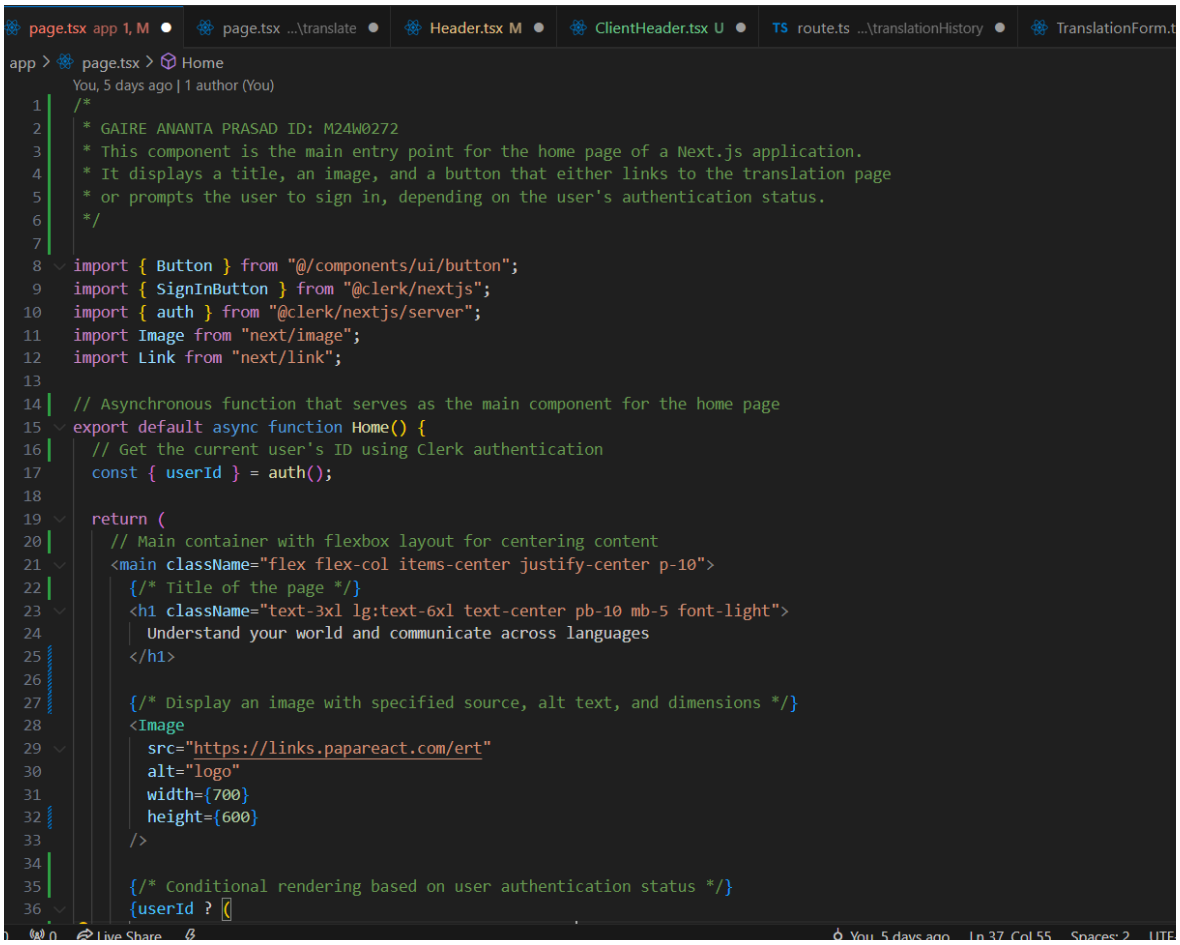
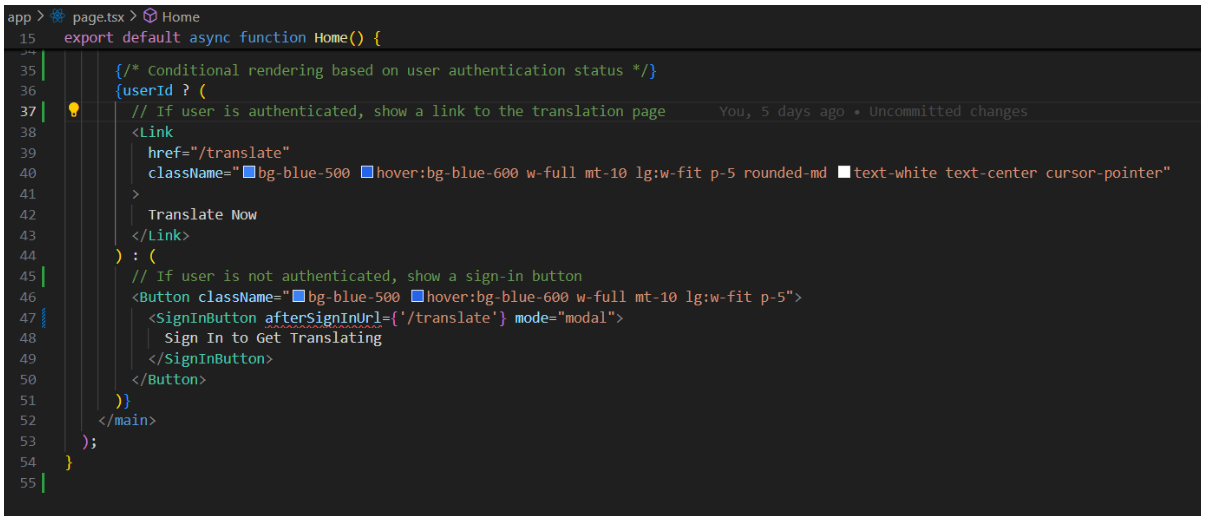
GAIRE ANANTA PRASAD (M24W0272)

Google Translate (JavaScript (React.js (Next.js)), Html5, CSS(Tailwind CSS), Microsoft Azure, Cosmos DB(MongoDB(Mongoose)), Cleark Authentication).

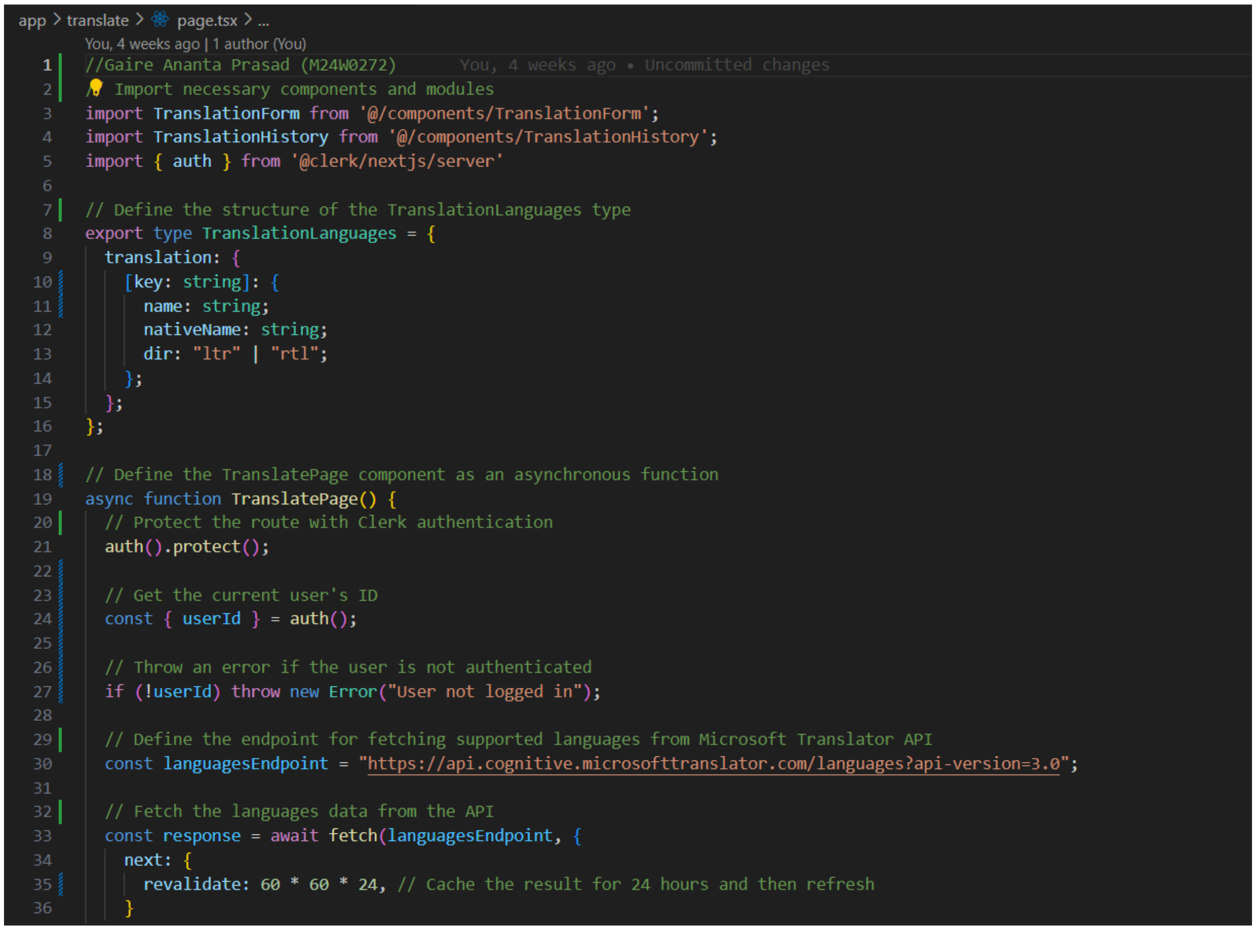


```
/*
 * GAIRE ANANTA PRASAD ID: M24W0272
 * This component is the main entry point for the home page of a Next.js application.
 * It displays a title, an image, and a button that either links to the translation page
 * or prompts the user to sign in, depending on the user's authentication status.
 */

import { Button } from "@/components/ui/button";
import { SignInButton } from "@clerk/nextjs";
import { auth } from "@clerk/nextjs/server";
import Image from "next/image";
import Link from "next/link";

// Asynchronous function that serves as the main component for the home page
export default async function Home() {
  // Get the current user's ID using Clerk authentication
  const { userId } = auth();

  return (
    // Main container with flexbox layout for centering content
    <main className="flex flex-col items-center justify-center p-10">
      {/* Title of the page */}
      <h1 className="text-3xl lg:text-6xl text-center pb-10 mb-5 font-light">
        Understand your world and communicate across languages
      </h1>

      {/* Display an image with specified source, alt text, and dimensions */}
      <Image
        src="https://links.papareact.com/ert"
        alt="logo"
        width={700}
        height={600}
      />

      {/* Conditional rendering based on user authentication status */}
      {userId ? (
```

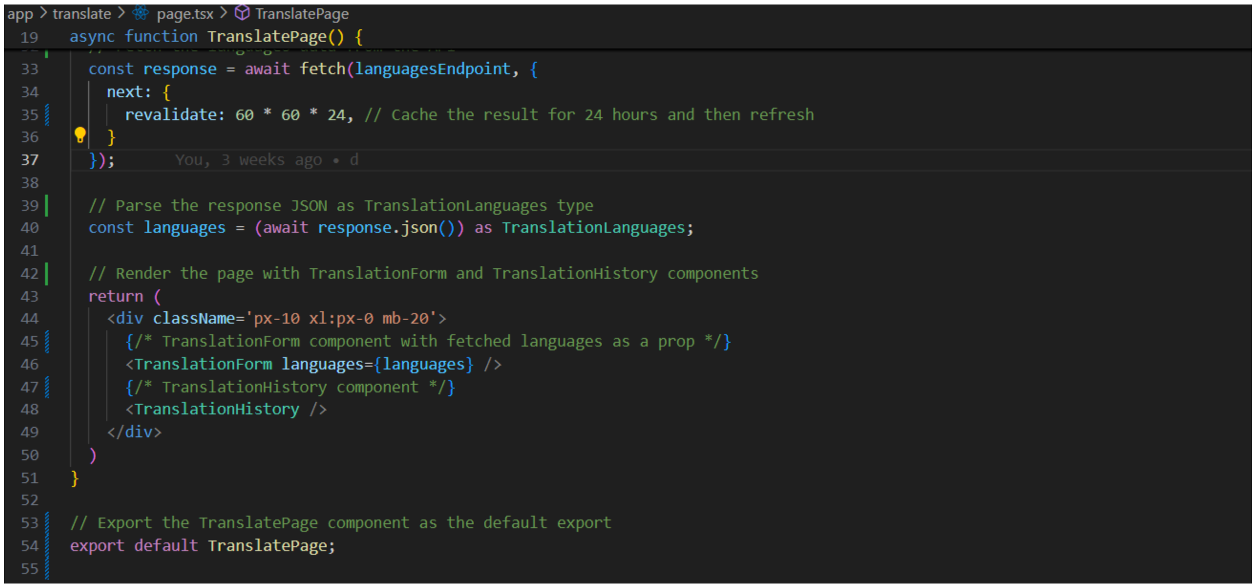


```
export default async function Home() {
      {/* Conditional rendering based on user authentication status */}
      {userId ? (
        // If user is authenticated, show a link to the translation page
        <Link
          href="/translate"
          className="bg-blue-500 hover:bg-blue-600 w-full mt-10 lg:w-fit p-5 rounded-md text-white text-center cursor-pointer"
        >
          Translate Now
        </Link>
      ) : (
        // If user is not authenticated, show a sign-in button
        <Button className="bg-blue-500 hover:bg-blue-600 w-full mt-10 lg:w-fit p-5">
          <SignInButton afterSignInUrl={'/translate'} mode="modal">
            Sign In to Get Translating
          </SignInButton>
        </Button>
      )}
    </main>
  );
}
```

GAIRE ANANTA PRASAD M24W0272

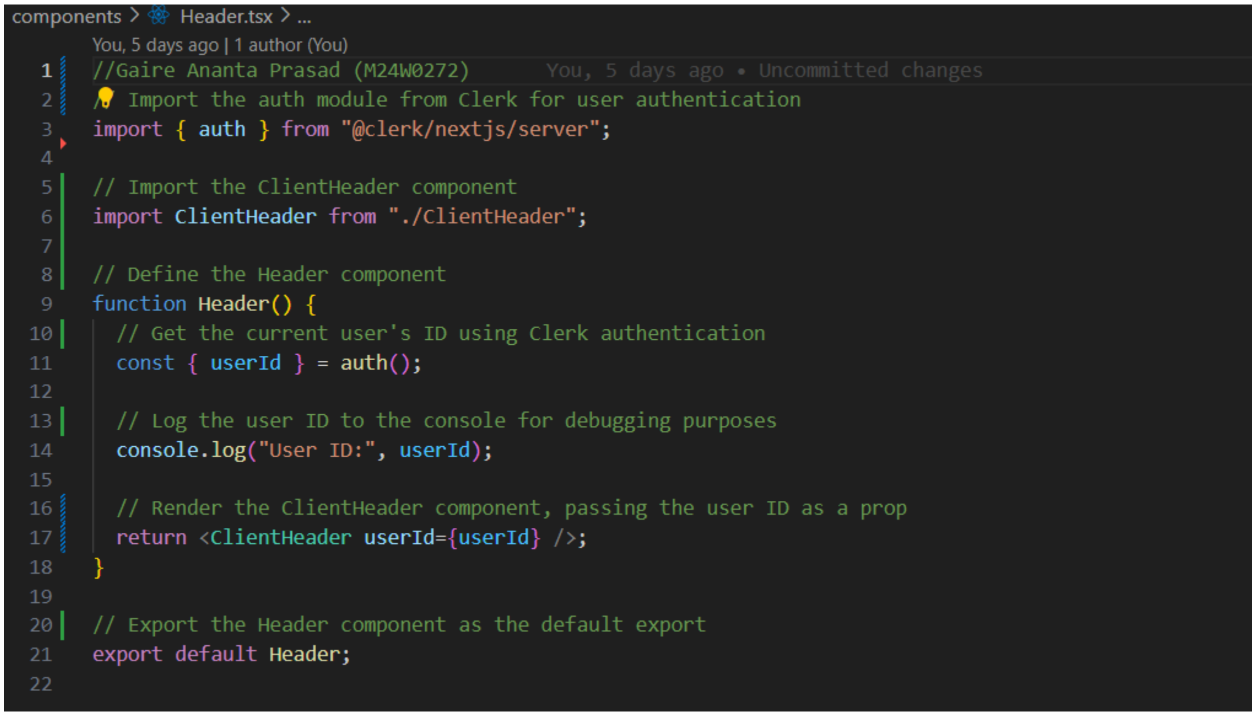

app > translate > page.tsx > ...
You, 4 weeks ago | 1 author (You)

```tsx
//Gaire Ananta Prasad (M24W0272)       You, 4 weeks ago • Uncommitted changes
// Import necessary components and modules
import TranslationForm from '@/components/TranslationForm';
import TranslationHistory from '@/components/TranslationHistory';
import { auth } from '@clerk/nextjs/server'

// Define the structure of the TranslationLanguages type
export type TranslationLanguages = {
  translation: {
    [key: string]: {
      name: string;
      nativeName: string;
      dir: "ltr" | "rtl";
    };
  };
};

// Define the TranslatePage component as an asynchronous function
async function TranslatePage() {
  // Protect the route with Clerk authentication
  auth().protect();

  // Get the current user's ID
  const { userId } = auth();

  // Throw an error if the user is not authenticated
  if (!userId) throw new Error("User not logged in");

  // Define the endpoint for fetching supported languages from Microsoft Translator API
  const languagesEndpoint = "https://api.cognitive.microsofttranslator.com/languages?api-version=3.0";

  // Fetch the languages data from the API
  const response = await fetch(languagesEndpoint, {
    next: {
      revalidate: 60 * 60 * 24, // Cache the result for 24 hours and then refresh
    }
```


app > translate > page.tsx > TranslatePage

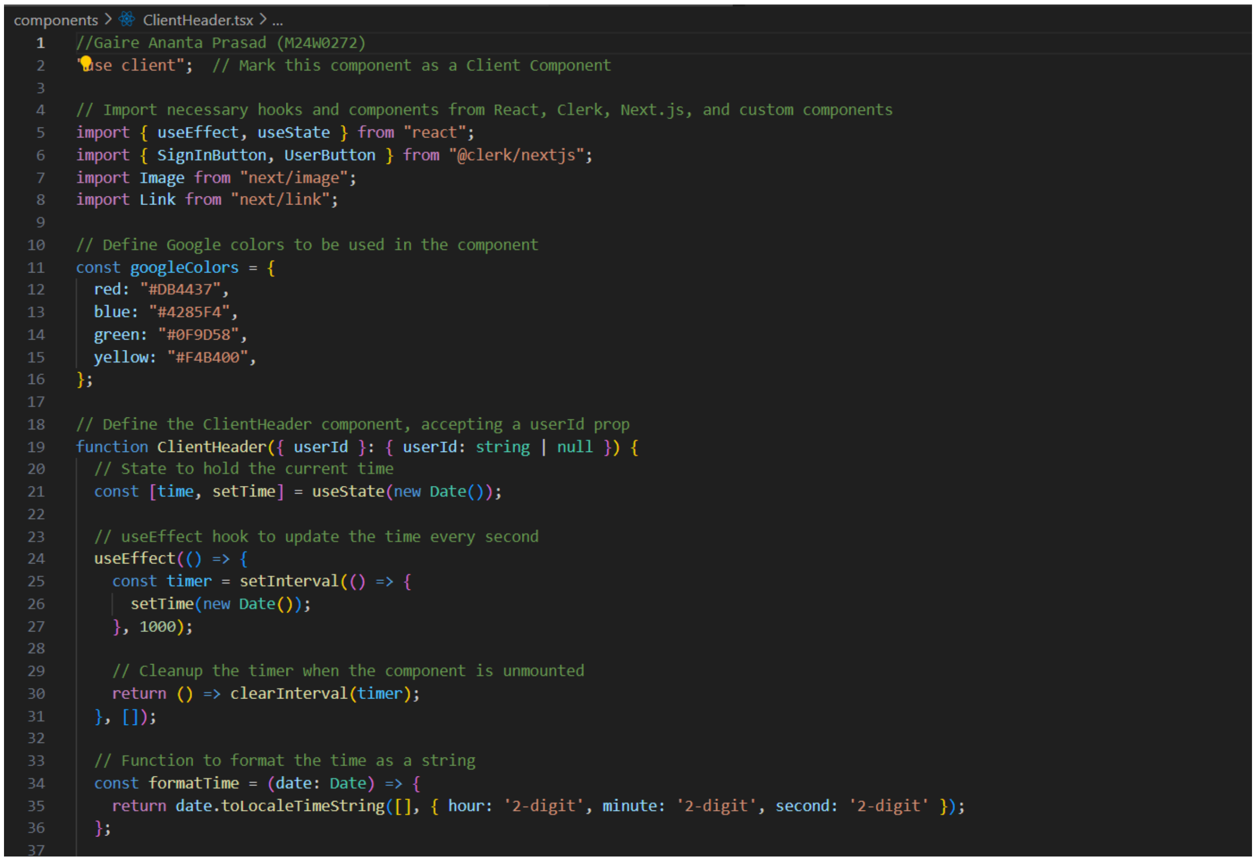
```tsx
async function TranslatePage() {
  const response = await fetch(languagesEndpoint, {
    next: {
      revalidate: 60 * 60 * 24, // Cache the result for 24 hours and then refresh
    }
  });       You, 3 weeks ago • d

  // Parse the response JSON as TranslationLanguages type
  const languages = (await response.json()) as TranslationLanguages;

  // Render the page with TranslationForm and TranslationHistory components
  return (
    <div className='px-10 xl:px-0 mb-20'>
      {/* TranslationForm component with fetched languages as a prop */}
      <TranslationForm languages={languages} />
      {/* TranslationHistory component */}
      <TranslationHistory />
    </div>
  )
}

// Export the TranslatePage component as the default export
export default TranslatePage;
```

GAIRE ANANTA PRASAD M24W0272


components > Header.tsx > ...

