NAME; MAAZ BIN JAMEEL KHATTAK

RNO; 8574

SkillSwap Mobile Application: Software Requirements Specification (SRS)

# *ASSIGNMENT1;*

## *SRS FOR SKILLSWAP APP;*

# 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.

## 1.2 Scope

The application enables users to create profiles, list skills they can teach or want to learn, book sessions, and review others. The system facilitates skill exchange in a non-monetary model.

## 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.  
- MVP: Minimum Viable Product.  
- UML: Unified Modeling Language.

# 2. Overall Description

## 2.1 User Roles

Two main user roles are supported:  
  
1. Student (Tutor/Learner): Can create and edit profiles, post skill offers, book sessions, and leave reviews.  
2. Admin: Can view all user data, delete inappropriate content, and suspend users.

## 2.2 User Stories

- As a learner, I want to filter tutors by skill and rating.  
- As a tutor, I want to set my available time slots.  
- As a student, I want to track my past and upcoming sessions.  
- As an admin, I want to remove inappropriate content.

# 3. Functional Requirements

The system shall provide functionalities including user registration, login, profile management, skill posting, search and booking, notifications, reviews, and admin moderation.

# 4. Non-Functional Requirements

- Usability: Simple, intuitive UI.  
- Performance: Each screen loads within 2 seconds.  
- Security: Passwords hashed and SSL enforced.  
- Reliability: 99.9% uptime with data backups.

# 5. Database Schema

Collections include:  
  
- Users: id, email, passwordHash, name, bio, skills, avgRating.  
- Offers: id, title, description, category, createdBy.  
- Sessions: id, offerId, tutorId, learnerId, scheduledTime, status.  
- Reviews: id, fromUser, toUser, rating, comment.

# 6. UML Diagrams

## 6.1 Use Case Diagram

Shows interactions between Student and Admin with system use cases. (Insert diagram here).

## 6.2 Class Diagram

The following diagram illustrates main classes and their relationships. (Insert diagram PNG here).

# 7. MVP Frontend Development

## 7.1 Login/Signup Screen

Simple forms for user authentication.

## 7.2 Home Feed Screen

Displays a scrollable list of skill offers.

## 7.3 Create Post Screen

Form for creating a new skill offer.

## 7.4 Profile Screen

Displays user information and listed skills.

## 7.5 Code Implementation

The MVP is implemented in React Native with Expo. The main navigation and screen components are coded in App.js as shown below:

import React from "react";

import { View, Text, TextInput, Button, FlatList, TouchableOpacity } from "react-native";

import { NavigationContainer } from "@react-navigation/native";

import { createNativeStackNavigator } from "@react-navigation/native-stack";

const Stack = createNativeStackNavigator();

*// Dummy Data*

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" },

];

const user = {

  name: "Your Name",

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

  bio: "A passionate developer and musician looking to share my skills.",

};

*// Screens*

function LoginScreen({ navigation }) {

  return (

    <View style={{ flex: 1, justifyContent: "center", padding: 20 }}>

      <Text style={{ fontSize: 24, marginBottom: 20 }}>Login</Text>

      <TextInput placeholder="Email" style={{ borderWidth: 1, marginBottom: 10, padding: 8 }} />

      <TextInput placeholder="Password" secureTextEntry style={{ borderWidth: 1, marginBottom: 10, padding: 8 }} />

      <Button title="Login" onPress={() => navigation.replace("Home")} />

      <Button title="Go to Signup" onPress={() => navigation.navigate("Signup")} />

    </View>

  );

}

function SignupScreen({ navigation }) {

  return (

    <View style={{ flex: 1, justifyContent: "center", padding: 20 }}>

      <Text style={{ fontSize: 24, marginBottom: 20 }}>Signup</Text>

      <TextInput placeholder="Email" style={{ borderWidth: 1, marginBottom: 10, padding: 8 }} />

      <TextInput placeholder="Password" secureTextEntry style={{ borderWidth: 1, marginBottom: 10, padding: 8 }} />

      <Button title="Signup" onPress={() => navigation.replace("Home")} />

    </View>

  );

}

function HomeScreen({ navigation }) {

  return (

    <View style={{ flex: 1, padding: 20 }}>

      <Text style={{ fontSize: 24, marginBottom: 10 }}>Skill Offers</Text>

      <FlatList

        data={dummyOffers}

        keyExtractor={(item) => item.id}

        renderItem={({ item }) => (

          <TouchableOpacity

            style={{ padding: 15, borderWidth: 1, marginBottom: 10, borderRadius: 8 }}

            onPress={() => alert(`Booking ${item.title}`)}

          >

            <Text style={{ fontSize: 18 }}>{item.title}</Text>

            <Text>By: {item.user}</Text>

          </TouchableOpacity>

        )}

      />

      <Button title="Create Post" onPress={() => navigation.navigate("CreatePost")} />

      <Button title="Profile" onPress={() => navigation.navigate("Profile")} />

    </View>

  );

}

function CreatePostScreen({ navigation }) {

  return (

    <View style={{ flex: 1, padding: 20 }}>

      <Text style={{ fontSize: 24, marginBottom: 10 }}>Create Skill Offer</Text>

      <TextInput placeholder="Title" style={{ borderWidth: 1, marginBottom: 10, padding: 8 }} />

      <TextInput placeholder="Description" style={{ borderWidth: 1, marginBottom: 10, padding: 8 }} />

      <Button

        title="Post"

        onPress={() => {

          console.log("Post created!");

          navigation.goBack();

        }}

      />

    </View>

  );

}

function ProfileScreen() {

  return (

    <View style={{ flex: 1, padding: 20 }}>

      <Text style={{ fontSize: 24, marginBottom: 10 }}>{user.name}</Text>

      <Text style={{ marginBottom: 5 }}>Bio: {user.bio}</Text>

      <Text style={{ marginBottom: 5 }}>Skills:</Text>

      {user.skills.map((skill, index) => (

        <Text key={index}>- {skill}</Text>

      ))}

    </View>

  );

}

*// Main App*

export default function App() {

  return (

    <NavigationContainer>

      <Stack.Navigator initialRouteName="Login">

        <Stack.Screen name="Login" component={LoginScreen} />

        <Stack.Screen name="Signup" component={SignupScreen} />

        <Stack.Screen name="Home" component={HomeScreen} />

        <Stack.Screen name="CreatePost" component={CreatePostScreen} />

        <Stack.Screen name="Profile" component={ProfileScreen} />

      </Stack.Navigator>

    </NavigationContainer>

  );

}