React Native Development Environment Setup and Todo List Application

Harshil Sharma

November 17, 2024

GitHub Repository: https://github.com/xxender13/Lab3

1 Introduction to React Native

React Native is an open-source framework developed by Facebook that allows developers to create mobile applications using JavaScript and React. Its primary advantage is the ability to develop apps for both iOS and Android platforms from a single codebase, significantly reducing development time and effort. This means you can write your application once and deploy it on both platforms, taking advantage of native performance and user experience.

1.1 Key Features of React Native

- **Cross-Platform Compatibility:** Build applications that run seamlessly on both iOS and Android.
- **Hot Reloading:** Instantly see the results of changes made to your code, speeding up the development process.
- **Rich Ecosystem:** Access a wide range of libraries and components, making it easier to add complex functionalities.

Task 1: Set Up the Development Environment (50 Points)

In this task, I set up the development environment to build React Native applications.

1.2 Step 1: Install Node.js and Watchman

To complete this step:

1. Installed Node.js from the official website, which also included npm.
2. Installed Watchman (optional) using Homebrew on macOS:

   ```
   brew install watchman
   ```

1.3 Step 2: Install React Native CLI

Installed the React Native CLI using:

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

Alternatively, used npx:

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

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

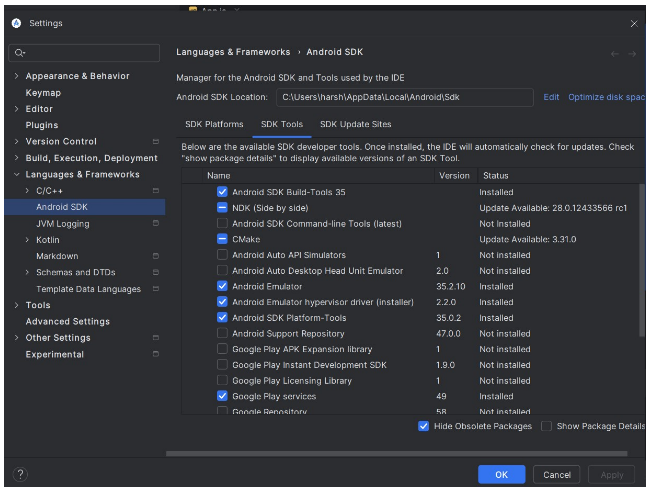
For Android:

1. Installed Android Studio and enabled SDK tools including Android SDK Build-Tools, Platform-Tools, Emulator, and Google Play Services.
2. The setup was verified as shown in Figure 1.

For iOS:

- Installed Xcode and command line tools using:

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



Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\Users\harsh\AppData\Local\Android\Sdk

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

	Name	Version	Status
✓	Android SDK Build-Tools 35		Installed
–	NDK (Side by side)		Update Available: 28.0.12433566 rc1
☐	Android SDK Command-line Tools (latest)		Not Installed
–	CMake		Update Available: 3.31.0
☐	Android Auto API Simulators	1	Not installed
☐	Android Auto Desktop Head Unit Emulator	2.0	Not installed
✓	Android Emulator	35.2.10	Installed
✓	Android Emulator hypervisor driver (installer)	2.2.0	Installed
✓	Android SDK Platform-Tools	35.0.2	Installed
☐	Android Support Repository	47.0.0	Not installed
☐	Google Play APK Expansion library	1	Not installed
☐	Google Play Instant Development SDK	1.9.0	Not installed
☐	Google Play Licensing Library	1	Not installed
✓	Google Play services	49	Installed
☐	Google Repository	58	Not installed

✓ Hide Obsolete Packages ☐ Show Package Details

Figure 1: Android SDK Setup in Android Studio

1.5 Step 4: Create a New React Native Project

Initialized a new project using:

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

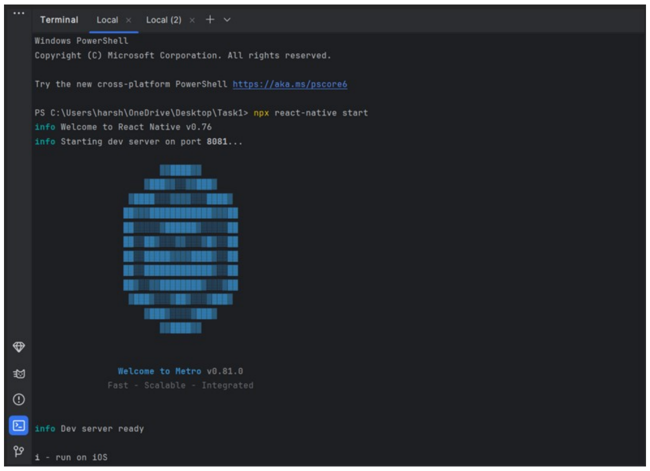
1.6 Step 5: Open the Project in Visual Studio Code

Opened the folder in VS Code and installed the React Native Tools extension.

