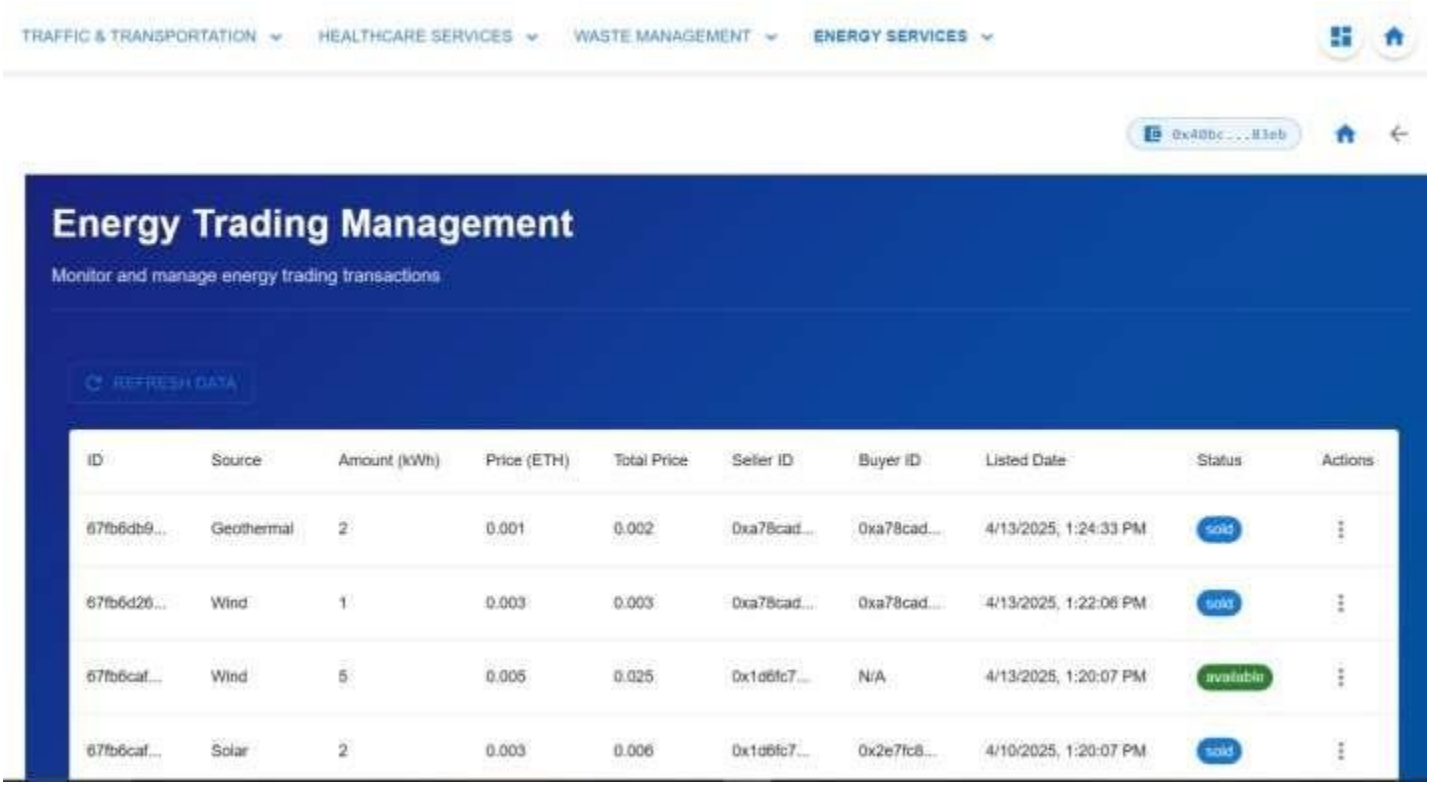
Screen 15 : carbon Footprint Management



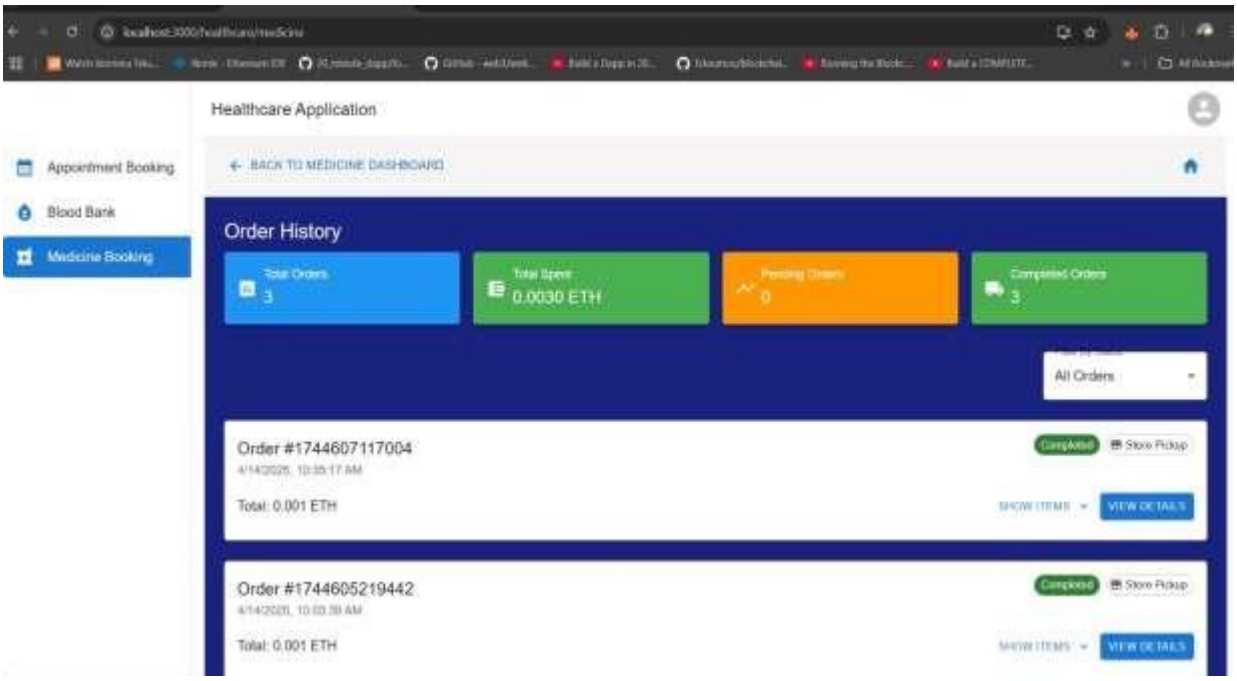
Screen 16 : Energy Department Adminstration



