**React Native Environment Setup Documentation**

**1. System Requirements**

**Minimum System Requirements:**

* **CPU**: Intel Core i5 or equivalent
* **RAM**: Minimum 8GB
* **Operating System**: Windows 10 (64-bit) or later, macOS High Sierra or later, or Linux

**System Configuration Used (My Laptop):**

* **CPU**: Intel Core i5
* **RAM**: 16GB
* **Operating System**:  Windows 11 Professional

A screenshot of a computer program

Description automatically generated

Fig.1: Personal PC System Configuration

Source: Primary

**2. Installation Instructions**

Following these steps to install the necessary tools and dependencies:

1. **Node.js**
   * Download the latest Long-Term Support (LTS) version of Node.js from the official website
   * Follow the installation instructions provided for your operating system.
   * Verify the installation by running the following commands in your terminal:

A screenshot of a computer program

Description automatically generated

Fig.2: Personal PC System Configuration

Source: [2]

I’ve already installed Node.js earlier, so there’s no need for reinstallation. The figure below shows the version of both my Node.js and NPM(Node Package Manager) installations.

A screenshot of a computer program

Description automatically generated

Fig.3: Personal PC System Configuration

Source: [Primary]

1. **React Native CLI**:
   * To Install the React Native CLI globally using npm the following command will be inputted at the command line interface as described in the figure below.

A black screen with a black border

Description automatically generated

Fig.4: Installing the React-Native Global CLI

Source: [1]

Moreover, I installed this during class time and do not require a reinstallation. The figure below confirms my installation and installed version.

A screenshot of a computer

Description automatically generated

Fig.5: React-Native CLI version

Source: [Primary]

1. **Java Development Kit (JDK)**:
   * Download and install the Java SE Development Kit (JDK) from Oracle website [3]. I have installed the Java Development Kit in my Java class (Object-Oriented Programming 3), which is a requirement for Java programming. The figure below indicates the JDK version I have installed.

A screenshot of a computer

Description automatically generated

Fig.6: JDK Installed version

Source: [Primary]

1. **Android Studio**:
   * Downloading and installing the Android Studio can be done from the Android Developers website [3]. Once the Installation of the Android Studio is completed, click on “**More Actions**” to proceed with the SDK Manager settings.

A screenshot of a computer

Description automatically generated

Fig.7: Opening Android Studio after Initial Installation

Source: [1]

A screenshot of a computer

Description automatically generated

Fig.8: My Android Studio Welcome Screen

Source: [Primary]

**3. Configuration Steps**

Configure the necessary settings to ensure your environment is ready for React Native development:

1. **Configure Android Studio**:
   * To configure, the Android Studio, open Android Studio and navigate to the gear icon in the toolbar and select the SDK Manager.
   * Alternatively, navigate to **File > Settings > Appearance & Behavior > Languages & Framework > Android SDK**.
   * Under **SDK Tools**, in-class installation, make sure to check **Android SDK Build-tools** , **Android Emulator**, and **Android SDK Platform-Tools**.
   * Ensure that the required SDK platforms and tools are installed.

During the in-class installation, I ensured as instructed by the instructor that all the Android SDK, SDK Platform, and Android Virtual Device (AVD) were selected and installed. Specifically, the **Android 14** (UpsideDownCake) SDK was selected as the preferred option. This can be see and confirmed as shown in Fig. 9 below. Also, for the **SDK Platform-tools**, the following options were selected:

* + 1. Android SDK Platform 34 (API Level 34 i.e the Android OS version)
    2. Intel x86 Atom\_64 System Image

Finally, clicking on “Apply” initiates the download and installation of the Android SDK and related build tools. I have this installed and can seen the figure below. I accessed this from the SDK manager settings in Android Studio.

A screenshot of a computer

Description automatically generated

Fig.9: SDK Manager Settings

Source: [Primary]

1. **Setting Up Environment Variables**:
   * Add the Android SDK to your system's environment variables: This has also been implemented during the in-person class with support from the instructor. Below is the figure showing the configured environment variables and PATH variable already implemented. This was accessed from the **Control Panel > System & Security > Systems > Advance System Settings > Environment Variables**.