1.7 Step 6: Start the Metro Bundler

Started the Metro Bundler using:

```
npx react-native start
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\harsh\OneDrive\Desktop\Task1> 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

i - run on iOS
```

Figure 2: Metro Bundler Started in Terminal

1.8 Step 7: Run the App on Emulator or Device

For Android:

```
npx react-native run-android
```

For iOS:

```
npx react-native run-ios
```

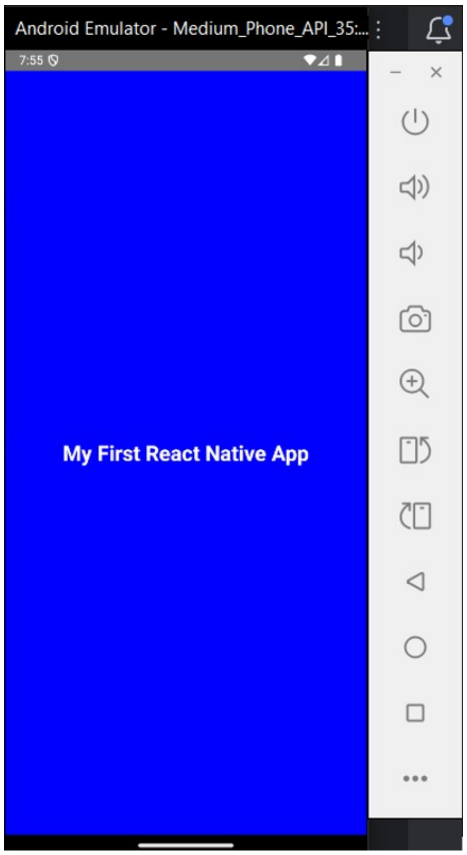

Android Emulator - Medium_Phone_API_35:...
7:55
My First React Native App

Figure 3: App running on Android Emulator

1.9 Step 8: Run the App Using Expo

Installed and created a new Expo project:

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

Connected a physical device using the Expo Go app.

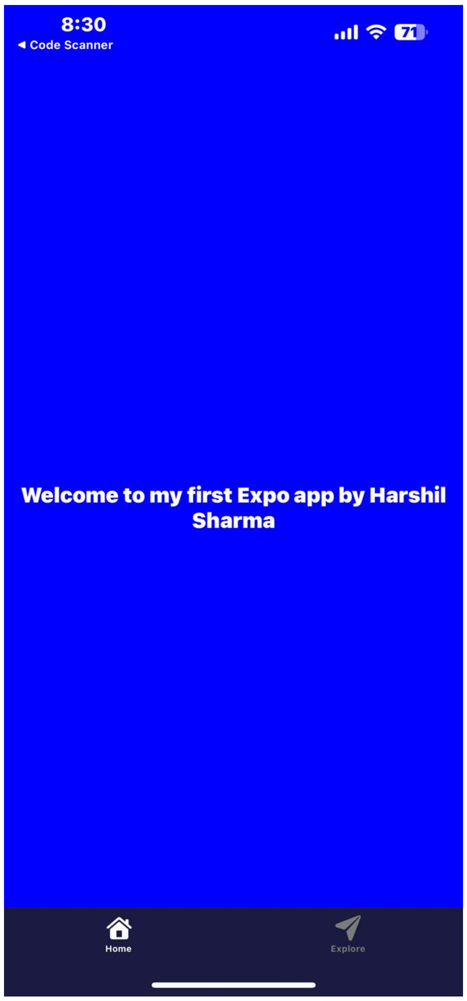


Figure 4: App running on Physical Device using Expo

1.10 Submission Requirements for Task 1

- **Screenshots**:
  - Figure 3 shows the app running on the Android emulator.
  - Figure 4 shows the app running on a physical device using Expo.
  - Figure 2 shows the Metro Bundler running in the terminal.
- **Setting Up an Emulator**: Steps to set up the emulator are explained in Section 1. Challenges faced included issues with hardware acceleration, which were resolved by enabling virtualization in BIOS.

