**CS624 Full Stack Development – Mobile App**

**HOS07A: React Navigation II**

Revised by Sam Chung on May 5, 2023

Reviewed by Clark Ngo on February 17, 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 continue to learn to create a cross-platform mobile application using React Native. Along with the Cities and AddCity components that we created in the previous module, we are going to add one more component City to the application and use the React Navigation library to create a Native Stack Navigator in addition to the bottom tab navigation for the user.

**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: Finishing the CitiesApp.
* Section 3: 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 Codespaces on main".
2. Wait for the Codespaces environment to load. Once loaded, you will be able to access the terminal, file explorer, and other tools to start working on your project.

Note: To access the Codespaces environment, you need a GitHub account and an active internet connection.

**Section 2: Finishing the CitiesApp.**

This is the continuation of HOS07. Assume you have the CitiesApp of HOS07 running.

**The “City.js” file**

Under the “Cities” folder, create a file named City.js. Add the code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module08/CitiesApp/Cities/City.js) to City.js.

This City component represents a screen for a particular city. It displays a list of locations within that city and provides a form for adding new locations to the list. Here is a breakdown of the main sections of the code:

1. Import statements: The component imports several modules from the React Native library, as well as some custom components and style definitions.
2. Class declaration: The **City** component is declared as a class, which allows it to have state and lifecycle methods.
3. Static **navigationOptions** method: This method sets the title and styling of the navigation header for this screen. It takes in props, which include the **city** object passed as a parameter to this screen.
4. State initialization: The **state** object is initialized with **name** and **info** properties, which are used to track the values entered in the form fields.
5. **onChangeText** method: This method is called when the user types in one of the form fields and updates the corresponding property in the **state** object.
6. **addLocation** method: This method is called when the user taps the "Add Location" button. It checks if the form fields are empty, and if not, creates a new **location** object with the values entered by the user, and calls a function passed as a parameter to this screen to add the location to the list of locations for this city. It then resets the form fields to their initial values.
7. **render** method: This method returns the JSX markup for the component. It displays a scrollable list of locations for the current city or a message indicating that there are no locations yet. It also displays two form fields for entering new location data and a button to submit the form. The **styles** object is used to define the appearance of the various components.
8. **styles** object: This object defines a set of CSS-like styles for various components of the **City** screen. These styles are referenced in the **render** method using the **style** attribute.

The project structure now should look like this after adding the City.js,

Graphical user interface, application

Description automatically generated

**The “Cities.js” file**

Replace the code in Cities.js with the code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module08/CitiesApp/Cities/Cities.js). The “navigate” function was updated with a city’s location info.

**The “AddCity.js” file**

Replace the code in AddCity.js with the code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module08/CitiesApp/AddCity/AddCity.js). The “submit” function was updated with the correct uuidV4().

**Refactoring the “App.js” file**

Now let us refactor the code in App.js to use Native Stack Navigator along with the Bottom Stack Navigator that we created.

Replace the code in App.js with the code from [here](https://github.com/cityuseattle/cs451-examples/blob/main/Module08/CitiesApp/App.js).

This code in **App.js** uses the **@react-navigation/native** and **@react-navigation/bottom-tabs** libraries to implement a simple multi-screen navigation system. The code imports the required libraries at the top of the file and then defines two components: **CitiesStackScreen** and **App**.

**CitiesStackScreen** is a functional component that creates a stack of two screens: **Cities** and **City**. This stack is defined using the **createNativeStackNavigator** function from the **@react-navigation/native-stack** library. The **Cities** screen displays a list of cities, and the City screen displays the details of a specific city. The **cities**, **addCity**, and **addLocation** properties are passed as **props** to both screens to ensure that they always receive the latest state and functions.

**App** is a class component that defines the main app screen. It has a state object containing a **cities** property, which is initialized as an empty array. The component has two methods: **addCity** and **addLocation**. These methods update the state by adding new cities and locations to the **cities** array.

The **render()** method of the **App** component returns a **NavigationContainer** component from the **@react-navigation/native** library, which provides the navigation context for the entire app. The **NavigationContainer** component contains a **Tab.Navigator** component from the **@react-navigation/bottom-tabs** library, which creates a bottom tab navigation bar. The **Tab.Navigator** has two screens: **Cities** and **AddCity**. The **cities**, **addCity**, and **addLocation** properties are passed as props to both screens to ensure that they always receive the latest state and functions.

**Steps to verify the changes:**

* Open the terminal in the GitHub Codespaces environment and navigate to the project folder.
* 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.**
* **Tap on the City you just added to navigate to City screen.**
* **Try adding multiple locations to the City. Observe the changes.**
* Capture the screenshots from the mobile, save them under the module0X directory, and **explain your understanding**.

**Note: if after adding the location, the location does not show in a City, please hit the back button and press on the same city again.**

|  |  |  |
| --- | --- | --- |
| Figure 1.  Initial Screen with two tabs and a stack (Cities) | Figure 2.  The Cities Stack | Figure 3.  The City Stack with no location info |
| Table  Description automatically generated | A picture containing shape  Description automatically generated | Graphical user interface, application  Description automatically generated |
| Figure 4.  The City Stack with location info | Figure 5.  The Cities Stack |  |
| A picture containing graphical user interface  Description automatically generated | A picture containing shape  Description automatically generated |  |

**Section 3: Pushing your work to GitHub.**

* 1. Go to Source Control on your GitHub Codespaces 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