MANIPAL UNIVERSITY
JAIPUR

Minor Project

Title - FitGamify

MCA - III Sem

Subject Code: CA7131

Submitted By

Student Name – Piyush Dixit

Student Registration – 2210020053

Student Name - Priyam

Student Registration - 221020039

Faculty Coordinator

Dr. Linesh Raja, Associate Professor

DEPARTMENT OF COMPUTER APPLICATIONS

Declaration

We, Piyush Dixit and Priyam, solemnly declare that the work presented in this project report titled "FitGamify" is entirely our own effort. We confirm that this work has not been submitted previously for any examination, nor has it been used to fulfill the requirements of any other course or degree. All sources of information used in this project have been duly acknowledged and referenced. Any assistance received in the preparation of this project, whether technical or intellectual, has been appropriately acknowledged.

Signature: Piyush Dixit, Priyam

Date: 16 December 2023

Acknowledgment

We extend my sincere gratitude to Dr. Linesh Raja for their invaluable guidance, support, and encouragement throughout the development of this project. Their expertise and feedback played a pivotal role in shaping this endeavor. We would like to express my appreciation to Dr. Linesh Raja for providing the necessary resources, access to facilities, and a conducive environment that facilitated the execution of this project. We are grateful to our friends and family for their patience, understanding, and unwavering support during this academic pursuit.

Signature: Piyush Dixit, Priyam

Date: 12 December 2023

Table of Contents

Abstract

In today's fast-paced world, maintaining a healthy lifestyle can be a challenge. FitGamify emerges as a groundbreaking solution, bridging the gap between fitness and entertainment. This innovative platform is designed to transform the way we approach exercise, creating a unique and engaging experience for users of all fitness levels.

FitGamify is more than just another fitness app; it's a complete fitness ecosystem that leverages the power of technology and human nature's innate competitive spirit. As users log in, they are greeted with a dynamic and user-friendly interface that presents them with a personalized workout plan. Whether you're a fitness enthusiast or just starting on your journey, FitGamify caters to your needs by offering exercises tailored to your goals.

At the heart of FitGamify is a point-based scoring system that gamifies the fitness journey. By completing exercises and achieving daily goals, users earn points. These points not only motivate users to stay consistent with their workout routines but also unlock prestigious "Fire Badges," displayed on their profiles as a testament to their dedication.

But FitGamify doesn't stop at individual accomplishments; it thrives on the power of community and competition. Users can join groups, competing with friends and fellow fitness enthusiasts. At the end of each week, the top performers are rewarded with gold, silver, and bronze badges, creating an atmosphere of healthy rivalry and mutual encouragement.

The FitGamify project envisions an ever-evolving fitness platform that caters to real-world variables and the introduction of admin features for professional trainers. It promises to be more than just an app but a lifestyle-changing experience that combines the physical and virtual worlds to create a healthier, more motivated you. Join us on this exciting fitness journey with FitGamify and let's revolutionize the way we stay active and healthy.

Introduction

Introducing FitGamify: Revolutionizing Fitness with Gamification

In today's fast-paced world, maintaining a healthy lifestyle can be a challenge. FitGamify emerges as a groundbreaking solution, bridging the gap between fitness and entertainment. This innovative platform is designed to transform the way we approach exercise, creating a unique and engaging experience for users of all fitness levels.

Technology Stack:

FitGamify is built on a robust technological foundation, employing a diverse set of technologies to ensure a seamless and feature-rich experience. The project utilizes a range of languages such as HTML, CSS, JavaScript/TypeScript, and JSX/TSX, along with SQL for database interactions. Prettier and ESLint contribute to code formatting and linting, ensuring clean and consistent code. The runtime environment is powered by Node.js, and NPM serves as the package repository.

The user interface is crafted using React, and the React framework, both on the client and server side, is implemented through Next.js. TailwindCSS provides the CSS framework, while the shadcn/ui component library enhances UI elements. NextAuth.js/Auth.js handles authentication, Drizzle serves as the Object-Relational Mapping (ORM) tool, and Zod validates the schema. Git on GitHub manages version control, and Vercel, with its edge network and serverless functions, facilitates seamless deployment. The database is powered by Turso/libSQL, utilizing SQLite.

Fitness Gamification:

FitGamify introduces a revolutionary point-based scoring system that gamifies the fitness journey. Users receive 100 points for completing their daily workout plan, promoting consistency and motivation. The platform's gamification extends beyond individual achievements to foster community and healthy competition. Users form groups, competing for gold, silver, and bronze badges based on weekly point standings, creating a supportive yet competitive environment.

Future Development:

FitGamify aims to continually evolve, incorporating real-world variables into the scoring system and exploring the addition of admin features for professional trainers. The project envisions personalized workout plans for users, making fitness more accessible and tailored. Historical records of badges and user achievements, visible on profiles, ensure a sense of accomplishment and motivation.

In its preliminary phase, FitGamify stands as a promising fitness platform that seamlessly blends the virtual and real worlds. With its potential for growth and new releases, FitGamify is not just an app but a lifestyle-changing experience. Join us on this exciting fitness journey and let's revolutionize the way we stay active and healthy.

Requirement Analysis

1. Problem Definition

Fitness has been a rising talk specially after COVID-19 but there has been a slight issue with consistency in doing so. The apps existing talks just about the numbers of how much of a particular exercise you must do but apparently failing in motivating the users to stay consistent with them. The problem with consistency required to be addressed so seeing the fascination of people with gaming and competitiveness and urge to succeed others or to win could have been attached with the former problem of inconsistency. When fitness is attached with game it gives user that competitiveness which keep them on track to push themselves daily in achieving a goal which is of higher rewards than what their friends have reached.

2. Drawbacks in previous system

1. Limited Personalization:

Fitness apps often offer one-size-fits-all workout plans that may not align with individual goals and fitness levels, lacking the necessary customization.

2. Accountability Challenges:

The absence of personal trainers or workout partners in fitness apps can result in a lack of accountability, potentially leading to decreased adherence to exercise routines.

3. Monotony and Boredom:

Despite gamification features, certain users may find fitness apps monotonous over time due to limited exercise variety or repetitive routines.

4. Data Privacy Worries:

Collection of personal and health-related data by fitness apps raises concerns about data privacy and security, potentially discouraging users who are uncertain about data handling practices.

5. Social Isolation:

While some apps encourage social interaction, there is a risk of promoting solitary fitness routines, impacting motivation for individuals who thrive on group dynamics.

6. Not a Substitute for Professionals:

Fitness apps are valuable guidance tools but cannot replace personalized advice from fitness professionals, particularly for individuals with specific health conditions or those requiring customized programs.

7. Adherence Challenges:

Sustaining long-term engagement with fitness apps can be difficult, as users may lose interest, forget to use the app regularly, or struggle to incorporate exercise into their daily routines.

Methodology

1. Technologies used:-

1. Languages - html, css, js/typescript, jsx/tsx, SQL
2. Formatter - Prettier
3. Linter - ESLint
4. Runtime - Node.js
5. Package Repository - NPM
6. UI Library - React
7. React Framework(Client and Server Side) - Next.js
8. CSS Framework – TailwindCSS
9. Component Library – shadcn/ui
10. Authentication Provider - NextAuth.js / Auth.js
11. ORM(Object-Relational Mapping) Tool - Drizzle
12. Schema Validator - Zod
13. Version Control - Git (GitHub)
14. Deployment - Vercel (Edge network with Serverless functions)
15. DataBase Provider – Turso/libSQL(SQLite)

2. Procedure and implication

We have made a website deployed on the edge. To manage the different versions of our website and to enable a better multi-developer experience, we use GitHub and Git as our Software Version Manager. The website uses the latest standards and aims to provide a good developer experience to future developers as well being user friendly and effective for the customer.

Our website runs on a Node.js runtime. We choose to use Node because it comes with the large list of packages and tools which other developers have made to make repetitive tasks easier. Another reason to use node is the fact that our production deployment also uses V8 Javascript engine based node runtimes.

We deploy our website by using the services of Vercel. Vercel is a company which offers edge based rutimes to deploy websites and apps. Vercel is also the creator of many open source projects like Next.js itself. We have chosen to deploy our website on the edge as it eliminates the need of constantly running a server which can end

up being very costly. In the Edge Runtime model, when a client sends a request to our website, our code is sent to a Vercel Node Runtime server which is hosted close where the request originated from, leading to faster response times. The code is then compiled on this runtime and this Node.js instance temporarily acts as our server.

The next decision which was to be made was the UI library which we shall use. React is currently the most popular UI library for Javascript in the world. But React in the modern world faced the issue of the waterfall model of data fetching. Since React was a client-sided UI Library, our server needed to send a file instructing our client to first fetch the React Library after which there would be subsequent fetches which would further lead to an unpleasant experience. Since React was only client-sided and the modern web development paradigm usually requires the need of a server, a developer would need to host a server written in an entirely differently language with it's own syntax. This led to the creation of JSON API. Even after this, there was a privacy and security risk in letting React be our primary framework. Since data was only evaluated on the client, we had to use forced layers of extra encryption and function calls both at our client and then at our server as well (eg: form validation).

To solve this problem, we are using Next.js v14+ which uses the React Server Component Model. In this model, react can now be compiled and ran on the server. This is a major shift in paradigms as it allows as to introduce concepts like Server Side Rendering and Hydration methodologies.

When a client now makes a request to our server, our server evaluates out React code and then sends the compiled output to the client which the client can render directly. This eliminates the multiple network calls the client would have made in the previous model. Since other infrastructure like the database is also usually situated close to the server runtime, it makes our React compilation step much faster.

Yet, there are times when certain code must run on the client. Cases which handle the state of the user and their session need to be dealt with on the client only. Thus, the React Server Component Model allows for declaring certain components to be client-rendered. In such cases, majority of our codebase will elect to still compile at the server and only the JS necessary to maintain interactivity will not be compiled at the server.

As a developer, we must still realise that there will be times when the functionality of a server is needed alongside a client sided feature (think getting information from a database to render certain data but the data is reliant on the state of the client). For such cases, we have implemented Server Actions, which are essentially JS functions which are run on the server but can be called from our client. This is possible because the RSC model understands these situations and automatically prepares API endpoints.

We use typescript in our website which is a superset of javascript. Javascript is a powerful language but it lacks any type definitions which can lead to unforeseen errors and a much worse developer experience. Typescript on the other hand implements types, which let us avoid many of these pitfalls. The aim of typescript is to ne typesafe.

To be fully type-safe, we need to make sure that data transfer between our server and our database is also type-safe. To implement this we use Drizzle which is an Object Relational Model and Zod which is a library which checks for type-safety on data which Typescript doesn't have access to pre-runtime. Drizzle creates Typescript-compatible datatypes and schemas which provides us a layer of abstraction over complex and unsafe database calls.

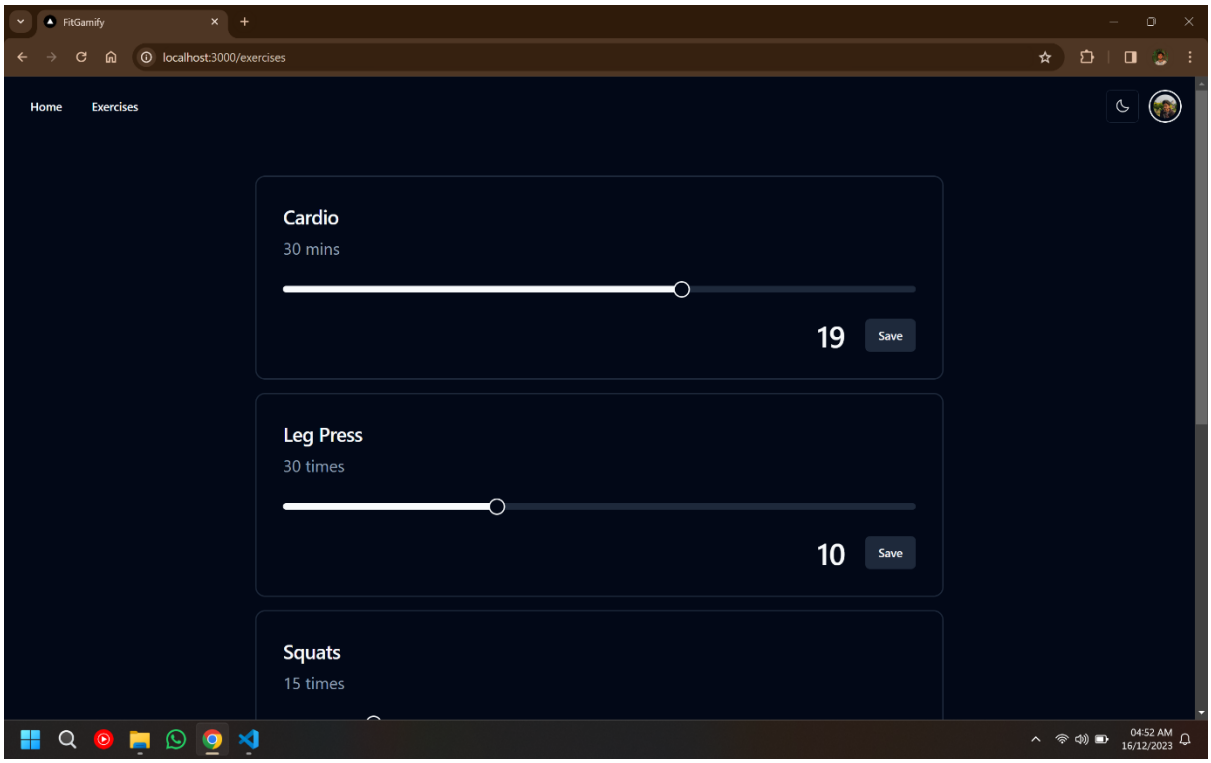
Talking about the database, we choose to utilise a SQL based database. It is a proven standard in the industry and lets us create relationships between our data types and tables. Other implementations of databases which avoid using SQL usually end up facing pitfalls of document based file indexing and Object Oriented Models. So, we decide to use the services of Turso, which is company providing us with edge hosted database instances. The database uses libSQL which is an open source fork of SQLite. By using SQLite, we enable out database to also be deployed on the edge and makes debugging using a development only file based database possible.

