**CS624 Full Stack Development – Mobile App (React Native)**

**HOS03A: Building React Native App – Part 2**

Created by Veerendra Jagatha on March 9, 2023

Reviewed by Anahita Raeiszadeh on March 20, 2023

Revised by Sam Chung on March 23, 2023

Reviewed by Anahita Raeiszadeh on September 22, 2023

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* Screenshots may be different from your environment.
* The directory path shown in screenshots may be different from yours.
* Version numbers may not match the most current version at the time of writing. If given the option to choose between the stable release (long-term support) or the most recent, please select the **stable release** rather than the beta-testing version.
* 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.
* If you are not familiar with a terminal, command line, and bash scripts, check out this video: <https://youtu.be/Dp7uw9c6QH8>
* All the steps and concepts in this tutorial are from the textbook, so if you encounter problems, please **try to read and compare the textbook to solve the problem**. If you still can't solve the problem, don't hesitate to contact your course TA.
* **Avoid copy-pasting code from the book or the GitHub repository**. Instead, type out the code yourself as much as you can. Resort to copy-pasting only when you are stuck and find things not working as expected.
* 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, ask a TA for help.

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

* Visit [CS 624 Repository for Examples](https://github.com/samchung0117/cs624-examples).
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.
* Dabit, N. (2019). [React Native in Action](https://learning.oreilly.com/library/view/react-native-in/9781617294051/). Manning Publications. (ISBN 9781617294051)
  + Chapter 3: *Building your first React Native app (The different file structure is used.)*

**Learning Outcomes**

* Section 1: Accessing GitHub Codespaces
* Section 2: Creating a mobile app
* Section 3: Copying the application “todos” from PE03 to module04 HOS
* Section 4: Creating the Heading user interface
* Section 5: Creating the TodoList component
* Section 6: Updating the App component with the Todo and TodoList components
* Section 7: Updating the App component for the Done and Delete Buttons
* Section 8: Creating the TodoButton
* Section 9: Updating the App.js for the TodoButton
* Section 10: Updating the TodoList component
* Section 11: Updating the Todo component again
* Section 12: Testing the App component
* Section 13: Pushing your work to GitHub

**Section 1: Accessing GitHub Codespaces**

It would be best to have a GitHub account and an active internet connection to access the Codespaces environment.

If you have questions about this section, please revisit HOS01 Section 1.  
You will repeat the same steps for each HOS.

1. From your course shell, visit the HOS.   
   Visit the HOS GitHub Classroom.
2. Go to your repository created from Brightspace on the GitHub classroom.
3. Download the given HOS guidelines that you will use for your hands-on practices.  
   Also, check the README.md for your HOS.
4. Click the "<>Code" dropdown menu in the top-right corner.
5. Select "Create a codespace on main."

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

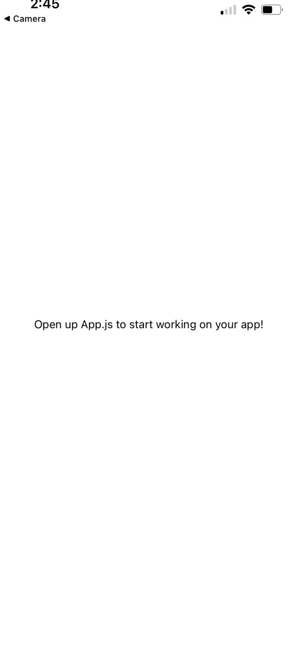
Description automatically generated

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.

Once your codespace is created, the template repository will be automatically cloned into it. Now you can run the application and launch it in a browser called cloud IDE (Integrated Development Environment). We use Visual Studio Code for the web.

**Section 2: Creating a mobile app**

1. Suppose you are under the current HOS directory, HOS02.  
   Run the command “**expo init todos”** to create a new Expo project.  
   Then Select a blank template for your mobile app.
2. You can see a directory called “todos” was created.
3. Change the current directory to the newly created project using the command “**cd todos**.”
4. Run the command “**expo start --tunnel”** to start the development server.
5. Scan the QR code and you can have the following screen below.



**Section 3: Copying the application “todos” from PE03 to module04 HOS**

For this exercise, you must complete part 1 of PE03. You must finish PE03 to continue PE03. If you have any issues with finishing PE03, please contact your TA.

