

**Convert Angular to React Next.js Project**

.gitignore

GITIGNORE 582B

angular.json

JSON 2.88KB

README.md

MD 1.44KB

tsconfig.app.json

JSON 424B

tsconfig.json

JSON 915B

tsconfig.spec.json

JSON 434B

change into react nextjs

To convert the Angular project into a React Next.js project, you'll need to make several changes to the configuration files and project structure. Below is a step-by-step guide to help you transition from Angular to React Next.js:

**1. Update .gitignore**

The .gitignore file for a Next.js project will be similar but may include some additional entries specific to Next.js. Here’s an updated version:

plaintext

Copy

# See https://docs.github.com/get-started/getting-started-with-git/ignoring-files for more about ignoring files.

# Compiled output

/dist

/tmp

/out-tsc

/bazel-out

/.next

/out

# Node

/node\_modules

npm-debug.log

yarn-error.log

# IDEs and editors

.idea/

.project

.classpath

.c9/

\*.launch

.settings/

\*.sublime-workspace

# Visual Studio Code

.vscode/

.vscode/settings.json

.vscode/tasks.json

.vscode/launch.json

.vscode/extensions.json

.history/\*

# Miscellaneous

/.angular/cache

.sass-cache/

/connect.lock

/coverage

/libpeerconnection.log

testem.log

/typings

# System files

.DS\_Store

Thumbs.db

**2. Create next.config.js**

Next.js uses a next.config.js file for configuration. Create this file in the root of your project:

javascript

Copy

/\*\* @type {import('next').NextConfig} \*/

const nextConfig = {

reactStrictMode: true,

swcMinify: true,

images: {

domains: ['example.com'], // Add any domains you need for images

},

};

module.exports = nextConfig;

**3. Update package.json**

Update the package.json to include Next.js dependencies and scripts:

json

Copy

{

"name": "dashboard-nextjs",

"version": "1.0.0",

"scripts": {

"dev": "next dev",

"build": "next build",

"start": "next start",

"lint": "next lint"

},

"dependencies": {

"next": "^14.0.0",

"react": "^18.2.0",

"react-dom": "^18.2.0"

},

"devDependencies": {

"eslint": "^8.0.0",

"eslint-config-next": "^14.0.0",

"typescript": "^5.0.0",

"@types/react": "^18.2.0",

"@types/node": "^20.0.0"

}

}

**4. Create tsconfig.json for Next.js**

Next.js uses TypeScript, so you’ll need to update the tsconfig.json:

json

Copy

{

"compilerOptions": {

"target": "es5",

"lib": ["dom", "dom.iterable", "esnext"],

"allowJs": true,

"skipLibCheck": true,

"strict": true,

"forceConsistentCasingInFileNames": true,

"noEmit": true,

"esModuleInterop": true,

"module": "esnext",

"moduleResolution": "node",

"resolveJsonModule": true,

"isolatedModules": true,

"jsx": "preserve",

"incremental": true

},

"include": ["\*\*/\*.ts", "\*\*/\*.tsx"],

"exclude": ["node\_modules"]

}

**5. Create pages Directory**

Next.js uses a pages directory for routing. Create this directory and add an index.tsx file:

bash

Copy

mkdir pages

touch pages/index.tsx

Add the following content to pages/index.tsx:

tsx

Copy

import React from 'react';

const HomePage: React.FC = () => {

return (

<div>

<h1>Welcome to Next.js!</h1>

</div>

);

};

export default HomePage;

**6. Remove Angular-Specific Files**

Remove Angular-specific files such as angular.json, tsconfig.app.json, and tsconfig.spec.json. These files are no longer needed in a Next.js project.

**7. Update README.md**

Update the README.md to reflect the new project structure and commands:

markdown

Copy

# DashboardNextJs

This project was generated using [Next.js](https://nextjs.org/) version 14.0.0.

## Development server

To start a local development server, run:

```bash

npm run dev

Once the server is running, open your browser and navigate to http://localhost:3000/. The application will automatically reload whenever you modify any of the source files.

**Building**

To build the project run:

bash

Copy

npm run build

This will compile your project and store the build artifacts in the .next/ directory.

**Running the production server**

To start the production server, run:

bash

Copy

npm run start

**Additional Resources**

For more information on using Next.js, including detailed command references, visit the [Next.js Documentation](https://nextjs.org/docs) page.

Copy

### 8. Install Dependencies

Finally, install the dependencies:

```bash

npm install

**9. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/.

**Summary**

* Updated .gitignore to include Next.js specific files.
* Created next.config.js for Next.js configuration.
* Updated package.json with Next.js dependencies and scripts.
* Updated tsconfig.json for Next.js TypeScript configuration.
* Created a pages directory with an index.tsx file for the home page.
* Removed Angular-specific configuration files.
* Updated README.md to reflect the new project structure.

This should give you a basic Next.js project structure. You can now start building your React components and pages using Next.js conventions.

tsconfig.app.json

JSON 424B

tsconfig.json

JSON 915B

tsconfig.spec.json

JSON 434B

convert the Angular project into a React Next.js

To convert the Angular project into a React Next.js project, you need to make significant changes to the project structure and configuration files. Below is a step-by-step guide to help you transition from Angular to React Next.js:

**1. Remove Angular-Specific Files**

First, remove Angular-specific files that are no longer needed in a Next.js project:

* angular.json
* tsconfig.app.json
* tsconfig.spec.json
* src/main.ts (Angular entry point)
* src/styles.scss (Angular global styles)
* Any Angular-specific components, services, and modules.

