Lab 3: React Native To-Do List Application

Mahaboob Pasha Mohammad
Instructor: Flavio Esposito
Course: Web Technologies

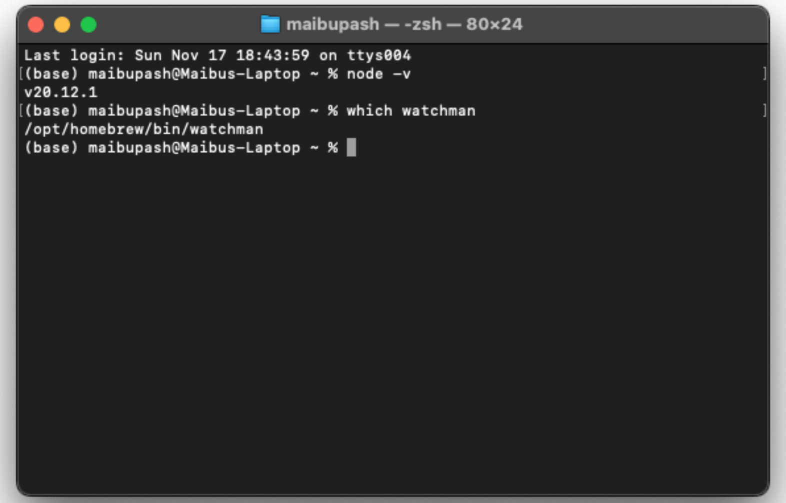
November 18, 2024

Task 1: Set Up the Dev Environment

In this task, you will set up your development environment to build React Native applications.

1.4 Step 1: Install Node.js and Watchman

1. **Install Node.js:**
2. **Install Watchman (Optional for macOS/Linux):**



```
Last login: Sun Nov 17 18:43:59 on ttys004
(base) maibupash@Maibus-Laptop ~ % node -v
v20.12.1
(base) maibupash@Maibus-Laptop ~ % which watchman
/opt/homebrew/bin/watchman
(base) maibupash@Maibus-Laptop ~ %
```

1.5 Step 2: Install React Native CLI

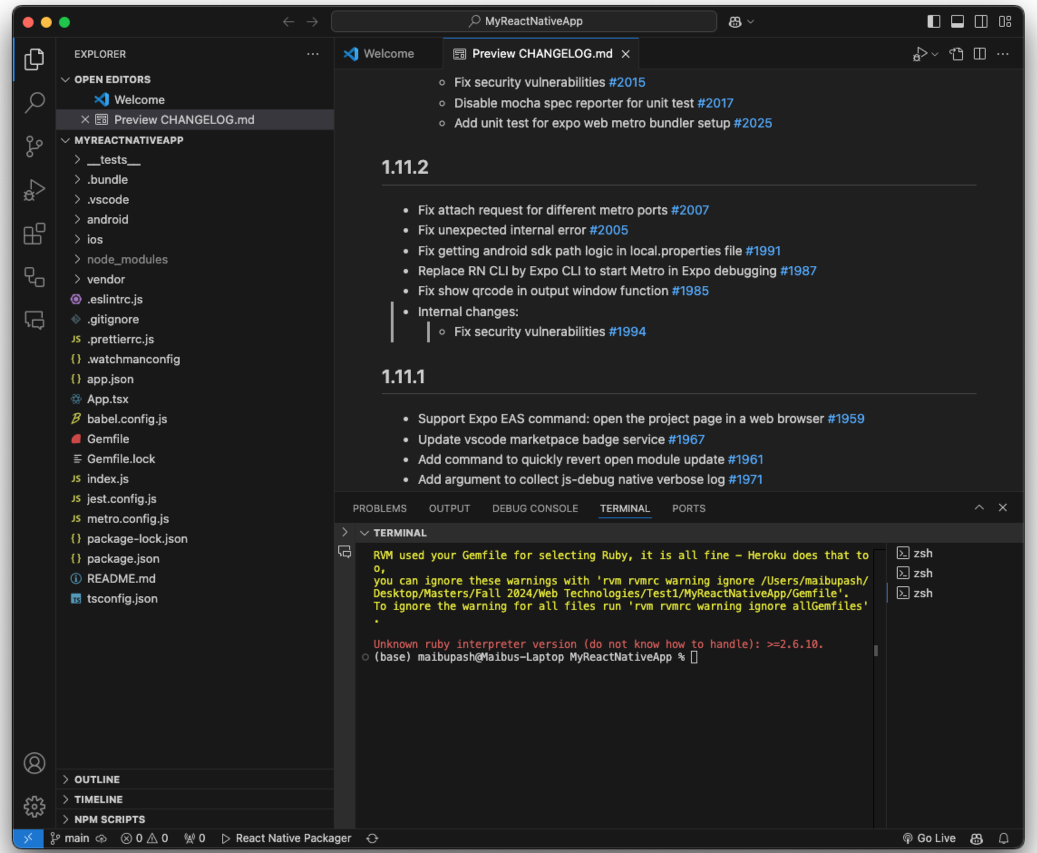
The React Native CLI (Command Line Interface) allows you to create new React Native projects and run them easily.

- Open your terminal and run:

```
npm install -g react-native-cli
```

- Note: If the global 'react-native-cli' package has been deprecated, you can use the local version with 'npx':

```
npx react-native init YourProjectName
```



- Fix security vulnerabilities #2015
- Disable mocha spec reporter for unit test #2017
- Add unit test for expo web metro bundler setup #2025

1.11.2

- Fix attach request for different metro ports #2007
- Fix unexpected internal error #2005
- Fix getting android sdk path logic in local.properties file #1991
- Replace RN CLI by Expo CLI to start Metro in Expo debugging #1987
- Fix show qrcode in output window function #1985
- Internal changes:
  - Fix security vulnerabilities #1994

1.11.1

- Support Expo EAS command: open the project page in a web browser #1959
- Update vscode marketpace badge service #1967
- Add command to quickly revert open module update #1961
- Add argument to collect js-debug native verbose log #1971

```
RVM used your Gemfile for selecting Ruby, it is all fine - Heroku does that to
o,
you can ignore these warnings with 'rvm rvmrc warning ignore /Users/maibupash/
Desktop/Masters/Fall 2024/Web Technologies/Test1/MyReactNativeApp/Gemfile'.
To ignore the warning for all files run 'rvm rvmrc warning ignore allGemfiles'
.

Unknown ruby interpreter version (do not know how to handle): >=2.6.10.
(base) maibupash@Maibus-Laptop MyReactNativeApp % 
```

1.6 Step 3: Set Up Android Studio (or Xcode for iOS)

For Android Users:

(a) Enable Recommended SDK Tools under the SDK Tools tab:
  - Android SDK Build-Tools (install the latest version)
  - Android SDK Platform-Tools
  - Android Emulator
  - Google Play Services (if your app needs Google services)

For iOS Users:

- Ensure Xcode is installed from the App Store.
- Install Xcode Command Line Tools:

```
xcode-select --install
```

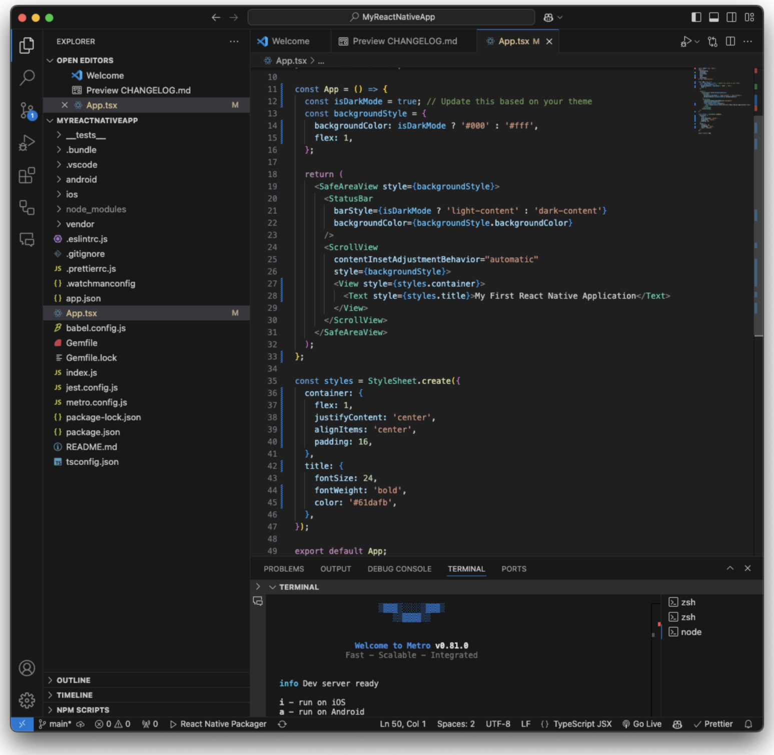
1.7 Step 4: Create a New React Native Project

