# **📄 Software Requirements Specification (SRS)**

## **1. Introduction**

### **1.1 Purpose**

This document defines the requirements for developing *Skillshareing*, a peer-to-peer skill sharing platform. The platform enables users to share, discover, and learn skills from others in their community. It will support user registration, skill posting, search and booking, reviews, messaging, and notifications to ensure a smooth, interactive learning experience.

### **1.2 Intended Audience**

* **Developers**: For implementing the system as specified
* **Testers/QA**: For verifying the system meets requirements
* **Project Managers**: For tracking scope and progress
* **Instructors**: For evaluating project progress and completeness
* **Stakeholders**: For understanding system goals and functionality

### **1.3 Intended Use**

* As a contract for development, ensuring all features are delivered
* As a guide for testing, defining acceptance criteria
* As documentation for future maintenance or upgrades

### **1.4 Product Scope**

SkillLink is a web application enabling peer-to-peer skill sharing. Users can register as skill sharers or learners. Skill sharers can post and manage their skills. Learners can search for skills, send booking requests, and leave reviews. The system includes messaging and notifications for better coordination.

**Goals and Benefits:**

* Promote accessible, community-based learning
* Enable monetization of skills for providers
* Foster a trusted, review-driven learning ecosystem
* Support ease of communication and scheduling

### **1.5 Risk Definitions**

* **Data Privacy Risk**: User data could be exposed if security is weak.
* **Scalability Risk**: System might underperform with high user load.
* **Integration Risk**: APIs (e.g., for maps) may change or fail.
* **Timeline Risk**: Features might be rushed near deadlines, reducing quality.
* **Adoption Risk**: Users might not engage without good UX.

## 

## 

## **2. Overall Description**

### **2.1 User Classes and Characteristics**

| **User Class** | **Description** |
| --- | --- |
| **Skill Sharer** | Registers, lists skills, manages bookings, responds to messages, receives reviews |
| **Learner** | Registers, searches for skills, sends booking requests, leaves reviews, chats |
| **Admin (Optional)** | Monitors users, reviews, manages inappropriate content |

### **2.2 User Needs**

* Easy, secure sign-up and login
* Ability to list skills with details and pricing
* Search and filter skills efficiently
* Ability to request/book learning sessions
* Ability to communicate (messaging)
* Transparent feedback through ratings and reviews
* Notifications to stay updated about requests and messages
* Clean, intuitive, responsive UI

### **2.3 Operating Environment**

* Web browser (Chrome, Firefox, Safari, Edge)
* Frontend: React, Angular, Vue, or Laravel Blade
* Backend: Node.js/Express, Laravel, or Django
* Database: MySQL, PostgreSQL, or MongoDB
* Hosting: Heroku, Vercel, AWS, or local deployment
* Tools: GitHub (version control, wiki), Trello (task tracking), Slack/Discord (team communication)

### **2.4 Constraints**

* Must use MVC architecture
* Must have unit tests for core features
* Must use Git for version control with meaningful commits
* Must use approved frameworks/tools
* Must meet deadline and milestone reviews (Weeks 7, 9, 12, and final demo Week 14)
* Must avoid writing all documentation/comments at the end

### **2.5 Assumptions**

* Users have internet access and modern browsers
* Team members will use Trello, Slack, GitHub as collaboration tools
* Any APIs used (e.g., Google Maps) will remain available during development
* User base will start small but may need to scale
* Users will input data correctly, but basic validation will be implemented

## 

## **3. Requirements**

### **3.1 Functional Requirements**

**FR1. User Registration and Login**

* Users can register with email and password
* Login and logout securely
* User roles: Skill Sharer or Learner
* Passwords hashed in database

**FR2. Post and Manage Skills**

* Skill Sharers can:  
  + Add new skill listing with title, description, category, price, photo
  + Edit existing listings
  + Delete their listings

**FR3. Search and Filter Skills**

* Learners can:  
  + Search by keyword
  + Filter by category, location (optional), rating
  + Sort results

**FR4. Booking / Contact Requests**

* Learners can send booking/contact requests
* Skill Sharers can accept/decline requests
* Booking request status tracked in dashboard

**FR5. Reviews and Comments**

* Learners can leave ratings and text reviews after sessions
* Reviews visible on skill and user profile pages
* (Optional) Skill Sharers can respond to reviews

**FR6. User Profile Page**

* Public profile page for each user showing:  
  + Profile info and photo
  + Listed skills
  + Average rating
  + Reviews

**FR7. Messaging / Chat System**

* Users can send and receive text messages
* Messaging unlocked after booking is accepted
* (Optional) Notifications for new messages

**FR8. Notifications System**

* In-app or email notifications for:  
  + New booking requests
  + Booking status updates
  + New reviews
  + New messages

### **3.2 Non-Functional Requirements**

| **Category** | **Requirement** |
| --- | --- |
| **Performance** | Skill search should return results under 1 second. |
| **Scalability** | System should be able to support 1000+ users with basic scaling options. |
| **Security** | Passwords stored hashed; JWT or sessions for authentication; HTTPS used. |
| **Usability** | Responsive UI for desktop and mobile; intuitive navigation. |
| **Maintainability** | Code should be modular, documented, and follow MVC structure. |
| **Reliability** | System should handle invalid input gracefully and provide error messages. |
| **Compatibility** | Should work in major modern browsers. |

## **Notes for Team**

* All features must be versioned with Git and well-documented.
* Use GitHub Wiki for design details, setup instructions, and developer guidelines.
* Use Trello to track tasks and milestones.
* Use Slack/Discord for communication.

## **Appendices (for SDS)**

* Detailed architecture diagrams (MVC)
* ERD / Database Schema
* Sequence diagrams (optional)
* API design (if REST)