**CS624 Full Stack Development – Mobile App**

**HOS07A: React Native Navigation I**

Revised by Sam Chung on May 5, 2023

Reviewed by Clark Ngo on February 15, 2025

Reviewed by Naveena Moddu on May 13, 2025

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Introduction:**

In this hands-on guide, you will learn how to create a cross-platform mobile application using React Native and React Navigation. The application consists of two main components: Cities and AddCity. The app uses the React Navigation library to create a bottom tab navigation interface for the user. Also, it also uses stack-based navigation for Cities and City under Cities directory.

* React Native Navigation: Understanding how to navigate between screens in a React Native app using the React Navigation library.
* Codespaces and Expo: The cloud-based development of React Native mobile app with multiple screens.
* Tab Navigator: The tab navigator is used to navigate between screens by switching between tabs at the bottom. It best suits applications with multiple top-level views that users can switch between.
* Stack Navigator: The stack navigator navigates between screens by pushing new screens on top of the stack. This type of navigator is suitable for creating a hierarchical structure for an application.

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

#### **Readings and Examples:**

* Visit the [CS624 Repository for Examples.](https://github.com/cityuseattle/cs624-examples)
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.
  + This module will use [React Navigation](https://reactnavigation.org/)
* Dabit, N. (2019). [React Native in Action](https://learning.oreilly.com/library/view/react-native-in/9781617294051/). Manning Publications. (ISBN 9781617294051)
  + Chapter 6: *Navigation*

**Activities**

* Section 1: Accessing GitHub Codespaces.
* Section 2: Installing React Navigation Packages.
* Section 3: Modifying the CitiesApp.
* Section 5: Pushing your work to GitHub.

**Section 1: Accessing GitHub Codespaces.**

Instructions on accessing GitHub Codespaces:

1. Go to your repository created from Brightspace from module0X on the GitHub website.
2. In the top-right corner, click on the Code drop-down menu.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Select "Create a Codespace on main."
2. Wait for the Codespaces environment to load. Once loaded, you can access the terminal, file explorer, and other tools to start working on your project with the MS Visual Studio Code Cloud.

**Note:** You need a GitHub account and an active internet connection to access the Codespaces environment.

**Section 2: Installing React Navigation packages.**

Follow the steps outlined in the earlier modules to Set Up React Native Environment with Expo Go and create a project with the name **CitiesApp**.

* Run in Codespaces terminal: **npx create-expo-app CitiesApp**
* Check the dependencies in the “package.json” file.

Before we get started, we need to install the required packages in your React Native project. In the terminal, change the directory to the **CitiesApp** folder and then install the packages below.

* Run in Codespaces terminal: **cd CitiesApp**

React Navigation is made up of some core utilities and those are then used by navigators to create the navigation structure in your app.

* Run in Codespaces terminal: **npm install @react-navigation/native**

Install dependencies into an Expo managed project

* Run in Codespaces terminal: **npx expo install react-native-screens react-native-safe-area-context**

To start working on the Stack and Bottom Tab navigator, we need to install the packages below. Go ahead and install using the two commands.

* Run in Codespaces terminal:

**npm install @react-navigation/bottom-tabs**

**npm install @react-navigation/native-stack**

Finally, install React Native UUID (Universally Unique Identifier).

* Run in Codespaces terminal: **npm install react-native-uuid --save**

Test your mobile app.

**npx expo start --tunnel**

A screenshot of a cell phone

Description automatically generated

Also, check your dependencies. For example, you can find the dependencies in the “package.json” file.

A screenshot of a computer program

Description automatically generated

**Section 3: Modifying the CitiesApp**

Under the project root folder, create a folder named “src.”

* Run in Codespaces terminal:

Create “src” and navigate into directory: **mkdir src && cd src**

Under the “src” folder, create three more folders,

Run in Codespaces terminal:

* Create AddCity directory: **mkdir AddCity**
* Create City directory: **mkdir Cities**
* Create components: **mkdir components**

The project structure should look like this,

Graphical user interface, application

Description automatically generated

**The “theme.js” file**

Under the “src” folder, create a file named **theme.js** and update it with the following code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module07/CitiesApp/src/theme.js). You can find the source code from “Listing 6.1 Creating a theme file with a primary color,” too.

**The “CenterMessage.js” file**

Next, under the “components” folder, create a reusable component “CenterMessage.js” file and update it with the following code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module07/CitiesApp/src/components/CenterMessage.js). You can find the source code from “Listing 6.4 CenterMessage component,” too.