1. Open your terminal and run:

```
npx react-native init YourProjectName
cd YourProjectName
```

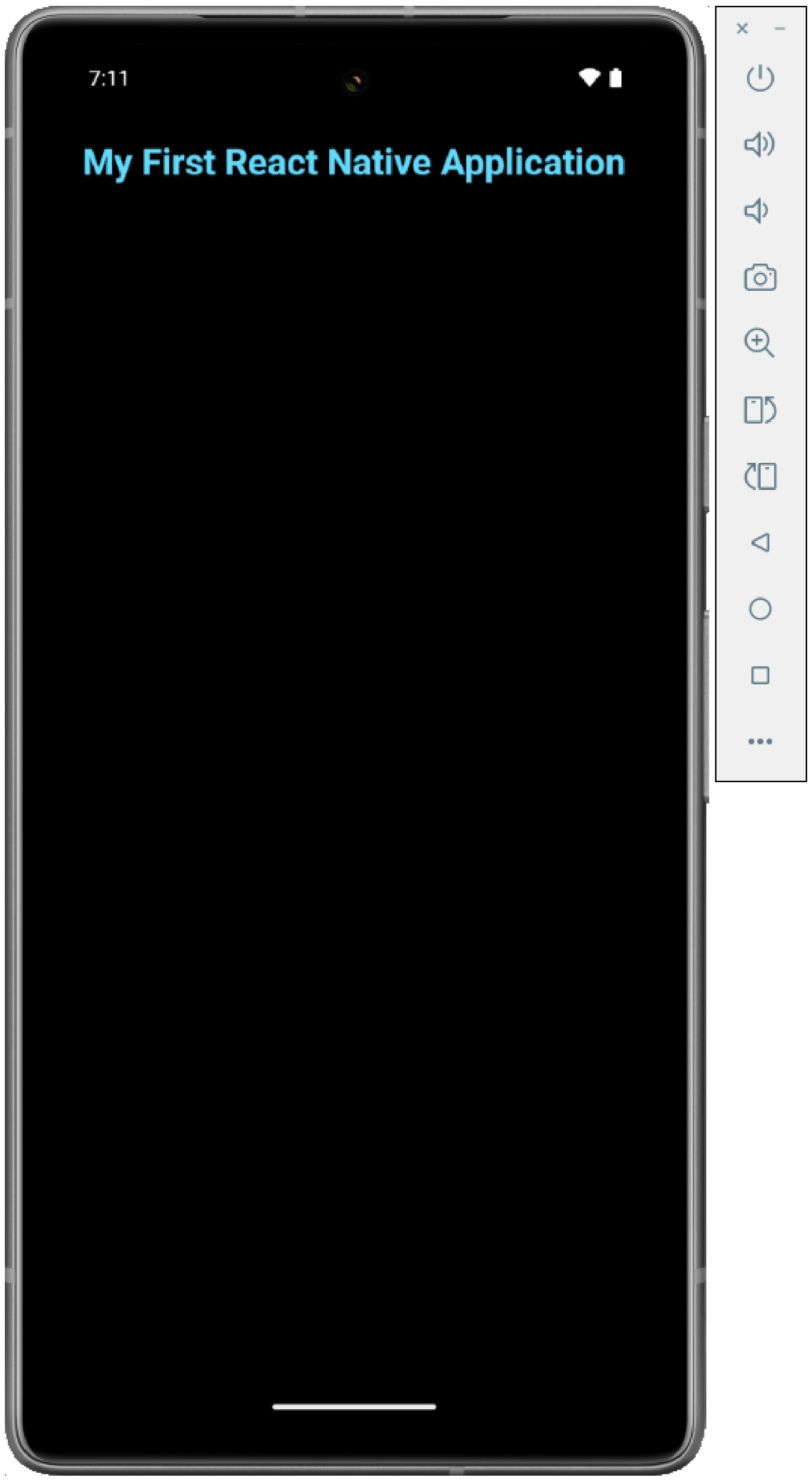
1.8 Step 5: Open the Project in Visual Studio Code

1. Install the React Native Tools extension in VS Code for a better development experience.
2. Open App.js and modify the content to display "My First React Native Application".



```
const App = () => {
  const isDarkMode = true; // Update this based on your theme
  const backgroundStyle = {
    backgroundColor: isDarkMode ? '#000' : '#fff',
    flex: 1,
  };

  return (
    <SafeAreaView style={backgroundStyle}>
      <StatusBar
        barStyle={isDarkMode ? 'light-content' : 'dark-content'}
        backgroundColor={backgroundStyle.backgroundColor}
      />
      <ScrollView
        contentInsetAdjustmentBehavior="automatic"
        style={backgroundStyle}>
        <View style={styles.container}>
          <Text style={styles.title}>My First React Native Application</Text>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 16,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#61dafb',
  },
});

export default App;
```

```
Welcome to Metro v0.81.0
Fast - Scalable - Integrated


info Dev server ready

i - run on iOS
a - run on Android
```



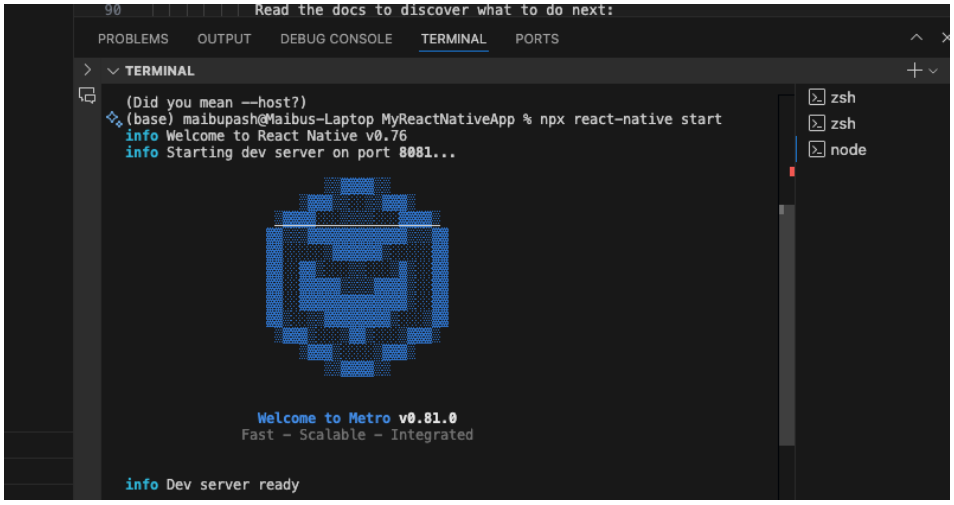
0.1 Step 6: Start the Metro Bundler

The Metro Bundler is a JavaScript bundler specifically for React Native. It watches your files and serves the appropriate JavaScript code.

- In your terminal, run:

```
npx react-native start
```

- This command initializes the Metro Bundler and opens a new terminal window to display logs and errors.



```
Read the docs to discover what to do next:
(Did you mean --host?)
(base) maibupash@Maibus-Laptop MyReactNativeApp % npx react-native start
info Welcome to React Native v0.76
info Starting dev server on port 8081...

              Welcome to Metro v0.81.0
            Fast - Scalable - Integrated

info Dev server ready
```

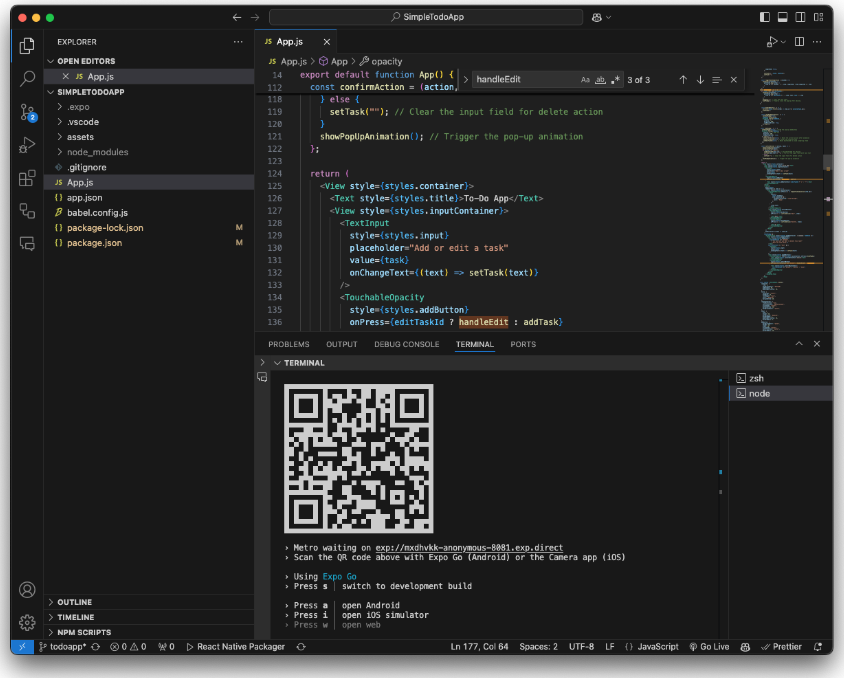
0.2 Step 8: Run Your App on a Mobile Device Using Expo

On November 5th, 2024, we had a guest lecture on Expo.dev. Study those slides before continuing.

