EX NO:1	WRITE THE COMPLETE PROBLEM STATEMENT
DATE	

AIM:

To prepare PROBLEM STATEMENT for any project.

ALGORITHM:

1. The problem statement is the initial starting point for a project.
2. A problem statement describes what needs to be done without describing how.
3. It is basically a one-to-three-page statement that everyone on theproject agrees with that describes what will be done at a high level.
4. The problem statement is intended for a broad audience and shouldbe written in non- technical terms.
5. It helps the non-technical and technical personnel communicate byproviding a description of a problem.
6. It doesn't describe the solution to the problem.

INPUT:

1. The input to requirement engineering is the problem statement prepared by customer.
2. It may give an overview of the existing system along with broad expectations from the new system.
3. The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements.
4. Here, requirements are identified with the help of customer and existing system processes.

Problem:

Waste management is a growing concern globally, as improper disposal, lack of community participation, and inefficient collection systems lead to environmental pollution and health hazards. Despite the critical importance of waste management, many municipalities and communities still rely on outdated or manual methods for managing waste collection, recycling, and reporting. This results in issues like delayed waste pickup, poor segregation practices, and a lack of real-time visibility into waste disposal activities. To address these inefficiencies, a digital solution is needed that empowers users, promotes transparency, and fosters better waste management practices.

Background:

With rapid urbanization and increasing waste generation, the need for an effective waste management system has become paramount. Traditional waste management systems often lack integration and rely heavily on manual operations, which can result in errors, inefficiencies, and inadequate resource allocation. Citizens face challenges in reporting issues, accessing collection schedules, and understanding their role in sustainable waste practices. Furthermore, the absence of real-time data prevents authorities from optimizing their waste management strategies. A centralized system that integrates waste collection, issue reporting, and recycling progress tracking can significantly improve waste management efficiency while encouraging community participation.

Relevance:

An efficient waste management system is essential for creating cleaner, healthier, and more sustainable communities. By integrating technology into waste management practices, we can address common challenges such as delayed pickups, improper disposal, and poor recycling rates. A streamlined system benefits both citizens and municipal authorities by providing transparency, reducing inefficiencies, and promoting eco-friendly behaviors. Additionally, real-time data tracking and reporting capabilities empower decision-makers to optimize resource allocation, reduce environmental impact, and meet sustainability goals.

Objectives:

The primary objective of this project is to develop a user-friendly, Streamlit-based Waste Management System that enhances operational efficiency, promotes recycling, and ensures better communication between citizens and waste management authorities. Specific objectives include:

1. **Waste Issue Reporting**
   - Enable citizens to report waste collection issues or request pickups, including the ability to upload images for better clarity.
2. **Collection Schedule Management**
   - Provide users with real-time access to waste collection schedules based on their location and waste type (e.g., organic, recyclable, hazardous).
3. **Recycling Progress Tracking**
   - Allow users to track their recycling contributions and community progress, encouraging sustainable waste disposal practices.
4. **Admin Tools for Request Management**
   - Equip administrators with tools to review, approve, and manage citizen reports and waste collection schedules effectively.
5. **Inventory and Resource Monitoring**
   - Implement real-time monitoring of waste collection resources, ensuring optimal allocation and minimizing delays.
6. **Report Generation**
   - Generate detailed reports on waste collection, issue resolutions, and recycling rates to support data-driven decision-making and compliance

Result:

he problem statement was written successfully by the following the above steps

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE	

INTRODUCTION

1.1 Purpose

This document outlines the requirements for developing a Waste Management System (WMS). The system aims to improve waste collection, scheduling, tracking, and reporting to make waste management more efficient and sustainable.

1.2 Scope

The WMS will be a web and mobile-based application accessible to municipal authorities, waste collectors, and citizens. It will manage waste collection requests, route optimization, billing, and compliance with environmental standards.

1.3 Definitions, Acronyms, and Abbreviations

- WMS: Waste Management System
- Admin: Municipal Authority Administrator
- Collector: Waste collection personnel or service providers
- Citizen: Residents requesting waste collection services

1.4 overview

This document details the functionalities, interfaces, and performance standards of WMS to ensure the system meets the needs of all stakeholders.

2. Overall Description

2.1 Product Perspective

WMS is a centralized platform to streamline waste collection and management. It integrates with IoT sensors and GIS for real-time tracking and scheduling.

2.2 Product Functions

- User Registration: Register citizens, collectors, and admins.
- Waste Collection Scheduling: Citizens can request waste pickups and track their status.
- Route Optimization: Automatic route planning for collectors based on pickup locations.
- Billing and Payment: Manage invoices and payments for waste collection services.
- Reporting and Analytics: Generate reports on waste volume, collection efficiency, and compliance.

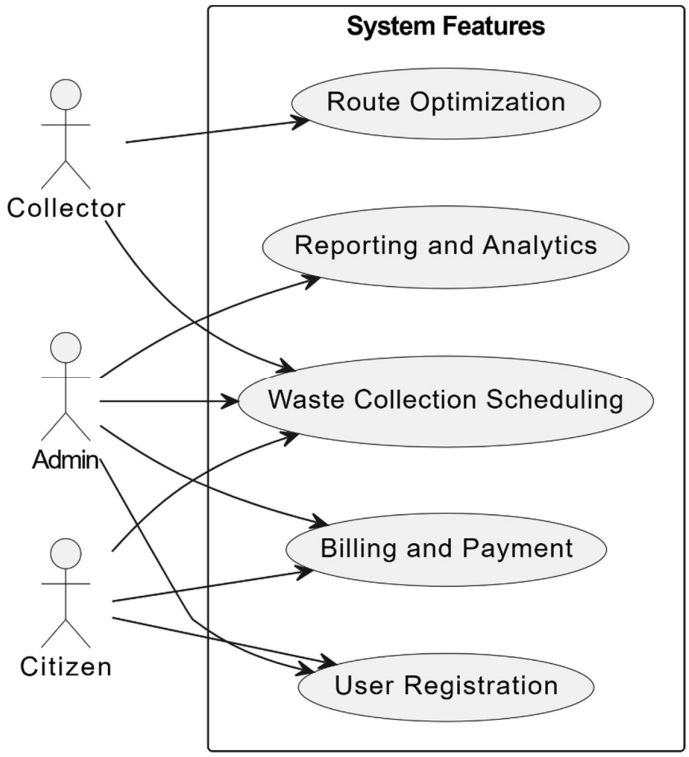
2.3 User Classes and Characteristics

- **Admin**: Oversees operations, approves schedules, and manages complaints.
- **Collector**: Views schedules and optimizes collection routes.
- **Citizen**: Requests services and monitors status.

2.4 Operating Environment

The system is accessible via web and mobile applications.

OUTPUT:


System Features
Route Optimization
Collector
Reporting and Analytics
Waste Collection Scheduling
Admin
Billing and Payment
Citizen
User Registration

2.5 Design and Implementation Constraints

- Data should be secured and accessible only to authorized users.
- The system must handle up to 10,000 concurrent users.

2.6 Assumptions and Dependencies

- Users have internet access.
- The system relies on GPS and IoT integrations for tracking.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Registration

- Citizens, collectors, and admins can register and log in with unique credentials.
- User data is validated to ensure accuracy.

3.1.2 Waste Collection Scheduling

- Citizens can schedule pickups for specific waste categories.
- Notifications are sent for upcoming or missed pickups.

3.1.3 Route Optimization

- **The system suggests optimal routes based on collection points.**
- **Collectors can view and adjust routes in real-time.**

3.1.4 Billing and Payment

- **Citizens receive automated invoices for services.**
- **Payment gateways are integrated for secure transactions.**

3.1.5 Reporting and Analytics

- **Admins can generate reports on collection performance and compliance.**
- **Reports can be filtered by location, date, or waste type.**

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

The system must respond within 3 seconds for all operations.

