JCU GYM MANAGEMENT

TECHNICAL DOCUMENTATION

Thant Thu Aung and Sein Linn

by group 4

Table of Contents

Frontend Stack

Next.js 15.2.4

We're using Next.js (App Router) as our main framework for building our frontend. It is server-side rendering, dynamic routing, and API routes enabled, which is ideal in a full-stack setup. The App Router delivers better file structure and improved layout handling, which remains modular and scalable.

React 19

React is our UI framework. React 19 provides us with performance improvements and better concurrency support (like improved use hook feature). It's great for creating responsive and reusable components across the app.

TypeScript 5

TypeScript helps us to catch bugs earlier and have a more stable codebase, especially with multiple contributors. Strict typing also improves developer experience while handling large forms, API responses, and components.

Tailwind CSS 3.4.17

We're styling with Tailwind. It's a utility-first CSS library that lets us build consistent and responsive UIs without writing custom CSS for every single thing. It also keeps our files neat and concise.

Radix UI

To make accessible and headless UI components, we're leveraging Radix UI. It comes in handy when we want the behavior and logic of a component (e.g., a modal or dropdown), but then still completely style it with Tailwind.

React Hook Form

The app's all forms are built using React Hook Form. It's lightweight, easy to integrate, and helps in form state management and form validation. It integrates well with uncontrolled inputs and makes things happen quickly.

Zod

Zod is used together with React Hook Form to validate form schemas. It helps us define expected form input shapes and enforce validation rules in a clean and predictable way.

Backend & Database

PostgreSQL (via Neon.tech)

For our database, we're using PostgreSQL on Neon.tech. It's SQL-focused, stable, and fine to use with relational data. Deployment is also easy and scalable with Neon's serverless support.

node-postgres (pg)

We connect to our PostgreSQL database using the pg library. It's a low-level but lightweight PostgreSQL client for Node.js. We wrote custom queries for the majority of our DB operations instead of using an ORM in order to have control over logic and performance.

Next.js API Routes

Backend logic is handled through Next.js API routes. Each endpoint is defined inside the /app/api directory. We're handling things like login, registration, user data fetches, and session management here.

JWT

We use JSON Web Tokens to manage user authentication. After login, a JWT is generated and stored in cookies for stateless session handling.

bcryptjs

Passwords are also hashed with bcryptjs before they are saved into the database. This adds an extra layer of security and ensures plain-text passwords will never be stored.

Key Libraries

Lucide React

For application icons, we use Lucide React. It provides modern and consistent SVG icons which are easy to import and style.

date-fns

All date comparing, formatting, and math is handled using date-fns. It's lean and gets along with TypeScript nicely.

react-day-picker

We use react-day-picker as our date chooser calendar component (e.g., scheduling, date filtering). It's accessible, very customizable, and works great with Tailwind.

Summary

The technology stack was chosen to offer a balance between performance, developer experience, and scalability. With the use of the latest technology like Next.js App Router, TypeScript, and PostgreSQL, we were able to develop a full-stack application that is easy to maintain and quick to run.