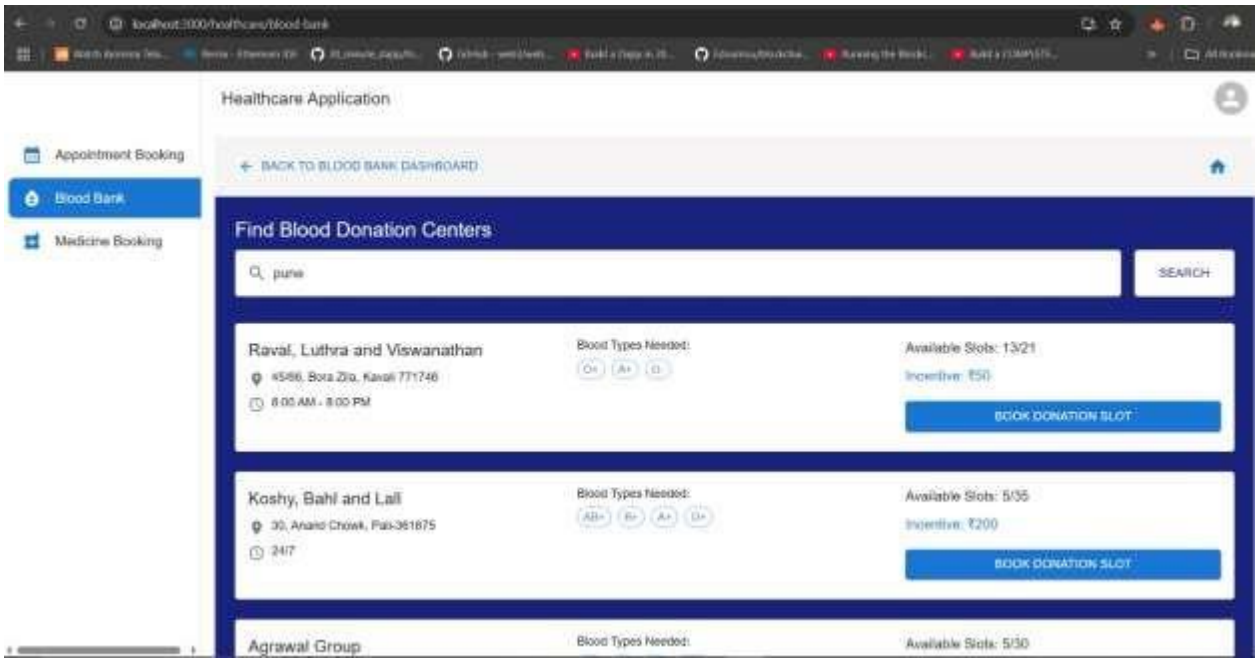
Screen 17 :Subscription Management



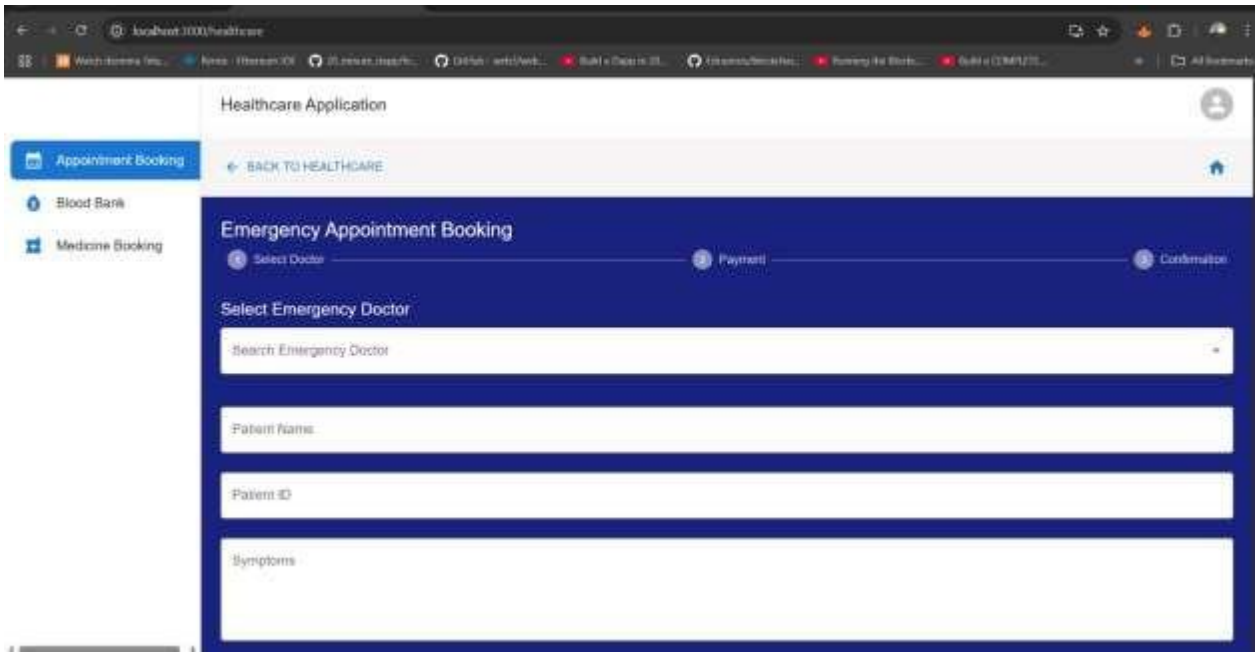
Screen 18: Energy Trading management



Screen 19 : Healthcare Application