1. **Install and create a new Expo project:**

   ```
   npm install -g expo-cli
   npx expo init YourProjectName
   cd YourProjectName
   npx expo start
   ```

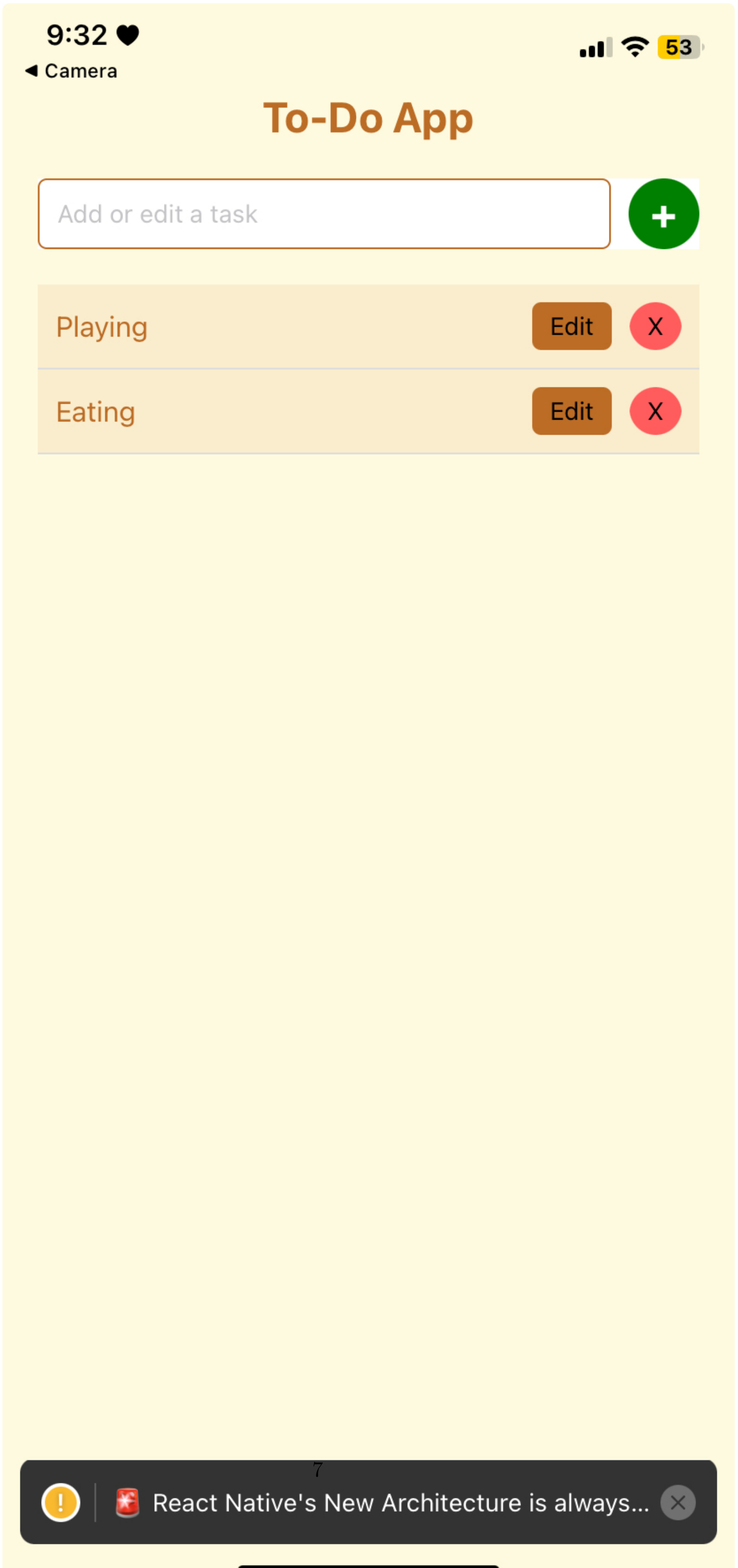
2. **Connect Your Device:**
   - Ensure your mobile device is connected to the same Wi-Fi network as your development machine.
3. **Open the Expo Go App:**
   - Install the Expo Go app from the App Store (iOS) or Google Play Store (Android).
4. **Scan the QR Code:**
   - In the Expo developer tools opened in your browser, you will see a QR code.
   - Use the Expo Go app to scan the QR code.



```
export default function App() {
  const confirmAction = (action,
    } else {
      setTask(""); // Clear the input field for delete action
    }
    showPopUpAnimation(); // Trigger the pop-up animation
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>To-Do App</Text>
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.input}
          placeholder="Add or edit a task"
          value={task}
          onChangeText={(text) => setTask(text)}
        />
        <TouchableOpacity
          style={styles.addButton}
          onPress={editTaskId ? handleEdit : addTask}
```

```
› Metro waiting on exp://mxdhvkk-anonymous-8081.exp.direct
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Using Expo Go
› Press s │ switch to development build

› Press a │ open Android
› Press i │ open iOS simulator
› Press w │ open web
```

5. **Run the App:**
   - Once the QR code is scanned, your React Native app will start running on your mobile device.



To-Do App

Playing

Eating

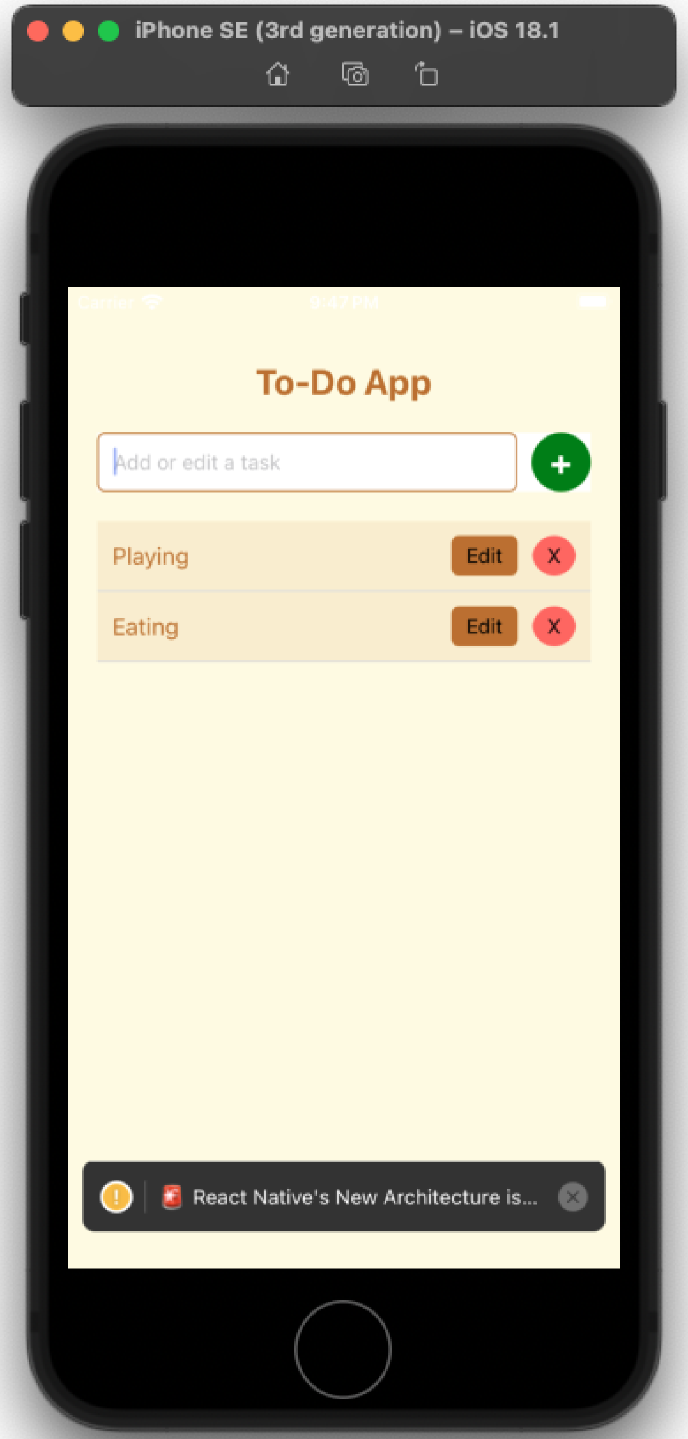
React Native's New Architecture is always...