3.2.2 Security Requirements

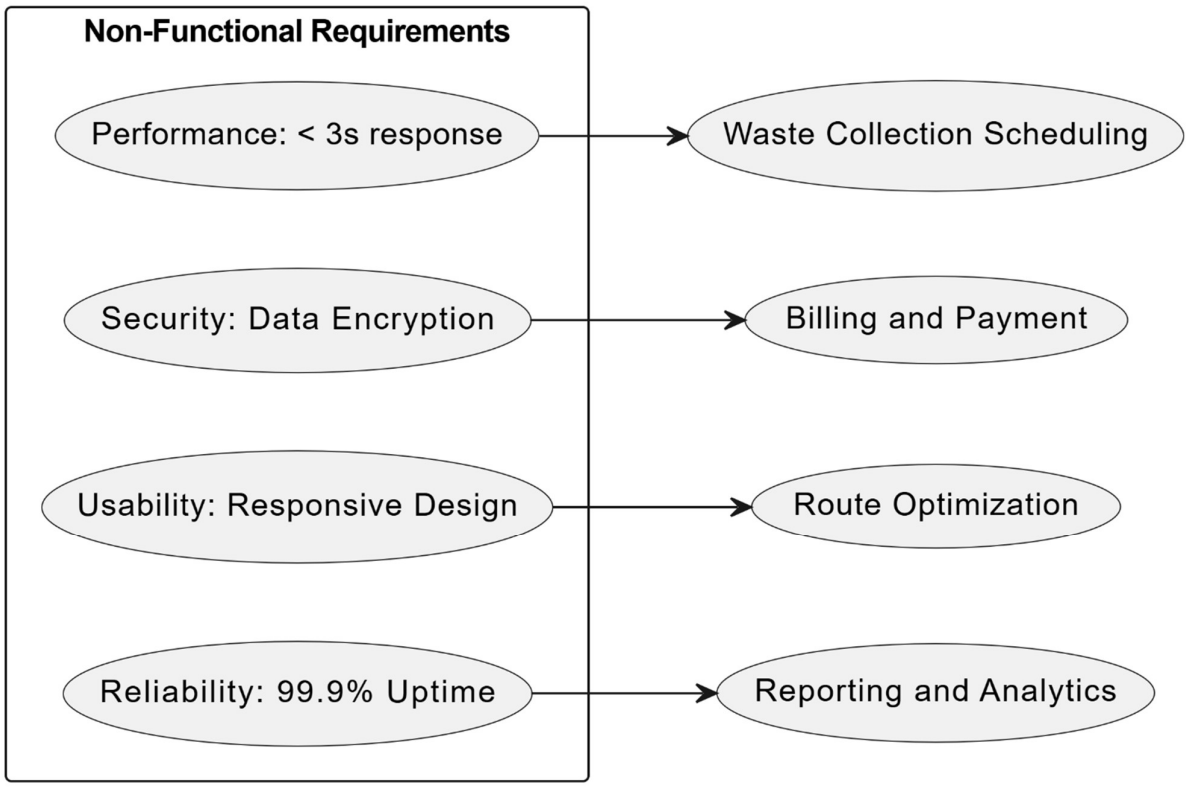
- User authentication is mandatory.
- Data is encrypted during transmission and storage.

3.2.3 Usability Requirements

- The interface should be intuitive and responsive across devices.

3.2.4 Reliability Requirements

- The system should maintain an uptime of 99.9%.
- Critical issues must be resolved within 10 minutes.


Non-Functional Requirements
Performance: < 3s response
Security: Data Encryption
Usability: Responsive Design
Reliability: 99.9% Uptime
Waste Collection Scheduling
Billing and Payment
Route Optimization
Reporting and Analytics

External Interface Requirements

3.3 User Interfaces

- The system will be responsive, adapting to different screen sizes for a smooth user experience.
- Each user type (admin, hospital staff, donor) will have access to functions relevant to their role.

3.4 Hardware Interfaces

- Compatible with standard desktop and mobile devices.

3.5 Software Interfaces

- The system will use an SQL database for secure data storage.
- The system will support SMS and email services for reminders and notifications.

4. Additional Requirements

4.1 User Interfaces

- **A responsive design suitable for web browsers and mobile devices.**

.

4.2 Hardware Interfaces

- **Compatible with IoT sensors for bin status monitoring**

Result:

The software requirements sheet was made successfully

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE	

AIM:

To Draw the Entity Relationship Diagram for Waste management system

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

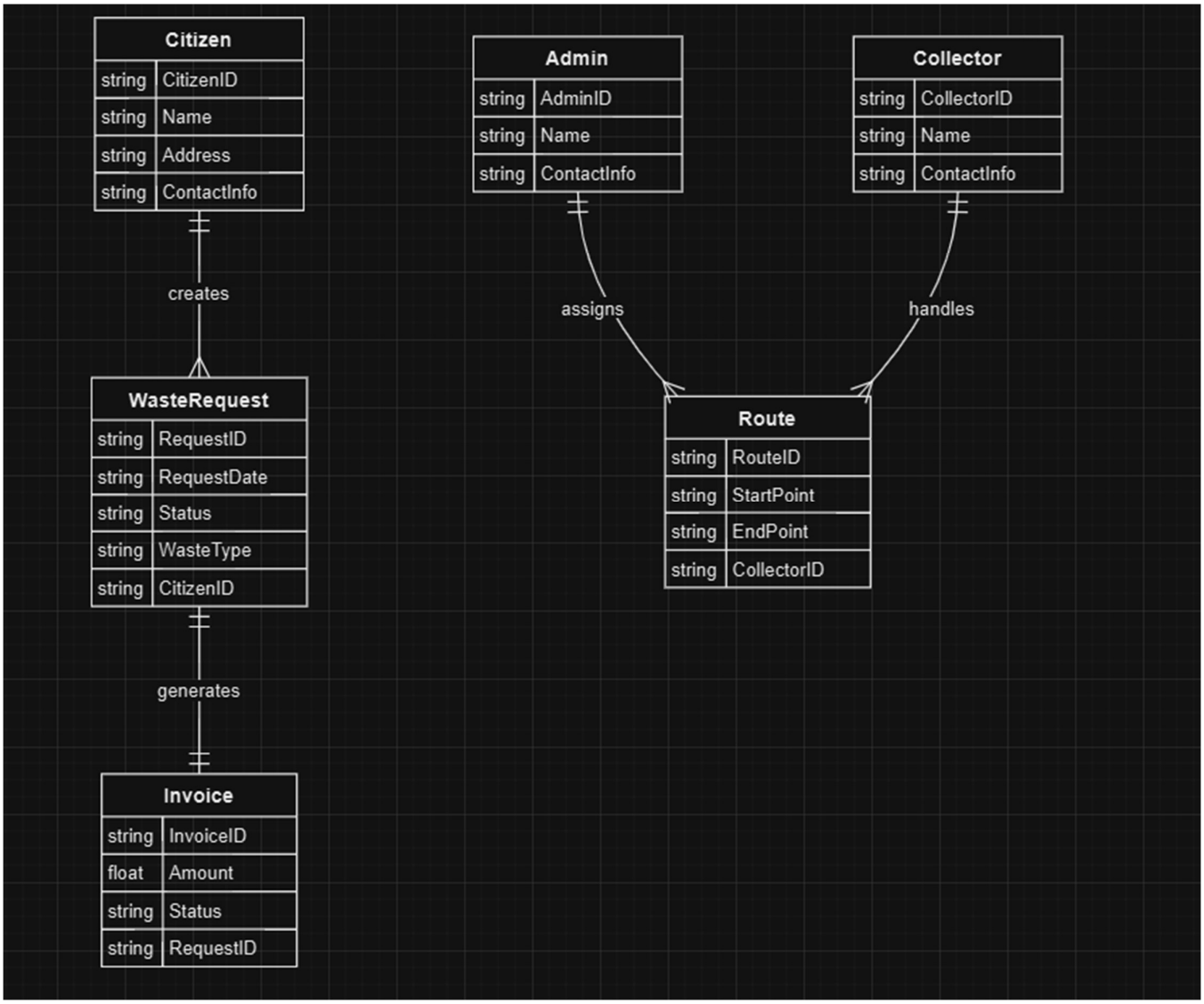
Entity Relationship Matrix

Primary Keys

Attributes

Mapping of Attributes with Entities

OUTPUT:


Citizen
string CitizenID
string Name
string Address
string ContactInfo
creates
WasteRequest
string RequestID
string RequestDate
string Status
string WasteType
string CitizenID
generates
Invoice
string InvoiceID
float Amount
string Status
string RequestID
Admin
string AdminID
string Name
string ContactInfo
assigns
Collector
string CollectorID
string Name
string ContactInfo
handles
Route
string RouteID
string StartPoint
string EndPoint
string CollectorID

Result:

The entity relationship diagram was made successfully

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE	

AIM:

To Draw the Data Flow Diagram for Waste management system and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