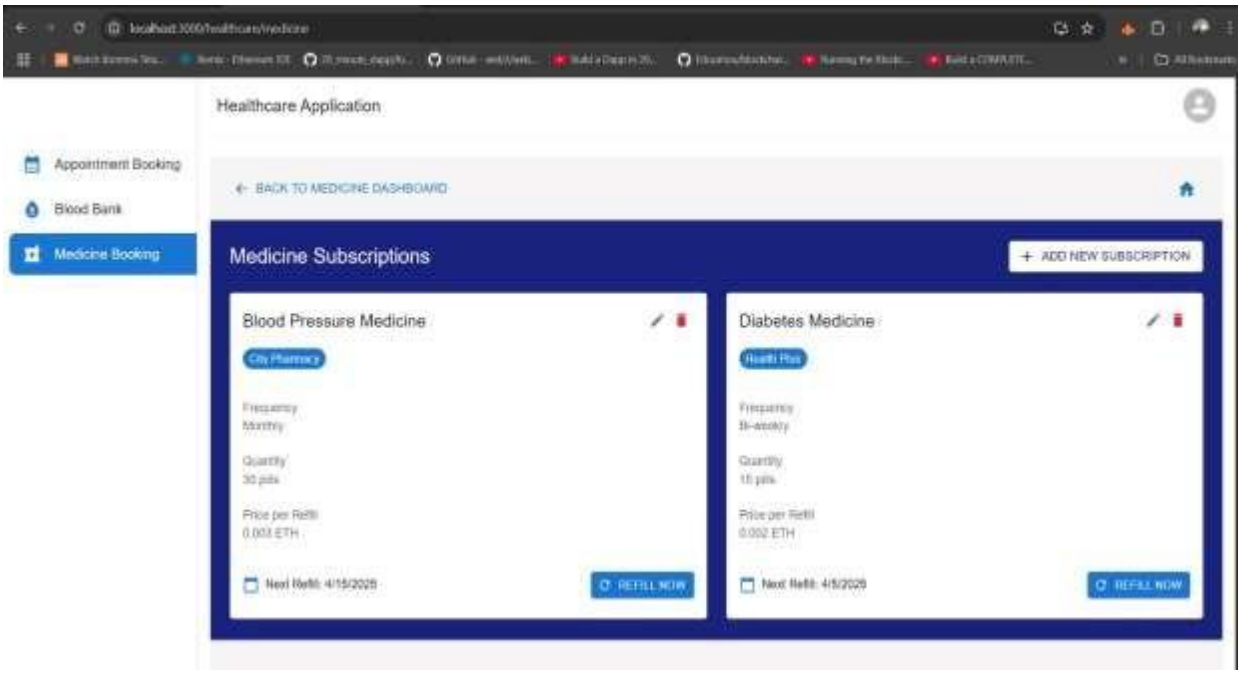
Screen 20: Blood Bank



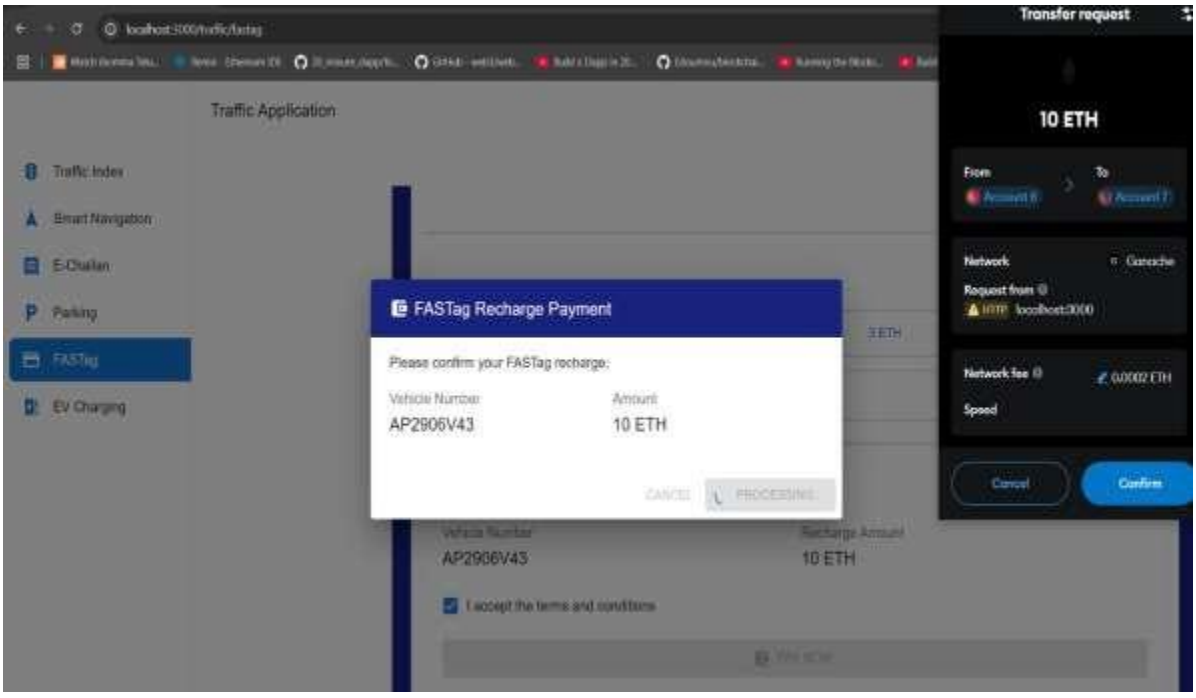
Screen21 :Appointment Booking

Waste Management Dashboard

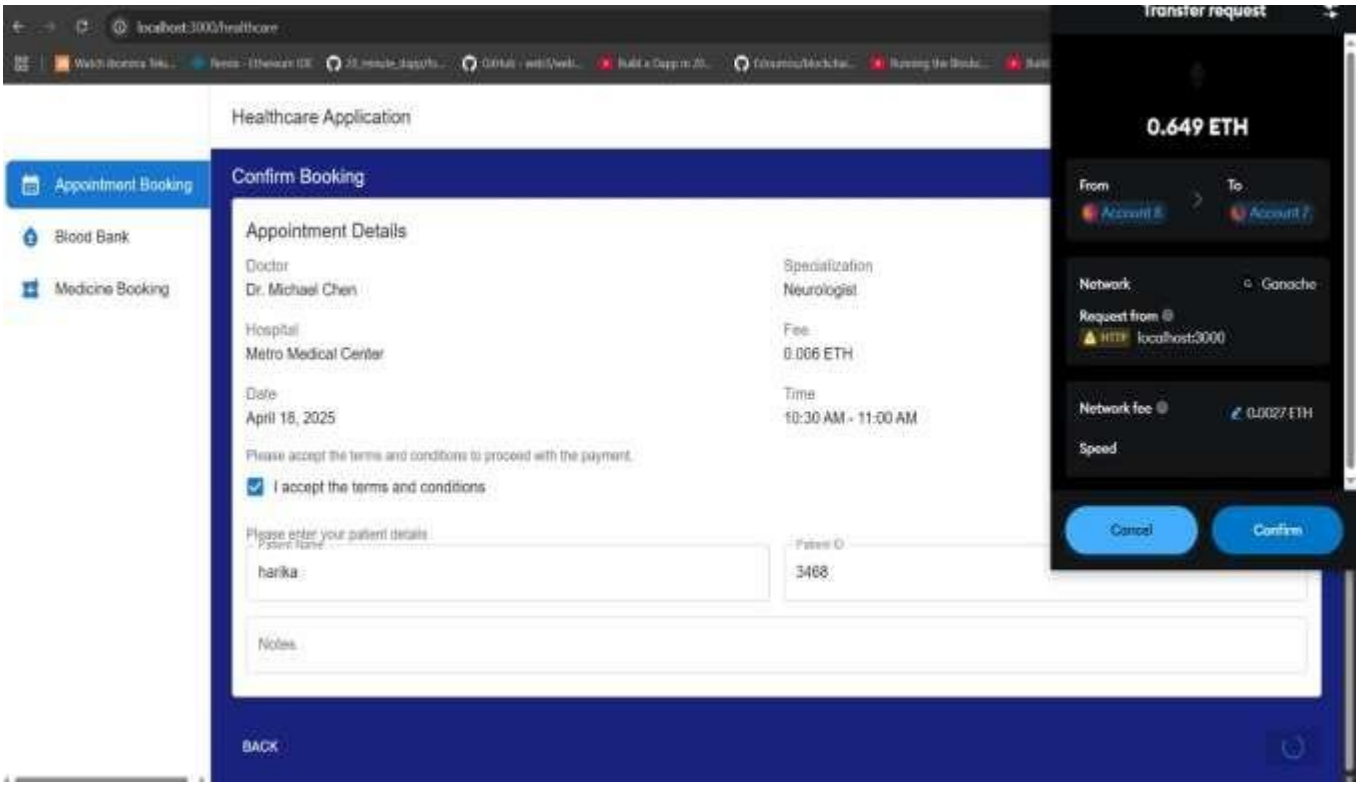
