**Software Requirements Specification**

**Version 1.0**

**April 15, 2025**

**Arranging Homestays and Cultural Exchange Platform**

**Authors:**

**Abdelrahman Badawi**

**Faisal Mahmoud**

**Mahmoud Saeid**

**Malak Hany**

**Malak Ragab**

Table of Contents

[1 Introduction 3](#_Toc195655703)

[1.1 Purpose 3](#_Toc195655704)

[1.2 Project Scope 3](#_Toc195655705)

[1.3 Glossary and Abbreviations 3](#_Toc195655706)

[1.4 List of System Stakeholders 4](#_Toc195655707)

[1.5 References 4](#_Toc195655708)

[2 Functional Requirements 5](#_Toc195655709)

[2.1 User Requirements Specification 5](#_Toc195655710)

[2.2 System Requirements Specification 6](#_Toc195655711)

[2.3 Requirements’ Priorities 6](#_Toc195655712)

[3 Non-functional Requirements 7](#_Toc195655713)

[3.1 General Types/Categories 7](#_Toc195655714)

[3.2 Specification 7](#_Toc195655715)

[3.3 Fit Criteria 8](#_Toc195655716)

[3.4 Effect on Architecture 8](#_Toc195655717)

[4 Design & Implementation Constraints 8](#_Toc195655718)

[5 System Evolution 8](#_Toc195655719)

[5.1 Anticipated Changes 8](#_Toc195655720)

[5.2 Effect on Design 9](#_Toc195655721)

[6 Requirements Discovery Approaches 9](#_Toc195655722)

[7 Requirements Validation Techniques 9](#_Toc195655723)

# 1 Introduction

## Purpose

The purpose of this document is to present a detailed description of the Arranging Homestays and Cultural Exchange Platform. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

## Project Scope

This project involves the development of a platform designed to facilitate homestays and cultural exchanges between hosts and travelers. The system aims to connect individuals globally, encouraging cultural immersion, skill sharing, and budget-friendly travel solutions. It allows hosts to specify their needs and accommodations while enabling travelers to create profiles showcasing their skills and preferences. The platform supports secure communication and efficient management of arrangements, fostering meaningful interactions between diverse communities by promoting cultural exchange, offering affordable travel options through skill-based contributions, supporting hosts in isolated or underserved communities, providing flexible arrangements for both short-term and long-term stays, and streamlining profile creation, communication, and arrangement management.

## Glossary and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Active Article | The document that is tracked by the system; it is a narrative that is planned to be posted to the public website. |
| Author | Person submitting an article to be reviewed. In case of multiple authors, this term refers to the *principal author*, with whom all communication is made. |
| Database | Collection of all the information monitored by this system. |
| Editor | Person who receives articles, sends articles for review, and makes final judgments for publications. |
| Field | A cell within a form. |
| Historical Society Database | The existing membership database (also HS database). |
| Member | A member of the Historical Society listed in the HS database. |
| Reader | Anyone visiting the site to read articles. |
| Review | A written recommendation about the appropriateness of an article for publication; may include suggestions for improvement. |
| Reviewer | A person that examines an article and has the ability to recommend approval of the article for publication or to request that changes be made in the article. |
| Software Requirements Specification | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Stakeholder | Any person with an interest in the project who is not a developer. |
| User | Reviewer or Author. |

## List of System Stakeholders

1. Travelers/Volunteers: Individuals looking for affordable accommodations and cultural immersion experiences. They contribute their time and skills in exchange for food and lodging.

2. Hosts: People or families offering accommodations and seeking assistance with specific tasks. They are central to the platform's operations.

3. Platform Administrators: Responsible for managing the platform, maintaining functionality, overseeing user activity, and ensuring smooth interactions.

4. Communities: Local communities hosting travelers, benefiting from the cultural exchange and skill-sharing aspect.

5. Language Learners: Both hosts and travelers aiming to improve their foreign language skills through immersive experiences.

6. Security and Verification Entities: Stakeholders involved in verifying user profiles, ensuring secure payments, and promoting safe interactions.

7. Potential Sponsors/Advertisers: Organizations or brands interested in partnering with the platform to promote services aligned with the platform's goals.

8. Government and Regulatory Authorities: Monitor legal compliance, data security, and public safety aspects of such operations.

## References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

<https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc>

<https://www.grammarly.com>

# Functional Requirements

## User Requirements Specification

**2.1.1 User Registration & Profiles**

* Travelers and hosts must be able to create accounts with personal details (e.g., name, skills, preferences).
* Hosts should describe their accommodation, required help, and expectations.
* Travelers must be able to edit or delete the information in the profiles

**2.1.2 Search & Matching**

* Travelers can search/filter hosts by location, type of work (e.g., gardening, farming), duration, and amenities.
* Hosts can search for travelers based on skills, availability, and interests.

**2.1.3 Communication**

* Secure messaging system for travelers and hosts to discuss arrangements.

**2.1.4 Agreement Management**

* Platform should allow users to formalize agreements (e.g., hours/day, duration, lodging details).

