🚀 Ultimate Guide: Building a Job Portal App with React Native (Expo) From Scratch to APK Installation for Beginners

🌟 Introduction This guide helps you create a Job Portal app called JobConnector using React Native and Expo. We’ll build a minimal app with a Home page showing "JobConnector Mobile," style it (optionally with Tailwind CSS), and generate an APK for your Android phone. No prior React Native knowledge is needed—just follow these steps!

**Goal:** Build, test, and install a simple job portal app. **Tools Needed:** Computer, internet, Android phone (optional), and enthusiasm! 😊 **Current Date:** February 26, 2025.

🛠️ Step 1: Set Up Your Environment Prepare your computer for development.

### Install Node.js

* Download from [nodejs.org](https://nodejs.org/) and install.
* Verify in CMD or PowerShell:
* node -v

**Why?** Powers Expo and React Native.

### Install Expo CLI and EAS CLI

Run in CMD or PowerShell (global terminal):

npm install -g expo-cli eas-cli

**Why?** expo-cli creates projects; eas-cli builds APKs.

### Optional: Install Git

Get from [git-scm.com](https://git-scm.com/) for version control.

### Optional: Install ADB

Download Android SDK tools from [developer.android.com](https://developer.android.com/). **Why?** Installs APKs via USB.

👤 Step 2: Create the Job Portal Project Let’s start building!

### Navigate to Workspace

Run in CMD or PowerShell:

cd O:\JobConnector

### Remove Old Project (if exists)

rmdir /s /q mobileapp

**Why?** Ensures a fresh start.

### Create Project Folder

mkdir mobileapp

cd mobileapp

### Initialize Expo Project

npx create-expo-app . --template blank

Choose "blank" when prompted. **Why?** Sets up a minimal Expo project.

**Note:** npx create-react-app@latest . is for web apps, not Expo. Use Expo’s command instead.

🎨 Step 3: Optional - Add Tailwind CSS Style your app easily (if Tailwind doesn’t work, we’ll troubleshoot).

### Install Tailwind CSS

Run inside mobileapp:

npm install -D tailwindcss@3 postcss autoprefixer

### Initialize Tailwind

npx tailwindcss init -p

**Why?** Creates tailwind.config.js and postcss.config.js.

### Configure Tailwind

Edit tailwind.config.js:

module.exports = {

content: ["./App.js", "./pages/\*\*/\*.{js,jsx}"],

theme: { extend: {} },

plugins: [],

};

**Why?** Tells Tailwind where to apply styles.

**Note:** Tailwind CSS isn’t native to Expo. Use nativewind if it doesn’t work (see troubleshooting).

📚 Step 4: Understand Project Structure Inside O:\JobConnector\mobileapp:

* **assets/**: Stores icon.png, splash.png, adaptive-icon.png.
* **App.js**: Entry point, links to Home.js.
* **package.json**: Lists dependencies (Expo, React).
* **app.json**: Configures app name, icon, etc.
* **node\_modules/**: Auto-managed libraries.
* **eas.json (added later)**: Configures APK builds.
* **pages/** (you’ll create): Holds screens like Home.js.
* **tailwind.config.js (optional)**: Tailwind settings.

🛠️ Step 5: Clean and Verify Setup Ensure everything’s ready.

### Uninstall Unused Library

npm uninstall @types/react-native

**Why?** Avoids Expo conflicts.

### Clean Dependencies

Remove-Item -Recurse -Force node\_modules

Remove-Item -Force package-lock.json

npm install

**Why?** Refreshes dependencies.

### Check Expo and EAS Versions

expo --version

eas --version

**Why?** Confirms tools are installed.

### Run Expo Doctor

npx expo-doctor

**Why?** Checks for setup issues.

💼 Step 6: Build the App

### Create Pages Folder

mkdir pages

### Add Home.js

Create Home.js inside pages/:

import React from 'react';

import { View, Text } from 'react-native';

const Home = () => {

return (

<View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

<Text>JobConnector Mobile</Text>

</View>

);

};

export default Home;

### Update App.js

Edit App.js:

import React from 'react';

import { View } from 'react-native';

import Home from './pages/Home';

export default function App() {

return (

<View style={{ flex: 1 }}>

<Home />

</View>

);

}

📱 Step 7: Test Locally

### Start with Expo Go

npx expo start

Scan QR code with Expo Go app on your phone.

