Software Requirements Specification (SRS) Document for CEP

1. Introduction

1.1 Purpose

The “SwachhaRakshak” web application aims to address urban waste management issues by providing a quick, efficient, and transparent reporting system. Through a browser-based platform, citizens can report uncollected garbage using uploaded images, location information, and direct integration with municipal services, ensuring rapid cleanup responses.

1.2 Scope

The web application will serve urban and semi-urban localities facing waste management challenges. It will enable citizens to report garbage piles, track resolution progress, and assist municipal authorities in identifying and resolving waste issues in real-time.

1.3 Definitions, Acronyms, and Abbreviations

- GPS: Global Positioning System
- UI/UX: User Interface/User Experience
- Backend: The server-side infrastructure of the web app
- Web App: CleanSnap Web Application

1.4 References

- Literature: IoT-based Waste Management Systems
- Backend Services: Firebase/Cloud services
- Web Development Frameworks: React.js, Angular, or Vue.js
- Mapping APIs: Google Maps API

1.5 Overview

This document outlines the system requirements, including functional and non-functional requirements, user interactions, hardware and software interfaces, and system architecture for

CleanSnap Web Application.

---

2. System Overview

The CleanSnap web app will allow users to log in via browsers, report garbage using images and location information, and track report status. The backend will integrate with municipal dashboards for real-time updates. Additionally, an incentive system will encourage user participation in waste reporting.

---

3. Functional Requirements

3.1 User Registration

- Feature: Users can register via email, phone number, or social accounts.
- Inputs: Name, email, phone number, and optional location (for verification).
- Outputs: Account creation, login confirmation.

3.2 Garbage Reporting

- Feature: Citizens report garbage through a simple form.
- Inputs: Uploaded photo (via browser), location (auto-captured with browser's geolocation API or manually entered), waste category (wet, dry, hazardous), brief description.
- Outputs: Report confirmation, progress tracking (pending, in progress, resolved).

3.3 Municipal Integration

- Feature: Seamless forwarding of reports to municipal teams.
- Inputs: Report data (location, uploaded photo, waste category).
- Outputs: Notifications to municipal dashboards, tracking updates.

3.4 Incentive System

- Feature: Citizens earn redeemable coins or vouchers per report.
- Inputs: Validated report submission (time, location).
- Outputs: Reward balance update, redemption history.

3.5 Real-Time Status Updates

- Feature: Citizens track garbage resolution status.
- Inputs: Status updates from municipal teams.
- Outputs: Display of updated status on user dashboard.

3.6 Feedback System

- Feature: Citizens rate cleanup quality.
- Inputs: Star rating, comments.
- Outputs: Feedback stored and forwarded to municipal teams.

---

4. Non-Functional Requirements

4.1 Performance

- Must handle 10,000+ simultaneous users.
- Updates delivered with <2 seconds latency.

4.2 Security

- All data (photos, location) encrypted during transfer/storage.
- Two-factor authentication (2FA) for login.

4.3 Usability

- User-friendly, accessible via all major browsers (Chrome, Firefox, Edge, Safari).
- Accessibility features: screen reader compatibility, high-contrast themes, larger fonts.

4.4 Scalability

- Expandable to support multiple municipalities.
- Cloud-based backend (Firebase/Google Cloud).

4.5 Reliability

- 99% uptime with redundant data backups.
- Graceful error handling for connectivity failures.

---

5. System Architecture

5.1 Overview

- Frontend: React.js/Angular/Vue.js web app, integrated with Google Maps API for geolocation and Firebase for storage.
- Backend: Firebase (real-time database, authentication) + Google Cloud Functions for processing.
- Database: Firestore for user, report, and reward data.

5.2 Data Flow

1. The user submits a waste report (image, description, location).
2. Backend stores report and notifies the municipal dashboard.
3. Municipal team updates status.
4. Updates sent back to the user's dashboard in real-time.

---

6. User Interface (UI) Design

6.1 Home Dashboard

- Quick access to report form, last report status, reward points.

6.2 Report Garbage Page

- Map with user’s location (auto/manual), photo upload option, waste category dropdown, submit button.

6.3 Rewards Page

- Display earned coins, voucher redemption options.

6.4 Status Tracking Page

- History of user reports with current status (pending, resolved, rejected).

---

7. External Interface Requirements

7.1 Hardware Interfaces

- Any desktop/laptop/smartphone with internet access and camera (optional).

7.2 Software Interfaces

- Google Maps API: For location tracking.
- Firebase Cloud Storage: For storing uploaded images.
- Google Cloud Functions: For backend processing.

7.3 Communication Interfaces

- REST APIs for client-server communication.
- Secure HTTPS for data transfer.

---

8. System Models

8.1 Use Case Diagram

- Actors: Citizens (Users), Municipal Teams.
- Use Cases: Register/Login, Report Garbage, Track Status, Redeem Rewards, Give Feedback.

8.2 Activity Diagram

- Workflow: User reports waste → Data stored → Municipal team notified → Cleanup action → Status updated → Reward issued → User feedback.

---

9. Appendices

9.1 Glossary

- Waste Categories: Wet, Dry, Hazardous, Mixed.
- Municipal Dashboard: Web-based portal for municipal staff to track and resolve waste reports.

9.2 References

- IEEE Conference Paper on IoT in Waste Management.
- Smart City Waste Management Guidelines (Journal, 2023).