2795026 Ethan Honis
Developer's Guide

Getting Started

You can visit the full program at climbtrak.com
First, Download the whole project as a zip from https://github.com/ehonis/ClimbTrak

Once installed, open the project in a code editor and open up a terminal.

once in a terminal, run: npm install. This will download all the dependencies for the project.

Note for TA: For this to function, you need the .env file. I will not push this to github for security purposes. The .env file will be in the submission for this project.

once you have the .env and all the dependencies downloaded, move the .env file to the root of the page. This is where schema.prisma is, or anywhere outside of the app directory.

from there, run: npm run dev, then in a browser open up http://localhost:3000

if everything is working correctly, it should up open the page with no errors.

Useful Commands

npm run dev: runs development localhost:3000
npm run build: Builds the project to detect for errors
npx prisma studio: opens up a studio viewer for the database

Basic Information

Next.js requires a very specific layout of files. Inside the app directory is where all the main routing and page logic resides. Each folder within app can represent a route, and any page.js or page.jsx file within a folder defines the content for that route.

1. app **Directory**:
   • This is the core directory for building your app in Next.js. It follows the App Router paradigm, which organizes routes hierarchically.
   • Each folder under app corresponds to a route, and you can include files like page.js for content, layout.js for layouts, and loading.js for loading states.

2. **schema.prisma:**
   • This file defines the database schema for Prisma ORM. Any changes to the schema require running migrations using Prisma CLI.
3. **Static Files**:
   • The public directory is where you store static assets such as images, fonts, or other files. These files are served directly at / and do not require imports.

4. **Global Configurations**:
   • Global styles and configurations (like globals.css) are often located in the app directory or imported into layout.js.
   • Environment variables are managed in the .env file, which should be placed at the root of the project.

5. **API Routes**:
   • Inside the app/api folder, you can define API endpoints. Each file or folder under api becomes an endpoint.

6. **Component Organization**:
   • Reusable components are typically stored in a ui directory, either within the app directory or at the root of the project.
   • This helps separate UI elements from pages and routes for better modularity.

7. **Database and Backend**:
   • This project uses Prisma as the database ORM. Prisma helps interact with the database through type-safe queries.
   • Run npx prisma studio to visualize and edit the database.

8. **Build and Deployment**:
   • Running npm run build ensures the project is production-ready and detects any errors.
   • After building, the project can be deployed to platforms like Vercel, which is optimized for Next.js.

Specific Information

Inside the app directory, you will see many folders. Any folder that is not either API or UI is a page on the website. The folder directly maps to a route, such as the routes folder correlates to the localhost:3000/routes. Whenever there is a [slug] folder inside a regular folder this denotes a dynamic route. In my projects case, [slug] will get the url that the user inputs, such as /routes/{id} and render out the page with that information of the route.

Client and server components are a huge part of Nextjs. I cannot explain them in detail but, for my application, since Prisma cannot be run on the client, it must be sent to an API to run on the server. For any interactivity, the component must be a client component. Most data fetching will be in a server component and then sent down to the client components.

For authorization, Auth.js is used. In our case we are using Github and Google callbacks. So in the server components, a session is acquired with const session = authjs() if no session is present nothing will be returned. This is how we obtain the user information to display different things.