Screen 22 : Waste Management Admin



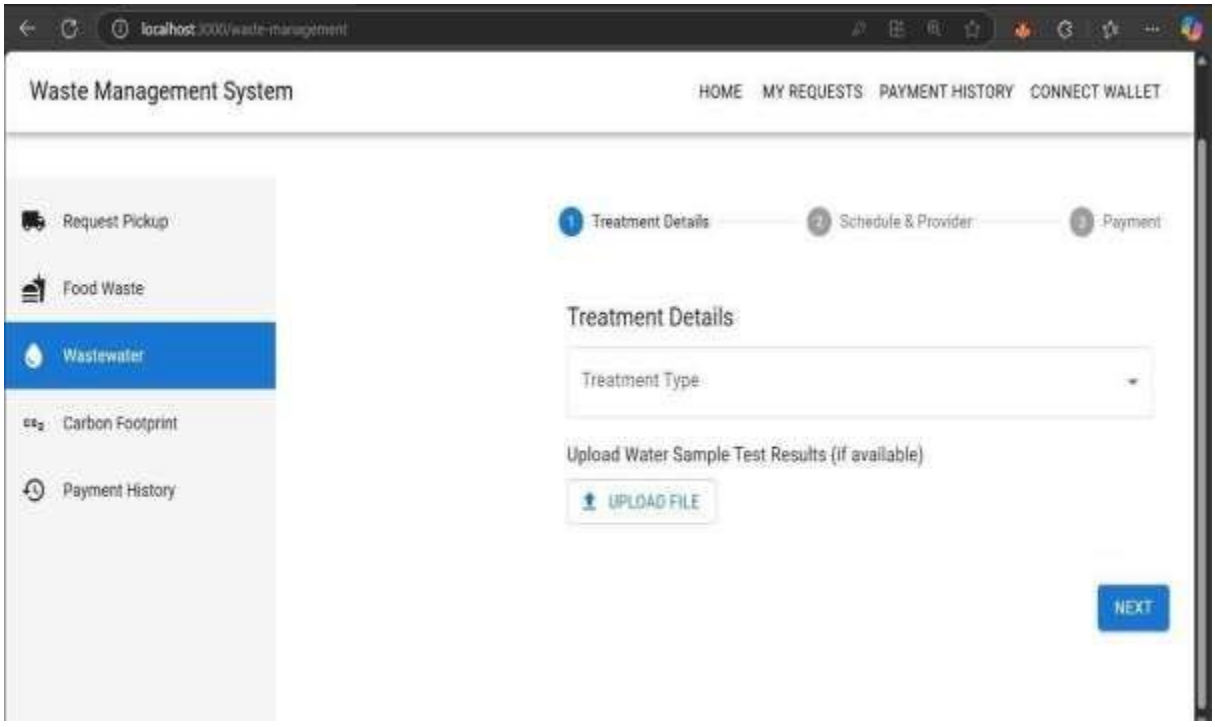
Screen 23 : Medicine Booking



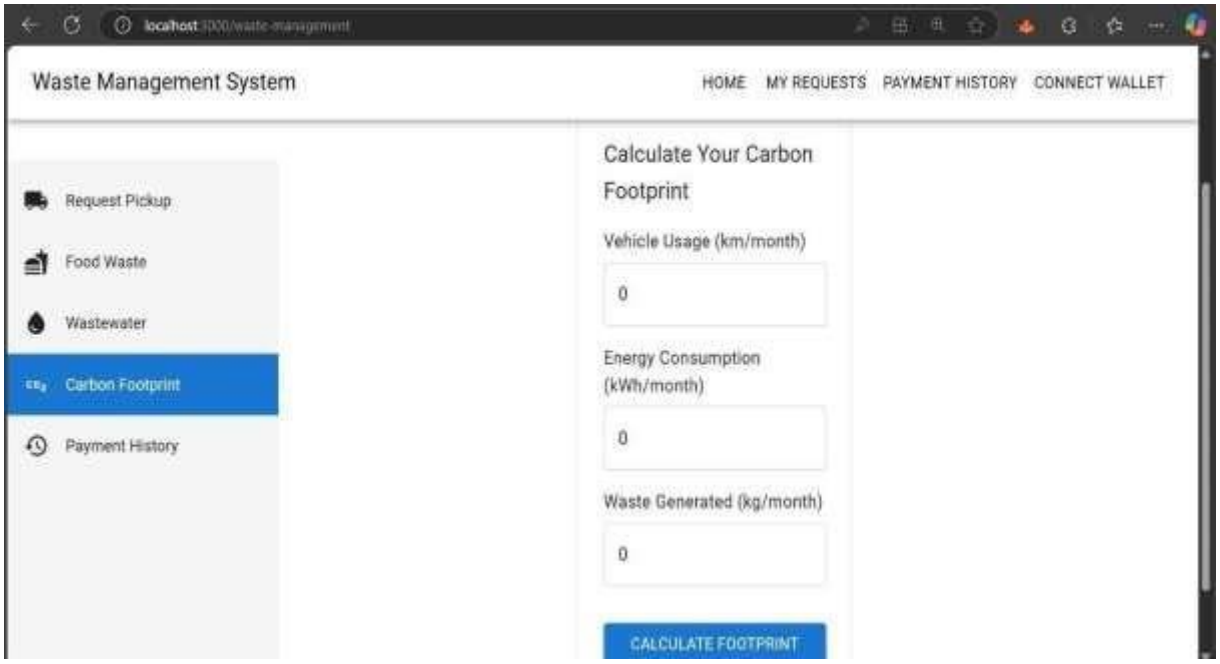
Screen 24: FASTag Recharge Payment



