Mobile Applications

Lab 1

Introduction to React Native and Expo

Mateusz Pawełkiewicz

Objective

The objective of this exercise is to get familiar with the Expo environment, create and run your first mobile application using React Native, and understand the practical use of the useState hook for managing component state.

Required Software

Before you begin, make sure you have the following tools installed:

1. **Node.js** (LTS version): Essential for running the JavaScript environment.
2. **npm or yarn package manager**: Installed together with Node.js. We will use npx (a tool for executing npm packages)
3. **Expo Go application** on a physical device (Android/iOS) or a configured emulator/simulator on your computer.

Step 1: Creating a New Expo Project

1. Open your terminal (command line) in a directory of your choice.
2. Use the command below to create a new project. You can replace MyFirstProject with your own project name.

   ```
   npx create-expo-app MyFirstProject
   ```

3. After the project is successfully created, navigate into its folder:

   ```
   cd MyFirstProject
   ```

Step 2: Running the Application

1. While in the main project directory, start the Expo development server with the following command:

   ```
   npx expo start
   ```

2. After a moment, a QR code and a menu with options will appear in the terminal.
3. **To run the app:**
   - **On a physical device:** Open the **Expo Go** app and scan the QR code displayed in the terminal.
   - **On an emulator/simulator:** In the terminal, press a (for Android) or i (for iOS).

After completing these steps correctly, the default application should appear on your device's screen.

Step 3: Introduction to useState - A Simple Counter

State in React is a mechanism that allows a component to "remember" information and re-render the UI in response to changes. The useState hook is the fundamental tool for declaring state variables in functional components.

Example: We will build a simple counter that increments its value when a button is pressed.

1. Open the App.js file in your code editor.
2. Replace its entire content with the code below:

```
import React, { useState } from 'react';
import { StyleSheet, Text, View, Button, SafeAreaView } from 'react-native';

export default function App() {
 const [count, setCount] = useState(0);

 const incrementCounter = () => {
  setCount(prevCount => prevCount + 1);
 };

 return (
  <SafeAreaView style={styles.container}>
   <View style={styles.content}>
    <Text style={styles.title}>Simple Counter</Text>
    <Text style={styles.counterText}>Value: {count}</Text>
    <Button
     title="Increase by 1"
     onPress={incrementCounter}
     color="#841584"
    />
   </View>
  </SafeAreaView>
 );
}

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: '#f0f0f0',
 },
 content: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  padding: 20,
 },
 title: {
  fontSize: 28,
  fontWeight: 'bold',
  marginBottom: 20,
 },
 counterText: {
  fontSize: 22,
  marginBottom: 30,
```

```
    color: '#333',
  },
});
```

3. Save the changes in the App.js file. The application on your device should automatically refresh to show the new interface.
4. Test the button's functionality – each press should increment the displayed counter value.

Assignment

Note: The main task to be solved independently, based on the concepts presented above, will be provided and explained by the instructor during class. Please prepare your environment and ensure that the example above works correctly.