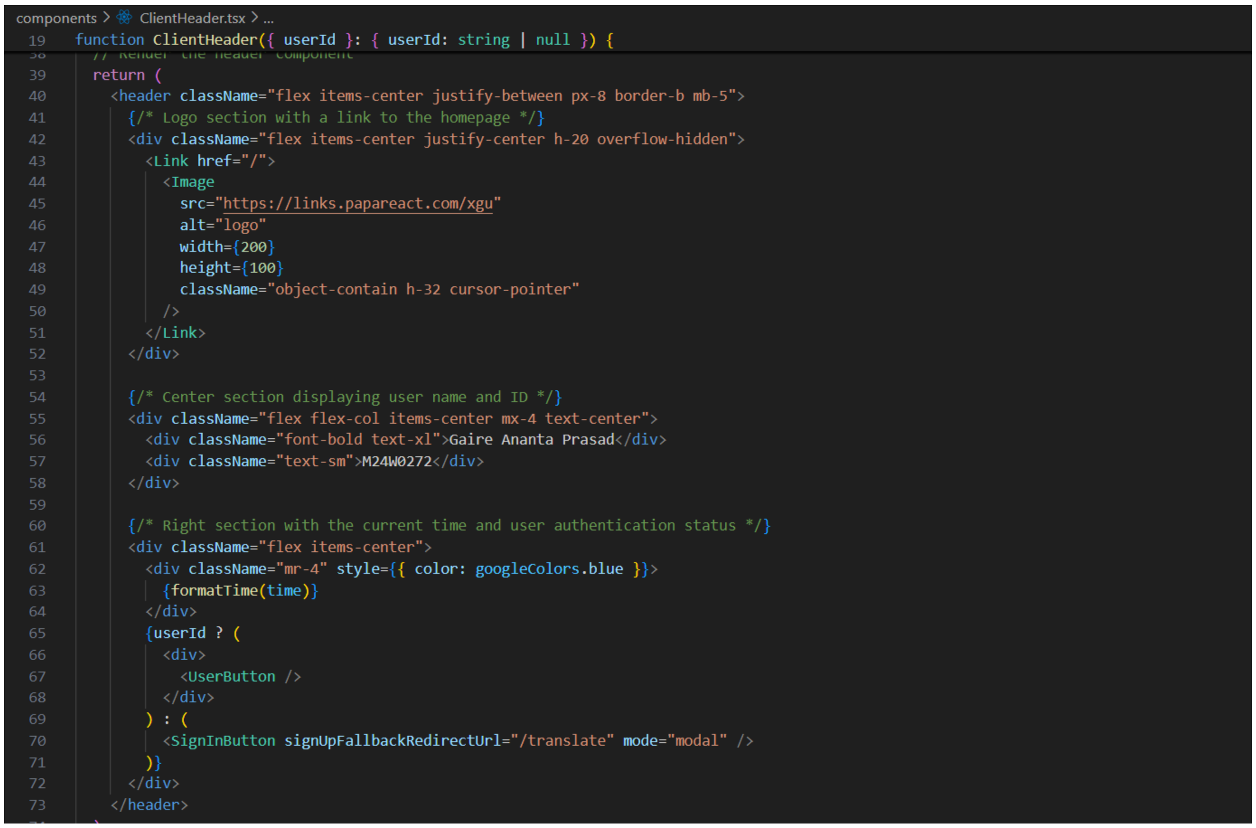
You, 5 days ago | 1 author (You)

```tsx
//Gaire Ananta Prasad (M24W0272)        You, 5 days ago • Uncommitted changes
// Import the auth module from Clerk for user authentication
import { auth } from "@clerk/nextjs/server";

// Import the ClientHeader component
import ClientHeader from "./ClientHeader";

// Define the Header component
function Header() {
  // Get the current user's ID using Clerk authentication
  const { userId } = auth();

  // Log the user ID to the console for debugging purposes
  console.log("User ID:", userId);

  // Render the ClientHeader component, passing the user ID as a prop
  return <ClientHeader userId={userId} />;
}

// Export the Header component as the default export
export default Header;

```

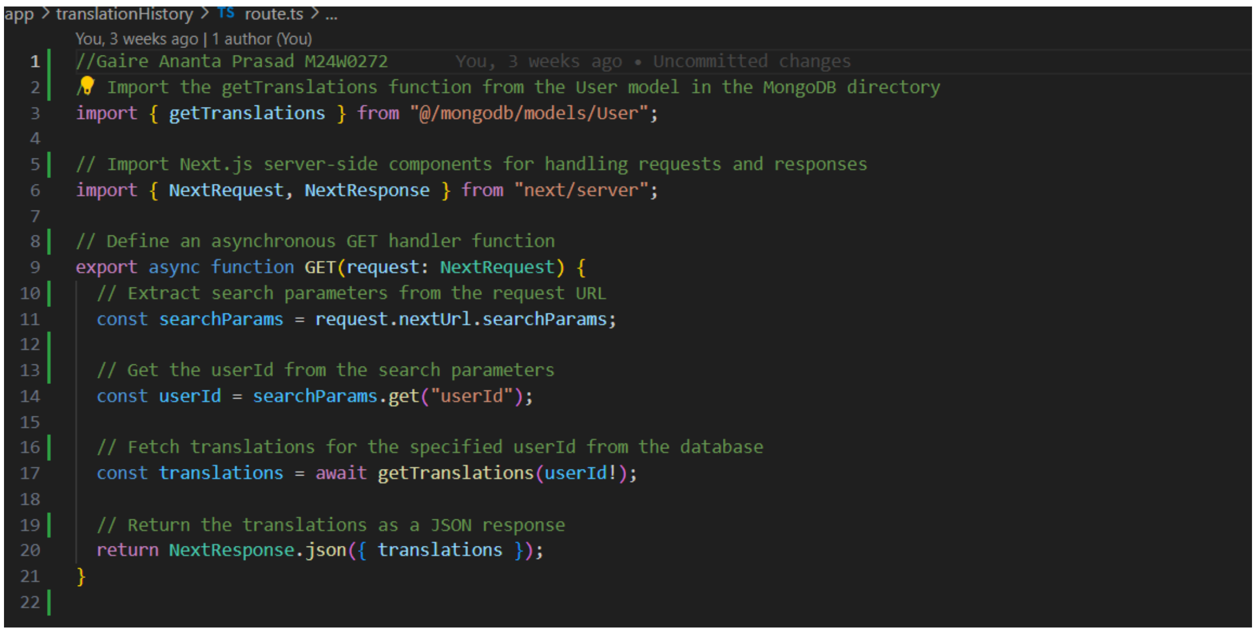

components > ClientHeader.tsx > ...

```tsx
//Gaire Ananta Prasad (M24W0272)
"use client";  // Mark this component as a Client Component

// Import necessary hooks and components from React, Clerk, Next.js, and custom components
import { useEffect, useState } from "react";
import { SignInButton, UserButton } from "@clerk/nextjs";
import Image from "next/image";
import Link from "next/link";

// Define Google colors to be used in the component
const googleColors = {
  red: "#DB4437",
  blue: "#4285F4",
  green: "#0F9D58",
  yellow: "#F4B400",
};

// Define the ClientHeader component, accepting a userId prop
function ClientHeader({ userId }: { userId: string | null }) {
  // State to hold the current time
  const [time, setTime] = useState(new Date());

  // useEffect hook to update the time every second
  useEffect(() => {
    const timer = setInterval(() => {
      setTime(new Date());
    }, 1000);

    // Cleanup the timer when the component is unmounted
    return () => clearInterval(timer);
  }, []);

  // Function to format the time as a string
  const formatTime = (date: Date) => {
    return date.toLocaleTimeString([], { hour: '2-digit', minute: '2-digit', second: '2-digit' });
  };

```

GAIRE ANANTA PRASAD M24W0272


components > ClientHeader.tsx > ...

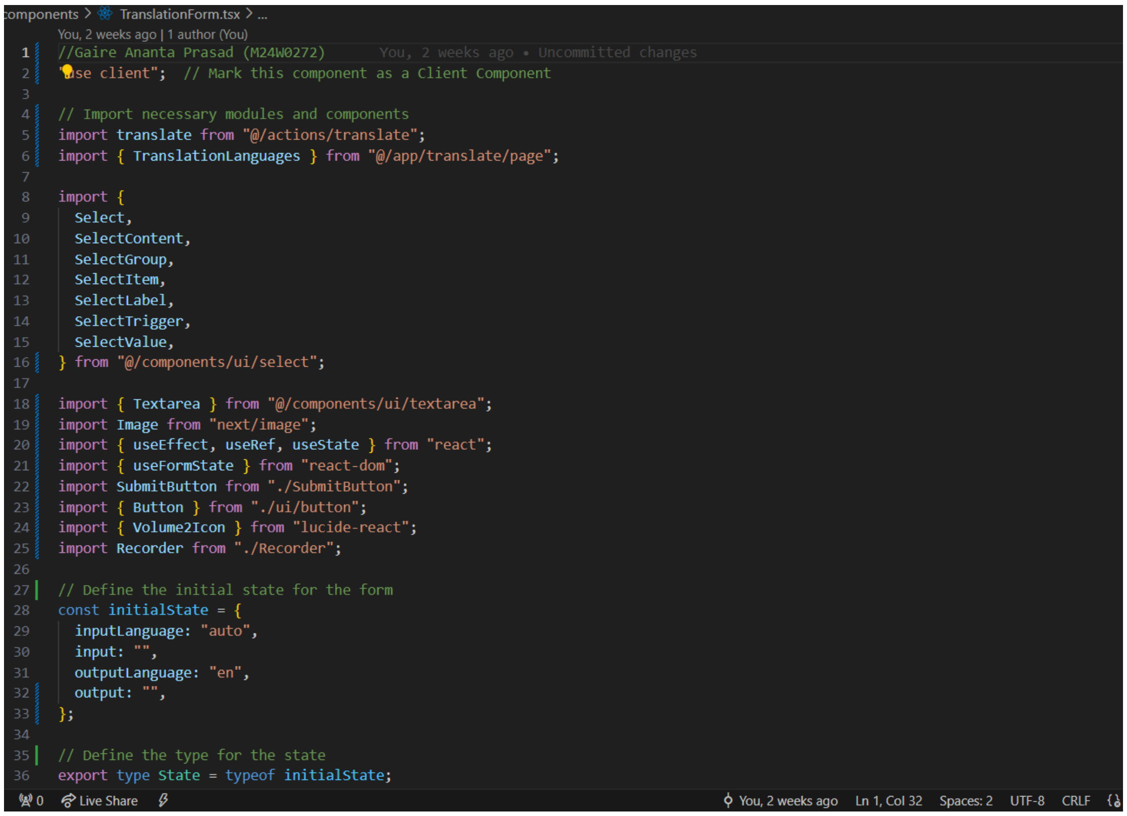
```tsx
function ClientHeader({ userId }: { userId: string | null }) {
  // Render the header component
  return (
    <header className="flex items-center justify-between px-8 border-b mb-5">
      {/* Logo section with a link to the homepage */}
      <div className="flex items-center justify-center h-20 overflow-hidden">
        <Link href="/">
          <Image
            src="https://links.papareact.com/xgu"
            alt="logo"
            width={200}
            height={100}
            className="object-contain h-32 cursor-pointer"
          />
        </Link>
      </div>

      {/* Center section displaying user name and ID */}
      <div className="flex flex-col items-center mx-4 text-center">
        <div className="font-bold text-xl">Gaire Ananta Prasad</div>
        <div className="text-sm">M24W0272</div>
      </div>

      {/* Right section with the current time and user authentication status */}
      <div className="flex items-center">
        <div className="mr-4" style={{ color: googleColors.blue }}>
          {formatTime(time)}
        </div>
        {userId ? (
          <div>
            <UserButton />
          </div>
        ) : (
          <SignInButton signUpFallbackRedirectUrl="/translate" mode="modal" />
        )}
      </div>
    </header>
```


app > translationHistory > route.ts > ...

You, 3 weeks ago | 1 author (You)

```ts
//Gaire Ananta Prasad M24W0272        You, 3 weeks ago • Uncommitted changes
// Import the getTranslations function from the User model in the MongoDB directory
import { getTranslations } from "@/mongodb/models/User";

// Import Next.js server-side components for handling requests and responses
import { NextRequest, NextResponse } from "next/server";

// Define an asynchronous GET handler function
export async function GET(request: NextRequest) {
  // Extract search parameters from the request URL
  const searchParams = request.nextUrl.searchParams;

  // Get the userId from the search parameters
  const userId = searchParams.get("userId");

  // Fetch translations for the specified userId from the database
  const translations = await getTranslations(userId!);

  // Return the translations as a JSON response
  return NextResponse.json({ translations });
}

```

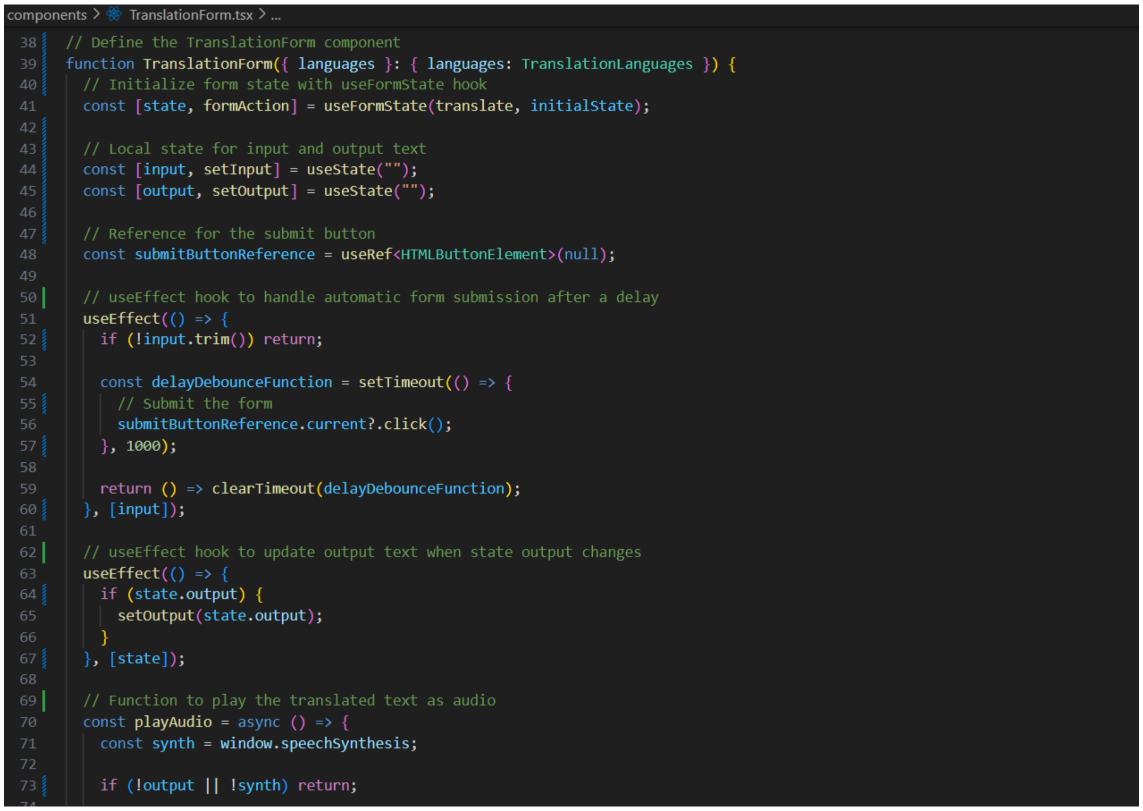
GAIRE ANANTA PRASAD M24W0272


components > TranslationForm.tsx > ...

You, 2 weeks ago | 1 author (You)

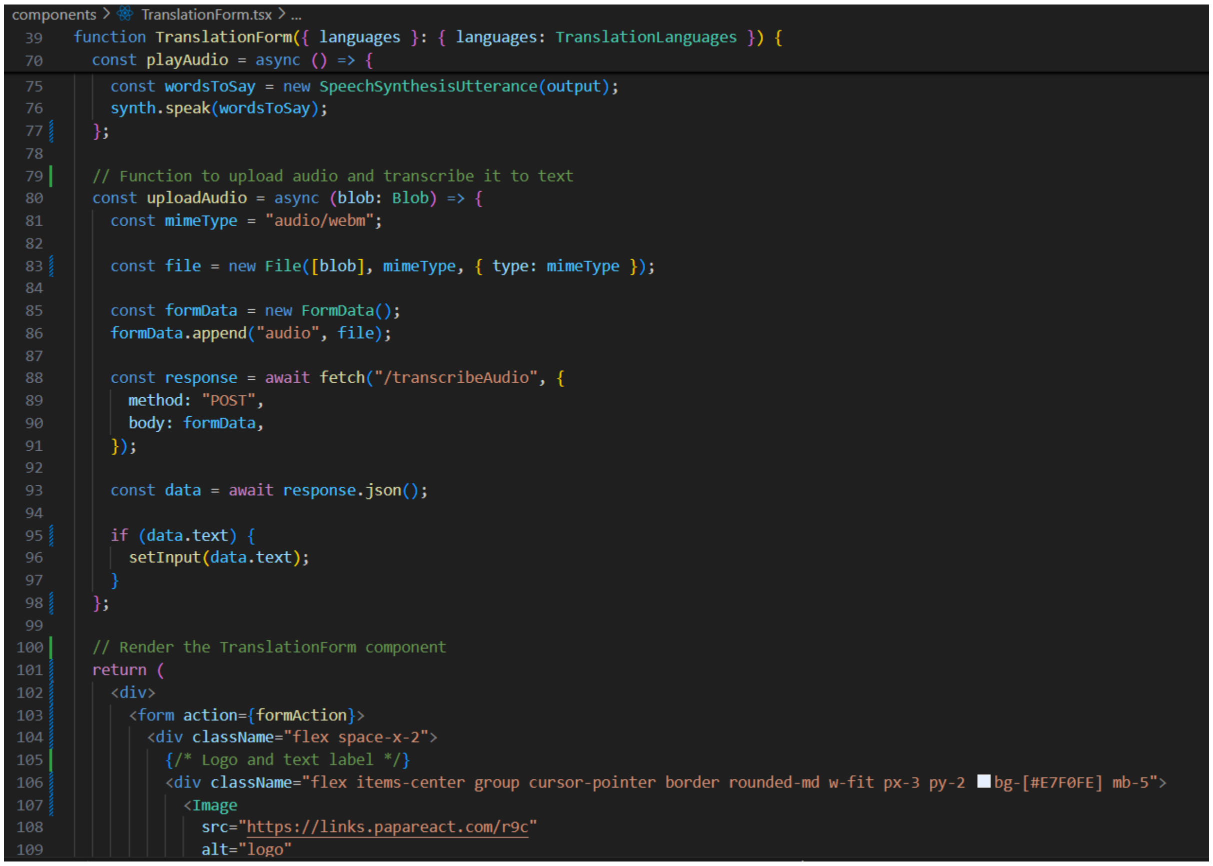
```tsx
//Gaire Ananta Prasad (M24W0272)      You, 2 weeks ago • Uncommitted changes
"use client";  // Mark this component as a Client Component

// Import necessary modules and components
import translate from "@/actions/translate";
import { TranslationLanguages } from "@/app/translate/page";

import {
  Select,
  SelectContent,
  SelectGroup,
  SelectItem,
  SelectLabel,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";

import { Textarea } from "@/components/ui/textarea";
import Image from "next/image";
import { useEffect, useRef, useState } from "react";
import { useFormState } from "react-dom";
import SubmitButton from "./SubmitButton";
import { Button } from "./ui/button";
import { Volume2Icon } from "lucide-react";
import Recorder from "./Recorder";

// Define the initial state for the form
const initialState = {
  inputLanguage: "auto",
  input: "",
  outputLanguage: "en",
  output: "",
};

// Define the type for the state
export type State = typeof initialState;
```