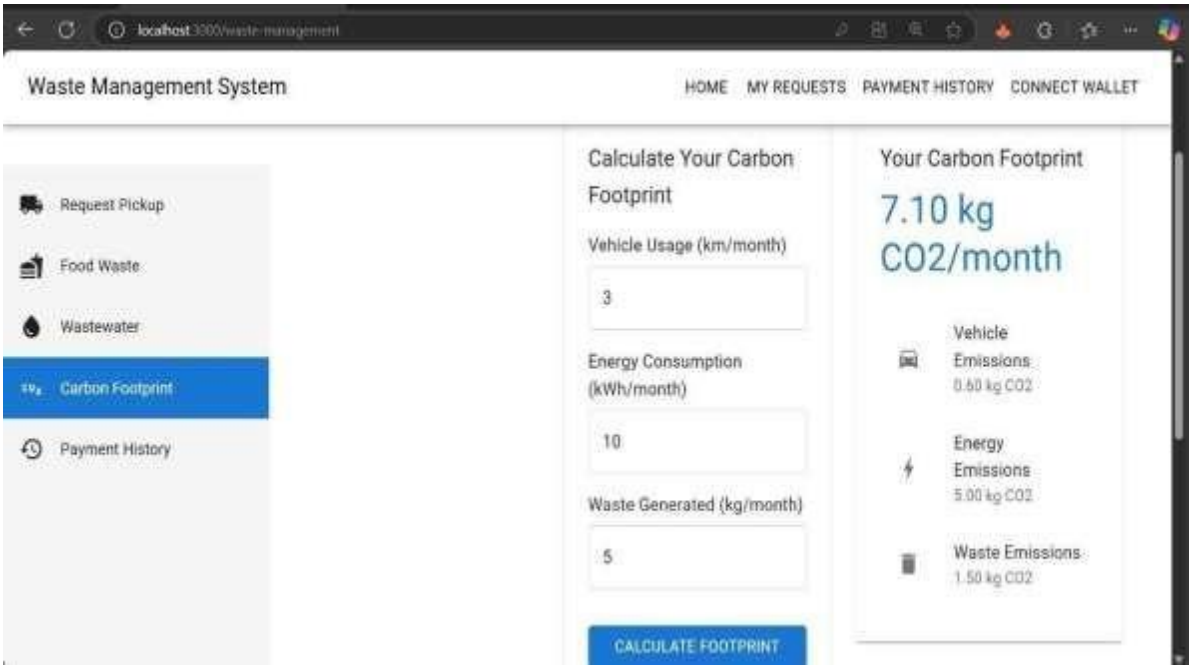
Screen 25 :Transaction Requests



Screen 26 : Waste Water

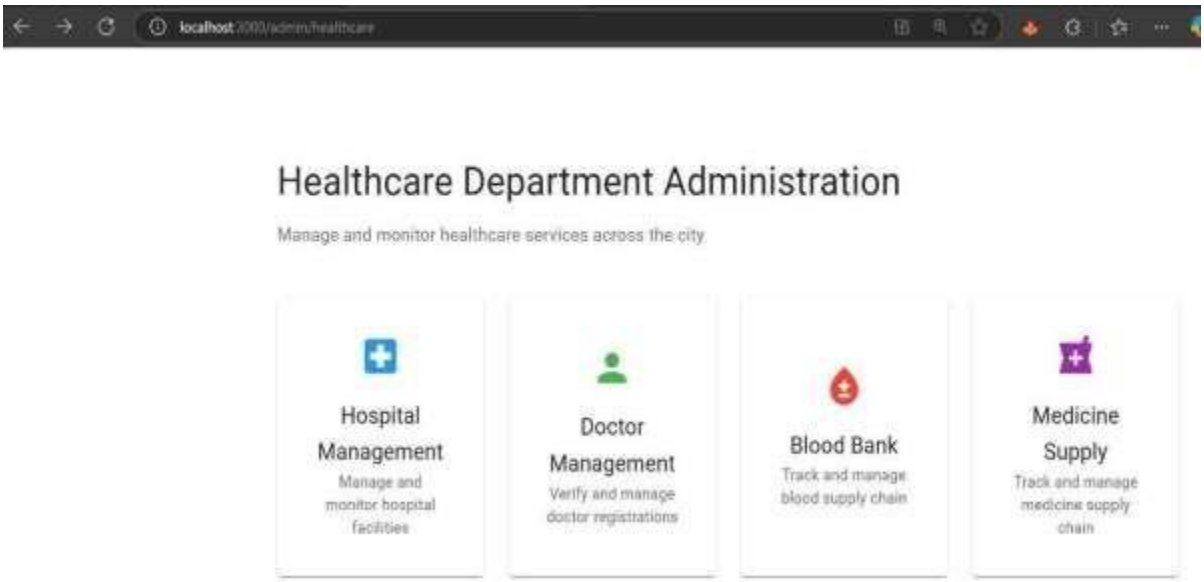


Screen 27 : carbon Footprint

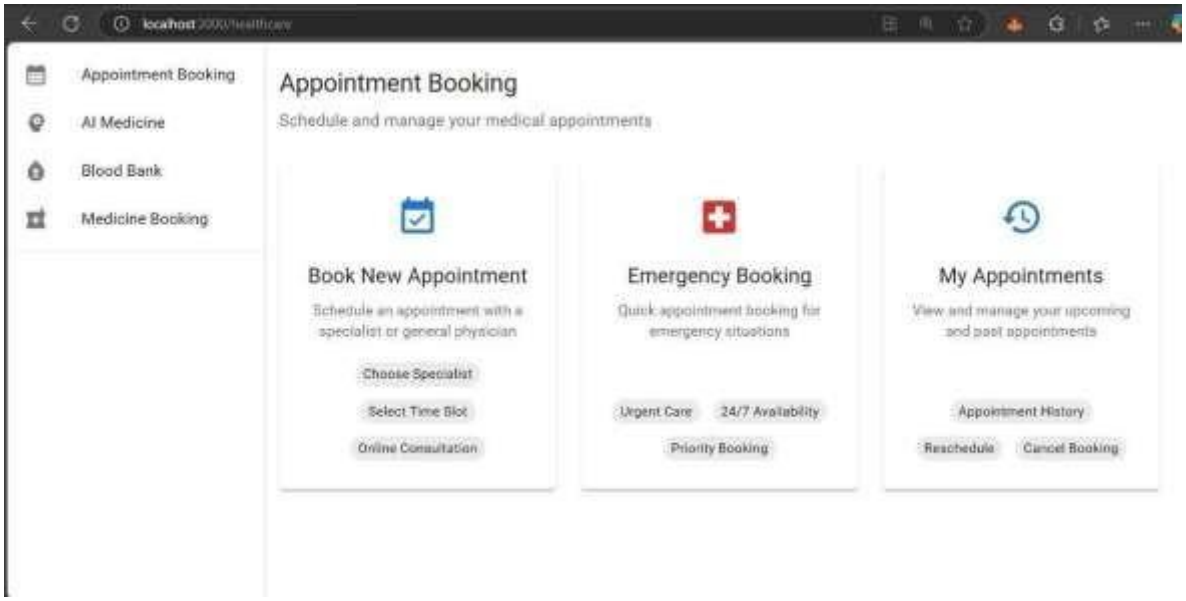


