DIT 637 Smart and Secure Systems
TT02: Experiencing Edge Computing – Full-Stack Mobile App
06/17/2024 Developed by Clark Ngo
7/1/2024 Reviewed by Sam Chung
School of Technology & Computing (STC) @ City University of Seattle (CityU)

Key Concepts and Tools for Mobile App Development

- **Frontend:** To create the visual part of a mobile app that users interact with.
- **React Native:** To efficiently build dynamic and responsive user interfaces for mobile apps, such as the interactive components of Instagram.
- **Expo Go:** To simplify the development and testing of React Native apps by providing a managed workflow and easy access to development tools without needing to set up complex environments, much like a toolkit that streamlines various tasks into one app.
- **Movie Search:** To practice and demonstrate the capabilities of React Native in creating real-world applications, like searching for movies on Netflix.
- **Codespaces:** To provide a cloud-based development environment that's easy to set up and use from anywhere, similar to Google Docs, but for coding.
- **GitHub as Repo:** To store and manage code with version control, making collaboration easier, just like using Google Drive for sharing documents.

1) User Case

As a movie enthusiast using a **mobile device**, I want to search and browse a list of movies with details such as title, genre, and year so that I can easily find information about movies I am interested in **while on the go**.

Features Included

1. **Movie List Display**:
   - **What**: Shows a list of movies with their titles, genres, and release years.
   - **Why**: To provide users with a comprehensive view of available movies.
2. **Search Functionality**:
   - **What**: An input field that filters the movie list based on the user's search term.
   - **Why**: Allows users to quickly find specific movies by typing part of the title.

Flow of Interaction

3. **Initial View**:
   - The user sees a list of all available movies along with a search bar at the top.
4. **Using the Search Bar**:
   - As the user types into the search bar, the list of movies dynamically updates to show only those that match the search term.

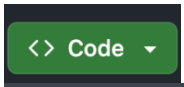
Technical Implementation

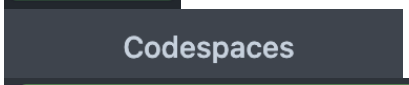
- **State Management**: Uses React's ‘useState’ hook to manage the search term and filtered movie list.
- **Search Handling**: Updates the displayed movie list in real time based on user input.

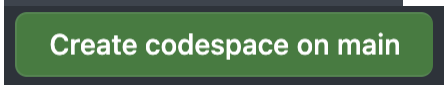
- **Styling**: Utilizes Material-UI components and custom styles for a clean and responsive user interface.

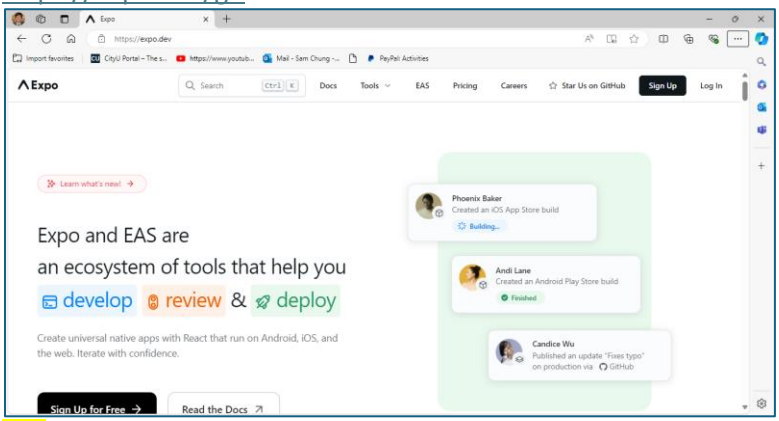
This user story ensures that users have a seamless and interactive experience when searching and discovering movies in the application.

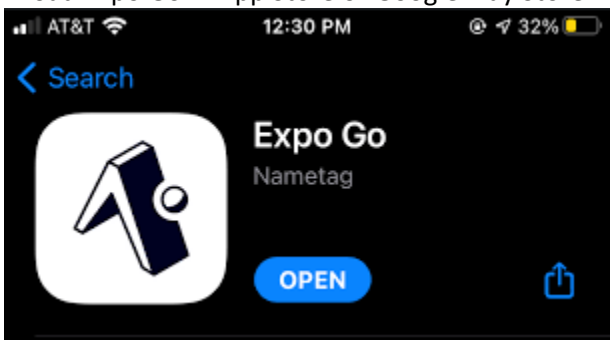
2) Setup

1. Open the project in GitHub.
2. Create a cloud-based development project with GitHub Codespaces:
   a. 

   b. 

   c. 