0.3 What to Submit for Task 1

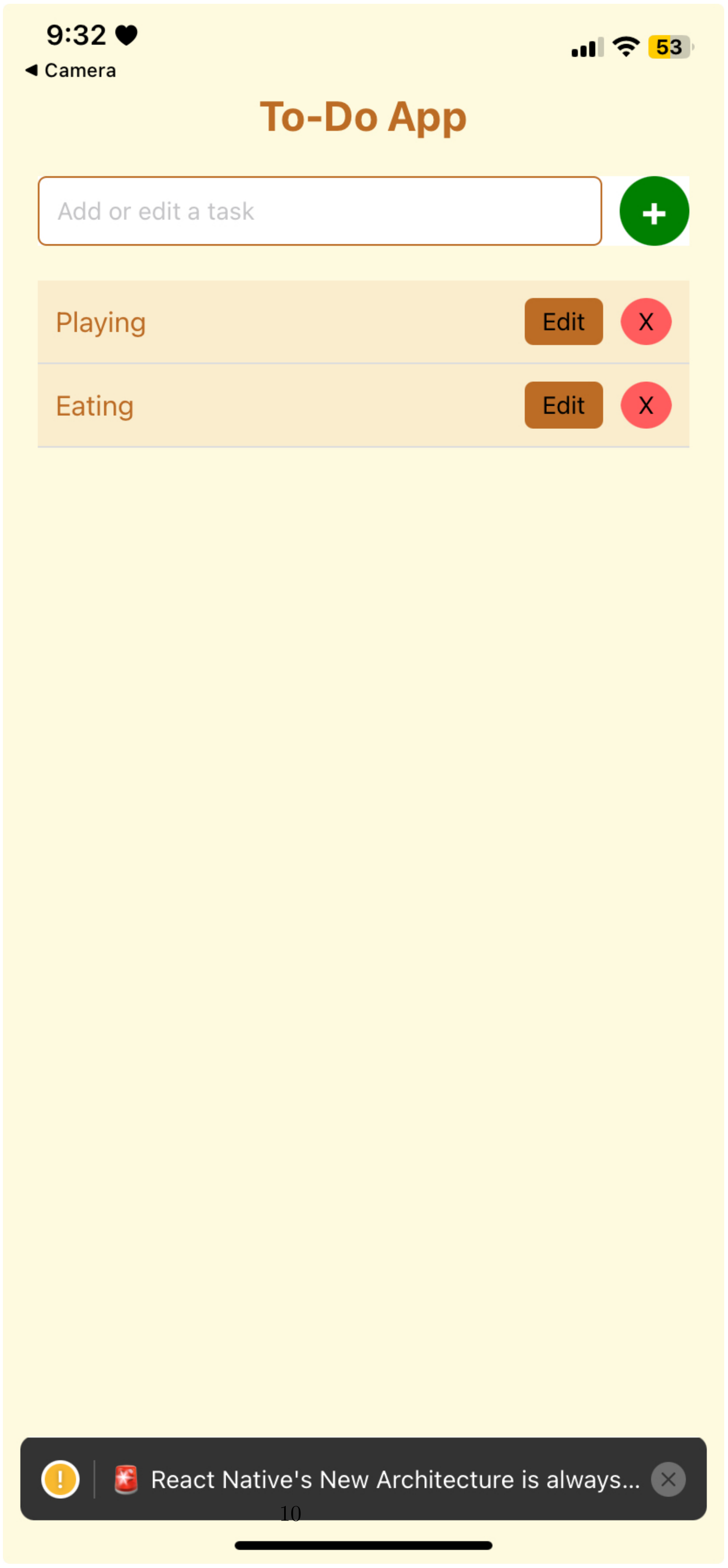
Provide detailed answers to the following questions, including any necessary screenshots:

(a) **Screenshots of Your App (5 Points)**

- Attach screenshots of your app running on an emulator and on a physical Android or iOS device.
  - Screen shot of Emulator:


iPhone SE (3rd generation) – iOS 18.1
To-Do App
Add or edit a task
+
Playing
Edit
X
Eating
Edit
X
React Native's New Architecture is...

- Screen shot of Physical:


9:32
◀ Camera
53

To-Do App

Add or edit a task

Playing — Edit — X

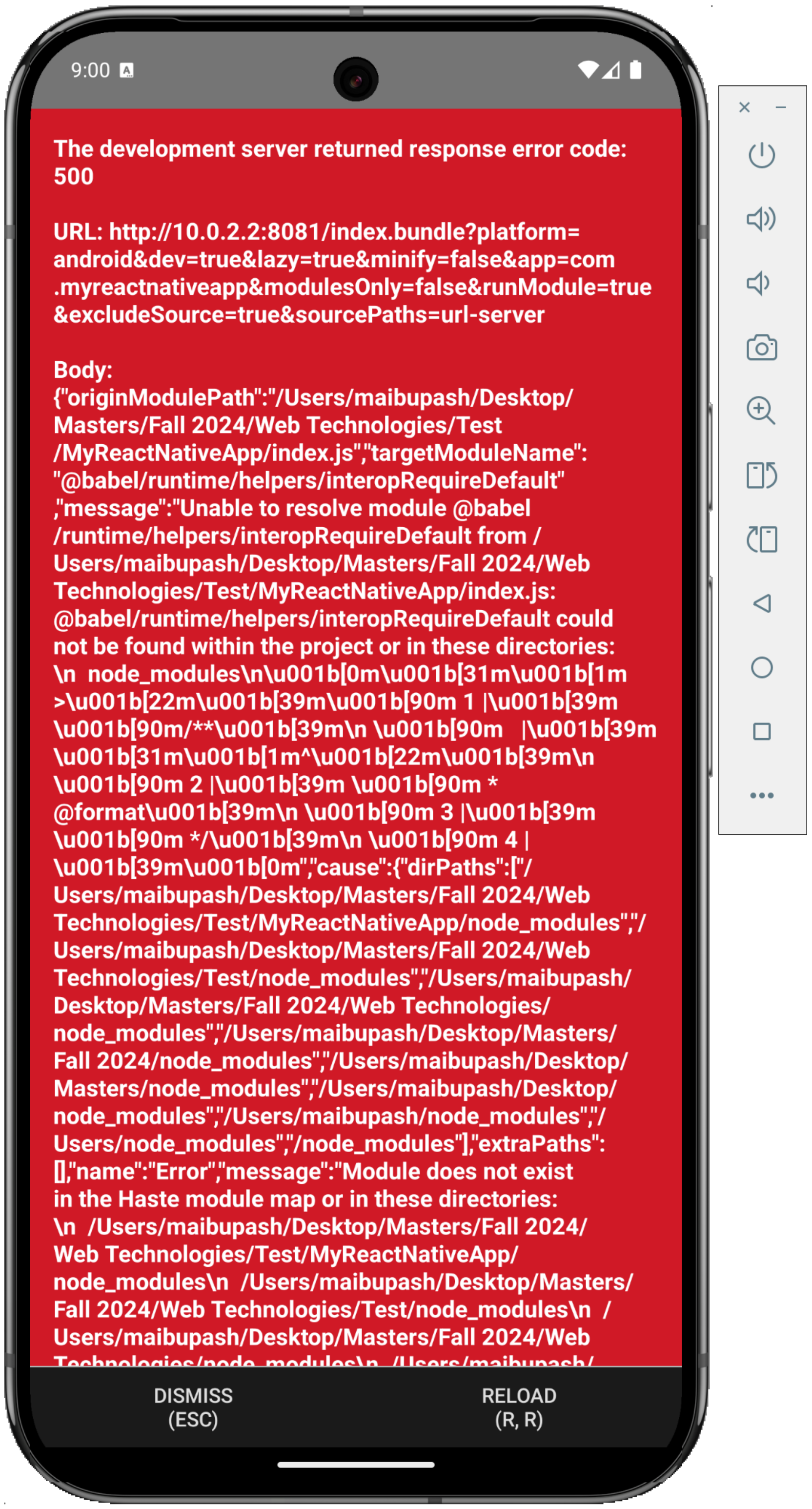
Eating — Edit — X

🚨 React Native's New Architecture is always...

