📄 **Resourcify SDS (Software Design Specification)**

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

### 1. Introduction

**Purpose**

This SDS describes the architectural design and detailed component design of the Resourcify system.

**Scope**

Covers backend architecture, frontend architecture, database design, WebSocket integration, and security considerations.

**Definitions, Acronyms, and Abbreviations**

(Refer to SRS)

**References**

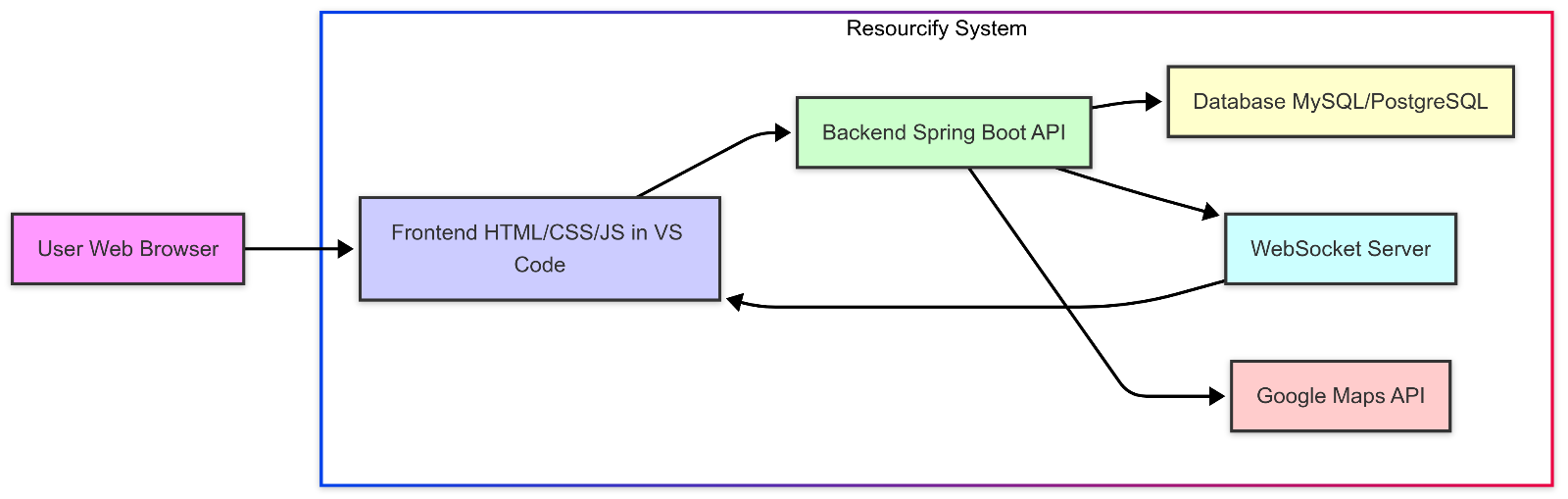
* Gao, P., & Thierer, A. (2018). "Smart resource allocation: The role of digital platforms in optimizing supply chains."
* Smith, J. (2020). "Data-driven decision-making in resource management."

**2. System Architecture Design**

**Overview**

* Backend: Spring Boot (REST APIs, WebSocket, JPA for persistence)
* Frontend: HTML, CSS, JavaScript
* Database: MySQL
* External APIs: Google Maps, SockJS, STOMP

**Architectural Diagram**

****

**Design Patterns**

* MVC for backend
* Observer pattern for WebSockets
* Singleton for DB connection (handled by Spring Boot)

**3. Component Design**

| **Component** | **Responsibility** |
| --- | --- |
| AuthController | User authentication & authorization |
| ResourceController | CRUD operations on resources |
| ResourceRequestController | Submit/view resource requests |
| WebSocketConfig | Configure WebSocket endpoints |

**Component Diagrams**

* REST controllers
* Services (business logic)
* Repositories (database access)

**Pseudocode/Flowcharts**

**Submit Request Flow:**

User submits request --> Validate resource quantity -->

Deduct quantity --> Save request --> Send WebSocket notification

**4. Interface Design**

**User Interfaces (Based on Project Files)**

* Home.html
* About.html
* Services.html
* Terms.html
* Login.html
* dashboard.html

**System Interfaces**

* REST APIs (JSON format)
* WebSocket (/ws) endpoint
* External: Google Maps API

**5. Data Design**

**Data Structures**

* ResourceItem (id, name, description, quantity, location)
* UserRequest (id, resourceId, userId, status, requestDate)

**Database Design**

* ERD connecting Users, Resources, Requests
* Relationships: One resource can have many requests

**6. Algorithm Design**

**Algorithms**

* Request validation and deduction
* Real-time notification via WebSocket
* Resource availability check

**Performance Considerations**

* Indexing on resource name and location
* Lazy loading where necessary

**7. Security and Privacy Considerations**

* Password encryption (BCrypt)
* Role-based access control
* CSRF protection (Spring Security)
* Data privacy on sensitive user info

**8. Error Handling and Logging**

* Custom exception handlers
* Logs stored using Spring Boot logging
* WebSocket errors logged separately

**9. Performance Considerations**

* Asynchronous calls where applicable
* Pagination for large datasets
* Efficient WebSocket handling

**10. Testing Strategy**

* Unit tests for service layer
* Integration tests for API endpoints
* Frontend testing via manual QA
* WebSocket testing with mock clients