**2. Update tsconfig.json**

Next.js uses TypeScript, so you’ll need to update the tsconfig.json file to reflect Next.js conventions. Replace the content of tsconfig.json with the following:

json

Copy

{

"compilerOptions": {

"target": "es5",

"lib": ["dom", "dom.iterable", "esnext"],

"allowJs": true,

"skipLibCheck": true,

"strict": true,

"forceConsistentCasingInFileNames": true,

"noEmit": true,

"esModuleInterop": true,

"module": "esnext",

"moduleResolution": "node",

"resolveJsonModule": true,

"isolatedModules": true,

"jsx": "preserve",

"incremental": true

},

"include": ["\*\*/\*.ts", "\*\*/\*.tsx"],

"exclude": ["node\_modules"]

}

**3. Create next.config.js**

Next.js uses a next.config.js file for configuration. Create this file in the root of your project:

javascript

Copy

/\*\* @type {import('next').NextConfig} \*/

const nextConfig = {

reactStrictMode: true,

swcMinify: true,

images: {

domains: ['example.com'], // Add any domains you need for images

},

};

module.exports = nextConfig;

**4. Update package.json**

Update the package.json to include Next.js dependencies and scripts. Replace the content of package.json with the following:

json

Copy

{

"name": "dashboard-nextjs",

"version": "1.0.0",

"scripts": {

"dev": "next dev",

"build": "next build",

"start": "next start",

"lint": "next lint"

},

"dependencies": {

"next": "^14.0.0",

"react": "^18.2.0",

"react-dom": "^18.2.0"

},

"devDependencies": {

"eslint": "^8.0.0",

"eslint-config-next": "^14.0.0",

"typescript": "^5.0.0",

"@types/react": "^18.2.0",

"@types/node": "^20.0.0"

}

}

**5. Create pages Directory**

Next.js uses a pages directory for routing. Create this directory and add an index.tsx file:

bash

Copy

mkdir pages

touch pages/index.tsx

Add the following content to pages/index.tsx:

tsx

Copy

import React from 'react';

const HomePage: React.FC = () => {

return (

<div>

<h1>Welcome to Next.js!</h1>

</div>

);

};

export default HomePage;

**6. Update .gitignore**

Update the .gitignore file to include Next.js specific files:

plaintext

Copy

# See https://docs.github.com/get-started/getting-started-with-git/ignoring-files for more about ignoring files.

# Compiled output

/dist

/tmp

/out-tsc

/bazel-out

/.next

/out

# Node

/node\_modules

npm-debug.log

yarn-error.log

# IDEs and editors

.idea/

.project

.classpath

.c9/

\*.launch

.settings/

\*.sublime-workspace

# Visual Studio Code

.vscode/

.vscode/settings.json

.vscode/tasks.json

.vscode/launch.json

.vscode/extensions.json

.history/\*

# Miscellaneous

/.angular/cache

.sass-cache/

/connect.lock

/coverage

/libpeerconnection.log

testem.log

/typings

# System files

.DS\_Store

Thumbs.db

**7. Update README.md**

Update the README.md to reflect the new project structure and commands:

markdown

Copy

# DashboardNextJs

This project was generated using [Next.js](https://nextjs.org/) version 14.0.0.

## Development server

To start a local development server, run:

```bash

npm run dev

Once the server is running, open your browser and navigate to http://localhost:3000/. The application will automatically reload whenever you modify any of the source files.

**Building**

To build the project run:

bash

Copy

npm run build

This will compile your project and store the build artifacts in the .next/ directory.

**Running the production server**

To start the production server, run:

bash

Copy

npm run start

**Additional Resources**

For more information on using Next.js, including detailed command references, visit the [Next.js Documentation](https://nextjs.org/docs) page.

Copy

### 8. Install Dependencies

Finally, install the dependencies:

```bash

npm install

**9. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/.

**Summary**

* Removed Angular-specific files.
* Updated tsconfig.json for Next.js TypeScript configuration.
* Created next.config.js for Next.js configuration.
* Updated package.json with Next.js dependencies and scripts.
* Created a pages directory with an index.tsx file for the home page.
* Updated .gitignore to include Next.js specific files.
* Updated README.md to reflect the new project structure.

This should give you a basic Next.js project structure. You can now start building your React components and pages using Next.js conventions.

index.html

HTML 1.55KB

main.ts

TS 250B

styles.scss

SCSS 183B

To convert the Angular project into a React Next.js project, you need to replace Angular-specific files and configurations with React Next.js equivalents. Below is a step-by-step guide to help you transition:

**1. Remove Angular-Specific Files**

Remove the following Angular-specific files, as they are no longer needed in a Next.js project:

* index.html
* main.ts
* styles.scss
* Any Angular-specific components, services, and modules.

**2. Create a New pages/\_app.tsx File**

Next.js uses a \_app.tsx file to initialize pages. This file is where you can include global styles or scripts. Create the file in the pages directory:

bash

Copy

mkdir -p pages

touch pages/\_app.tsx

Add the following content to pages/\_app.tsx:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css'; // Import global styles here

function MyApp({ Component, pageProps }: AppProps) {

return <Component {...pageProps} />;

}

export default MyApp;

**3. Create a New pages/index.tsx File**

Next.js uses the pages directory for routing. The index.tsx file will serve as the home page. Create it in the pages directory:

bash

Copy

touch pages/index.tsx

Add the following content to pages/index.tsx:

tsx

Copy

import React from 'react';

const HomePage: React.FC = () => {

return (

<div>

<h1>Welcome to ERP Version II (Next.js)</h1>

<p>This is the home page of the ERP application.</p>

</div>

);

};

export default HomePage;

**4. Move Global Styles to styles/globals.css**

Next.js uses CSS for styling. Move the global styles from styles.scss to a new styles/globals.css file:

bash

Copy

mkdir styles

touch styles/globals.css

Add the following content to styles/globals.css:

css

Copy

/\* Import AG-Grid styles \*/

@import 'ag-grid-community/styles/ag-grid.css';

@import 'ag-grid-community/styles/ag-theme-alpine.css';

/\* Add additional global styles here \*/

body {

margin: 0;

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

**5. Update package.json**

Update the package.json to include Next.js dependencies and scripts. Replace the content of package.json with the following:

json

Copy

{

"name": "erp-nextjs",

"version": "1.0.0",

"scripts": {

"dev": "next dev",

"build": "next build",

"start": "next start",

"lint": "next lint"