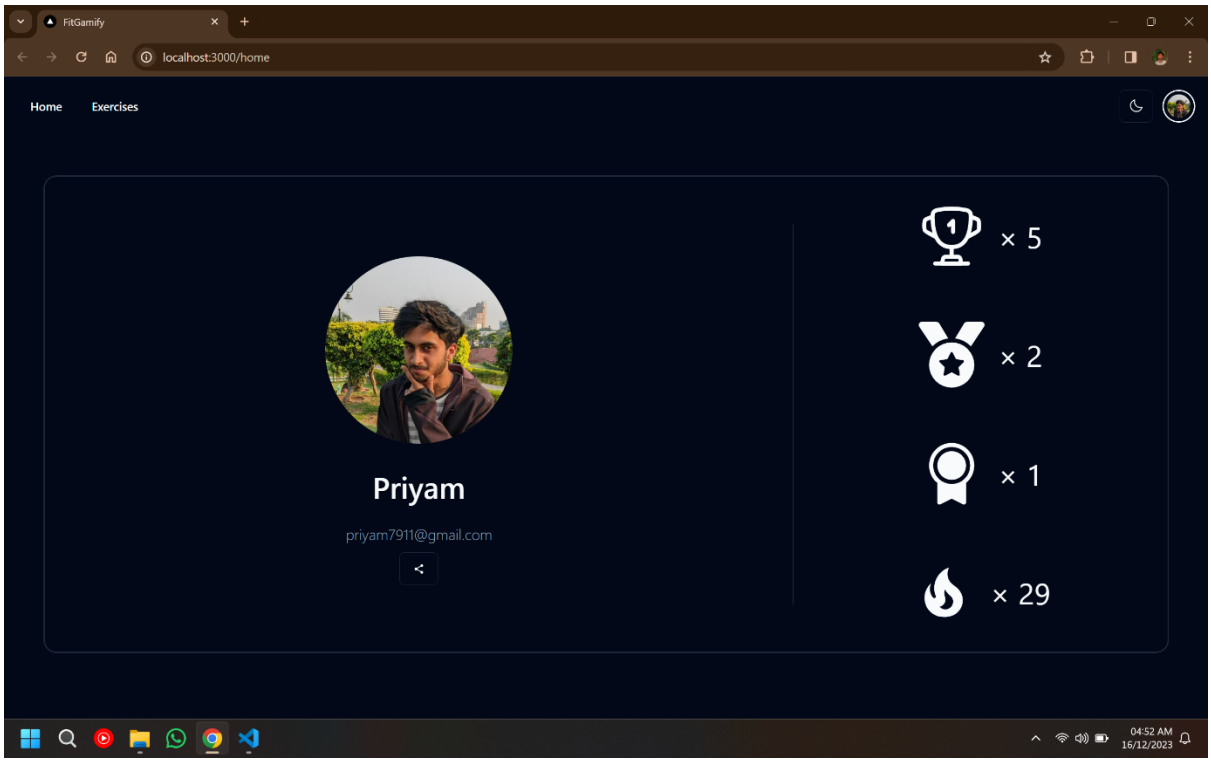
Next we must think about user authentication. The modern web has shown that simple passwords are not secure enough and leave much to be desired. So instead we turn towards implementations like OAuth which uses the APIs of well established software service providers like Google and Github which then take over the responsibility of authentication using technologies like passkeys and Multi Factor Authentication. Even then, implementing methods to communicate with the various standards like OIDC and custom implementations which a provider might have is a very complicated and tedious process. So instead of taking a risk with the security and privacy of our users, we elect to use the open source NextAuth.js Library to implement authentication for our website. This ensures that we have proper encryption standards and a middleware to authorise users on proper routes.

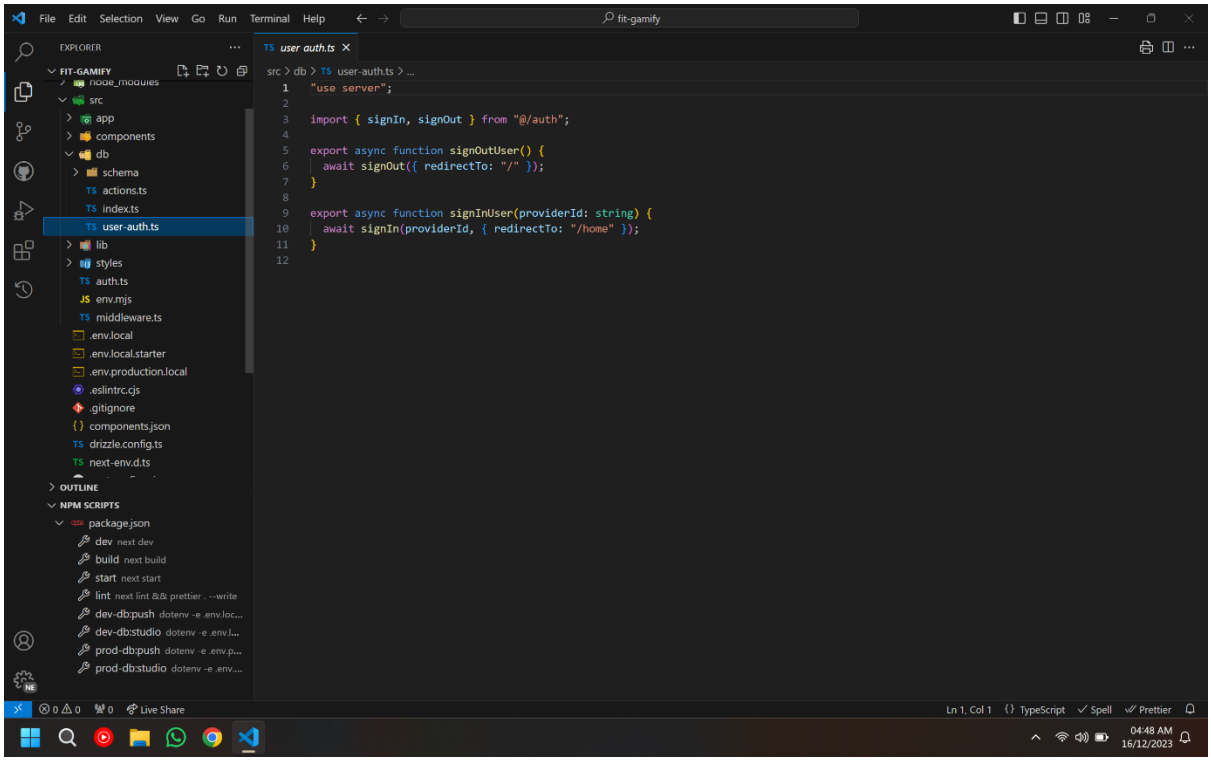
Finally we must talk about the UI itself. CSS is a powerful tool for writing styles on the web but when dealing with large projects, especially ones which use UI libraries like React or Angular, CSS can lead to unforeseen problems. The specificity order in which stylesheets are applied on a component can often become ambiguous and confusing as a developer. Even worse, a change in a CSS class for a different page can lead to changes on another page which also happens to use the same CSS class name. To avoid these pitfalls, we uses TailwindCSS and a component library like RadixUI, shadcn/ui and React Aria by Adobe, all which are open source. Component libraries provide us with pre-styled components with proper accessibility support for commonly used components like buttons and headings. TaillwindCSS offers us simple css classes which it applies to our components on runtime. TailwindCSS allows us to write simple names like m-4 for margin level 4, styles which are reusable across pages and projects. This makes our CSS classes be collated with our React component, making debugging and styling much easier.