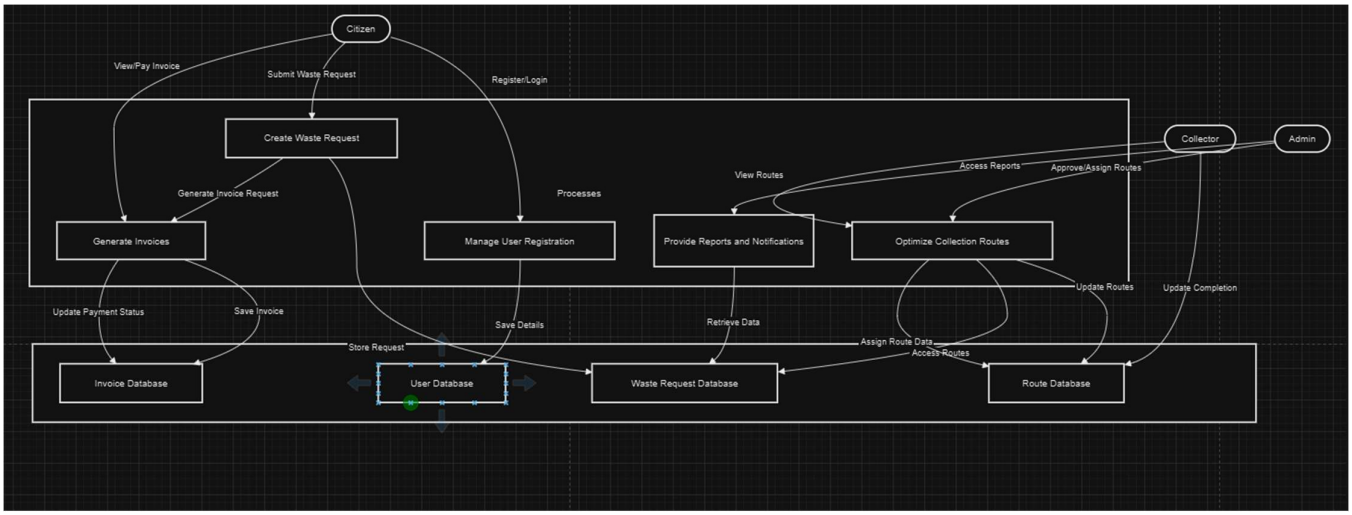
INPUT:

Processes

Datastores

External Entities

OUTPUT:


Citizen
View/Pay Invoice
Submit Waste Request
Register/Login
Create Waste Request
Collector
Admin
Access Reports
Approve/Assign Routes
View Routes
Generate Invoice Request
Processes
Generate Invoices
Manage User Registration
Provide Reports and Notifications
Optimize Collection Routes
Update Routes
Update Completion
Update Payment Status
Save Invoice
Save Details
Retrieve Data
Store Request
Assign Route Data
Access Routes
Invoice Database
User Database
Waste Request Database
Route Database

Result:

The data flow diagram was made successfully

EX NO:5	DRAW USE CASE DIAGRAM
DATE	

AIM:

To Draw the Use Case Diagram for waste management system

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

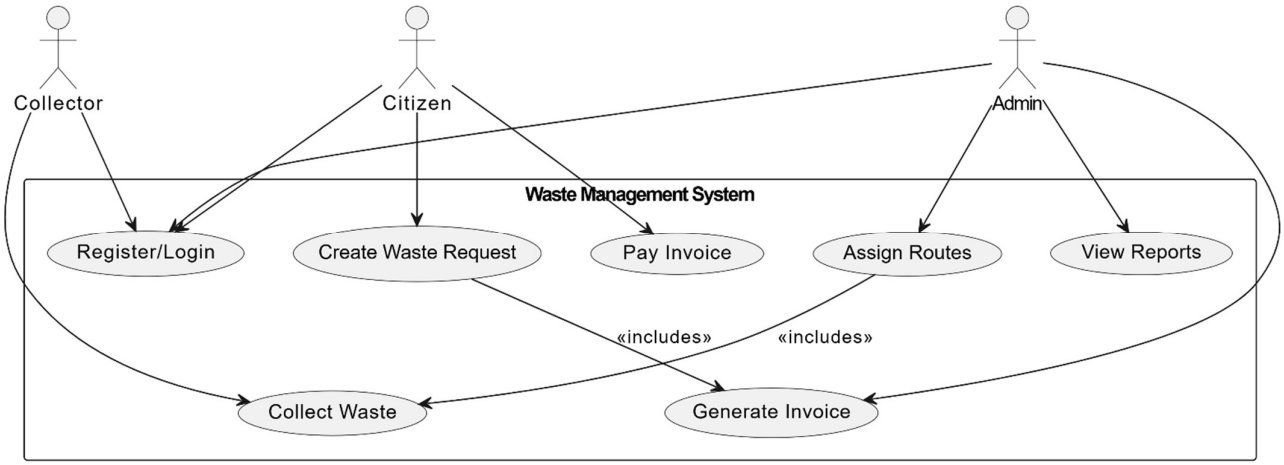
INPUTS:

Actors

Use Cases

Relations

OUTPUT:


Collector
Citizen
Admin
Waste Management System
Register/Login
Create Waste Request
Pay Invoice
Assign Routes
View Reports
«includes»
«includes»
Collect Waste
Generate Invoice

Result:

The use case diagram for the waste management system

EX NO:6	DRAW ACTIVITY DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the activity Diagram for waste management system

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

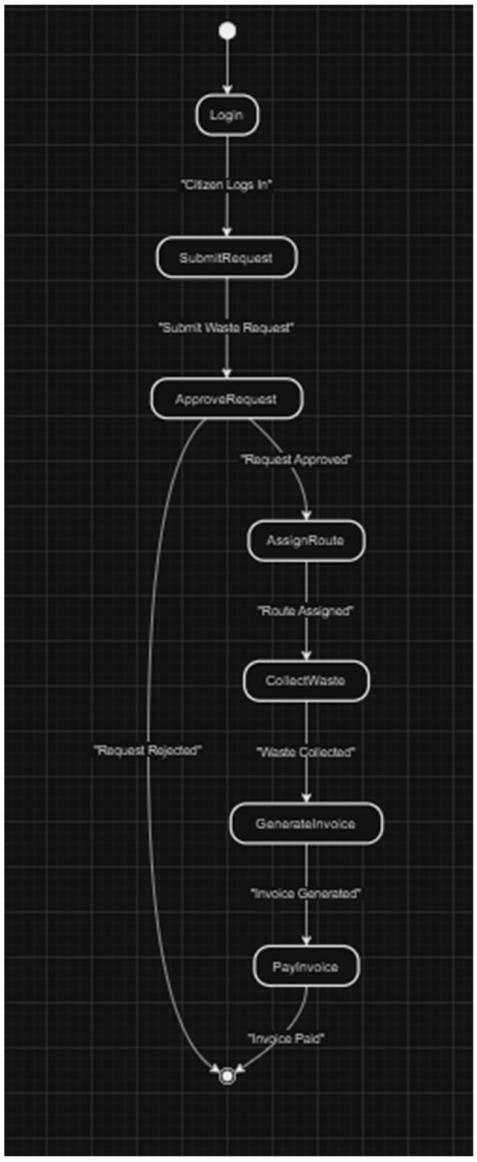
Decision Points

Guards

Parallel Activities

Conditions

OUTPUT:


Login
"Citizen Logs In"
SubmitRequest
"Submit Waste Request"
ApproveRequest
"Request Approved"
AssignRoute
"Route Assigned"
CollectWaste
"Request Rejected"
"Waste Collected"
GenerateInvoice
"Invoice Generated"
PayInvoice
"Invoice Paid"

Result:

The activity diagram for waste management system is drawn successfully

EX NO:7	DRAW STATE CHART DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the State Chart Diagram for Waste management system

