# Software Requirements Specification (SRS)

## SkillSwap Mobile Application

---

## 1. Introduction

### 1.1 Purpose

This Software Requirements Specification (SRS) defines the functional and non-functional requirements for the SkillSwap mobile application, a peer-to-peer platform designed to enable university students to exchange skills without monetary transactions. This document serves as a comprehensive guide for developers, designers, and stakeholders to understand the system's scope, features, and constraints.

### 1.2 Scope

SkillSwap is a mobile application that facilitates skill-sharing among university students, allowing them to act as both tutors (offering skills) and learners (seeking skills). The initial version includes the following functionalities:

- **User Profiles**: Creation and management of profiles with personal details and skill listings.
- **Skill Listings**: Posting and searching for skill offers.
- **Booking System**: Scheduling skill exchange sessions.
- **Ratings and Reviews**: Providing feedback on completed sessions.

**Out of Scope**:

- In-app payments or monetary transactions.
- In-app communication features (e.g., video calls, instant messaging).
- Advanced analytics or recommendation systems.

### 1.3 Definitions and Acronyms

- **Tutor**: A user offering a skill to teach.
- **Learner**: A user seeking to learn a skill.
- **SkillSwap**: The act of exchanging skills without monetary payment.
- **CRUD**: Create, Read, Update, Delete—standard database operations.
- **MVP**: Minimum Viable Product.

### 1.4 Assumptions and Constraints

- **Assumptions**:
    - Users have access to smartphones with reliable internet connectivity.
    - Users are university students familiar with basic mobile app navigation.
- **Constraints**:
    - The application must be compatible with iOS 13+ and Android 9+.
    - The initial version will not support real-time communication features.

## 2. Overall Description

### 2.1 User Roles

| Role | Key Permissions & Goals |
|---|---|
| **Student (Tutor/Learner)** | Can switch between tutor and learner roles, create/edit profiles, post/search skill offers, book sessions, leave ratings/reviews, and report inappropriate content. |
| **Admin** | Can view all user data, remove inappropriate content, suspend/delete user accounts, and handle reported content to maintain platform integrity. |

### 2.2 User Stories

- As a learner, I want to filter skill offers by category, rating, or location, so I can find a reliable tutor quickly.
- As a tutor, I want to set my availability, so learners can only book me when I'm free.
- As a student, I want to view my past and upcoming sessions, so I can manage my commitments.
- As a student, I want to report inappropriate skill offers or user behavior, so I can ensure a safe community.
- As an admin, I want to moderate skill offers and user accounts, so I can maintain a professional and safe platform.
- As a student, I want to receive notifications for session confirmations or cancellations, so I stay informed about my bookings.

### 2.3 Operating Environment

- **Platform**: iOS and Android.
- **Technology**: React Native for the frontend, MongoDB for the backend database, and Node.js for server-side logic (assumed for MVP).

- **Connectivity**: Requires a stable internet connection for data synchronization and notifications.

### 2.4 Constraints

- The system must prioritize simplicity to ensure usability for non-technical users.
- The database must support scalability for up to 10,000 users in the initial phase.

## 3. Functional Requirements

| ID | Requirement Description |
|---|---|
| FR1 | The system shall allow users to register an account using a valid email and password. |
| FR2 | The system shall allow registered users to log in and log out securely. |
| FR3 | The system shall allow users to create and edit profiles, including name, bio, profile picture, and a list of skills. |
| FR4 | The system shall allow users to create a skill offer with fields for title, description, category, and duration. |
| FR5 | The system shall allow users to search skill offers by keywords, categories, or user location. |
| FR6 | The system shall display a list of skill offers, including offer details and the tutor's profile information. |
| FR7 | The system shall allow users to book a session for a specific skill offer and time slot. |
| FR8 | The system shall notify both the tutor and learner via in-app notifications upon successful session booking. |
| FR9 | The system shall allow users to rate (1–5 stars) and write a review for another user after a completed session. |
| FR10 | The system shall calculate and display the average rating for each user profile based on received reviews. |
| FR11 | The system shall allow users to delete their own skill offers or cancel booked sessions. |

FR12 The system shall allow admins to delete any skill offer or user account for policy violations.

