**Software Requirements Specification (SRS)**

**SkillSwap Mobile Application**

**1. Introduction**

**1.1 Purpose**

The SkillSwap application is designed to create a peer-to-peer skill exchange platform where students and individuals can share their expertise and learn from others without monetary transactions. The system will allow users to act both as tutors and learners, enabling collaborative skill development.

**1.2 Scope**

The system will include:

* User registration and profile management.
* Posting, searching, and filtering skill offers.
* Session booking and scheduling.
* Ratings and reviews.
* Admin management for content moderation.

Out of scope for MVP:

* In-app payments.
* Built-in video/voice calling.

**1.3 Definitions, Acronyms, and Abbreviations**

* **Tutor** – A user offering a skill.
* **Learner** – A user seeking a skill.
* **SkillSwap** – Exchanging knowledge/skills without money.
* **CRUD** – Create, Read, Update, Delete.
* **MVP** – Minimum Viable Product.
* **UML** – Unified Modeling Language.

**2. Overall Description**

**2.1 User Roles**

* **Student (Tutor/Learner)**: Create/edit profile, post offers, book sessions, leave reviews.
* **Admin**: Manage content, moderate users, and ensure platform safety.

**2.2 User Stories**

* As a learner, I want to filter tutors by skill and rating to find reliable help.
* As a tutor, I want to set my available slots so learners can book accordingly.
* As a student, I want to track my past and upcoming sessions.
* As an admin, I want to remove inappropriate posts.

**3. Functional Requirements**

* **FR1**: User registration/login/logout.
* **FR2**: Profile creation/editing (name, bio, picture, skills).
* **FR3**: Skill offer creation (title, description, category).
* **FR4**: Search and filter offers.
* **FR5**: Session booking and scheduling.
* **FR6**: Notifications on successful booking.
* **FR7**: Ratings and reviews.
* **FR8**: Report inappropriate content.
* **FR9**: Admin moderation (delete offers/users).
* **FR10**: Persistent database storage.

**4. Non-Functional Requirements**

* **Usability**: Simple interface, 3 taps to create an offer.
* **Performance**: Load screens <2s.
* **Security**: Hashed/salted passwords, HTTPS.
* **Reliability**: 99.9% uptime, data backups.

**5. Database Schema**

* **Users**: \_id, email, passwordHash, name, bio, profilePic, skills[], avgRating
* **Offers**: \_id, title, description, category, createdBy, createdAt
* **Sessions**: \_id, offerId, tutorId, learnerId, scheduledTime, status
* **Reviews**: \_id, fromUser, toUser, rating, comment, createdAt

**6. UML Diagrams (placeholders)**

* **Use Case Diagram**: Shows Student/Admin interactions (Login, Post, Book, Review, Moderate).
* **Class Diagram**: Shows User, Offer, Session, Review with attributes and relations.

**7. MVP Frontend (React Native)**

* **Login/Signup Screen** – email + password.
* **Home Feed** – list of skill offers.
* **Create Post Screen** – form to add skill offer.
* **Profile Screen** – user info + skills + bio.

Dummy data for prototype:

const dummyOffers = [

{ id: 1, title: 'Python Tutoring', user: 'Ali' },

{ id: 2, title: 'Guitar Lessons', user: 'Fatima' },

{ id: 3, title: 'Drawing Basics', user: 'Ahmed' },

{ id: 4, title: 'Yoga & Meditation', user: 'Sara' },

];