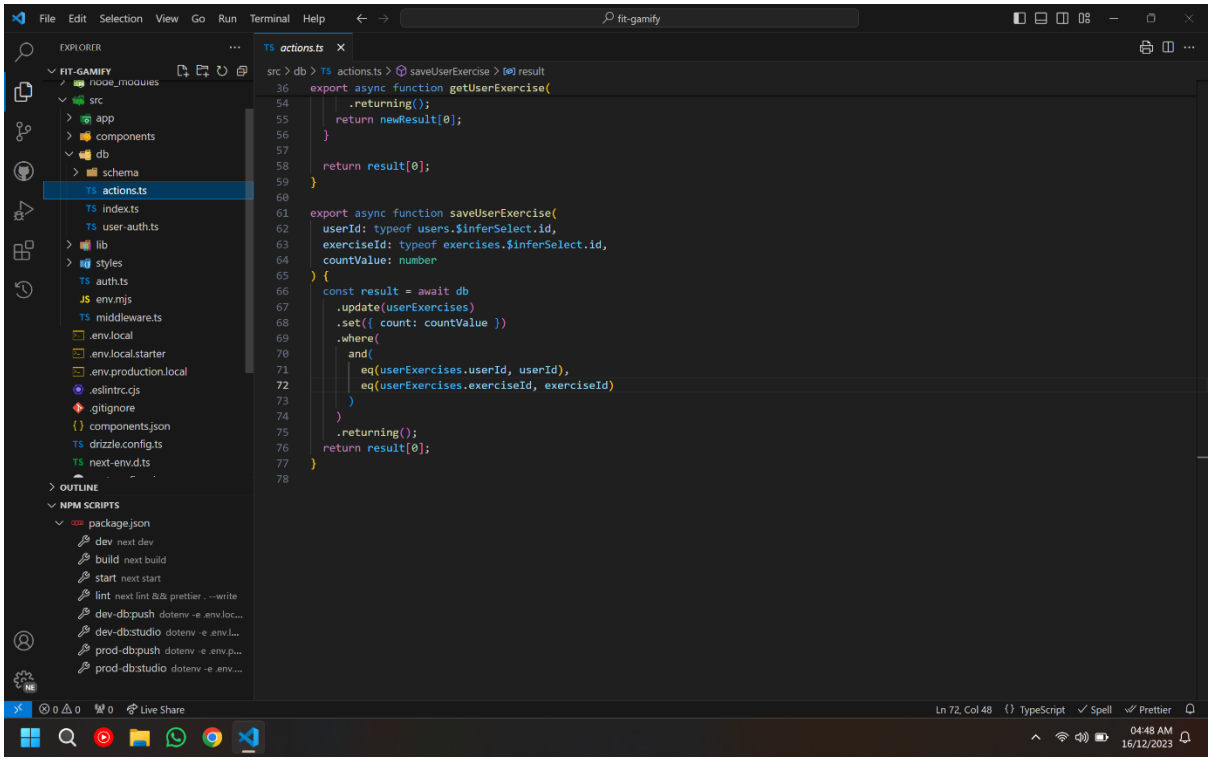
Coding Section

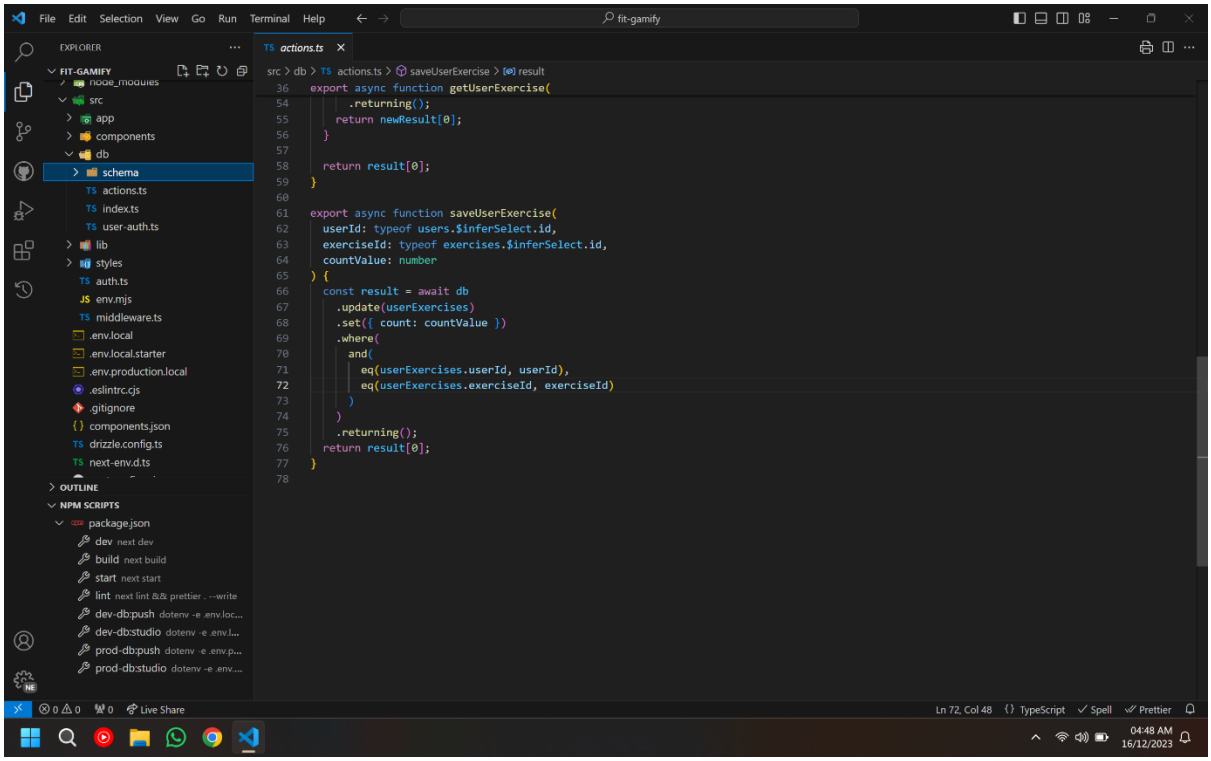
`

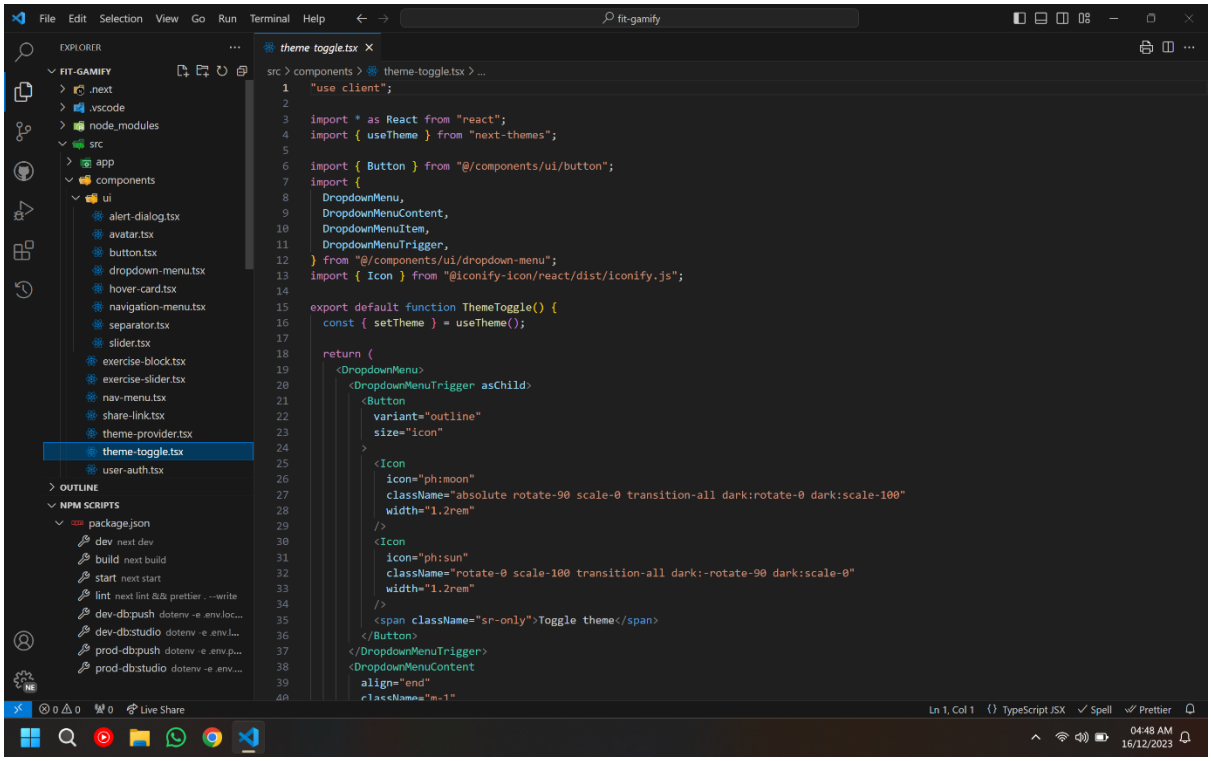


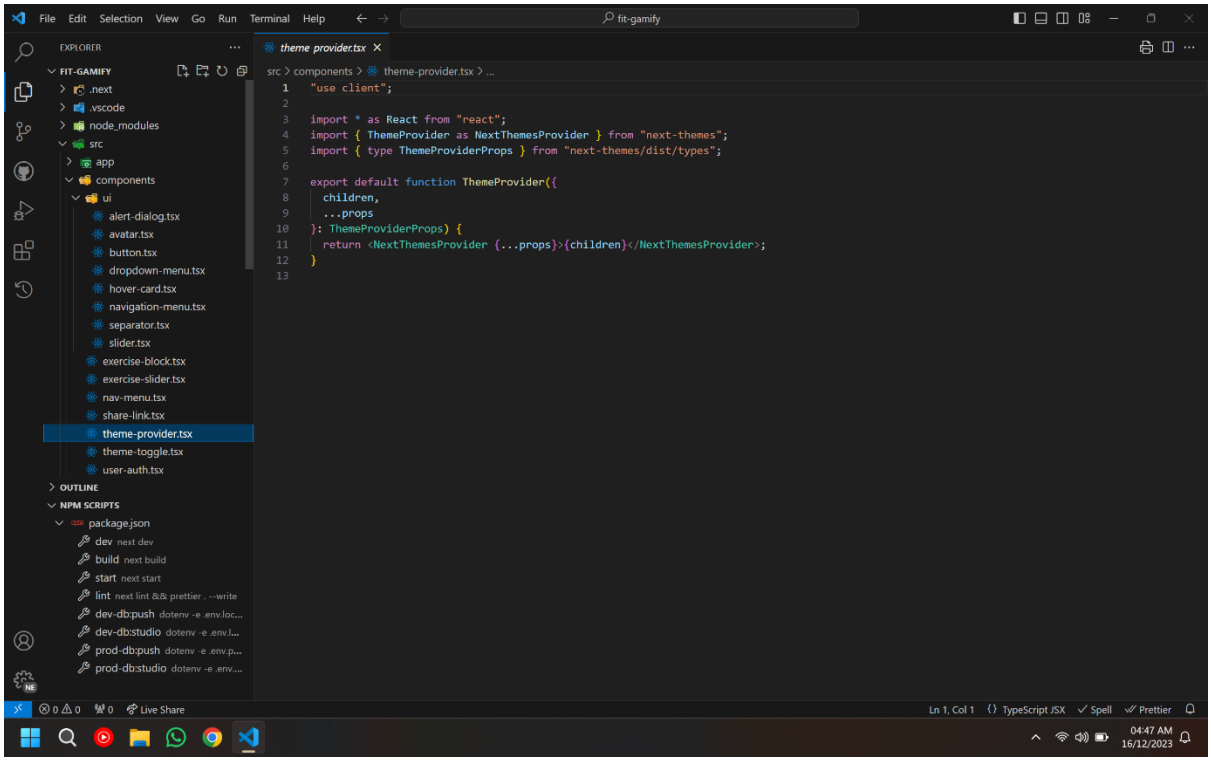


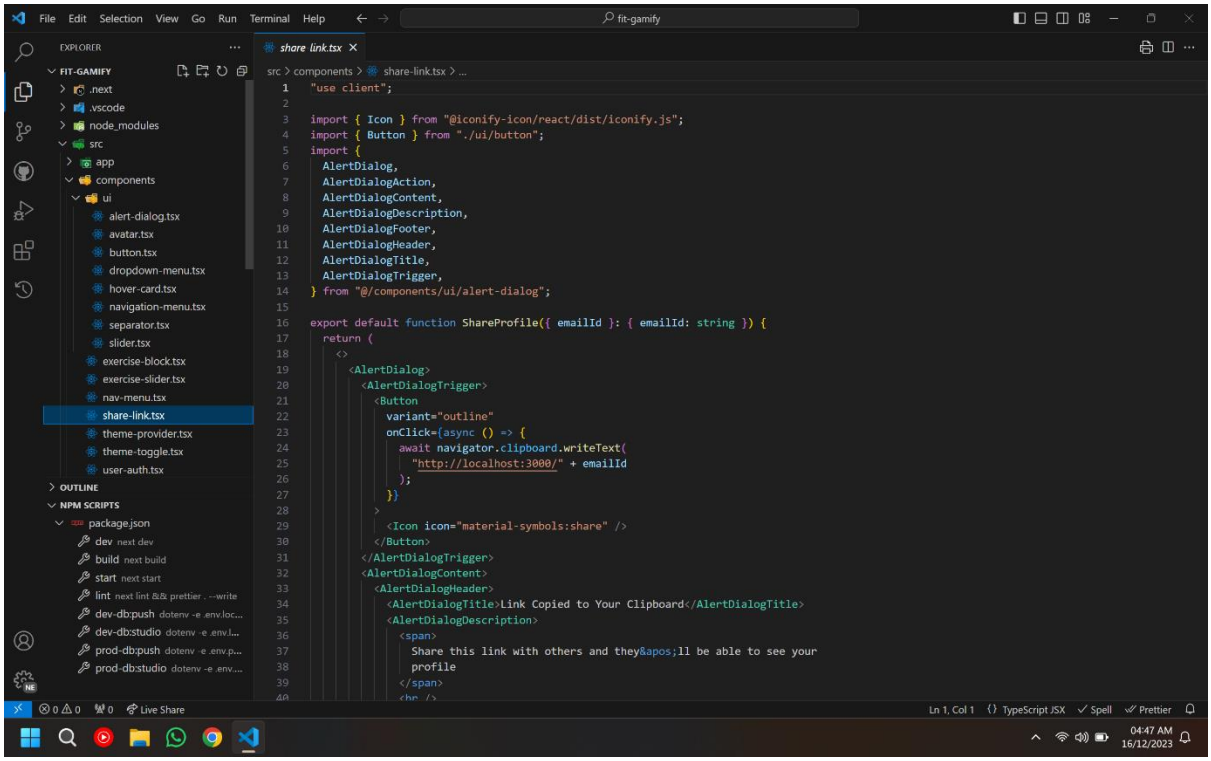

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
FIT-GAMIFY
user auth.ts
src > db > user-auth.ts > ...
"use server";
import { signIn, signOut } from "@/auth";
export async function signOutUser() {
await signOut({ redirectTo: "/" });
}
export async function signInUser(providerId: string) {
await signIn(providerId, { redirectTo: "/home" });
}
OUTLINE
NPM SCRIPTS
package.json
Ln 1, Col 1
TypeScript
Spell
Prettier
Live Share
04:48 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
FIT-GAMIFY
actions.ts
src > db > actions.ts > saveUserExercise > result
export async function getUserExercise(
.returning();
return newResult[0];
}
return result[0];
}
export async function saveUserExercise(
userId: typeof users.$inferSelect.id,
exerciseId: typeof exercises.$inferSelect.id,
countValue: number
) {
const result = await db
.update(userExercises)
.set({ count: countValue })
.where(
and(
eq(userExercises.userId, userId),
eq(userExercises.exerciseId, exerciseId)
)
)
.returning();
return result[0];
}
OUTLINE
NPM SCRIPTS
package.json
Ln 72, Col 48
TypeScript
Spell
Prettier
Live Share
04:48 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
FIT-GAMIFY
src
app
components
db
schema
actions.ts
index.ts
user-auth.ts
lib
styles
auth.ts
env.mjs
middleware.ts
.env.local
.env.local.starter
.env.production.local
.eslintrc.cjs
.gitignore
components.json
drizzle.config.ts
next-env.d.ts
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
actions.ts
src > db > actions.ts > saveUserExercise > result
export async function getUserExercise(
.returning();
return newResult[0];
}
return result[0];
}
export async function saveUserExercise(
userId: typeof users.$inferSelect.id,
exerciseId: typeof exercises.$inferSelect.id,
countValue: number
) {
const result = await db
.update(userExercises)
.set({ count: countValue })
.where(
and(
eq(userExercises.userId, userId),
eq(userExercises.exerciseId, exerciseId)
)
)
.returning();
return result[0];
}
Live Share
Ln 72, Col 48
TypeScript
Spell
Prettier
04:48 AM
16/12/2023