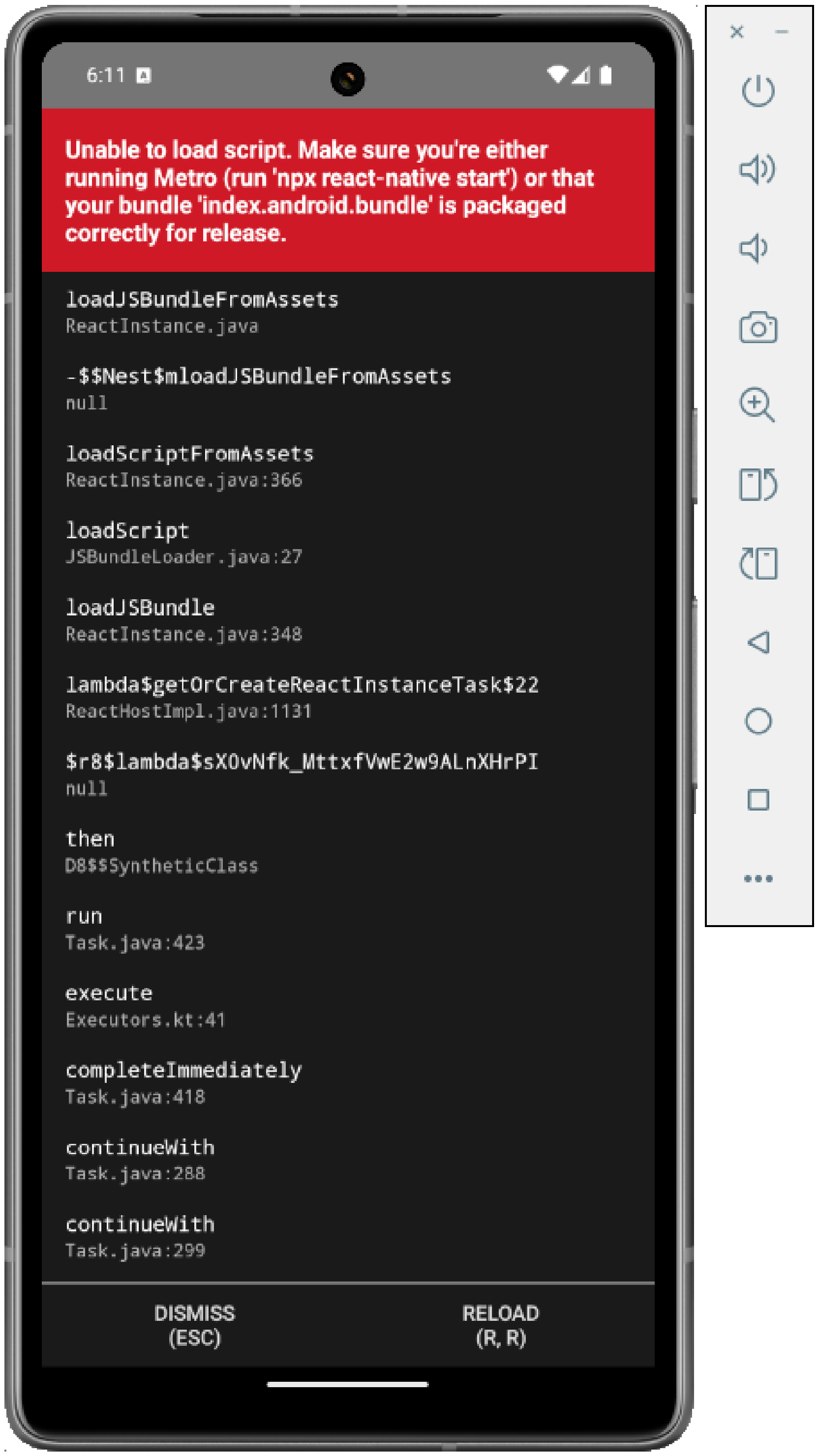
  - **Differences Observed:**
  - The app runs faster on the physical device compared to the emulator.
  - Running the app on emulator is slow, theanimations are rendered fast when compared to the physical device
  - The emulator occasionally showed slower animations and increased load times, likely due to the resources used by the macOS environment.

(b) **Setting Up an Emulator (10 Points)**

- **Steps Followed:**
  1. **Android Studio (Also I have run the app I android studio as the code was giving build errors at the begining):**
     - Installed Android Studio from the official site.
     - Opened the AVD Manager and created a new emulator with Android .
     - Selected the device profile (e.g., Pixel 9 Pro XL API 35) and system image.
     - Launched the emulator from the AVD Manager. and also directly from the terminal .
  2. **Xcode (iOS):**
     - Opened Preferences ¿ Components.
     - Downloaded the desired iOS simulator (e.g., iOS 18.1 for iPhone SE).
     - Opened the project in Xcode, selected the simulator, and built the app.
- **Challenges and Solutions:**
  - **Challenge:** Emulator not appearing or running slowly.


9:00

```
The development server returned response error code:
500

URL: http://10.0.2.2:8081/index.bundle?platform=
android&dev=true&lazy=true&minify=false&app=com
.myreactnativeapp&modulesOnly=false&runModule=true
&excludeSource=true&sourcePaths=url-server

Body:
{"originModulePath":"/Users/maibupash/Desktop/
Masters/Fall 2024/Web Technologies/Test
/MyReactNativeApp/index.js","targetModuleName":
"@babel/runtime/helpers/interopRequireDefault"
,"message":"Unable to resolve module @babel
/runtime/helpers/interopRequireDefault from /
Users/maibupash/Desktop/Masters/Fall 2024/Web
Technologies/Test/MyReactNativeApp/index.js:
@babel/runtime/helpers/interopRequireDefault could
not be found within the project or in these directories:
\n  node_modules\n\u001b[0m\u001b[31m\u001b[1m
>\u001b[22m\u001b[39m\u001b[90m 1 |\u001b[39m
\u001b[90m/**\u001b[39m\n \u001b[90m   |\u001b[39m
\u001b[31m\u001b[1m^\u001b[22m\u001b[39m\n
\u001b[90m 2 |\u001b[39m \u001b[90m *
@format\u001b[39m\n \u001b[90m 3 |\u001b[39m
\u001b[90m */\u001b[39m\n \u001b[90m 4 |
\u001b[39m\u001b[0m","cause":{"dirPaths":["/
Users/maibupash/Desktop/Masters/Fall 2024/Web
Technologies/Test/MyReactNativeApp/node_modules","/
Users/maibupash/Desktop/Masters/Fall 2024/Web
Technologies/Test/node_modules","/Users/maibupash/
Desktop/Masters/Fall 2024/Web Technologies/
node_modules","/Users/maibupash/Desktop/Masters/
Fall 2024/node_modules","/Users/maibupash/Desktop/
Masters/node_modules","/Users/maibupash/Desktop/
node_modules","/Users/maibupash/node_modules","/
Users/node_modules","/node_modules"],"extraPaths":
[],"name":"Error","message":"Module does not exist
in the Haste module map or in these directories:
\n  /Users/maibupash/Desktop/Masters/Fall 2024/
Web Technologies/Test/MyReactNativeApp/
node_modules\n  /Users/maibupash/Desktop/Masters/
Fall 2024/Web Technologies/Test/node_modules\n  /
Users/maibupash/Desktop/Masters/Fall 2024/Web
```

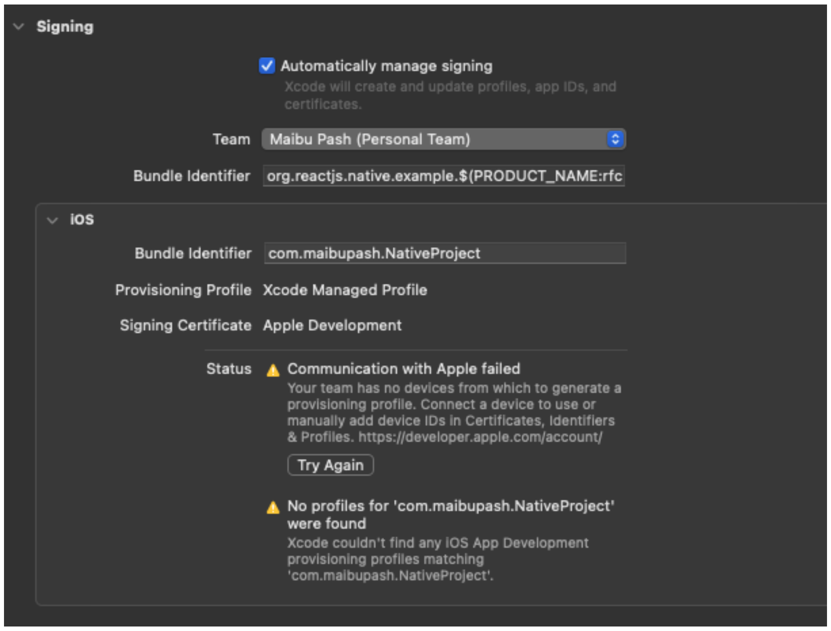


Unable to load script. Make sure you're either running Metro (run 'npx react-native start') or that your bundle 'index.android.bundle' is packaged correctly for release.

```
loadJSBundleFromAssets
ReactInstance.java

-$$Nest$mloadJSBundleFromAssets
null

loadScriptFromAssets
ReactInstance.java:366

loadScript
JSBundleLoader.java:27

loadJSBundle
ReactInstance.java:348

lambda$getOrCreateReactInstanceTask$22
ReactHostImpl.java:1131

$r8$lambda$sXOvNfk_MttxfVwE2w9ALnXHrPI
null

then
D8$$SyntheticClass

run
Task.java:423

execute
Executors.kt:41

completeImmediately
Task.java:418

continueWith
Task.java:288

continueWith
Task.java:299
```

∗ **Solution:** Restarted Android Studio/Xcode and ensured sufficient RAM was allocated to the emulator.