- **Running on a Physical Device Using Expo**: The process for running the app on a physical device is explained in Step 8, including troubleshooting connection issues.
- **Comparison of Emulator vs. Physical Device**:
  - Advantages of Emulator: Easier debugging, supports multiple device configurations.
  - Disadvantages of Emulator: Requires more resources, can be slower.
  - Advantages of Physical Device: Realistic testing experience, better performance.
  - Disadvantages of Physical Device: Requires manual setup, debugging can be cumbersome.
- **Troubleshooting Common Errors**:
  - Encountered an issue with the default `App.tsx` file. Resolved by changing the extension to `App.js`.
  - JAVA_HOME path was not being verified. Used an LLM to obtain the correct command for Visual Studio to fix the path.

2 Task 2: Building a Simple To-Do List App (60 Points)

In this task, I built a simple To-Do List application using React Native.

2.1 App Features

- **Add New Tasks:** Users can input text into a form and add it as a task to the to-do list.

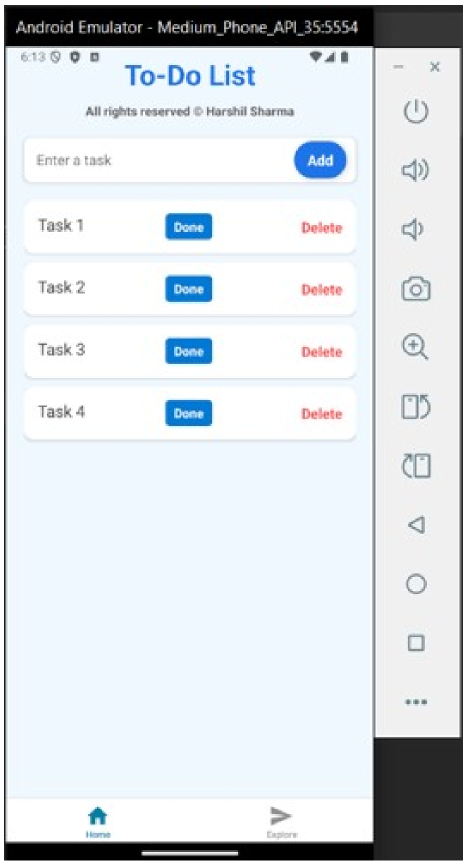
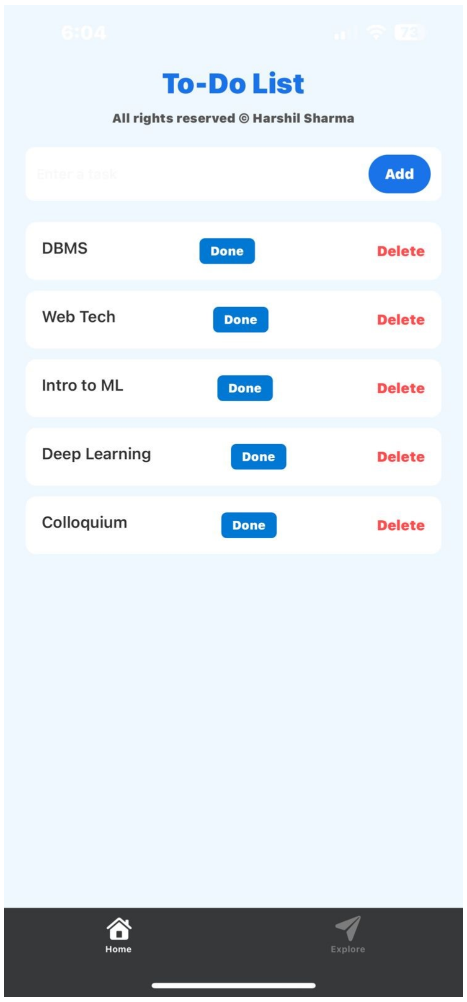


Figure 5: Adding Task on Emulator



To-Do List

All rights reserved © Harshil Sharma

DBMS

Web Tech

Intro to ML

Deep Learning

Colloquium

Figure 6: Adding Task on Physical Device

- **Update Existing Tasks:** Users can modify tasks they have already created.