**11. Deployment Strategy**

* Local deployment on Apache NetBeans
* Future: Docker containerization
* MySQL and Spring Boot dependencies

**12. Maintenance and Updates**

* Git for version control
* CI/CD pipeline for deployment (future)
* Bug tracking via issue boards

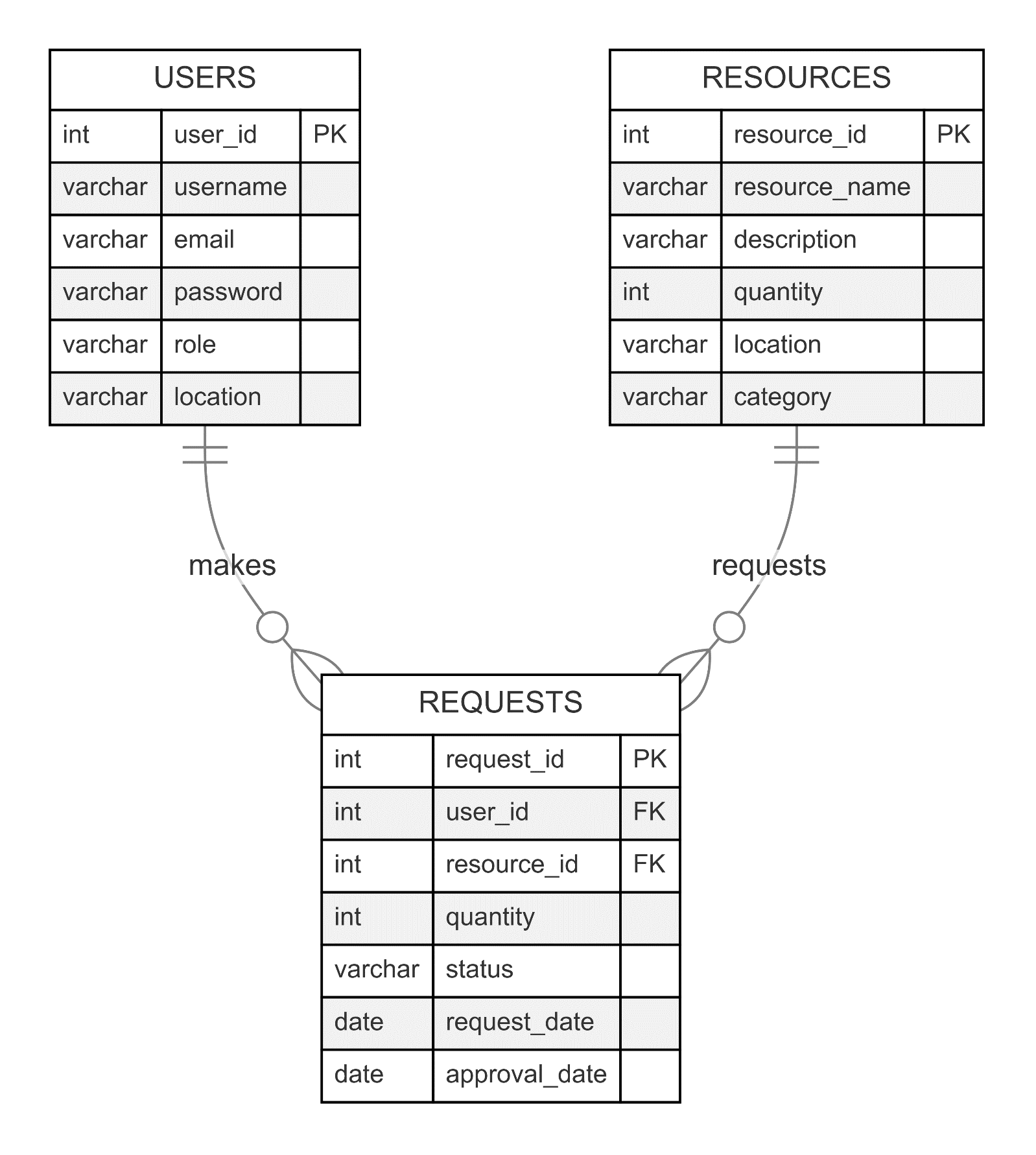
**13. Appendices**

* Sequence diagrams
* Screenshots of frontend
* API collections

**Diagrams (Embedded/Attached)**

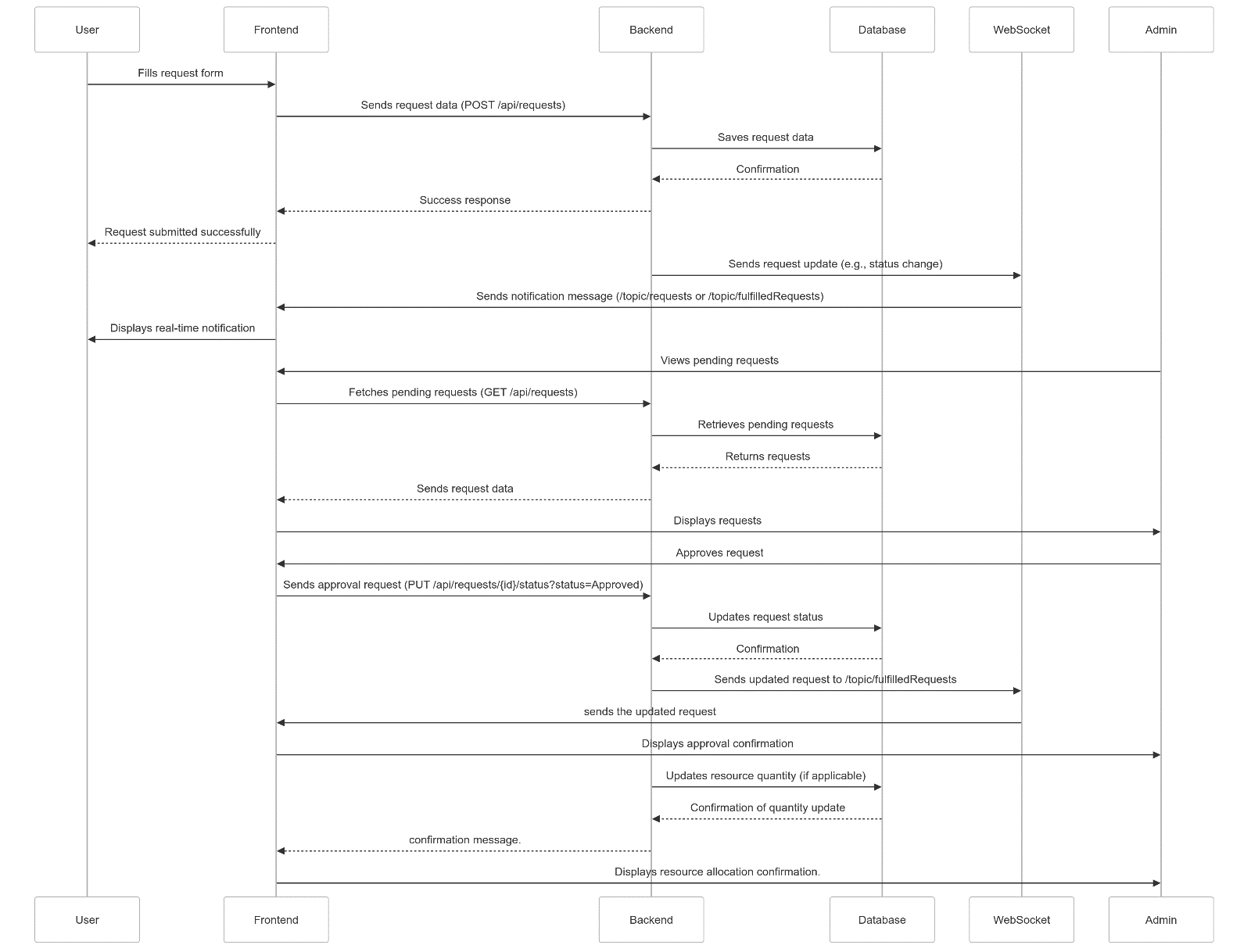
**ER Diagram**

(ERD connecting Users, Resources, Requests)

