**CEBU INSTITUTE OF TECHNOLOGY**

**UNIVERSITY**

COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for

**Garbage Management System**

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision No. | Changes | Date | Changed By |
| 1 | Initial Draft of Everything | 1/28/2025 | Paraiso, Gella, Largo |

Table of Contents

Change History 2

Table of Contents 3

1. Introduction 4

1.1. Purpose 4

1.2. Scope 4

1.3. Definitions, Acronyms and Abbreviations 4

1.4. References 4

2. Overall Description 5

2.1. Product perspective 5

2.2. User characteristics 5

2.4. Constraints 5

2.5. Assumptions and dependencies 6

3. Specific Requirements 7

3.1. External interface requirements 7

3.1.1. Hardware interfaces 7

3.1.2. Software interfaces 7

3.1.3. Communications interfaces 7

3.2. Functional requirements 7

Module 1 7

Module 2 8

3.4 Non-functional requirements 8

Performance 8

Security 8

Reliability 8

# Introduction

## Purpose

### This Software Requirements Specification (SRS) outlines the detailed requirements for the Garbage Management System (GMS), which will manage waste collection, provide notifications, track garbage trucks, and offer waste reduction tips. The document covers both functional and non-functional requirements to ensure the system meets stakeholder expectations.

### The SRS serves as a communication tool between developers, project managers, and end-users, ensuring the final product aligns with business objectives. It will also guide the development, testing, and future maintenance of the system.

### ****Intended Audience****

The intended audience for this SRS document includes:

1. **Project Managers**: To understand the scope, timeline, and technical requirements of the project.
2. **System Developers**: To clearly define the system's features, functionalities, and interactions to guide them in the development phase.
3. **Stakeholders**: Including local government agencies, municipalities, and organizations involved in waste management, to align the system with their operational needs.
4. **End Users**: Residents, municipal staff, and street cleanup organizations, who will use the application to manage trash pickup schedules, report issues, donate for cleanup, and more.

## Scope

* **Software Products**: The Garbage Management System will include a web-based administrative interface, a mobile application, and a backend server. Specific products include:
  + **Garbage Truck Tracker**: A GPS-based real-time tracking system.
  + **Waste Reporting Module**: Allows users to report waste piles with pictures and pin locations on maps.
  + **Biometric Login System**: Provides secure authentication for mobile app users.
  + **Notification System**: Sends reminders for trash pickup and alerts for delays or rescheduling.
  + **Donation Module**: Facilitates contributions for street cleanup initiatives.
* **What the Software Will Do**:
  + Enable real-time tracking of garbage trucks and display their locations on maps.
  + Allow residents to report waste piles and verify issues with photo uploads.
  + Provide notifications for upcoming garbage pickups and community initiatives.
  + Support secure biometric login to ensure user data protection.
  + Integrate Google Maps API for accurate location tracking.
  + Enable communication via Facebook Messenger for updates and alerts.
* **What the Software Will Not Do**:
  + The system will not physically handle waste collection or truck management.
  + It will not provide functionality for non-garbage-related community services.
* **Application and Benefits**:
  + **Application**: The system will be used by municipalities, garbage truck drivers, and residents to streamline waste collection and management.
  + **Benefits**:
    - Improved efficiency in garbage collection.
    - Enhanced community involvement through reporting and donations.
    - Real-time updates and notifications for all stakeholders.
  + **Objectives and Goals**:
    - Reduce response time to waste pile issues.
    - Encourage cleaner streets through community participation.
    - Provide a user-friendly interface for both residents and administrators.

## Definitions, Acronyms and Abbreviations

* *provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS*
* **GMS**: Garbage Management System
* **API**: Application Programming Interface
* **CRUD**: Create, Read, Update, Delete
* **RDS**: Relational Database Service

## References

* *Provide a complete list of all documents referenced elsewhere in the SRS;*
* *Identify each document by title, report number (if applicable), date, and publishing organization;*
* *Specify the sources from which the references can be obtained.*

* Google Maps API Documentation
* Facebook Messenger API Documentation
* Java Spring Boot Documentation
* Android Kotlin Documentation

# Overall Description

## Product perspective

* *Put software product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.*
* *A block diagram showing the major components of the larger system, interconnections, and external inter- faces can be helpful.*
* *Describe the modular decomposition of the components using the format below:*

*Module 1*

*Transaction 1.1*

*Transaction 1.2*

*Module 2*

*Transaction 2.1*

*Transaction 2.2*

*. . .*

The GMS is a client-server system. The backend is built with Java Spring Boot, the web frontend uses ReactJS, and the mobile application is developed in Kotlin. Google Maps API provides location services, while Facebook Messenger API handles communication.