It is used in “Cities.js” and “City.js” to display a message when the array is empty. For example, when the app starts, it won’t have any cities to list; you can display a message instead of just showing a blank screen.

**The “Cities.js” file**

Create **Cities.js** file under the “Cities” folder and update the file with the code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module07/CitiesApp/src/Cities/Cities.js).

You can find the source code from the “Listing 6.8 Cities route” for styling.

The **Cities** component displays a list of saved cities, each with its name and country. The component uses a **ScrollView** to allow scrolling through the list of cities. If no saved cities exist, it displays the message "No saved cities!" using the **CenterMessage** component. The list of cities is passed down to this component as a prop from the parent component. Each city in the list is wrapped in a **TouchableWithoutFeedback** component, which provides a touchable area for navigating to a different page for the selected city when tapped. The styles for the component are defined using the **StyleSheet** module, including styles for the city container, city name, and country name. The component has a static **navigationOptions** property which sets the title and style of the header for this page.

**The “AddCity.js” file**

Now under the “AddCity” folder, create “**AddCity.js”** file and update it with code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module07/CitiesApp/src/AddCity/AddCity.js). You can find the source code from "Listing 6.5 AddCity tab” for functionality and “Listing 6.6 AddCity tab” for styling.

The code in the above file is a React Native component that allows the user to add a city. The component has two TextInputs where the user can enter the city and country names. The component also has a TouchableOpacity button, which when pressed, will submit the form and add the city. The city and country values are stored in the component's state. The submit function checks if the city and country inputs are not empty, then creates an object with the city, country, and a unique id using the **uuidV4** library. The city object is then passed as a prop to the **addCity** function, which updates the app's state.

**The “App.js” file**

Create the App.js file at the root level. Now let us update the App.js file with the navigation logic. Replace the contents of **App.js** with the code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module07/CitiesApp/App.js).

The code defines the main component of a React Native app. It uses the **NavigationContainer** component from the **@react-navigation/native** library to handle the navigation between different app screens. The **createBottomTabNavigator** from the same library creates a bottom tab navigator for the app.

The component has a state property **cities,** an array that will be used to store the cities the user adds. The **addCity** method is used to add a city to the **cities** array. It takes a city object as an argument and pushes it to the **cities** array.

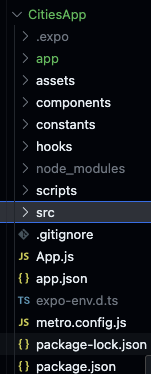
The component returns a bottom tab navigator with two screens, **Cities** and **AddCity**. These screens are imported from the **./src/Cities/Cities** and **./src/AddCity/AddCity** files, respectively. The **initialParams** prop is passed to both screens with the **cities** array and **addCity** function so that they can access the same data.

Update the “package.json” to update the entry point of the app. The default is app/(tabs)/index.tsx.

Find the key: “name” and update as follows:

"main": "node\_modules/expo/AppEntry.js",

Example code for package.json is [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module07/CitiesApp/package.json)



**Note**: Ignore any warnings in the console, as we will finish building the app in the next module.

**Steps to verify the changes:**

* Open the terminal in the GitHub Codespaces environment and navigate to the project folder in our case **CitiesApp**.
* Type “**npx expo start --tunnel”** and press Enter to start the expo development server.
* Wait for the development server to load and show the QR code.
* Open the "Expo Go" app from your mobile device.
* Scan the QR code shown in the terminal with the “Expo Go” or Camera (iOS) app.
* Wait for the app to load on the mobile device.
* Verify that the Cities App is loaded with two tabs.
* Try adding a city using AddCity Tab and observe what will happen.
* Capture the screenshots from the mobile, save them under the module0X directory, and **explain your understanding**.

|  |  |  |
| --- | --- | --- |
| Figure 1.  Initial Screen with two tabs | Figure 2.  The AddCity Tab | Figure 3.  Data Entry 1 |
|  |  |  |
| Figure 4.  The Cities Tab with 1 entry | Figure 5.  Data Entry 2 | Figure 6.  The Cities Tab with 2 entries |
|  |  |  |

**Section 4: Pushing your work to GitHub**

* 1. Go to Source Control on your GitHub codespace and observe the pending changes.

Graphical user interface, text, application

Description automatically generated

* 1. Type the Message for your changes in the Message box on the top. For example,” **Submission for Module0X – Your Name**”
  2. Click on the dropdown beside the commit button and select **Commit & Push** to update the changes to your repository main branch.
  3. Select **Yes** when prompted.

Graphical user interface, application

Description automatically generated