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

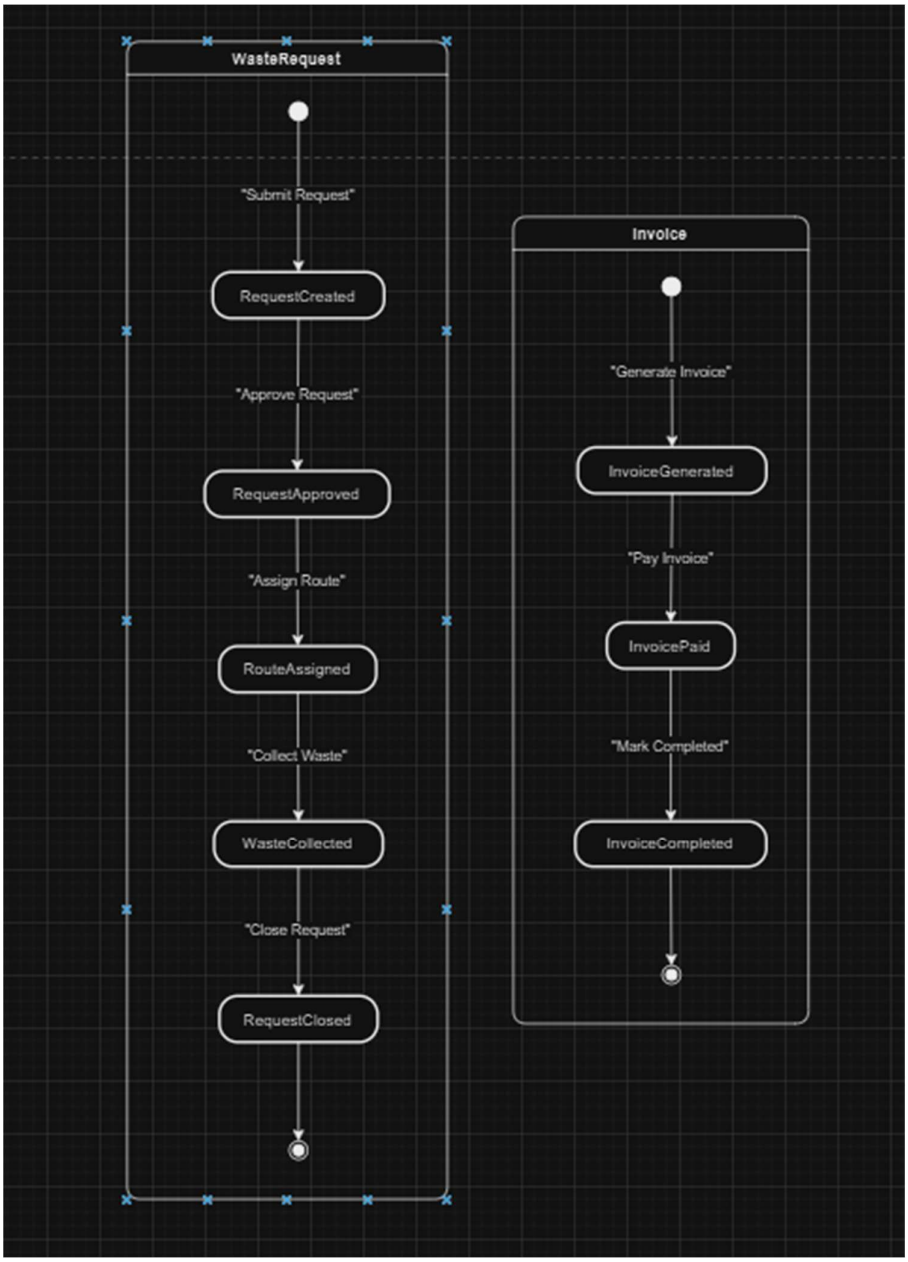
INPUTS:

Objects

States

Events

OUTPUT:


WasteRequest
"Submit Request"
RequestCreated
"Approve Request"
RequestApproved
"Assign Route"
RouteAssigned
"Collect Waste"
WasteCollected
"Close Request"
RequestClosed
Invoice
"Generate Invoice"
InvoiceGenerated
"Pay Invoice"
InvoicePaid
"Mark Completed"
InvoiceCompleted

Result:

The state chart diagram has been created succesfully

EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the Sequence Diagram for Waste management system

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

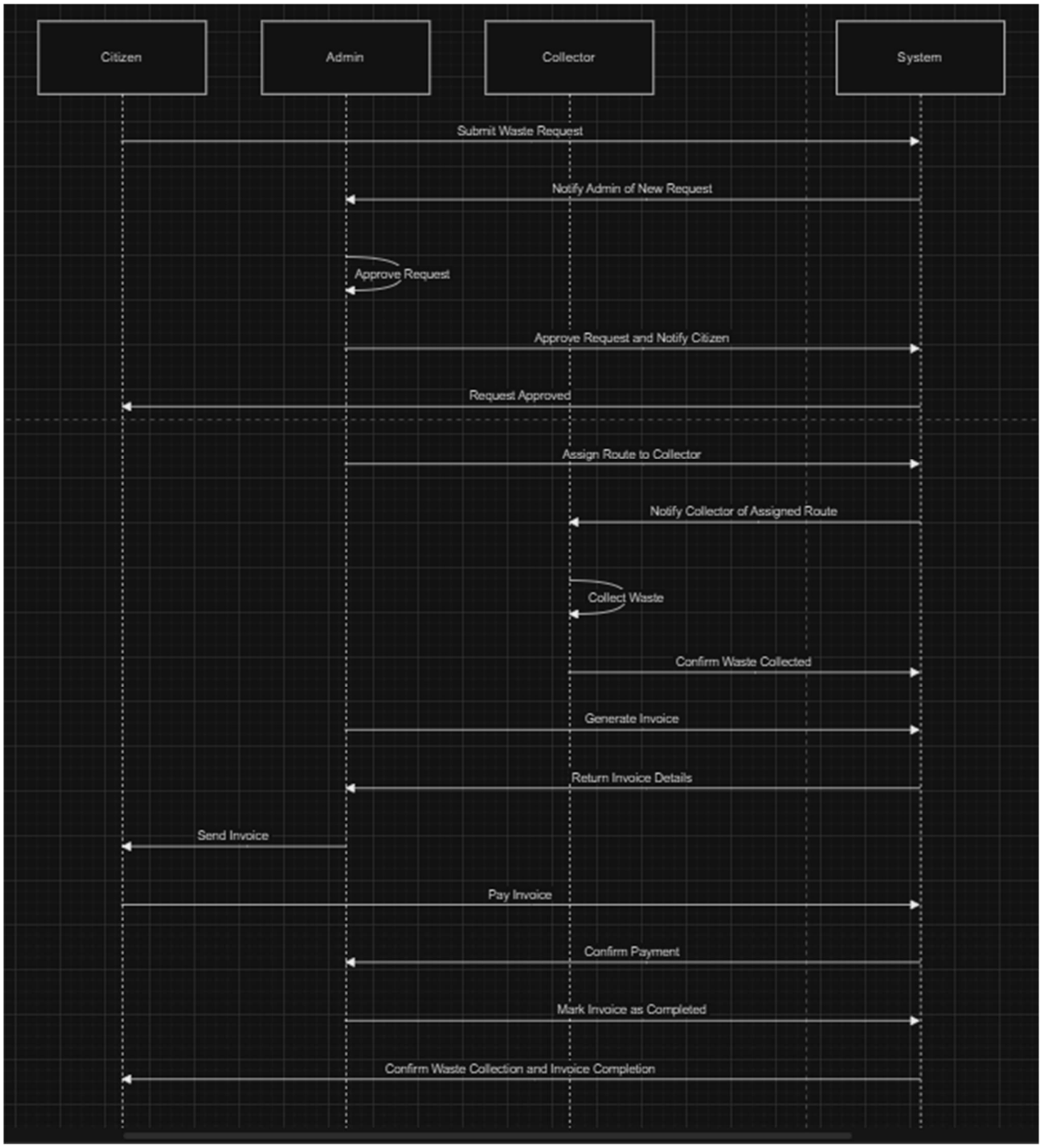
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

OUTPUT:


Citizen
Admin
Collector
System
Submit Waste Request
Notify Admin of New Request
Approve Request
Approve Request and Notify Citizen
Request Approved
Assign Route to Collector
Notify Collector of Assigned Route
Collect Waste
Confirm Waste Collected
Generate Invoice
Return Invoice Details
Send Invoice
Pay Invoice
Confirm Payment
Mark Invoice as Completed
Confirm Waste Collection and Invoice Completion

Result:

The sequence diagram has been created successfully.