},

"dependencies": {

"next": "^14.0.0",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"ag-grid-community": "^30.0.0", // Add AG-Grid if needed

"ag-grid-react": "^30.0.0" // Add AG-Grid React wrapper if needed

},

"devDependencies": {

"eslint": "^8.0.0",

"eslint-config-next": "^14.0.0",

"typescript": "^5.0.0",

"@types/react": "^18.2.0",

"@types/node": "^20.0.0"

}

}

**6. Install Dependencies**

Install the required dependencies by running:

bash

Copy

npm install

**7. Update .gitignore**

Update the .gitignore file to include Next.js-specific files:

plaintext

Copy

# See https://docs.github.com/get-started/getting-started-with-git/ignoring-files for more about ignoring files.

# Compiled output

/dist

/tmp

/out-tsc

/bazel-out

/.next

/out

# Node

/node\_modules

npm-debug.log

yarn-error.log

# IDEs and editors

.idea/

.project

.classpath

.c9/

\*.launch

.settings/

\*.sublime-workspace

# Visual Studio Code

.vscode/

.vscode/settings.json

.vscode/tasks.json

.vscode/launch.json

.vscode/extensions.json

.history/\*

# Miscellaneous

/.angular/cache

.sass-cache/

/connect.lock

/coverage

/libpeerconnection.log

testem.log

/typings

# System files

.DS\_Store

Thumbs.db

**8. Update tsconfig.json**

Update the tsconfig.json file to reflect Next.js conventions. Replace the content of tsconfig.json with the following:

json

Copy

{

"compilerOptions": {

"target": "es5",

"lib": ["dom", "dom.iterable", "esnext"],

"allowJs": true,

"skipLibCheck": true,

"strict": true,

"forceConsistentCasingInFileNames": true,

"noEmit": true,

"esModuleInterop": true,

"module": "esnext",

"moduleResolution": "node",

"resolveJsonModule": true,

"isolatedModules": true,

"jsx": "preserve",

"incremental": true

},

"include": ["\*\*/\*.ts", "\*\*/\*.tsx"],

"exclude": ["node\_modules"]

}

**9. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/.

**10. Add External Scripts and Styles**

If you need to include external scripts or styles (e.g., Bootstrap, Font Awesome, Chart.js), you can add them to the pages/\_app.tsx file or directly in the pages/index.tsx file.

For example, to include Bootstrap and Font Awesome in pages/\_app.tsx:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css';

import 'bootstrap/dist/css/bootstrap.min.css';

import '@fortawesome/fontawesome-free/css/all.min.css';

function MyApp({ Component, pageProps }: AppProps) {

return <Component {...pageProps} />;

}

export default MyApp;

**Summary**

* Removed Angular-specific files (index.html, main.ts, styles.scss).
* Created a new pages/\_app.tsx file for global initialization.
* Created a new pages/index.tsx file for the home page.
* Moved global styles to styles/globals.css.
* Updated package.json with Next.js dependencies and scripts.
* Updated .gitignore to include Next.js-specific files.
* Updated tsconfig.json for Next.js TypeScript configuration.
* Installed dependencies and started the development server.

This should give you a basic Next.js project structure. You can now start building your React components and pages using Next.js conventions.

app.component.html

HTML 286B

app.component.scss

SCSS 1.06KB

app.component.spec.ts

TS 940B

app.component.ts

TS 575B

app.config.ts

TS 323B

app.routes.ts

TS 479B

To convert the Angular components and routing into a React Next.js project, you need to replace Angular-specific code with React components and Next.js routing. Below is a step-by-step guide to help you transition:

**1. Remove Angular-Specific Files**

Remove the following Angular-specific files, as they are no longer needed in a Next.js project:

* app.component.html
* app.component.scss
* app.component.spec.ts
* app.component.ts
* app.config.ts
* app.routes.ts

**2. Create a New components Directory**

Next.js uses a components directory to organize React components. Create this directory and add the necessary components:

bash

Copy

mkdir components

**3. Create a Navbar Component**

Create a Navbar component in the components directory:

bash

Copy

touch components/Navbar.tsx

Add the following content to components/Navbar.tsx:

tsx

Copy

import React from 'react';

interface NavbarProps {

onSidebarToggle: () => void;

}

const Navbar: React.FC<NavbarProps> = ({ onSidebarToggle }) => {

return (

<nav className="top-bar">

<button onClick={onSidebarToggle}>Toggle Sidebar</button>

</nav>

);

};

export default Navbar;

**4. Create a Sidebar Component**

Create a Sidebar component in the components directory:

bash

Copy

touch components/Sidebar.tsx

Add the following content to components/Sidebar.tsx:

tsx

Copy

import React from 'react';

interface SidebarProps {

isOpen: boolean;

}

const Sidebar: React.FC<SidebarProps> = ({ isOpen }) => {

return (

<div className={`top-bar\_\_side-bar ${isOpen ? 'open' : ''}`}>

<div className="sidebar-content">

<p>Sidebar Content</p>

</div>

</div>

);

};

export default Sidebar;

**5. Update pages/\_app.tsx**

Update the pages/\_app.tsx file to include global styles and state management for the sidebar:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css';

import { useState } from 'react';

import Navbar from '../components/Navbar';

import Sidebar from '../components/Sidebar';

function MyApp({ Component, pageProps }: AppProps) {

const [isSidebarOpen, setIsSidebarOpen] = useState(true);

const handleSidebarToggle = () => {

setIsSidebarOpen(!isSidebarOpen);

};

return (

<div className="app-container">

<Navbar onSidebarToggle={handleSidebarToggle} />

<Sidebar isOpen={isSidebarOpen} />

<div className="top-bar\_\_content-section">

<Component {...pageProps} />

</div>

</div>

);

}

export default MyApp;

**6. Add Global Styles**

Update the styles/globals.css file to include the styles for the Navbar, Sidebar, and content-section:

css

Copy

/\* Global Styles \*/

body {

margin: 0;

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

.app-container {

display: grid;

grid-template-areas:

"navbar navbar"

