# IOT-Based Smart Waste Management System (Comprehensive Report)

**Group Members**

Riddhima Singh – BTech CSE – VI A – A80105222081
Aadyasha Patjoshi – BTech CSE – VI B – A80105222070

## Methodology

### 1. Requirement Analysis

- Identify and document the needs of **citizens** and the **Municipal Corporation**.
- Define key features such as **waste pickup requests, complaint registration, tracking waste collection trucks, waste segregation tips, and analytics for the municipality**.
- Gather regulatory and environmental compliance requirements.

### 2. System Design

- **Architecture Design:** Use a **client-server architecture** with cloud-based storage.
- **UI/UX Design:**
  - Design two separate dashboards (**Citizens & Municipal Corporation**) using Flutter's **Material Design & Cupertino widgets**.
  - Ensure a responsive UI for mobile, tablet, and web compatibility.
- **Database Design:** Structure the database for storing **user profiles, complaint reports, waste pickup schedules, truck tracking data, etc.**

### 3. Technology Stack Selection

- **Frontend:** Flutter (Dart)
- **Backend:**
  - Firebase (Firestore, Authentication, Cloud Functions) – Serverless
  - OR Node.js (Express) / Python (FastAPI, Django) for custom APIs
- **Database:** Firebase Firestore / PostgreSQL / MySQL
- **Cloud Services:** AWS/GCP/Azure for hosting and storage

- **IoT Integration (if applicable):** MQTT or Firebase Realtime Database for smart waste bins
- **Google Maps API:** For real-time truck tracking

## 4. Development Phase

- **Citizen Dashboard Features:**
  - Register/Login (Google, Email, OTP-based)
  - Request Waste Pickup
  - File Complaints & Track Status
  - View Waste Segregation Tips
  - Get Notifications & Alerts
- **Municipal Corporation Dashboard Features:**
  - Monitor Waste Collection Requests
  - Assign & Track Waste Trucks (Google Maps API)
  - Generate Reports & Analytics
  - Respond to Citizen Complaints
  - Manage Workforce

## 5. Testing & Quality Assurance

- **Unit Testing:** Test individual Flutter components.
- **Integration Testing:** Ensure Flutter frontend communicates correctly with the backend.
- **Performance Testing:** Optimize animations and API calls.
- **Security Testing:** Secure Firebase rules, API endpoints, and user data.

## 6. Deployment

- **Mobile App:** Deploy on **Google Play Store** (Android) and **App Store** (iOS).
- **Web Version (Optional):** Host via **Firebase Hosting or AWS Amplify**.
- **Admin Dashboard:** Host as a web app for Municipal Corporation access.

### 7. Maintenance & Updates

- Regularly update the app for bug fixes and security patches.
- Gather user feedback and improve features.
- Implement AI/ML (if required) for predictive analytics on waste collection trends.