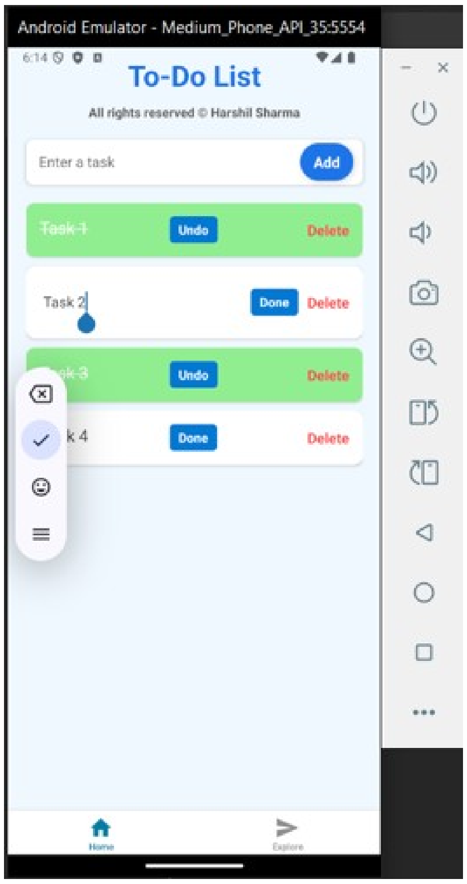
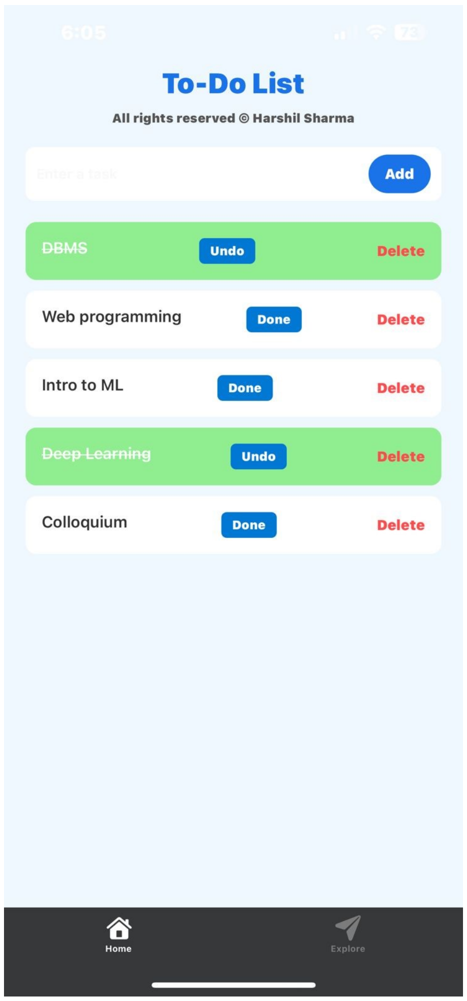


Figure 7: Editing Task on Emulator


To-Do List

All rights reserved © Harshil Sharma

~~DBMS~~	Undo	Delete
Web programming	Done	Delete
Intro to ML	Done	Delete
~~Deep Learning~~	Undo	Delete
Colloquium	Done	Delete

Figure 8: Editing Task on Physical Device

- **Delete Tasks:** Users can remove tasks from the list.

