Next.js

Version History

S. No	Revision Date	Version	Modifications	Modified By
1	03/10/2023	V1.0	Initially Created the Nest.js Document	Pradeepa

Table of Contents

What is Next JS ?

Next.js is a popular open-source React framework that is used for building modern web applications. It is designed to make the process of building React applications more streamlined and empowers developers to build scalable, fast, and maintainable web applications.

PURPOSE OF USING NEXT JS :

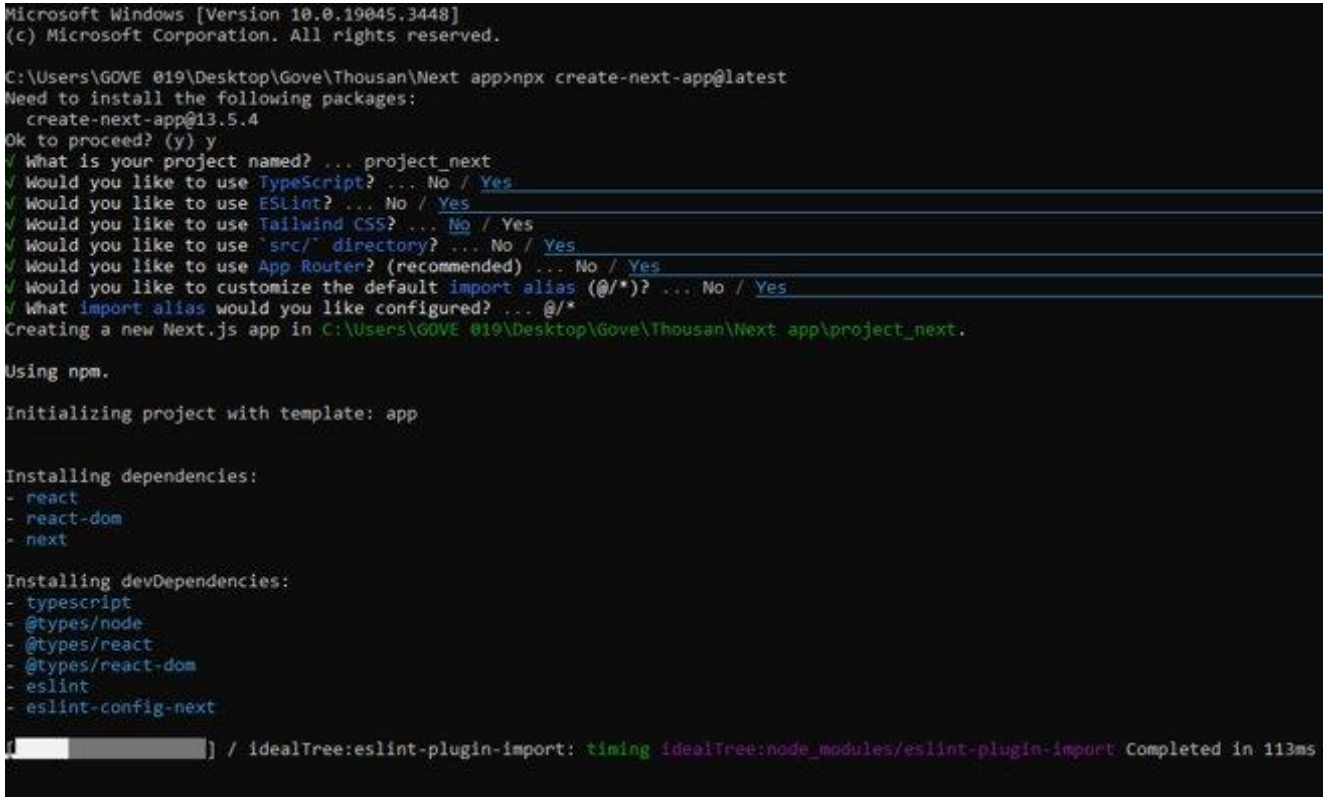
- Server-Side Rendering (SSR)
- Static Site Generation (SSG)
- Simplified Routing
- TypeScript Support
- Page based routing

KEY FEATURES : [WE USE]