EX NO:9	DRAW COLLABORATION DIAGRAM OF ALL USE CASES
DATE	

AIM:

To Draw the Collaboration Diagram for waste management system

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

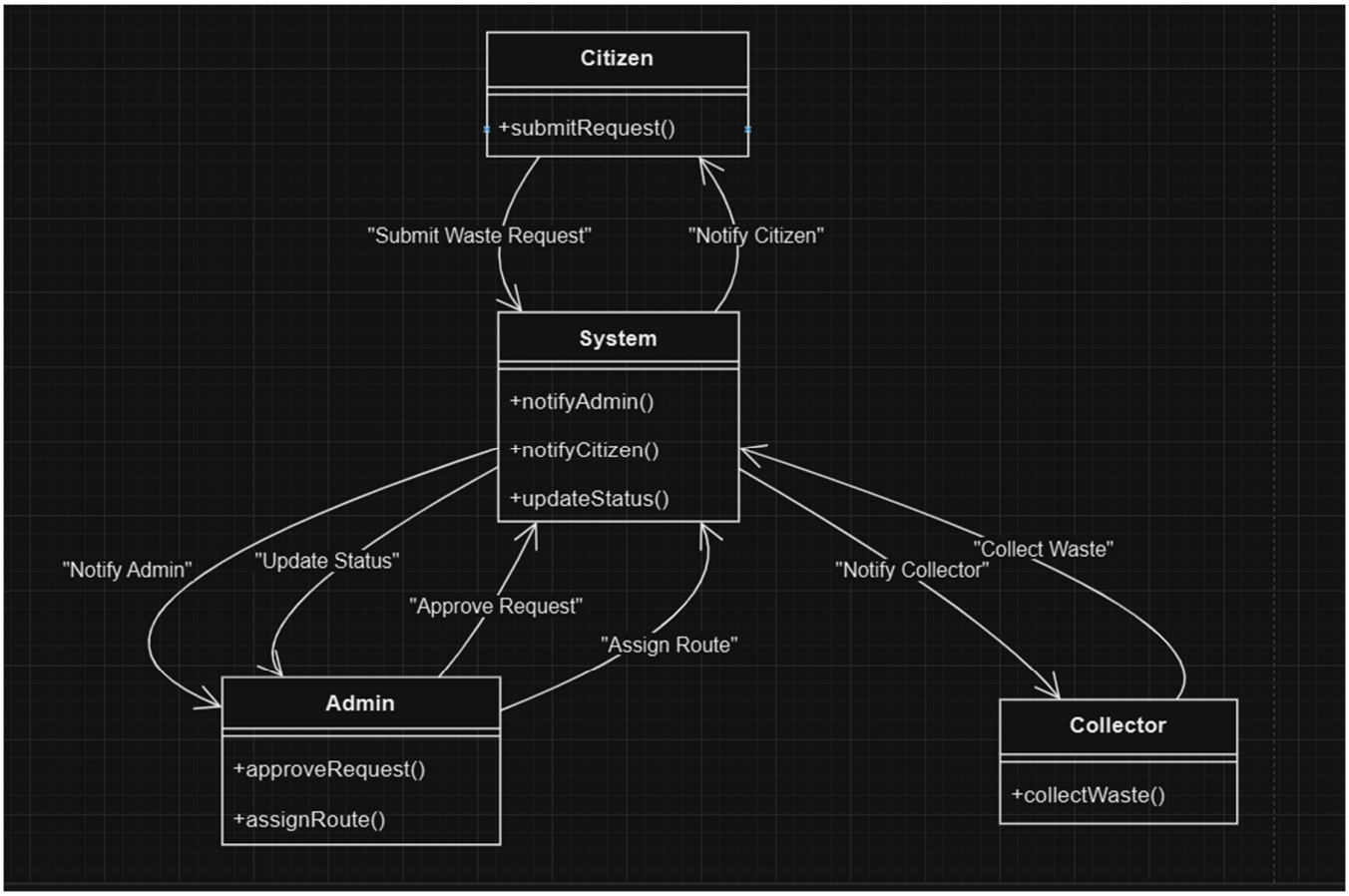
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

OUTPUT:


Citizen
+submitRequest()
"Submit Waste Request"
"Notify Citizen"
System
+notifyAdmin()
+notifyCitizen()
+updateStatus()
"Notify Admin"
"Update Status"
"Approve Request"
"Assign Route"
"Notify Collector"
"Collect Waste"
Admin
+approveRequest()
+assignRoute()
Collector
+collectWaste()

Result:

The collaboration diagram has been created successfully

EX NO:10	ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.
DATE	

AIM:

To Draw the Class Diagram for waste management system

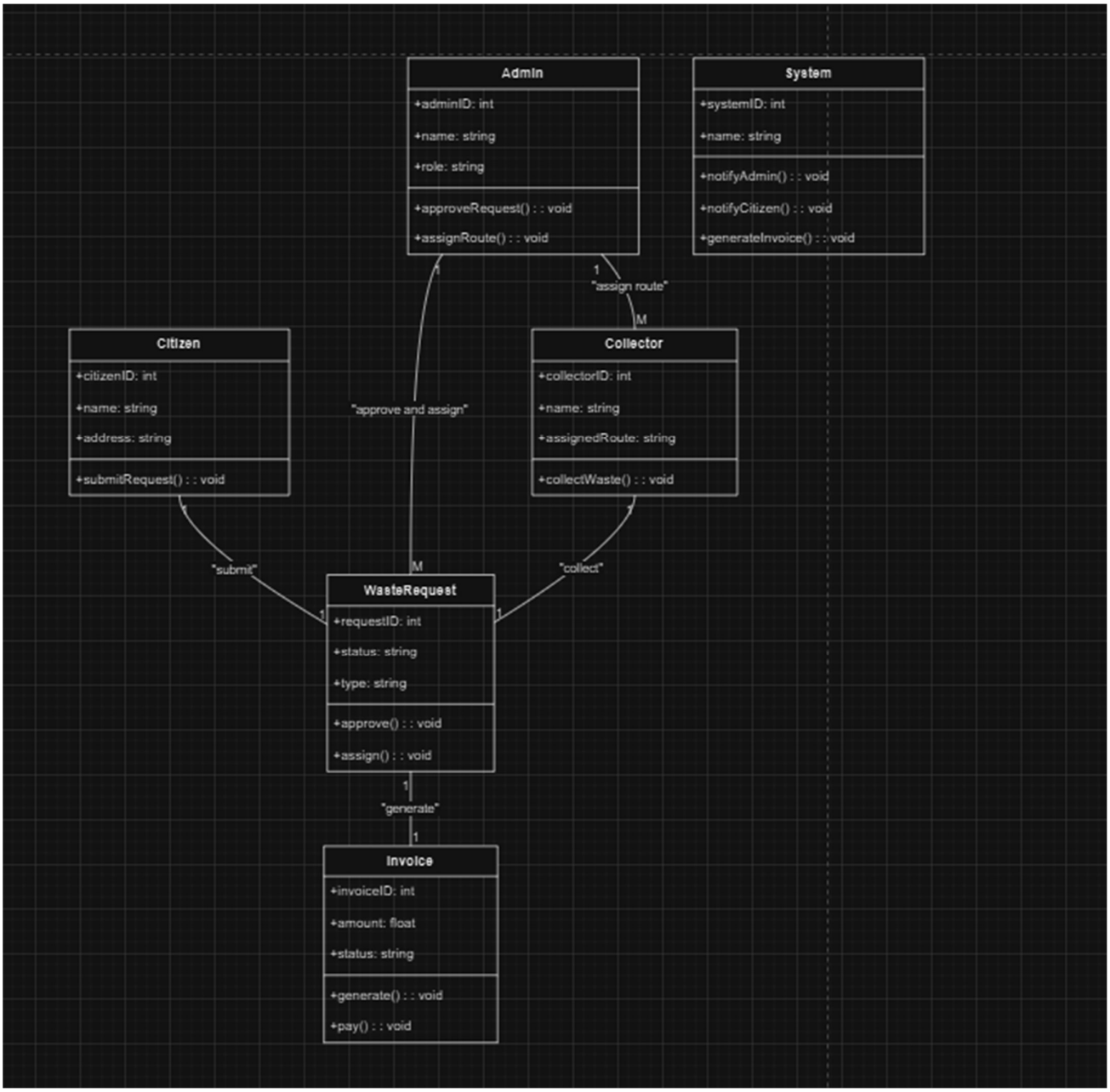
ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

OUTPUT:


Admin
+adminID: int
+name: string
+role: string
+approveRequest() : : void
+assignRoute() : : void
System
+systemID: int
+name: string
+notifyAdmin() : : void
+notifyCitizen() : : void
+generateInvoice() : : void
1
"assign route"
M
Citizen
+citizenID: int
+name: string
+address: string
+submitRequest() : : void
"approve and assign"
Collector
+collectorID: int
+name: string
+assignedRoute: string
+collectWaste() : : void
"submit"
M
"collect"
WasteRequest
+requestID: int
+status: string
+type: string
+approve() : : void
+assign() : : void
1
"generate"
1
Invoice
+invoiceID: int
+amount: float
+status: string
+generate() : : void
+pay() : : void

Result:

The class diagram for waste management system has been create successfully

EX NO:11	MINI PROJECT-WASTE MANAGEMENT SYSTEM
DATE	

Aim:

The Waste Management System aims to optimize waste collection, segregation, and disposal processes, ensuring effective resource utilization and environmental sustainability. It supports real-time tracking, efficient scheduling, and systematic recycling of waste.