– **Challenge:** Missing simulators in Xcode.



Signing

Automatically manage signing [✓]
Xcode will create and update profiles, app IDs, and certificates.

Team: Maibu Pash (Personal Team)
Bundle Identifier: org.reactjs.native.example.$(PRODUCT_NAME:rfc

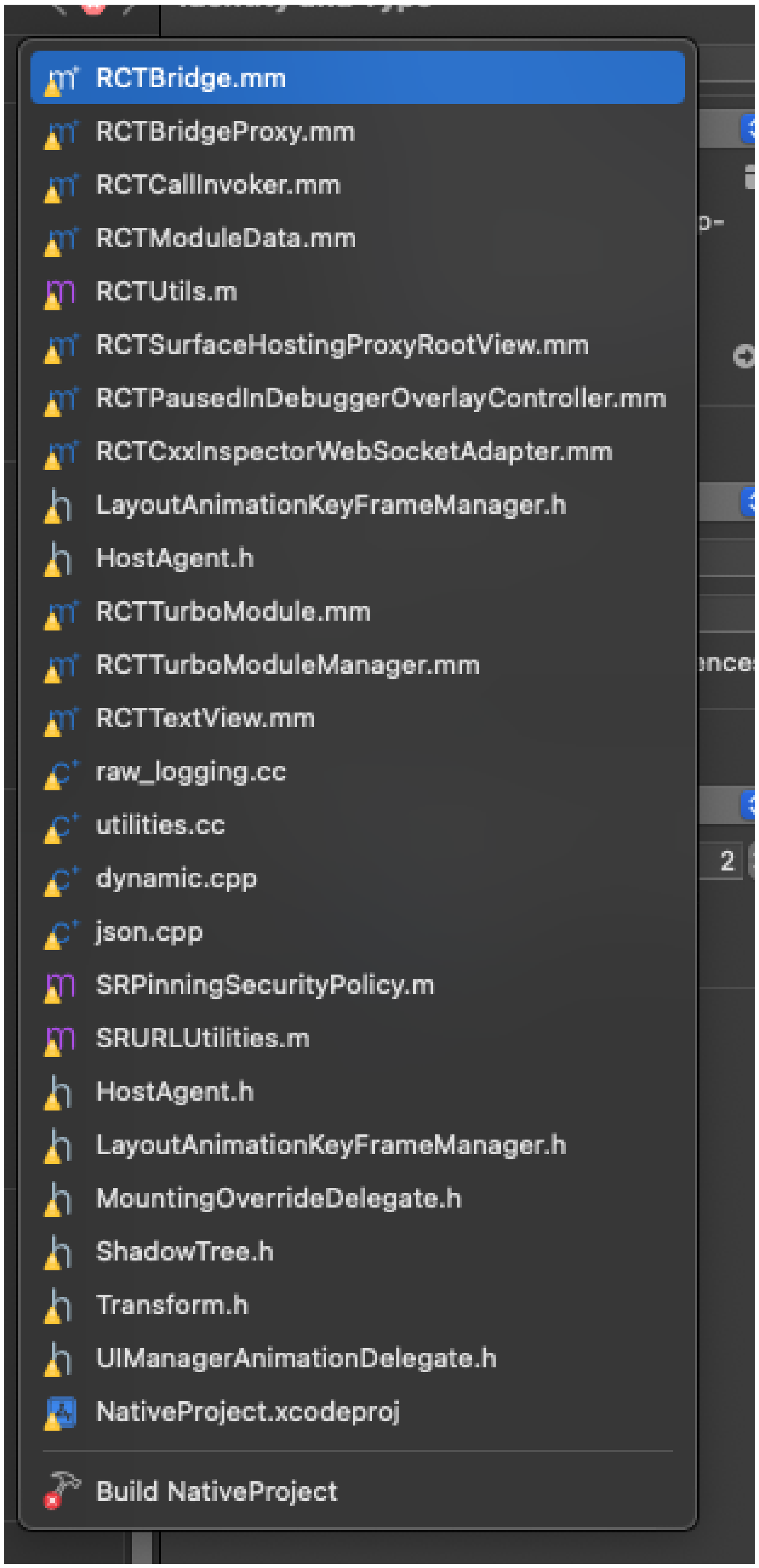
iOS

Bundle Identifier: com.maibupash.NativeProject
Provisioning Profile: Xcode Managed Profile
Signing Certificate: Apple Development

Status: Communication with Apple failed
Your team has no devices from which to generate a provisioning profile. Connect a device to use or manually add device IDs in Certificates, Identifiers & Profiles. https://developer.apple.com/account/

No profiles for 'com.maibupash.NativeProject' were found
Xcode couldn't find any iOS App Development provisioning profiles matching 'com.maibupash.NativeProject'.

* **Solution:** Downloaded missing components through Xcode's Preferences.



(c) **Running the App on a Physical Device Using Expo (10 Points)**

- **Steps:**
  i. Installed Expo Go app on the physical device.
  ii. Initialized the React Native app with Expo CLI using `npx expo init TestProject`.
  iii. Connected the physical device via USB and scanned the QR code generated by the Expo CLI.
  iv. Opened the app on the physical device using Expo Go.
  v. Also I have run the app using the commands like npx expo start –tunnel, which will create a metro bundler and will generate a QR code, by scanning the QR code , we can open the app in expo go of mobile
- **Troubleshooting Steps:**
  - **Issue:** Expo CLI not detecting the physical device.
    - **Solution:** Ensured the device was connected to the same Wi-Fi network as the computer.
  - **Issue:** Expo CLI was having the build errors physical device.
    - **Solution:** Ensured I have downloaded the necessary dependencies and updated versions which are suitable with the environments set up of expo app
  - **Issue:** Expo QR code not working.
    - **Solution:** Manually entered the IP address provided by Expo CLI in the Expo Go app.

0.4 Comparison of Emulator vs. Physical Device

- **Performance:**
  - Emulator: Slower, have build errros due to sharing system resources.
  - Physical Device: Faster and smoother animations. Run time is fast.
- **Testing Device-Specific Features:**
  - Emulator: Limited to the emulator's capabilities.
  - Physical Device: Allows testing of hardware features such as the camera. many features and fast application loading.
- **Setup:**
  - Emulator: Easy to set up and switch between multiple device profiles.
  - Physical Device: Requires additional configuration, such as signing.
- **Real-World Accuracy:**
  - Emulator: Simulates device behavior but is not entirely realistic.
  - Physical Device: Reflects real-world conditions more accurately.
- **Advantages:**
  - Emulator: Quick and flexible for testing multiple device profiles.
  - Physical Device: Better for debugging and testing on actual hardware.
- **Disadvantages:**
  - Emulator: May not accurately reflect physical performance or real-device behavior.
  - Physical Device: Requires access to physical hardware and may involve extra setup time.

(d) **Troubleshooting a Common Error (5 Points)**

- **Error:** `"xcodebuild" exited with error code '65'.`
- **Cause:**
  - This error is related to Xcode's inability to build the app due to signing issues or incorrect project settings.
- **Resolution:**
  i. Opened the project in Xcode (`TestProject.xcworkspace`).

ii. Configured signing under Signing & Capabilities:
   - Selected the correct development team.
   - Ensured the bundle identifier matched the provisioning profile.

iii. Ran `pod install` in the `ios` folder to ensure all dependencies were correctly installed.

iv. Cleaned the Xcode build folder and rebuilt the project. Have to install the necessary dependencies like wacthan, pods, ruby, etc,.

v. Verified the iOS Deployment Target matched the iOS simulator version.

vi. Both the devices have trouble shooting errors if we don't have necessary dependencies for the app to build and run the appplication

- **Reflection:**
  - This error highlighted the importance of proper environment setup and dependency management.
  - the errors made to dig deep with build process and errors such as error 65 in mac xcode to build the application and show in the emulator. and signing capabilities to show the physical device as well. Made me to go through all the errors and solve them step by step. Also the google search gives multiple answers rather than switching to the llm, which only gives a one solution.