A screenshot of a computer

Description automatically generated

Fig.10: Environment Variable for Android Home

Source: [Primary]

A screenshot of a computer program

Description automatically generated

Fig.11: System Variables for PATH

Source: [Primary]

**4. Checking the Android Emulator**

Once Android Studio and SDK were set up, I launched an Android emulator.

1. In Android Studio, I went to **Tools > AVD Manager** and either created a new virtual device or selected an existing one. But I have earlier created a Medium Phone AVD as shown in the figure below (Fig. 12).
2. After configuring the device, I clicked the **Play** button to start the emulator.

Below, I capture a screenshot of the Android Virtual Device (AVD) Manager with the selected emulator starting (Fig. 13).

A blue line on a black background

Description automatically generated

Fig.12: Medium Phone API 34 Android Virtual Device

Source: [Primary]

A screenshot of a computer

Description automatically generated

Fig.13: Starting the Android Virtual Device

Source: [Primary]

**5. Project Creation**

For my React Native project, I decided to use the npx create-expo-app@latest approach as explained during the in-class session to streamline the setup process. Here are the detailed steps I took, alongside the necessary screenshots to illustrate each activity.

1. **Create a new expo App called “IncredibleTodoListApp”**:

I started by opening my command line terminal, then navigating to the parent folder (C:/CPRG303) where I chose to create the new app and then ran the following command as seen the the Fig. 14 below to create a new expo app.

A screenshot of a computer

Description automatically generated

Fig.14: Creating a New Expo App (IncredibleTodoList)

Source: [Primary]

This command set up the project structure and installed the necessary dependencies. The screenshot of my terminal after running the command is seen below (Fig. 15).

A screenshot of a computer program

Description automatically generated

Fig.15: Project App Creation Completed

Source: [Primary]

1. **Navigate to the Project Directory**:

Once the project was created, I displayed the content of the parent folder just to confirm the project folder's existence. Next, I navigated into the project directory as seen in the figure below (Fig. 16). This is where all the project files and configurations are located.

A screenshot of a computer

Description automatically generated

Fig.16: Navigating to Project Directory

Source: [Primary]

**6. Running the Project**

The next step was to run the project on an Android Device Simulator. Here is my step by step approach itemized below:

1. **Start Android Studio using AVD**:
   * I opened Android Studio and navigated to the **AVD Manager** to create a new Android Virtual Device (AVD). I selected the device specifications and configuration which has been created earlier (Medium Phone API 34 Android Virtual Device).
   * After creating the AVD, I started it from the AVD Manager. The emulator booted up and displayed the Android home screen.
   * Screenshot of the Android Emulator running:
2. **Run the Expo App**:
   * Back in the terminal, I made sure I was in the project directory and ran the following command to start the project as seen Fig. 17 below. This command

A black background with white text

Description automatically generated

Fig.17: Navigating to Project Directory

Source: [Primary]

This command opened a new browser window with the Expo Developer Tools. With access to a web interface to opens in a browser, allowing you to manage your app, view logs, and access development options (<http://localhost:8081> ).

A screenshot of a computer

Description automatically generated

Fig.18: Running the Expo Start Command

Source: [Primary]

1. **Run on Android Emulator**:
   * In the Expo Developer Tools, I selected “w” an option to open the app in a web browser as seen in figure below (Fig. 19).

A screenshot of a computer

Description automatically generated

Fig.19: Opens the Expo App on a web browser

Source: [Primary]

* + Also in the Expo Developer Tools, I selected “a” an option to open the app in an Android Studio Emulator (Medium Phone API 34) as seen in figure below (Fig. 20).

**A screenshot of a phone

Description automatically generated**

Fig.20: App **FAILED** to Run on Android Emulator

Source: [Primary]

**7. Troubleshooting**

Address common issues and troubleshooting steps:

1. **Android Emulator Issues**:

I encountered the following issue while trying to run my project on the Android Emulator as seen in the screenshot below (Fig. 21).