### Clear Cache and Start

expo r -c

### Test Production Mode

npx expo start --no-dev --minify

🔧 Step 8: Build the APK

### Log In to Expo

eas login

### Initialize EAS

eas project:init

### Build APK

eas build --platform android --profile preview

### eas.json Configuration

{

"cli": {

"version": ">= 15.0.12",

"appVersionSource": "remote"

},

"build": {

"development": {

"developmentClient": true,

"distribution": "internal"

},

"preview": {

"android": {

"buildType": "apk"

}

},

"production": {

"autoIncrement": true

}

},

"submit": {

"production": {}

}

}

🛠️ Step 9: Install the APK

### Via ADB

adb install path\to\your\apk.apk

### Manual Install

Transfer APK to phone, open it, and allow "Unknown sources."

🛠️ Step 10: Troubleshoot APK Issues

### Full Reset and Rebuild

npm uninstall @types/react-native

Remove-Item -Recurse -Force node\_modules

Remove-Item -Force package-lock.json

npm install

npx expo-doctor

eas build --platform android --profile preview

adb install path\to\your\apk.apk

adb logcat \*:E

Your **JobConnector** app is now ready! 🚀

## 🛠️ **Git Workflow for Old & New Users**

## **🔹 First Phase: For Old Git Users** (Already Using Git)

If you **already have Git installed and set up**, follow these steps:

### ✅ ****1. Navigate to Your Project Folder****

cd path/to/your/project

👉 Move into your project directory.

### ✅ ****2. Check Git Status****

git status

👉 See which files have changed.

### ✅ ****3. Add Changes to Staging****

git add .

👉 Adds all modified and new files to be committed.

### ✅ ****4. Commit the Changes****

git commit -m "Your commit message here"

👉 Saves your changes locally with a message.

### ✅ ****5. Push to GitHub****

git push origin main

👉 Uploads your changes to the main branch of the repository.

💡 **If you are using master instead of main:**

git push origin master

## **🔹 Second Phase: For New Git Users** (First-Time Git Setup)

If you are **new to Git** and haven't set it up before, follow these steps:

### ✅ ****1. Install Git (If Not Installed)****

👉 **Windows:** Download and install Git from [git-scm.com](https://git-scm.com/downloads).  
👉 **Mac/Linux:** Install using terminal:

sudo apt install git # For Ubuntu/Debian

brew install git # For MacOS

### ✅ ****2. Check Git Installation****

git --version

👉 Ensures Git is installed and shows the version.

### ✅ ****3. Configure Git for First-Time Use****

git config --global user.name "Your Name"

git config --global user.email "your-email@example.com"

👉 Saves your identity for commits.

### ✅ ****4. Navigate to Your Project Folder****

cd path/to/your/project

### ✅ ****5. Initialize Git in the Project****

git init

👉 Creates a new Git repository in your project folder.

### ✅ ****6. Connect to GitHub Repository****

git remote add origin https://github.com/yourusername/YourRepo.git

👉 Links your local project to GitHub.

### ✅ ****7. Add Files to Git****

git add .

👉 Adds all files to be committed.

### ✅ ****8. Commit the Files****

git commit -m "Initial commit"

👉 Saves your changes with an initial commit message.

### ✅ ****9. Push to GitHub****

git push -u origin main

👉 Uploads the project to the main branch.