Screen 28 : Caluculate Carboon Footprint

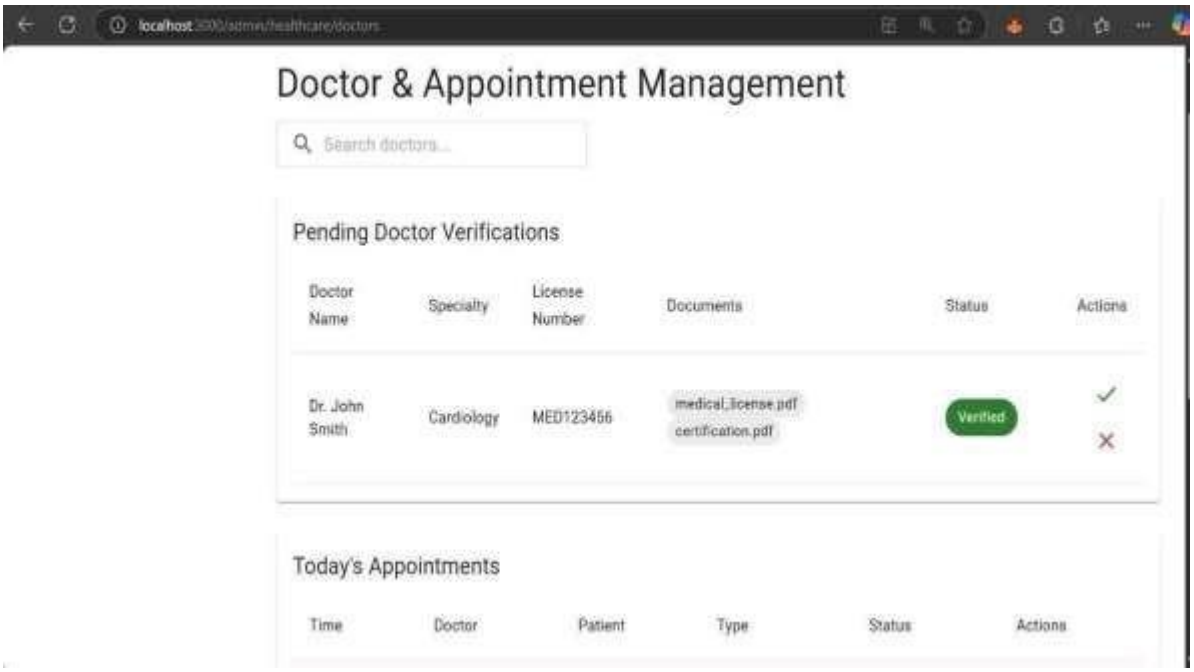
Healthcare Sevice DashBoard



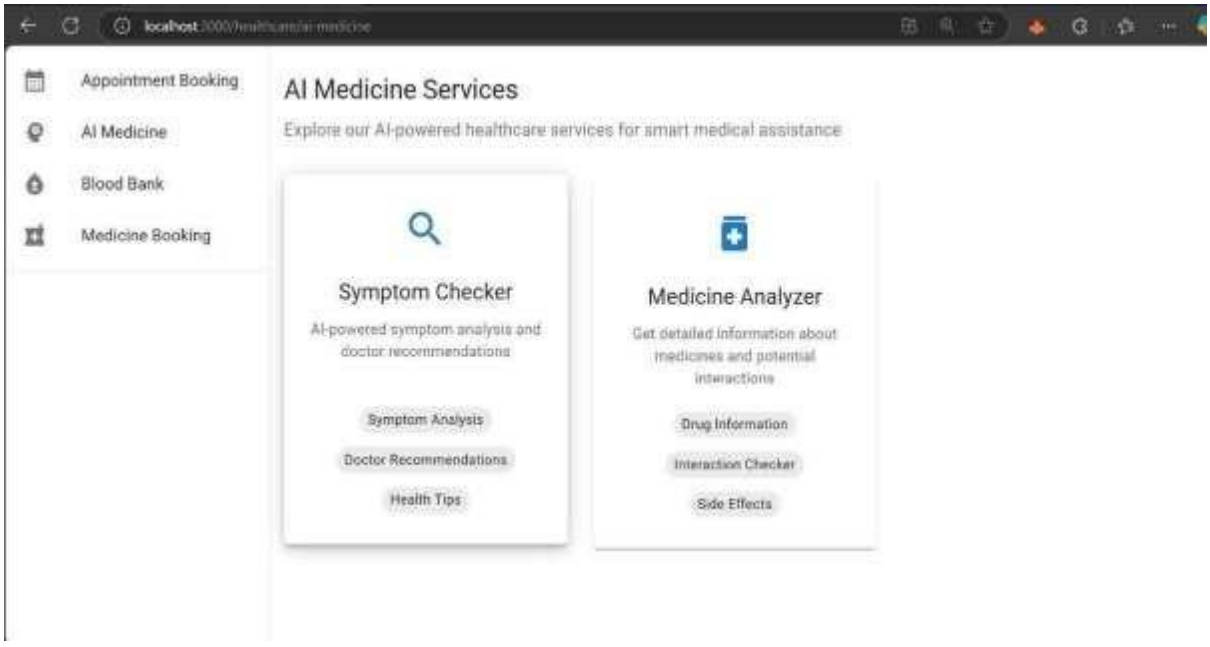
Screen 29: Healthcare Department Adminstration