FR13 The system shall store all user data (profiles, offers, sessions, reviews) persistently in a database.

FR14 The system shall display the number of available sessions a tutor has left to offer, based on their set availability.

FR15 The system shall provide a mechanism for users to report inappropriate content or behavior, with details stored for admin review.

FR16 The system shall allow users to recover their account via email-based password reset.

## 4. Non-Functional Requirements

- **Usability**:
  - The user interface shall be intuitive, enabling users to post a skill offer within three taps from the home screen.
  - Navigation shall be consistent, with clear labels and icons across all screens.
  - The app shall provide tooltips or help text for key actions (e.g., posting an offer).
- **Performance**:
  - Screens shall load within 2 seconds on a standard Internet Connection.
  - The system shall support up to 1,000 concurrent users without significant performance degradation.
- **Security**:
  - User passwords shall be hashed and salted (e.g., using bcrypt) before storage in the database.
  - All data transmission between client and server shall use HTTPS/SSL encryption.
  - User sessions shall expire after 30 minutes of inactivity for security.
- **Reliability**:
  - The system shall maintain 99.9% uptime, excluding scheduled maintenance.
  - User data shall be backed up daily to prevent loss due to system failures.
- **Accessibility**:
  - The app shall support screen readers for visually impaired users (WCAG 2.1 compliance).
  - Text shall have a contrast ratio of at least 4.5:1 for readability.
- **Scalability**:
  - The system shall handle up to 10,000 registered users and 1,000 active sessions daily.
- **Compatibility**:
  - The app shall function on iOS and Android devices.

- The app shall support both portrait and landscape orientations.

## 5. Database Schema

The application will use a NoSQL database (MongoDB) to store data persistently. The following collections are defined:

| Collection Name | Fields | Description |
|---|---|---|
| **Users** | `_id`, `email`, `passwordHash`, `name`, `bio`, `profilePic`, `skills[]`, `avgRating`, `location` | Stores user account and profile information, including location for search filtering. |
| **Offers** | `_id`, `title`, `description`, `category`, `createdBy` (links to Users._id), `createdAt`, `duration`, `availableSlots[]` | Stores details of skill offers, including available time slots. |
| **Sessions** | `_id`, `offerId` (links to Offers._id), `tutorId`, `learnerId`, `scheduledTime`, `status` (pending, confirmed, completed, cancelled) | Manages booking and status of skill exchange sessions. |
| **Reviews** | `_id`, `fromUser`, `toUser` (links to Users._id), `rating` (1–5), `comment`, `createdAt` | Stores user ratings and reviews for completed sessions. |
| **Reports** | `_id`, `reportedBy`, `targetId` (user or offer ID), `reason`, `createdAt`, `status` (pending, resolved) | Stores reports of inappropriate content or behavior for admin review. |

## 6. System Interfaces

### 6.1 User Interface

- The app shall provide a mobile-friendly interface built with React Native.
- Key screens include:
    - **Login/Signup**: Form for user authentication.
    - **Home Feed**: Scrollable list of skill offers.
    - **Create Post**: Form to post new skill offers.
    - **Profile**: Display of user details and skills.

### 6.2 Hardware Interfaces

- The app shall run on smartphones with at least 2GB RAM and a modern processor.
- No specific hardware sensors (e.g., GPS, camera) are required for the MVP.

### 6.3 Software Interfaces

- **Frontend**: React Native for cross-platform iOS and Android development.
- **Backend**: Node.js with Express for API endpoints (assumed for MVP).
- **Database**: MongoDB for NoSQL data storage.
- **External APIs**: None required for the MVP.

### 6.4 Communication Interfaces

- The app shall use HTTPS for secure data transmission.
- In-app notifications shall be delivered via a push notification service (e.g., Firebase Cloud Messaging).