"sidebar content";

grid-template-columns: auto 1fr;

grid-template-rows: auto 1fr;

height: 100vh;

overflow-x: hidden;

}

.top-bar {

grid-area: navbar;

}

.top-bar\_\_side-bar {

grid-area: sidebar;

width: 0;

transition: width 0.3s ease;

overflow-x: hidden;

white-space: nowrap;

}

.top-bar\_\_side-bar.open {

width: 250px;

}

.top-bar\_\_side-bar .sidebar-content {

opacity: 0;

visibility: hidden;

transition: opacity 0.2s ease, visibility 0.2s ease;

}

.top-bar\_\_side-bar.open .sidebar-content {

opacity: 1;

visibility: visible;

}

.top-bar\_\_content-section {

grid-area: content;

transition: margin-left 0.3s ease;

}

**7. Create Pages for Routing**

Next.js uses the pages directory for routing. Create the following pages:

**pages/index.tsx (Dashboard Page)**

tsx

Copy

import React from 'react';

const Dashboard: React.FC = () => {

return (

<div>

<h1>Dashboard</h1>

<p>Welcome to the Dashboard page.</p>

</div>

);

};

export default Dashboard;

**pages/charts.tsx (Charts Page)**

tsx

Copy

import React from 'react';

const Charts: React.FC = () => {

return (

<div>

<h1>Charts</h1>

<p>Welcome to the Charts page.</p>

</div>

);

};

export default Charts;

**pages/tables.tsx (Tables Page)**

tsx

Copy

import React from 'react';

const Tables: React.FC = () => {

return (

<div>

<h1>Tables</h1>

<p>Welcome to the Tables page.</p>

</div>

);

};

export default Tables;

**8. Update package.json**

Ensure your package.json includes the necessary dependencies for Next.js and React:

json

Copy

{

"name": "erp-nextjs",

"version": "1.0.0",

"scripts": {

"dev": "next dev",

"build": "next build",

"start": "next start",

"lint": "next lint"

},

"dependencies": {

"next": "^14.0.0",

"react": "^18.2.0",

"react-dom": "^18.2.0"

},

"devDependencies": {

"eslint": "^8.0.0",

"eslint-config-next": "^14.0.0",

"typescript": "^5.0.0",

"@types/react": "^18.2.0",

"@types/node": "^20.0.0"

}

}

**9. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/.

**Summary**

* Removed Angular-specific files.
* Created React components for Navbar and Sidebar.
* Updated pages/\_app.tsx to include global state and layout.
* Added global styles in styles/globals.css.
* Created pages for routing (index.tsx, charts.tsx, tables.tsx).
* Updated package.json with Next.js dependencies.

This should give you a basic Next.js project structure with a responsive layout. You can now start building your React components and pages using Next.js conventions.

charts.component.html

HTML 302B

charts.component.scss

SCSS 60B

charts.component.spec.ts

TS 592B

charts.component.ts

TS 2.15KB

To convert the Angular ChartsComponent into a React component for a Next.js project, you need to replace Angular-specific code with React and use a compatible charting library like highcharts or react-highcharts. Below is a step-by-step guide to help you transition:

**1. Install Highcharts for React**

First, install the necessary Highcharts packages for React:

bash

Copy

npm install highcharts highcharts-react-official

**2. Create a Charts Component**

Create a new Charts component in the components directory:

bash

Copy

touch components/Charts.tsx

Add the following content to components/Charts.tsx:

tsx

Copy

import React from 'react';

import Highcharts from 'highcharts';

import HighchartsReact from 'highcharts-react-official';

const Charts: React.FC = () => {

// Define an area chart using Highcharts

const areaChartOptions: Highcharts.Options = {

chart: {

type: 'area',

},

title: {

text: 'Area Chart',

},

xAxis: {

categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],

},

yAxis: {

title: {

text: 'Value',

},

},

series: [

{

name: 'Series 1',

type: 'area',

data: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],

color: '#7cb5ec',

fillOpacity: 0.3,

},

],

credits: {

enabled: false,

},

plotOptions: {

area: {

marker: {

enabled: false,

},

},

},

};

// Define a pie chart using Highcharts

const pieChartOptions: Highcharts.Options = {

chart: {

plotBackgroundColor: undefined,

plotBorderWidth: undefined,

plotShadow: false,

type: 'pie',

},

title: {

text: 'HR Employees Based on Department',

},

tooltip: {

pointFormat: '{series.name}: <b>{point.percentage:.1f}%</b>',

},

accessibility: {

point: {

valueSuffix: '%',

},

},

plotOptions: {

pie: {

allowPointSelect: true,

cursor: 'pointer',

dataLabels: {

enabled: true,

format: '<b>{point.name}</b>: {point.percentage:.1f} %',

},

},

},

series: [

{

name: 'Department',

type: 'pie',

data: [

{ name: 'Computer Science', y: 70.67, sliced: true, selected: true },

{ name: 'Engineering', y: 14.77 },

{ name: 'Management', y: 4.86 },

{ name: 'Economics', y: 2.63 },

{ name: 'Marketing', y: 1.53 },

{ name: 'Others', y: 5.54 },

],

},

],

credits: {

enabled: false,

},

};

return (

<div className="charts-root">

<HighchartsReact highcharts={Highcharts} options={areaChartOptions} />

<HighchartsReact highcharts={Highcharts} options={pieChartOptions} />

</div>

);

};

export default Charts;

**3. Add Styles for the Charts Component**

Update the styles/globals.css file to include styles for the charts-root class:

css

Copy

.charts-root {

display: flex;

flex-direction: row;

gap: 20px;

padding: 20px;

}

.charts-root > div {

flex: 1;

}

**4. Update the pages/charts.tsx File**

Update the pages/charts.tsx file to use the Charts component:

tsx

Copy

import React from 'react';

import Charts from '../components/Charts';

const ChartsPage: React.FC = () => {

return (

<div>

<h1>Charts</h1>

<Charts />

</div>

);

};

export default ChartsPage;