0 Live Share   You, 2 weeks ago Ln 1, Col 32 Spaces: 2 UTF-8 CRLF

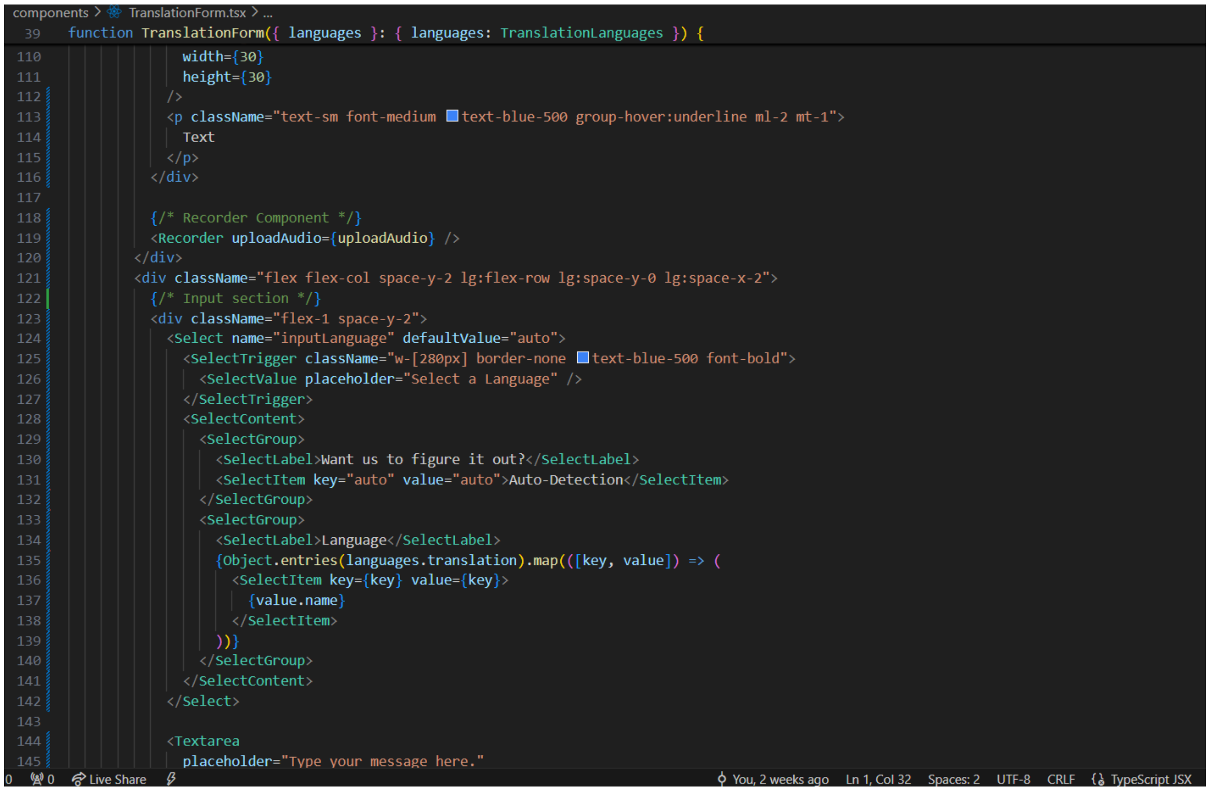

components > TranslationForm.tsx > ...

```tsx
// Define the TranslationForm component
function TranslationForm({ languages }: { languages: TranslationLanguages }) {
  // Initialize form state with useFormState hook
  const [state, formAction] = useFormState(translate, initialState);

  // Local state for input and output text
  const [input, setInput] = useState("");
  const [output, setOutput] = useState("");

  // Reference for the submit button
  const submitButtonReference = useRef<HTMLButtonElement>(null);

  // useEffect hook to handle automatic form submission after a delay
  useEffect(() => {
    if (!input.trim()) return;

    const delayDebounceFunction = setTimeout(() => {
      // Submit the form
      submitButtonReference.current?.click();
    }, 1000);

    return () => clearTimeout(delayDebounceFunction);
  }, [input]);

  // useEffect hook to update output text when state output changes
  useEffect(() => {
    if (state.output) {
      setOutput(state.output);
    }
  }, [state]);

  // Function to play the translated text as audio
  const playAudio = async () => {
    const synth = window.speechSynthesis;

    if (!output || !synth) return;
```

GAIRE ANANTA PRASAD M24W0272

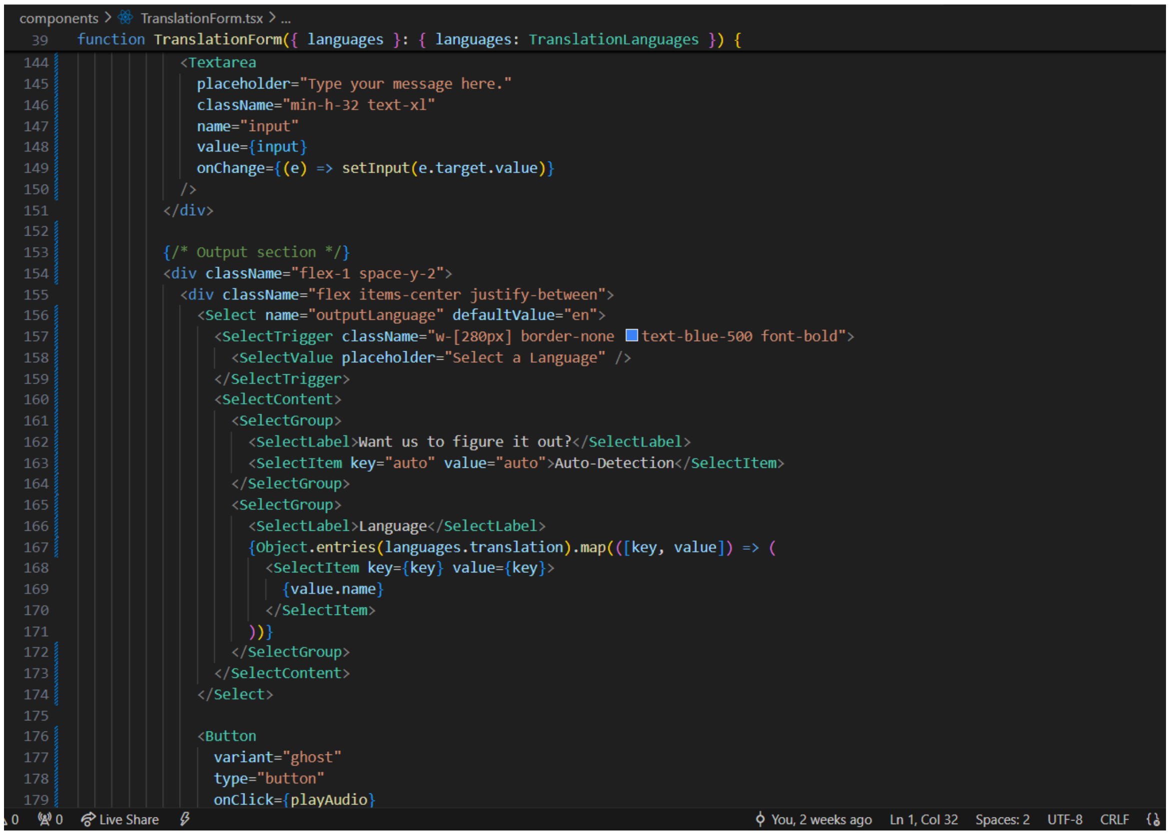


```tsx
function TranslationForm({ languages }: { languages: TranslationLanguages }) {
  const playAudio = async () => {
    const wordsToSay = new SpeechSynthesisUtterance(output);
    synth.speak(wordsToSay);
  };

  // Function to upload audio and transcribe it to text
  const uploadAudio = async (blob: Blob) => {
    const mimeType = "audio/webm";

    const file = new File([blob], mimeType, { type: mimeType });

    const formData = new FormData();
    formData.append("audio", file);

    const response = await fetch("/transcribeAudio", {
      method: "POST",
      body: formData,
    });

    const data = await response.json();

    if (data.text) {
      setInput(data.text);
    }
  };

  // Render the TranslationForm component
  return (
    <div>
      <form action={formAction}>
        <div className="flex space-x-2">
          {/* Logo and text label */}
          <div className="flex items-center group cursor-pointer border rounded-md w-fit px-3 py-2 bg-[#E7F0FE] mb-5">
            <Image
              src="https://links.papareact.com/r9c"
              alt="logo"
```


components > TranslationForm.tsx > ...

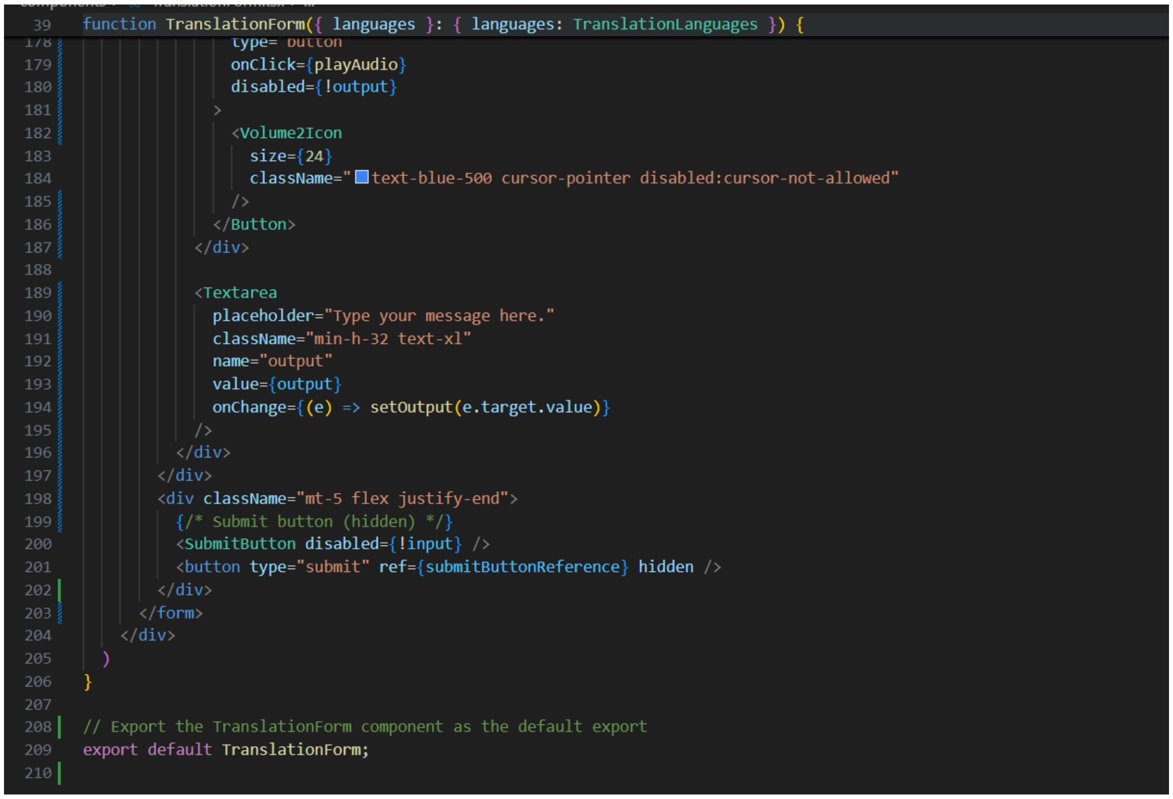
```tsx
function TranslationForm({ languages }: { languages: TranslationLanguages }) {
              width={30}
              height={30}
            />
            <p className="text-sm font-medium text-blue-500 group-hover:underline ml-2 mt-1">
              Text
            </p>
          </div>

          {/* Recorder Component */}
          <Recorder uploadAudio={uploadAudio} />
        </div>
        <div className="flex flex-col space-y-2 lg:flex-row lg:space-y-0 lg:space-x-2">
          {/* Input section */}
          <div className="flex-1 space-y-2">
            <Select name="inputLanguage" defaultValue="auto">
              <SelectTrigger className="w-[280px] border-none text-blue-500 font-bold">
                <SelectValue placeholder="Select a Language" />
              </SelectTrigger>
              <SelectContent>
                <SelectGroup>
                  <SelectLabel>Want us to figure it out?</SelectLabel>
                  <SelectItem key="auto" value="auto">Auto-Detection</SelectItem>
                </SelectGroup>
                <SelectGroup>
                  <SelectLabel>Language</SelectLabel>
                  {Object.entries(languages.translation).map(([key, value]) => (
                    <SelectItem key={key} value={key}>
                      {value.name}
                    </SelectItem>
                  ))}
                </SelectGroup>
              </SelectContent>
            </Select>

            <Textarea
              placeholder="Type your message here."
```
0 0 Live Share
You, 2 weeks ago Ln 1, Col 32 Spaces: 2 UTF-8 CRLF TypeScript JSX

GAIRE ANANTA PRASAD M24W0272

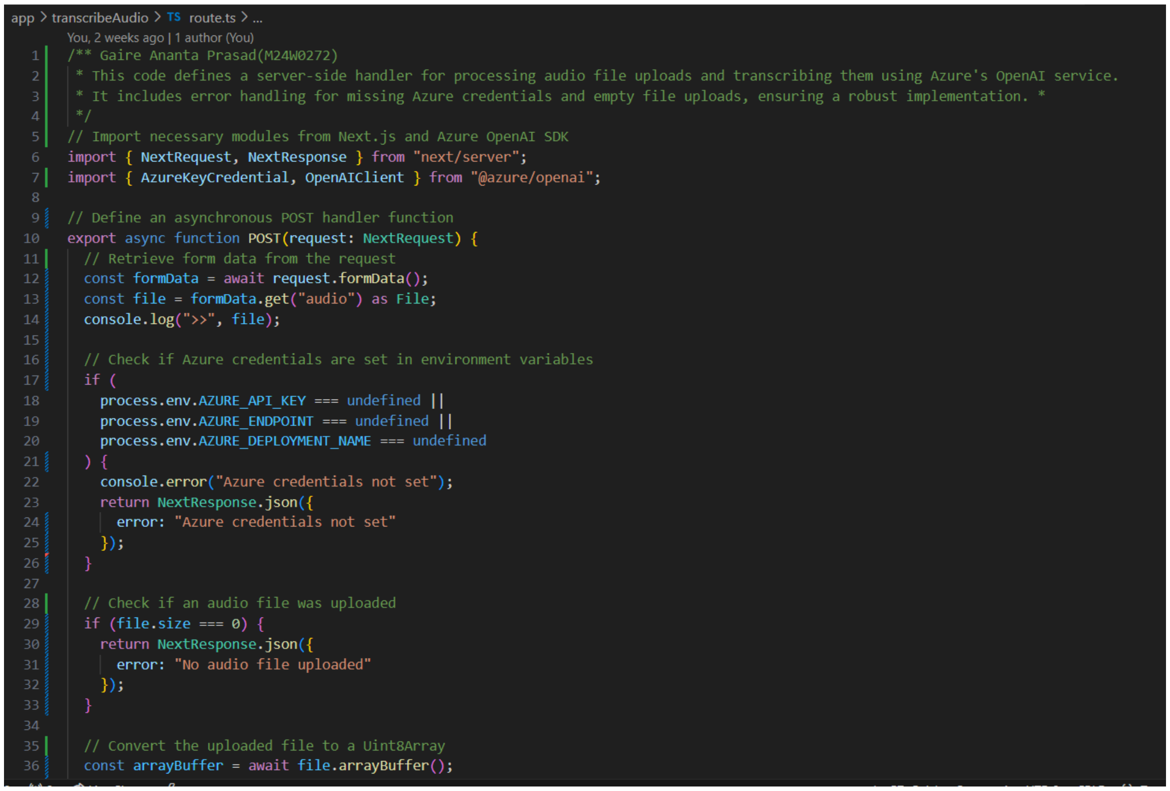

components > TranslationForm.tsx > ...

```tsx
function TranslationForm({ languages }: { languages: TranslationLanguages }) {
            <Textarea
              placeholder="Type your message here."
              className="min-h-32 text-xl"
              name="input"
              value={input}
              onChange={(e) => setInput(e.target.value)}
            />
          </div>

          {/* Output section */}
          <div className="flex-1 space-y-2">
            <div className="flex items-center justify-between">
              <Select name="outputLanguage" defaultValue="en">
                <SelectTrigger className="w-[280px] border-none text-blue-500 font-bold">
                  <SelectValue placeholder="Select a Language" />
                </SelectTrigger>
                <SelectContent>
                  <SelectGroup>
                    <SelectLabel>Want us to figure it out?</SelectLabel>
                    <SelectItem key="auto" value="auto">Auto-Detection</SelectItem>
                  </SelectGroup>
                  <SelectGroup>
                    <SelectLabel>Language</SelectLabel>
                    {Object.entries(languages.translation).map(([key, value]) => (
                      <SelectItem key={key} value={key}>
                        {value.name}
                      </SelectItem>
                    ))}
                  </SelectGroup>
                </SelectContent>
              </Select>

              <Button
                variant="ghost"
                type="button"
                onClick={playAudio}
```

You, 2 weeks ago Ln 1, Col 32 Spaces: 2 UTF-8 CRLF


```tsx
function TranslationForm({ languages }: { languages: TranslationLanguages }) {
                type="button"
                onClick={playAudio}
                disabled={!output}
              >
                <Volume2Icon
                  size={24}
                  className="text-blue-500 cursor-pointer disabled:cursor-not-allowed"
                />
              </Button>
            </div>

            <Textarea
              placeholder="Type your message here."
              className="min-h-32 text-xl"
              name="output"
              value={output}
              onChange={(e) => setOutput(e.target.value)}
            />
          </div>
        </div>
        <div className="mt-5 flex justify-end">
          {/* Submit button (hidden) */}
          <SubmitButton disabled={!input} />
          <button type="submit" ref={submitButtonReference} hidden />
        </div>
      </form>
    </div>
  )
}

// Export the TranslationForm component as the default export
export default TranslationForm;
```

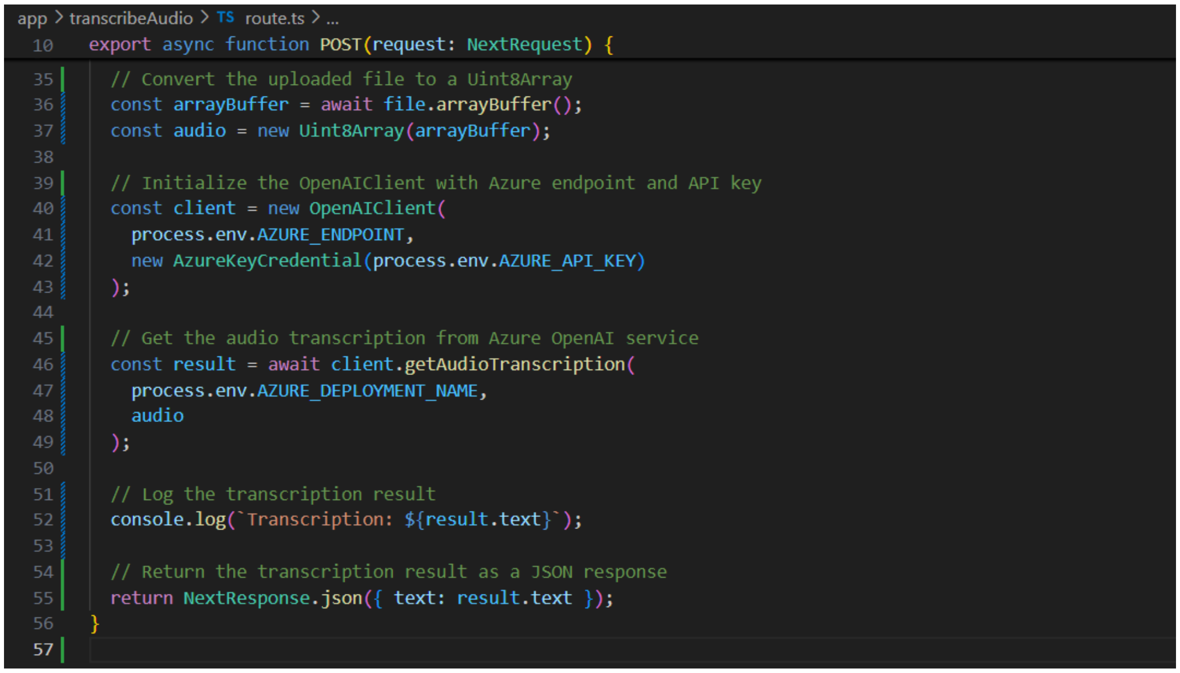
GAIRE ANANTA PRASAD M24W0272



app > transcribeAudio > TS route.ts > ...