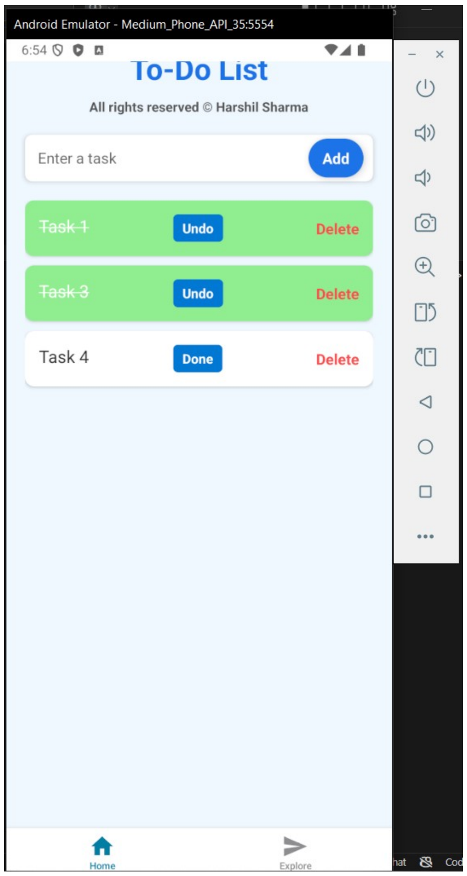


Figure 9: Deleting Task on Emulator

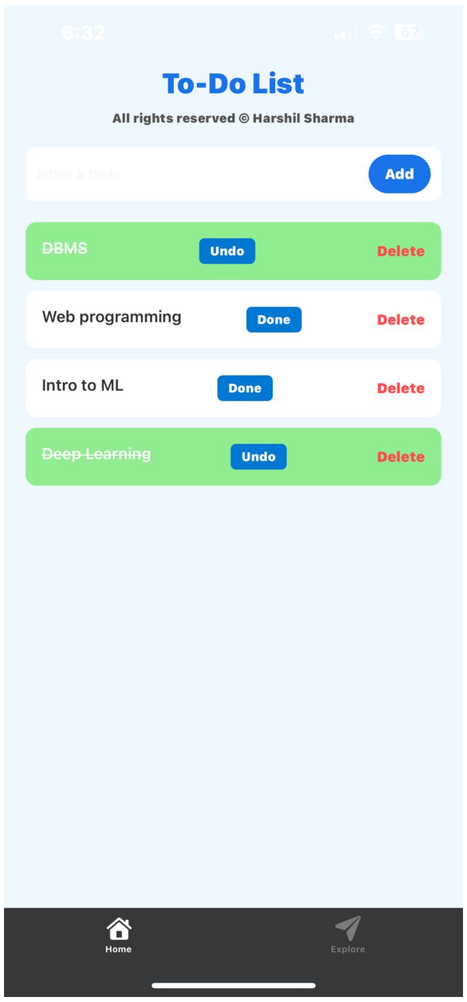


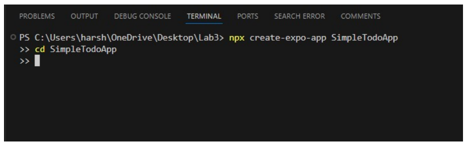
Figure 10: Deleting Task on Physical Device

- **Scrollable Task List:** The to-do list supports scrolling, allowing navigation through a large number of tasks.
- **User-Friendly Interface:** The app provides a simple and intuitive interface for managing tasks.

2.2 Step 1: Set Up the Project

1. Create and navigate to the new project:

```
npx react-native init SimpleTodoApp
cd SimpleTodoApp
```



```
PS C:\Users\harsh\OneDrive\Desktop\Lab3> npx create-expo-app SimpleTodoApp
>> cd SimpleTodoApp
>>
```

Figure 11: Setting up the To-Do List Project

2. Open the project in Visual Studio Code:

```
code .
```

2.3 Step 2: Create the Basic To-Do List Structure

Replace the content of `App.js` with the following code:

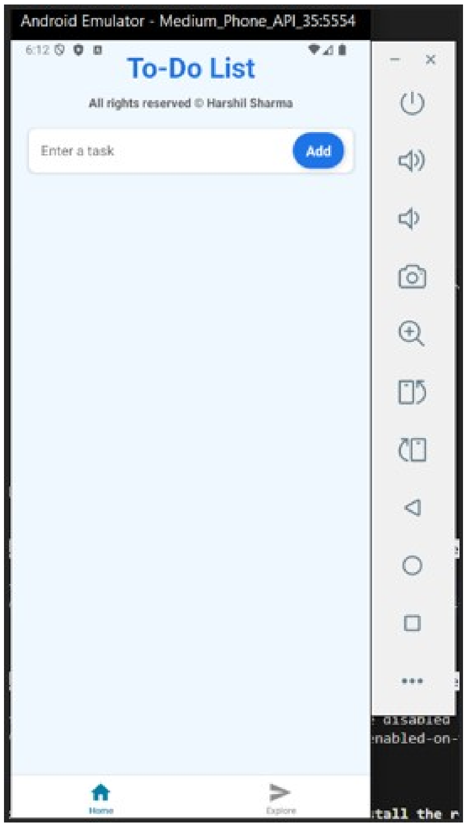

Android Emulator - Medium_Phone_API_35:5554
6:12
To-Do List
All rights reserved © Harshil Sharma
Enter a task
Add
Home
Explore

Figure 12: Initial View of To-Do List App on Android Emulator

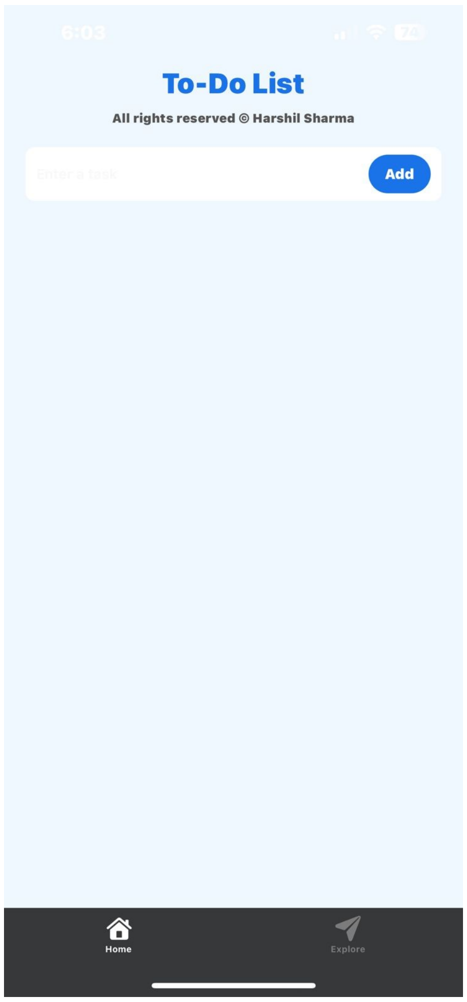


Figure 13: Initial View of To-Do List App on Physical Device

2.4 Explanation of the Code

- **State Management**
  - The `useState` hook is used to manage the state of the input field (task) and the list of tasks (tasks).
  - When a new task is added, it updates the tasks array, and the input field is cleared.
- **Adding a Task**
  - The `addTask` function checks if the input is not empty.