For copying the application “todos” from HOS03 to PE03, visit the [CS 624 Repository for Examples](https://github.com/samchung0117/cs624-examples). You can find the [CopyRepo](https://github.com/samchung0117/cs624-examples/blob/main/Module03/CopyRepo.md) guideline under Module03.

The following table shows what user interfaces will be covered in HOS04,and PE03 Part 2.

|  |  |  |
| --- | --- | --- |
| **HOS or PE** | **User Interface** | **Required** |
| HOS03A | Heading, Input |  |
| PE03 Part 1 | Button | Copy HOS03A |
| **HOS04A** | **Todo List, Todo, Todo Button** | **Copy PE03 Part 1** |
| PE03 Part 2 | Tab bar and Tab bar item | Copy HOS04A |

By the end of PE03 part 1, you will have the application where you are adding todos to the array of todos. The screenshots below show the screen before and after HOS04A and PE03 Part 2, respectively.

**HOS04A**

|  |  |
| --- | --- |
| **Before** | **After** |
| **A screenshot of a computer  Description automatically generated with low confidence** | **A screenshot of a computer  Description automatically generated with low confidence** |

**PE03 Part 2**

|  |  |
| --- | --- |
| **Before** | **After** |
| **A screenshot of a computer  Description automatically generated with low confidence** | **Graphical user interface, application  Description automatically generated** |

**Section 4: Creating the Todo component**

Now we must render them on screen. To begin, you must create the TodoList and Todo components. TodoList will display the list of tasks and use the Todo component for each task. Create a file called Todo.js **in the root folder** (“todos”) to get started. Update Todo.js with the following code.

Timeline

Description automatically generated with low confidence

**Section 5: Creating the TodoList component**

Next, we need to create the TodoList component. Add the TodoList.js file to the **root** folder and update it with the following code.

Text

Description automatically generated with low confidence

**Section 6: Updating the App component with the Todo and Todo List components**

Currently, the TodoList component only accepts an array of todos as a property. The next step is to map over these todos, making a new Todo component (imported at the start of the file) for each todo and providing the todo as a property to the Todo component.

Let us now import the TodoList component into the App.js file and pass in the todos as a property. Update the render method in app.js as shown below and import the TodoList at the top of the app.js file.

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Run the application by following the instructions to use the Simulator at the end of this document to confirm the modifications. As you add a todo, you should see it appear in the list of todos, as shown in the image below. Capture the screenshot from your mobile and save it in the todos folder.

Graphical user interface, application

Description automatically generated

**Section 7: Updating the App component for the Done and Delete Buttons**

The button text should be bold and green after you click Done. The to-do item should be removed from the list of tasks if you click Delete. Add the toggleComplete and deleteTodo functions to our app.js. First, we need to bind the functions to the class in the constructor and add the functions in the app.js file.

Update the app.js as shown below.

Text

Description automatically generated

**Section 8: Creating the TodoButton**

To hook in these functions, we need to create a button component to pass into the todo. Please create a new file in the root folder called TodoButton.js and update it with the following code.

A picture containing table

Description automatically generated

**Section 9: Updating the App.js for the TodoButton**

Now we need to pass toggleComplete and deleteTodo as props to the TodoList from app.js. Update the render method in app.js as shown below,

Text

Description automatically generated

**Section 10: Updating the TodoList component**

We have to update the TodoList.js and pass the toggleComplete and the deleteTodo as props to Todo. Update TodoList.js as shown below,

Text

Description automatically generated

**Section 11: Updating the Todo component again**

To add the new TodoButton component and some styling for the button container, open Todo.js and alter the Todo component as below.

A picture containing text

Description automatically generated

**Section 12: Testing the App component**

If you now test the app with the simulator, you should see the UI as shown below. Check the app's functionality by adding tasks, completing them, and deleting a few tasks. Take a screenshot of your app UI from the mobile and save it under the todos folder.

Graphical user interface, application

Description automatically generated

**Steps to verify the changes using the Simulator:**

1. Open the terminal in the GitHub Codespaces environment.
2. Type **expo start --tunnel** and press Enter to start the expo development server.
3. Wait for the development server to load and show the QR code.
4. Open the "Expo Go" app from your mobile device.
5. Scan the QR code shown in the terminal with the "Expo Go" app or the Camera app (iOS).
6. Wait for the app to load on the mobile device.
7. If you are already connected and want to reload, you can press r in the terminal.

**Section 13: 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 on the Message box on the top. Example, “**Submission for Module04 – Your Name**”
  2. Click on the dropdown beside 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