**5. Update pages/\_app.tsx**

Ensure the pages/\_app.tsx file includes the necessary layout and styles:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css';

import { useState } from 'react';

import Navbar from '../components/Navbar';

import Sidebar from '../components/Sidebar';

function MyApp({ Component, pageProps }: AppProps) {

const [isSidebarOpen, setIsSidebarOpen] = useState(true);

const handleSidebarToggle = () => {

setIsSidebarOpen(!isSidebarOpen);

};

return (

<div className="app-container">

<Navbar onSidebarToggle={handleSidebarToggle} />

<Sidebar isOpen={isSidebarOpen} />

<div className="top-bar\_\_content-section">

<Component {...pageProps} />

</div>

</div>

);

}

export default MyApp;

**6. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/. Navigate to http://localhost:3000/charts to see the charts.

**Summary**

* Installed highcharts and highcharts-react-official for charting.
* Created a Charts component using Highcharts for React.
* Added styles for the charts in styles/globals.css.
* Updated the pages/charts.tsx file to use the Charts component.
* Ensured the layout and styles are applied in pages/\_app.tsx.

This should give you a fully functional React Next.js project with Highcharts integration. You can now customize the charts further or add more components as needed.

dashboard.component.html

HTML 3.81KB

dashboard.component.spec.ts

TS 613B

dashboard.component.ts

TS 390B

To convert the Angular DashboardComponent into a React component for a Next.js project, you need to replace Angular-specific code with React components and JSX. Below is a step-by-step guide to help you transition:

**1. Remove Angular-Specific Files**

Remove the following Angular-specific files, as they are no longer needed in a Next.js project:

* dashboard.component.html
* dashboard.component.spec.ts
* dashboard.component.ts

**2. Create a Dashboard Component**

Create a new Dashboard component in the components directory:

bash

Copy

touch components/Dashboard.tsx

Add the following content to components/Dashboard.tsx:

tsx

Copy

import React from 'react';

import Charts from './Charts';

import Table from './Table'; // Assuming you have a Table component

const Dashboard: React.FC = () => {

return (

<div id="layoutSidenav\_content">

<main>

<div className="container-fluid px-4">

<h1 className="mt-4">Dashboard</h1>

<ol className="breadcrumb mb-4">

<li className="breadcrumb-item active">Dashboard</li>

</ol>

<div className="row">

<div className="col-xl-3 col-md-6">

<div className="card bg-primary text-white mb-4">

<div className="card-body">HR manager</div>

<div className="card-footer d-flex align-items-center justify-content-between">

<a className="small text-white stretched-link" href="#">View Details</a>

<div className="small text-white"><i className="fas fa-angle-right"></i></div>

</div>

</div>

</div>

<div className="col-xl-3 col-md-6">

<div className="card bg-warning text-white mb-4">

<div className="card-body">Payroll</div>

<div className="card-footer d-flex align-items-center justify-content-between">

<a className="small text-white stretched-link" href="#">View Details</a>

<div className="small text-white"><i className="fas fa-angle-right"></i></div>

</div>

</div>

</div>

<div className="col-xl-3 col-md-6">

<div className="card bg-success text-white mb-4">

<div className="card-body">Procurement</div>

<div className="card-footer d-flex align-items-center justify-content-between">

<a className="small text-white stretched-link" href="#">View Details</a>

<div className="small text-white"><i className="fas fa-angle-right"></i></div>

</div>

</div>

</div>

<div className="col-xl-3 col-md-6">

<div className="card bg-danger text-white mb-4">

<div className="card-body">Lookup</div>

<div className="card-footer d-flex align-items-center justify-content-between">

<a className="small text-white stretched-link" href="#">View Details</a>

<div className="small text-white"><i className="fas fa-angle-right"></i></div>

</div>

</div>

</div>

</div>

<div className="row">

<Charts />

</div>

<Table />

</div>

</main>

<footer className="py-4 bg-light mt-auto">

<div className="container-fluid px-4">

<div className="d-flex align-items-center justify-content-between small">

<div className="text-muted">Copyright © 2025 INSA - All rights reserved</div>

<div>

<a href="https://web.facebook.com/INSA.ETHIOPIA?\_rdc=1&\_rdr#" target="\_blank" className="text-primary me-3">

<i className="fab fa-facebook fa-lg"></i>

</a>

<a href="https://insa.gov.et" target="\_blank" className="text-primary me-3">

<i className="fas fa-globe fa-lg"></i>

</a>

<a href="https://www.youtube.com/channel/UCTc5kO3p6xAQSObDdx\_Eg3w" target="\_blank" className="text-primary me-3">

<i className="fab fa-youtube fa-lg"></i>

</a>

<a href="https://mail.insa.gov.et" target="\_blank" className="text-primary">

<i className="fas fa-envelope fa-lg"></i>

</a>

</div>

</div>

</div>

</footer>

</div>

);

};

export default Dashboard;

**3. Create a Table Component**

If you don't already have a Table component, create one in the components directory:

bash

Copy

touch components/Table.tsx

Add the following content to components/Table.tsx:

tsx

Copy

import React from 'react';

const Table: React.FC = () => {

return (

<div className="card mb-4">

<div className="card-header">

<i className="fas fa-table me-1"></i>

Data Table

</div>

<div className="card-body">

<table className="table table-bordered">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Position</th>

<th>Salary</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>John Doe</td>

<td>Software Engineer</td>

<td>$120,000</td>

</tr>

<tr>

<td>2</td>

<td>Jane Smith</td>

<td>Product Manager</td>

<td>$150,000</td>

</tr>

</tbody>

</table>

</div>

</div>

);

};

export default Table;

**4. Update pages/index.tsx**

Update the pages/index.tsx file to use the Dashboard component:

tsx

Copy

import React from 'react';

import Dashboard from '../components/Dashboard';

const HomePage: React.FC = () => {

return <Dashboard />;

};

export default HomePage;

**5. Add Bootstrap and Font Awesome**

To ensure the styles and icons work correctly, add Bootstrap and Font Awesome to your project.