  - It adds a new task with a unique ID (using the current timestamp) to the tasks array.
  - The input field is then reset to an empty string.

- **Deleting a Task**
  - The `deleteTask` function filters out the task with the specified ID from the tasks array.
  - This updates the state and re-renders the list without the deleted task.

- **Rendering the List**
  - The `FlatList` component efficiently renders the list of tasks.
  - Each item in the list displays the task text and a delete button.

2.5 Step 4: Running the App

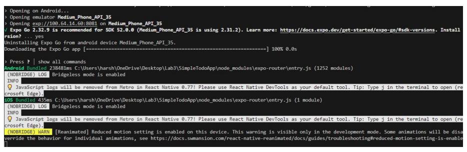
1. In your terminal, run:

   ```
   npx react-native run-android
   ```

   or

   ```
   npx react-native run-ios
   ```

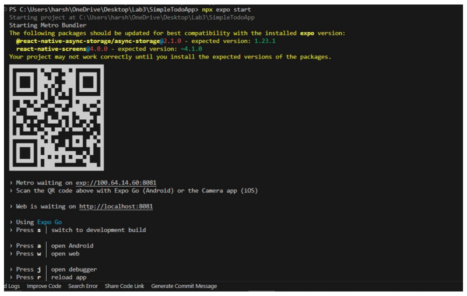
2. This compiles and runs your app on the selected platform.



```
› Opening on Android...
› Opening emulator Medium_Phone_API_35
› Opening exp://100.64.14.60:8081 on Medium_Phone_API_35
√ Expo Go 2.32.9 is recommended for SDK 52.0.0 (Medium_Phone_API_35 is using 2.31.2). Learn more: https://docs.expo.dev/get-started/expo-go/#sdk-versions. Install
rsion? ... yes
Uninstalling Expo Go from android device Medium_Phone_API_35.
Downloading the Expo Go app [================================================================] 100% 0.0s

› Press ? │ show all commands
Android Bundled 238481ms C:\Users\harsh\OneDrive\Desktop\Lab3\SimpleTodoApp\node_modules\expo-router\entry.js (1252 modules)
 (NOBRIDGE) LOG  Bridgeless mode is enabled
 INFO
 💡 JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (re
crosoft Edge).
iOS Bundled 435ms C:\Users\harsh\OneDrive\Desktop\Lab3\SimpleTodoApp\node_modules\expo-router\entry.js (1 module)
 (NOBRIDGE) LOG  Bridgeless mode is enabled
 INFO
 💡 JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (re
crosoft Edge).
 (NOBRIDGE) WARN  [Reanimated] Reduced motion setting is enabled on this device. This warning is visible only in the development mode. Some animations will be disa
verride the behavior for individual animations, see https://docs.swmansion.com/react-native-reanimated/docs/guides/troubleshooting#reduced-motion-setting-is-enable
```

Figure 14: Running the To-Do List App on Android Emulator



```
PS C:\Users\harsh\OneDrive\Desktop\Lab3\SimpleTodoApp> npx expo start
Starting project at C:\Users\harsh\OneDrive\Desktop\Lab3\SimpleTodoApp
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
  @react-native-async-storage/async-storage@2.1.0 - expected version: 1.23.1
  react-native-screens@4.0.0 - expected version: ~4.1.0
Your project may not work correctly until you install the expected versions of the packages.

› Metro waiting on exp://100.64.14.60:8081
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Web is waiting on http://localhost:8081

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

› Press a │ open Android
› Press w │ open web

› Press j │ open debugger
› Press r │ reload app
```

Figure 15: Running the To-Do List App on Physical Device

2.6 Submission (Total 60 Points)

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

Extending Functionality (60 Points)

- **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.