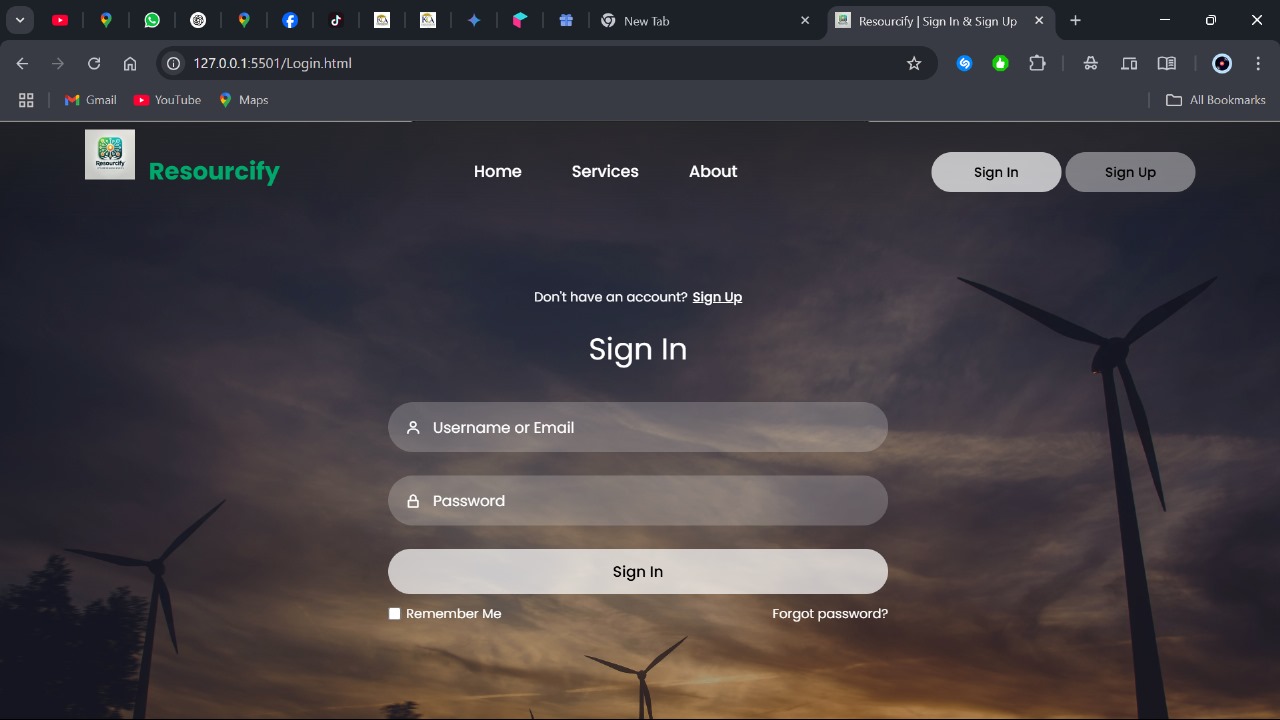


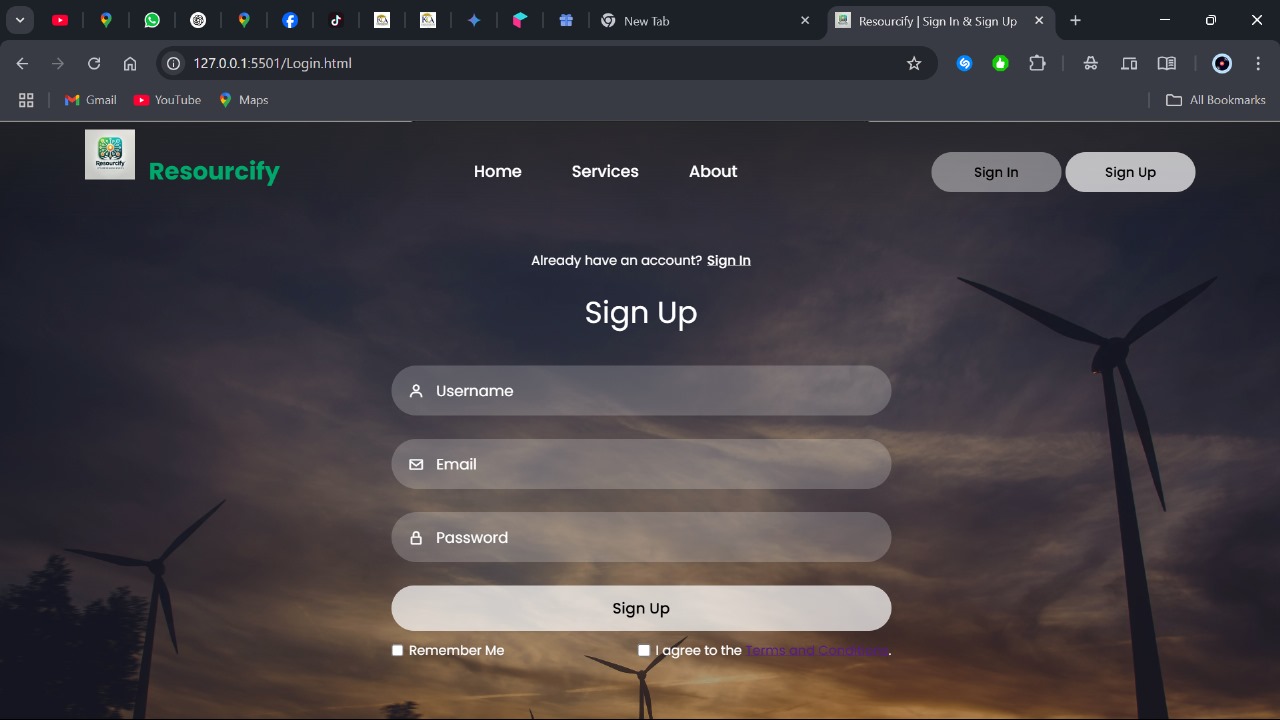
**Sequence Diagram**