You, 2 weeks ago | 1 author (You)

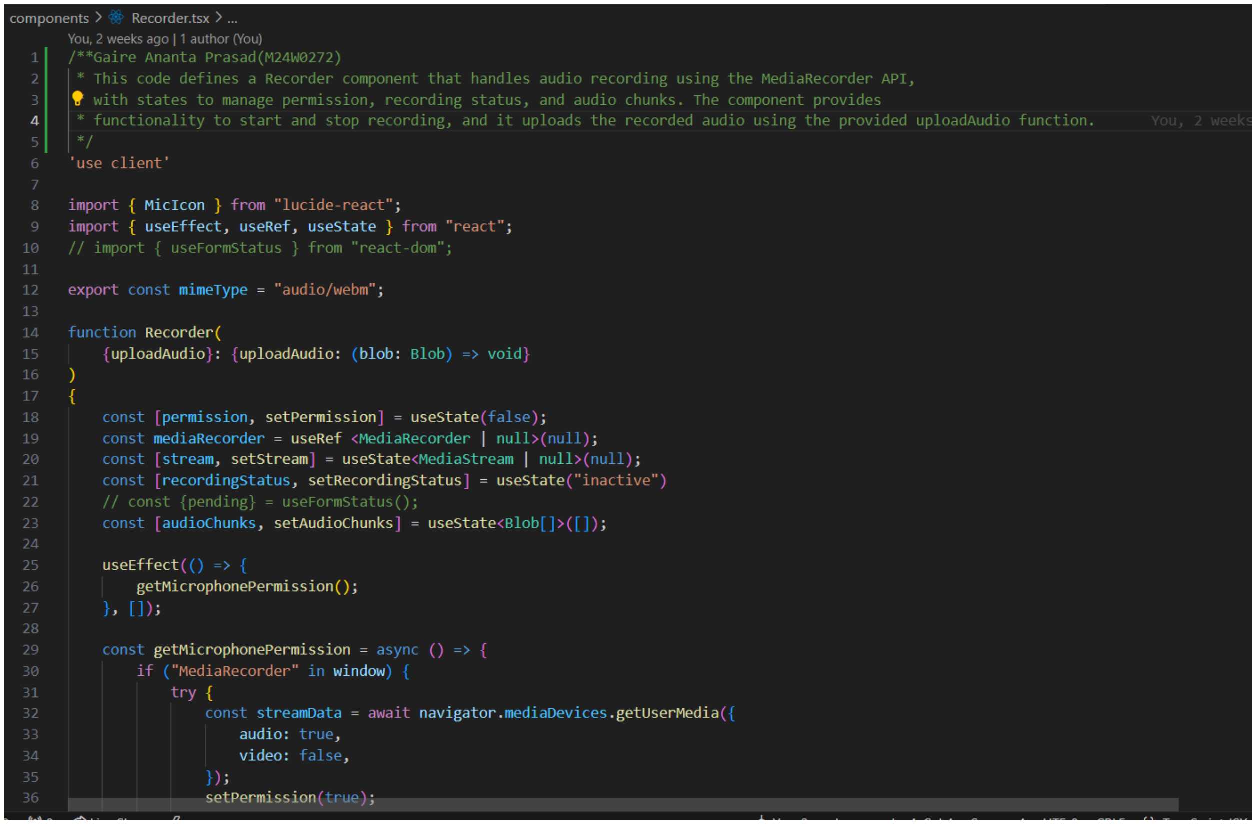
```ts
/** Gaire Ananta Prasad(M24W0272)
 * This code defines a server-side handler for processing audio file uploads and transcribing them using Azure's OpenAI service.
 * It includes error handling for missing Azure credentials and empty file uploads, ensuring a robust implementation. *
 */
// Import necessary modules from Next.js and Azure OpenAI SDK
import { NextRequest, NextResponse } from "next/server";
import { AzureKeyCredential, OpenAIClient } from "@azure/openai";

// Define an asynchronous POST handler function
export async function POST(request: NextRequest) {
  // Retrieve form data from the request
  const formData = await request.formData();
  const file = formData.get("audio") as File;
  console.log(">>", file);

  // Check if Azure credentials are set in environment variables
  if (
    process.env.AZURE_API_KEY === undefined ||
    process.env.AZURE_ENDPOINT === undefined ||
    process.env.AZURE_DEPLOYMENT_NAME === undefined
  ) {
    console.error("Azure credentials not set");
    return NextResponse.json({
      error: "Azure credentials not set"
    });
  }

  // Check if an audio file was uploaded
  if (file.size === 0) {
    return NextResponse.json({
      error: "No audio file uploaded"
    });
  }

  // Convert the uploaded file to a Uint8Array
  const arrayBuffer = await file.arrayBuffer();
```



app > transcribeAudio > TS route.ts > ...

```ts
export async function POST(request: NextRequest) {
  // Convert the uploaded file to a Uint8Array
  const arrayBuffer = await file.arrayBuffer();
  const audio = new Uint8Array(arrayBuffer);

  // Initialize the OpenAIClient with Azure endpoint and API key
  const client = new OpenAIClient(
    process.env.AZURE_ENDPOINT,
    new AzureKeyCredential(process.env.AZURE_API_KEY)
  );

  // Get the audio transcription from Azure OpenAI service
  const result = await client.getAudioTranscription(
    process.env.AZURE_DEPLOYMENT_NAME,
    audio
  );

  // Log the transcription result
  console.log(`Transcription: ${result.text}`);

  // Return the transcription result as a JSON response
  return NextResponse.json({ text: result.text });
}
```

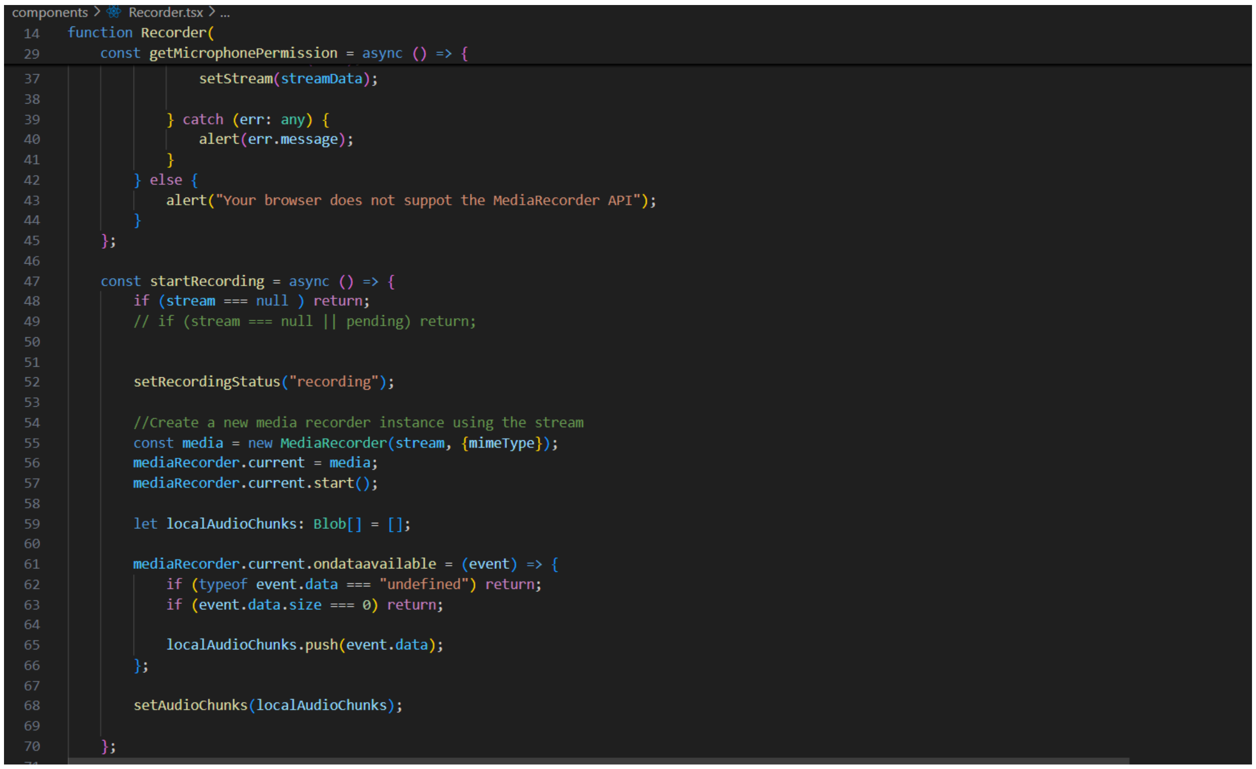
GAIRE ANANTA PRASAD M24W0272



components > Recorder.tsx > ...

You, 2 weeks ago | 1 author (You)

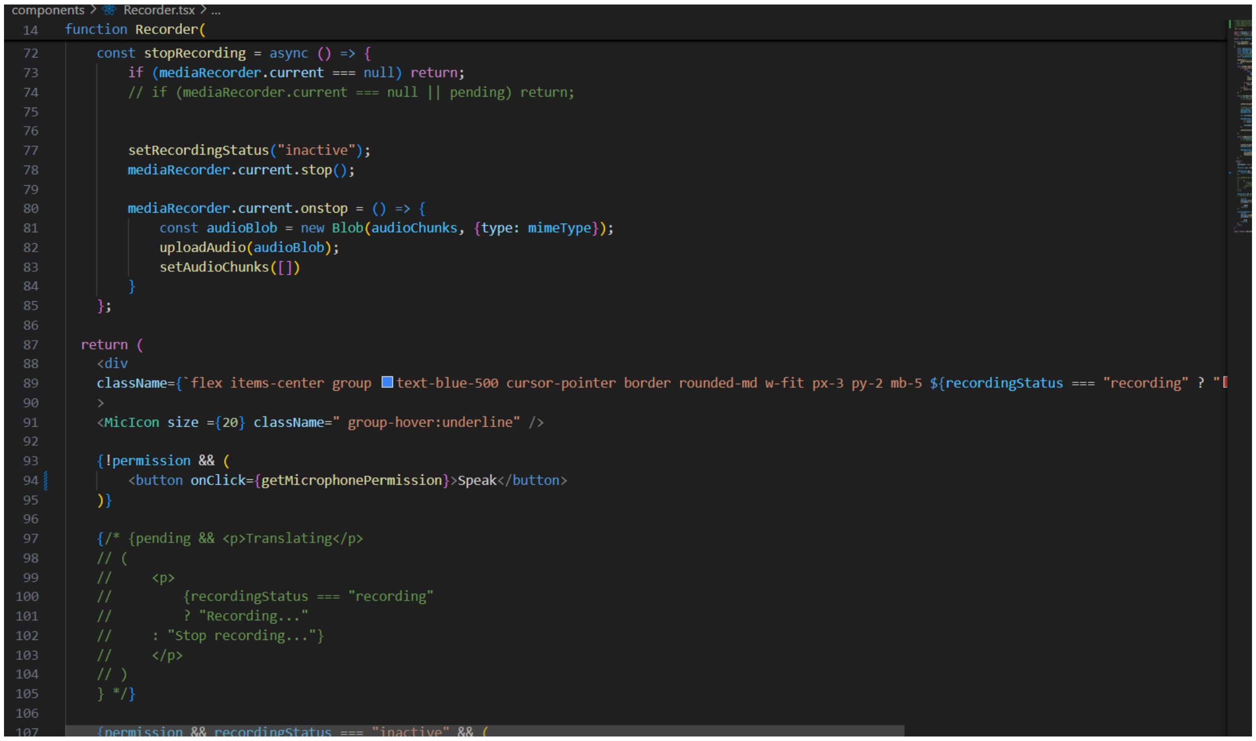
```tsx
/**Gaire Ananta Prasad(M24W0272)
 * This code defines a Recorder component that handles audio recording using the MediaRecorder API,
 * with states to manage permission, recording status, and audio chunks. The component provides
 * functionality to start and stop recording, and it uploads the recorded audio using the provided uploadAudio function.     You, 2 weeks
 */
'use client'

import { MicIcon } from "lucide-react";
import { useEffect, useRef, useState } from "react";
// import { useFormStatus } from "react-dom";

export const mimeType = "audio/webm";

function Recorder(
    {uploadAudio}: {uploadAudio: (blob: Blob) => void}
)
{
    const [permission, setPermission] = useState(false);
    const mediaRecorder = useRef <MediaRecorder | null>(null);
    const [stream, setStream] = useState<MediaStream | null>(null);
    const [recordingStatus, setRecordingStatus] = useState("inactive")
    // const {pending} = useFormStatus();
    const [audioChunks, setAudioChunks] = useState<Blob[]>([]);

    useEffect(() => {
        getMicrophonePermission();
    }, []);

    const getMicrophonePermission = async () => {
        if ("MediaRecorder" in window) {
            try {
                const streamData = await navigator.mediaDevices.getUserMedia({
                    audio: true,
                    video: false,
                });
                setPermission(true);
```



components > Recorder.tsx > ...

```tsx
function Recorder(
    const getMicrophonePermission = async () => {
                setStream(streamData);

            } catch (err: any) {
                alert(err.message);
            }
        } else {
            alert("Your browser does not suppot the MediaRecorder API");
        }
    };

    const startRecording = async () => {
        if (stream === null ) return;
        // if (stream === null || pending) return;


        setRecordingStatus("recording");

        //Create a new media recorder instance using the stream
        const media = new MediaRecorder(stream, {mimeType});
        mediaRecorder.current = media;
        mediaRecorder.current.start();

        let localAudioChunks: Blob[] = [];

        mediaRecorder.current.ondataavailable = (event) => {
            if (typeof event.data === "undefined") return;
            if (event.data.size === 0) return;

            localAudioChunks.push(event.data);
        };

        setAudioChunks(localAudioChunks);

    };
```

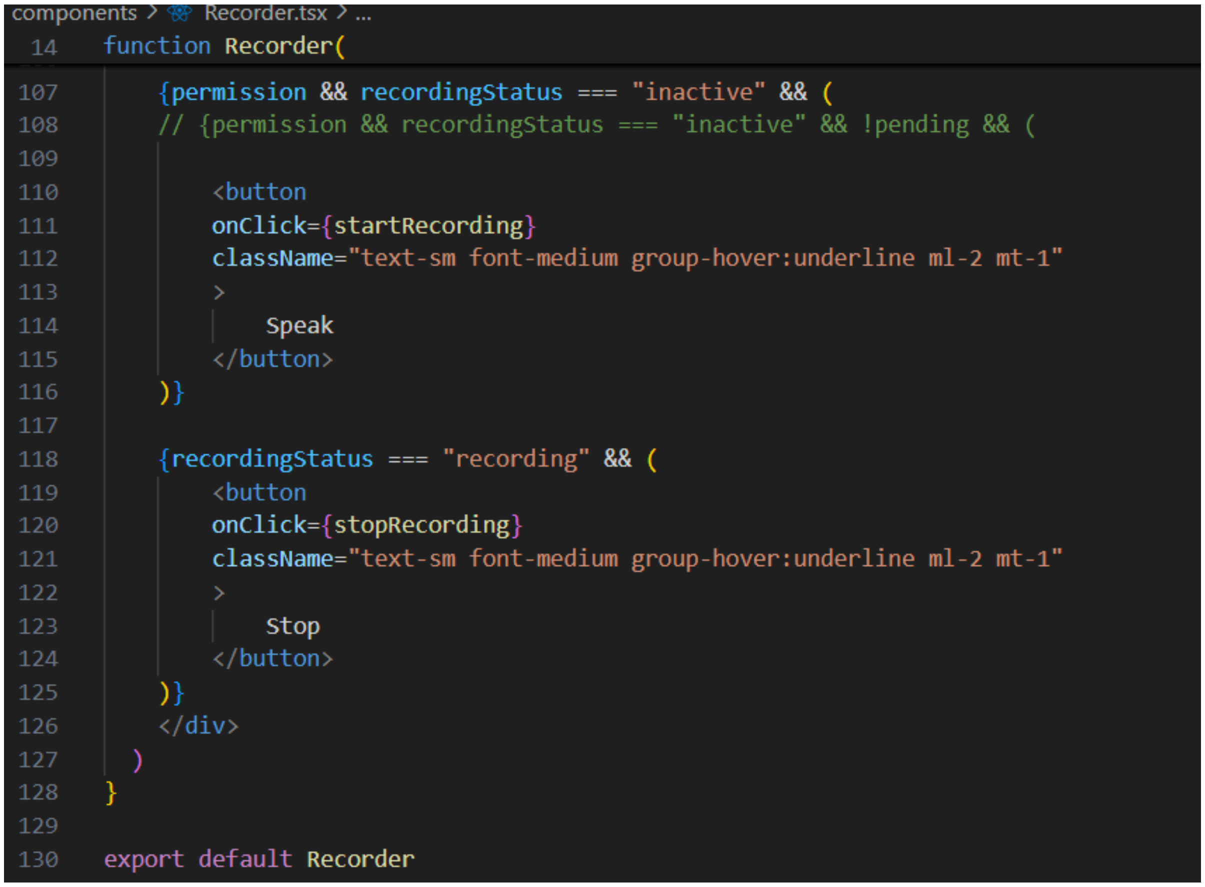
GAIRE ANANTA PRASAD M24W0272



components > Recorder.tsx > ...

```
function Recorder(
```

```
    const stopRecording = async () => {
        if (mediaRecorder.current === null) return;
        // if (mediaRecorder.current === null || pending) return;


        setRecordingStatus("inactive");
        mediaRecorder.current.stop();

        mediaRecorder.current.onstop = () => {
            const audioBlob = new Blob(audioChunks, {type: mimeType});
            uploadAudio(audioBlob);
            setAudioChunks([])
        }
    };

  return (
    <div
    className={`flex items-center group text-blue-500 cursor-pointer border rounded-md w-fit px-3 py-2 mb-5 ${recordingStatus === "recording" ? "
    >
    <MicIcon size ={20} className=" group-hover:underline" />

    {!permission && (
        <button onClick={getMicrophonePermission}>Speak</button>
    )}

    {/* {pending && <p>Translating</p>
    // (
    //     <p>
    //         {recordingStatus === "recording"
    //         ? "Recording..."
    //     : "Stop recording..."}
    //     </p>
    // )
    } */}

```

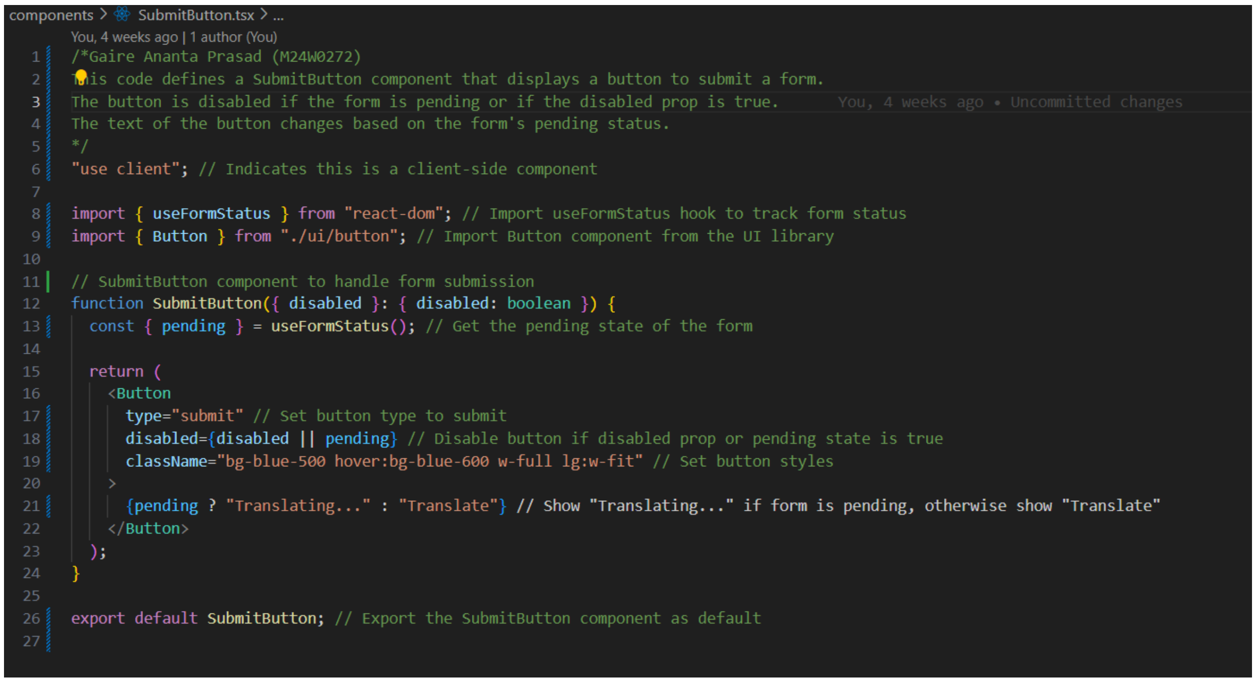


components > Recorder.tsx > ...