**2.1.5 Membership & Payments**

* Travelers pay a yearly fee to access host listings; hosts list for free.
* Payment gateway integration (e.g., credit card, PayPal).

**2.1.6 Reviews & Ratings**

* Both travelers and hosts can rate/review each other after exchanges.
* Hosts must be able to accept or reject applications

**2.1.7 Safety & Verification**

* ID verification for users (e.g., passport, phone number).
* Reporting system for suspicious activity.

**2.1.8 Localization**

* Multilingual support for global users.

## System Requirements Specification

**Backend**

* Database to store user profiles, listings, messages, and agreements.
* Algorithm for matching travelers/hosts based on preferences.

**Frontend**

* Responsive UI for web/mobile (e.g., React, Flutter).
* Intuitive forms for profile creation and search filters.

**Admin Panel**

* Dashboard to manage users, content and reports ,resolve disputes and monitor reviews.

## Requirements’ Priorities

|  |  |  |
| --- | --- | --- |
| **Priority** | **Requirement** | **Category** |
| MUST | User registration/profile creation | Core functionality |
| MUST | Search/filter hosts/travelers | Core functionality |
| MUST | Secure messaging system | Core functionality |
| MUST | Payment processing for travelers | Core functionality |
| SHOULD | ID verification & safety features | Trust/Safety |
| SHOULD | Review/rating system | Quality assurance |
| COULD | Multilingual support | Enhanced UX |

# Non-functional Requirements

## General Types/Categories

**The following categories will be addressed in this system:**

* Look and feel: The spirit of platform appearance
* Performance: Defines system speed and response times.
* Security: Focuses on protecting data and ensuring secure operations.
* Scalability: Ensures the system can handle increased user load or data volume efficiently.
* Usability: Addresses the user-friendliness and accessibility of the platform.
* Availability: Defines system uptime and continuity of service.
* Maintainability: Outlines ease of updating or repairing the system.
* Legal: The laws and standard that apply to the product
* Portability: Covers the ability of the system to operate across different platforms and devices

## Specification

**Each requirement is specified under its respective category:**

1. Look and feel

* The platform should only use three colors

2. Performance

* The platform must respond to user interactions within 2 seconds for 95% of cases under a load of 500 concurrent users.

3. Security

* All user data must be encrypted using AES-256 both at rest and during transmission.

4. Scalability

* The system must support up to 1 million user profiles without degradation of performance.

5. Usability

* The platform must be accessible to users with visual impairments, supporting screen readers and keyboard navigation.

6. Availability

* The system must maintain 99% uptime, with no single downtime exceeding 5 hours per year

7. Maintainability

* Updates to the system should not require more than 3 hours of downtime and should be supported by automated deployment tools.

8. Portability

* The platform must be compatible with the latest versions of commonly used web browsers (Chrome, Firefox, Safari, Edge) and operate on Android and iOS devices

## Fit Criteria

## Effect on Architecture

# Design & Implementation Constraints

* **a) Technology Stack Constraints**

**Definition:** These constraints define the programming languages, tools, and frameworks that **must** or **must not** be used throughout the system's development process.

**Examples:**

* **Mandatory Use of Specific Technologies:**
  + The backend of the application must be developed using **PHP 8** and follow **Object-Oriented Programming (OOP)** principles.
  + The frontend should use **HTML5**, **CSS3**, and **JavaScript** with the **Bootstrap 5** framework for responsive design.
  + The database should be **MySQL**.
* **Prohibited Technologies:**
  + Usage of outdated technologies such as **PHP 5.x** or insecure libraries is not allowed.

**Rationale:** Defining a consistent tech stack ensures smooth integration, better maintainability, and allows team members to work efficiently using familiar technologies.

* **b) Compliance Constraints**

**Definition:** These constraints ensure that the application follows **legal**, **ethical**, and **organizational** standards related to user data and system security.

**Examples:**

* **Data Privacy Regulations:**
  + The system must comply with **GDPR** (General Data Protection Regulation) when handling personal information like user profiles, messages, and location data.
* **Age Restriction:**
  + Users under the age of 18 should not be able to register as travelers or hosts.
* **Terms of Use and Consent:**
  + Users must accept the terms and conditions before accessing platform features.

**Rationale:** Compliance with these constraints helps avoid legal issues, protects users' rights, and ensures ethical system behavior.

* **c) Hardware Limitations**

**Definition:** These constraints involve the **minimum hardware capabilities** required to run the system effectively on different user devices.

**Examples:**

* **Minimum Device Requirements:**
  + The web app must operate smoothly on devices with at least **2GB RAM** and **dual-core processors**.
* **Browser and Platform Compatibility:**
  + The platform should work properly on modern browsers such as **Google Chrome**, **Firefox**, and **Microsoft Edge**.
  + Mobile responsiveness must support both **Android 9+** and **iOS 12+** via responsive web design (not native apps).

**Rationale:** By considering hardware limitations, we ensure that a broad audience—including budget travelers—can access the platform without performance issues.