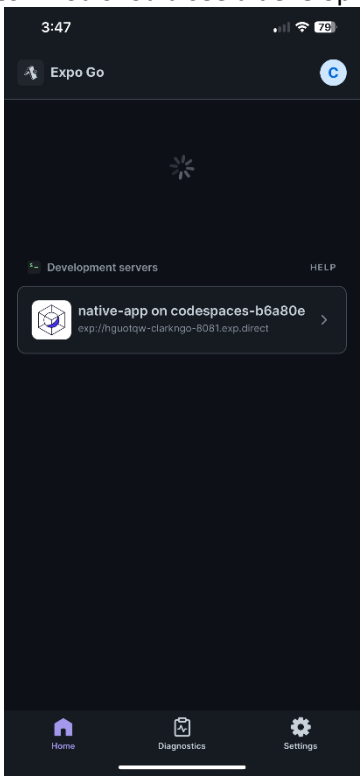
3. In the terminal type:
   a. Do you have your expo account? If not, you need to create an account in https://expo.dev/go
   

   b. npx expo login
   (It requires your Expo email or username and password.)
   c. npm install
   d. npx expo start --tunnel
4. Download Expo Go in App Store or Google Play Store
   

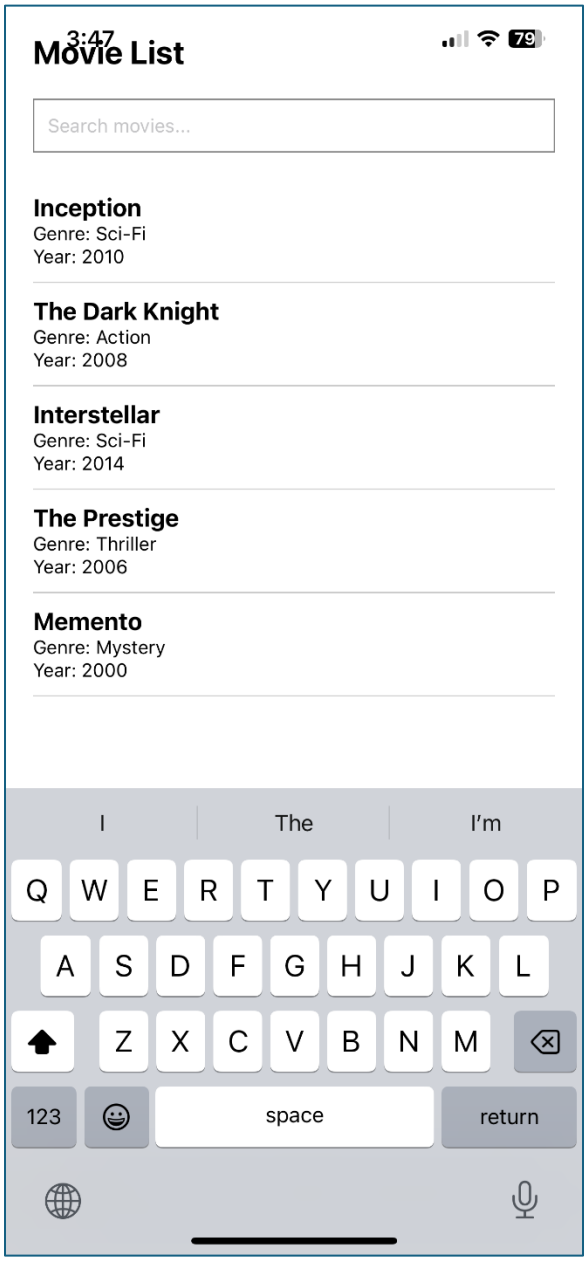
   a.
   b. Open then log in.

The screen does not show no local development server yet.

You can touch and drag down to refresh. You should see a development server like below:



You need to scan the QR code above with Expo Go (Android) or the Camera app (iOS).



3) Screenshot Summary

In the provided screenshot, we see:

1. **Movie List Title**: Displayed using a 'Text' component with a large, bold font style to indicate the screen's main purpose.
2. **Search Bar**: A 'TextInput' component at the top allows users to type in their search queries, dynamically filtering the movie list as they type.
3. **Movie Items**: Each movie is displayed within a View styled as a list item, showing the title in bold, along with the genre and year in smaller text beneath it.
4. **Separation**: Each movie item is visually separated by a bottom border, providing a clean and organized layout, making it easy to distinguish between different movies.

4) Components Overview

- **View (Container)**:
  - **What**: Acts as the main wrapper for the entire screen layout.
  - **Why**: Provides a structured container to hold all other components and ensure proper spacing and padding.
- **Text (Title)**:
  - **What**: Displays the title "Movie List" at the top of the screen.
  - **Why**: Gives users an immediate understanding of the screen's content.
- **TextInput (Search Bar)**:
  - **What**: An input field that allows users to type and search for movies.
  - **Why**: Provides a dynamic way to filter the movie list based on user input.
- **FlatList (Movie List)**:
  - **What**: A list component that renders the filtered list of movies.
  - **Why**: Efficiently displays a scrollable list of movie items, updating in real-time based on the search term.
- **View (ListItem)**:
  - **What**: A container for each individual movie item in the list.
  - **Why**: Groups together the movie title, genre, and year, providing a consistent structure and styling.
- **Text (ItemTitle)**:
  - **What**: Displays the movie title in bold.
  - **Why**: Highlights the main piece of information for each movie, making it easily identifiable.
- **Text (ItemText)**:
  - **What**: Displays the movie genre and year.
  - **Why**: Provides additional details about each movie, helping users to quickly understand the type and release year of the movie.