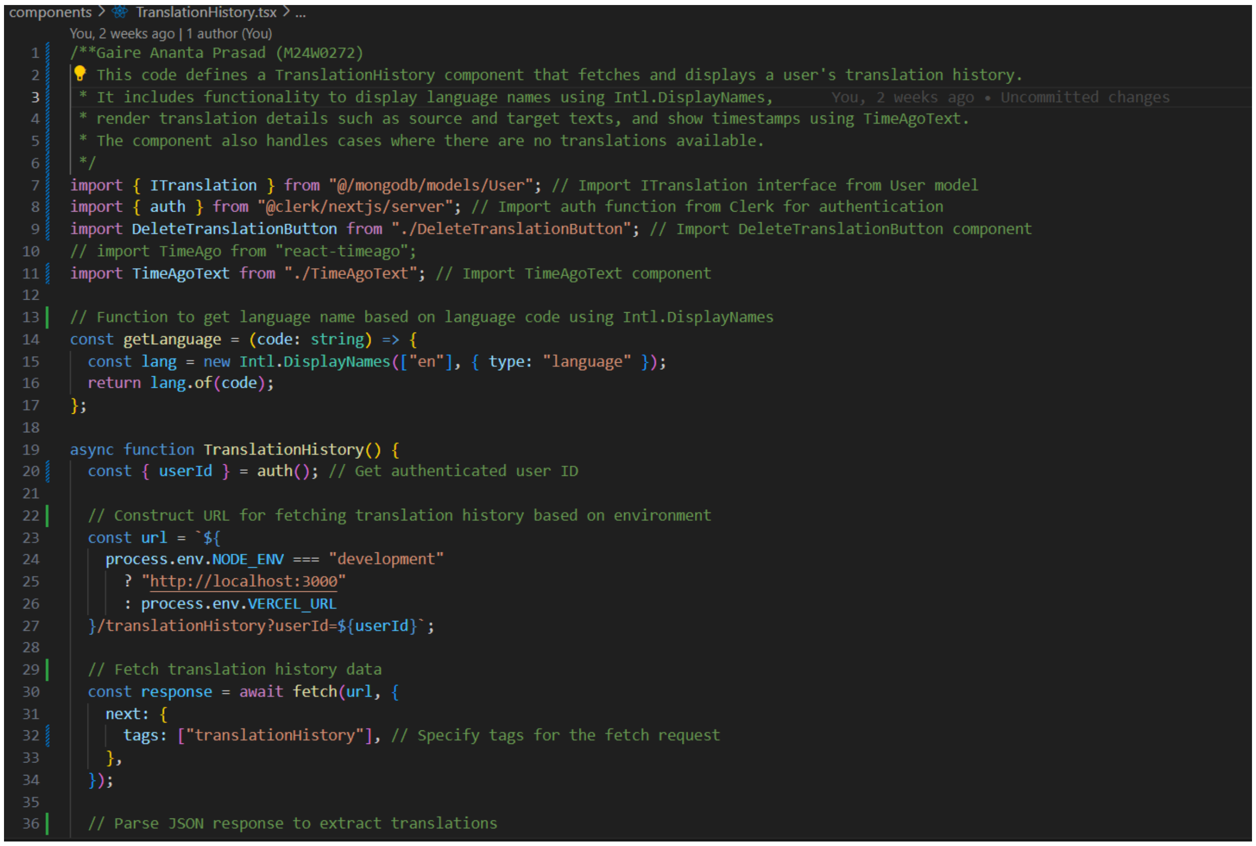
```
function Recorder(
```

```
    {permission && recordingStatus === "inactive" && (
    // {permission && recordingStatus === "inactive" && !pending && (

        <button
        onClick={startRecording}
        className="text-sm font-medium group-hover:underline ml-2 mt-1"
        >
            Speak
        </button>
    )}

    {recordingStatus === "recording" && (
        <button
        onClick={stopRecording}
        className="text-sm font-medium group-hover:underline ml-2 mt-1"
        >
            Stop
        </button>
    )}
    </div>
  )
}

export default Recorder
```

GAIRE ANANTA PRASAD M24W0272

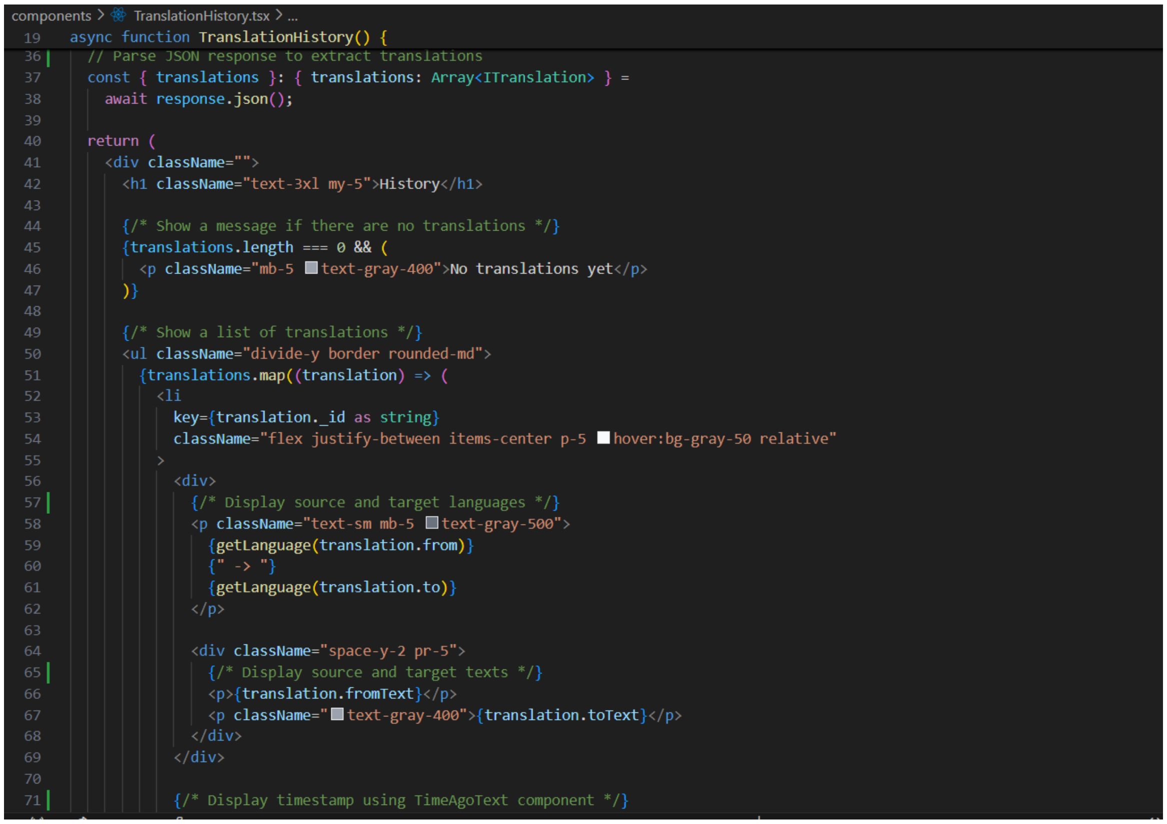

components > SubmitButton.tsx > ...
You, 4 weeks ago | 1 author (You)

```tsx
/*Gaire Ananta Prasad (M24W0272)
This code defines a SubmitButton component that displays a button to submit a form.
The button is disabled if the form is pending or if the disabled prop is true.      You, 4 weeks ago • Uncommitted changes
The text of the button changes based on the form's pending status.
*/
"use client"; // Indicates this is a client-side component

import { useFormStatus } from "react-dom"; // Import useFormStatus hook to track form status
import { Button } from "./ui/button"; // Import Button component from the UI library

// SubmitButton component to handle form submission
function SubmitButton({ disabled }: { disabled: boolean }) {
  const { pending } = useFormStatus(); // Get the pending state of the form

  return (
    <Button
      type="submit" // Set button type to submit
      disabled={disabled || pending} // Disable button if disabled prop or pending state is true
      className="bg-blue-500 hover:bg-blue-600 w-full lg:w-fit" // Set button styles
    >
      {pending ? "Translating..." : "Translate"} // Show "Translating..." if form is pending, otherwise show "Translate"
    </Button>
  );
}

export default SubmitButton; // Export the SubmitButton component as default
```

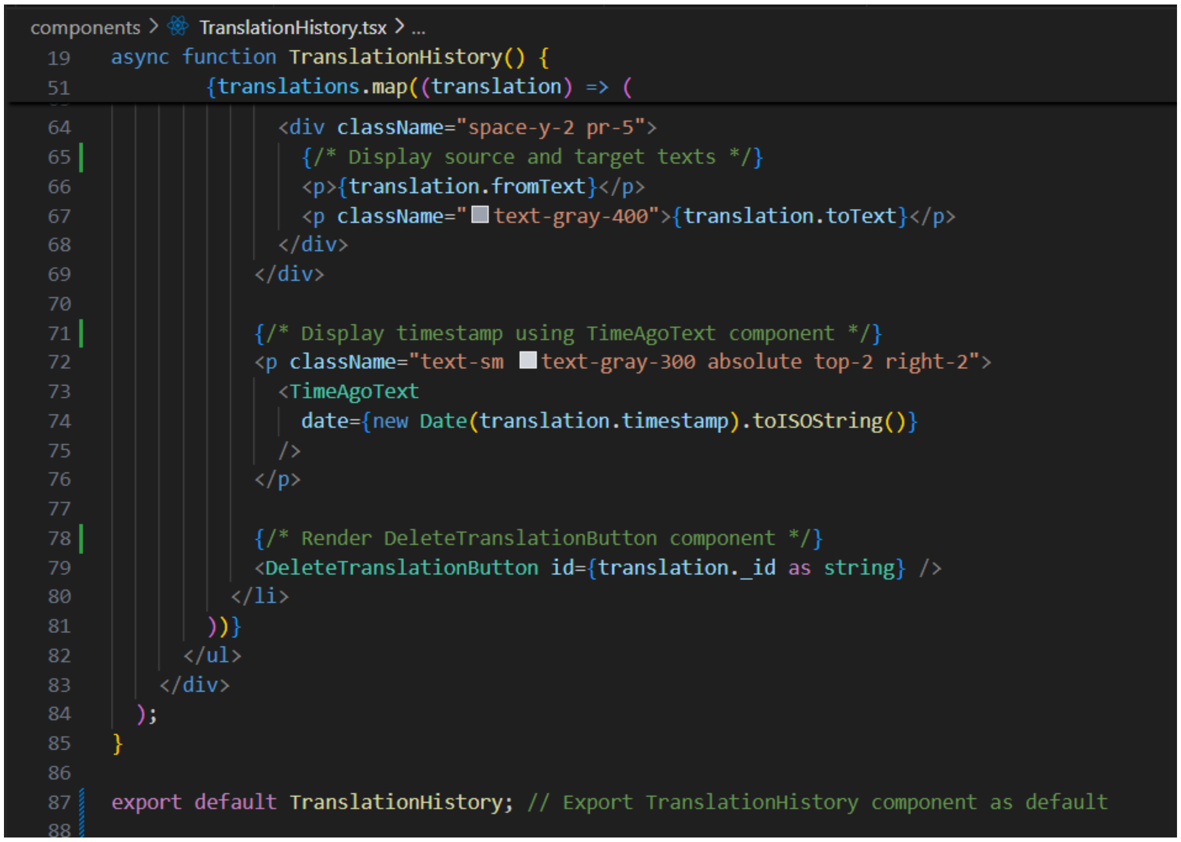

components > TranslationHistory.tsx > ...
You, 2 weeks ago | 1 author (You)

```tsx
/**Gaire Ananta Prasad (M24W0272)
 * This code defines a TranslationHistory component that fetches and displays a user's translation history.
 * It includes functionality to display language names using Intl.DisplayNames,      You, 2 weeks ago • Uncommitted changes
 * render translation details such as source and target texts, and show timestamps using TimeAgoText.
 * The component also handles cases where there are no translations available.
 */
import { ITranslation } from "@/mongodb/models/User"; // Import ITranslation interface from User model
import { auth } from "@clerk/nextjs/server"; // Import auth function from Clerk for authentication
import DeleteTranslationButton from "./DeleteTranslationButton"; // Import DeleteTranslationButton component
// import TimeAgo from "react-timeago";
import TimeAgoText from "./TimeAgoText"; // Import TimeAgoText component

// Function to get language name based on language code using Intl.DisplayNames
const getLanguage = (code: string) => {
  const lang = new Intl.DisplayNames(["en"], { type: "language" });
  return lang.of(code);
};

async function TranslationHistory() {
  const { userId } = auth(); // Get authenticated user ID

  // Construct URL for fetching translation history based on environment
  const url = `${
    process.env.NODE_ENV === "development"
      ? "http://localhost:3000"
      : process.env.VERCEL_URL
  }/translationHistory?userId=${userId}`;

  // Fetch translation history data
  const response = await fetch(url, {
    next: {
      tags: ["translationHistory"], // Specify tags for the fetch request
    },
  });

  // Parse JSON response to extract translations
```

GAIRE ANANTA PRASAD M24W0272

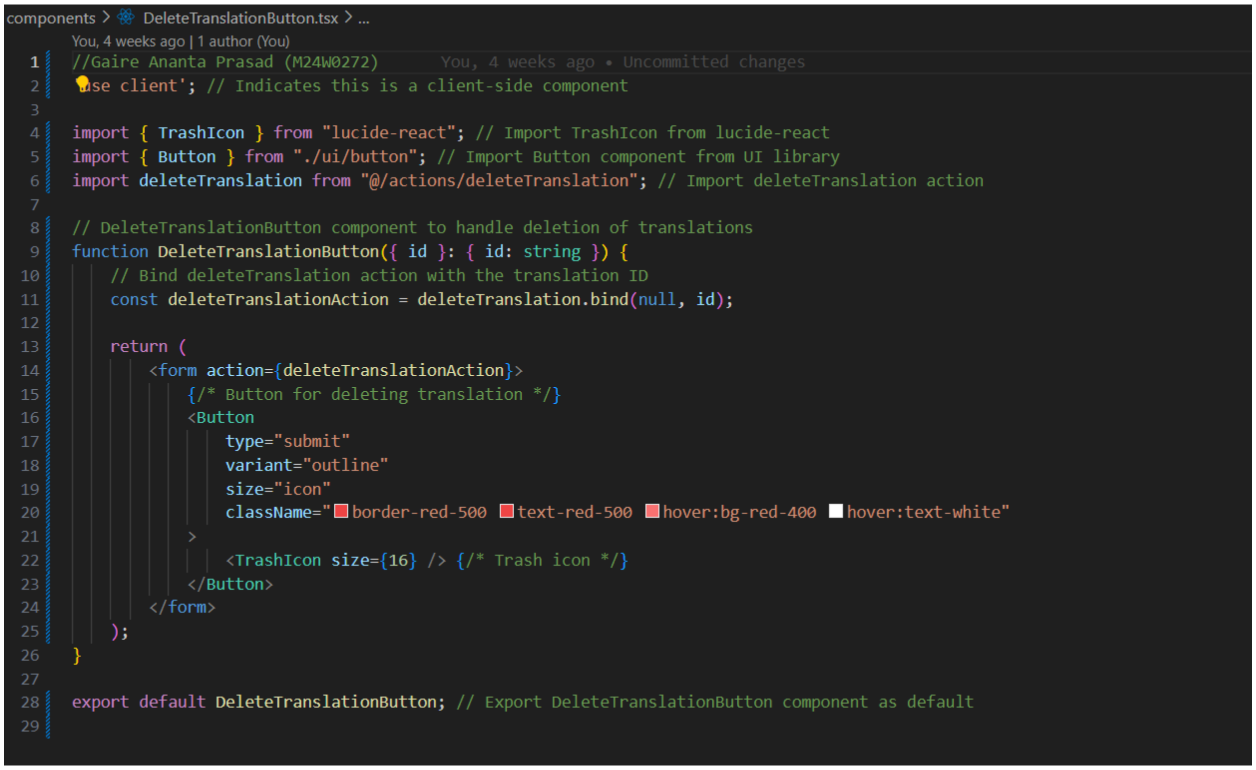


```
components > TranslationHistory.tsx > ...
async function TranslationHistory() {
  // Parse JSON response to extract translations
  const { translations }: { translations: Array<ITranslation> } =
    await response.json();

  return (
    <div className="">
      <h1 className="text-3xl my-5">History</h1>

      {/* Show a message if there are no translations */}
      {translations.length === 0 && (
        <p className="mb-5 text-gray-400">No translations yet</p>
      )}

      {/* Show a list of translations */}
      <ul className="divide-y border rounded-md">
        {translations.map((translation) => (
          <li
            key={translation._id as string}
            className="flex justify-between items-center p-5 hover:bg-gray-50 relative"
          >
            <div>
              {/* Display source and target languages */}
              <p className="text-sm mb-5 text-gray-500">
                {getLanguage(translation.from)}
                {" -> "}
                {getLanguage(translation.to)}
              </p>

              <div className="space-y-2 pr-5">
                {/* Display source and target texts */}
                <p>{translation.fromText}</p>
                <p className="text-gray-400">{translation.toText}</p>
              </div>
            </div>

            {/* Display timestamp using TimeAgoText component */}
```



```
components > TranslationHistory.tsx > ...
async function TranslationHistory() {
        {translations.map((translation) => (
              <div className="space-y-2 pr-5">
                {/* Display source and target texts */}
                <p>{translation.fromText}</p>
                <p className="text-gray-400">{translation.toText}</p>
              </div>
            </div>

            {/* Display timestamp using TimeAgoText component */}
            <p className="text-sm text-gray-300 absolute top-2 right-2">
              <TimeAgoText
                date={new Date(translation.timestamp).toISOString()}
              />
            </p>

            {/* Render DeleteTranslationButton component */}
            <DeleteTranslationButton id={translation._id as string} />
          </li>
        ))}
      </ul>
    </div>
  );
}

export default TranslationHistory; // Export TranslationHistory component as default
```

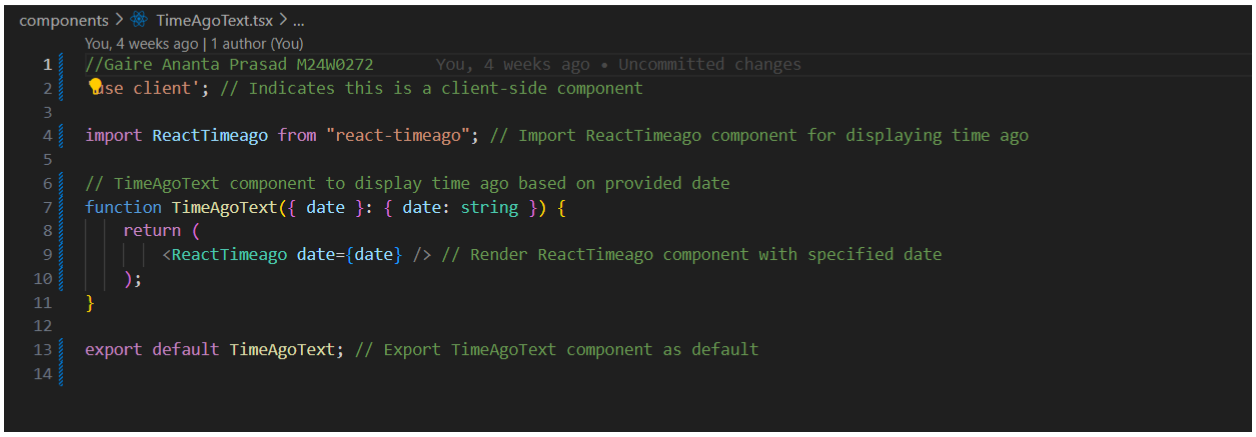
GAIRE ANANTA PRASAD M24W0272



components > DeleteTranslationButton.tsx > ...