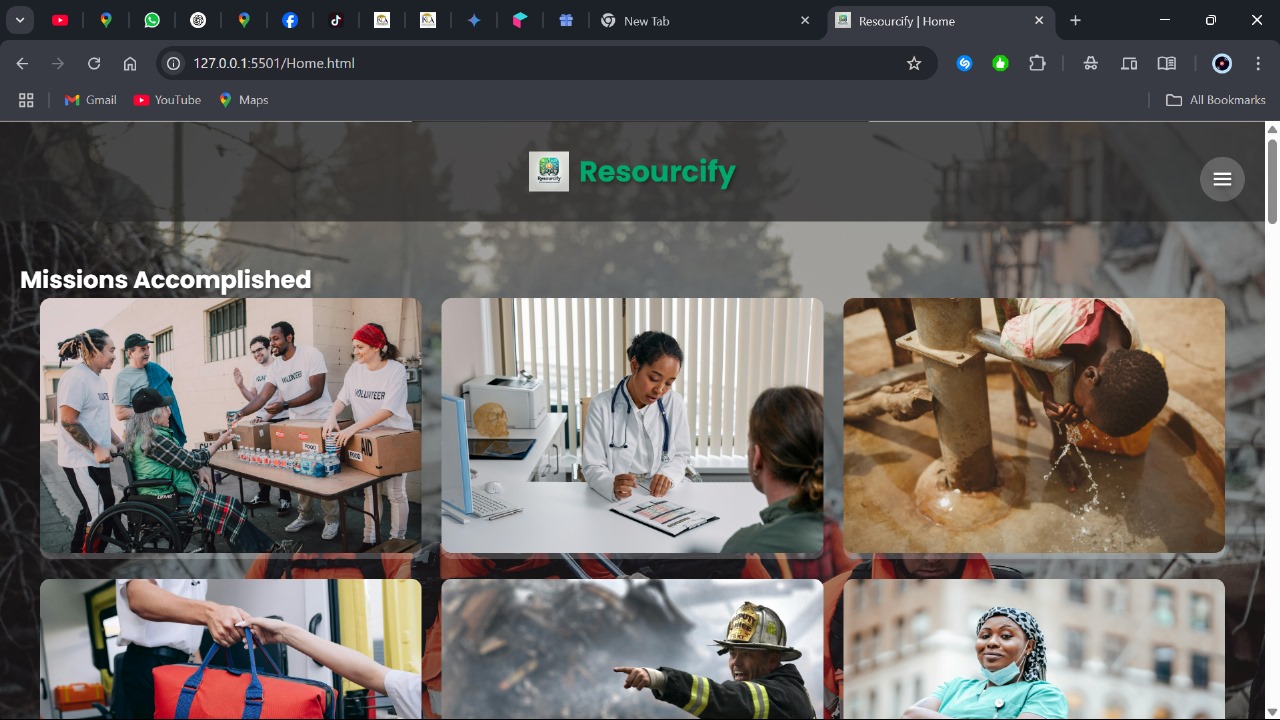
1. Submit Request
2. Real-time Notifications
3. Resource Allocation