- Server Side Rendering
- Routing

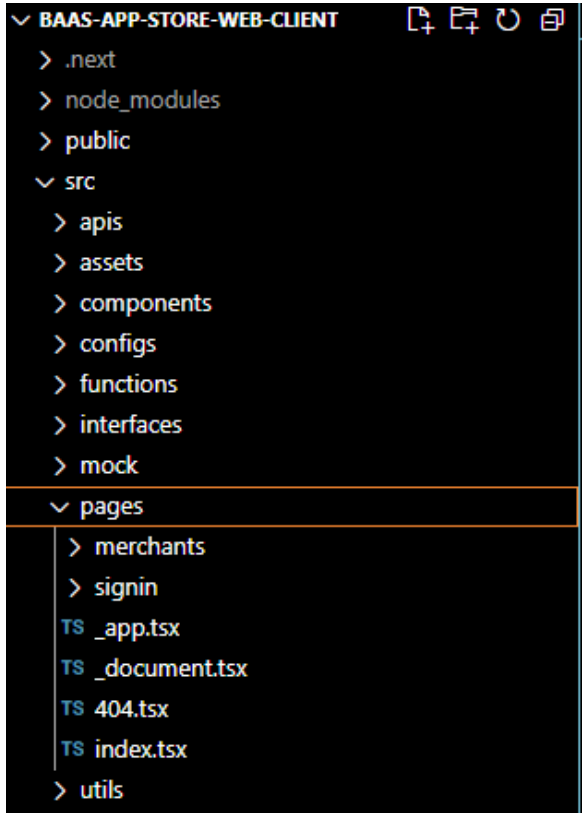
INSTALLATION GUIDE :

-- Using the command `npx create-next-app@latest` we can create our next app



PAGE BASED ROUTING :

Page-based routing is a key feature of Next.js, and it simplifies the organization and structure of your application by associating pages with specific route paths



PAGE DIRECTORY :

- In a Next.js project, the `pages` directory is a special directory where you organize your React components to represent different pages of your application.
- Each `.js`, `.jsx`, `.ts`, or `.tsx` file inside the `pages` directory becomes a route in your application.

DEFAULT ROUTE :

- The `index.tsx` file inside a directory represents the default page for that directory
- Example : pages/index.tsx or page.tsx [as mentioned here]

_app.tsx :

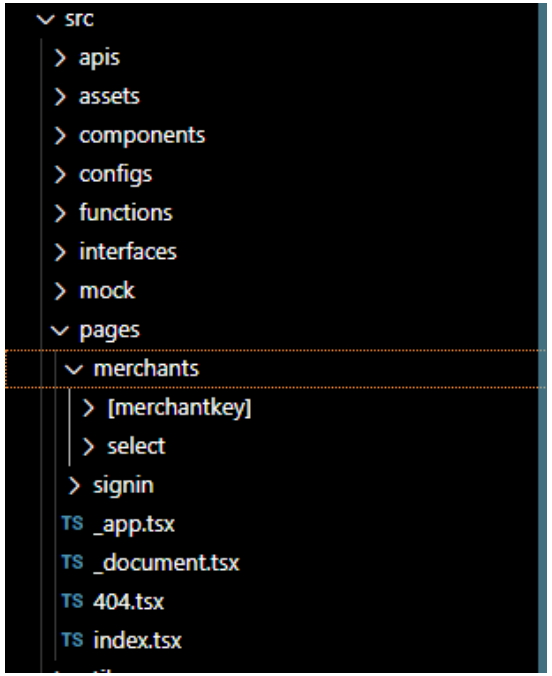
In a Next.js project, the _app.tsx file is a special file used to customize the default behavior of the App component. This file is located in the pages directory and is typically named _app.js or _app.tsx.

FILE NAMING AND ROUTING :

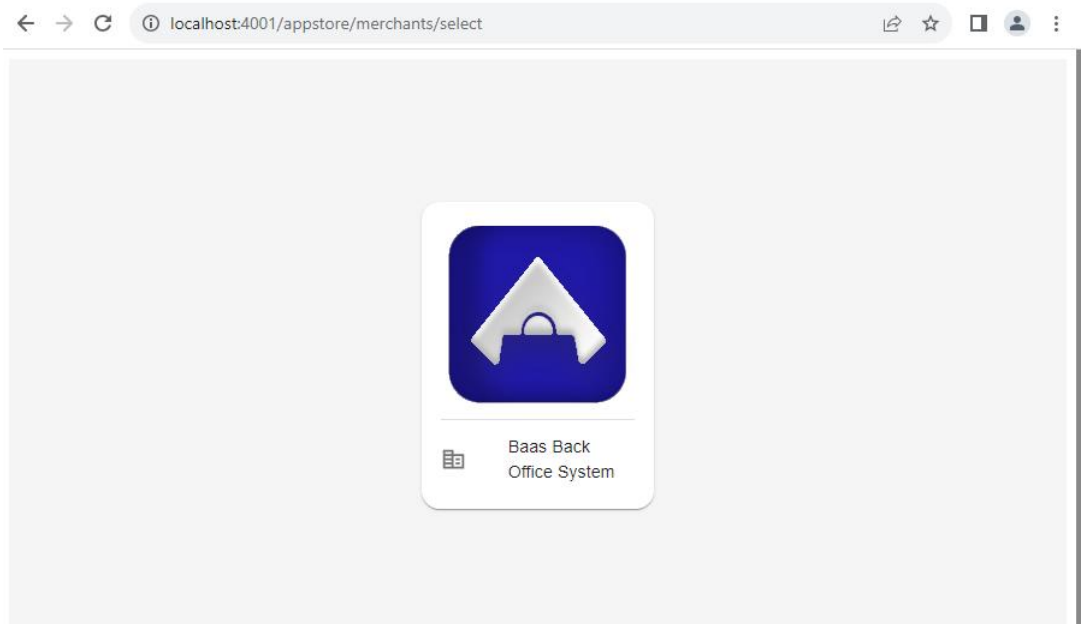
- The file name determines the route path.
- For example, `index.tsx` in the `pages` directory corresponds to the root route (`/`), and `about.js` corresponds to the `/about`