**Install Bootstrap and Font Awesome:**

bash

Copy

npm install bootstrap @fortawesome/fontawesome-free

**Import Bootstrap and Font Awesome in styles/globals.css:**

css

Copy

@import 'bootstrap/dist/css/bootstrap.min.css';

@import '@fortawesome/fontawesome-free/css/all.min.css';

**6. Update pages/\_app.tsx**

Ensure the pages/\_app.tsx file includes the necessary layout and styles:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css';

import { useState } from 'react';

import Navbar from '../components/Navbar';

import Sidebar from '../components/Sidebar';

function MyApp({ Component, pageProps }: AppProps) {

const [isSidebarOpen, setIsSidebarOpen] = useState(true);

const handleSidebarToggle = () => {

setIsSidebarOpen(!isSidebarOpen);

};

return (

<div className="app-container">

<Navbar onSidebarToggle={handleSidebarToggle} />

<Sidebar isOpen={isSidebarOpen} />

<div className="top-bar\_\_content-section">

<Component {...pageProps} />

</div>

</div>

);

}

export default MyApp;

**7. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/. The dashboard will display the cards, charts, and table as defined in the Dashboard component.

**Summary**

* Removed Angular-specific files.
* Created a Dashboard component using React and JSX.
* Created a Table component for displaying data.
* Updated pages/index.tsx to use the Dashboard component.
* Added Bootstrap and Font Awesome for styling and icons.
* Ensured the layout and styles are applied in pages/\_app.tsx.

This should give you a fully functional React Next.js project with a dashboard layout. You can now customize the dashboard further or add more components as needed.

navbar.component.html

HTML 1.36KB

navbar.component.spec.ts

TS 592B

navbar.component.ts

TS 459B

To convert the Angular NavbarComponent into a React component for a Next.js project, you need to replace Angular-specific code with React and JSX. Below is a step-by-step guide to help you transition:

**1. Remove Angular-Specific Files**

Remove the following Angular-specific files, as they are no longer needed in a Next.js project:

* navbar.component.html
* navbar.component.spec.ts
* navbar.component.ts

**2. Create a Navbar Component**

Create a new Navbar component in the components directory:

bash

Copy

touch components/Navbar.tsx

Add the following content to components/Navbar.tsx:

tsx

Copy

import React, { useState } from 'react';

interface NavbarProps {

onSidebarToggle: (isOpen: boolean) => void;

}

const Navbar: React.FC<NavbarProps> = ({ onSidebarToggle }) => {

const [toggleSidebar, setToggleSidebar] = useState(false);

const handleButtonClick = () => {

const newToggleState = !toggleSidebar;

setToggleSidebar(newToggleState);

onSidebarToggle(newToggleState);

};

return (

<nav className="sb-topnav navbar navbar-expand navbar-dark bg-dark">

<a className="navbar-brand ps-3">INSA ERP</a>

<button className="btn btn-link btn-sm order-1 order-lg-0 me-4 me-lg-0" onClick={handleButtonClick}>

<i className="fas fa-bars"></i>

</button>

<form className="d-none d-md-inline-block form-inline ms-auto me-0 me-md-3 my-2 my-md-0">

<div className="input-group">

<input

className="form-control"

type="text"

placeholder="Search for..."

aria-label="Search for..."

aria-describedby="btnNavbarSearch"

/>

<button className="btn btn-primary" id="btnNavbarSearch" type="button">

<i className="fas fa-search"></i>

</button>

</div>

</form>

<ul className="navbar-nav ms-auto ms-md-0 me-3 me-lg-4">

<li className="nav-item dropdown">

<a

className="nav-link dropdown-toggle"

id="navbarDropdown"

role="button"

data-bs-toggle="dropdown"

aria-expanded="false"

></a>

<ul className="dropdown-menu dropdown-menu-end" aria-labelledby="navbarDropdown">

<li>

<a className="dropdown-item" href="#!">

Settings

</a>

</li>

<li>

<a className="dropdown-item" href="#!">

Activity Log

</a>

</li>

<li>

<hr className="dropdown-divider" />

</li>

<li>

<a className="dropdown-item" href="#!">

Logout

</a>

</li>

</ul>

</li>

</ul>

</nav>

);

};

export default Navbar;

**3. Add Bootstrap and Font Awesome**

To ensure the styles and icons work correctly, add Bootstrap and Font Awesome to your project.

**Install Bootstrap and Font Awesome:**

bash

Copy

npm install bootstrap @fortawesome/fontawesome-free

**Import Bootstrap and Font Awesome in styles/globals.css:**

css

Copy

@import 'bootstrap/dist/css/bootstrap.min.css';

@import '@fortawesome/fontawesome-free/css/all.min.css';

**4. Update pages/\_app.tsx**

Ensure the pages/\_app.tsx file includes the Navbar component and handles the sidebar toggle state:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css';

import { useState } from 'react';

import Navbar from '../components/Navbar';

import Sidebar from '../components/Sidebar';

function MyApp({ Component, pageProps }: AppProps) {

const [isSidebarOpen, setIsSidebarOpen] = useState(true);

const handleSidebarToggle = (isOpen: boolean) => {

setIsSidebarOpen(isOpen);

};

return (

<div className="app-container">

<Navbar onSidebarToggle={handleSidebarToggle} />

<Sidebar isOpen={isSidebarOpen} />

<div className="top-bar\_\_content-section">

<Component {...pageProps} />

</div>

</div>

);

}

export default MyApp;

**5. Update Sidebar Component**

Ensure the Sidebar component (if it exists) receives the isOpen prop and adjusts its visibility accordingly. For example:

tsx

Copy

import React from 'react';

interface SidebarProps {

isOpen: boolean;

}

const Sidebar: React.FC<SidebarProps> = ({ isOpen }) => {

return (

<div className={`sidebar ${isOpen ? 'open' : ''}`}>

<p>Sidebar Content</p>

</div>

);

};

export default Sidebar;

**6. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/. The navbar will display correctly, and the sidebar toggle functionality will work as expected.