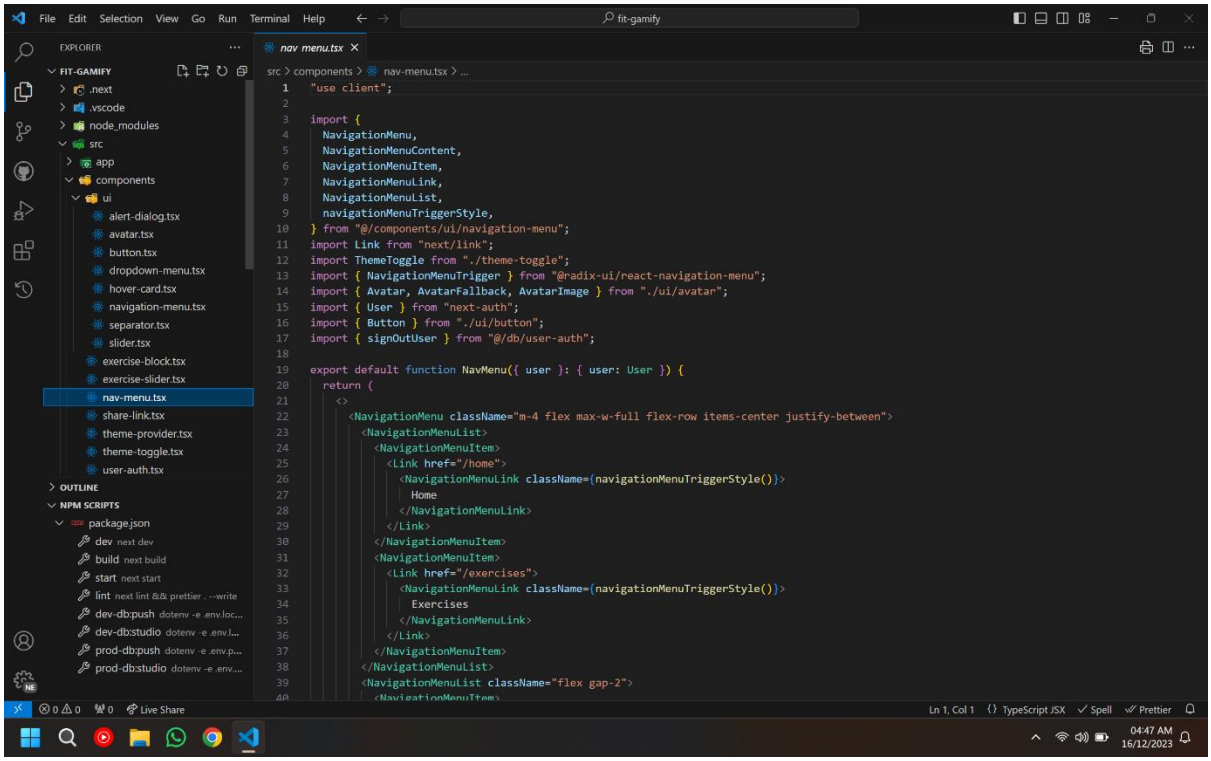

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
theme toggle.tsx
src > components > theme-toggle.tsx > ...
"use client";
import * as React from "react";
import { useTheme } from "next-themes";
import { Button } from "@/components/ui/button";
import {
DropdownMenu,
DropdownMenuContent,
DropdownMenuItem,
DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import { Icon } from "@iconify-icon/react/dist/iconify.js";
export default function ThemeToggle() {
const { setTheme } = useTheme();
return (
<DropdownMenu>
<DropdownMenuTrigger asChild>
<Button
variant="outline"
size="icon"
>
<Icon
icon="ph:moon"
className="absolute rotate-90 scale-0 transition-all dark:rotate-0 dark:scale-100"
width="1.2rem"
/>
<Icon
icon="ph:sun"
className="rotate-0 scale-100 transition-all dark:-rotate-90 dark:scale-0"
width="1.2rem"
/>
Toggle theme
</Button>
</DropdownMenuTrigger>
<DropdownMenuContent
align="end"
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:48 AM
16/12/2023

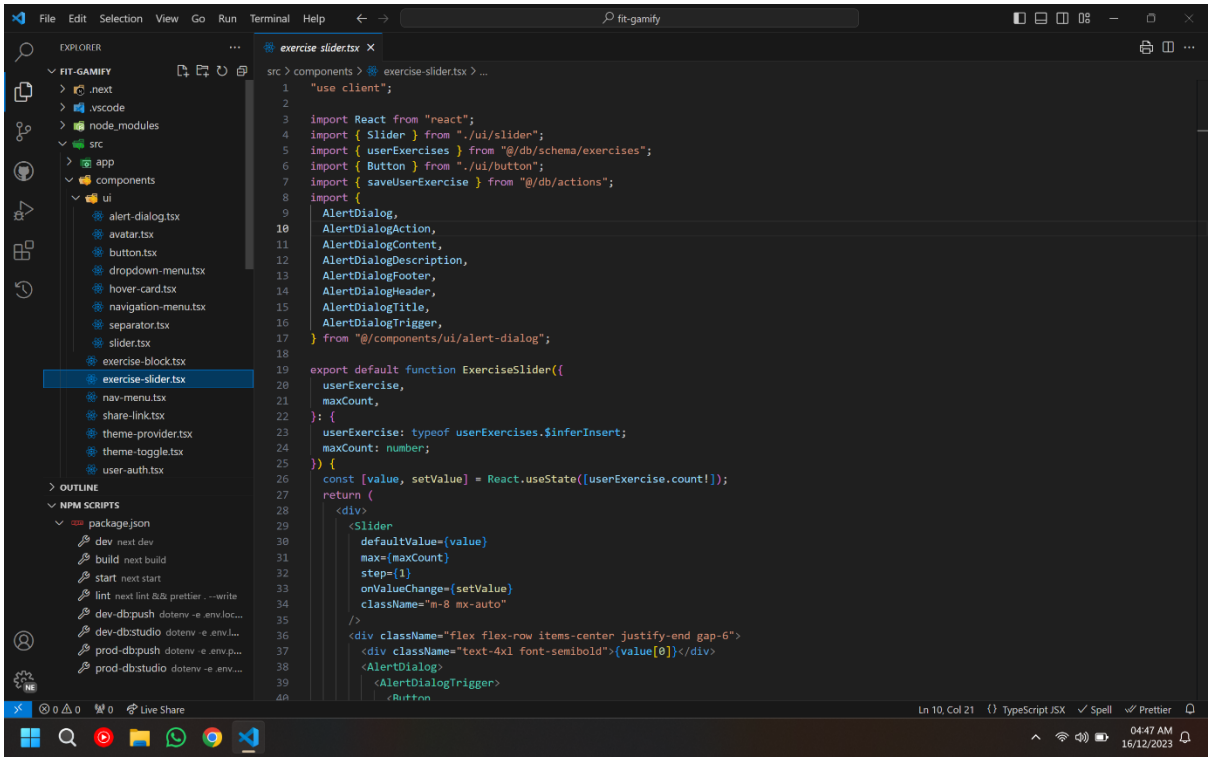

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
theme provider.tsx
src > components > theme-provider.tsx > ...
"use client";
import * as React from "react";
import { ThemeProvider as NextThemesProvider } from "next-themes";
import { type ThemeProviderProps } from "next-themes/dist/types";
export default function ThemeProvider({
children,
...props
}: ThemeProviderProps) {
return <NextThemesProvider {...props}>{children}</NextThemesProvider>;
}
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
share link.tsx
src > components > share-link.tsx > ...
"use client";
import { Icon } from "@iconify-icon/react/dist/iconify.js";
import { Button } from "./ui/button";
import {
AlertDialog,
AlertDialogAction,
AlertDialogContent,
AlertDialogDescription,
AlertDialogFooter,
AlertDialogHeader,
AlertDialogTitle,
AlertDialogTrigger,
} from "@/components/ui/alert-dialog";
export default function ShareProfile({ emailId }: { emailId: string }) {
return (
<>
<AlertDialog>
<AlertDialogTrigger>
<Button
variant="outline"
onClick={async () => {
await navigator.clipboard.writeText(
"http://localhost:3000/" + emailId
);
}}
>
<Icon icon="material-symbols:share" />
</Button>
</AlertDialogTrigger>
<AlertDialogContent>
<AlertDialogHeader>
<AlertDialogTitle>Link Copied to Your Clipboard</AlertDialogTitle>
<AlertDialogDescription>

Share this link with others and they'll be able to see your
profile

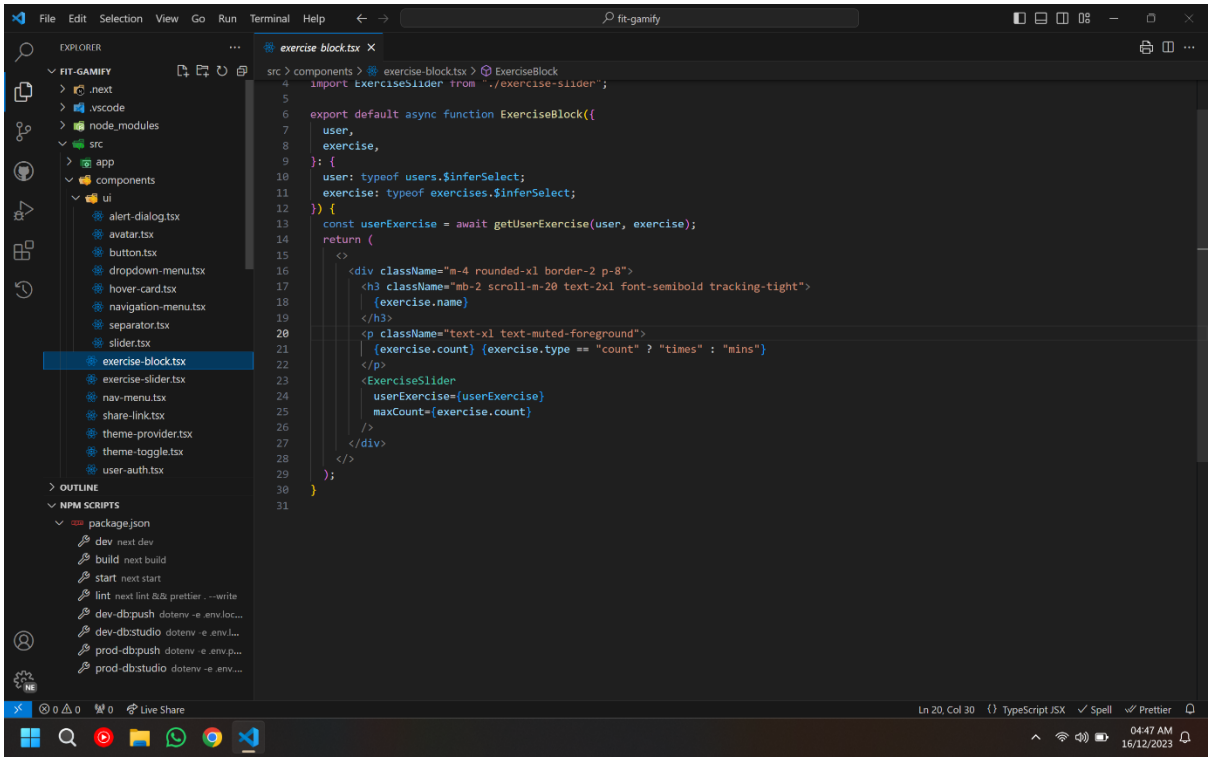
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
nav menu.tsx
src > components > nav-menu.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
"use client";
import {
NavigationMenu,
NavigationMenuContent,
NavigationMenuItem,
NavigationMenuLink,
NavigationMenuList,
navigationMenuTriggerStyle,
} from "@/components/ui/navigation-menu";
import Link from "next/link";
import ThemeToggle from "./theme-toggle";
import { NavigationMenuTrigger } from "@radix-ui/react-navigation-menu";
import { Avatar, AvatarFallback, AvatarImage } from "./ui/avatar";
import { User } from "next-auth";
import { Button } from "./ui/button";
import { signOutUser } from "@/db/user-auth";
export default function NavMenu({ user }: { user: User }) {
return (
<>
<NavigationMenu className="m-4 flex max-w-full flex-row items-center justify-between">
<NavigationMenuList>
<NavigationMenuItem>
<Link href="/home">
<NavigationMenuLink className={navigationMenuTriggerStyle()}>
Home
</NavigationMenuLink>
</Link>
</NavigationMenuItem>
<NavigationMenuItem>
<Link href="/exercises">
<NavigationMenuLink className={navigationMenuTriggerStyle()}>
Exercises
</NavigationMenuLink>
</Link>
</NavigationMenuItem>
</NavigationMenuList>
<NavigationMenuList className="flex gap-2">
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
exercise slider.tsx
src > components > exercise-slider.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
"use client";
import React from "react";
import { Slider } from "./ui/slider";
import { userExercises } from "@/db/schema/exercises";
import { Button } from "./ui/button";
import { saveUserExercise } from "@/db/actions";
import {
AlertDialog,
AlertDialogAction,
AlertDialogContent,
AlertDialogDescription,
AlertDialogFooter,
AlertDialogHeader,
AlertDialogTitle,
AlertDialogTrigger,
} from "@/components/ui/alert-dialog";
export default function ExerciseSlider({
userExercise,
maxCount,
}: {
userExercise: typeof userExercises.$inferInsert;
maxCount: number;
}) {
const [value, setValue] = React.useState([userExercise.count!]);
return (

<Slider
defaultValue={value}
max={maxCount}
step={1}
onValueChange={setValue}
className="m-8 mx-auto"
/>

