Mobile Applications

Lab 5

Context API + AsyncStorage + SecureStore + Camera (Expo, TypeScript)

Mateusz Pawełkiewicz

Exercise Goal Creating an application with global state (Context API) that can:

- Remember data (e.g., username, list of photos) after app restart (**AsyncStorage**).
- Securely store login tokens (**SecureStore**).
- Handle the camera, take photos, and save them to the phone's gallery (**Expo Camera + Media Library**).

Required Software Project from previous laboratories (TypeScript).

Library Installation Execute the following commands in the terminal in the project folder:

```
# 1. Storage for regular data (AsyncStorage)
npm install @react-native-async-storage/async-storage

# 2. Secure storage for passwords/tokens (SecureStore)
npx expo install expo-secure-store

# 3. Camera and photo gallery handling
npx expo install expo-camera expo-media-library
```

Step 1: Global State Structure (Context API) We will create a context that manages three things: username, list of taken photos, and login token.

Create file: src/context/AppContext.tsx

```
import React, { createContext, useContext, useEffect, useState } from "react";
import * as SecureStore from "expo-secure-store";
import { loadAppData, saveAppData } from "../utils/storage";

export type Photo = {
  id: string;
  uri: string;
  createdAt: number;
};

type AppState = {
  displayName: string;
  photos: Photo[];
  token: string | null;
};

type AppContextType = {
  state: AppState;
  setDisplayName: (name: string) => void;
  addPhoto: (photo: Photo) => void;
  removePhoto: (id: string) => void;
  login: (token: string) => Promise<void>;
  logout: () => Promise<void>;
};

const AppContext = createContext<AppContextType | null>(null);

export const AppProvider = ({ children }: { children: React.ReactNode }) => {
  const [state, setState] = useState<AppState>({
```

```
  displayName: "",
  photos: [],
  token: null,
 });

 useEffect(() => {
  const initApp = async () => {
   try {

    const savedData = await loadAppData();

    const savedToken = await SecureStore.getItemAsync("auth_token");

    setState((prev) => ({
     ...prev,
     displayName: savedData?.displayName ?? "",
     photos: savedData?.photos ?? [],
     token: savedToken ?? null,
    }));
   } catch (error) {
    console.error("Error loading data:", error);
   }
  };
  initApp();
 }, []);


 useEffect(() => {
  // We only save regular data to AsyncStorage
  saveAppData({
   displayName: state.displayName,
   photos: state.photos,
  });
 }, [state.displayName, state.photos]);

 const setDisplayName = (name: string) =>
  setState((s) => ({ ...s, displayName: name }));

 const addPhoto = (photo: Photo) =>
  setState((s) => ({ ...s, photos: [photo, ...s.photos] }));

 const removePhoto = (id: string) =>
  setState((s) => ({ ...s, photos: s.photos.filter((p) => p.id !== id) }));

 const login = async (token: string) => {
  await SecureStore.setItemAsync("auth_token", token);
  setState((s) => ({ ...s, token }));
 };


 const logout = async () => {
  await SecureStore.deleteItemAsync("auth_token");
  setState((s) => ({ ...s, token: null }));
 };

 return (
  <AppContext.Provider
```

```
    value={{ state, setDisplayName, addPhoto, removePhoto, login, logout }}
   >
    {children}
   </AppContext.Provider>
  );
};

export const useAppContext = () => {
  const context = useContext(AppContext);
  if (!context) {
   throw new Error("useAppContext must be used within AppProvider");
  }
  return context;
};
```

Remember: Connect AppProvider in App.tsx

Step 2: AsyncStorage Handling (Data Saving) Create file: src/utils/storage.ts

```
import AsyncStorage from "@react-native-async-storage/async-storage";
import { Photo } from "../context/AppContext";

// Key under which we save data on the phone
const STORAGE_KEY = "MY_APP_DATA_V1";

type SavedData = {
  displayName: string;
  photos: Photo[];
};

export const saveAppData = async (data: SavedData) => {
  try {
   const jsonValue = JSON.stringify(data);
   await AsyncStorage.setItem(STORAGE_KEY, jsonValue);
  } catch (e) {
   console.error("Failed to save data", e);
  }
};

export const loadAppData = async (): Promise<SavedData | null> => {
  try {
   const jsonValue = await AsyncStorage.getItem(STORAGE_KEY);
   return jsonValue != null ? JSON.parse(jsonValue) : null;
  } catch (e) {
   console.error("Failed to load data", e);
   return null;
  }
};
```

Step 3: Login with SecureStore Authorization tokens are sensitive data. Never use AsyncStorage for them (because data there is saved in plain text). We use Expo SecureStore, which uses the operating system's keychain/keystore.

Add the login method call from the context to the login screen.

Step 4: Camera (CameraView) In the latest Expo, we use the CameraView component (it replaced the old Camera).