You, 4 weeks ago | 1 author (You)

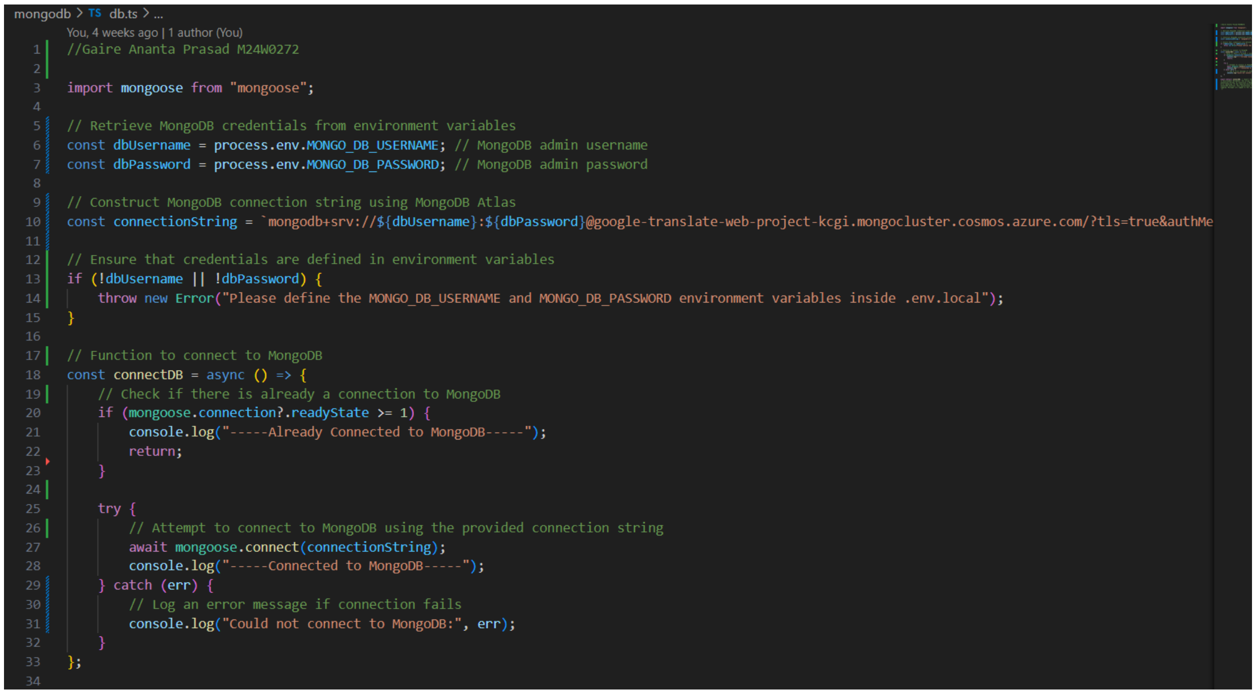
```tsx
//Gaire Ananta Prasad (M24W0272)       You, 4 weeks ago • Uncommitted changes
'use client'; // Indicates this is a client-side component

import { TrashIcon } from "lucide-react"; // Import TrashIcon from lucide-react
import { Button } from "./ui/button"; // Import Button component from UI library
import deleteTranslation from "@/actions/deleteTranslation"; // Import deleteTranslation action

// DeleteTranslationButton component to handle deletion of translations
function DeleteTranslationButton({ id }: { id: string }) {
    // Bind deleteTranslation action with the translation ID
    const deleteTranslationAction = deleteTranslation.bind(null, id);

    return (
        <form action={deleteTranslationAction}>
            {/* Button for deleting translation */}
            <Button
                type="submit"
                variant="outline"
                size="icon"
                className="border-red-500 text-red-500 hover:bg-red-400 hover:text-white"
            >
                <TrashIcon size={16} /> {/* Trash icon */}
            </Button>
        </form>
    );
}

export default DeleteTranslationButton; // Export DeleteTranslationButton component as default

```



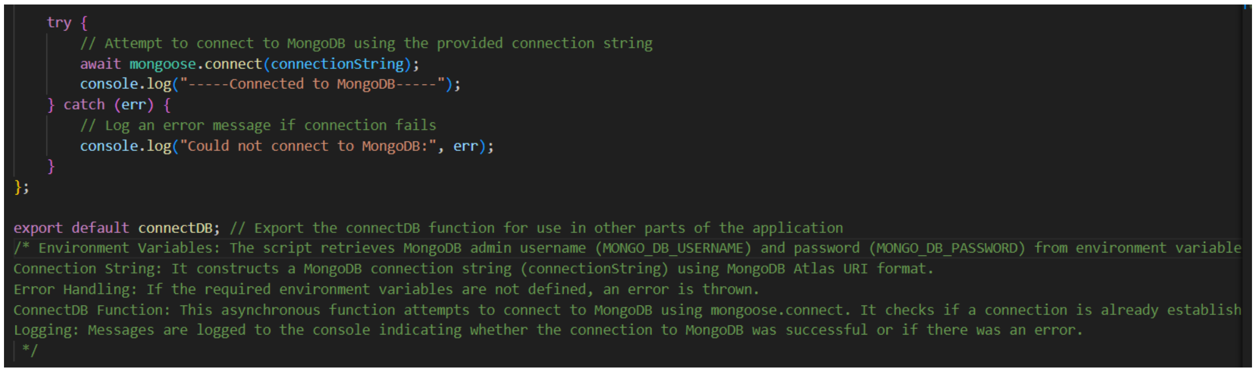
components > TimeAgoText.tsx > ...

You, 4 weeks ago | 1 author (You)

```tsx
//Gaire Ananta Prasad M24W0272       You, 4 weeks ago • Uncommitted changes
'use client'; // Indicates this is a client-side component

import ReactTimeago from "react-timeago"; // Import ReactTimeago component for displaying time ago

// TimeAgoText component to display time ago based on provided date
function TimeAgoText({ date }: { date: string }) {
    return (
        <ReactTimeago date={date} /> // Render ReactTimeago component with specified date
    );
}

export default TimeAgoText; // Export TimeAgoText component as default

```

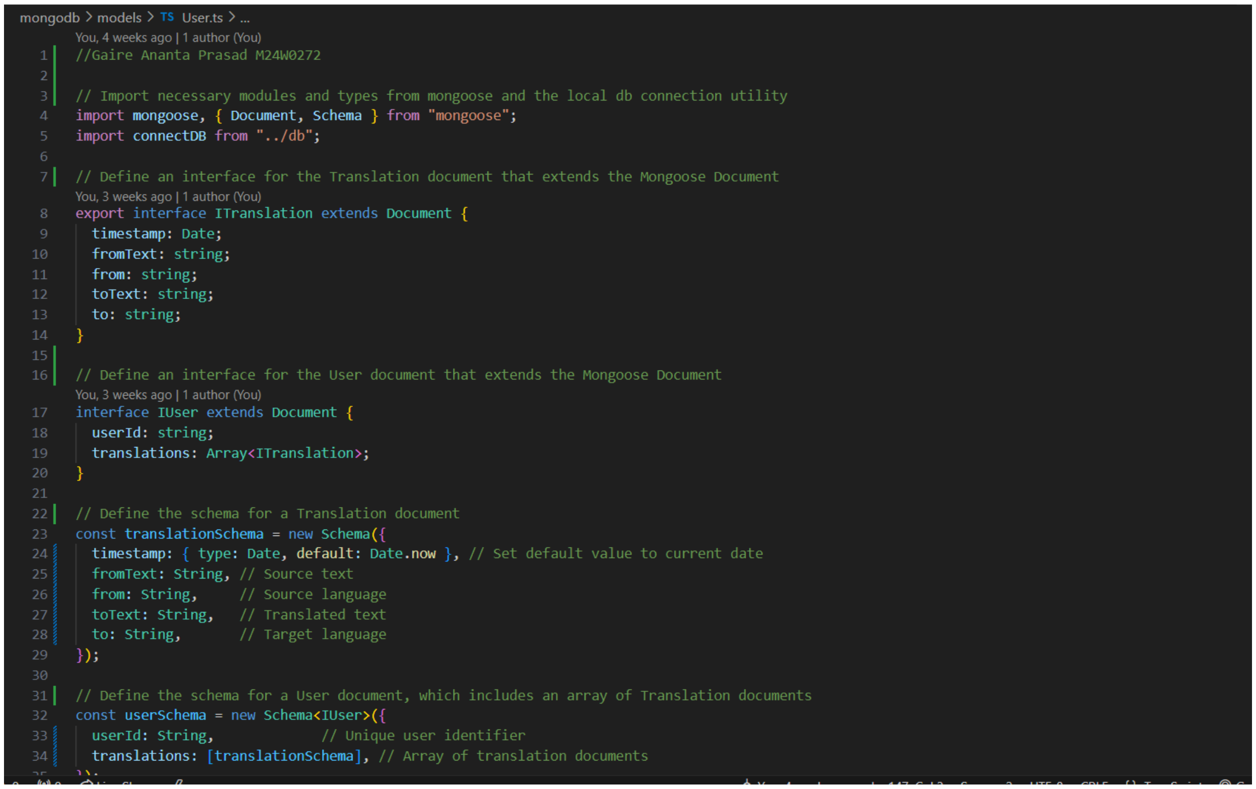


```
mongodb > TS db.ts > ...
You, 4 weeks ago | 1 author (You)
//Gaire Ananta Prasad M24W0272

import mongoose from "mongoose";

// Retrieve MongoDB credentials from environment variables
const dbUsername = process.env.MONGO_DB_USERNAME; // MongoDB admin username
const dbPassword = process.env.MONGO_DB_PASSWORD; // MongoDB admin password

// Construct MongoDB connection string using MongoDB Atlas
const connectionString = `mongodb+srv://${dbUsername}:${dbPassword}@google-translate-web-project-kcgi.mongocluster.cosmos.azure.com/?tls=true&authMe

// Ensure that credentials are defined in environment variables
if (!dbUsername || !dbPassword) {
    throw new Error("Please define the MONGO_DB_USERNAME and MONGO_DB_PASSWORD environment variables inside .env.local");
}

// Function to connect to MongoDB
const connectDB = async () => {
    // Check if there is already a connection to MongoDB
    if (mongoose.connection?.readyState >= 1) {
        console.log("-----Already Connected to MongoDB-----");
        return;
    }

    try {
        // Attempt to connect to MongoDB using the provided connection string
        await mongoose.connect(connectionString);
        console.log("-----Connected to MongoDB-----");
    } catch (err) {
        // Log an error message if connection fails
        console.log("Could not connect to MongoDB:", err);
    }
};

```



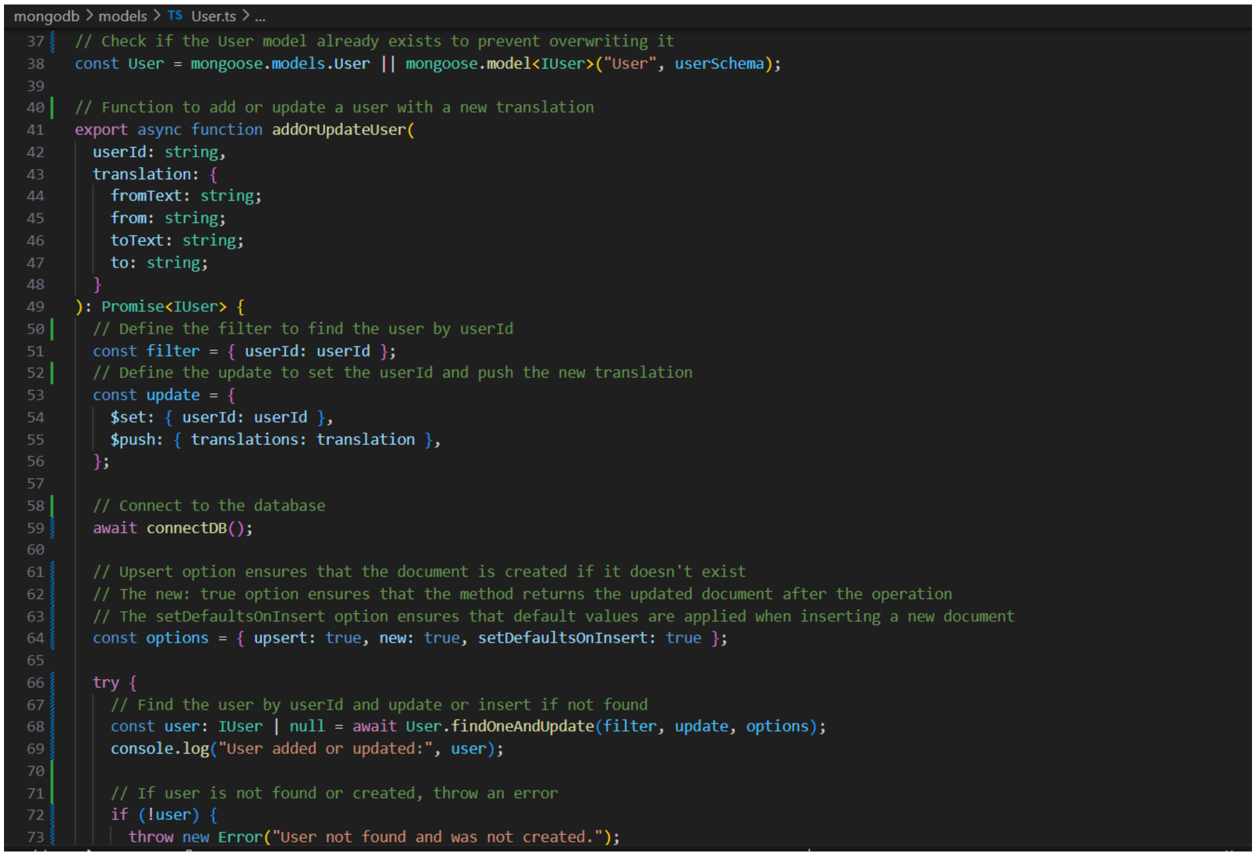
```
    try {
        // Attempt to connect to MongoDB using the provided connection string
        await mongoose.connect(connectionString);
        console.log("-----Connected to MongoDB-----");
    } catch (err) {
        // Log an error message if connection fails
        console.log("Could not connect to MongoDB:", err);
    }
};

export default connectDB; // Export the connectDB function for use in other parts of the application
/* Environment Variables: The script retrieves MongoDB admin username (MONGO_DB_USERNAME) and password (MONGO_DB_PASSWORD) from environment variable
Connection String: It constructs a MongoDB connection string (connectionString) using MongoDB Atlas URI format.
Error Handling: If the required environment variables are not defined, an error is thrown.
ConnectDB Function: This asynchronous function attempts to connect to MongoDB using mongoose.connect. It checks if a connection is already establish
Logging: Messages are logged to the console indicating whether the connection to MongoDB was successful or if there was an error.
 */
```

GAIRE ANANTA PRASAD M24W0272



mongodb > models > TS User.ts > ...

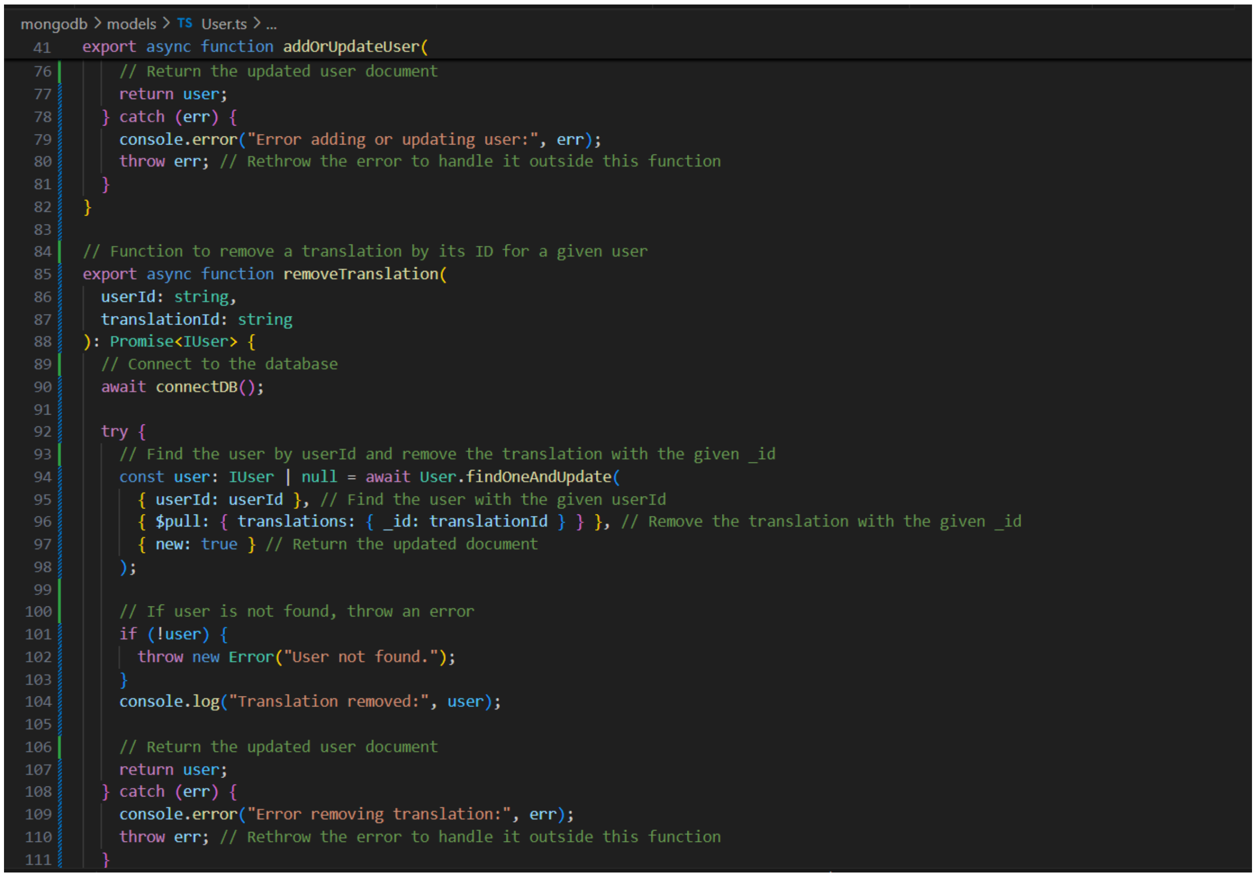
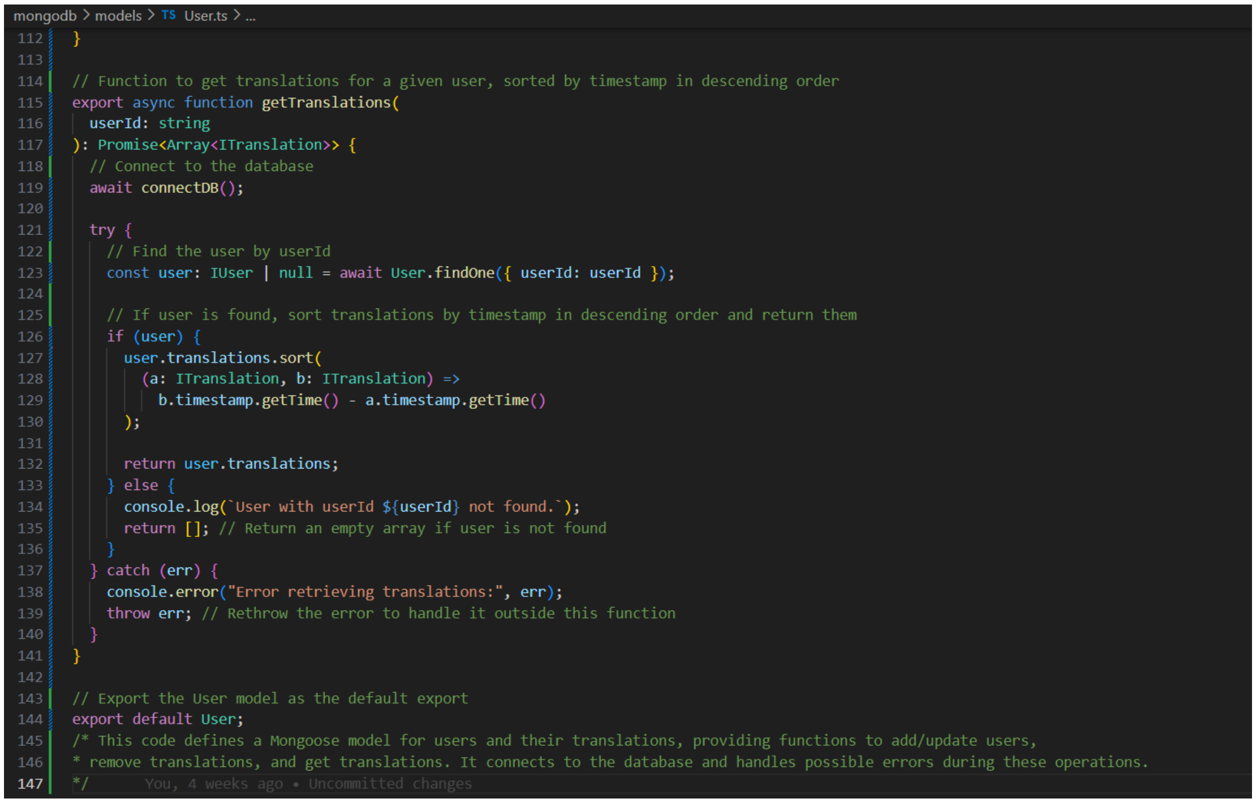
```typescript
You, 4 weeks ago | 1 author (You)
//Gaire Ananta Prasad M24W0272

// Import necessary modules and types from mongoose and the local db connection utility
import mongoose, { Document, Schema } from "mongoose";
import connectDB from "../db";

// Define an interface for the Translation document that extends the Mongoose Document
You, 3 weeks ago | 1 author (You)
export interface ITranslation extends Document {
  timestamp: Date;
  fromText: string;
  from: string;
  toText: string;
  to: string;
}

// Define an interface for the User document that extends the Mongoose Document
You, 3 weeks ago | 1 author (You)
interface IUser extends Document {
  userId: string;
  translations: Array<ITranslation>;
}

// Define the schema for a Translation document
const translationSchema = new Schema({
  timestamp: { type: Date, default: Date.now }, // Set default value to current date
  fromText: String, // Source text
  from: String,     // Source language
  toText: String,   // Translated text
  to: String,       // Target language
});

// Define the schema for a User document, which includes an array of Translation documents
const userSchema = new Schema<IUser>({
  userId: String,             // Unique user identifier
  translations: [translationSchema], // Array of translation documents
```



