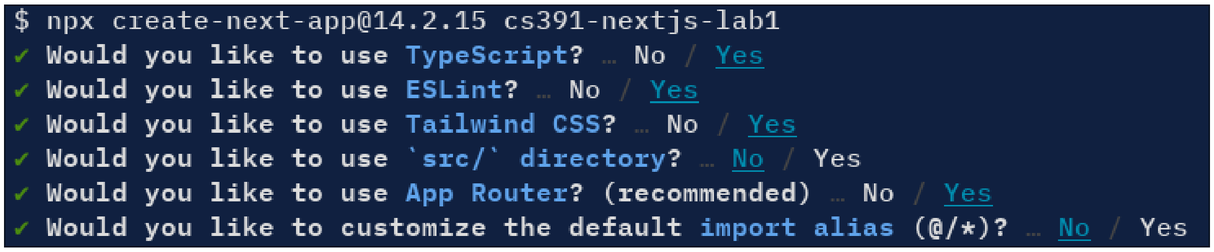
Next.js: Pages and Static Routing

In this discussion we will create the foundations of a simple message board using Next.js by learning about routing and project structure in Next.js

Note: This project was styled using tailwindcss. If you prefer to not use tailwind, feel free to use styled-components or anything else you may be more comfortable with.

Task 1: Initialize the Next.js project

1. In your terminal, navigate to the location you would like to initialize a new project (e.g. **/cs391/labs**)
2. Use the command **npx create-next-app@14.2.15** ***project-name*** to create a new Next.js project with the name of your choice. Select the following

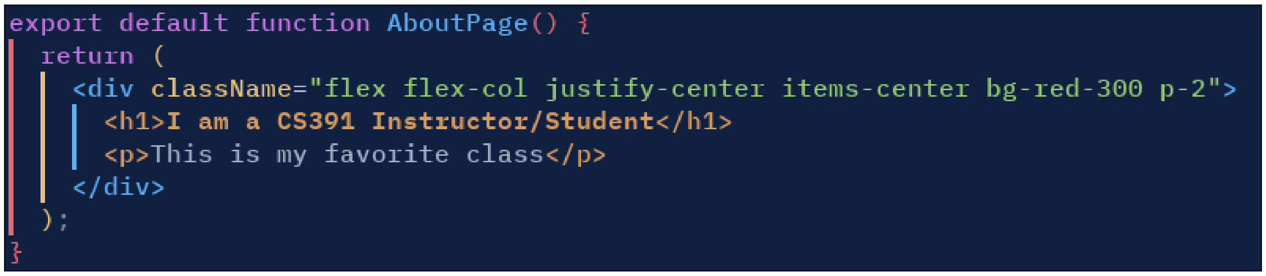


```
$ npx create-next-app@14.2.15 cs391-nextjs-lab1
✔ Would you like to use TypeScript? … No / [✓ Yes]
✔ Would you like to use ESLint? … No / [✓ Yes]
✔ Would you like to use Tailwind CSS? … No / [✓ Yes]
✔ Would you like to use `src/` directory? … [✓ No] / Yes
✔ Would you like to use App Router? (recommended) … No / [✓ Yes]
✔ Would you like to customize the default import alias (@/*)? … [✓ No] / Yes
```

3. Enter the root of your project using the command **cd** ***project-name***
4. Use the command **npm install** to install any necessary dependencies
5. You can now use the command **npm run dev** to see the default state of the project at http://localhost:3000
6. This project should automatically hot-reload when you make changes, so you don't need to constantly re-run the project

Task 2: Create an about page

1. Create a directory with the path **/app/about**
2. Inside that directory, create a file called **page.tsx**
3. Now, we will create the content that will be displayed on this page

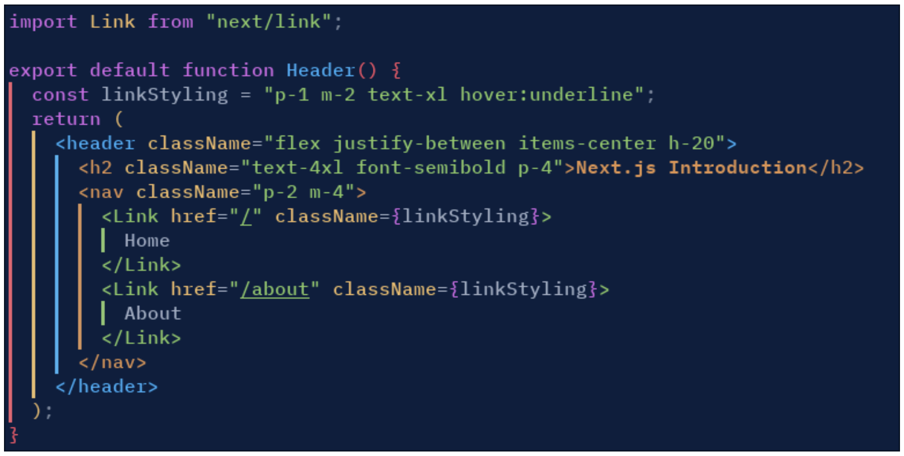


```
export default function AboutPage() {
  return (
    <div className="flex flex-col justify-center items-center bg-red-300 p-2">
      <h1>I am a CS391 Instructor/Student</h1>
      <p>This is my favorite class</p>
    </div>
  );
}
```