**Summary**

* Removed Angular-specific files.
* Created a Navbar component using React and JSX.
* Added Bootstrap and Font Awesome for styling and icons.
* Updated pages/\_app.tsx to include the Navbar component and handle the sidebar toggle state.
* Ensured the Sidebar component responds to the isOpen prop.

This should give you a fully functional React Next.js project with a responsive navbar and sidebar. You can now customize the navbar further or add more components as needed.

sidebar.component.html

HTML 6.46KB

sidebar.component.spec.ts

TS 599B

sidebar.component.ts

TS 286B

To convert the Angular SidebarComponent into a React component for a Next.js project, you need to replace Angular-specific code with React and JSX. Below is a step-by-step guide to help you transition:

**1. Remove Angular-Specific Files**

Remove the following Angular-specific files, as they are no longer needed in a Next.js project:

* sidebar.component.html
* sidebar.component.spec.ts
* sidebar.component.ts

**2. Create a Sidebar Component**

Create a new Sidebar component in the components directory:

bash

Copy

touch components/Sidebar.tsx

Add the following content to components/Sidebar.tsx:

tsx

Copy

import React from 'react';

import Link from 'next/link';

interface SidebarProps {

isOpen: boolean;

}

const Sidebar: React.FC<SidebarProps> = ({ isOpen }) => {

return (

<div id="layoutSidenav\_nav" style={{ height: '100%' }}>

<nav className={`sb-sidenav accordion sb-sidenav-dark ${isOpen ? 'open' : ''}`} id="sidenavAccordion">

<div className="sb-sidenav-menu">

<div className="nav">

{/\* Dashboard Section \*/}

<div className="sb-sidenav-menu-heading">Core</div>

<Link href="/" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-tachometer-alt"></i></div>

Dashboard

</Link>

{/\* Payroll Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-money-bill-wave"></i></div>

Payroll

</Link>

{/\* Procurement Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-shopping-cart"></i></div>

Procurement

</Link>

{/\* Material Management Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-box"></i></div>

Material Management

</Link>

{/\* Lookup Manager Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-search"></i></div>

Lookup Manager

</Link>

{/\* HR Management Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-users"></i></div>

HR Management

</Link>

{/\* Profile Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-user"></i></div>

Profile

</Link>

{/\* Internal Revenue Finance Section \*/}

<Link href="#" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-money-check-alt"></i></div>

Internal Revenue Finance

</Link>

{/\* Layouts Section \*/}

<div className="sb-sidenav-menu-heading">Interface</div>

<a

className="nav-link collapsed"

href="#"

data-bs-toggle="collapse"

data-bs-target="#collapseLayouts"

aria-expanded="false"

aria-controls="collapseLayouts"

>

<div className="sb-nav-link-icon"><i className="fas fa-columns"></i></div>

Layouts

<div className="sb-sidenav-collapse-arrow"><i className="fas fa-angle-down"></i></div>

</a>

<div className="collapse" id="collapseLayouts" aria-labelledby="headingOne" data-bs-parent="#sidenavAccordion">

<nav className="sb-sidenav-menu-nested nav">

<Link href="/layout-static" className="nav-link">

Static Navigation

</Link>

<Link href="/layout-sidenav-light" className="nav-link">

Light Sidenav

</Link>

</nav>

</div>

{/\* Pages Section \*/}

<a

className="nav-link collapsed"

href="#"

data-bs-toggle="collapse"

data-bs-target="#collapsePages"

aria-expanded="false"

aria-controls="collapsePages"

>

<div className="sb-nav-link-icon"><i className="fas fa-book-open"></i></div>

Pages

<div className="sb-sidenav-collapse-arrow"><i className="fas fa-angle-down"></i></div>

</a>

<div className="collapse" id="collapsePages" aria-labelledby="headingTwo" data-bs-parent="#sidenavAccordion">

<nav className="sb-sidenav-menu-nested nav accordion" id="sidenavAccordionPages">

<a

className="nav-link collapsed"

href="#"

data-bs-toggle="collapse"

data-bs-target="#pagesCollapseAuth"

aria-expanded="false"

aria-controls="pagesCollapseAuth"

>

Authentication

<div className="sb-sidenav-collapse-arrow"><i className="fas fa-angle-down"></i></div>

</a>

<div className="collapse" id="pagesCollapseAuth" aria-labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">

<nav className="sb-sidenav-menu-nested nav">

<Link href="/login" className="nav-link">

Login

</Link>

<Link href="/register" className="nav-link">

Register

</Link>

<Link href="/password" className="nav-link">

Forgot Password

</Link>

</nav>

</div>

<a

className="nav-link collapsed"

href="#"

data-bs-toggle="collapse"

data-bs-target="#pagesCollapseError"

aria-expanded="false"

aria-controls="pagesCollapseError"

>

Error

<div className="sb-sidenav-collapse-arrow"><i className="fas fa-angle-down"></i></div>

</a>

<div className="collapse" id="pagesCollapseError" aria-labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">

<nav className="sb-sidenav-menu-nested nav">

<Link href="/401" className="nav-link">

401 Page

</Link>

<Link href="/404" className="nav-link">

404 Page

</Link>

<Link href="/500" className="nav-link">

500 Page

</Link>

</nav>

</div>

</nav>

</div>

{/\* Addons Section \*/}

<div className="sb-sidenav-menu-heading">Addons</div>

<Link href="/charts" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-chart-area"></i></div>

Charts

</Link>

<Link href="/tables" className="nav-link">

<div className="sb-nav-link-icon"><i className="fas fa-table"></i></div>

Tables

</Link>

</div>

</div>

{/\* Footer Section \*/}

<div className="sb-sidenav-footer">

<div className="small">Logged in as:</div>

INSA ERP

</div>

</nav>

</div>

);

};

export default Sidebar;

**3. Add Bootstrap and Font Awesome**

To ensure the styles and icons work correctly, add Bootstrap and Font Awesome to your project.

