**SkillSwap Mobile Application: Software Requirements Specification (SRS)**

**1. Introduction**

**1.1. Purpose**

This document defines the requirements for the SkillSwap mobile application. The primary purpose of this application is to create a peer-to-peer skill exchange network where users can offer their skills and learn new ones from others without the need for monetary transactions. The application will serve as a platform to facilitate the discovery, exchange, and management of skill-based sessions between users.

**1.2. Scope**

The following functionalities are in scope for the initial version of the SkillSwap application:

* **User Profiles:** Creation and management of user profiles with personal information and skill listings.
* **Skill Listings:** Functionality for users to post and search for available skills.
* **Booking System:** A system to allow users to book and schedule skill exchange sessions.
* **Ratings and Reviews:** A mechanism for users to rate and review each other after a session.

The following functionalities are out of scope for the initial version:

* In-app payments or any form of monetary transactions.
* In-app communication features like video calling or instant messaging.

**1.3. Definitions and Acronyms**

* **Tutor:** A user offering a skill.
* **Learner:** A user seeking a skill.
* **SkillSwap:** The act of trading skills without money.
* **CRUD:** Create, Read, Update, Delete—the four basic functions of a persistent storage system.
* **MVP:** Minimum Viable Product.
* **UML:** Unified Modeling Language.

**2. Overall Description**

**2.1. User Roles**

The application will support two main user roles:

| Role | Key Permissions & Goals |
| --- | --- |
| **Student (Tutor/Learner)** | Can switch between roles (tutor or learner). Can create and edit a personal profile. Can post offers for skills they possess and requests for skills they want to learn. Can book sessions with other users and leave ratings and reviews. |
| **Admin** | Can view all user data, including profiles and posts. Can remove inappropriate content and suspend or delete user accounts as necessary to maintain platform integrity. |

Export to Sheets

**2.2. User Stories**

The following user stories illustrate the desired functionality from the perspective of the end-user:

* As a learner, I want to filter tutors by skill and rating, so that I can quickly find a reliable helper.
* As a tutor, I want to set my available time slots, so that learners can only book me when I am free.
* As a student, I want to see a history of my past and upcoming sessions, so that I can keep track of my commitments.
* As an admin, I want to flag and remove inappropriate skill offers, so that I can maintain a safe and professional community.

**3. Functional Requirements**

The system shall meet the following functional requirements:

* **FR1:** The system shall allow a user to register an account using a valid email address and a password.
* **FR2:** The system shall allow a registered user to log in and log out securely.
* **FR3:** The system shall allow a user to create and edit their profile, including fields for name, bio, and profile picture.
* **FR4:** The system shall allow a user to create a new "Skill Offer" with fields for title, description, category, and duration.
* **FR5:** The system shall allow a user to search for skill offers based on keywords, categories, and user location.
* **FR6:** The system shall display a list of skill offers, including the offer details and the profile of the tutor.
* **FR7:** The system shall allow a user to book a session for a specific skill offer and time slot.
* **FR8:** The system shall notify both the tutor and the learner upon a successful session booking.
* **FR9:** The system shall allow a user to rate 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.
* **FR11:** The system shall allow a user to delete their own skill offers or cancel a booked session.
* **FR12:** The system shall allow an admin to delete any skill offer or user account.
* **FR13:** All user data, including profiles, offers, and ratings, shall be stored persistently in a database.
* **FR14:** The system shall display the number of available sessions a tutor has left to offer.
* **FR15:** The system shall provide a mechanism for users to report inappropriate content or behavior.

**4. Non-Functional Requirements**

* **Usability:** The user interface shall be intuitive enough for a non-technical user to post a skill offer within three taps of opening the application. The navigation between screens should be clear and consistent.
* **Performance:** No screen shall take longer than two seconds to load on a standard Wi-Fi connection. The system should handle concurrent sessions without significant degradation in performance.
* **Security:** User passwords shall be securely encrypted (hashed and salted) before storage in the database. All data transmission between the client and server shall be secured using HTTPS/SSL.
* **Reliability:** The system shall have a minimum uptime of 99.9% and be resilient to common errors. User data shall be backed up regularly to prevent loss.

**5. Database Schema**

The application will use a NoSQL database, such as MongoDB, to store the following collections:

| Collection Name | Fields | Description |
| --- | --- | --- |
| **Users** | \_id, email, passwordHash, name, bio, profilePic, skills[], avgRating | Stores all user information, including login credentials and profile details. |
| **Offers** | \_id, title, description, category, createdBy (links to Users.\_id), createdAt | Stores details of each skill offer posted by users. |
| **Sessions** | \_id, offerId, tutorId, learnerId, scheduledTime, status | Manages the booking and status of skill exchange sessions. |
| **Reviews** | \_id, fromUser, toUser, rating, comment, createdAt | Stores user ratings and reviews. |

Export to Sheets

**6. UML Diagrams**

**6.1. Use Case Diagram**

(Note: This section is a placeholder. The diagram would be inserted here, showing the actors (Student, Admin) and the use cases (e.g., Login, Post Offer, Book Session) and their relationships.)

**6.2. Class Diagram**

(Note: This section is a placeholder. The diagram would be inserted here, showing the classes (e.g., User, Offer, Session) with their attributes and methods, and the relationships between them.)

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

(Note: This section is a placeholder. The code for the React Native prototype would be inserted here, organized by screen.)

**7.1. Login/Signup Screen**

* **Description:** Two simple forms allowing users to either log in with existing credentials or create a new account.
* **Dummy Data:**
  + email: 'test@student.com'
  + password: '12345'

**7.2. Home Feed Screen**

* **Description:** A scrollable list of skill offer cards.
* **Dummy Data:**

JavaScript

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' },

];

**7.3. Create Post Screen**

* **Description:** A form with text inputs and a button to submit a new skill offer.
* **Functionality:** When the "Post" button is pressed, the data should be logged to the console, and the user should be navigated back to the Home Feed.

**7.4. Profile Screen**

* **Description:** Displays the user's personal information, skills, and bio.
* **Dummy Data:**

JavaScript

const user = {

name: 'Your Name',

skills: ['React Native', 'Guitar', 'Photography'],

bio: 'A passionate developer and musician looking to share my skills with the world.',

};