Cross Platform App Development No. 3

Aim: Programs regarding concepts of JavaScript, JSX, XAML and C.

Objectives:

The objective of this experiment is to have an understanding of JavaScript, JSX, XAML, and C.

Requirements:

1. Computer with internet access.
2. Windows Operating System
3. Visual Studio IDE
4. Node.js and yarn for React Native
5. Android Studio for Android development
6. Git for version control

Tools:

1. Visual Studio IDE
2. Node.js
3. Android Studio
4. Git

Theory:

1. JavaScript:

- **Role in React Native:** JavaScript is the primary programming language used in React Native. React Native enables developers to build mobile applications using a combination of JavaScript and React, a popular JavaScript library for building user interfaces.
- **Key Features:** JavaScript is an event based and dynamic scripting language. It allows developers to create interactive and responsive user interfaces. In React Native, JavaScript is used to define the logic, components, and behavior of the mobile application.

2. JSX (JavaScript XML):

- **Integration in React Native:** In React Native, JSX is used to define the structure of UI components. It provides a more readable way to describe the UI hierarchy and component relationships.

3. XAML (eXtensible Application Markup Language) and C:

- React Native primarily uses JavaScript, XAML and C are used in Xamarin to create cross-platform mobile applications. Xamarin allows developers to write the business logic in C and define the user interface using XAML.

<u>Class Components:</u>

1. Login Page (Class Component):

- State Management: Class components use the `state` property to manage local component state. In the login page, state variables (`username` and `password`) are defined to store user input.
- Event Handling: The `handleLogin` method is triggered when the login button is pressed. It typically contains the logic for authenticating the user based on the provided credentials.

2. Register Page (Class Component):

- State Management: Similar to the login page, the register page uses the `state` property to manage local state variables (`username`, `email`, and `password`).
- Event Handling: The `handleRegister` method is responsible for executing the registration logic when the user presses the register button.

Functional Components:

1. Login Page (Functional Component):

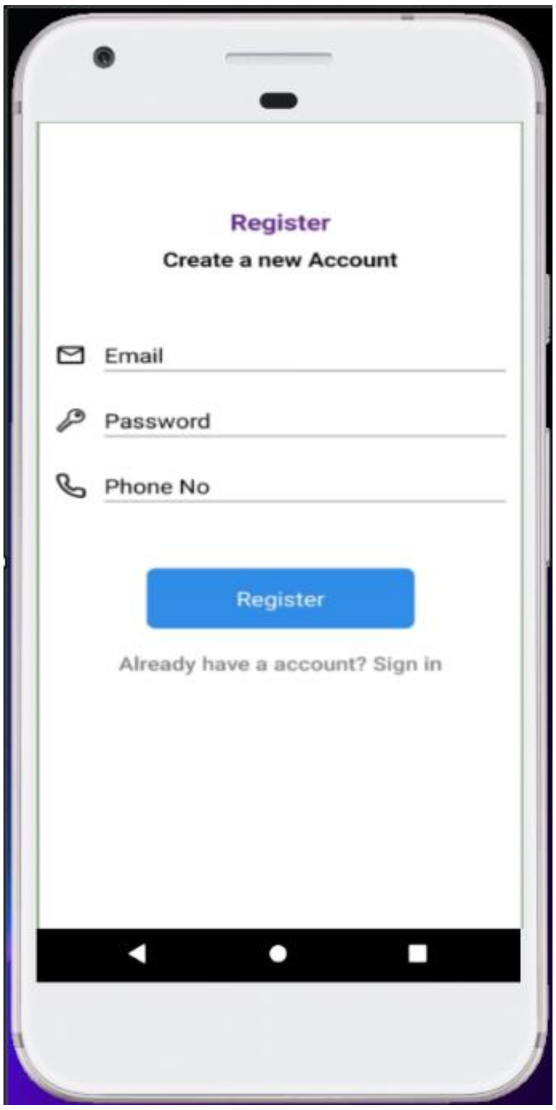
- State with Hooks: Functional components use hooks like `useState` to manage local component state. `useState` is employed to create state variables (`username` and `password`).
- Event Handling: The `handleLogin` function is used for handling the login action. It captures the entered username and password and executes the authentication logic.
- Return JSX: Functional components use the `return` statement to specify the component's structure. JSX is used to define the UI