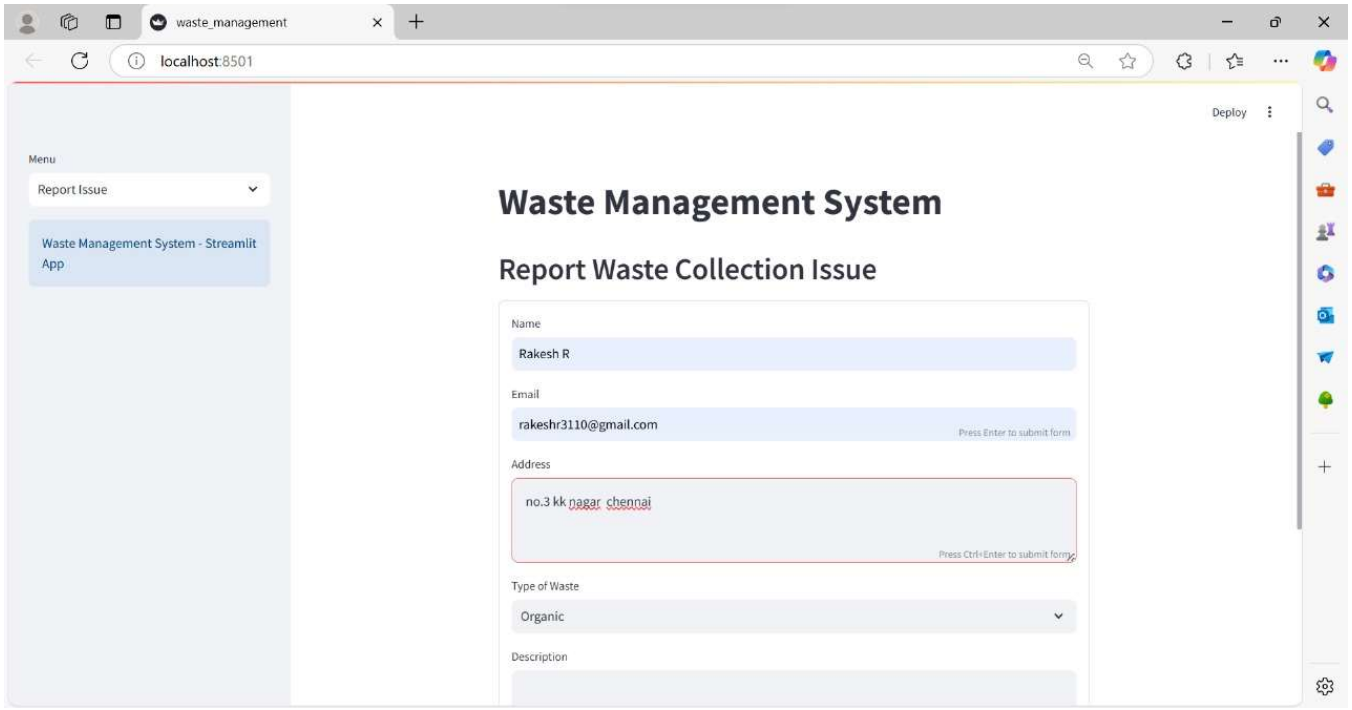
Algorithm:

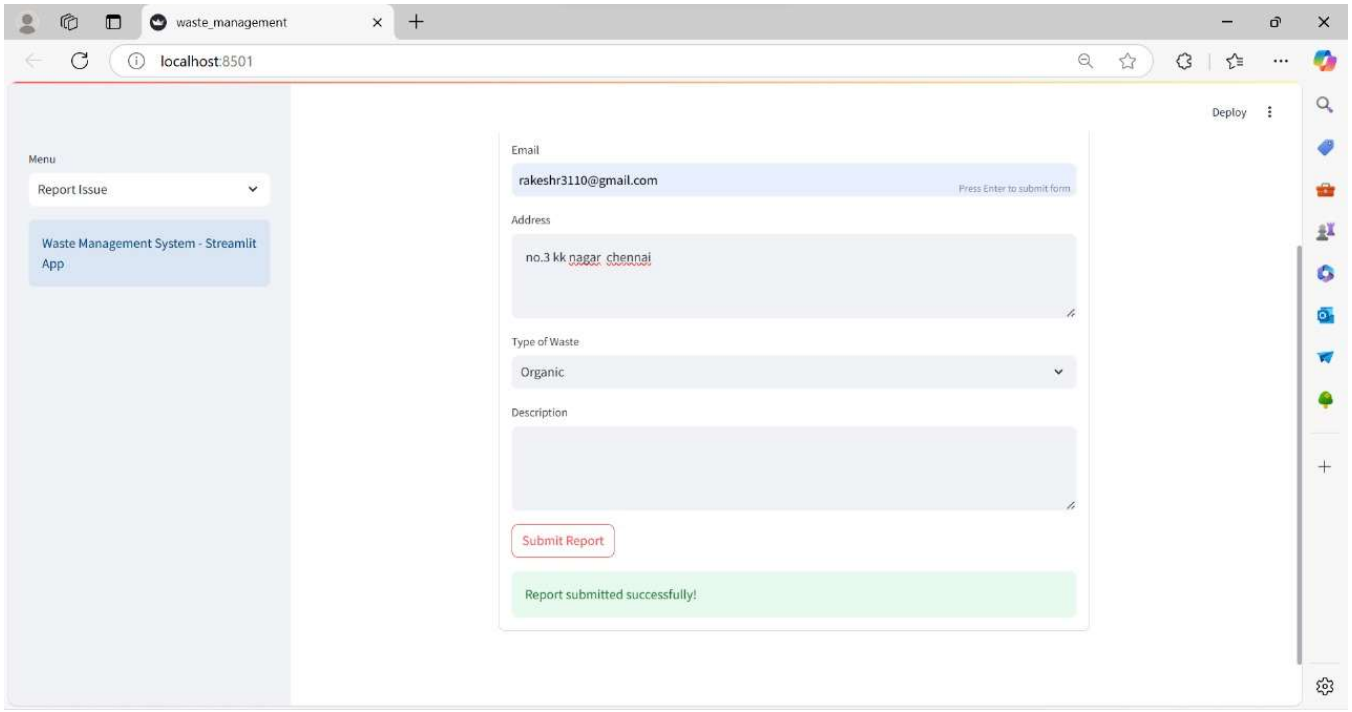
1. **User Registration**: Collect user details, assigning roles (household, commercial, municipal staff).
2. **Waste Entry**: Record types and quantities of waste collected (organic, recyclable, hazardous).
3. **Segregation Process**: Sort waste into categories for recycling, composting, or safe disposal.
4. **Collection Scheduling**: Schedule waste pickups, ensuring route optimization and timely services.
5. **Disposal Tracking**: Monitor disposal or recycling of waste to ensure environmental compliance.
6. **Alert Mechanism**: Notify users of collection dates or any issues in processing.
7. **Generate Reports**: Summarize data on waste collected, recycled materials, and environmental metrics for stakeholders.

Program:

```
import streamlit as st
import pandas as pd
import datetime as dt

# Initialize session state for data
if "users" not in st.session_state:
    st.session_state["users"] = []
if "waste_logs" not in st.session_state:
    st.session_state["waste_logs"] = []
if "schedules" not in st.session_state:
    st.session_state["schedules"] = []

# App Title
st.title("Waste Management System")

# Sidebar for navigation
menu = st.sidebar.radio("Menu", ["Register User", "Log Waste", "View Reports", "Schedule Pickup", "Admin
Dashboard"])

# **User Registration**
if menu == "Register User":
    st.header("User Registration")
    name = st.text_input("Enter Name:")
    address = st.text_area("Enter Address:")
    role = st.selectbox("Select Role:", ["Household", "Commercial", "Municipal Staff"])
    contact = st.text_input("Enter Contact Number:")

    if st.button("Register"):
        if name and address and role and contact:
            st.session_state["users"].append({"name": name, "address": address, "role": role, "contact": contact})
            st.success(f"{name} registered successfully as {role}.")
        else:
            st.error("Please fill all the fields.")
```