{value[0]}
<AlertDialog>
<AlertDialogTrigger>
Live Share
Ln 10, Col 21
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

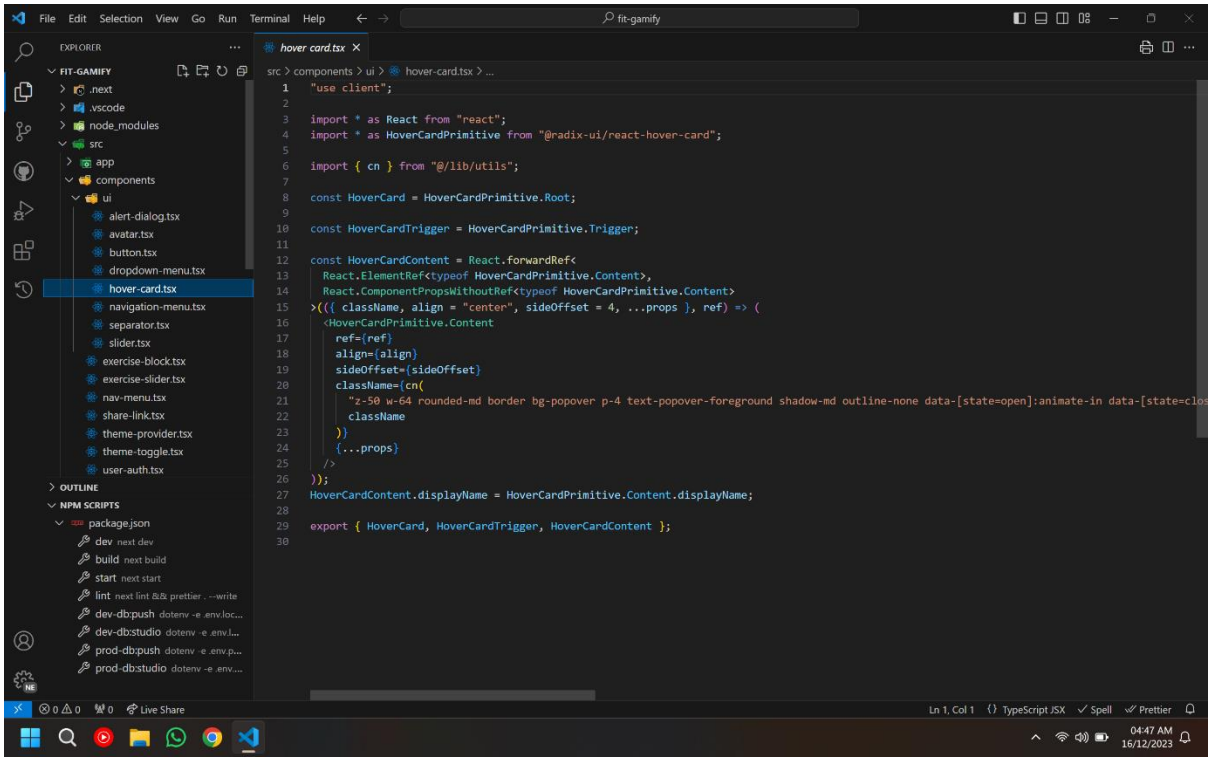

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
exercise block.tsx
src > components > exercise-block.tsx > ExerciseBlock
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
import ExerciseSlider from "./exercise-slider";
export default async function ExerciseBlock({
user,
exercise,
}: {
user: typeof users.$inferSelect;
exercise: typeof exercises.$inferSelect;
}) {
const userExercise = await getUserExercise(user, exercise);
return (
<>

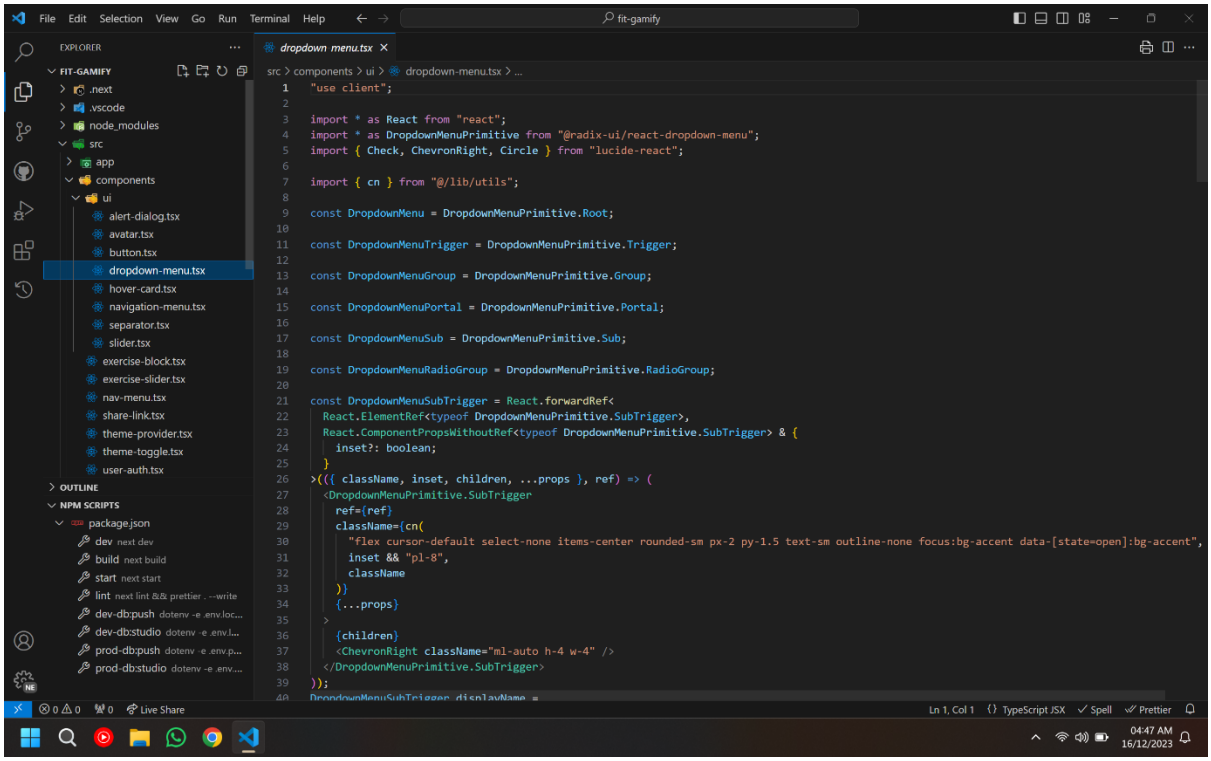
<h3 className="mb-2 scroll-m-20 text-2xl font-semibold tracking-tight">
{exercise.name}
</h3>

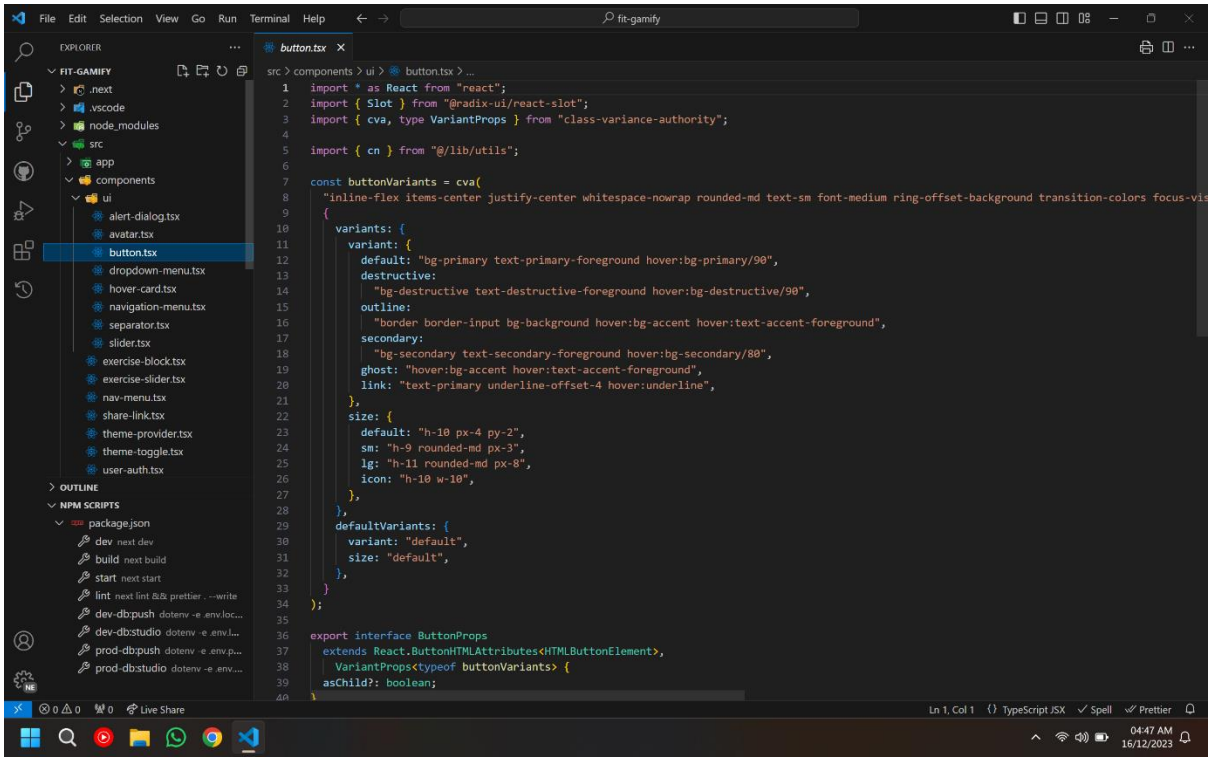
{exercise.count} {exercise.type == "count" ? "times" : "mins"}

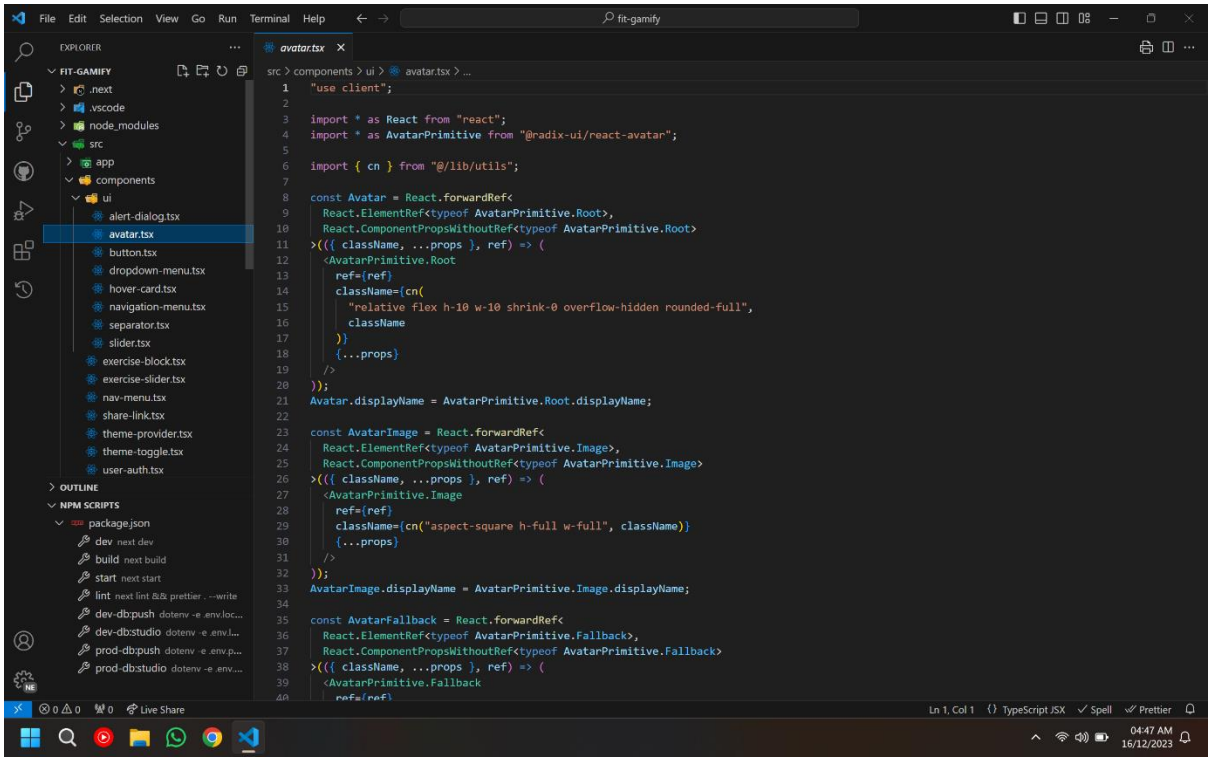
<ExerciseSlider
userExercise={userExercise}
maxCount={exercise.count}
/>