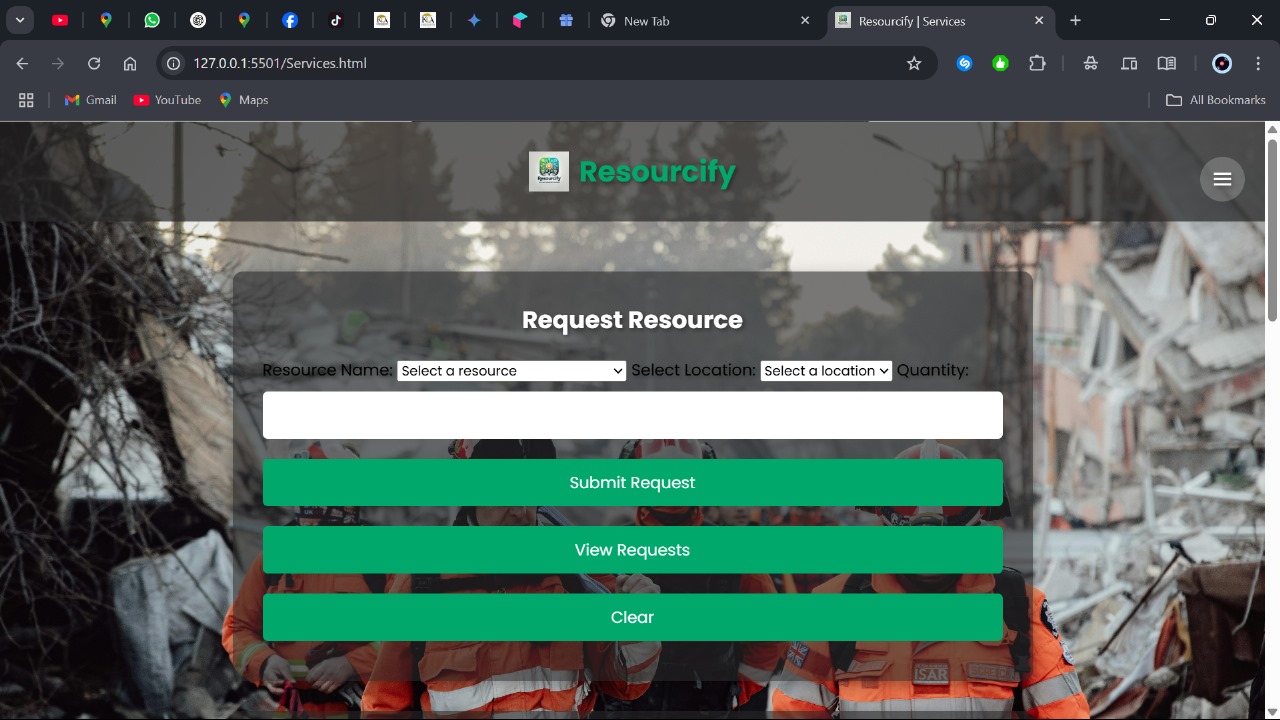


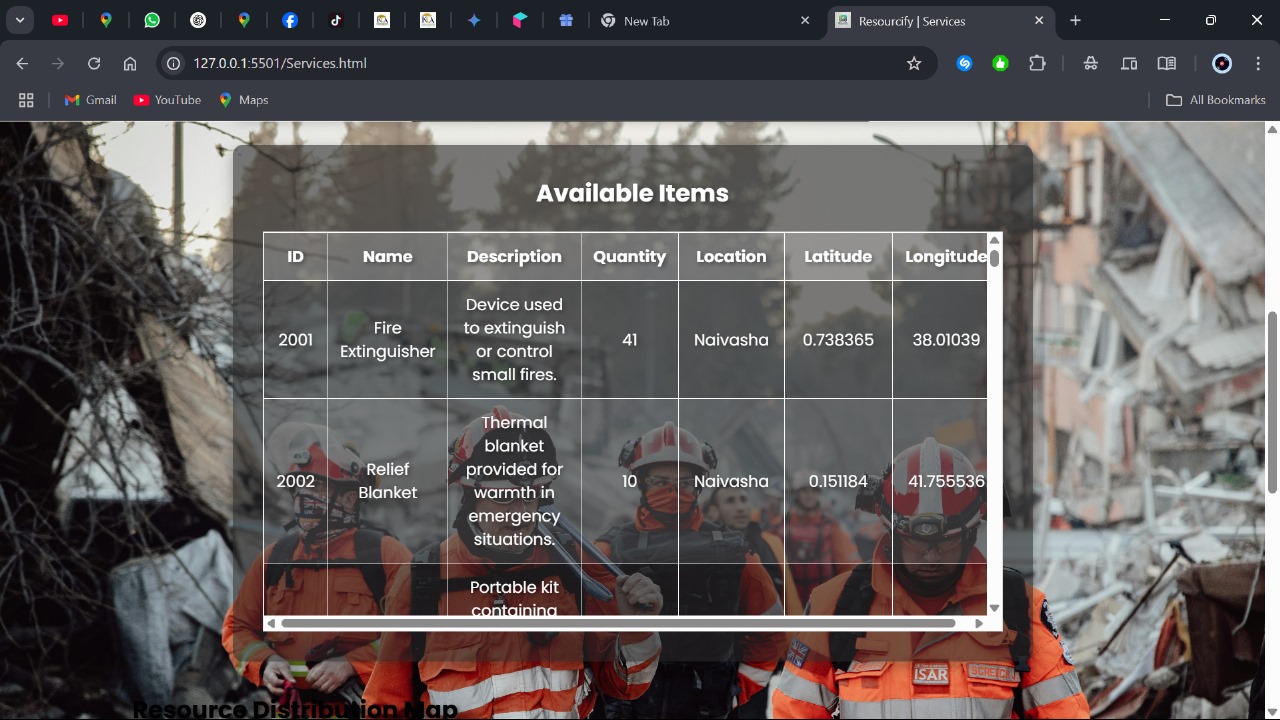
### Screenshots of Frontend

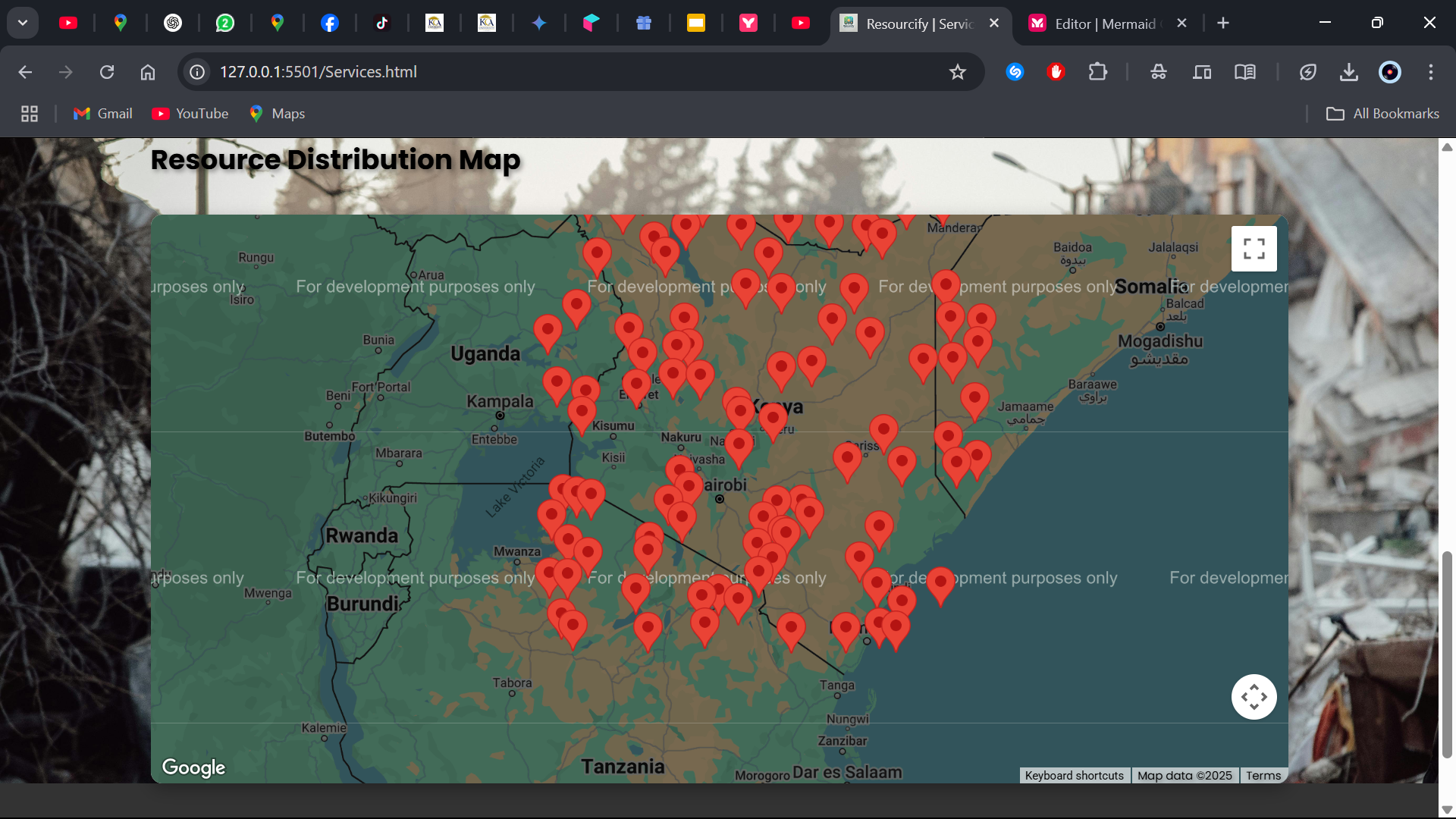


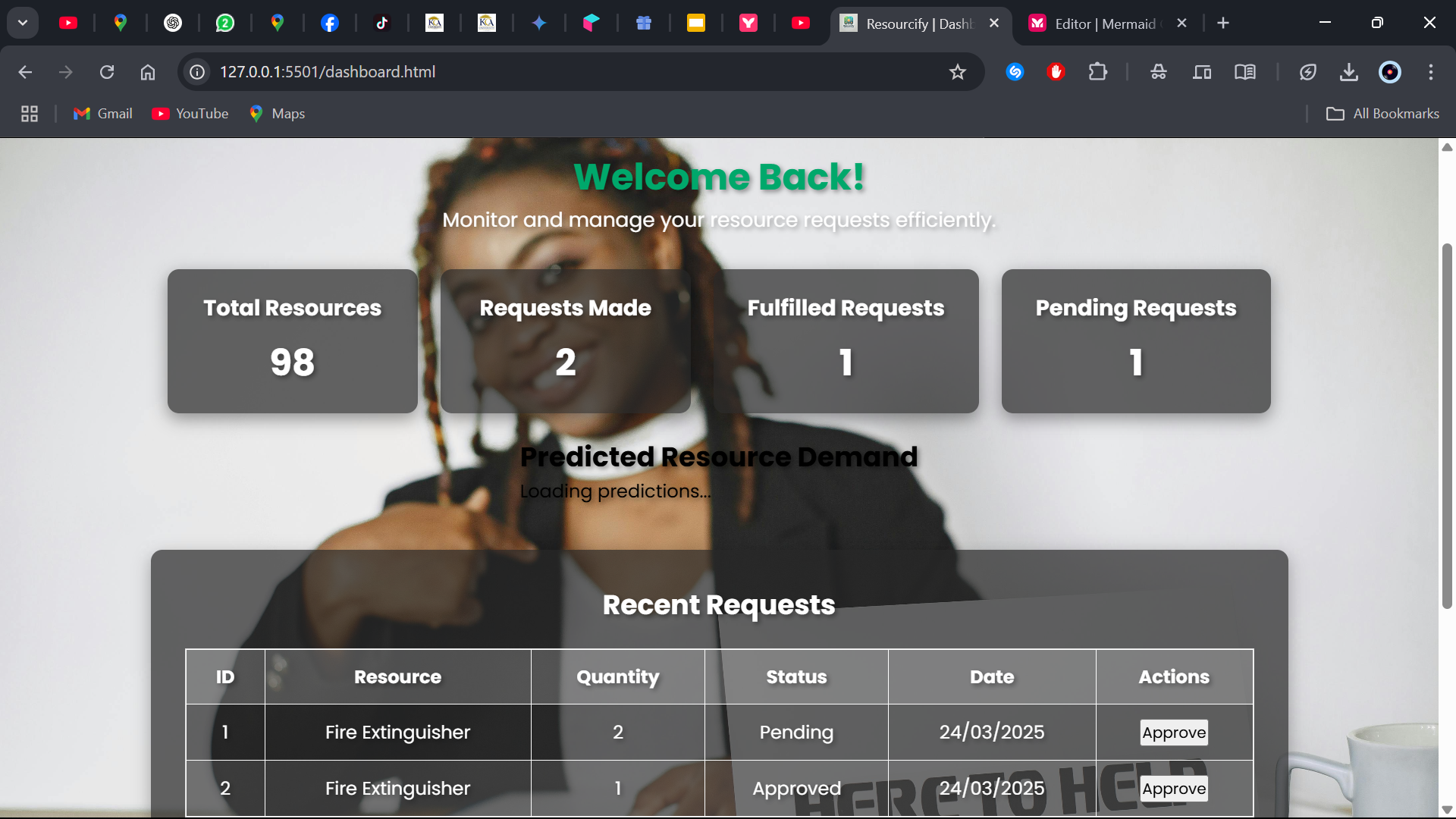


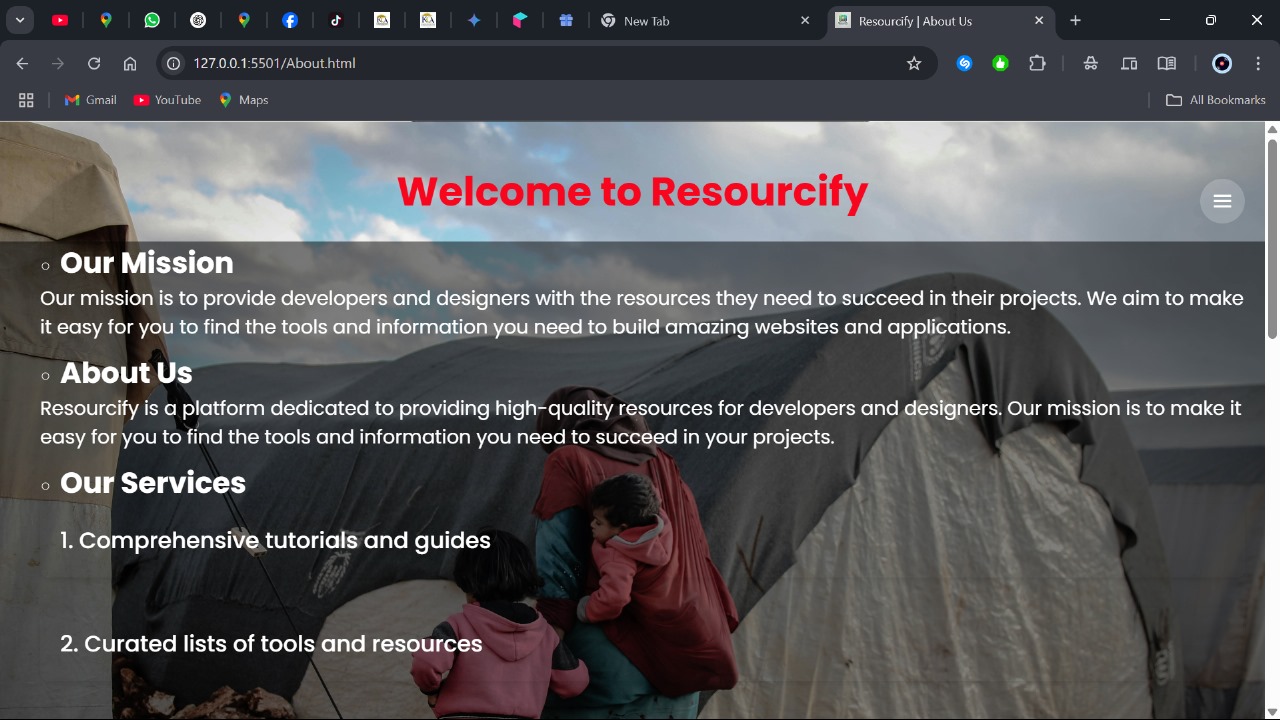