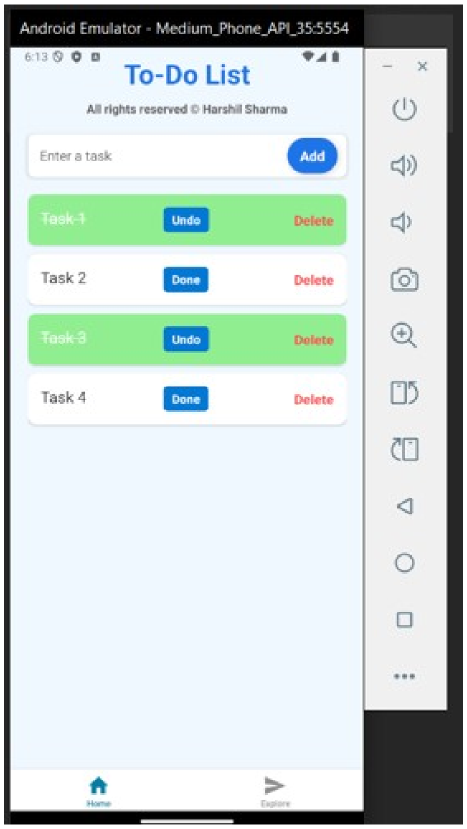


Figure 16: Task Marked as Complete on Emulator

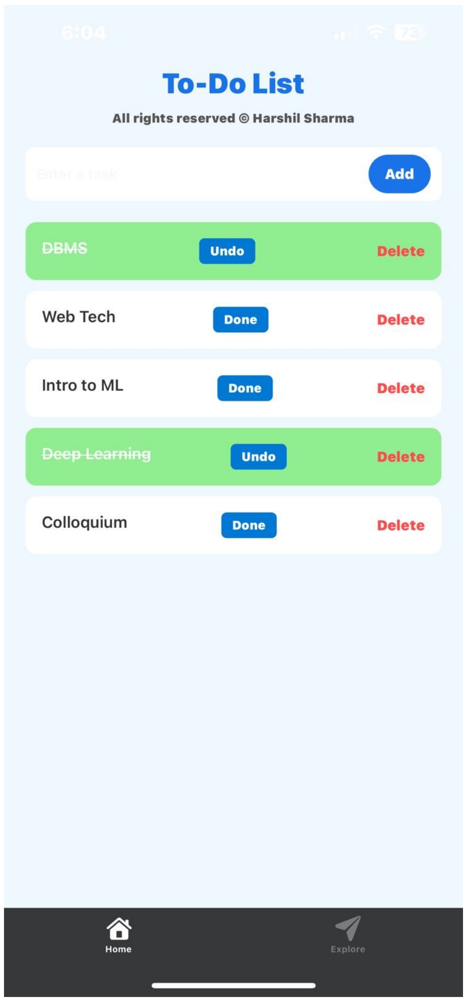
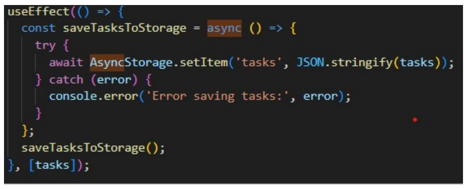


Figure 17: Task Marked as Complete on Physical Device

- Explain how you updated the state to reflect the completion status of tasks.
- In the state array for tasks, I added to each task object a property called ”completed” to keep the state for completion status. This will enable that, upon clicking on the ”mark as complete” button, the fired function is toggleCompletion; finding the task by ID, and then toggling its property ”completed” between true and false. Then, it updates the state through setTasks to ensure the UI will re-render with the correct completion status for each task.

- **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.



```
useEffect(() => {
  const saveTasksToStorage = async () => {
    try {
      await AsyncStorage.setItem('tasks', JSON.stringify(tasks));
    } catch (error) {
      console.error('Error saving tasks:', error);
    }
  };
  saveTasksToStorage();
}, [tasks]);
```

Figure 18: AsyncStorage Code Snippet

- **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.