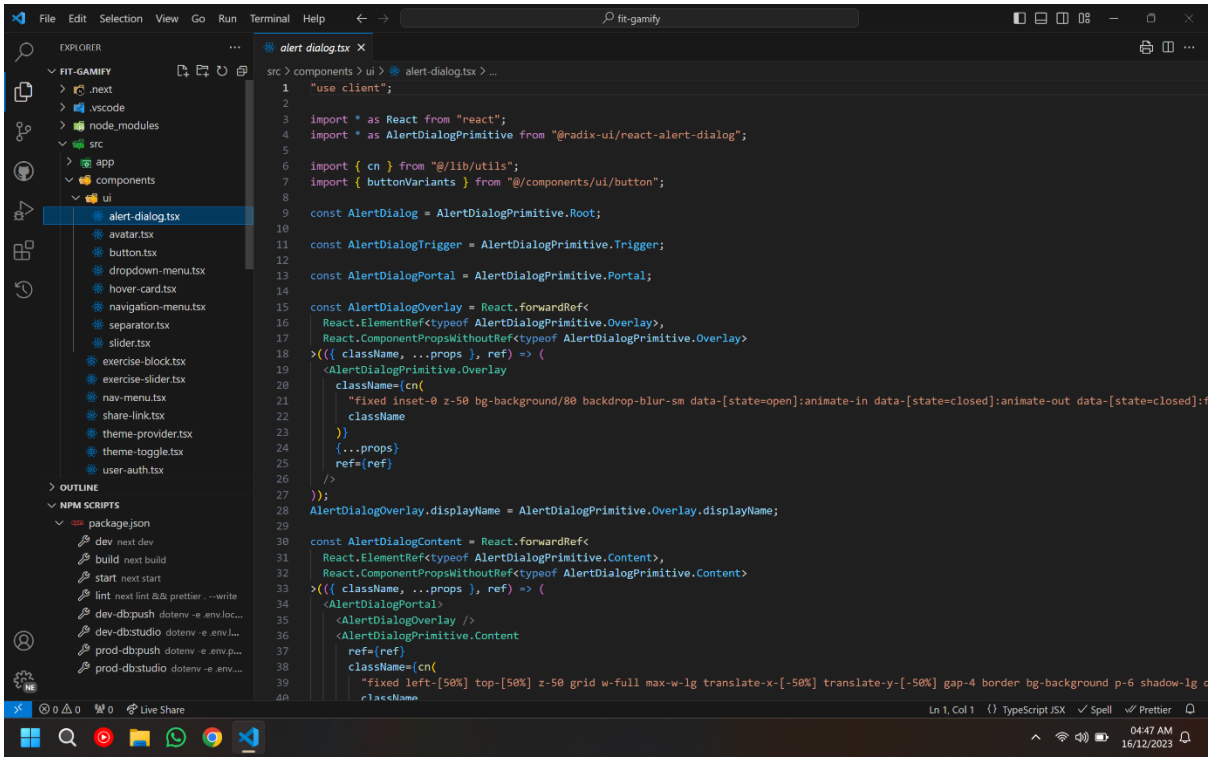
</>
);
}
Live Share
Ln 20, Col 30
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

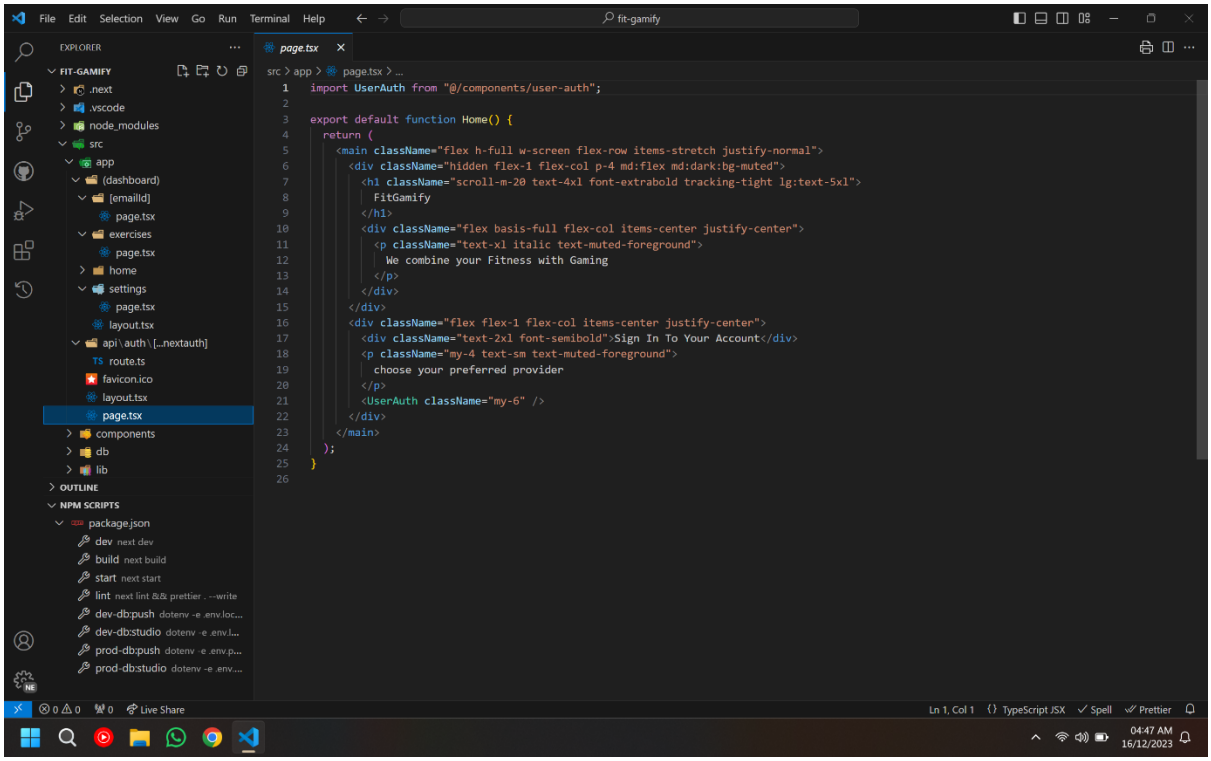

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
hover card.tsx
src > components > ui > hover-card.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
"use client";
import * as React from "react";
import * as HoverCardPrimitive from "@radix-ui/react-hover-card";
import { cn } from "@/lib/utils";
const HoverCard = HoverCardPrimitive.Root;
const HoverCardTrigger = HoverCardPrimitive.Trigger;
const HoverCardContent = React.forwardRef<
React.ElementRef<typeof HoverCardPrimitive.Content>,
React.ComponentPropsWithoutRef<typeof HoverCardPrimitive.Content>
>(({ className, align = "center", sideOffset = 4, ...props }, ref) => (
<HoverCardPrimitive.Content
ref={ref}
align={align}
sideOffset={sideOffset}
className={cn(
"z-50 w-64 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=clos
className
)}
{...props}
/>
));
HoverCardContent.displayName = HoverCardPrimitive.Content.displayName;
export { HoverCard, HoverCardTrigger, HoverCardContent };
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

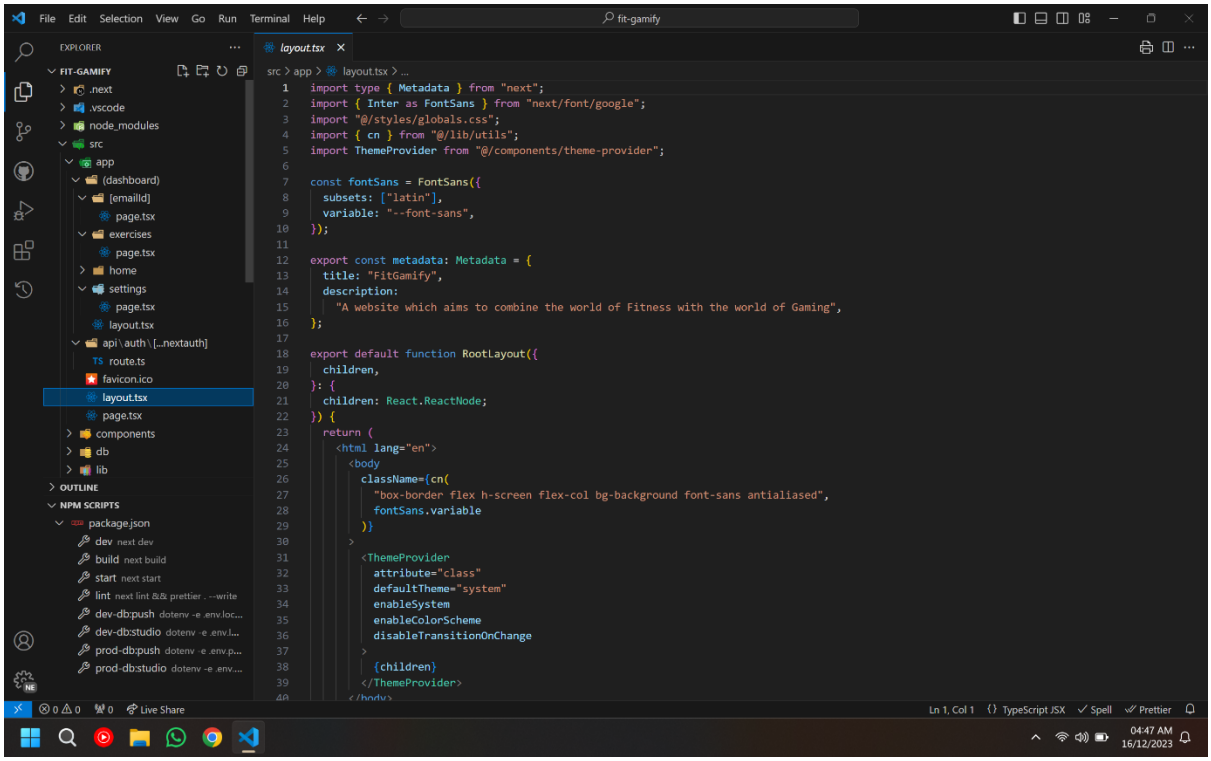

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
dropdown menu.tsx
src > components > ui > dropdown-menu.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
"use client";
import * as React from "react";
import * as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu";
import { Check, ChevronRight, Circle } from "lucide-react";
import { cn } from "@/lib/utils";
const DropdownMenu = DropdownMenuPrimitive.Root;
const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger;
const DropdownMenuGroup = DropdownMenuPrimitive.Group;
const DropdownMenuPortal = DropdownMenuPrimitive.Portal;
const DropdownMenuSub = DropdownMenuPrimitive.Sub;
const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup;
const DropdownMenuSubTrigger = React.forwardRef<
React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,
React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {
inset?: boolean;
}
>(({ className, inset, children, ...props }, ref) => (
<DropdownMenuPrimitive.SubTrigger
ref={ref}
className={cn(
"flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent data-[state=open]:bg-accent",
inset && "pl-8",
className
)}
{...props}
>
{children}
<ChevronRight className="ml-auto h-4 w-4" />
</DropdownMenuPrimitive.SubTrigger>
));
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

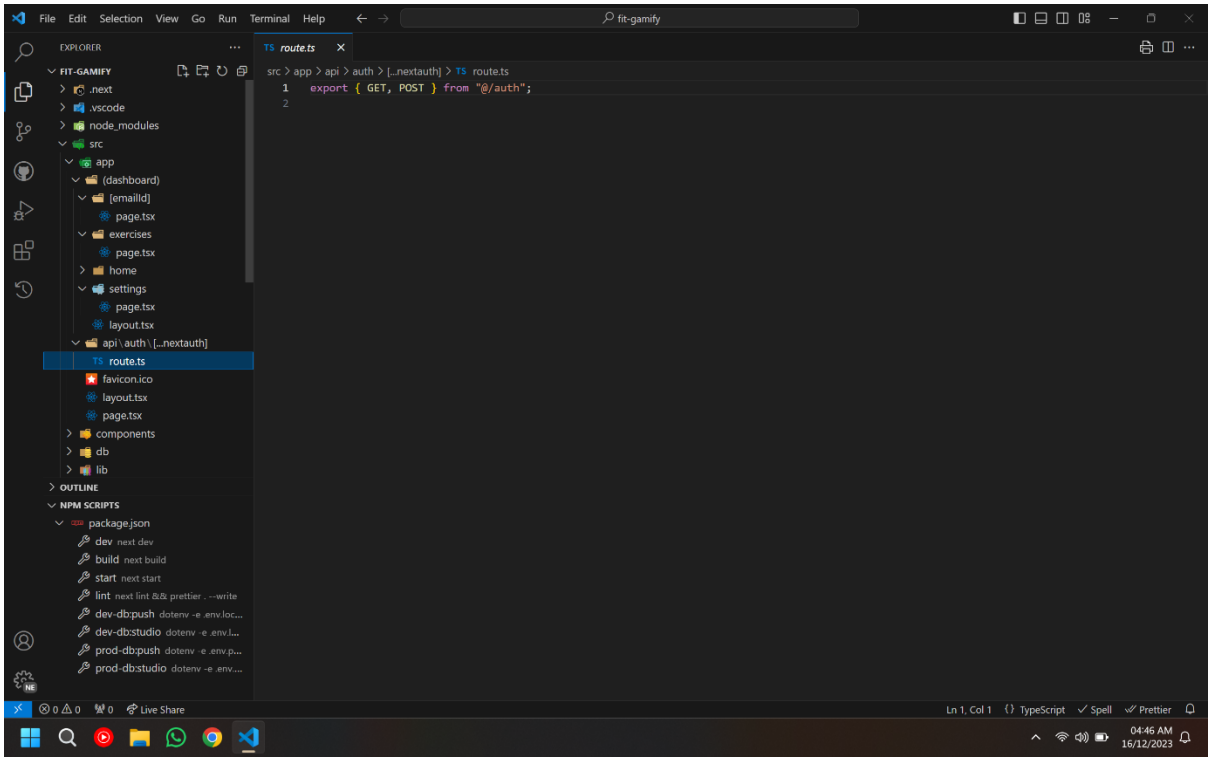

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
button.tsx
src > components > ui > button.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
import * as React from "react";
import { Slot } from "@radix-ui/react-slot";
import { cva, type VariantProps } from "class-variance-authority";
import { cn } from "@/lib/utils";
const buttonVariants = cva(
"inline-flex items-center justify-center whitespace-nowrap rounded-md text-sm font-medium ring-offset-background transition-colors focus-vis
{
variants: {
variant: {
default: "bg-primary text-primary-foreground hover:bg-primary/90",
destructive:
"bg-destructive text-destructive-foreground hover:bg-destructive/90",
outline:
"border border-input bg-background hover:bg-accent hover:text-accent-foreground",
secondary:
"bg-secondary text-secondary-foreground hover:bg-secondary/80",
ghost: "hover:bg-accent hover:text-accent-foreground",
link: "text-primary underline-offset-4 hover:underline",
},
size: {
default: "h-10 px-4 py-2",
sm: "h-9 rounded-md px-3",
lg: "h-11 rounded-md px-8",
icon: "h-10 w-10",
},
},
defaultVariants: {
variant: "default",
size: "default",
},
}
);
export interface ButtonProps
extends React.ButtonHTMLAttributes<HTMLButtonElement>,
VariantProps<typeof buttonVariants> {
asChild?: boolean;
}
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