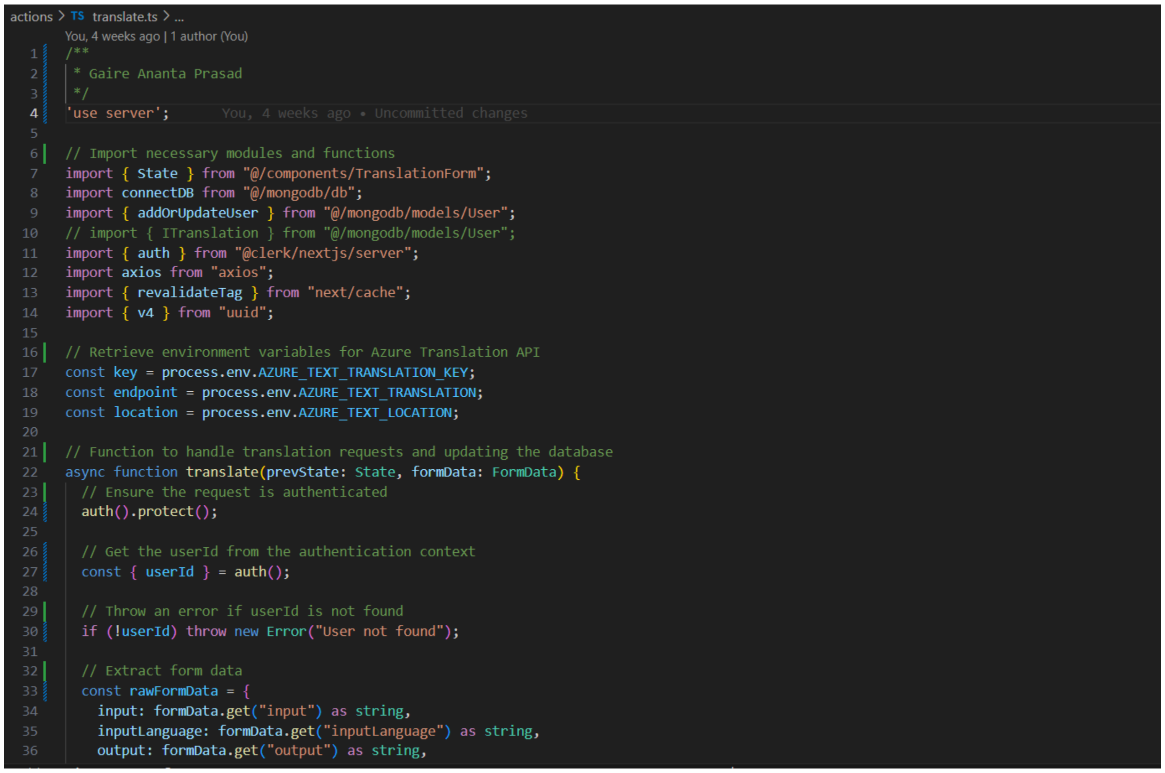
mongodb > models > TS User.ts > ...

```typescript
// Check if the User model already exists to prevent overwriting it
const User = mongoose.models.User || mongoose.model<IUser>("User", userSchema);

// Function to add or update a user with a new translation
export async function addOrUpdateUser(
  userId: string,
  translation: {
    fromText: string;
    from: string;
    toText: string;
    to: string;
  }
): Promise<IUser> {
  // Define the filter to find the user by userId
  const filter = { userId: userId };
  // Define the update to set the userId and push the new translation
  const update = {
    $set: { userId: userId },
    $push: { translations: translation },
  };

  // Connect to the database
  await connectDB();

  // Upsert option ensures that the document is created if it doesn't exist
  // The new: true option ensures that the method returns the updated document after the operation
  // The setDefaultsOnInsert option ensures that default values are applied when inserting a new document
  const options = { upsert: true, new: true, setDefaultsOnInsert: true };

  try {
    // Find the user by userId and update or insert if not found
    const user: IUser | null = await User.findOneAndUpdate(filter, update, options);
    console.log("User added or updated:", user);

    // If user is not found or created, throw an error
    if (!user) {
      throw new Error("User not found and was not created.");
```

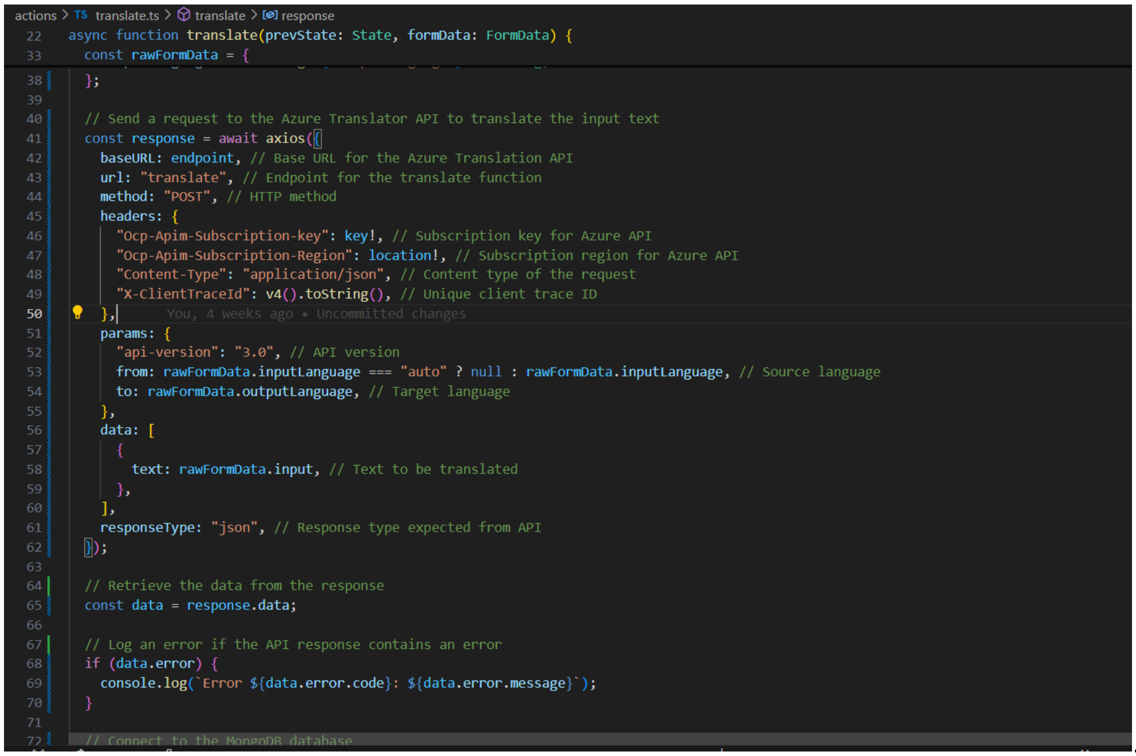
GAIRE ANANTA PRASAD M24W0272


mongodb > models > TS User.ts > ...
```
export async function addOrUpdateUser(
      // Return the updated user document
      return user;
    } catch (err) {
      console.error("Error adding or updating user:", err);
      throw err; // Rethrow the error to handle it outside this function
    }
  }

  // Function to remove a translation by its ID for a given user
  export async function removeTranslation(
    userId: string,
    translationId: string
  ): Promise<IUser> {
    // Connect to the database
    await connectDB();

    try {
      // Find the user by userId and remove the translation with the given _id
      const user: IUser | null = await User.findOneAndUpdate(
        { userId: userId }, // Find the user with the given userId
        { $pull: { translations: { _id: translationId } } }, // Remove the translation with the given _id
        { new: true } // Return the updated document
      );

      // If user is not found, throw an error
      if (!user) {
        throw new Error("User not found.");
      }
      console.log("Translation removed:", user);

      // Return the updated user document
      return user;
    } catch (err) {
      console.error("Error removing translation:", err);
      throw err; // Rethrow the error to handle it outside this function
    }
```

mongodb > models > TS User.ts > ...
```
  }

  // Function to get translations for a given user, sorted by timestamp in descending order
  export async function getTranslations(
    userId: string
  ): Promise<Array<ITranslation>> {
    // Connect to the database
    await connectDB();

    try {
      // Find the user by userId
      const user: IUser | null = await User.findOne({ userId: userId });

      // If user is found, sort translations by timestamp in descending order and return them
      if (user) {
        user.translations.sort(
          (a: ITranslation, b: ITranslation) =>
            b.timestamp.getTime() - a.timestamp.getTime()
        );

        return user.translations;
      } else {
        console.log(`User with userId ${userId} not found.`);
        return []; // Return an empty array if user is not found
      }
    } catch (err) {
      console.error("Error retrieving translations:", err);
      throw err; // Rethrow the error to handle it outside this function
    }
  }

  // Export the User model as the default export
  export default User;
  /* This code defines a Mongoose model for users and their translations, providing functions to add/update users,
  * remove translations, and get translations. It connects to the database and handles possible errors during these operations.
  */      You, 4 weeks ago • Uncommitted changes
```

GAIRE ANANTA PRASAD M24W0272


actions > TS translate.ts > ...
You, 4 weeks ago | 1 author (You)
You, 4 weeks ago • Uncommitted changes

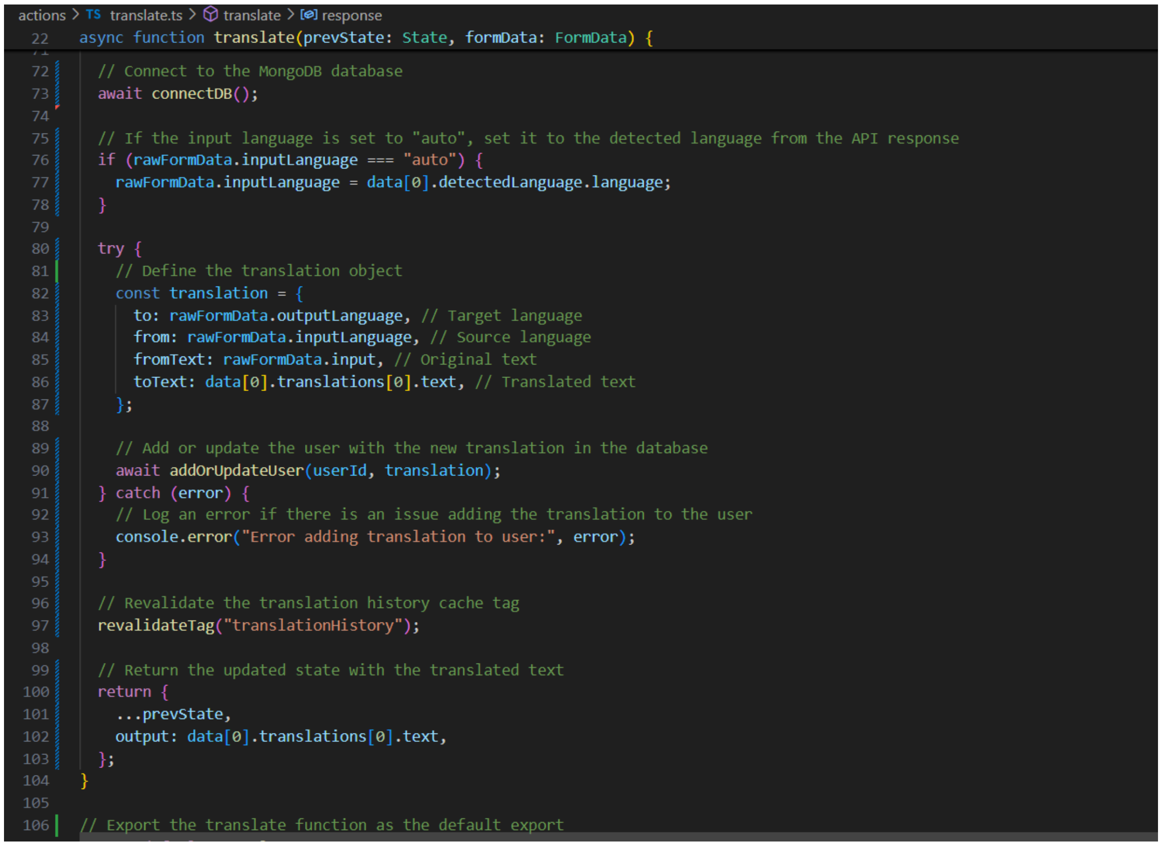
```typescript
/**
 * Gaire Ananta Prasad
 */
'use server';

// Import necessary modules and functions
import { State } from "@/components/TranslationForm";
import connectDB from "@/mongodb/db";
import { addOrUpdateUser } from "@/mongodb/models/User";
// import { ITranslation } from "@/mongodb/models/User";
import { auth } from "@clerk/nextjs/server";
import axios from "axios";
import { revalidateTag } from "next/cache";
import { v4 } from "uuid";

// Retrieve environment variables for Azure Translation API
const key = process.env.AZURE_TEXT_TRANSLATION_KEY;
const endpoint = process.env.AZURE_TEXT_TRANSLATION;
const location = process.env.AZURE_TEXT_LOCATION;

// Function to handle translation requests and updating the database
async function translate(prevState: State, formData: FormData) {
  // Ensure the request is authenticated
  auth().protect();

  // Get the userId from the authentication context
  const { userId } = auth();

  // Throw an error if userId is not found
  if (!userId) throw new Error("User not found");

  // Extract form data
  const rawFormData = {
    input: formData.get("input") as string,
    inputLanguage: formData.get("inputLanguage") as string,
    output: formData.get("output") as string,
```


actions > TS translate.ts > translate > response
You, 4 weeks ago • Uncommitted changes

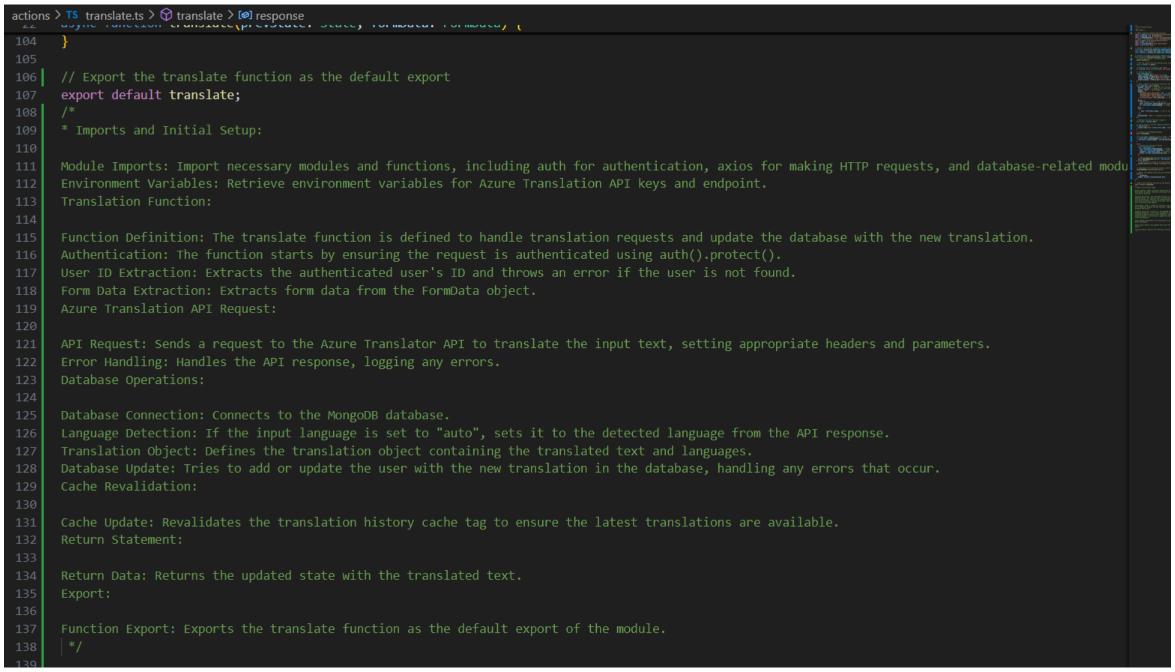
```typescript
async function translate(prevState: State, formData: FormData) {
  const rawFormData = {
```

```typescript
  };

  // Send a request to the Azure Translator API to translate the input text
  const response = await axios({
    baseURL: endpoint, // Base URL for the Azure Translation API
    url: "translate", // Endpoint for the translate function
    method: "POST", // HTTP method
    headers: {
      "Ocp-Apim-Subscription-Key": key!, // Subscription key for Azure API
      "Ocp-Apim-Subscription-Region": location!, // Subscription region for Azure API
      "Content-Type": "application/json", // Content type of the request
      "X-ClientTraceId": v4().toString(), // Unique client trace ID
    },
    params: {
      "api-version": "3.0", // API version
      from: rawFormData.inputLanguage === "auto" ? null : rawFormData.inputLanguage, // Source language
      to: rawFormData.outputLanguage, // Target language
    },
    data: [
      {
        text: rawFormData.input, // Text to be translated
      },
    ],
    responseType: "json", // Response type expected from API
  });

  // Retrieve the data from the response
  const data = response.data;

  // Log an error if the API response contains an error
  if (data.error) {
    console.log(`Error ${data.error.code}: ${data.error.message}`);
  }

  // Connect to the MongoDB database
```

GAIRE ANANTA PRASAD M24W0272

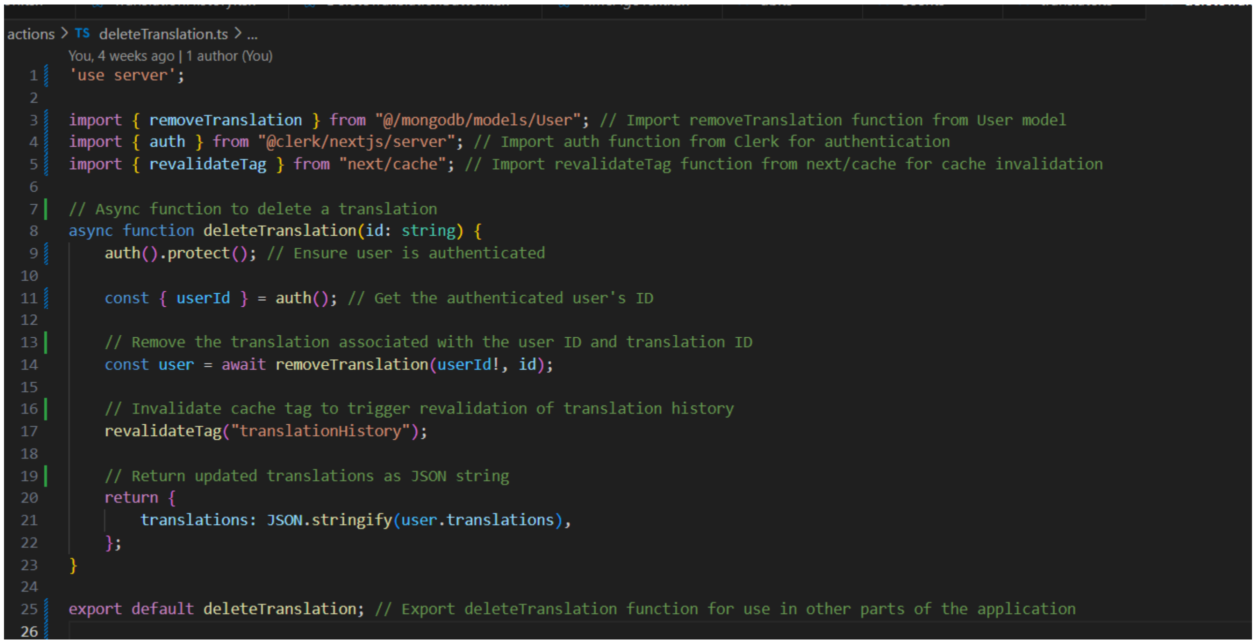


```
async function translate(prevState: State, formData: FormData) {
  // Connect to the MongoDB database
  await connectDB();

  // If the input language is set to "auto", set it to the detected language from the API response
  if (rawFormData.inputLanguage === "auto") {
    rawFormData.inputLanguage = data[0].detectedLanguage.language;
  }

  try {
    // Define the translation object
    const translation = {
      to: rawFormData.outputLanguage, // Target language
      from: rawFormData.inputLanguage, // Source language
      fromText: rawFormData.input, // Original text
      toText: data[0].translations[0].text, // Translated text
    };

    // Add or update the user with the new translation in the database
    await addOrUpdateUser(userId, translation);
  } catch (error) {
    // Log an error if there is an issue adding the translation to the user
    console.error("Error adding translation to user:", error);
  }

  // Revalidate the translation history cache tag
  revalidateTag("translationHistory");

  // Return the updated state with the translated text
  return {
    ...prevState,
    output: data[0].translations[0].text,
  };
}

// Export the translate function as the default export
```



```
}

// Export the translate function as the default export
export default translate;
/*
* Imports and Initial Setup:

Module Imports: Import necessary modules and functions, including auth for authentication, axios for making HTTP requests, and database-related modu
Environment Variables: Retrieve environment variables for Azure Translation API keys and endpoint.
Translation Function:

Function Definition: The translate function is defined to handle translation requests and update the database with the new translation.
Authentication: The function starts by ensuring the request is authenticated using auth().protect().
User ID Extraction: Extracts the authenticated user's ID and throws an error if the user is not found.
Form Data Extraction: Extracts form data from the FormData object.
Azure Translation API Request:

API Request: Sends a request to the Azure Translator API to translate the input text, setting appropriate headers and parameters.
Error Handling: Handles the API response, logging any errors.
Database Operations:

Database Connection: Connects to the MongoDB database.
Language Detection: If the input language is set to "auto", sets it to the detected language from the API response.
Translation Object: Defines the translation object containing the translated text and languages.
Database Update: Tries to add or update the user with the new translation in the database, handling any errors that occur.
Cache Revalidation:

Cache Update: Revalidates the translation history cache tag to ensure the latest translations are available.
Return Statement:

Return Data: Returns the updated state with the translated text.
Export:

Function Export: Exports the translate function as the default export of the module.
 */
```