Example : here merchants/select is page and the folder structure also look like that

Page folder



Page Route



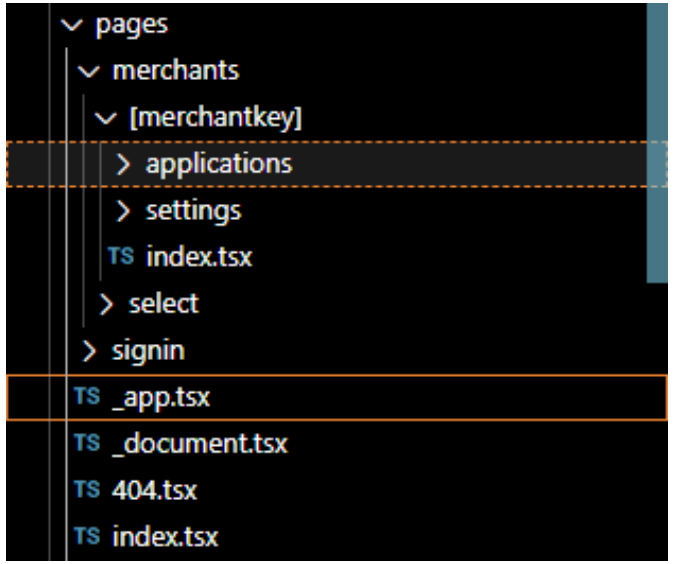
DYNAMIC ROUTES :

- Dynamic routes in Next.js allow you to create pages with variable parts in the URL.
- These variable parts are denoted by square brackets **[]** in the file name

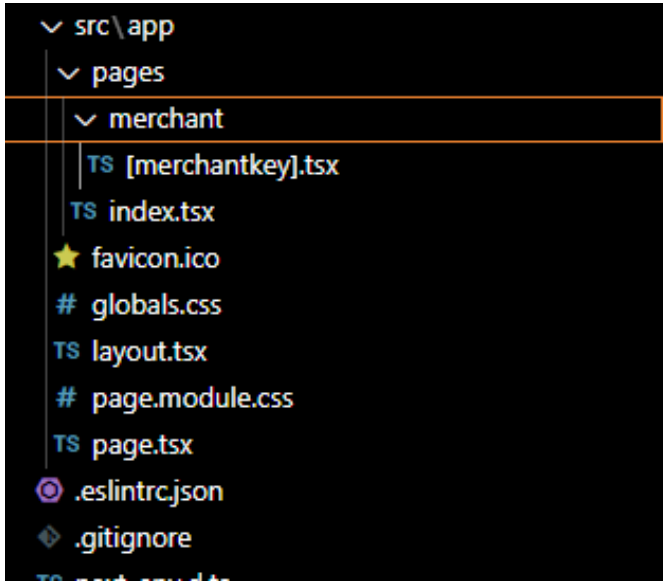
Example :

- Folder structure - pages/merchants/[merchantkey]/application/[applicationid]/settings/connectors
- Dynamic Url : /merchants/BBOS-MERCHANTKEY/applications/31/settings/connectors

Dynamic Page with child



Dynamic Page without child



PROGRAMMATIC NAVIGATION :

You can use the `useRouter` hook from `next/router` to perform programmatic navigation within your components