## **📌 Summary of Commands: Old vs. New Users**

| **Action** | **Old User Commands** | **New User Commands** |
| --- | --- | --- |
| Check Git installation | git --version | git --version |
| Navigate to project | cd project-folder | cd project-folder |
| Check file status | git status | git status |
| Add files | git add . | git add . |
| Commit changes | git commit -m "message" | git commit -m "message" |
| Push to GitHub | git push origin main | git push -u origin main |
| Clone repository | git clone URL | git clone URL |
| Set up Git (first time) | Already configured | git config --global user.name "Your Name"git config --global user.email "your-email@example.com" |
| Initialize Git | Already initialized | git init |
| Connect to GitHub | Already connected | git remote add origin URL |

Now, whether you're an **old user** or **new user**, you have the complete Git workflow to manage your projects efficiently! 🚀

# 🚀 ****Ultimate Guide: Building a Job Portal App with React Native (Expo)****

### From Scratch to APK Installation for Beginners

## 🌟 **Introduction**

This guide helps you create a **Job Portal app** called **JobConnector** using React Native and Expo. We’ll build a minimal app with a **Home page** showing "JobConnector Mobile," style it (optionally with Tailwind CSS), and generate an APK for your Android phone. No prior React Native knowledge is needed—just follow these steps!

* **Goal**: Build, test, and install a simple job portal app.
* **Tools Needed**: Computer, internet, Android phone (optional), and enthusiasm! 😊
* **Current Date**: February 26, 2025.

## 🛠️ **Step 1: Set Up Your Environment**

Prepare your computer for development.

1. **Install Node.js**
   * Download from [nodejs.org](https://nodejs.org/) and install.
   * Verify in **CMD** or **PowerShell**:

-----

Commands

node -v

* + - **Why?** Powers Expo and React Native.

1. **Install Expo CLI and EAS CLI**
   * In **CMD** or **PowerShell** (global terminal):

-----

Commands

npm install -g expo-cli eas-cli

* + - **Why?** expo-cli creates projects; eas-cli builds APKs.

1. **Optional: Install Git**
   * Get from [git-scm.com](https://git-scm.com/) for version control.
2. **Optional: Install ADB**
   * Download Android SDK tools from [developer.android.com](https://developer.android.com/).
   * **Why?** Installs APKs via USB.

## 📂 **Step 2: Create the Job Portal Project**

Let’s start building!

1. **Navigate to Workspace**
   * In **CMD** or **PowerShell**:

-----

Commands

cd O:\JobConnector

1. **Remove Old Project (if exists)**
   * Run:

-----

Commands

rmdir /s /q mobileapp

* + - **Why?** Ensures a fresh start.

1. **Create Project Folder**
   * Run:

-----

Commands

mkdir mobileapp

cd mobileapp

1. **Initialize Expo Project**
   * Run:

-----

Commands

npx create-expo-app . --template blank

* + - Choose **"blank"** when prompted.
    - **Why?** Sets up a minimal Expo project.
  + **Note**: npx create-react-app@latest . is for web apps, not Expo. Use Expo’s command instead.

## 🎨 **Step 3: Optional - Add Tailwind CSS**

Style your app easily (if Tailwind doesn’t work, we’ll troubleshoot).

1. **Install Tailwind CSS**
   * In **project terminal** (inside mobileapp):

-----

Commands

npm install -D tailwindcss@3 postcss autoprefixer

1. **Initialize Tailwind**
   * Run:

-----

Commands

npx tailwindcss init -p

* + - **Why?** Creates tailwind.config.js and postcss.config.js.

1. **Configure Tailwind**
   * Edit tailwind.config.js:
     + Add paths to your files (e.g., ./App.js, ./pages/\*\*/\*.{js,jsx}).
   * **Why?** Tells Tailwind where to apply styles.
   * **Note**: Tailwind CSS isn’t native to Expo. Use libraries like nativewind if it doesn’t work (see troubleshooting).

## 🗂️ **Step 4: Understand Project Structure**

What’s inside O:\JobConnector\mobileapp?