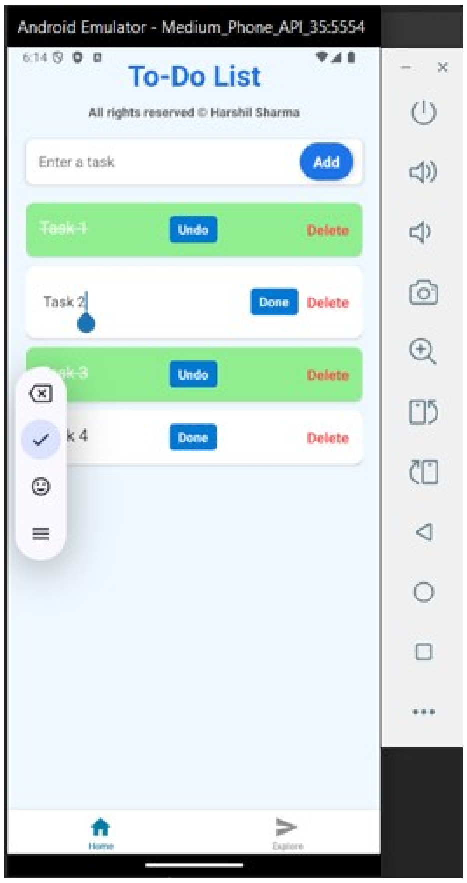


Figure 19: Editing Task in the To-Do List App on Emulator

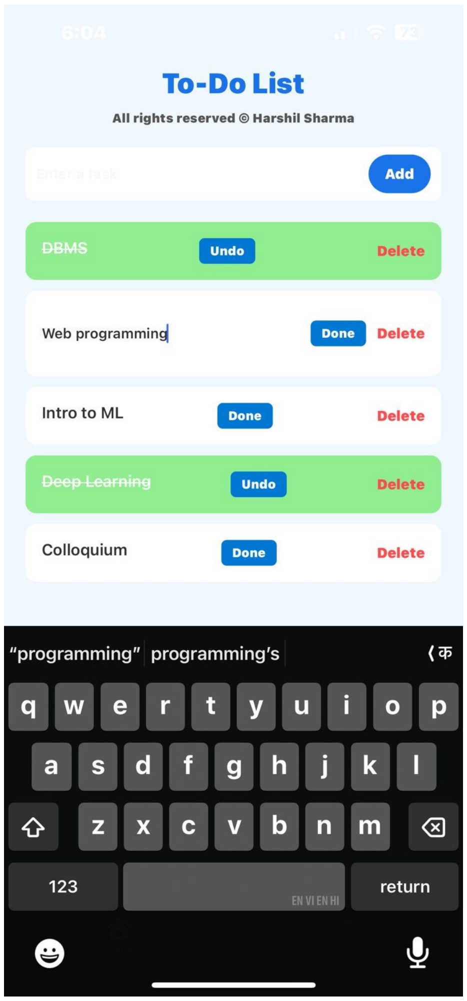
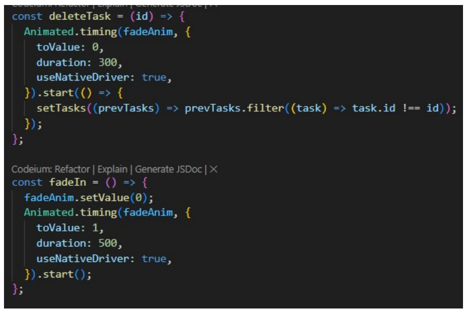


Figure 20: Editing Task in the To-Do List App on Physical Device

- Explain how you managed the UI for editing tasks.
- To manage the UI for editing tasks, I have implemented an edit mode for each task. Upon clicking, it reveals an input field with a current value of a task that can be changed by a user. I used conditional rendering to either show an input box or task text, depending on if the task was in edit mode. Once the user is through with editing and confirms the change, the changed value is written back into the state of the task, and edit mode is turned off. This makes the transition between viewing and editing tasks absolutely seamless.

- **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.
– I have used React Native's Animated API for fade-in and slide-out animation. When a new task gets added, it should fade in-fashion so that the user gets to feel the transition being smooth and natural into the list. Then, when the task is deleted, this is told visually by the slide-out that one item is removed. These animations, in turn, provide improved feedback to the users, make the interactions more engaging, and bestow a fine, polished, modern touch upon the application. This does not only make the app more fascinating to use but also keeps the user better informed about the state changes happening within the app.



```
const deleteTask = (id) => {
  Animated.timing(fadeAnim, {
    toValue: 0,
    duration: 300,
    useNativeDriver: true,
  }).start(() => {
    setTasks((prevTasks) => prevTasks.filter((task) => task.id !== id));
  });
};

Codeium: Refactor | Explain | Generate JSDoc | ×
const fadeIn = () => {
  fadeAnim.setValue(0);
  Animated.timing(fadeAnim, {
    toValue: 1,
    duration: 500,
    useNativeDriver: true,
  }).start();
};
```

Figure 21: Code Snippet for Adding Animations Using Animated API

3 Conclusion

This report provides an overview of setting up a development environment for React Native, configuring an emulator, comparing development options, and building a simple To-Do List app with extended functionality. The setup allows for efficient app development, testing, and deployment on both emulators and physical devices.

4 Acknowledgment of LLM Assistance

During the course of setting up the development environment and fixing certain errors, I utilized a Language Learning Model (LLM) for assistance. Specifically, the LLM provided solutions for resolving JAVA_HOME path verification issues and offered advice on handling the default `App.tsx` compilation problem.