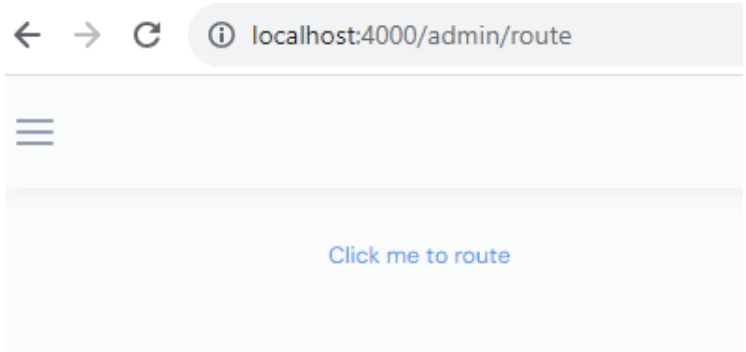
src > pages > route > TS index.tsx > ...

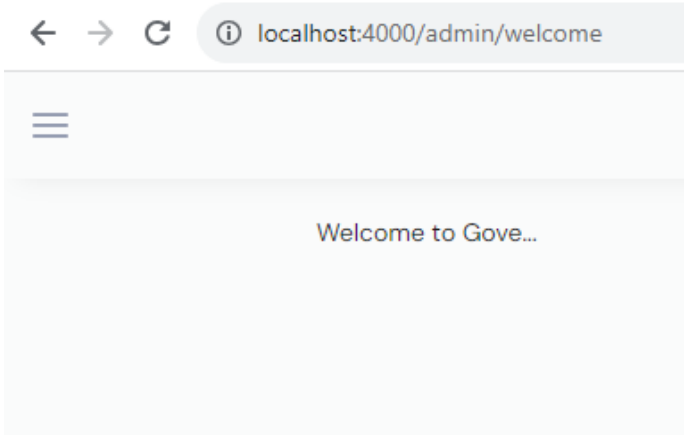
```tsx
import React from 'react'
import Button from '@mui/material/Button'
import { useRouter } from 'next/router'

export default function Route() {
    const router = useRouter()
    const handleroute = () => {
        router.push('/welcome')
    }
    return (
        <>
            <Button onClick={() => {handleroute()}}>
                Click me to route
            </Button>
        </>
    )
}
```

This component reroute the page to this component

src > pages > welcome > TS index.tsx > Welcome

```tsx
import React from 'react';

export default function Welcome() {
    return (
        <>
            Welcome to Gove...
        </>
    )
}
```





SERVER SIDE RENDERING :

- The server refers to the computer in a data center that stores your application code, receives requests from a client, and sends back an appropriate response.
- Server-Side Rendering (SSR) is a technique used in web development to render web pages on the server before sending them to the client's browser. In the context of Next.js, SSR is a prominent feature that provides several benefits, including improved performance, search engine optimization (SEO), and a better user experience

RENDERING PROCESS :

- When a user makes a request to a page in a Next.js application, the server renders the React components for that page on the server.
- This process involves executing the React component code on the server, fetching any required data, and generating the HTML markup for the page.

IMPLEMENTING IN NEXT JS :

```
export async function getServerSideProps(context) {
  // ...

  return {
    props: {
      // Data to be passed as props
    },
    // Set fallback to true or false
    // ...
  };
}
```

- To enable Server-Side Rendering for a page, you can export an async function called getServerSideProps in your page component.
- This function runs on the server and can fetch data, which will be passed as props to the React component

```
export const getServerSideProps = async (context: any) => {

  if (!platformHelper.checkUserCookieStorage(context)) {
    return {
      redirect: {
        destination: "/signin",
        permanent: false,
      },
    };
  }

  let globalSettingsMenuList = await globalSettingsFunction.readSchemaByModuleCode(constants?.GLOBAL_SETTINGS_MODULE_CODE);
  let applicationType = await applicationTypesFunction?.readApplicationTypesSSR({
    filter: { IsDeleted: false }, sort: { ApplicationTypeID: "desc" }
  })
  return {
    props: {
      globalSettingsMenuList: globalSettingsMenuList,
      applicationTypes: applicationType
    }
  }
}
```

CLIENT-SIDE RENDERING :

- The client refers to the browser on a user's device that sends a request to a server for your application code. It then turns the response from the server into a user interface.
- Client-Side Rendering (CSR) is a web development technique where the initial rendering of a web page is done on the client's browser using JavaScript. In the context of Next.js, Client-Side Rendering is a default behavior for pages

```
// pages/example.js

const ExamplePage = () => {
  return (
    <div>
      <h1>Client-Side Rendered Page</h1>
      {/* Content rendered on the client side */}
    </div>
  );
};

export default ExamplePage;
```