* 📁 **assets/**: Stores icon.png, splash.png, adaptive-icon.png.
* 📄 **App.js**: Entry point, links to Home.js.
* 📄 **package.json**: Lists dependencies (Expo, React).
* 📄 **package-lock.json**: Locks dependency versions.
* 📄 **app.json**: Configures app name, icon, etc.
* 📁 **node\_modules/**: Auto-managed libraries.
* 📄 **eas.json** (added later): Configures APK builds.
* 📁 **pages/** (you’ll create): Holds screens like Home.js.
* 📄 **tailwind.config.js** (optional): Tailwind settings.
* 📄 **postcss.config.js** (optional): PostCSS settings.

## ⚙️ **Step 5: Clean and Verify Setup**

Ensure everything’s ready.

1. **Uninstall Unused Library**
   * Run:

-----

Commands

npm uninstall @types/react-native

* + - **Why?** Avoids Expo conflicts.

1. **Clean Dependencies**
   * Run:

-----

Commands

Remove-Item -Recurse -Force node\_modules

Remove-Item -Force package-lock.json

npm install

* + - **Why?** Refreshes dependencies.

1. **Check Expo and EAS Versions**
   * Run:

-----

Commands

expo --version

eas --version

* + - **Why?** Confirms tools are installed (e.g., Expo ~52.0.0, EAS 5.x.x).
    - If missing, reinstall:

-----

Commands

npm install -g eas-cli

1. **Run Expo Doctor**
   * Run:

-----

Commands

npx expo-doctor

* + - **Why?** Checks for setup issues.

## 📝 **Step 6: Build the App**

Create the app’s structure.

1. **Create Pages Folder**
   * Run:

-----

Commands

mkdir pages

1. **Add Home.js**
   * In pages/, create Home.js to show "JobConnector Mobile."
   * **Purpose**: Your app’s welcome screen.
2. **Update App.js**
   * Edit App.js to link to Home.js.
   * **Purpose**: Displays the Home page.
3. **Routing (Future)**
   * Add more pages (e.g., Jobs.js) and use **Expo Router** later.

## 🖼️ **Step 7: Add Assets**

Give your app a look!

1. **Create Assets Folder**
   * Run:

-----

Commands

mkdir assets

1. **Add Images**
   * Place in assets/:
     + icon.png (512x512)
     + splash.png (1242x2436)
     + adaptive-icon.png (432x432)
   * **Why?** Visuals for app icon and loading screen.

## ✅ **Step 8: Test Locally**

See your app in action!

1. **Start with Expo Go**
   * Run:

-----

Commands

npx expo start

* + - Scan QR code with **Expo Go** app on your phone.
    - **Expected**: "JobConnector Mobile" shows.

1. **Clear Cache and Start**
   * Run:

-----

Commands

expo r -c

* + - **Why?** Refreshes the app.

1. **Test Production Mode**
   * Run:

-----

Commands

npx expo start --no-dev --minify

* + - **Why?** Simulates built app behavior.

## 🛠️ **Step 9: Build the APK**

Create an installable file!

1. **Log In to Expo**
   * Run:

-----

Commands

eas login

* + - **Why?** Links to Expo servers.

1. **Initialize EAS**
   * Run:

-----

Commands

eas project:init

* + - **Why?** Sets up EAS.

1. **Configure EAS Build**
   * Run:

-----

Commands

eas build:configure

* + - **Why?** Prepares build settings (creates eas.json if needed).

1. **Build APK (Preview)**
   * Run:

-----

Commands

eas build --platform android --profile preview

* + - **Why?** Generates an APK for testing.

1. **Build APK (Production)**
   * Run:

-----

Commands

eas build -p android --profile production

* + - **Why?** Creates a production-ready APK or AAB.

1. **Download APK**
   * Get the link from [expo.dev](https://expo.dev).

## 📱 **Step 10: Install the APK**

Put it on your phone!