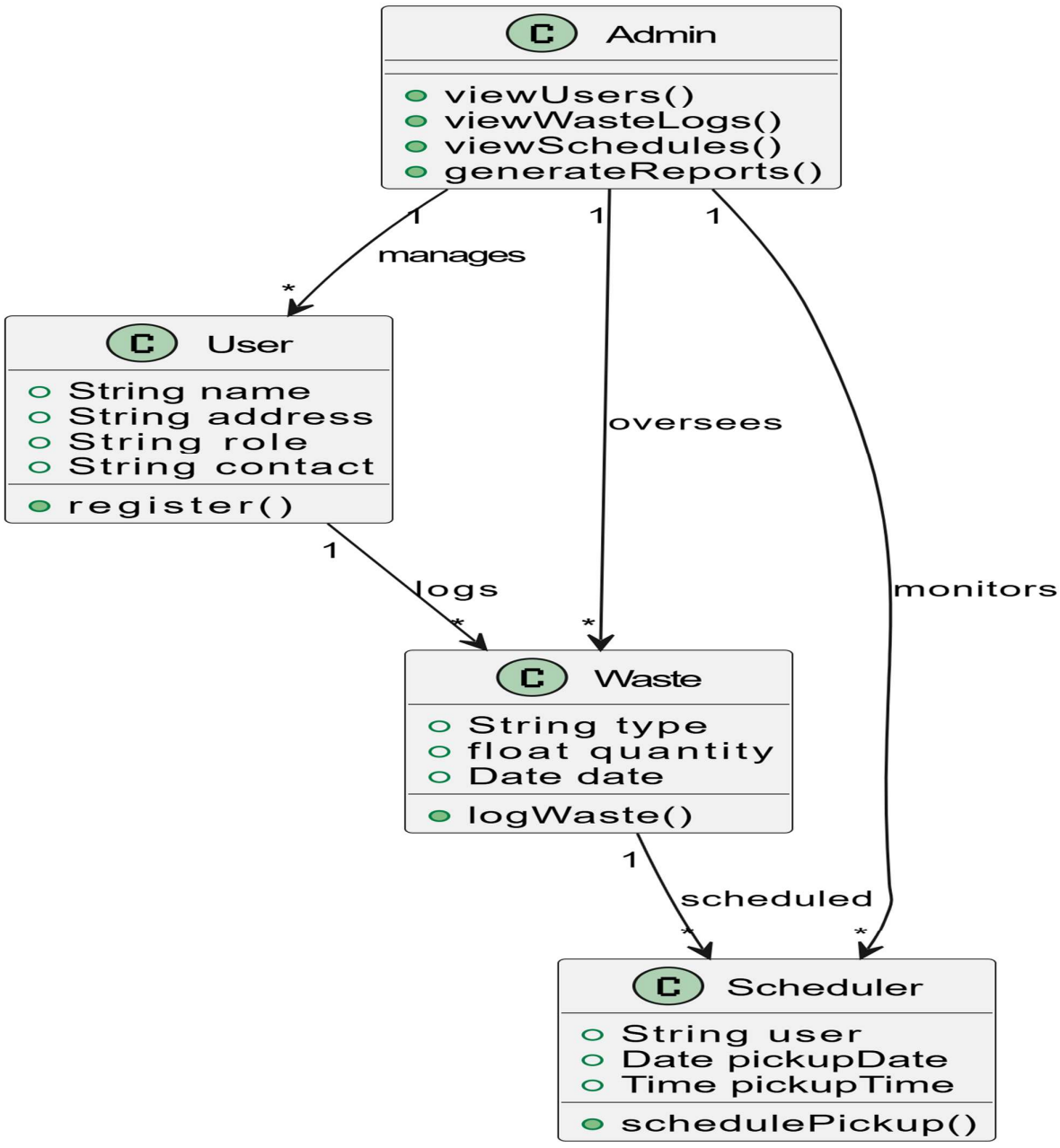
OUTPUT:


waste_management
localhost:8501
Deploy
Menu
Report Issue
Waste Management System - Streamlit App
Waste Management System
Report Waste Collection Issue
Name
Rakesh R
Email
rakeshr3110@gmail.com
Press Enter to submit form
Address
no.3 kk nagar chennai
Press Ctrl+Enter to submit form
Type of Waste
Organic
Description


waste_management
localhost:8501
Deploy
Menu
Report Issue
Waste Management System - Streamlit App
Email
rakeshr3110@gmail.com
Press Enter to submit form
Address
no.3 kk nagar chennai
Type of Waste
Organic
Description
Submit Report
Report submitted successfully!

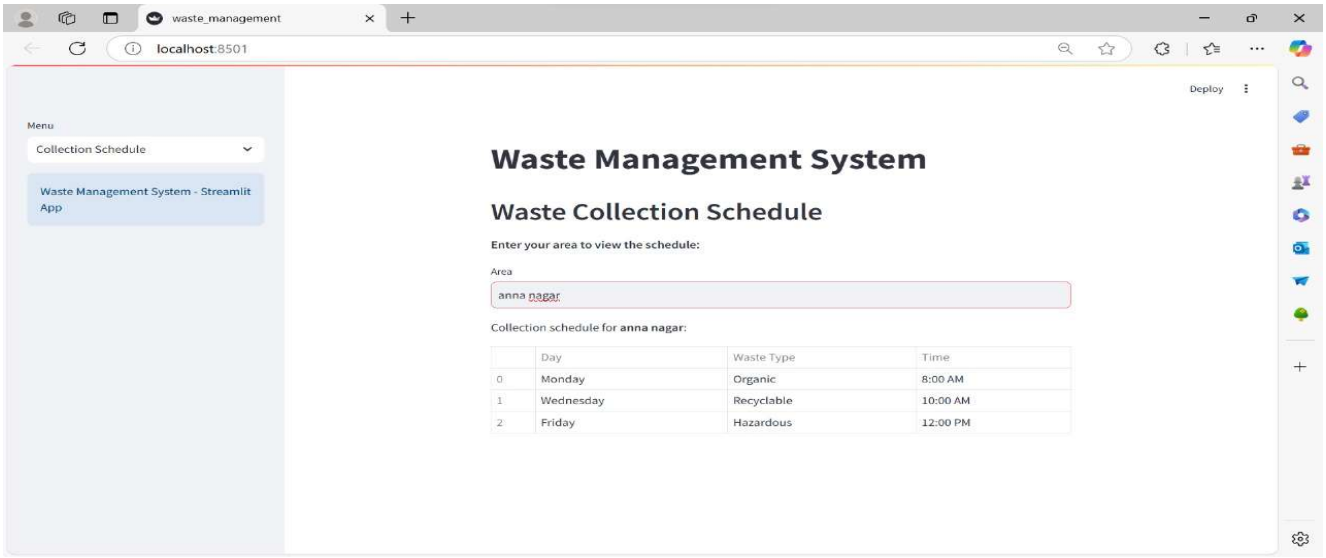
```
# **Log Waste**
elif menu == "Log Waste":
    st.header("Log Waste Details")

    if not st.session_state["users"]:
        st.warning("No users registered. Please register first.")
    else:
        user = st.selectbox("Select User:", [u["name"] for u in st.session_state["users"]])
        waste_type = st.selectbox("Waste Type:", ["Organic", "Recyclable", "Hazardous"])
        quantity = st.number_input("Enter Quantity (kg):", min_value=0.1, step=0.1)
        date = st.date_input("Date of Waste Generation:", max_value=dt.date.today())

        if st.button("Log Waste"):
            st.session_state["waste_logs"].append({
                "user": user,
                "waste_type": waste_type,
                "quantity": quantity,
                "date": date
            })
            st.success("Waste logged successfully!")

# **View Reports**
elif menu == "View Reports":
    st.header("Waste Management Reports")
    if not st.session_state["waste_logs"]:
        st.info("No waste logs available.")
    else:
        waste_df = pd.DataFrame(st.session_state["waste_logs"])
        st.subheader("Waste Summary by Type")
        summary = waste_df.groupby("waste_type")["quantity"].sum().reset_index()
        st.table(summary)

        st.subheader("Detailed Waste Logs")
        st.dataframe(waste_df)

# **Schedule Pickup**
elif menu == "Schedule Pickup":
    st.header("Schedule Waste Pickup")

    if not st.session_state["users"]:
        st.warning("No users registered. Please register first.")
    else:
        user = st.selectbox("Select User:", [u["name"] for u in st.session_state["users"]])
        pickup_date = st.date_input("Select Pickup Date:", min_value=dt.date.today())
        pickup_time = st.time_input("Select Pickup Time:")

        if st.button("Schedule Pickup"):
            st.session_state["schedules"].append({
                "user": user,
                "pickup_date": pickup_date,
                "pickup_time": pickup_time
            })
            st.success(f"Pickup scheduled for {user} on {pickup_date} at {pickup_time}.")
```