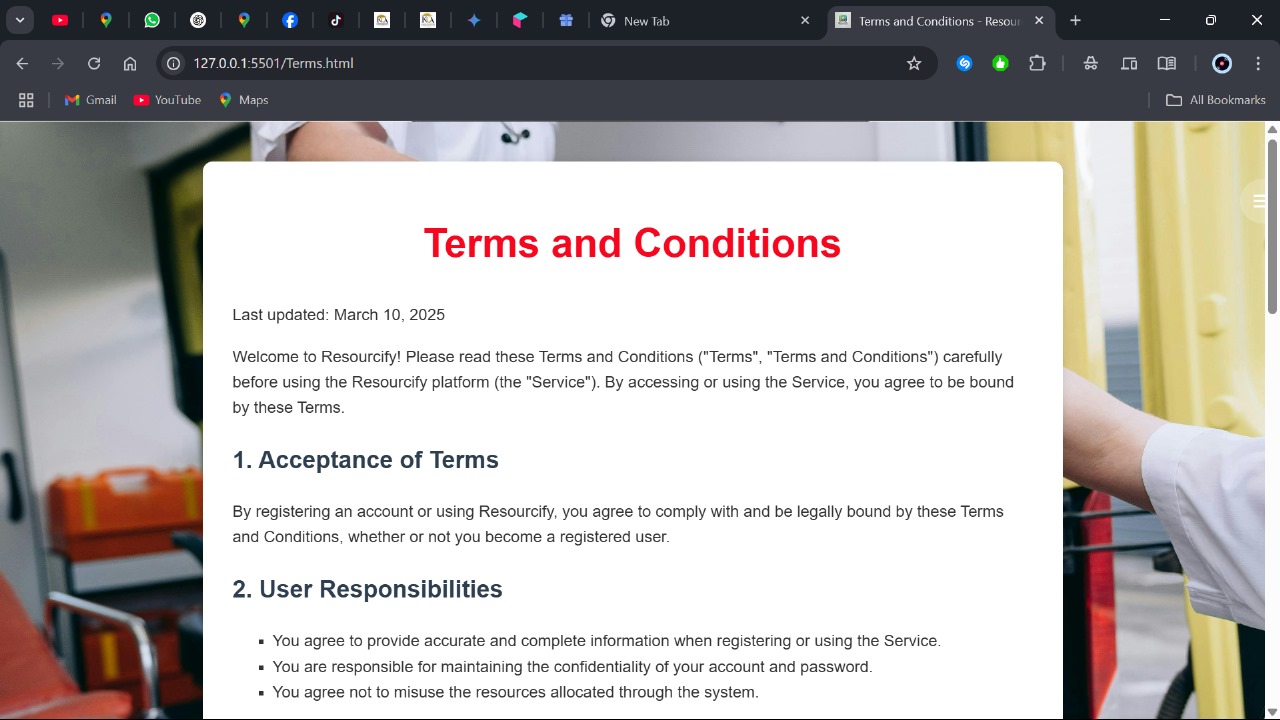












**API Endpoints Documentation**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/auth/register | POST | Register a new user |
| /api/auth/login | POST | Login and receive token |
| /api/resources | GET | Get all resources |
| /api/resources/{id} | GET | Get resource by ID |
| /api/resources | POST | Add a new resource |
| /api/resources/{id} | PUT | Update resource |
| /api/resources/{id} | DELETE | Delete resource |
| /api/requests | GET | Get all requests |
| /api/requests/{id} | GET | Get request by ID |
| /api/requests | POST | Submit a new request |
| /ws | WS | WebSocket endpoint for notifications |