Project Details

Languages and Utilities Used

White Album is built with a powerful combination of **Languages** such as

Typescript

and **Frameworks** such as

React JS

Next JS

Tailwind CSS

This stack ensures that the application is not only efficient and reliable but also elegant and user-friendly. **TypeScript** provides a strong typing system that improves the quality of the code and makes it more maintainable. **Tailwind CSS** offers a comprehensive set of utility classes that make it easy to style components and create beautiful user interfaces.

ReactJS and **Next JS 13 Beta** are two of the most popular frameworks for building web applications, and they provide a robust and flexible foundation for White Album. Together, these technologies make White Album a top-tier application for managing and viewing images in storage servers.

The Backend of this project is done with **Node JS** , powered with **Next JS** to enhance the abilities of react frontend . to an other level of development.

White Album fetches data from **Cloudinary**, a cloud platform for storing images and other graphical media. Cloudinary is the connection used in this project prototype , although the site can be extended to other servers as well.

The **Login Page and its Validation** is done by using **Supabase service.**

Supabase

Supabase is an open source Firebase alternative that provides a backend-as-a-service (BaaS) platform for web and mobile developers. It offers a variety of features, including a PostgreSQL database, authentication, instant APIs, edge functions, real time subscriptions, and storage.

PostgreSQL

Supabase's PostgreSQL database is a powerful and scalable relational database that is used by millions of developers around the world. It supports a wide range of features, including ACID transactions, foreign keys, indexes, and views.

Prisma ORM

Prisma ORM is a powerful tool that allows you to interact with your database using a variety of languages, including JavaScript, Python, Go, and Java. It provides a high-level abstraction over the database, making it easy to write efficient and maintainable code.

Schema.prisma

`schema.prisma` file is a declarative language that is used to define the structure of your database. It is a powerful tool that allows you to define your database schema in a clear and concise way.

Here is a brief overview of the schema.prisma file:

- The `generator client` directive specifies that the Prisma client should be generated for the JavaScript language.
- The `datasource db` directive specifies that the database should be a PostgreSQL database.

- The `url` property specifies the URL of the PostgreSQL database.
- The `model Auth` directive defines a model called `Auth`.
  - The `id` property is the primary key of the model.
  - The `username` property is a unique string property.
  - The `password` property is a string property.

Benefits of using Supabase

- It is open source and free to use.
- It is easy to use and configure.
- It is scalable and reliable.
- It has a wide range of features.
- It is well-documented and supported.

Structure of the Project

Next.js 13 beta app based routing allows for dynamic routing based on the file system structure. This means that each subfolder in the `app` folder corresponds to a page in the application, and the URL reflects the file system structure. For instance, the `login` page can be accessed at the URL `/login`.

The directory structure for a Next.js 13 beta app based routing is designed to help organize the source code of the application in a more intuitive and easy-to-understand way. The `src` folder is the root of the application and contains all of the source code for the app, including the `app` folder. Inside the `app` folder, there are subfolders for each page of the app, including `login`, `home`, `gallery`, `slideshow`, and `upload`. These subfolders contain the respective page's components, styles, and logic.

Directory Structure

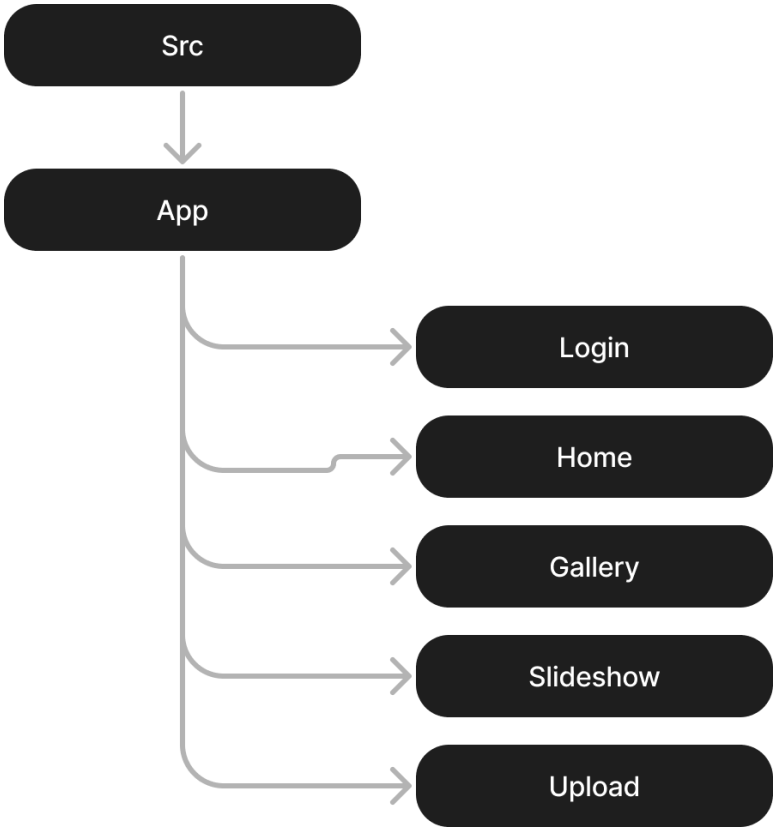


Fig . Directory Structure

The directory structure shown in the picture is an example of the directory structure that can be used in a Next.js 13 beta app based routing. This directory structure is designed to help organize the source code of the application in a more intuitive and easy-to-understand way.

The `src` folder is the root of the application and contains all of the source code for the app. It is where the `app` folder, which contains all of the pages, components, and styles of the app, is located.