# System Evolution

As platforms grow, they must **evolve** to meet user needs, improve security, and add features. The following outlines planned **future upgrades** and how they affect the **system's architecture**.

## Anticipated Changes

**5.1.1 Adding Host Verification through Government ID:**

* **What it is:** Hosts will be required to upload a valid government-issued ID for identity verification.
* **Purpose:** Increases trust and safety on the platform.
* **Example:** Similar to Airbnb's ID verification process [Airbnb Help Center](https://www.airbnb.com/help/all-topics).

**5.1.2 Mobile App with Offline Messaging :**

* **What it is:** A dedicated mobile app will allow volunteers and hosts to message each other even without internet, syncing messages when reconnected.
* **Purpose:** Supports users in remote or rural areas.
* **Example:** WhatsApp and Signal offer similar offline message queuing.

**5.1.3 Integration with Travel Insurance APIs:**

* **What it is:** The platform will integrate with third-party travel insurance providers (e.g., Safety Wing, World Nomads) to offer coverage options to travelers.
* **Purpose:** Adds value and ensures travelers are protected in case of injury, theft, etc.
* **Example:** APIs like Safety Wing’s Nomad Insurance can be embedded in booking flows [Safety Wing](https://safetywing.com/).

## Effect on Design

These changes will **impact the system's architecture** in the following ways:

**5.2.1. Plug-and-Play Authentication Providers :**

* The system must allow **easy integration** of different authentication methods like:
  + Government ID verification
  + Social logins (Google, Facebook)
* **Solution:** Use **OAuth 2.0 / OpenID Connect**, and design with **modular authentication** layers.
* **Why:** Makes it easy to add or remove authentication services without reworking the core system.

**5.2.2. Modular Profile System with Expandable Data Schema :**

* Profiles must be designed to **store additional fields** in the future (e.g., ID verification status, insurance details, app settings).
* **Solution:** Use a **schema-less database** (e.g., MongoDB) or design the relational schema to allow optional, versioned fields.
* **Why:** This supports evolving requirements without needing frequent database overhauls.

**5.2.3. API Versioning for Backward Compatibility :**

* APIs must support **multiple versions** to avoid breaking functionality for users on older mobile apps or integrations.
* **Solution:** Use versioned endpoints like /API/v1/users, /API/v2/users.
* **Why:** Ensures existing users aren't forced to update immediately when new features roll out.
* **Reference:** [Microsoft API Versioning Guidelines](https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design#versioning-a-restful-web-api)

# Requirements Discovery Approaches

# Requirements Validation Techniques

The goal of **requirements validation** is to ensure that all requirements are:

* Correct
* Complete
* Unambiguous
* Verifiable
* Feasible
* Aligned with stakeholder needs

**7. 1. Reviews:**

**Type: *Structured walkthrough or formal inspection [systematic review of a software document]***

* **What it does**: Involves reviewing the requirements document in a structured session with developers, analysts, and stakeholders.
* **Example**:  
  Suppose the requirement is:

"A traveler should be able to send a request to a host and communicate through the platform."

During review:

* + Is this clear enough?
  + What about [unusual] edge cases (unverified users trying to message hosts)?
  + Do we need to define communication limits?

The team may redefine the requirement or add constraints.

Source: [IEEE 830 - Software Requirements Specification Guidelines](https://ieeexplore.ieee.org/document/720574)

7.2.  **Prototyping:**

* **What it does**: Builds low-fidelity mockups [**basic visual representations**] of key parts of the system (like UI) to validate understanding.
* **Example**:  
  A prototype of the traveler profile page is shared with test users.
  + Users might suggest adding filters ("search hosts by help type or region").
  + Based on the feedback, you revise/add requirements.

📘 Source: [IEEE Standard 1058 - Software Project Management Plans](https://ieeexplore.ieee.org/document/741876)

7.3. Use Case Validation:

* **What it does**: Breaks down real-life interactions into step-by-step scenarios to see if requirements cover all cases.
* **Example**:  
  Use case: *Traveler contacts a host*  
  Steps:
  1. Traveler logs in
  2. Searches hosts
  3. Views host profile
  4. Sends request message
  5. Awaits reply

Stakeholders verify: Is that all that’s needed? Any missing features like notifications or language filters?

7.4. Requirements Traceability Matrix (RTM):

* **What it does**: Maps every requirement to its origin (business goal, stakeholder request) and ensures there’s a corresponding test case.
* **Example**:  
  Requirement: "Travelers must pay a yearly membership fee to contact hosts."

It’s mapped to:

* + Business objective: generate revenue [profit]
  + Test case: block messaging functionality for non-paying users

7.5. Surveys and Interviews:

* **What it does**: Collects feedback from real users (hosts/travelers) to validate if the requirements align with real-world needs.
* **Example**:
  + Interview with a host: Does the requirement to "describe help needed" make sense? Are the input fields adequate [suitable] ?
  + Survey for travelers: Would they find it useful to search by length of stay, type of help, or host reviews?