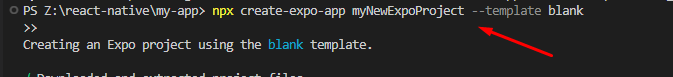
How to Create React Native App with Expo Go

* npm install -g expo-cli
* Run this command in terminal npx create-expo-app myNewExpoProject --template blank



* cd into project folder and run npx expo start



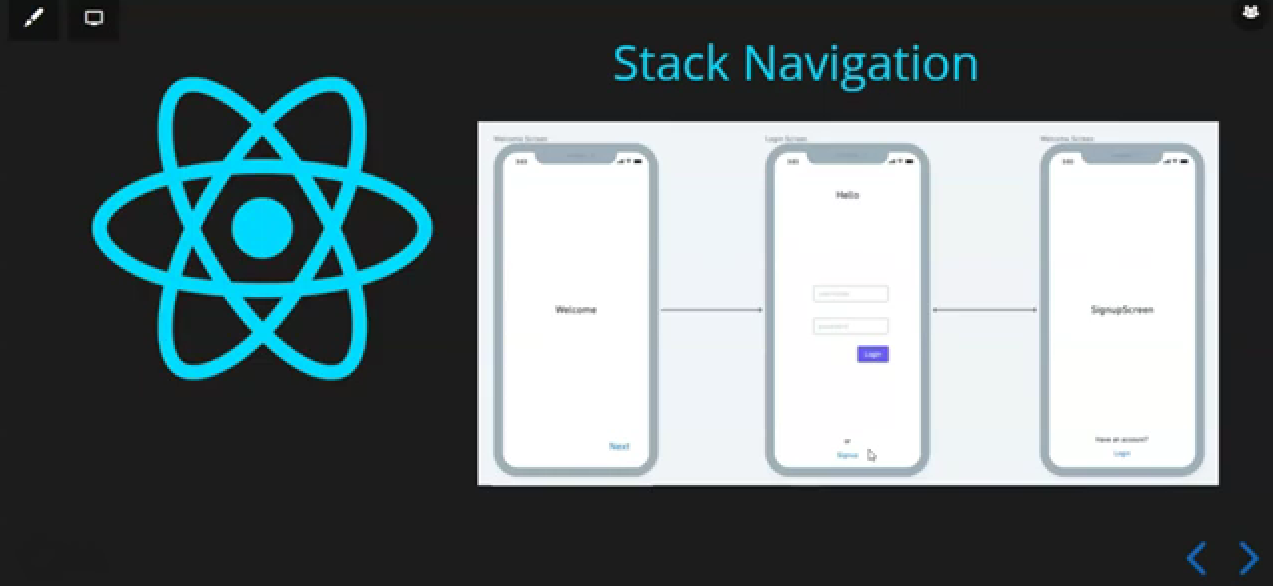
Navigation

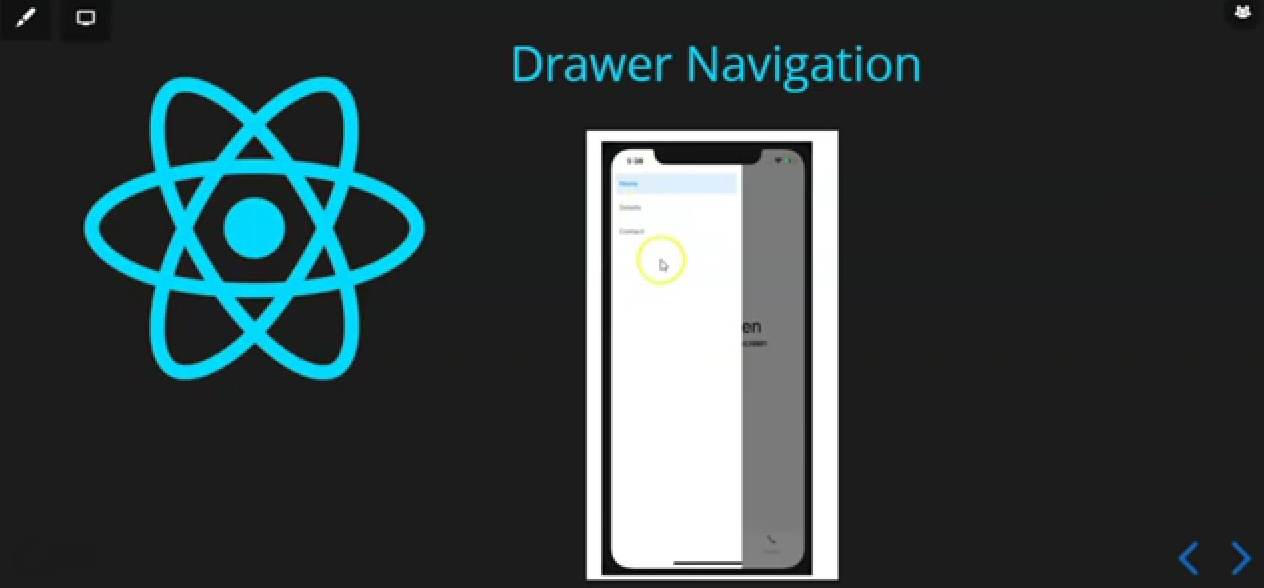
stack

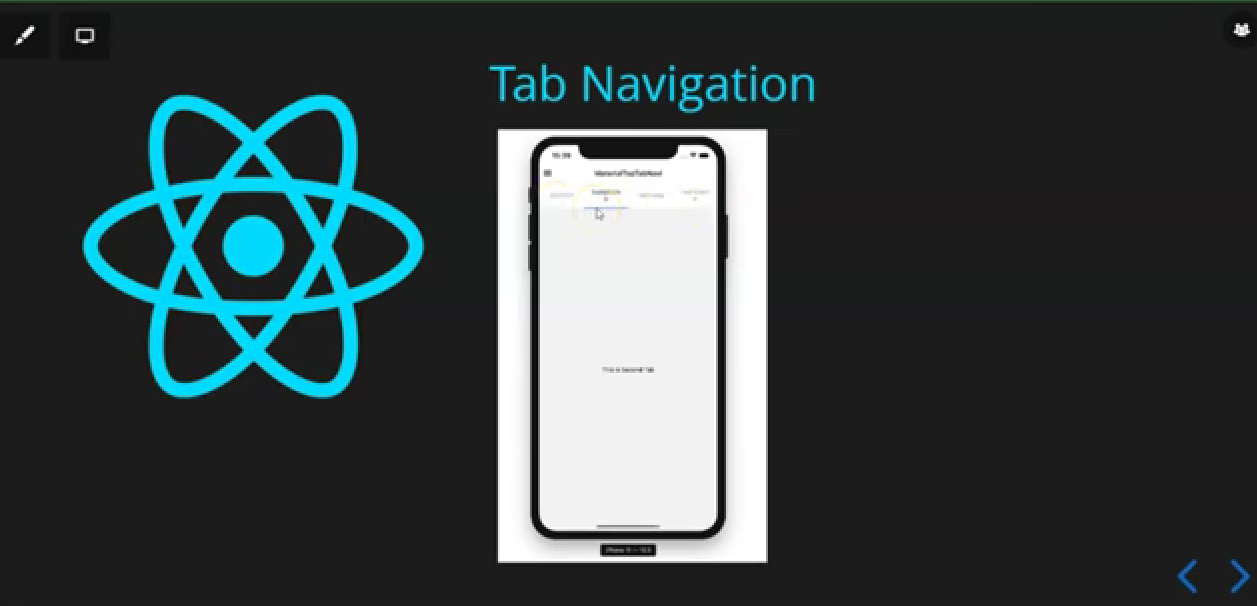
drawer

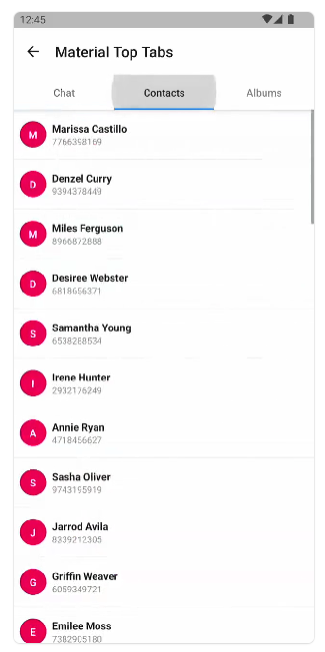
tab

top tabs









npm start --clean-cache

Json-Server Setup And Introduction

What is JSON Server?

JSON-Server is an npm(Node Package Manager) module that allows you to create a mock REST API using just a JSON file. It is highly useful for prototyping, testing, or building front-end applications without needing a complex back-end infrastructure. Data is transferred in JSON(JavaScript Object Notation) format between client and server using HTTP methods like GET, POST, PUT, PATCH, and DELETE.

## How to Set Up JSON-Server

### Step 1: Install JSON-Server

To get started with JSON-Server, install it globally using NPM. Open your terminal and run the following command:

npm install -g json-server

This installs JSON-Server globally, meaning you can access it from anywhere on your system.