1. **Via ADB**
   * Run:

-----

Commands

adb install path\to\your\apk.apk

* + - **Why?** Installs via USB.

1. **Manual Install**
   * Transfer APK to phone, open it, and allow "Unknown sources."

## ⚠️ **Step 11: Troubleshoot APK Not Working**

If the APK fails, follow these steps:

1. **Full Reset and Rebuild**
   * Run in project terminal:

-----

Commands

cd O:\JobConnector\mobileapp

npm uninstall @types/react-native

Remove-Item -Recurse -Force node\_modules

Remove-Item -Force package-lock.json

npm install

npx expo-doctor

eas build --platform android --profile preview

eas init

expo r -c

Remove-Item -Recurse -Force node\_modules

Remove-Item -Force package-lock.json

npm install

eas project:init

eas build --platform android --profile preview

adb install path\to\your\apk.apk

adb logcat \*:E

* + - **Why?** Resets everything and rebuilds.

1. **Check Files**
   * **App.js**: Ensure it’s simple:

javascript

Commands

import React from 'react';

import { View, ----- } from 'react-native';

import Home from './pages/Home';

export default function App() {

return (

<View style={{ flex: 1 }}>

<Home />

</View>

);

}

* + - **Replace With**: If crashing, use:

javascript

Commands

export default function App() {

return (

<View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

<----->Hello, JobConnector!</----->

</View>

);

}

* + - **Why?** Simplifies to debug.
  + **pages/Home.js**: Check:

javascript

Commands

import React from 'react';

import { View, ----- } from 'react-native';

const Home = () => {

return (

<View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

<----->JobConnector Mobile</----->

</View>

);

};

export default Home;

* + - **Why?** Ensures basic rendering.

1. **Tailwind Not Working**
   * Install nativewind:

-----

Commands

npm install nativewind

npm install -D tailwindcss

npx tailwindcss init

* + Update tailwind.config.js:

javascript

Commands

module.exports = {

content: ["./App.js", "./pages/\*\*/\*.{js,jsx}"],

theme: { extend: {} },

plugins: [],

};

* + **Why?** Tailwind needs nativewind for React Native compatibility.

## 📋 **Step 12: Full Command List**

* **Global Terminal**:
  + npm install -g expo-cli eas-cli: Installs tools.
  + cd O:\JobConnector: Navigates workspace.
  + rmdir /s /q mobileapp: Removes old project.
  + mkdir mobileapp: Creates folder.
  + cd mobileapp: Enters folder.
  + npx create-expo-app . --template blank: Starts project.
* **Project Terminal**:
  + npm install -D tailwindcss@3 postcss autoprefixer: Adds Tailwind.
  + npx tailwindcss init -p: Initializes Tailwind.
  + npm uninstall @types/react-native: Removes conflicts.
  + Remove-Item -Recurse -Force node\_modules: Clears dependencies.
  + Remove-Item -Force package-lock.json: Clears lock file.
  + npm install: Reinstalls dependencies.
  + npx expo-doctor: Checks setup.
  + expo --version: Checks Expo version.
  + eas --version: Checks EAS version.
  + npx expo start: Starts app.
  + expo r -c: Clears cache and starts.
  + npx expo start --no-dev --minify: Tests production mode.
  + eas build:configure: Configures EAS.
  + eas build --platform android --profile preview: Builds preview APK.
  + eas build -p android --profile production: Builds production APK.
  + eas project:init: Links to EAS.
  + adb install path\to\your\apk.apk: Installs APK.
  + adb logcat \*:E: Shows logs.

## 🗃️ **Step 13: Files and Roles**

* **App.js**: Links to Home.js.
* **pages/Home.js**: Displays "JobConnector Mobile."
* **app.json**: Sets name, icon, package.
* **eas.json**: Configures builds.
* **package.json**: Lists tools.
* **package-lock.json**: Locks versions.
* **assets/**: Holds images.
* **tailwind.config.js**: Tailwind settings (optional).