Create file: src/screens/CameraScreen.tsx

```
import React, { useRef, useState } from "react";
import { View, Text, TouchableOpacity, Image, StyleSheet, Alert } from "react-native";
import { CameraView, useCameraPermissions } from "expo-camera";
import { useAppContext } from "../context/AppContext";

export default function CameraScreen() {
 const [permission, requestPermission] = useCameraPermissions();
 const cameraRef = useRef<CameraView>(null);
 const { addPhoto } = useAppContext();
 const [photoUri, setPhotoUri] = useState<string | null>(null);


 if (!permission) return <View />;
 if (!permission.granted) {
  return (
   <View style={styles.container}>
    <Text style={{ textAlign: "center", marginBottom: 10 }}>
     Camera Access
    </Text>
    <TouchableOpacity onPress={requestPermission} style={styles.btn}>
     <Text style={styles.btnText}>Grant Permissions</Text>
    </TouchableOpacity>
   </View>
  );
 }


 const takePicture = async () => {
  if (cameraRef.current) {
   try {
    const photo = await cameraRef.current.takePictureAsync({
     quality: 0.7,
     skipProcessing: true,
    });

    if (photo?.uri) {
     setPhotoUri(photo.uri); // Show preview
    }
   } catch (error) {
    Alert.alert("Error", "Failed to take picture");
   }
  }
 };


 const savePhoto = () => {
  if (photoUri) {
   addPhoto({
    id: Date.now().toString(),
    uri: photoUri,
    createdAt: Date.now(),
```

```
    });
    setPhotoUri(null);
    Alert.alert("Success", "Photo saved in the app!");
   }
  };


  if (photoUri) {
   return (
    <View style={styles.container}>
     <Image source={{ uri: photoUri }} style={styles.preview} />
     <View style={styles.row}>
      <TouchableOpacity onPress={() => setPhotoUri(null)} style={[styles.btn, styles.outline]}>
       <Text style={{ color: "black" }}>Discard</Text>
      </TouchableOpacity>
      <TouchableOpacity onPress={savePhoto} style={styles.btn}>
       <Text style={styles.btnText}>Save</Text>
      </TouchableOpacity>
     </View>
    </View>
   );
  }


  return (
   <View style={styles.container}>
    <CameraView
     style={styles.camera}
     facing="back"
     ref={cameraRef}
    />
    <TouchableOpacity onPress={takePicture} style={styles.captureBtn}>
     <Text style={styles.btnText}>Photo</Text>
    </TouchableOpacity>
   </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, backgroundColor: "#fff", justifyContent: "center" },
  camera: { flex: 1 },
  preview: { flex: 1, resizeMode: "contain" },
  row: { flexDirection: "row", justifyContent: "space-around", padding: 20 },
  btn: { backgroundColor: "#007AFF", padding: 15, borderRadius: 10, minWidth: 100, alignItems: "center" },
  outline: { backgroundColor: "transparent", borderWidth: 1, borderColor: "#007AFF" },
  btnText: { color: "white", fontWeight: "bold" },
  captureBtn: {
   position: "absolute", bottom: 40, alignSelf: "center",
   backgroundColor: "white", width: 70, height: 70, borderRadius: 35,
   justifyContent: "center", alignItems: "center", borderWidth: 5, borderColor: "#ddd"
  }
});
```

Step 5: Photo List and Saving to Gallery (MediaLibrary) On the photo list, we will add a button allowing the photo to be saved to the phone's system gallery.

Fragment src/screens/GalleryScreen.tsx:

```
import React from "react";
import { FlatList, Image, Text, View, Button, Alert, StyleSheet } from "react-native";
import * as MediaLibrary from "expo-media-library";
import { useAppContext } from "../context/AppContext";

export default function GalleryScreen() {
  const { state, removePhoto } = useAppContext();


  const saveToSystemGallery = async (uri: string) => {


    try {
      await MediaLibrary.saveToLibraryAsync(uri);
      Alert.alert("Saved", "Photo saved to phone gallery!");
    } catch (e) {
      Alert.alert("Error", "Failed to save photo.");
    }
  };

  return (
    <View style={styles.container}>
      <FlatList
        data={state.photos}
        keyExtractor={(item) => item.id}
        renderItem={({ item }) => (
          <View style={styles.card}>
            <Image source={{ uri: item.uri }} style={styles.thumb} />
            <View style={styles.info}>
              <Text>{new Date(item.createdAt).toLocaleString()}</Text>
              <View style={styles.actions}>
                <Button title="Save to phone" onPress={() => saveToSystemGallery(item.uri)} />
                <Button title="Delete" color="red" onPress={() => removePhoto(item.id)} />
              </View>
            </View>
          </View>
        )}
        ListEmptyComponent={<Text style={styles.empty}>No photos in the app</Text>}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 10 },
  card: { flexDirection: "row", marginBottom: 15, backgroundColor: "#f9f9f9", padding: 10, borderRadius: 8 },
  thumb: { width: 80, height: 80, borderRadius: 8, marginRight: 10 },
  info: { flex: 1, justifyContent: "space-between" },
  actions: { flexDirection: "row", gap: 10, marginTop: 5 },
  empty: { textAlign: "center", marginTop: 50, fontSize: 16, color: "gray" }
});
```