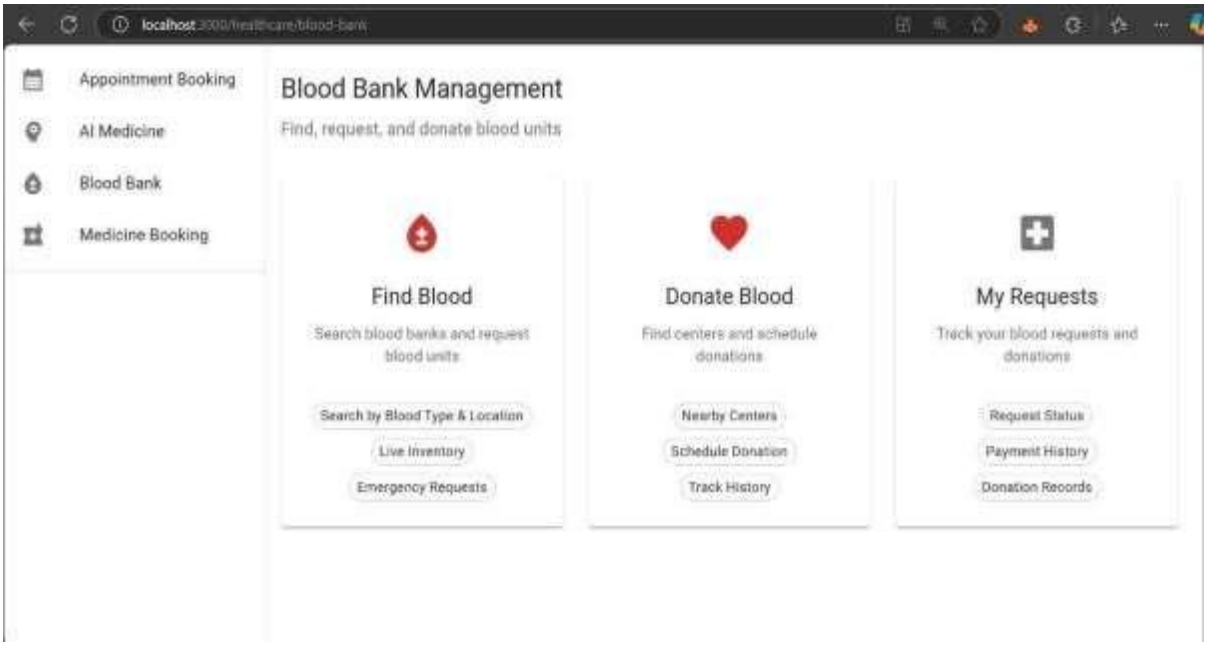
Screen 30 : Appointment Booking



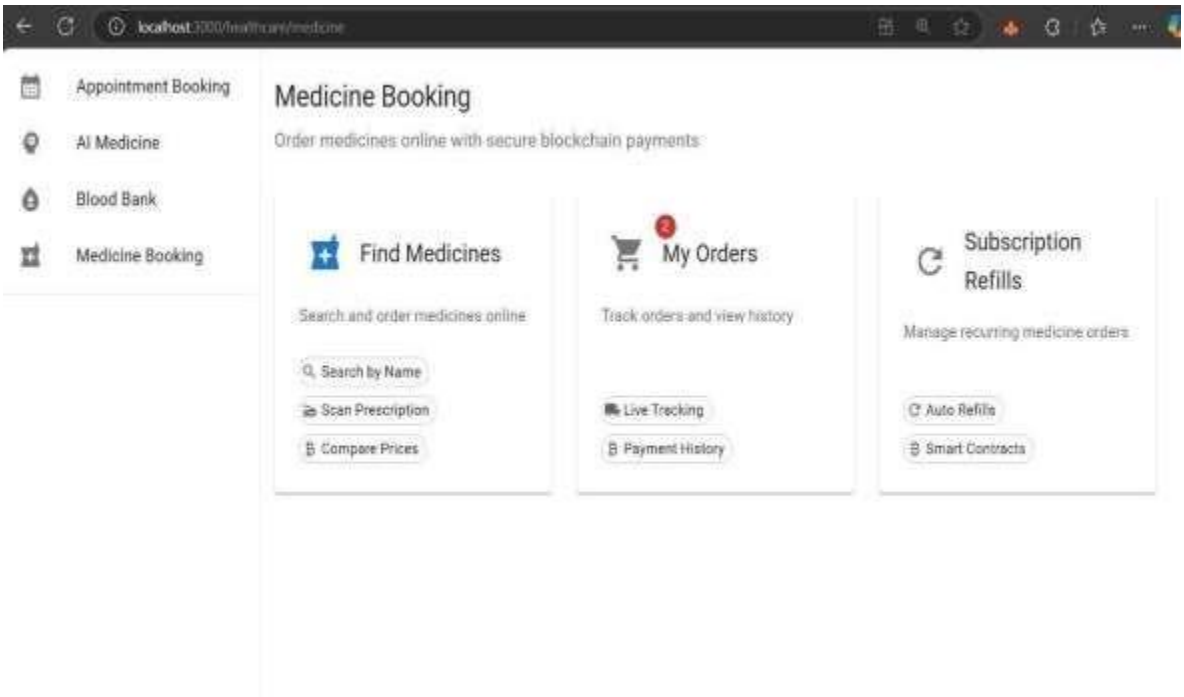
Screen31: Doctor & Appointment Managemet



Screen 32: AI Medicine Services



Screen 33: Blood Bank Management



Screen 34 : Medicine Booking

6 TESTING AND VALIDATION

6.2 INTRODUCTION

Testing and validation are critical components in the software development lifecycle. For the Smart City Portal, which integrates various services such as traffic management, energy monitoring, healthcare, waste management, and payment systems, rigorous testing ensures that

- Each module functions as expected under different conditions.
- The integration between services is seamless and error-free.
- The system is secure, responsive, and user-friendly.
- User roles (Authority and Citizen) are properly managed and restricted according to permissions.

The testing process involves multiple levels including Unit Testing, Integration Testing, System Testing, and User Acceptance Testing (UAT). It also includes Validation techniques to confirm that the final system satisfies all user requirements.

The outcome of this phase determines the system's readiness for deployment in a real-world environment.

6.1.1 Scope

The scope of testing for the Smart City Portal includes functional modules such as **Authority Services**, **Citizen Services**, and sub-systems like **Traffic Management**, **Healthcare**, **Energy**, **Waste Management**, **Payments**, and **Service Status Monitoring**. Testing covers both frontend and backend components to ensure data consistency, role-based access, seamless navigation, accurate computations, secure transactions, and reliable communication between modules.

6.1.2 Defects and Failures

Defects are errors found during development or testing, and failures are the result of those defects occurring during execution. Some examples include

- Incorrect role permissions (e.g., a citizen accessing authority-only features)
- Broken links or non-responsive pages

- Incorrect bill calculations in energy management
- Failure to schedule waste collection
- Invalid or missing payment receipt generation

All identified defects are logged, prioritized, and resolved before the final deployment.

6.1.3 Compatibility

The system is tested for

- Browser compatibility Chrome, Firefox, Edge, Safari
- Device compatibility Desktops, Tablets, and Mobile Devices
- Operating systems Windows, macOS, Android, and iOS
- API compatibility Ensuring proper communication between modules and third-party services

The goal is to ensure that the Smart City Portal works uniformly across different platforms and devices.

6.1.4 Input Combinations and Preconditions

Testing considers all possible combinations of valid and invalid inputs for each module. Examples

- Valid login credentials vs. invalid/missing ones
- Correct vs. incorrect phone number formats during citizen registration
- Valid vs. invalid payment amounts
- Scheduling waste pickup in past vs. future dates

Preconditions such as login, registration, and existing service entries are established before module testing begins.

6.1.5 Static vs. Dynamic Testing

- **Static Testing** involves reviewing the code, documentation, and design diagrams without executing the program. This helps identify syntax errors, logical flaws, and design inconsistencies.
- **Dynamic Testing** involves executing the code and verifying outputs against expected results. This includes testing user interactions, API responses, and database updates in real-time.

Both testing types are applied to maximize coverage and quality.

6.1.6 Types of Testing

(a) Unit Testing

Unit testing is performed on individual components or functions to ensure they work as expected in isolation. For example

- Testing the login() function of the User class
- Testing book Appointment() method in HealthcareManagement
- Testing generate Receipt() in the Payment class

Tools Used : JUnit (for Java) Node.js etc.

Each function is tested with multiple input cases to ensure robustness, boundary condition handling, and error reporting.

6. TESTING AND VALIDATION

6.1.6 Types of Testing (continued)

(i) Black Box Testing

Black box testing focuses on input-output behavior without knowledge of internal code. It is used to

- Validate service request forms (e.g., waste collection, appointment booking)
- Test payment processing by entering valid/invalid data
- Ensure successful login/logout for both authority and citizen roles

(ii) White Box Testing

White box testing examines the internal logic and structure of code. Developers

- Test individual methods like getTrafficUpdates() or processPayment()
- Check loops, conditional branches, and data flow
- Ensure logical correctness of algorithms and transactions

(iii) Gray Box Testing

Gray box testing combines internal knowledge (like white box) with external testing (like black box). It is useful when

- Validating data flow between modules (e.g., from Citizen to Payment and ServiceStatus)
- Testing API interactions with known internal database behavior

(b) Integration Testing

Integration testing ensures that individual modules work together correctly. Examples include

- Ensuring that makePayment() updates Payment and ServiceStatus
- Verifying that appointment booking correctly interacts with HealthcareManagement and stores records
- Confirming that service status changes reflect across Citizen and Authority dashboards

Integration testing checks for

- Data inconsistencies
- API or interface failures
- Module-to-module communication

6.1.7 Software Verification and Validation

- **Verification** ensures that the system is built correctly according to the design. Activities include inspections, reviews, and walkthroughs.
- **Validation** ensures that the right system is built as per user requirements. This involves executing the system and confirming it behaves as expected.

The Smart City Portal was verified using UML class and use case diagrams, and validated through rigorous testing across all modules.