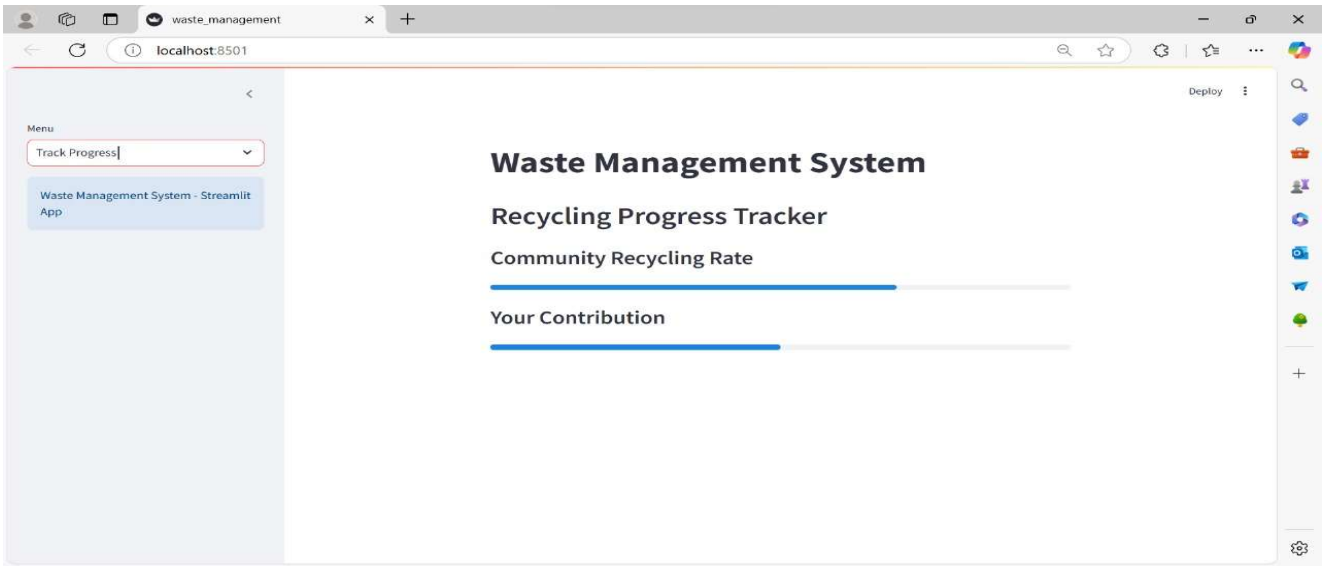
ER Diagram:

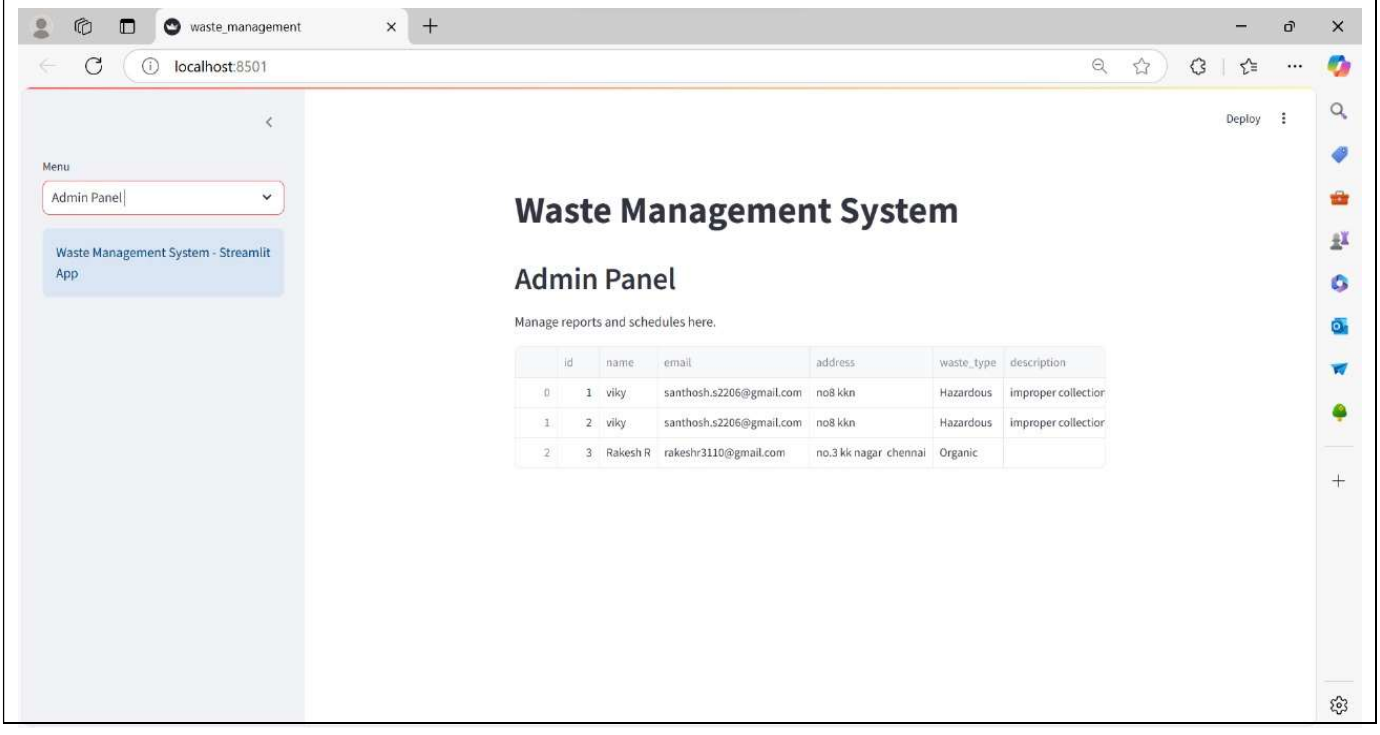

Admin
viewUsers()
viewWasteLogs()
viewSchedules()
generateReports()
1
manages
*
1
oversees
*
1
monitors
*
User
String name
String address
String role
String contact
register()
1
logs
*
Waste
String type
float quantity
Date date
logWaste()
1
scheduled
*
Scheduler
String user
Date pickupDate
Time pickupTime
schedulePickup()

```
# **Admin Dashboard**
elif menu == "Admin Dashboard":
    st.header("Admin Dashboard")

    st.subheader("Registered Users")
    if not st.session_state["users"]:
        st.write("No users registered.")
   else:
        users_df = pd.DataFrame(st.session_state["users"])
            st.dataframe(users_df)

    st.subheader("Scheduled Pickups")
    if not st.session_state["schedules"]:
        st.write("No pickups scheduled.")
    else:
        schedules_df = pd.DataFrame(st.session_state["schedules"])
        st.dataframe(schedules_df)

    st.subheader("Waste Logs")
    if not st.session_state["waste_logs"]:
        st.write("No waste logs available.")
    else:
        logs_df = pd.DataFrame(st.session_state["waste_logs"])
        st.dataframe(logs_df)
```

OUTPUT:


waste_management
localhost:8501
Deploy
Menu
Collection Schedule
Waste Management System - Streamlit App
Waste Management System
Waste Collection Schedule
Enter your area to view the schedule:
Area
anna nagar
Collection schedule for anna nagar:
Day
Waste Type
Time
0
Monday
Organic
8:00 AM
1
Wednesday
Recyclable
10:00 AM
2
Friday
Hazardous
12:00 PM


waste_management
localhost:8501
Deploy
Menu
Track Progress
Waste Management System - Streamlit App
Waste Management System
Recycling Progress Tracker
Community Recycling Rate
Your Contribution


waste_management
localhost:8501
Deploy
Menu
Admin Panel
Waste Management System - Streamlit App
Waste Management System
Admin Panel
Manage reports and schedules here.
id
name
email
address
waste_type
description
0
1
viky
santhosh.s2206@gmail.com
no8 kkn
Hazardous
improper collectior
1
2
viky
santhosh.s2206@gmail.com
no8 kkn
Hazardous
improper collectior
2
3
Rakesh R
rakeshr3110@gmail.com
no.3 kk nagar chennai
Organic

Conclusion:

The Waste Management System ensures effective handling of waste by streamlining processes like segregation, collection, and recycling. It promotes eco-friendly practices, reduces environmental impact, and provides data-driven insights for better decision-making. Implementing this system can significantly enhance operational efficiency and sustainability in waste management.