4. You should see the content of the about content when you visit http://localhost:3000/about
   a. Next.js has built-in directory-based routing, so we don't need to setup a router ourselves

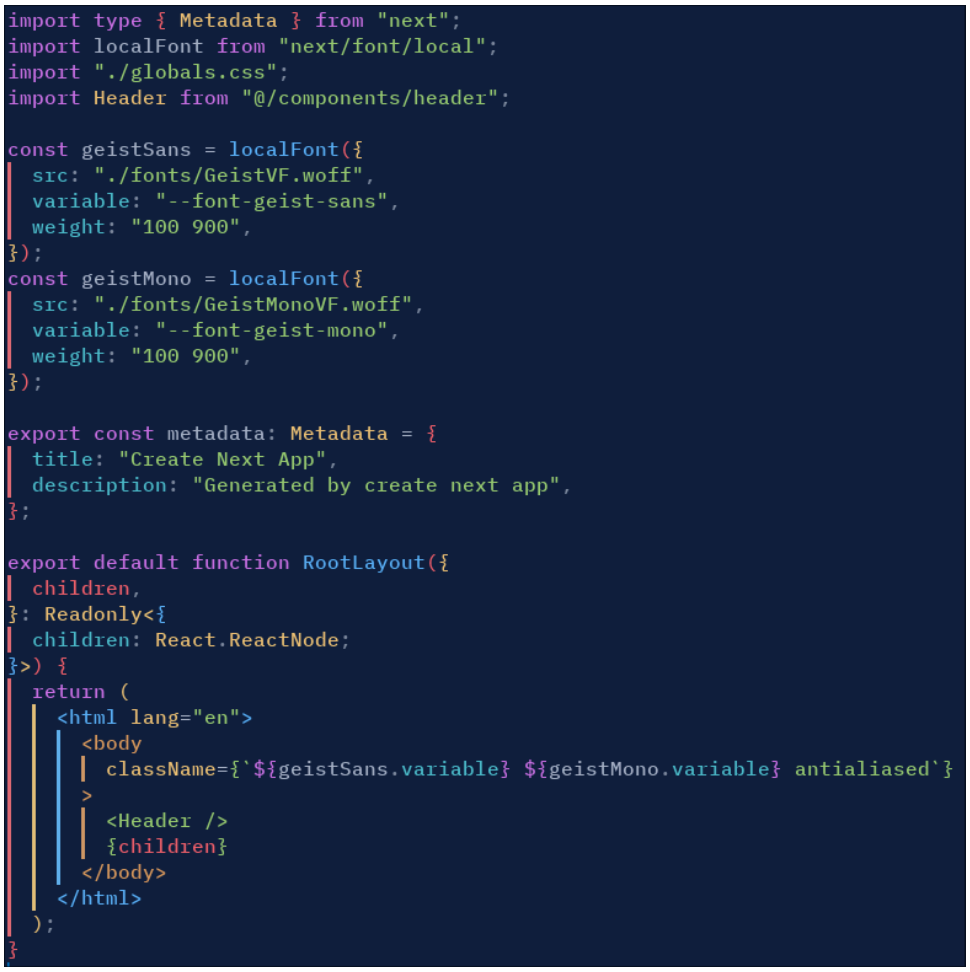
Task 3: Create a header with navigation

1. Create a directory with the path **/components**
2. Inside that directory, create a file called **header.tsx**
3. In this file we will create a header component with navigation



```tsx
import Link from "next/link";

export default function Header() {
  const linkStyling = "p-1 m-2 text-xl hover:underline";
  return (
    <header className="flex justify-between items-center h-20">
      <h2 className="text-4xl font-semibold p-4">Next.js Introduction</h2>
      <nav className="p-2 m-4">
        <Link href="/" className={linkStyling}>
          Home
        </Link>
        <Link href="/about" className={linkStyling}>
          About
        </Link>
      </nav>
    </header>
  );
}
```

4. In order to display this component on all pages, add it to the **app/layout.tsx** file. (Look towards the end of the this file)