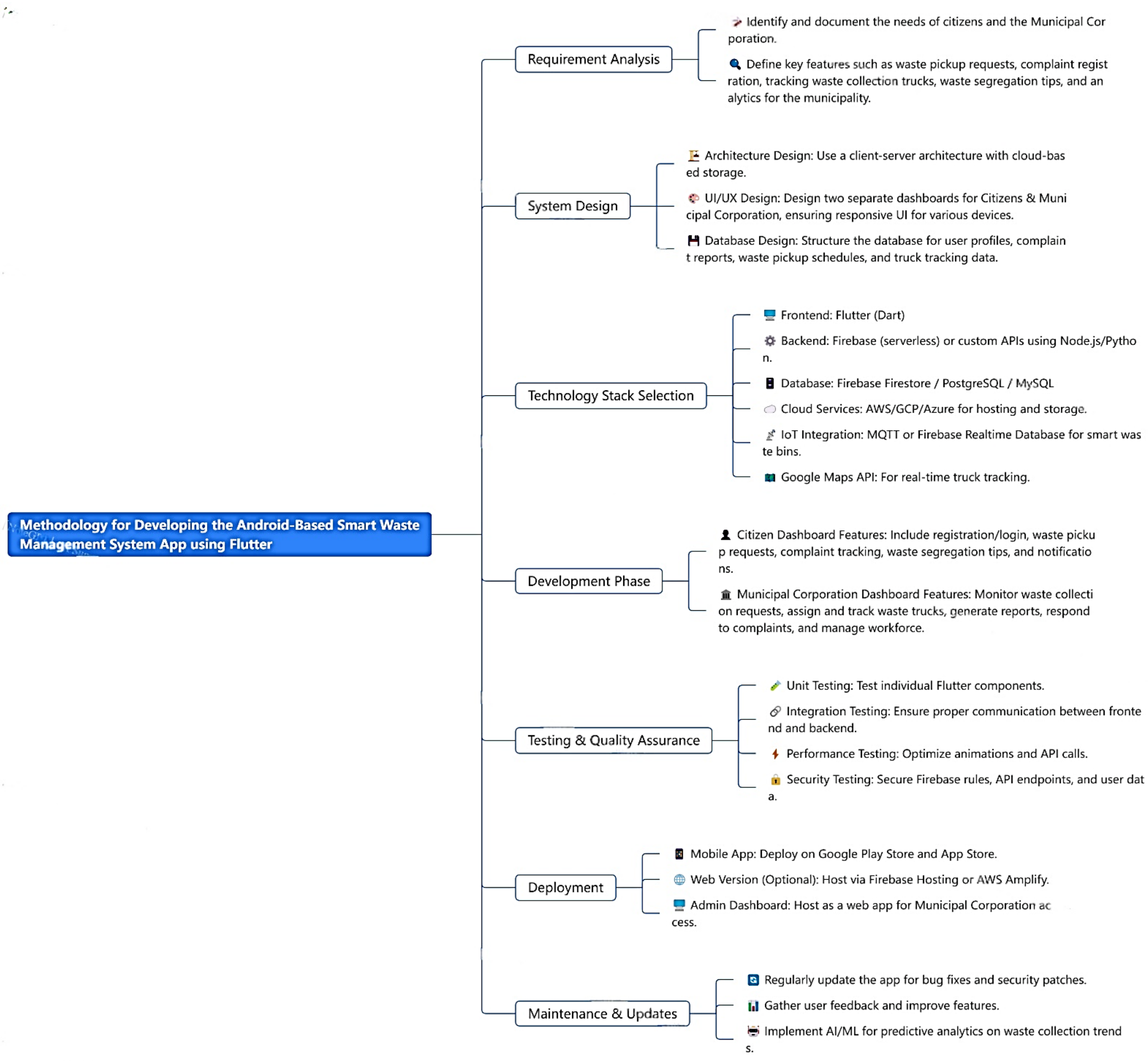
---

## Technology Stack Selection

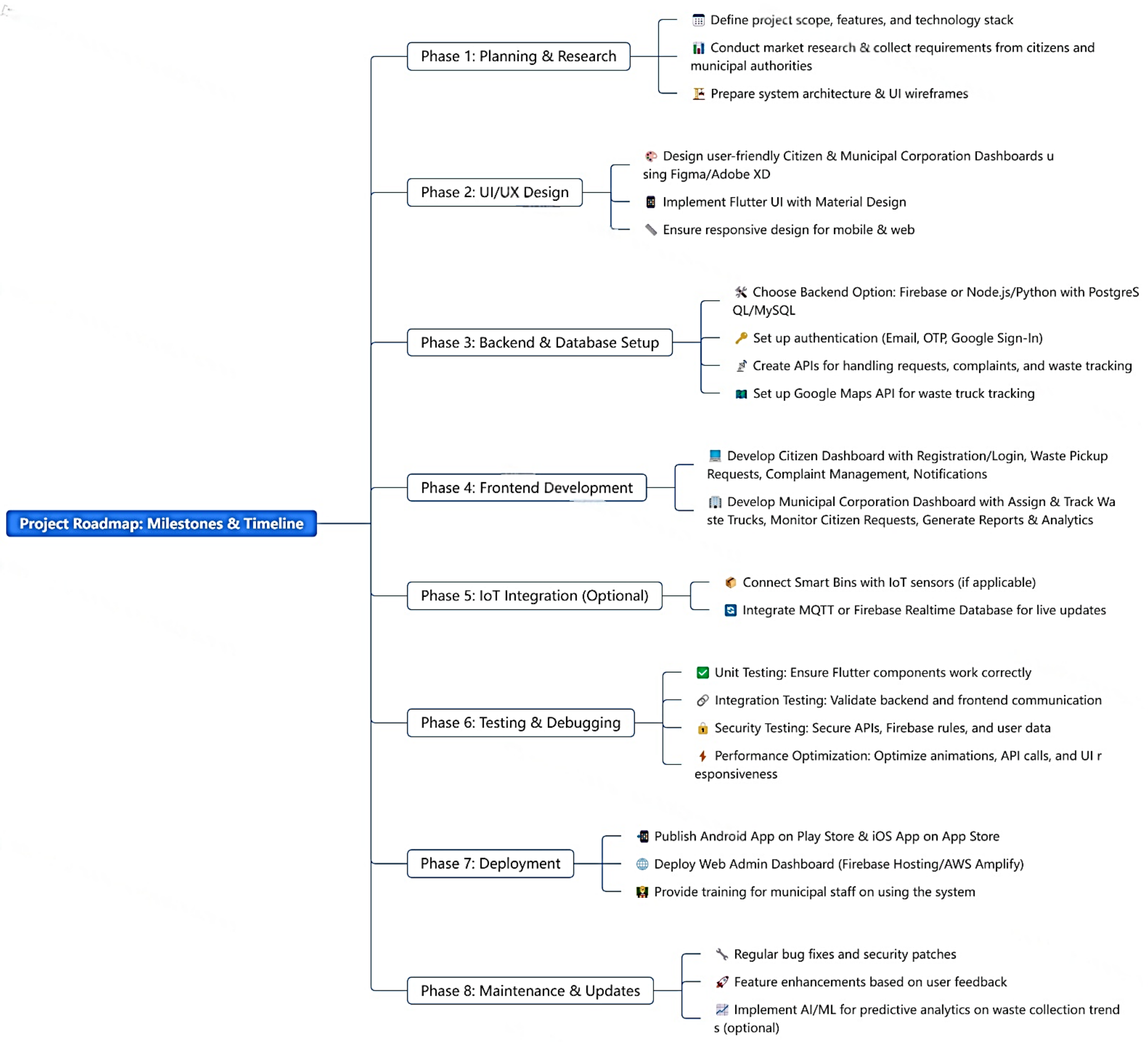
- **Frontend:** Flutter (Dart)
- **Backend:**
  - Firebase (Firestore, Authentication, Cloud Functions) – Serverless
  - OR Node.js (Express) / Python (FastAPI, Django) for custom APIs
- **Database:** Firebase Firestore / PostgreSQL / MySQL
- **Cloud Services:** AWS/GCP/Azure for hosting and storage
- **IoT Integration (if applicable):** MQTT or Firebase Realtime Database for smart waste bins
- **Google Maps API:** For real-time truck tracking
- **Security Measures:** JWT Authentication (Mechanism for secure user authentication and session management) GDPR Compliance (Adherence to ethical AI standards and data privacy regulations)