6.3 DESIGN OF TEST CASES AND SCENARIOS

Sample test cases were designed for major functionalities

Test Case	Description	Input	Expected Output	Status
TC01	Citizen Registration	Name, Email, Phone	Registration Successful	Pass
TC02	Authority Login	Valid Credentials	Dashboard Loaded	Pass
TC03	Book Appointment	Facility Name, Time	Confirmation Message	Pass
TC04	Payment Processing	Amount, Card Info	Receipt Generated	Pass
TC05	View Service Status	Service ID	Status Displayed	Pass

These test scenarios cover edge cases, valid data, invalid entries, and null inputs to ensure system reliability.

6.4 VALIDATION TESTING

Validation testing was performed to confirm the entire system meets its goals. Realistic user scenarios were executed, such as

- A citizen registering, making a payment, and viewing service status
- An authority logging in and monitoring urban data
- Real-time updates reflected across services
- Outcome system successfully passed validation and is ready for deployment.

6.5 CONCLUSION

The Smart City Portal was thoroughly tested for usability, performance, and correctness. Each module passed both functional and integration testing. The user experience was validated to ensure accessibility for both authority and citizen users.

The project fulfills its goal of providing a unified digital interface for managing and accessing urban services efficiently.

FUTURE SCOPE

The Smart City Portal holds significant potential for future enhancements to meet the growing demands of urbanization and technological advancement. One promising direction is the integration of Artificial Intelligence (AI) to enable predictive analytics for services like traffic congestion management, energy usage forecasting, and healthcare resource planning. Additionally, incorporating Blockchain technology can enhance transparency and security, particularly in managing service requests, digital identities, and financial transactions. To increase accessibility, a dedicated mobile application can be developed, allowing citizens to interact with services on the go. Moreover, features such as voice-based assistants and chatbots can be added to provide a more intuitive and user-friendly interface. The system can also be extended to support multi-language options, ensuring inclusivity for users from diverse linguistic backgrounds. These advancements will make the portal more intelligent, responsive, and scalable, paving the way for smarter, citizen-centric urban governance.

7 CONCLUSION

The **Smart City Portal** project has been conceptualized, designed, and developed to create a unified digital ecosystem that integrates and streamlines essential urban services for both administrators and citizens. With a focus on accessibility, scalability, and reliability, the platform effectively bridges the gap between government services and public engagement through a centralized, user-centric portal.This project addresses core functionalities of a modern smart city—such as traffic control, healthcare access, utility monitoring, waste management, service ticketing, and digital payments—under one platform. Each module was crafted using a modular architecture to allow for easy maintenance and future scalability. The portal's design incorporates role-based access control, real-time data interaction, and a responsive interface, ensuring that it caters to the diverse needs of both citizens and city administrators.

Thorough testing and validation were conducted to evaluate functionality, performance, integration, and security. Each component underwent unit testing, integration testing, and validation testing, ensuring smooth user experience and data accuracy. The system also passed verification benchmarks for both functional and non-functional requirements.

From a development perspective, the project demonstrates a strong command of software engineering principles such as requirement analysis, UML modeling, database structuring, frontend-backend separation, and full-stack deployment. It showcases a clear understanding of system design, secure authentication mechanisms, and data flow within complex applications.

In conclusion, the Smart City Portal stands as a robust and future-ready platform for digital urban governance. It not only addresses the current requirements of urban management but also lays the groundwork for upcoming innovations. With the integration of technologies such as blockchain for secure, tamper-proof service logs, AI for predictive analytics and citizen assistance, and mobile-first architecture for greater reach, the solution can evolve into a comprehensive digital hub for any smart city. The adaptability and extensibility of the system ensure that it remains relevant and impactful as cities continue to grow and digitize their services.

8. References

[1] P. Muralidhar rao ., & Deebak, B. D. (2023). Security and privacy issues in smart cities/industries: Technologies, applications, and challenges. Journal of Ambient Intelligence and Humanized Computing, https://doi.org/10.1007/s12652-022-03707-1

[2] Sharma, P. K., Moon, S. Y., & Park, J. H. (2018). Blockchain based hybrid architecture for smart cities. Future Generation Computer Systems, 86, 650–655. https //doi.org/10.1016/j.future.2018.03.066

[3] Biswas, K., & Muthukkumarasamy, V. (2016). Securing smart cities using blockchain. IEEE HPCC/SmartCity/DSS, 1392–1393.
https //doi.org/10.1109/HPCC-SmartCity-DSS.2016.0198

[4] Ali, O., Qadir, J., ur Rasool, R., & Sathiaseelan, A. (2019). Blockchain-based smart cities A comprehensive survey. IEEE Access, 7, 128937–128951. https //doi.org/10.1109/ACCESS.2019.2938861

[5] Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). Where is current research on blockchain technology A systematic review. PLOS ONE, 11(10), e0163477. https //doi.org/10.1371/journal.pone.0163477

[6] Zhang, Y., Deng, R. H., & Liu, Y. (2021). Blockchain-based secure data sharing system for smart cities. IEEE Internet of Things Journal, 8(5), 2761–2772. https //doi.org/10.1109/JIOT.2020.3009436

[7] Kshetri, N. (2018). 1 Blockchain's roles in meeting key supply chain management objectives. International Journal of Information Management, 39, 80–89. https //doi.org/10.1016/j.ijinfomgt.2017.12.005

[8] Shen, M., Zhang, Y., & Li, Y. (2020). Blockchain for secure healthcare systems A vision, challenges and applications. Computer Communications, 150, 242–250. https //doi.org/10.1016/j.comcom.2019.11.006

[9] Salah, K., Rehman, M. H., Nizamuddin, N., & Al-Fuqaha, A. (2019). Blockchain for AI Review and open research challenges. IEEE Access, 7, 10127–10149.
(used for understanding decentralized system benefits, not AI) https //doi.org/10.1109/ACCESS.2019.2890507

[10] Treiblmaier, H. (2018). The impact of blockchain on e-government A research agenda. Digital Policy, Regulation and Governance, 20(4), 322–333.

https //doi.org/10.1108/DPRG-10-2017-0051

[11] Zyskind, G., Nathan, O., & Pentland, A. (2015). Decentralizing privacy Using blockchain to protect personal data. IEEE Security and Privacy Workshops, 180–184. https //doi.org/10.1109/SPW.2015.27

[12] Xie, J., Tang, J., Huang, T., & Xie, H. (2021). A survey of blockchain technologies for smart city applications. Computer Science Review, 39, 100360. https //doi.org/10.1016/j.cosrev.2020.100360

[13] Kouhizadeh, M., & Sarkis, J. (2018). Blockchain practices, potentials, and perspectives in greening supply chains. Sustainability, 10(10), 3652.
https //doi.org/10.3390/su10103652

[14] Gourisetti, S. N. G., Mylrea, M., & Patangia, H. (2018). Blockchain for smart cities Challenges and opportunities. 2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE), 94–99.
https //doi.org/10.1109/SEGE.2018.8499528

[15] Ministry of Housing and Urban Affairs, Government of India. (2015). Smart Cities Mission Guidelines**.**
https //smartcities.gov.in