Software Requirements Specification

for

Lost and Found Community Platform

**Version 1.0 approved**

**Prepared by <Emir Çiçek / Aziz Türker>**

**<09.11.2025>**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Emir Çiçek  Aziz Türker | 9/11/2025 |  |  |
|  |  |  |  |

# **Software Requirements Specification (SRS) – Lost and Found Community Platform**

## **1. Introduction**

### **1.1 Purpose**

The purpose of this document is to define all software requirements for the **Lost and Found Community Platform**. It clearly describes the system’s goals, functionalities, and scope to ensure a shared understanding among developers, administrators, and end users.

The platform aims to connect individuals who have lost items with those who have found them in a secure, fast, and efficient way. This reduces stress and time loss while promoting community awareness and cooperation.

### **1.2 Document Conventions**

This document follows the IEEE SRS standard format. Section numbering and headings organize information logically. Technical terms related to web applications, databases, and communication protocols are presented in standard English.

### **1.3 Intended Audience and Reading Suggestions**

* **Developers:** Focus on system functionalities and interfaces (Sections 3–4).
* **Administrators/Project Managers:** Focus on system scope, operations, and nonfunctional requirements (Sections 2 and 5).
* **End Users:** May review overall description, product functions, and guidelines for usage.

### **1.4 Product Scope**

**Lost and Found Community Platform** is an online system developed to facilitate the return of lost items to their rightful owners. Users can post reports of lost or found items with images and details, search for items, and communicate securely with other users.

The system employs a matching algorithm and filtering options (category, date, location) to suggest potential matches quickly. Admins can monitor content and ensure authenticity. The platform operates on web browsers and mobile devices, supporting community-driven recovery of lost items.

### **1.5 References**

* IEEE SRS Standards
* Web development standards (HTML, CSS, JavaScript)
* Database design principles (MySQL, PostgreSQL)
* Security and data protection guidelines (GDPR, encryption methods)

## **2. Overall Description**

### **2.1 Product Perspective**

The platform is a web-based, independent application with a frontend-backend architecture. It can run in any modern web browser and interact with a central database. Future integration with external systems, such as local municipal lost-and-found databases or mobile apps, is possible via APIs.

### **2.2 Product Functions**

Main functions include:

* Posting lost or found items with images, descriptions, and locations
* Searching and filtering items by category, date, and location
* Suggesting potential matches using an algorithm
* Enabling secure messaging between users
* Admin moderation of users and content
* Map-based item searches for nearby locations

### **2.3 User Classes and Characteristics**

* **Lost Item Owners:** Users reporting lost items
* **Finders:** Users reporting found items
* **Administrators:** Monitor content, verify reports, ensure platform integrity
* **Guests:** Browse public listings without registration

### **2.4 Operating Environment**

* Standard web browsers: Chrome, Firefox, Edge, Safari
* Desktop and mobile devices
* Backend server using Node.js or Python with MySQL/PostgreSQL

### **2.5 Design and Implementation Constraints**

* Use open-source frameworks
* Compliance with GDPR and privacy standards
* Internet access required for full functionality

### **2.6 User Documentation**

Includes online tutorials, guides, and FAQ sections explaining reporting procedures, search tools, messaging, and item verification processes.

### **2.7 Assumptions and Dependencies**

* Users have basic computer and internet skills
* Stable web hosting, database functionality, and active server connection required

## **3. External Interface Requirements**

### **3.1 User Interfaces**

* Login/registration page
* Lost/found item posting forms
* Search and filter dashboard
* Messaging interface
* Map view for item locations

### **3.2 Hardware Interfaces**

No special hardware required; any device with a modern browser is sufficient.

### **3.3 Software Interfaces**

* Backend databases for storing item reports
* REST APIs for data exchange and user authentication
* Potential future integration with mobile apps or LMS systems

### **3.4 Communications Interfaces**

* HTTPS for secure data transfer
* Optional WebSocket for real-time messaging

## **4. System Features**

### **4.1 Item Reporting and Matching**

* Users can report lost or found items with details and images
* System provides automatic suggestions of potential matches  
   **Priority:** High

### **4.2 Secure Messaging**

* Facilitates communication between item owners and finders
* Optional anonymous messaging or verification codes for added security

### **4.3 Administration Tools**

* Admins can moderate content, verify reports, manage users, and generate reports

### **4.4 Map-Based Search**

* Users can locate nearby items using integrated map view and filters

## **5. Other Nonfunctional Requirements**

### **5.1 Performance Requirements**

* Support at least 1000 concurrent users
* Item searches and listings load in under 3 seconds
* Images optimized for fast upload and display

### **5.2 Safety Requirements**

* Daily backups of user and system data
* Error handling prevents data loss

### **5.3 Security Requirements**

* All communication over HTTPS
* Personal data encrypted and securely stored
* Reporting mechanism for fraudulent content
* Optional two-factor authentication (2FA)

### **5.4 Software Quality Attributes**

* Reliable, user-friendly, maintainable, and portable
* Accessible for users of all backgrounds

### **5.5 Business Rules**

* Only registered users can post or claim items
* Admins oversee platform operations

## **6. Other Requirements**

* Multi-language support (initially English and Turkish)
* Future updates may include AI-assisted item matching, notifications, and community features

*Mission: A gamified SQL problem in the system.*

*Badge: A reward for completing a challenge.*

*LMS: Learning Management System used in educational environments.*

## *Appendix B: Analysis Models*

*Use case and class diagrams describe the relationship between users, missions, and the query engine. Entity-relationship diagrams define the game database design.*

## *Appendix C: To Be Determined List*

* *Final design of UI themes and storylines*
* *Selection of backend framework (Flask/Django)*
* *Hosting provider for production release*