### Step 2: Create a JSON Data File

Next, you need to create a JSON file that will act as your database. For example, you can create a file named db.json and structure it as follows:

{

"posts": [

{

"id": 1,

"title": "JSON-Server",

"author": "Amit"

},

{

"id": 2,

"title": "Node.js",

"author": "Mohit"

}

],

"comments": [

{

"id": 1,

"body": "Great post!",

"postId": 1

},

{

"id": 2,

"body": "Informative!",

"postId": 2

}

],

"profile": {

"name": "Amit Kumar"

}

}

**This JSON structure defines three endpoints:**

* **/posts:** to handle posts-related data.
* **/comments:** to handle comments related to posts.
* **/profile:** to represent user profile information.

### Step 3: Start JSON-Server

Once you have created your db.json file, you can start the JSON-Server with a simple command:

json-server --watch db.json

This command will start the server and watch for changes in the **db.json** file. By default, the server will be hosted at **http://localhost:3000**, and you can start making HTTP requests to the endpoints defined in the JSON file.

## Example: Testing Endpoints

Now that the server is running, you can test the following endpoints using an API client like Postman, or even directly from your browser:

### 1. GET all posts:

http://localhost:3000/posts

This will return all the posts from the db.json file.

### 2. GET a single post by ID:

http://localhost:3000/posts/1

