**System Requirements Specification (SRS)**

**Project Title:** Resourcify: Smart Community Resource Management System  
**Student Number:** 23/05373  
**Student Name:** Nicholas Kariuki Wambui

### 1. Introduction

**1.1 Purpose**  
This document outlines the system requirements for **Resourcify**, a web-based platform designed to optimize the allocation and distribution of essential resources in communities, universities, and NGOs. The system aims to enhance transparency, minimize waste, and improve resource tracking and management through automation, data analytics, and geospatial mapping.

**1.2 Scope**

Resourcify allows users to view, request, donate, and track various resources across different locations. The system ensures efficient allocation and transparent tracking of resources, with features like request management, real-time updates via WebSockets, and Google Maps integration.

**1.3 Context Diagram**  
The system will integrate with external APIs (e.g., Google Maps API) and interact with users (Admins, Donors, and Users) to manage resources efficiently. A context diagram will be provided in the final design phase.

**1.4 Definitions, Acronyms, and Abbreviations**

| **Term** | **Definition** |
| --- | --- |
| UI | User Interface |
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |
| REST | Representational State Transfer |
| MVC | Model View Controller |

**1.5 References**

* Gao, P., & Thierer, A. (2018). "Smart resource allocation: The role of digital platforms in optimizing supply chains."
* Smith, J. (2020). "Data-driven decision-making in resource management."

### 2. General Description

**2.1 System Functions**  
The system will provide the following functionalities:

* **Resource Inventory Management:** Maintain an updated record of available resources.
* **Request System:** Allow users to submit resource requests.
* **Approval Workflow:** Admins can review, approve, or decline requests.
* **Predictive Analytics:** Analyze consumption trends to predict future demand.
* **Geographical Mapping:** Visualize resource locations using Google Maps API.
* **Role-Based Access:** Differentiate between Admins, Donors, and Users.
* **CRUD operations on resources**
* **WebSocket-based real-time notifications**

**2.2 User Characteristics**

* **Admins:** Full access to manage resources, approve requests, and view analytics.
* **Donors:** Can donate resources and track their contributions.
* **Users:** Can request resources and view available supplies.

### 3. General Constraints

**3.1 Product Perspective**

Resourcify is an independent web-based system integrating a backend (Spring Boot & MySQL) and a frontend (HTML, CSS, JavaScript). It interacts with Google Maps API for resource location and WebSocket for real-time communication.

**3.2 Software Constraints**

* The system must be compatible with modern web browsers (Chrome, Firefox, Edge).
* The backend will be developed using Java (Spring Boot), and the frontend will use HTML, CSS, and JavaScript.
* The database will be MySQL or SQLite.
* Google Maps API key usage limits

**3.3 Hardware Constraints**

* The system must support at least 100 concurrent users.
* Development requires a laptop with at least 8GB RAM and an i5 processor.

### 4. Assumptions and Dependencies

* Users will have internet access and a modern web browser.
* The system will use a MySQL/SQLite database for data storage.
* Users have basic web navigation knowledge
* Dependencies: Google Maps API, SockJS, STOMP, Spring Boot

### 5. Functional Requirements Master List

| **Req. ID** | **Requirement Name** | **Requirement Description** |
| --- | --- | --- |
| REQ-1.1 | User Authentication | Users must log in with a username and password to access the system. |
| REQ-1.2 | Resource Inventory | Admins can add, update, and delete resources from the inventory. |
| REQ-1.3 | Resource Request | Users can submit resource requests, which are sent to Admins for approval. |
| REQ-1.4 | Approval Workflow | Admins can approve or reject resource requests. |
| REQ-1.5 | Predictive Analytics | The system analyzes historical data to predict future resource demand. |
| REQ-1.6 | Geospatial Mapping | The system uses Google Maps API to visualize resource locations. |
| REQ-1.7 | Role-Based Access Control | Different user roles (Admin, Donor, User) have different access permissions. |

### 6. Functional Requirement REQ-1.1: User Authentication

**6.1 Description**  
Users must authenticate themselves using a username and password to access the system.

**6.2 System Input**

* Username and password.

**6.3 Display**

* Login screen with fields for username and password.

**6.4 System Processing**

* Validate user credentials against the database.

**6.5 System Output**

* Grant access to the system if credentials are valid.

**6.6 Constraints**

* Passwords must be encrypted.

**6.7 Data Handling**

* Username and password must be validated.

**6.8 Error Handling**

* Display an error message if login fails.

### 7. External Interface Requirements

**7.1 Data Interfaces**

* The system will interact with a MySQL/SQLite database for data storage.

**7.2 User Interfaces**

* The system will have a web-based interface accessible via modern browsers.

**7.3 Other Interfaces**

* The system will integrate with the Google Maps API for geospatial mapping.

### 8. Non-Functional Requirements

**8.1 System Performance**

* **PERF-1.1:** Login should take less than 1 second.
* **PERF-1.2:** Resource request submission should take less than 3 seconds.
* **PERF-1.3:** Predictive analytics processing should take less than 7 seconds.

**8.2 Information Security**

* **SEC-1.1:** User credentials must be encrypted during transmission.
* **SEC-1.2:** Data must not be stored in untested environments.

**8.3 Availability**

* **AVA-1.1:** The system must be available 99.9% of the time.

**8.4 Capacity**

* **CAP-1.1:** The system must support 100 concurrent users in year one.

**8.5 Software System Attributes**

* Security: JWT or Spring Security
* Usability: Intuitive frontend
* Scalability: Can add new features easily
* Reliability: Error handling and fallback mechanisms

**8.6 Database Requirements**

* MySQL relational database
* Tables: Users, Resources, Requests
* Foreign key relationships between requests and resources

### 9. Appendices

* ER Diagrams
* UI mockups