---

# Flowchart



- Methodology for Developing the Android-Based Smart Waste Management System App using Flutter
  - Requirement Analysis
    - Identify and document the needs of citizens and the Municipal Corporation.
    - Define key features such as waste pickup requests, complaint registration, tracking waste collection trucks, waste segregation tips, and analytics for the municipality.
  - System Design
    - Architecture Design: Use a client-server architecture with cloud-based storage.
    - UI/UX Design: Design two separate dashboards for Citizens & Municipal Corporation, ensuring responsive UI for various devices.
    - Database Design: Structure the database for user profiles, complaint reports, waste pickup schedules, and truck tracking data.
  - Technology Stack Selection
    - Frontend: Flutter (Dart)
    - Backend: Firebase (serverless) or custom APIs using Node.js/Python.
    - Database: Firebase Firestore / PostgreSQL / MySQL
    - Cloud Services: AWS/GCP/Azure for hosting and storage.
    - IoT Integration: MQTT or Firebase Realtime Database for smart waste bins.
    - Google Maps API: For real-time truck tracking.
  - Development Phase
    - Citizen Dashboard Features: Include registration/login, waste pickup requests, complaint tracking, waste segregation tips, and notifications.
    - Municipal Corporation Dashboard Features: Monitor waste collection requests, assign and track waste trucks, generate reports, respond to complaints, and manage workforce.
  - Testing & Quality Assurance
    - Unit Testing: Test individual Flutter components.
    - Integration Testing: Ensure proper communication between frontend and backend.
    - Performance Testing: Optimize animations and API calls.
    - Security Testing: Secure Firebase rules, API endpoints, and user data.
  - Deployment
    - Mobile App: Deploy on Google Play Store and App Store.
    - Web Version (Optional): Host via Firebase Hosting or AWS Amplify.
    - Admin Dashboard: Host as a web app for Municipal Corporation access.
  - Maintenance & Updates
    - Regularly update the app for bug fixes and security patches.
    - Gather user feedback and improve features.
    - Implement AI/ML for predictive analytics on waste collection trends.



**Project Roadmap: Milestones & Timeline**

- Phase 1: Planning & Research
  - 🗓️ Define project scope, features, and technology stack
  - 📊 Conduct market research & collect requirements from citizens and municipal authorities
  - 🏗️ Prepare system architecture & UI wireframes
- Phase 2: UI/UX Design
  - 🎨 Design user-friendly Citizen & Municipal Corporation Dashboards using Figma/Adobe XD
  - 📱 Implement Flutter UI with Material Design
  - 📏 Ensure responsive design for mobile & web
- Phase 3: Backend & Database Setup
  - 🛠️ Choose Backend Option: Firebase or Node.js/Python with PostgreSQL/MySQL
  - 🔑 Set up authentication (Email, OTP, Google Sign-In)
  - 📡 Create APIs for handling requests, complaints, and waste tracking
  - 🗺️ Set up Google Maps API for waste truck tracking
- Phase 4: Frontend Development
  - 💻 Develop Citizen Dashboard with Registration/Login, Waste Pickup Requests, Complaint Management, Notifications
  - 🏢 Develop Municipal Corporation Dashboard with Assign & Track Waste Trucks, Monitor Citizen Requests, Generate Reports & Analytics
- Phase 5: IoT Integration (Optional)
  - 📦 Connect Smart Bins with IoT sensors (if applicable)
  - 🔄 Integrate MQTT or Firebase Realtime Database for live updates
- Phase 6: Testing & Debugging
  - ✅ Unit Testing: Ensure Flutter components work correctly
  - 🔗 Integration Testing: Validate backend and frontend communication
  - 🔒 Security Testing: Secure APIs, Firebase rules, and user data
  - ⚡ Performance Optimization: Optimize animations, API calls, and UI responsiveness
- Phase 7: Deployment
  - 📲 Publish Android App on Play Store & iOS App on App Store
  - 🌐 Deploy Web Admin Dashboard (Firebase Hosting/AWS Amplify)
  - 👨‍🏫 Provide training for municipal staff on using the system
- Phase 8: Maintenance & Updates
  - 🔧 Regular bug fixes and security patches
  - 🚀 Feature enhancements based on user feedback
  - 📈 Implement AI/ML for predictive analytics on waste collection trends (optional)

# References

1. Kadus, T., Nirmal, P., & Kulkarni, K. (2020). *Smart waste management system using IoT*. International Journal of Engineering Research & Technology, 9(4). Retrieved from www.ijert.org
2. Sosunova, I., & Porras, J. (2022). *IoT-enabled smart waste management systems for smart cities: A systematic review*. IEEE Access, 10, 73326-73353. https://doi.org/10.1109/ACCESS.2022.3188308