This returns the post with an ID of 1.

### 3. POST a new post:

You can create a new post by sending a POST request to:

http://localhost:3000/posts

With the following JSON body:

{

"id": 3,

"title": "New Post",

"author": "Rohit"

}

### 4. PUT (Update a post):

Update an existing post with:

http://localhost:3000/posts/1

Along with a new body, such as:

{

"id": 1,

"title": "Updated Post",

"author": "Amit"

}

### 5. DELETE a post:

To delete a post:

http://localhost:3000/posts/1

**For Drawer**

npm install @react-navigation/native @react-navigation/drawer react-native-screens react-native-safe-area-context react-native-gesture-handler react-native-reanimated react-native-vector-icons

import React from "react";

import { NavigationContainer } from "@react-navigation/native";

import { createDrawerNavigator } from "@react-navigation/drawer";

import HomeScreen from "./Components/Home";

import AboutScreen from "./Components/About";

import ApiDataScreen from "./Components/ApiData";

import FontAwesome5 from 'react-native-vector-icons/FontAwesome5';

const App = () => {

const Drawer = createDrawerNavigator();

return (

<NavigationContainer>

<Drawer.Navigator

*initialRouteName*="Home"

*drawerPosition*="right"

*drawerType*="front"

*edgeWidth*={100}

*hideStatusBar*={false}

*overlayColor*="#000"