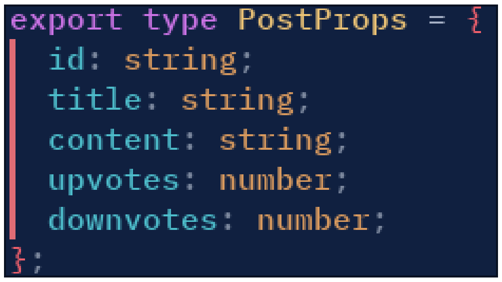
```tsx
import type { Metadata } from "next";
import localFont from "next/font/local";
import "./globals.css";
import Header from "@/components/header";

const geistSans = localFont({
  src: "./fonts/GeistVF.woff",
  variable: "--font-geist-sans",
  weight: "100 900",
});
const geistMono = localFont({
  src: "./fonts/GeistMonoVF.woff",
  variable: "--font-geist-mono",
  weight: "100 900",
});

export const metadata: Metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body
        className={`${geistSans.variable} ${geistMono.variable} antialiased`}
      >
        <Header />
        {children}
      </body>
    </html>
  );
}
```

5. You should now be able to navigate between the home page and about page

6. Feel free to edit the project's metadata as you see fit

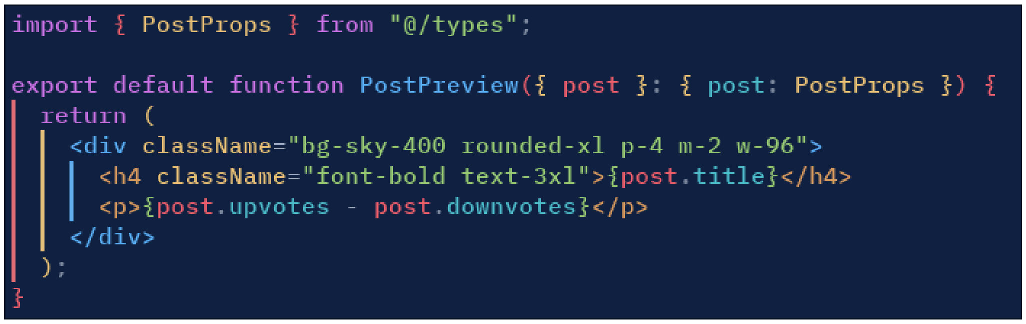
Task 4: Create a type for a message board post and a preview component

1. Create file called **types.ts** at the root of the project and declare the following type



```
export type PostProps = {
  id: string;
  title: string;
  content: string;
  upvotes: number;
  downvotes: number;
};
```

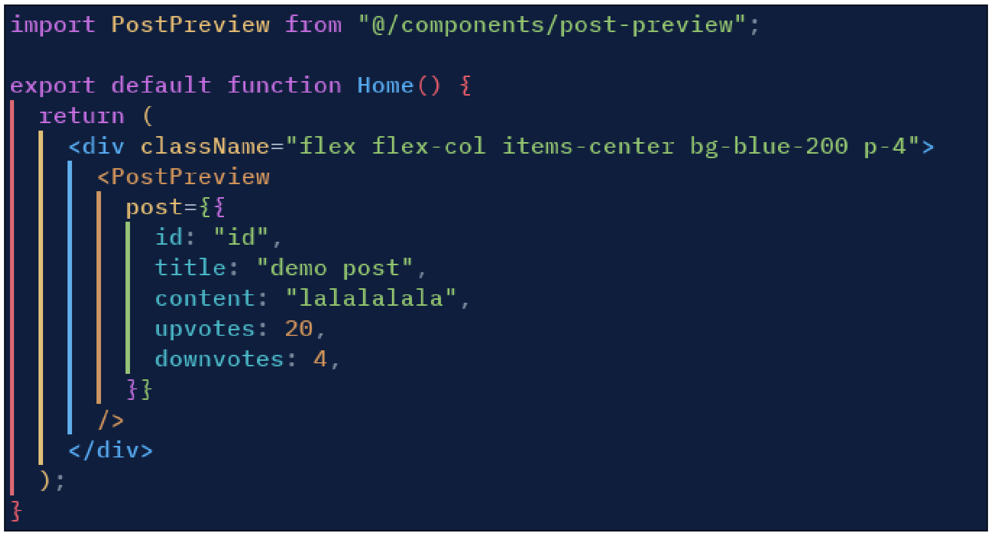
2. Create a file with the page **/components/post-preview.tsx** and create component to display a preview of a post



```
import { PostProps } from "@/types";

export default function PostPreview({ post }: { post: PostProps }) {
  return (
    <div className="bg-sky-400 rounded-xl p-4 m-2 w-96">
      <h4 className="font-bold text-3xl">{post.title}</h4>
      <p>{post.upvotes - post.downvotes}</p>
    </div>
  );
}
```

Task 5: Display a dummy Post preview on the home page

1. Completely overhaul the file **app/page.tsx** to roughly match the following



```
import PostPreview from "@/components/post-preview";

export default function Home() {
  return (
    <div className="flex flex-col items-center bg-blue-200 p-4">
      <PostPreview
        post={{
          id: "id",
          title: "demo post",
          content: "lalalalala",
          upvotes: 20,
          downvotes: 4,
        }}
      />
    </div>
  );
}
```

   a. The home page should no longer show the default Next.js content and should instead display a preview of the dummy post we created
   b. Add as many dummy posts as you'd like!