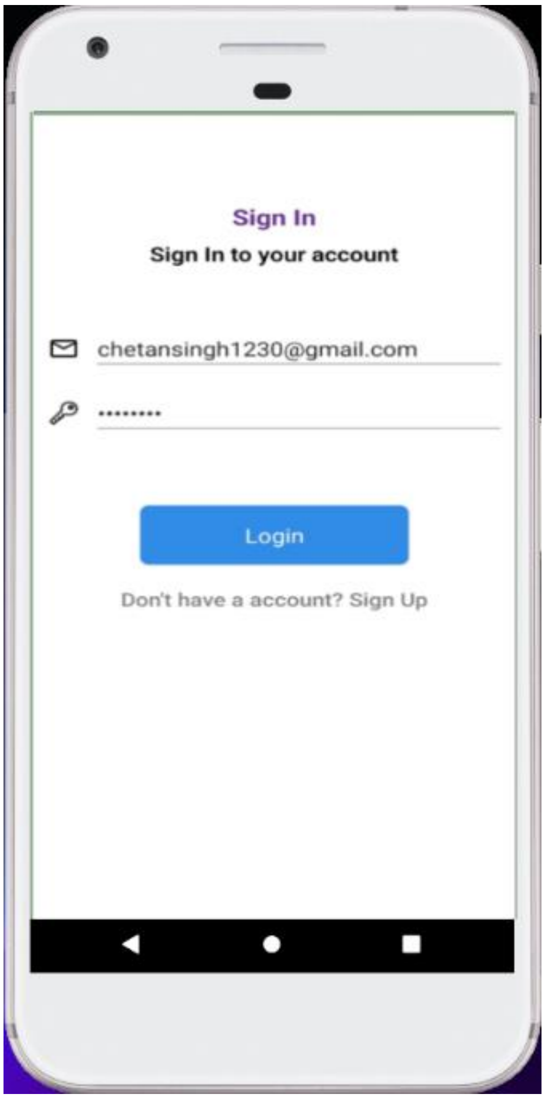
components, including `TextInput` for input fields and `Button` for triggering login.

2. Register Page (Functional Component):

- State with Hooks: Similar to the login page, the register page utilizes `useState` to manage state variables (`username`, `email`, and `password`).
- Event Handling: The `handleRegister` function is responsible for executing the registration logic when the user presses the register button.
- Return JSX: The component's UI is defined using JSX within the `return` statement. It includes `TextInput` components for user input and a `Button` component for registration.

Implementation/Code:





For Login:-

```
import { Ionicons } from "@expo/vector-icons";
import React, { useEffect, useState } from "react";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import { useNavigation } from "@react-navigation/native";
import { signInWithEmailAndPassword } from "firebase/auth";
import { auth } from "../firebase";
const LoginScreen = () => {
  const [email, setEmail] = useState("");
  const [loading,setLoading] = useState(false);
  const [password, setPassword] = useState("");
  const navigation = useNavigation();
  useEffect(() => {
    setLoading(true);
    const unsubscribe = auth.onAuthStateChanged((authUser) => {
      if(!authUser){
        setLoading(false);    }
      if(authUser){
        navigation.replace("Home");}});
    return unsubscribe;
  },[])
    const          login          =          ()          =>          {
signInWithEmailAndPassword(auth,email,password).then((userCredential) => {
      console.log("user credential",userCredential);
      const user = userCredential.user;
      console.log("user details",user)
    })
  }
  return (
    <SafeAreaView
      style={{
        flex: 1,
        backgroundColor: "white",
        alignItems: "center",
        padding: 10,
      }} >
      {loading ? (
        <View
style={{alignItems:"center",justifyContent:"center",flexDirection:"row",flex:1}}>
          <Text style={{marginRight:10}}>Loading</Text>
          <ActivityIndicator size="large" color={"red"}/>
        </View>
      ) : (
          <KeyboardAvoidingView>
          <View
            style={{
              justifyContent: "center",
              alignItems: "center",
              marginTop: 100,
            }}
          >
            <Text style={{ fontSize: 20, color: "#662d91", fontWeight: "bold" }}>
              Sign In
            </Text>

            <Text style={{ fontSize: 18, marginTop: 8, fontWeight: "600" }}>
              Sign In to your account
            </Text>
          </View>

          <View style={{ marginTop: 50 }}>
            <View style={{ flexDirection: "row", alignItems: "center" }}>
              <MaterialCommunityIcons
                name="email-outline"
                size={24}
                color="black"
            <Pressable onPress={() => navigation.navigate("Register")} style={{
      marginTop: 20 }}>
              <Text
                style={{
                  textAlign: "center",
                  fontSize: 17,
                  color: "gray",
                  fontWeight: "500"}} >
                Don't have a account? Sign Up
              </Text>
            </Pressable>
          </View>
        </KeyboardAvoidingView>  )}
      </SafeAreaView> );};
    export default LoginScreen;
const styles = StyleSheet.create({});
```

For Registration:-

```
import { Feather } from '@expo/vector-icons';
import { Ionicons } from "@expo/vector-icons";
import React, { useState } from "react";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import { useNavigation } from "@react-navigation/native";
import { createUserWithEmailAndPassword } from "firebase/auth";
import { doc, setDoc } from "firebase/firestore";
const RegisterScreen = () => {
  const [email,setEmail] = useState("");
  const [password,setPassword] = useState("");
  const [phone,setPhone] = useState("");
  const navigation = useNavigation();
  const register = () => {
   if(email === "" || password === "" || phone === ""){
    Alert.alert(
     "Invalid Details",
     "Please fill all the details",
       text: "Cancel",
       onPress: () => console.log("Cancel Pressed"),
       style: "cancel"
      { text: "OK", onPress: () => console.log("OK Pressed") }
     { cancelable: false }
   createUserWithEmailAndPassword(auth,email,password).then((userCredential)
    console.log("user credential",userCredential);
    const user = userCredential._tokenResponse.email;
    const myUserUid = auth.currentUser.uid;

    setDoc(doc(db,"users",`${myUserUid}`),{
     email:user,
 <View style={{ flexDirection: "row", alignItems: "center" }}>
     <Feather name="phone" size={24} color="black" />
      <TextInput
       value={phone}
       onChangeText={(text) => setPhone(text)}
       placeholder="Phone No"
       placeholderTextColor="black"
       style={{
        fontSize: password ? 18 : 18,
        borderBottomWidth: 1,
        borderBottomColor: "gray",
        marginLeft: 13,
        width: 300,
                marginVertical: 10,
             </View>
             <Pressable
             onPress={register}
               style={{
                 width: 200,
                 backgroundColor: "#318CE7",
                 padding: 15,
                 borderRadius: 7,
                 marginTop: 50,
                 marginLeft: "auto",
                 marginRight: "auto",
               } >
               <Text style={{ fontSize: 18, textAlign: "center", color: "white" }}>
                 Register
               </Text>
             </Pressable>
             <Pressable onPress={() => navigation.goBack()} style={{ marginTop: 20 }}>
               <Text
                 style={{
                   textAlign: "center",
                   fontSize: 17,
                   color: "gray",
                   fontWeight: "500",  }}>

                          Already have a account? Sign in
               </Text>
             </Pressable>
           </View>
         </KeyboardAvoidingView>
       </SafeAreaView>  );};
     export default RegisterScreen
const styles = StyleSheet.create({});
```

Conclusion:

From this Experiment we learnt different concepts of JSX, JS where we learnt how one should use functional components if we are writing a presentational component which doesn't have its own state or needs to access a lifecycle hook.

References:

- https://nodejs.org/en
- https://developer.android.com/studio