*drawerStyle*={{

backgroundColor: "#fff",

width: 200,

}}

*screenOptions*={{

headerShown: true,

swipeEnabled: true,

gestureEnabled: true,

headerTitleAlign: "center",

headerStyle: { backgroundColor: "#fff" },

headerTitleStyle: { fontSize: 25 },

}}

>

<Drawer.Screen

*name*="Home"

*component*={HomeScreen}

*options*={{

title: "My Home",

}}

/>

<Drawer.Screen

*name*="About"

*component*={AboutScreen}

*options*={{

title: "My About",

drawerIcon: ({ *focused* }) => (

<FontAwesome5

*name*="info-circle"

*size*={*focused* ? 25 : 20}

*color*={*focused* ? "skyblue" : "#000"}

/>

),

}}

/>

<Drawer.Screen

*name*="ApiData"

*component*={ApiDataScreen}

*options*={{

title: "API Data",

}}

/>

</Drawer.Navigator>

</NavigationContainer>

);

};

export default App;

### If search param not working then

npm uninstall -g json-server

npm i -g json-server@0.17.4

json-server --host 0.0.0.0 db.json

### App Icons

<https://easyappicon.com/>

<https://www.appicon.co/>

<https://docs.expo.dev/guides/progressive-web-apps/#favicons>

### Splash Screen

<https://docs.expo.dev/versions/latest/sdk/splash-screen/>

### Build

<https://docs.expo.dev/build/setup/>

<https://docs.expo.dev/build-reference/apk/>

### AsyncStorage

<https://docs.expo.dev/versions/latest/sdk/async-storage/>

<https://react-native-async-storage.github.io/async-storage/docs/usage/>

### Fonts

<https://docs.expo.dev/develop/user-interface/fonts/>

### react-native-expo-viewport-units

<https://www.npmjs.com/package/react-native-expo-viewport-units>

npm i react-native-expo-viewport-units

**import** { vw, vh, vmin, vmax } **from** 'react-native-expo-viewport-units';

<View style={{ width: vw(100), height: vh(100) }}>

...

<View>

var styles = StyleSheet.create({

Container: {

width: vmin(95),

height: vmax(70),

padding: vw(2.5),

margin: vh(5),

},

Font: {

fontSize: vw(3.75),

},

});

import { vw, vh, vmin, vmax } from "react-native-expo-viewport-units";

import { StyleSheet } from "react-native";

const ThemeStyle = StyleSheet.create({

WelcomeScreenBanner: {

width: vw(100),

height: vh(50),

},

});

export default ThemeStyle;

### Install for web

**npx expo install react-dom react-native-web @expo/metro-runtime**

### SVG

<https://www.npmjs.com/package/react-native-svg>

### React Native HTML

<https://www.npmjs.com/package/react-native-render-html>

### 

### Different colors on top and bottom of safeareaview

<https://stackoverflow.com/questions/65011812/how-to-have-different-colors-on-top-and-bottom-of-safeareaview-with-iphone-x>

### Copy to Clipboard

<https://docs.expo.dev/versions/latest/sdk/clipboard/>

### Map

<https://www.npmjs.com/package/react-native-maps>

<https://docs.expo.dev/versions/latest/sdk/map-view/>

### Prevent back button after login logout

For Login Screen 👇

useEffect(() => {

const checkToken = async () => {

const token = await AsyncStorage.getItem("token");

if (token) {

// Navigate to Dashboard

navigation.reset({

index: 0,

routes: [{ name: "HomeScreen" }], // Redirect to HomeScreen

});

}

};

checkToken(); // Call the function to check token

}, [navigation]); // Re-run the effect if navigation changes

Ye Signout ke liye 👇

useEffect(() => {

const handleSignout = async () => {

await AsyncStorage.clear(); // Clear all stored data

navigation.reset({

index: 0,

routes: [{ name: "Login" }], // Redirect to Login screen

});

};

handleSignout();

}, [navigation]);

### Screen Orientation

<https://docs.expo.dev/versions/latest/sdk/screen-orientation/>

### React native toast

<https://www.npmjs.com/package/react-native-simple-toast>

npm i react-native-simple-toast