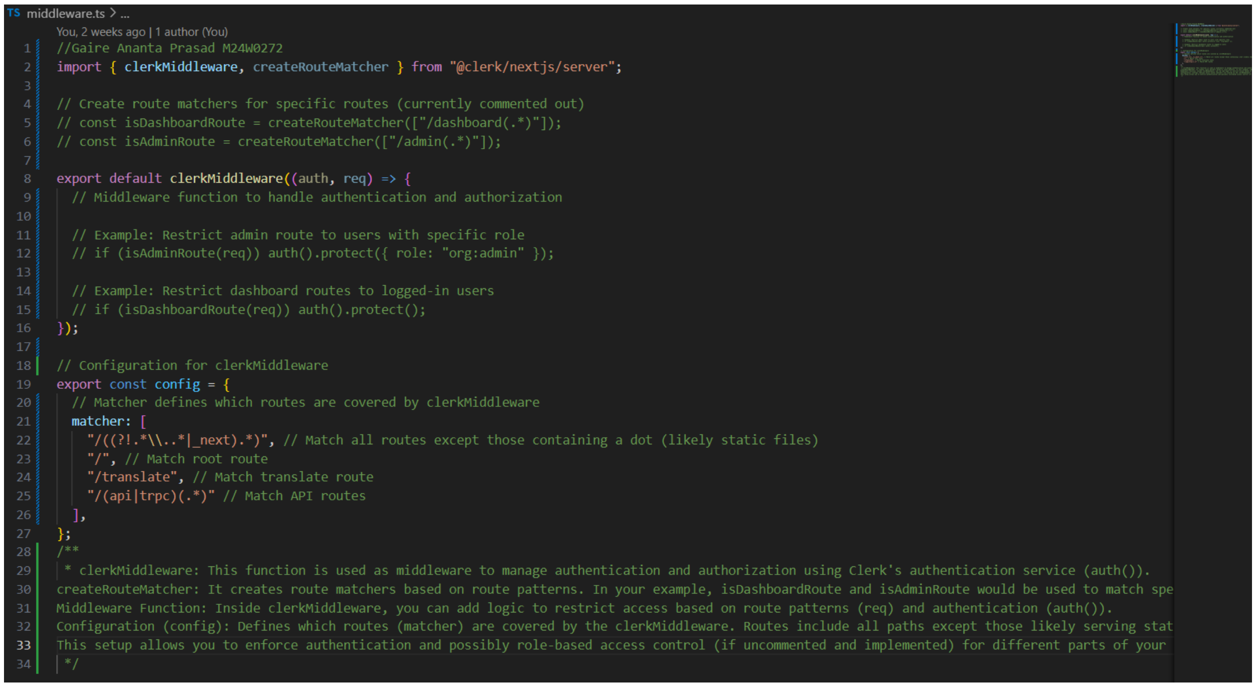
GAIRE ANANTA PRASAD M24W0272



actions > TS deleteTranslation.ts > ...

You, 4 weeks ago | 1 author (You)

```ts
'use server';

import { removeTranslation } from "@/mongodb/models/User"; // Import removeTranslation function from User model
import { auth } from "@clerk/nextjs/server"; // Import auth function from Clerk for authentication
import { revalidateTag } from "next/cache"; // Import revalidateTag function from next/cache for cache invalidation

// Async function to delete a translation
async function deleteTranslation(id: string) {
    auth().protect(); // Ensure user is authenticated

    const { userId } = auth(); // Get the authenticated user's ID

    // Remove the translation associated with the user ID and translation ID
    const user = await removeTranslation(userId!, id);

    // Invalidate cache tag to trigger revalidation of translation history
    revalidateTag("translationHistory");

    // Return updated translations as JSON string
    return {
        translations: JSON.stringify(user.translations),
    };
}

export default deleteTranslation; // Export deleteTranslation function for use in other parts of the application

```



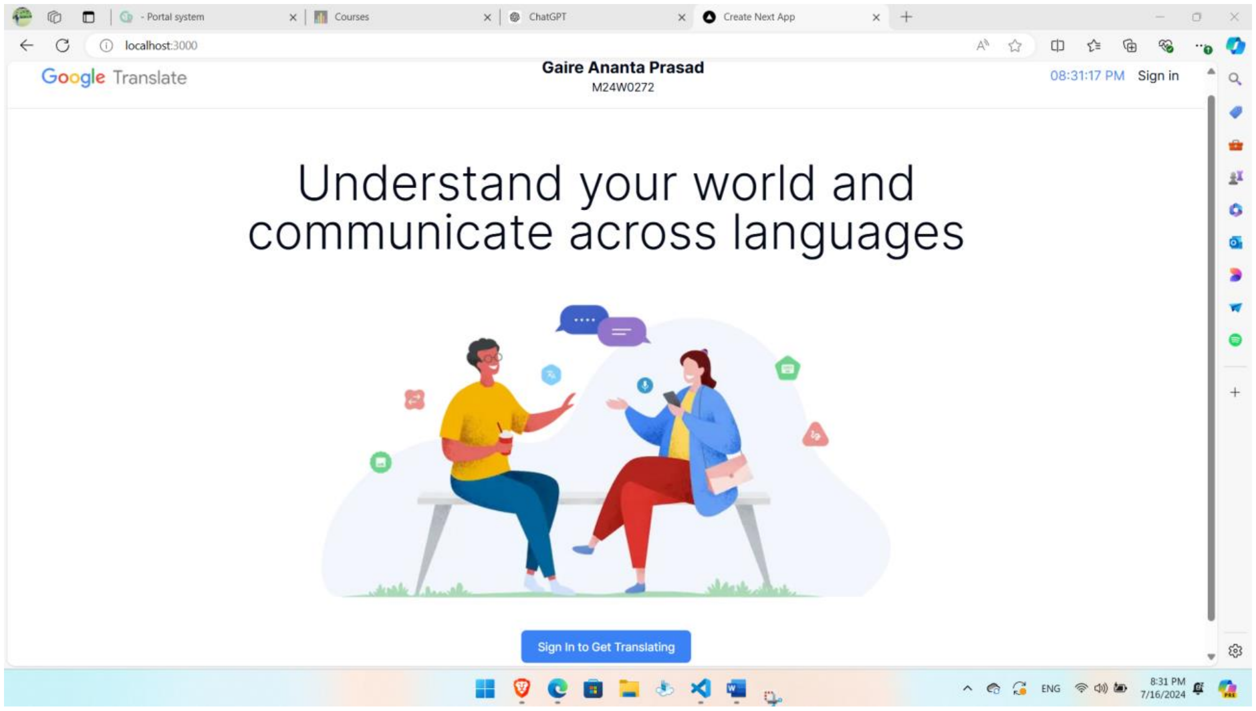
TS middleware.ts > ...

You, 2 weeks ago | 1 author (You)

```ts
//Gaire Ananta Prasad M24W0272
import { clerkMiddleware, createRouteMatcher } from "@clerk/nextjs/server";

// Create route matchers for specific routes (currently commented out)
// const isDashboardRoute = createRouteMatcher(["/dashboard(.*)"]);
// const isAdminRoute = createRouteMatcher(["/admin(.*)"]);

export default clerkMiddleware((auth, req) => {
  // Middleware function to handle authentication and authorization

  // Example: Restrict admin route to users with specific role
  // if (isAdminRoute(req)) auth().protect({ role: "org:admin" });

  // Example: Restrict dashboard routes to logged-in users
  // if (isDashboardRoute(req)) auth().protect();
});

// Configuration for clerkMiddleware
export const config = {
  // Matcher defines which routes are covered by clerkMiddleware
  matcher: [
    "/((?!.*\\..*|_next).*)", // Match all routes except those containing a dot (likely static files)
    "/", // Match root route
    "/translate", // Match translate route
    "/(api|trpc)(.*)" // Match API routes
  ],
};
/**
 * clerkMiddleware: This function is used as middleware to manage authentication and authorization using Clerk's authentication service (auth()).
createRouteMatcher: It creates route matchers based on route patterns. In your example, isDashboardRoute and isAdminRoute would be used to match spe
Middleware Function: Inside clerkMiddleware, you can add logic to restrict access based on route patterns (req) and authentication (auth()).
Configuration (config): Defines which routes (matcher) are covered by the clerkMiddleware. Routes include all paths except those likely serving stat
This setup allows you to enforce authentication and possibly role-based access control (if uncommented and implemented) for different parts of your
 */
```

GAIRE ANANTA PRASAD M24W0272

Output

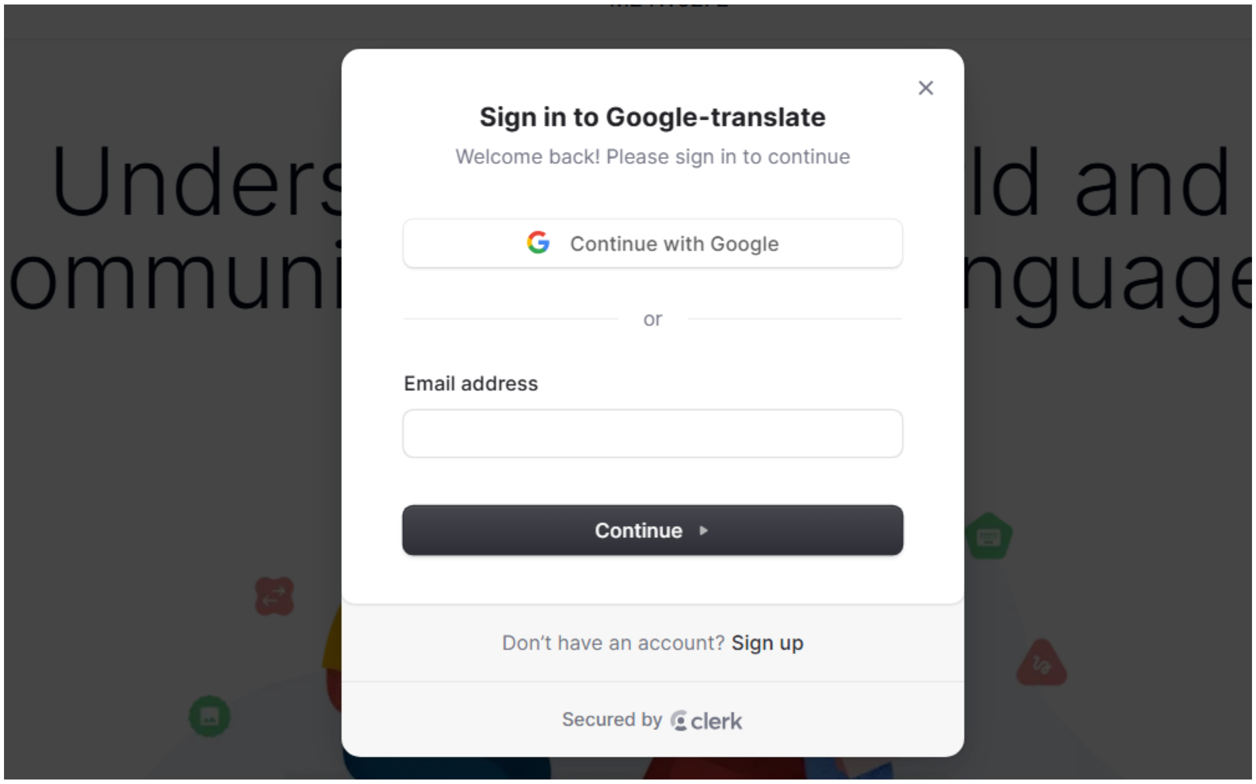


Gaire Ananta Prasad
M24W0272

Understand your world and communicate across languages

Sign In to Get Translating

Cleark Authentication



Sign in to Google-translate

Welcome back! Please sign in to continue

Continue with Google

or

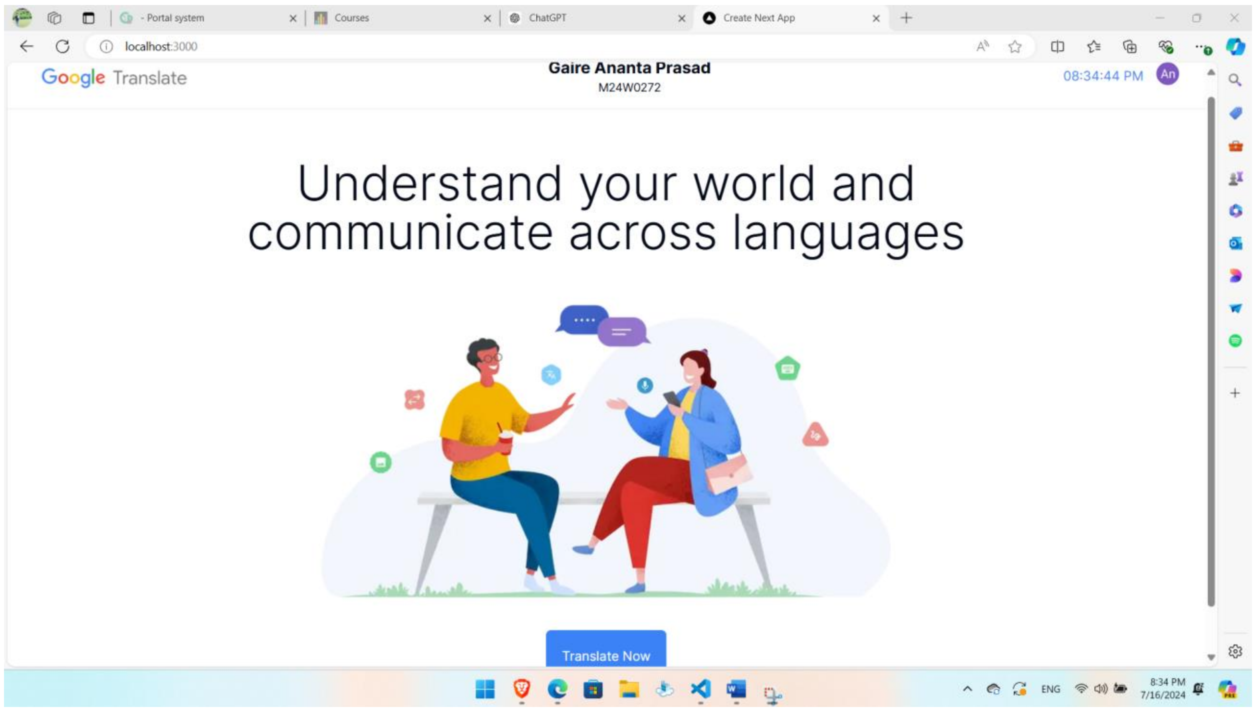
Email address

Continue

Don't have an account? Sign up

Secured by clerk

GAIRE ANANTA PRASAD M24W0272


localhost:3000

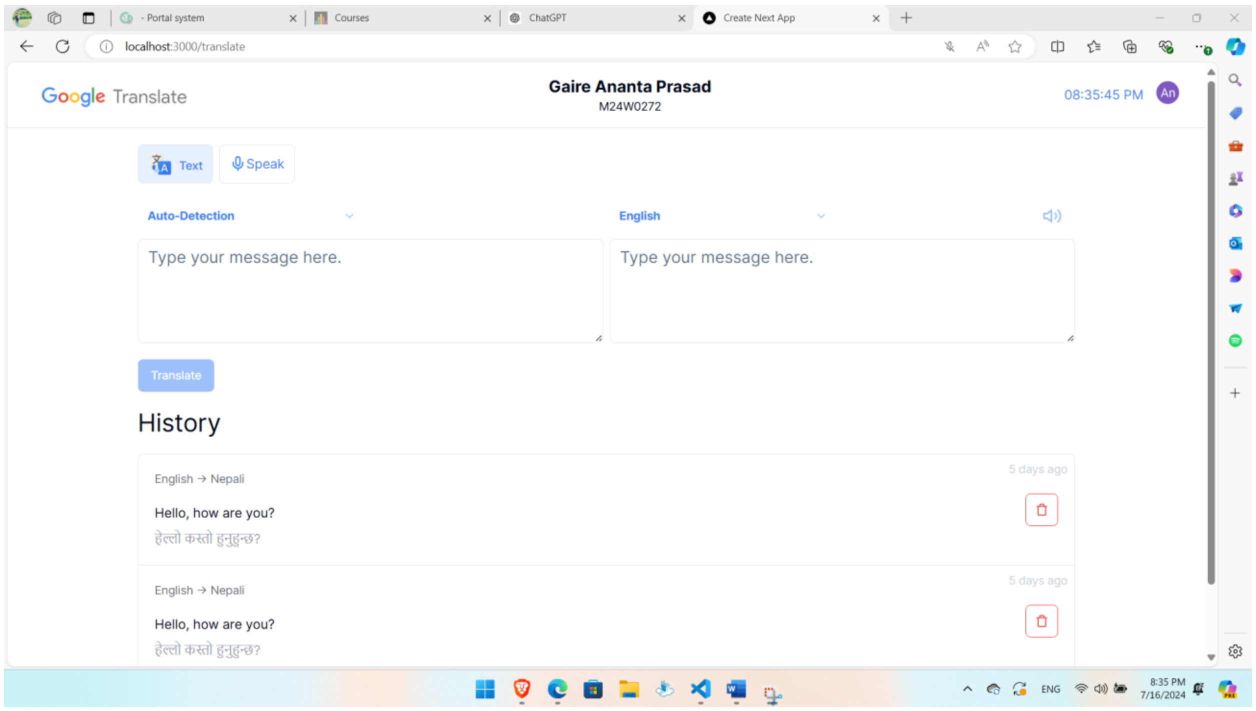
Google Translate

Gaire Ananta Prasad
M24W0272

08:34:44 PM

Understand your world and communicate across languages

Translate Now


localhost:3000/translate

Google Translate

Gaire Ananta Prasad
M24W0272

08:35:45 PM

Text | Speak

Auto-Detection | English

Type your message here. | Type your message here.

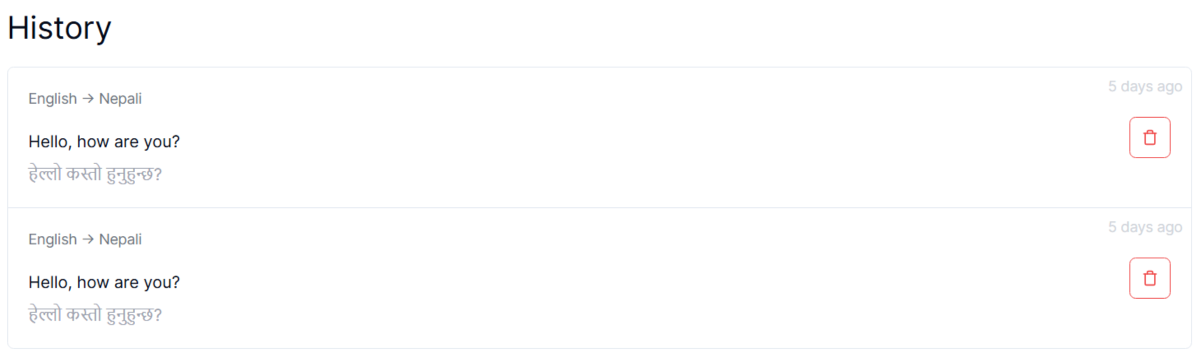
Translate

History

English → Nepali
5 days ago
Hello, how are you?
हेल्लो कस्तो हुनुहुन्छ?

English → Nepali
5 days ago
Hello, how are you?
हेल्लो कस्तो हुनुहुन्छ?

GAIRE ANANTA PRASAD M24W0272



History

English → Nepali

5 days ago

Hello, how are you?

हेल्लो कस्तो हुनुहुन्छ?

English → Nepali

5 days ago

Hello, how are you?

हेल्लो कस्तो हुनुहुन्छ?

GAIRE ANANTA PRASAD M24W0272