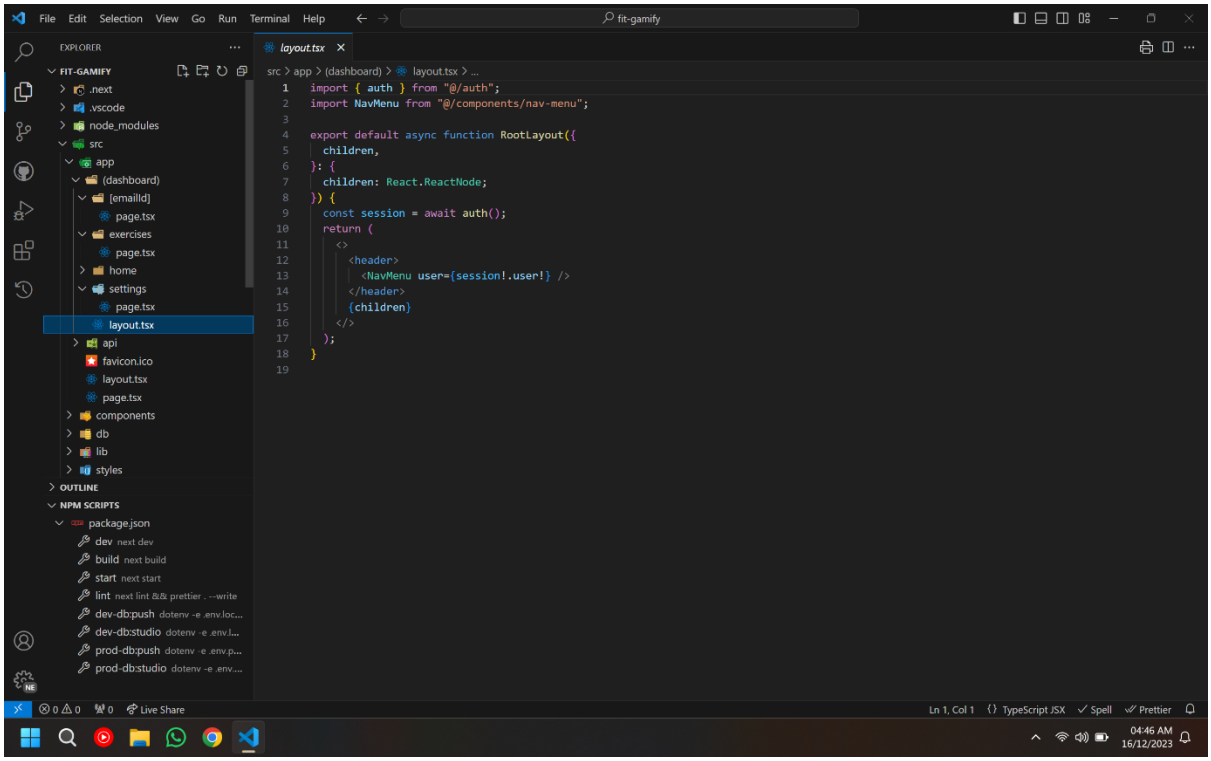

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
avatar.tsx
src > components > ui > avatar.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
"use client";
import * as React from "react";
import * as AvatarPrimitive from "@radix-ui/react-avatar";
import { cn } from "@/lib/utils";
const Avatar = React.forwardRef<
React.ElementRef<typeof AvatarPrimitive.Root>,
React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>
>(({ className, ...props }, ref) => (
<AvatarPrimitive.Root
ref={ref}
className={cn(
"relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full",
className
)}
{...props}
/>
));
Avatar.displayName = AvatarPrimitive.Root.displayName;
const AvatarImage = React.forwardRef<
React.ElementRef<typeof AvatarPrimitive.Image>,
React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>
>(({ className, ...props }, ref) => (
<AvatarPrimitive.Image
ref={ref}
className={cn("aspect-square h-full w-full", className)}
{...props}
/>
));
AvatarImage.displayName = AvatarPrimitive.Image.displayName;
const AvatarFallback = React.forwardRef<
React.ElementRef<typeof AvatarPrimitive.Fallback>,
React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>
>(({ className, ...props }, ref) => (
<AvatarPrimitive.Fallback
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
Live Share
04:47 AM
16/12/2023

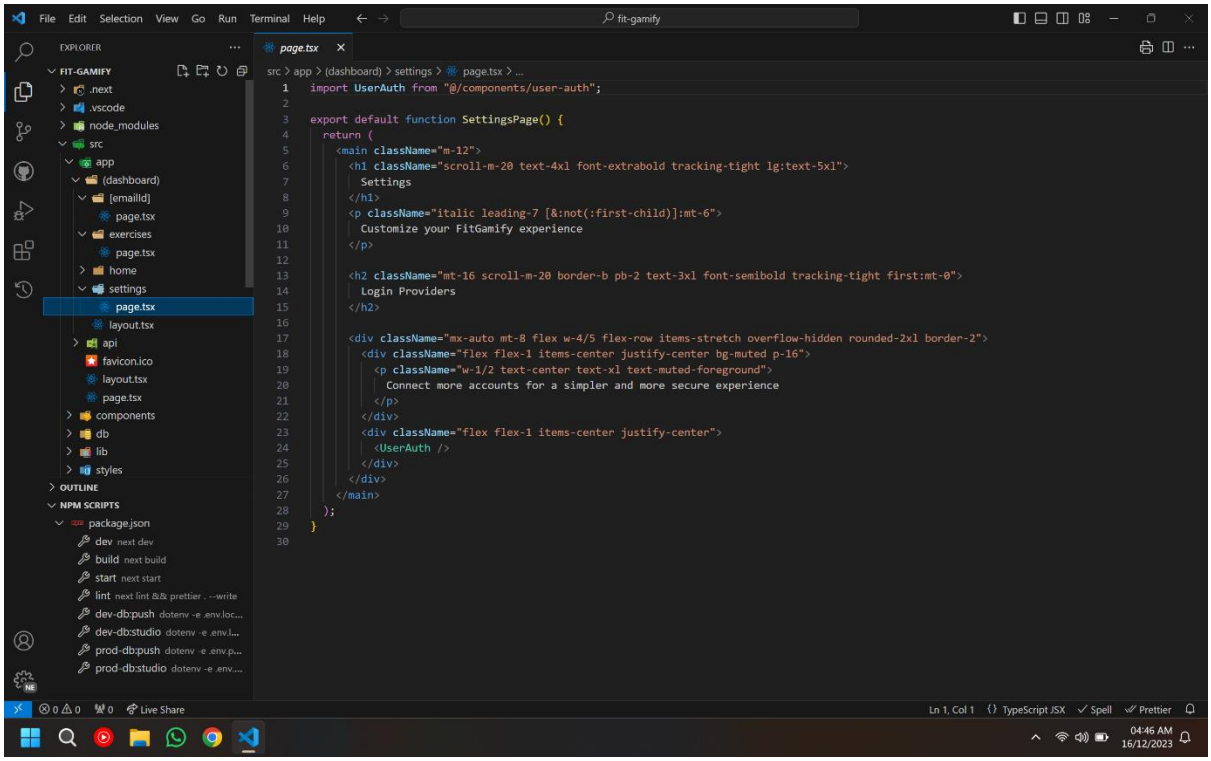

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
alert dialog.tsx
src > components > ui > alert-dialog.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
components
ui
alert-dialog.tsx
avatar.tsx
button.tsx
dropdown-menu.tsx
hover-card.tsx
navigation-menu.tsx
separator.tsx
slider.tsx
exercise-block.tsx
exercise-slider.tsx
nav-menu.tsx
share-link.tsx
theme-provider.tsx
theme-toggle.tsx
user-auth.tsx
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
"use client";
import * as React from "react";
import * as AlertDialogPrimitive from "@radix-ui/react-alert-dialog";
import { cn } from "@/lib/utils";
import { buttonVariants } from "@/components/ui/button";
const AlertDialog = AlertDialogPrimitive.Root;
const AlertDialogTrigger = AlertDialogPrimitive.Trigger;
const AlertDialogPortal = AlertDialogPrimitive.Portal;
const AlertDialogOverlay = React.forwardRef<
React.ElementRef<typeof AlertDialogPrimitive.Overlay>,
React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Overlay>
>(({ className, ...props }, ref) => (
<AlertDialogPrimitive.Overlay
className={cn(
"fixed inset-0 z-50 bg-background/80 backdrop-blur-sm data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:f
className
)}
{...props}
ref={ref}
/>
));
AlertDialogOverlay.displayName = AlertDialogPrimitive.Overlay.displayName;
const AlertDialogContent = React.forwardRef<
React.ElementRef<typeof AlertDialogPrimitive.Content>,
React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Content>
>(({ className, ...props }, ref) => (
<AlertDialogPortal>
<AlertDialogOverlay />
<AlertDialogPrimitive.Content
ref={ref}
className={cn(
"fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
Live Share
04:47 AM
16/12/2023

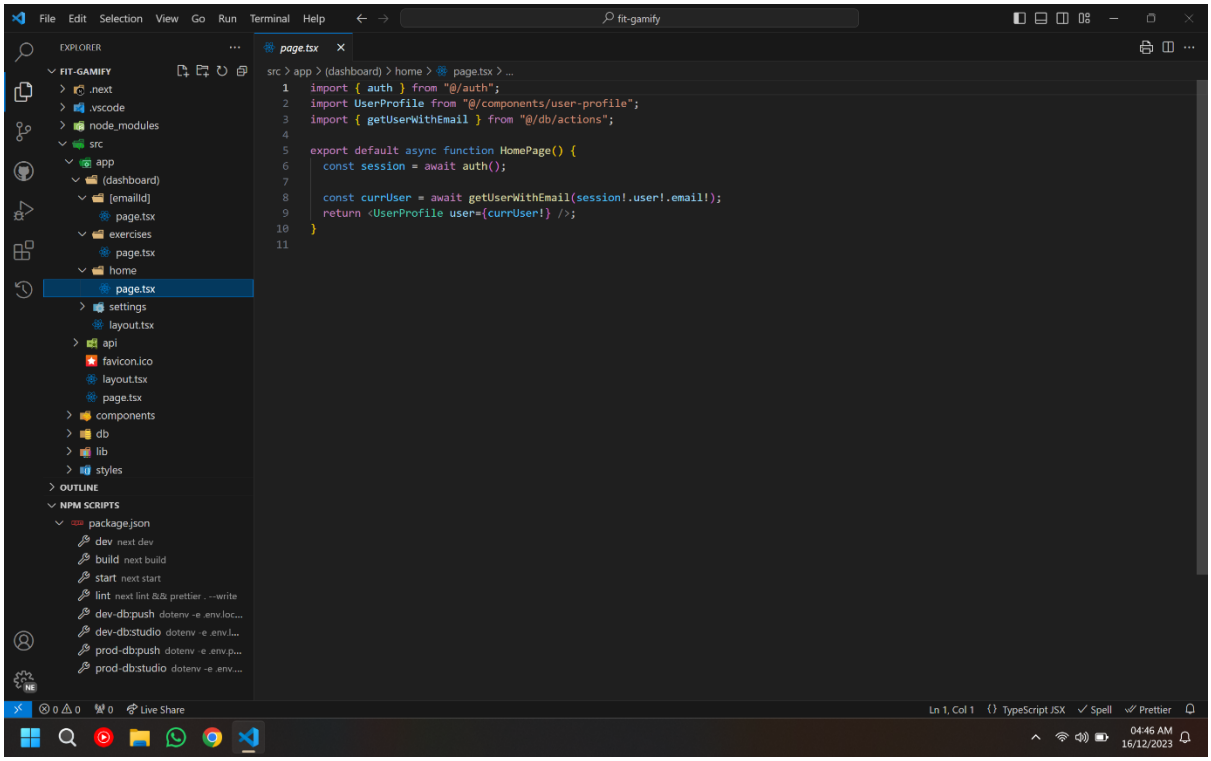

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
page.tsx
src > app > page.tsx > ...
1 import UserAuth from "@/components/user-auth";
3 export default function Home() {
4 return (
5 <main className="flex h-full w-screen flex-row items-stretch justify-normal">
6
7 <h1 className="scroll-m-20 text-4xl font-extrabold tracking-tight lg:text-5xl">
8 FitGamify
9 </h1>
10
11
12 We combine your Fitness with Gaming
13
14
15
16
17 Sign In To Your Account
18
19 choose your preferred provider
20
21 <UserAuth className="my-6" />
22
23 </main>
24);
25 }
26
FIT-GAMIFY
.next
.vscode
node_modules
src
app
(dashboard)
[emailId]
page.tsx
exercises
page.tsx
home
settings
page.tsx
layout.tsx
api\auth\[...nextauth]
route.ts
favicon.ico
layout.tsx
page.tsx
components
db
lib
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