0.5 Task 2

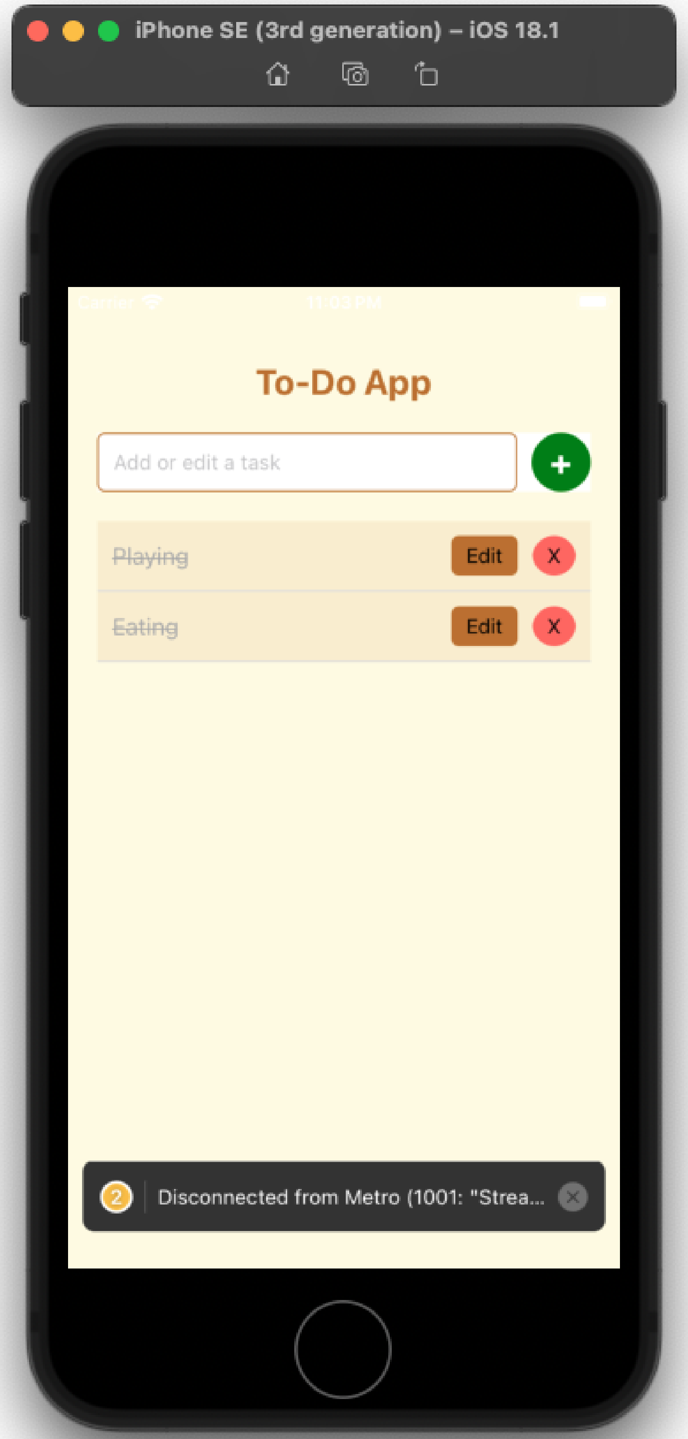
Provide detailed answers to the following questions, including any necessary screenshots:

0.5.1 Extending Functionality

Implement the following features to extend the functionality of your To-Do List app. Attach screenshots for each implemented feature and explain how you implemented them in your report for full points.

(a) **Mark Tasks as Complete (15 Points)**

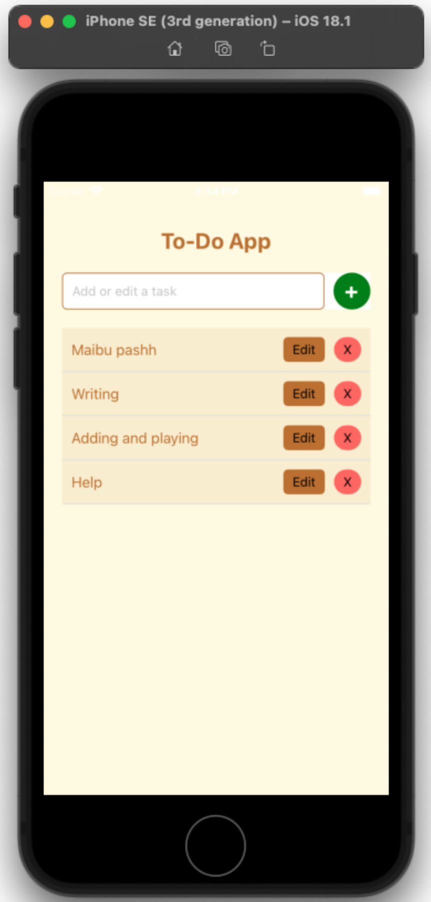
- Add a toggle function that allows users to mark tasks as completed.
- Style completed tasks differently, such as displaying strikethrough text or changing the text color.
- Explain how you updated the state to reflect the completion status of tasks.
- Answer: The toggleTaskCompletion function was implemented to update the completed property for the selected task. The setTasks function was used to update the tasks array in the React state.
- Conditional Styling: The UI was updated dynamically by applying conditional styles based on the completed property of the task. If item.completed is true, a strikethrough and lighter text color were applied.
- React Re-Rendering: When the tasks state is updated using setTasks, React triggers a re-render of the component. This ensures the updated completed status and styling are immediately reflected in the UI.


iPhone SE (3rd generation) – iOS 18.1
To-Do App
Add or edit a task
+
~~Playing~~
Edit
X
~~Eating~~
Edit
X
Disconnected from Metro (1001: "Strea...

(b) **Persist Data Using AsyncStorage (15 Points)**

- Implement data persistence so that tasks are saved even after the app is closed.

- Use `AsyncStorage` to store and retrieve the tasks list.
- Answer: Whenever the tasks state is updated, we use AsyncStorage.setItem to store the serialized tasks list.This ensures the tasks are saved persistently.
  - When the app starts, we use AsyncStorage.getItem inside a useEffect to load tasks from storage.The retrieved data is deserialized and set as the initial tasks state.



(c) **Edit Tasks (10 Points)**

- Allow users to tap on a task to edit its content.
- Implement an update function that modifies the task in the state array.
- Explain how you managed the UI for editing tasks.

- Answer: **Managing the UI for Editing Tasks**
  To manage the UI for editing tasks, the following approach was implemented:
  - **Triggering the Edit Pop-Up:** When the user clicks the `Edit` button on a task, a pop-up is triggered to allow editing. The `confirmAction` function is called to set the current action to `"edit"` and pre-fill the input field with the task's text.
  - Code snippet:

```
const confirmAction = (action, task) => {
    setCurrentAction(action);
    setSelectedTask(task);
    if (action === "edit") {
        setEditTaskId(task.id); // Set the task ID for editing
        setTask(task.text || ""); // Pre-fill input with task text
    }
    showPopUpAnimation(); // Show the pop-up
};
```

– **Pre-Filled Input Field:** The pop-up contains an input field pre-filled with the task's current text to allow the user to modify it.

```
{currentAction === "edit" && (
    <TextInput
        style={styles.input}
        value={task}
        onChangeText={(text) => setTask(text)}
    />
)}
```

– **Saving the Changes:** Once the user modifies the task and clicks `Save`, the `handleEdit` function updates the task's text in the `tasks` state, clears the input field, and closes the pop-up.