Inside the `app` folder, there are subfolders for each page of the app, including `login`, `home`, `gallery`, `slideshow`, and `upload`. These subfolders contain the respective page's components, styles, and logic.

The use of subfolders for each page of the app makes it easier to organize and find code related to a specific page. For example, if you are working on the `login` page, you can find all of the relevant code in the `login` subfolder of the `app` folder.

The Next.js 13 beta app based routing allows for dynamic routing based on the file system structure. This means that each subfolder in the `app` folder will correspond to

a page in the app, and the URL will reflect the file system structure. For example, the `login` page can be accessed at the URL `/login`.

The directory structure shown in the picture is just one example of how you can structure your app's source code. However, it is a good starting point for organizing your code in a way that makes it easy to understand and maintain.

Layouts and Pages

Next.js 13 beta /app based routing is a powerful tool that allows for dynamic routing based on the file system structure. One of the key features of this framework is its ability to separate the layout and content of each web page into separate files for easier management and organization of code.

Each web page in Next.js 13 beta /app based routing has its own `Layout.tsx` file for the layout of the page and a `Page.tsx` file that contains the main content. The `Layout.tsx` file is used to define the overall layout of the page, including the header, footer, and other common elements. The `Page.tsx` file is used to define the specific content of the page, such as text, images, and other components.

By separating the layout and content of a page into separate files, Next.js 13 beta /app based routing makes it easier to manage and organize the code for a web application. It also makes it easier to reuse common elements across multiple pages, as the layout can be defined in a single file and reused as needed.

The `Layout.tsx` file is typically stored in the `app/layouts` folder, while the `Page.tsx` file is stored in the respective page's folder. For example, the `Layout.tsx` file for the `login` page would be stored as `app/login/layout.tsx` , while the `Page.tsx` file for the `login` page would be stored in `app/login`.

The `Layout.tsx` file usually contains the following components:

- **Header:** Contains the logo, navigation menu, and other common elements that are displayed at the top of the page.
- **Main:** Contains the main content of the page, such as text, images, and other components.
- **Footer:** Contains the copyright notice, links to social media, and other common elements that are displayed at the bottom of the page.

The `Page.tsx` file contains the specific content of the page. This includes the following:

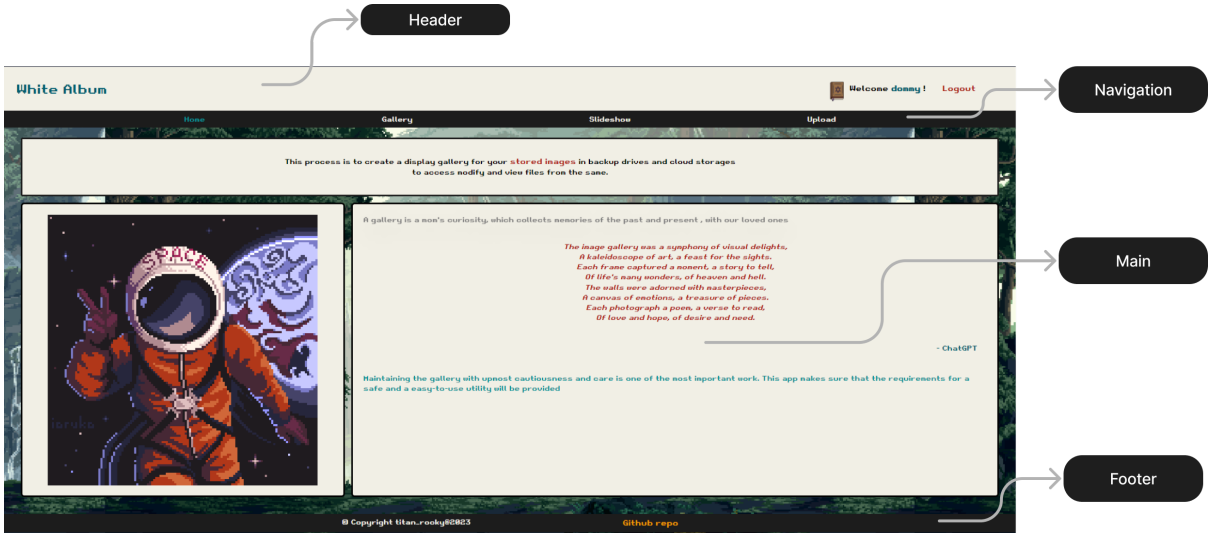
- **Title:** The title of the page, which is displayed in the browser's tab.
- **Meta Data:** Metadata about the page, such as the description and keywords.
- **Content:** The main content of the page, which can include text, images, and other components.

In addition to organizing the code for each web page, Next.js 13 beta /app based routing also allows for easy reuse of common elements across multiple pages. This is because the layout can be defined in a single file and reused as needed. For example, if you have a common header and footer that you want to use across multiple pages, you can define them in the `Layout.tsx` file and import them into the respective `Page.tsx` files.

Overall, Next.js 13 beta /app based routing is a powerful tool for building web applications with dynamic routing. Its ability to separate the layout and content of each web page into separate files makes it easier to manage and organize code, while also allowing for easy reuse of common elements across multiple pages.

Structure of a Page

A **Page** usually consists of a **Header** , **Footer** , **Navigation Bar** , and a **Main Section**. This project follows the same semantics as per the convention so that it will be useful during the **Search Engine Optimization (SEO)**.



The page also follows **Responsive Web Design** to ensure that the webpage does not get affected due to changing **Resolution** , thus supporting them. However this website is **Not Supported** for **Mobile Phones**