## User characteristics

* **Administrators**: Schedule and manage garbage collection.
* **Drivers**: Use the mobile app for live tracking and reporting.
* **Residents**: Use the app to view schedules, report issues, and receive notifications.

## 2.4. Constraints

* *Provide a general description of any other items that will limit the developer’s options.*
* *Regulatory policies;*
* *Hardware limitations (e.g., signal timing requirements);*
* *Interfaces to other applications;*
* *Parallel operation;*
* *Audit functions;*
* *Control functions;*
* *Reliability requirements;*
* *Criticality of the application;*
* *Safety and security considerations.*
* System must be hosted on a cloud platform (e.g., AWS, Azure).
* The mobile app must function on devices with Android 8.0 or higher.

## 2.5. Assumptions and dependencies

*This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not avail- able, the SRS would then have to change accordingly.*

* Reliable internet connectivity is required for all users.
* Google Maps and Facebook APIs must be available and functional.

# Specific Requirements

## External interface requirements

### 3.1.1. Hardware interfaces

*This should specify the logical characteristics of each interface between the software product and the hard- ware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.*

* Android smartphones for the mobile application.
* Computers for web-based admin interface.

### 3.1.2. Software interfaces

*This should specify the use of other required software products (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system).*

* Backend: Java Spring Boot
* Web Frontend: ReactJS
* Mobile App: Android Kotlin
* Database: MySQL hosted on AWS RDS, Firebase for Mobile

### 3.1.3. Communications interfaces

*This should specify the various interfaces to communications such as local network protocols, etc.*

* REST APIs for data exchange between web, mobile, and backend.
* Google Maps API for location services.
* Facebook Messenger API for communication.

## Functional requirements

### Module 1

#### **1.1 Pickup Schedule Viewer**

*Users can access an interactive calendar or schedule view showing the exact times and dates when garbage collection trucks will pick up trash in their area. This allows them to plan when to put out their garbage and avoid missed pickups.*

### Module 2

#### **2.1 Garbage Pile Location Map** The web app displays a map with markers indicating where the garbage piles are located within a specified area. This helps waste collectors easily identify locations and optimize their routes. Users can also report or update the status of their garbage pile on the map.

### Module 3

#### **3.1 Notifications and Alerts** The web app can send notifications (via email or on-screen alerts) reminding users of their scheduled garbage pickup time. Alerts could also be set to inform users about changes in schedule, delays, or special collection days (like holiday pickups).

### Module 4

#### **4.1 Push Notifications**

*The mobile app sends real-time push notifications directly to users' smartphones to remind them of upcoming pickup times. It can also notify users of delays or if the truck is nearby, ensuring that users are well-informed and don't miss their scheduled garbage collection.*

### Module 5

#### **5.1 QR Code Scanning for Garbage Reporting**

#### The web app can send notifications (via email or on-screen alerts) reminding users of their scheduled garbage pickup time. Alerts could also be set to inform users about changes in schedule, delays, or special collection days (like holiday pickups).

### Module 6

#### **6.1 Live Tracking of Garbage Truck**

#### The web app can send notifications (via email or on-screen alerts) reminding users of their scheduled garbage pickup time. Alerts could also be set to inform users about changes in schedule, delays, or special collection days (like holiday pickups).

#### The web app can send notifications (via email or on-screen alerts) reminding users of their scheduled garbage pickup time. Alerts could also be set to inform users about changes in schedule, delays, or special collection days (like holiday pickups).

## Non-functional requirements

### Performance

##### Details

* The system must handle up to 100 concurrent users.
* Real-time updates for GPS tracking must have latency below 2 seconds.

### Security

##### Details

* All data must be encrypted in transit and at rest.
* Biometric data is stored securely and never shared.

### Reliability

##### Details

* System uptime must be 99.9%.
* Automated backups of the database every 24 hours.