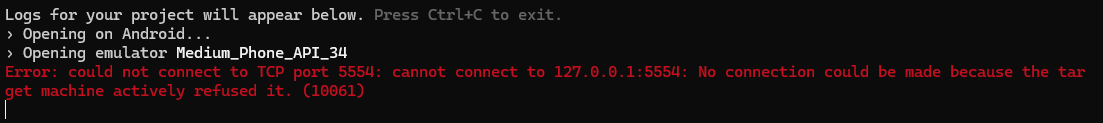


Fig.21: Error message during app run attempt on Android Emulator

Source: [Primary]

1. **Steps Taken to Resolve:**
   1. **Check AVD Configuration**: I ensured that the Android Virtual Device (AVD) was configured correctly [4].
   2. **Restart the AVD:** I restarted the AVD from the AVD Manager to see if it resolved the issue.
   3. **Restart the ADB Server:** I ran the following commands to restart the ADB server [5].
2. **Stop the ADB server:**

A black background with white text

Description automatically generated

Fig.22: Stop the ADB Server

Source: [Primary]

1. **Start the ADB server:**

A black background with white text

Description automatically generated

Fig.23: Staring the ADB Server

Source: [Primary]

1. **Check the connected device to the ADB server:**

A black background with white text

Description automatically generated

Fig.24: Checking for Connected Devices to ADB Server

Source: [Primary]

I noticed no virtual device was connected to the ADB Server, preventing me from launching my app on the Android Emulator during troubleshooting. I had to create a new Virtual Device in Android Studio to resolve this issue.

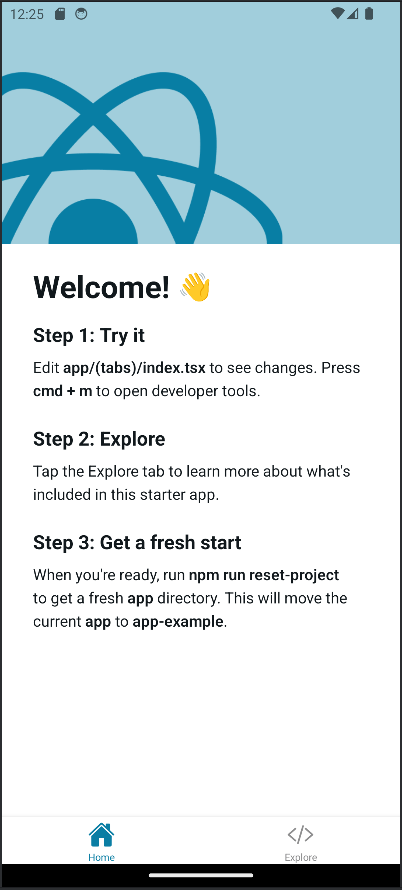
****

Fig.25: App **NOW** Runs on Android Emulator

Source: [Primary]

**References**

[1] “Get Started with React Native · React Native,” Reactnative.dev, Aug. 15, 2024. [Online]. Available: <https://reactnative.dev/docs/environment-setup?guide=native/> . Accessed: Oct. 14, 2024.

[2] Node.js Foundation, “Node.js,” *Node.js*, 2019. [Online]. Available: <https://nodejs.org/en/> . Accessed: Oct. 14, 2024.

[3] Android Studio, “Android Studio and SDK tools,” *Android Developers*, 2019. [Online]. Available: <https://developer.android.com/studio/> Accessed: Oct. 14, 2024.

[4] “Set up Android Emulator networking,” *Android Developers*. [Online]. Available: <https://developer.android.com/studio/run/emulator-networking> . Access: Oct. 14, 2024.

[5] ReactiveCircus, “could not connect to TCP port 5554: Connection refused · Issue #161· ReactiveCircus/android-emulator-runner,” *GitHub*, Jun. 23, 2021. [Online]. Available: <https://github.com/ReactiveCircus/android-emulator-runner/issues/161/> . Accessed Oct. 15, 2024.