5) Code Overview

This import statement is bringing in several modules and components necessary for building a React Native application.

- **React**: Core library for building the UI.
- **useState**: Hook to manage local state in functional components.
- **useEffect**: Hook to handle side effects like data fetching.
- **TextInput**: Component for text input from users.
- **FlatList**: Efficient component for rendering lists of data.
- **Text**: Component for displaying text.
- **View**: Container component for layout and styling.
- **StyleSheet**: Utility for creating and managing component styles.

```
1   import React, { useState, useEffect } from 'react';
2   import { TextInput, FlatList, Text, View, StyleSheet } from 'react-native';
```

Sample Data for Movies: Provides a predefined list of movies with their titles, genres, and release years, used to populate the initial movie list and demonstrate search functionality.

```
5   // Sample data for movies
6   const moviesData = [
7     { title: 'Inception', genre: 'Sci-Fi', year: 2010 },
8     { title: 'The Dark Knight', genre: 'Action', year: 2008 },
9     { title: 'Interstellar', genre: 'Sci-Fi', year: 2014 },
10    { title: 'The Prestige', genre: 'Thriller', year: 2006 },
11    { title: 'Memento', genre: 'Mystery', year: 2000 }
12  ];
```

Styled Components: Enhance the appearance and layout of the React Native application by adding custom styles.

```
48  const styles = StyleSheet.create
49    container: {
50      flex: 1,
51      padding: 20,
52      backgroundColor: '#fff',
53    },
54    title: {
55      fontSize: 24,
56      fontWeight: 'bold',
57      marginBottom: 20,
58    },
59    searchInput: {
60      height: 40,
61      borderColor: 'gray',
62      borderWidth: 1,
63      marginBottom: 20,
64      paddingHorizontal: 10,
65    },
66    listItem: {
67      borderBottomWidth: 1,
68      borderBottomColor: '#ccc',
69      paddingVertical: 10,
70    },
71    itemTitle: {
72      fontSize: 18,
73      fontWeight: 'bold',
74    },
75    itemText: {
76      fontSize: 14,
77    },
78  });
```

App Component: The main functional component that uses state to manage the search term and movie list, providing real-time filtering and rendering of movie data to create an interactive user interface.

```
const App = () => {
  const [searchTerm, setSearchTerm] = useState('');
  const [filteredData, setFilteredData] = useState(moviesData);

  useEffect(() => {
    const filteredMovies = moviesData.filter(movie =>
      movie.title.toLowerCase().includes(searchTerm.toLowerCase())
    );
    setFilteredData(filteredMovies);
  }, [searchTerm]);

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Movie List</Text>
      <TextInput
        style={styles.searchInput}
        placeholder="Search movies..."
        value={searchTerm}
        onChangeText={setSearchTerm}
      />
      <FlatList
        data={filteredData}
        keyExtractor={(item, index) => index.toString()}
        renderItem={({ item }) => (
          <View style={styles.listItem}>
            <Text style={styles.itemTitle}>{item.title}</Text>
            <Text style={styles.itemText}>Genre: {item.genre}</Text>
            <Text style={styles.itemText}>Year: {item.year}</Text>
          </View>
        )}
      />
    </View>
  );
};
```

6) Breakdown:

App Component Definition:

```
27    const App = () => {
```

- **What**: Defines the main functional component of the application.
- **Why**: Serves as the entry point for rendering the user interface.

State Declarations:

```
14    const [searchTerm, setSearchTerm] = useState('');
15    const [filteredData, setFilteredData] = useState(moviesData);
```

- **What**: Declares state variables for managing the search term and movie list.
- **Why**: Enables dynamic updates and re-rendering when the search term changes, or the movie list is filtered.

Handle Search Function:

```
17    useEffect(() => {
18      const filteredMovies = moviesData.filter(movie =>
19        movie.title.toLowerCase().includes(searchTerm.toLowerCase())
20      );
21      setFilteredData(filteredMovies);
22    }, [searchTerm]);
```

- **What**: Function to update the search term and filter the movie list based on user input.
- **Why**: Provides real-time search functionality, allowing users to find movies by title.

Render Method:

```
24      return (
25        <View style={styles.container}>
26          <Text style={styles.title}>Movie List</Text>
27          <TextInput
28            style={styles.searchInput}
29            placeholder="Search movies..."
30            value={searchTerm}
31            onChangeText={setSearchTerm}
32          />
33          <FlatList
34            data={filteredData}
35            keyExtractor={(item, index) => index.toString()}
36            renderItem={({ item }) => (
37              <View style={styles.listItem}>
38                <Text style={styles.itemTitle}>{item.title}</Text>
39                <Text style={styles.itemText}>Genre: {item.genre}</Text>
40                <Text style={styles.itemText}>Year: {item.year}</Text>
41              </View>
42            )}
43          />
44        </View>
45      );
46    };
```

- **What**: Renders the search input, movie list, and individual movie details.
- **Why**: Displays the user interface elements and ensures they update dynamically based on the search term and movie list state.

Export Default App:

```
69    export default App;
```

- **What**: Exports the App component as the default export.
- **Why**: Allows the App component to be imported and used in other parts of the application.