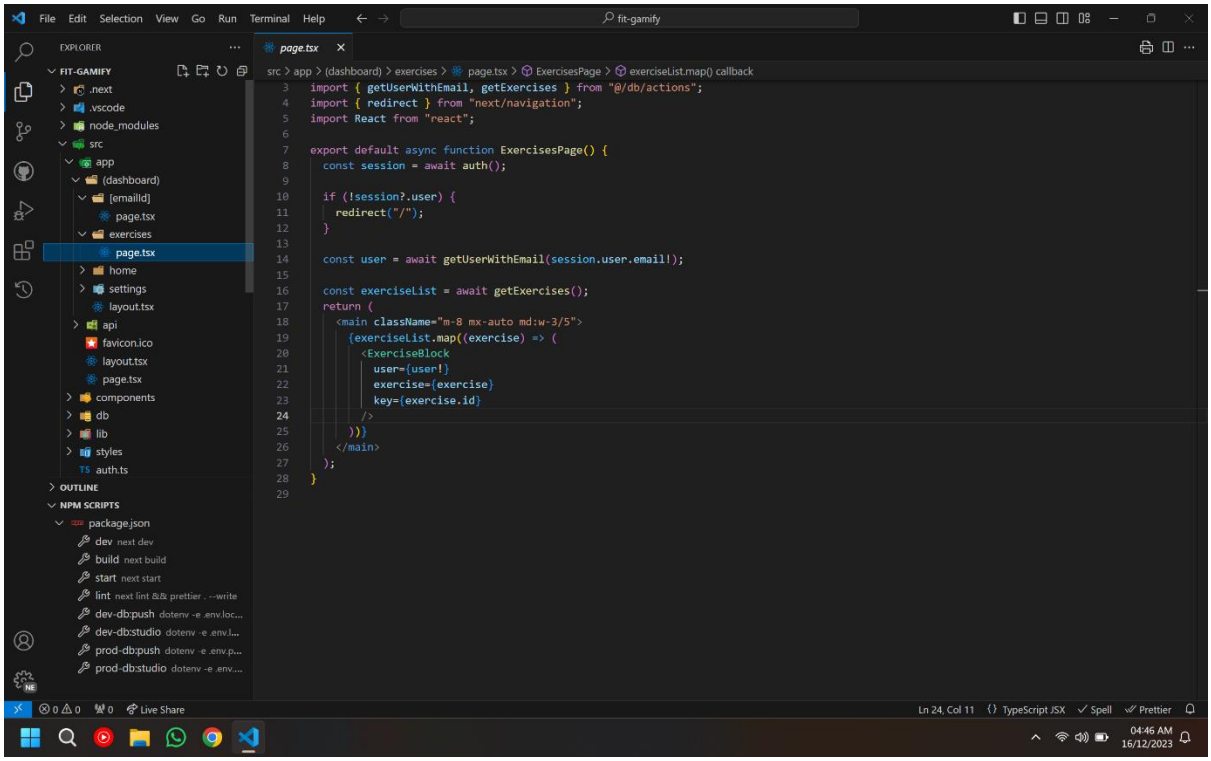

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
layout.tsx
src > app > layout.tsx > ...
1 import type { Metadata } from "next";
2 import { Inter as FontSans } from "next/font/google";
3 import "@/styles/globals.css";
4 import { cn } from "@/lib/utils";
5 import ThemeProvider from "@/components/theme-provider";
6
7 const fontSans = FontSans({
8 subsets: ["latin"],
9 variable: "--font-sans",
10 });
11
12 export const metadata: Metadata = {
13 title: "FitGamify",
14 description:
15 "A website which aims to combine the world of Fitness with the world of Gaming",
16 };
17
18 export default function RootLayout({
19 children,
20 }: {
21 children: React.ReactNode;
22 }) {
23 return (
24 <html lang="en">
25 <body
26 className={cn(
27 "box-border flex h-screen flex-col bg-background font-sans antialiased",
28 fontSans.variable
29)}
30 >
31 <ThemeProvider
32 attribute="class"
33 defaultTheme="system"
34 enableSystem
35 enableColorScheme
36 disableTransitionOnChange
37 >
38 {children}
39 </ThemeProvider>
FIT-GAMIFY
.next
.vscode
node_modules
src
app
(dashboard)
[emailId]
page.tsx
exercises
page.tsx
home
settings
page.tsx
layout.tsx
api\auth\[...nextauth]
route.ts
favicon.ico
layout.tsx
page.tsx
components
db
lib
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:47 AM
16/12/2023

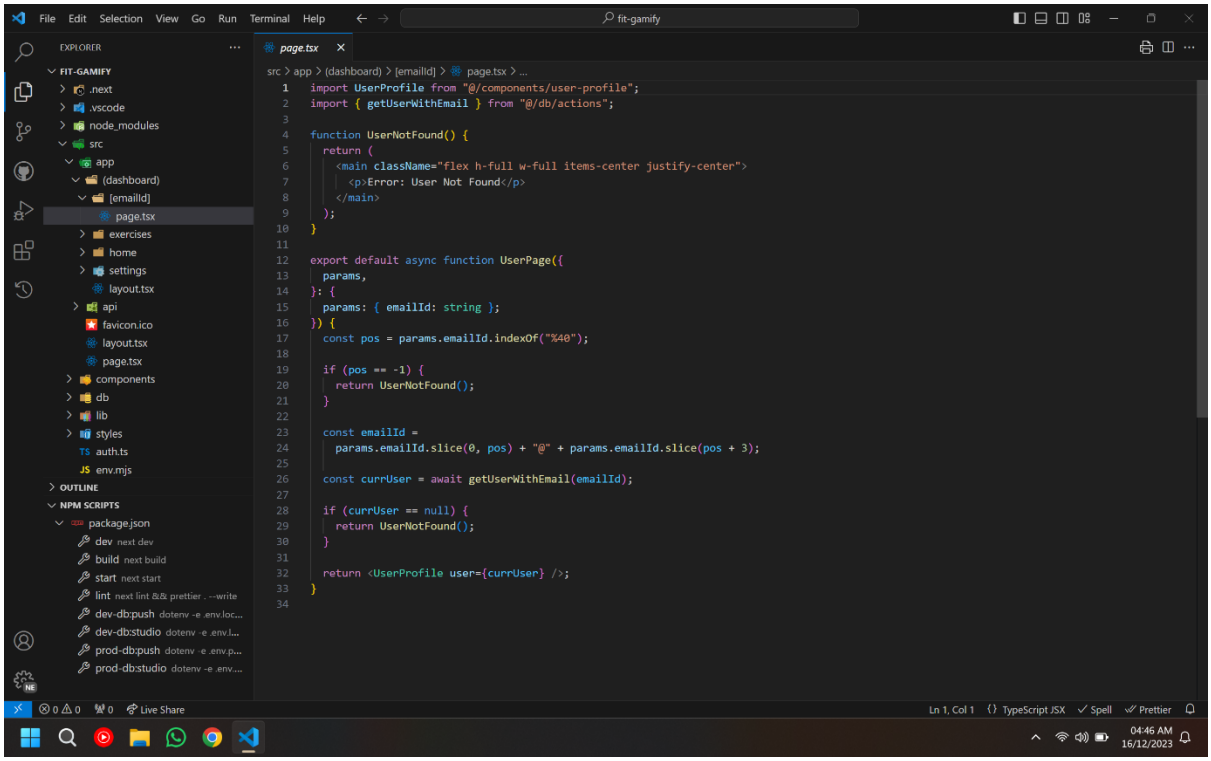

File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
route.ts
src > app > api > auth > [...nextauth] > route.ts
1 export { GET, POST } from "@/auth";
2
FIT-GAMIFY
.next
.vscode
node_modules
src
app
(dashboard)
[emailId]
page.tsx
exercises
page.tsx
home
settings
page.tsx
layout.tsx
api \ auth \ [...nextauth]
route.ts
favicon.ico
layout.tsx
page.tsx
components
db
lib
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
Live Share
Ln 1, Col 1
TypeScript
Spell
Prettier
04:46 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
layout.tsx
src > app > (dashboard) > layout.tsx > ...
1 import { auth } from "@/auth";
2 import NavMenu from "@/components/nav-menu";
3
4 export default async function RootLayout({
5 children,
6 }: {
7 children: React.ReactNode;
8 }) {
9 const session = await auth();
10 return (
11 <>
12 <header>
13 <NavMenu user={session!.user!} />
14 </header>
15 {children}
16 </>
17);
18 }
19
FIT-GAMIFY
.next
.vscode
node_modules
src
app
(dashboard)
[emailId]
page.tsx
exercises
page.tsx
home
settings
page.tsx
layout.tsx
api
favicon.ico
layout.tsx
page.tsx
components
db
lib
styles
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:46 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
page.tsx
src > app > (dashboard) > settings > page.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
(dashboard)
[emailId]
page.tsx
exercises
page.tsx
home
settings
page.tsx
layout.tsx
api
favicon.ico
layout.tsx
page.tsx
components
db
lib
styles
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
1 import UserAuth from "@/components/user-auth";
2
3 export default function SettingsPage() {
4 return (
5 <main className="m-12">
6 <h1 className="scroll-m-20 text-4xl font-extrabold tracking-tight lg:text-5xl">
7 Settings
8 </h1>
9
10 Customize your FitGamify experience
11
12
13 <h2 className="mt-16 scroll-m-20 border-b pb-2 text-3xl font-semibold tracking-tight first:mt-0">
14 Login Providers
15 </h2>
16
17
18
19
20 Connect more accounts for a simpler and more secure experience
21
22
23
24 <UserAuth />
25
26
27 </main>
28);
29 }
30
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:46 AM
16/12/2023


File Edit Selection View Go Run Terminal Help
fit-gamify
EXPLORER
page.tsx
src > app > (dashboard) > home > page.tsx > ...
FIT-GAMIFY
.next
.vscode
node_modules
src
app
(dashboard)
[emailId]
page.tsx
exercises
page.tsx
home
page.tsx
settings
layout.tsx
api
favicon.ico
layout.tsx
page.tsx
components
db
lib
styles
OUTLINE
NPM SCRIPTS
package.json
dev next dev
build next build
start next start
lint next lint && prettier . --write
dev-db:push dotenv -e .env.loc...
dev-db:studio dotenv -e .env.l...
prod-db:push dotenv -e .env.p...
prod-db:studio dotenv -e .env....
1 import { auth } from "@/auth";
2 import UserProfile from "@/components/user-profile";
3 import { getUserWithEmail } from "@/db/actions";
4
5 export default async function HomePage() {
6 const session = await auth();
7
8 const currUser = await getUserWithEmail(session!.user!.email!);
9 return <UserProfile user={currUser!} />;
10 }
11
Live Share
Ln 1, Col 1
TypeScript JSX
Spell
Prettier
04:46 AM
16/12/2023





The above given screenshots of the project are a small part of the whole coding for the complete codes please visit the Github profile.

Username- dixitpiyush & x-priyam

References

1. TypeScript: The starting point for learning TypeScript (typescriptlang.org)
2. Documentation | Node.js (nodejs.org)
3. npm Docs (npmjs.com)
4. React
5. Docs | Next.js (nextjs.org)
6. What is Prettier? · Prettier
7. Documentation - ESLint - Pluggable JavaScript Linter
8. Documentation - Tailwind CSS
9. NextAuth.js (next-auth.js.org)
10. Introduction | Auth.js (authjs.dev)
11. Drizzle ORM - DrizzleORM
12. Zod | Documentation
13. Git - Documentation (git-scm.com)
14. Vercel Documentation | Vercel Docs
15. PlanetScale Documentation — PlanetScale Documentation