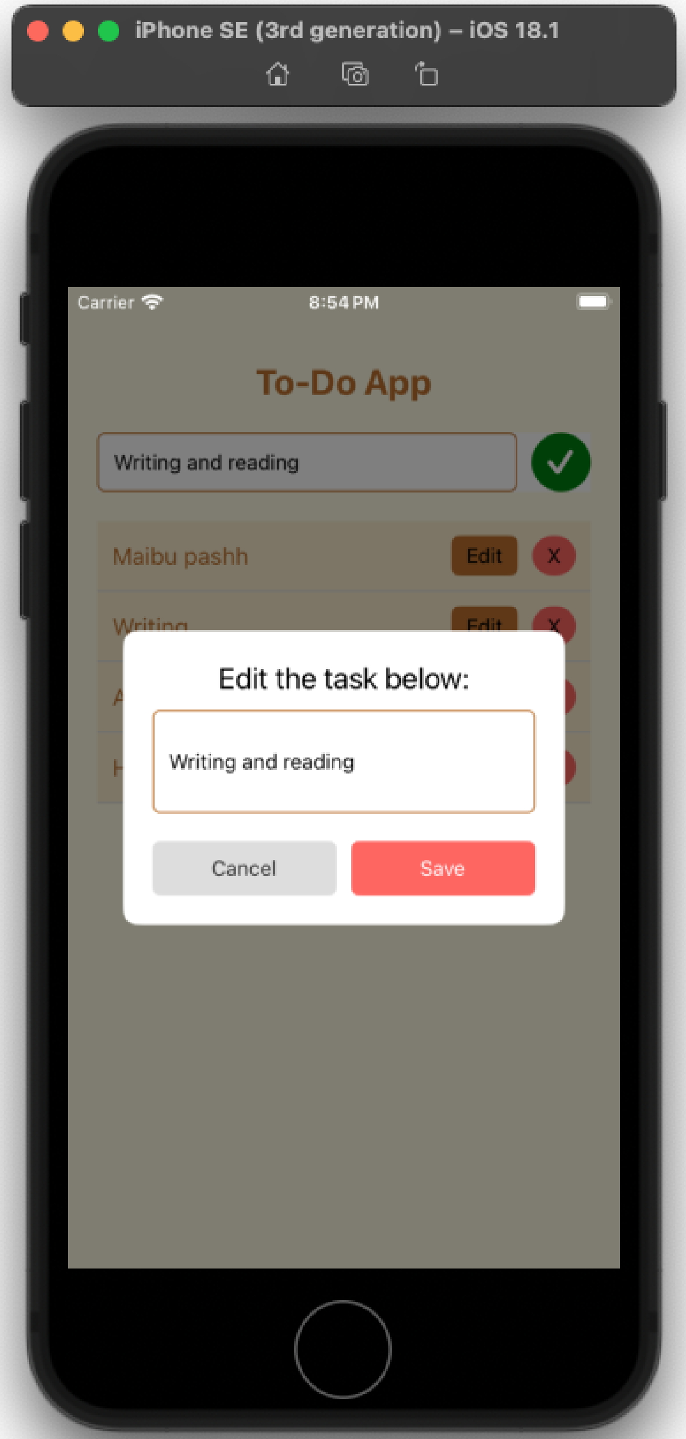
```
const handleEdit = () => {
    if (task.trim() && editTaskId) {
        setTasks((prevTasks) =>
            prevTasks.map((item) =>
                item.id === editTaskId ? { ...item, text: task } : item
            )
        );
        setTask(""); // Clear input
        setEditTaskId(null); // Reset edit state
        hidePopUp(); // Close the pop-up
    }
};
```

– **Closing the Pop-Up:** The `hidePopUp` function uses the `Animated` API to create a fade-out effect. It also resets the relevant states (`editTaskId`, `selectedTask`, etc.).

```
const hidePopUp = () => {
    Animated.timing(fadeAnim, {
        toValue: 0,
        duration: 300,
        useNativeDriver: true,
    }).start(() => {
        setShowPopUp(false);
        setEditTaskId(null);
        setSelectedTask(null);
    });
};
```

– **Dynamic UI Update:** React automatically re-renders the component to reflect the updated task in the task list.

This approach ensures a seamless and user-friendly editing experience with dynamic updates and smooth transitions.



(d) **Add Animations (10 Points)**

- Use the `Animated` API from React Native to add visual effects when adding or deleting tasks.

- Describe the animations you implemented and how they enhance user experience.
- **Answer:** Animations and Enhancing User Experience
  To improve the user experience, animations were implemented for various UI transitions using the `Animated` API in React Native. These animations provide smooth visual feedback and make interactions more intuitive.
  - **Fade-In and Fade-Out Animations:** - Code snippet provided -To improve the user experience, animations were implemented using the `Animated` API in React Native. These animations provide smooth visual feedback during various interactions, making the app more intuitive and visually appealing.
  - A fade-in and fade-out animation was applied to the pop-up used for editing and deleting tasks. This animation is controlled by the `fadeAnim` property, which adjusts the opacity of the pop-up. When the pop-up appears, the `showPopUpAnimation` function sets the initial opacity to zero and animates it to one, creating a fade-in effect. Similarly, the `hidePopUp` function fades the pop-up out by animating the opacity back to zero before closing it.
  - Additionally, a fade-in effect was used for tasks being added to the list. This ensures that newly added tasks appear smoothly, enhancing the clarity and flow of the user interface. For this, the `animateTaskAddition` function sets the opacity of the task to zero initially and animates it to full visibility.
  - These animations enhance the user experience by providing seamless transitions that help users focus on changes within the app, such as tasks being added, edited, or removed. The fade-in and fade-out effects make the appearance and disappearance of elements more natural and visually pleasing, reducing abruptness. Overall, these effects contribute to a polished and professional feel for the application.
  - Provided Code snippets:

```
const showPopUpAnimation = () => {
    setShowPopUp(true);
    fadeAnim.setValue(0); // Reset animation
    Animated.timing(fadeAnim, {
        toValue: 1, // Fade in
        duration: 300,
        useNativeDriver: true,
    }).start();
};

const hidePopUp = () => {
    Animated.timing(fadeAnim, {
        toValue: 0, // Fade out
        duration: 300,
        useNativeDriver: true,
    }).start(() => setShowPopUp(false)); // Hide pop-up after animation
};
```

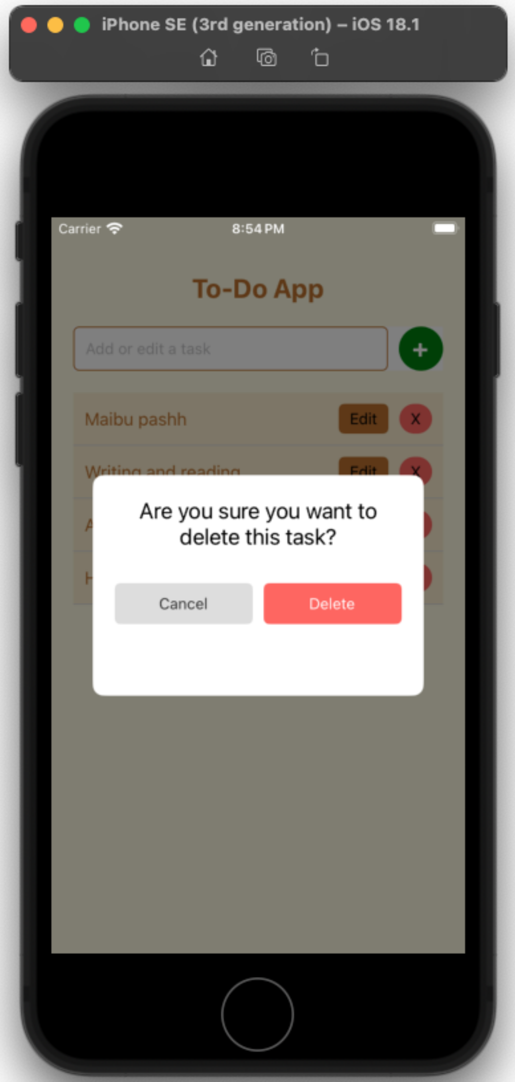
  - **Dynamic Task Feedback:** - A fade-in effect was also implemented for tasks being added to the list. - This ensures tasks appear smoothly, enhancing the visual clarity of the UI.

```
const animateTaskAddition = () => {
    fadeAnim.setValue(0); // Reset to invisible
    Animated.timing(fadeAnim, {
        toValue: 1, // Fade in
        duration: 500,
        useNativeDriver: true,
    }).start();
};
```

How These Animations Enhance UX:

* Animations provide smooth transitions that help users focus on the changes happening in the app, such as tasks being added, edited, or removed.
* The fade-in and fade-out effects make the pop-up appearance and disappearance more natural and visually appealing.
* These effects reduce abruptness, leading to a more polished and professional user experience.

Animations like delete feature which pops up



Are you sure you want to delete this task?

0.6 GitHub Repository

The complete source code for the project, including all implemented features and documentation, is available on GitHub. You can access the repository using the following link:

`https://github.com/miabu-pashh/Lab3/tree/todoapp`

Native project `https://github.com/miabu-pashh/Lab3Reactnative`

Feel free to explore the repository for further details or to contribute.

0.7 Use of AI Tools

AI tools were utilized throughout the project to enhance efficiency and address issues effectively. These tools assisted in:

- Debugging errors in the application code and identifying potential issues with state management and animations.
- Hlpeful in Formatting and structuring the LaTeX document,

The use of AI tools significantly reduced development time and ensured a smoother implementation of features, ultimately contributing to a polished and well-documented project.