**Install Bootstrap and Font Awesome:**

bash

Copy

npm install bootstrap @fortawesome/fontawesome-free

**Import Bootstrap and Font Awesome in styles/globals.css:**

css

Copy

@import 'bootstrap/dist/css/bootstrap.min.css';

@import '@fortawesome/fontawesome-free/css/all.min.css';

**4. Update pages/\_app.tsx**

Ensure the pages/\_app.tsx file includes the Sidebar component and handles the sidebar toggle state:

tsx

Copy

import type { AppProps } from 'next/app';

import '../styles/globals.css';

import { useState } from 'react';

import Navbar from '../components/Navbar';

import Sidebar from '../components/Sidebar';

function MyApp({ Component, pageProps }: AppProps) {

const [isSidebarOpen, setIsSidebarOpen] = useState(true);

const handleSidebarToggle = (isOpen: boolean) => {

setIsSidebarOpen(isOpen);

};

return (

<div className="app-container">

<Navbar onSidebarToggle={handleSidebarToggle} />

<Sidebar isOpen={isSidebarOpen} />

<div className="top-bar\_\_content-section">

<Component {...pageProps} />

</div>

</div>

);

}

export default MyApp;

**5. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/. The sidebar will display correctly, and the toggle functionality will work as expected.

**Summary**

* Removed Angular-specific files.
* Created a Sidebar component using React and JSX.
* Added Bootstrap and Font Awesome for styling and icons.
* Updated pages/\_app.tsx to include the Sidebar component and handle the sidebar toggle state.

This should give you a fully functional React Next.js project with a responsive sidebar. You can now customize the sidebar further or add more components as needed.

table.component.html

HTML 290B

table.component.spec.ts

TS 585B

table.component.ts

TS 2.44KB

To convert the Angular TableComponent into a React component for a Next.js project, you need to replace Angular-specific code with React and use a compatible table library like ag-grid-react. Below is a step-by-step guide to help you transition:

**1. Remove Angular-Specific Files**

Remove the following Angular-specific files, as they are no longer needed in a Next.js project:

* table.component.html
* table.component.spec.ts
* table.component.ts

**2. Install AG Grid for React**

Install the necessary AG Grid packages for React:

bash

Copy

npm install ag-grid-react ag-grid-community

**3. Create a Table Component**

Create a new Table component in the components directory:

bash

Copy

touch components/Table.tsx

Add the following content to components/Table.tsx:

tsx

Copy

import React, { useState } from 'react';

import { AgGridReact } from 'ag-grid-react';

import { ColDef, GridApi, GridReadyEvent } from 'ag-grid-community';

import 'ag-grid-community/styles/ag-grid.css';

import 'ag-grid-community/styles/ag-theme-alpine.css';

const Table: React.FC = () => {

const [gridApi, setGridApi] = useState<GridApi | null>(null);

const rowData = [

{

name: 'John Doe',

age: 30,

region: 'bbbb',

email: 'john@example.com',

salary: 75000,

department: 'Engineering',

status: 'Active',

joinDate: '2022-01-15',

},

{

name: 'Mary Smith',

age: 28,

region: 'bbbbb',

email: 'mary@example.com',

salary: 65000,

department: 'Marketing',

status: 'Active',

joinDate: '2022-03-20',

},

{

name: 'Bob Wilson',

age: 32,

region: 'bbbbb',

email: 'bob@example.com',

salary: 80000,

department: 'Sales',

status: 'Inactive',

joinDate: '2021-11-10',

},

];

const columnDefs: ColDef[] = [

{

field: 'name',

sortable: true,

filter: true,

headerName: 'Full Name',

},

{

field: 'age',

sortable: true,

filter: 'agNumberColumnFilter',

},

{

field: 'region',

sortable: true,

filter: true,

headerName: 'Region',

},

{

field: 'email',

sortable: true,

filter: true,

},

{

field: 'salary',

sortable: true,

filter: 'agNumberColumnFilter',

valueFormatter: (params: any) => `${params.value.toLocaleString()} ብር`, // Ethiopian Birr after the number

},

{

field: 'department',

sortable: true,

filter: true,

},

{

field: 'status',

sortable: true,

filter: true,

cellRenderer: (params: any) => {

return `<span class="badge ${params.value === 'Active' ? 'bg-success' : 'bg-danger'}">${params.value}</span>`;

},

},

{

field: 'joinDate',

sortable: true,

filter: 'agDateColumnFilter',

valueFormatter: (params: any) => new Date(params.value).toLocaleDateString(),

},

];

const onGridReady = (params: GridReadyEvent) => {

setGridApi(params.api);

};

return (

<div className="ag-theme-alpine" style={{ width: '100%', height: '500px', padding: '24px' }}>

<AgGridReact

columnDefs={columnDefs}

rowData={rowData}

pagination={true}

paginationPageSize={10}

animateRows={true}

onGridReady={onGridReady}

/>

</div>

);

};

export default Table;

**4. Add AG Grid Styles**

Ensure the AG Grid styles are imported in your project. You can add them to styles/globals.css:

css

Copy

@import 'ag-grid-community/styles/ag-grid.css';

@import 'ag-grid-community/styles/ag-theme-alpine.css';

**5. Update pages/index.tsx**

Update the pages/index.tsx file to include the Table component:

tsx

Copy

import React from 'react';

import Table from '../components/Table';

const HomePage: React.FC = () => {

return (

<div>

<h1>Dashboard</h1>

<Table />

</div>

);

};

export default HomePage;

**6. Start the Development Server**

You can now start the development server:

bash

Copy

npm run dev

Your project should now be running on http://localhost:3000/. The table will display correctly with sorting, filtering, and pagination.

**Summary**

* Removed Angular-specific files.
* Installed ag-grid-react and ag-grid-community for table functionality.
* Created a Table component using AG Grid for React.
* Added AG Grid styles to styles/globals.css.
* Updated pages/index.tsx to include the Table component.

This should give you a fully functional React Next.js project with a responsive and interactive table. You can now customize the table further or add more components as needed.