```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1" />
    <title>Juan-a Listen - Fun Audio Learning for Students</title>
    <meta name="description" content="Juan-a Listen is a fun, interactive audio-based learning
platform designed for Filipino Grade 7-8 students featuring listening exercises, comprehension
assessments, and sequencing activities aligned with the K-12 curriculum." />
    <meta property="og:title" content="Juan-a Listen - Fun Audio Learning for Students" />
    <meta property="og:description" content="Enhance your listening skills with fun, engaging
audio exercises and interactive activities aligned with the Philippine K-12 curriculum. Perfect for
Grade 7-8 students!" />
    <meta property="og:type" content="website" />
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>


import { RouterProvider, Route, Switch } from "wouter";
import { QueryClientProvider } from "@tanstack/react-query";
import { TooltipProvider } from "@/components/ui/tooltip";
import { Toaster } from "@/components/ui/toaster";
import { queryClient } from "@/lib/queryClient";

// Components
import Header from "@/components/Header";
import Footer from "@/components/Footer";

// Pages
import Dashboard from "@/pages/Dashboard";
import Lesson from "@/pages/Lesson";
import Exercise from "@/pages/Exercise";
import NotFound from "@/pages/not-found";

// Exercise-specific pages
import SequencingExercise from "@/pages/sequencing/SequencingExercise";
import SpotErrorExercise from "@/pages/spot-error/SpotErrorExercise";
import ListeningGame from "@/pages/listening-game/ListeningGame";
import RepeatAfterExercise from "@/pages/repeat-after/RepeatAfterExercise";
import ComprehensionExercise from "@/pages/comprehension/ComprehensionExercise";

function Router() {
  return (
    <div className="min-h-screen flex flex-col bg-gray-50">
      <Header />
      <div className="flex-grow">
        <Switch>
          <Route path="/" component={Dashboard} />
          <Route path="/lessons/:id" component={Lesson} />
          <Route path="/exercise/:id" component={Exercise} />
          <Route path="/sequencing/:id" component={SequencingExercise} />
          <Route path="/spot-error/:id" component={SpotErrorExercise} />
          <Route path="/listening-game/:id" component={ListeningGame} />
          <Route path="/repeat-after/:id" component={RepeatAfterExercise} />
          <Route path="/comprehension/:id" component={ComprehensionExercise} />
          <Route component={NotFound} />
        </Switch>
      </div>
      <Footer />
      <Toaster />
    </div>
  );
}

function App() {
  return (
    <QueryClientProvider client={queryClient}>
      <TooltipProvider>
        <RouterProvider>
          <Router />
        </RouterProvider>
      </TooltipProvider>
    </QueryClientProvider>
  );
}

export default App;


import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
  :root {
    --background: 0 0% 100%;
    --foreground: 222.2 84% 4.9%;

    --card: 0 0% 100%;
    --card-foreground: 222.2 84% 4.9%;

    --popover: 0 0% 100%;
    --popover-foreground: 222.2 84% 4.9%;

    --primary: 265 100% 50%;
    --primary-foreground: 210 40% 98%;

    --secondary: 210 84% 53%;
    --secondary-foreground: 222.2 47.4% 11.2%;

    --muted: 210 40% 96.1%;
    --muted-foreground: 215.4 16.3% 46.9%;

    --accent: 210 40% 96.1%;
    --accent-foreground: 222.2 47.4% 11.2%;

    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 210 40% 98%;

    --border: 214.3 31.8% 91.4%;
    --input: 214.3 31.8% 91.4%;
    --ring: 222.2 84% 4.9%;
```

```css
    --radius: 0.5rem;
  }

  .dark {
    --background: 222.2 84% 4.9%;
    --foreground: 210 40% 98%;

    --card: 222.2 84% 4.9%;
    --card-foreground: 210 40% 98%;

    --popover: 222.2 84% 4.9%;
    --popover-foreground: 210 40% 98%;

    --primary: 265 100% 50%;
    --primary-foreground: 222.2 47.4% 11.2%;

    --secondary: 210 84% 53%;
    --secondary-foreground: 210 40% 98%;

    --muted: 217.2 32.6% 17.5%;
    --muted-foreground: 215 20.2% 65.1%;

    --accent: 217.2 32.6% 17.5%;
    --accent-foreground: 210 40% 98%;

    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 210 40% 98%;

    --border: 217.2 32.6% 17.5%;
    --input: 217.2 32.6% 17.5%;
    --ring: 212.7 26.8% 83.9%;
  }
}

@layer base {
  * {
    @apply border-border;
  }
  body {
    @apply bg-background text-foreground;
    font-feature-settings: "rlig" 1, "calt" 1;
  }
}
```

```
.material-icons {
  font-family: 'Material Icons';
  font-weight: normal;
  font-style: normal;
  font-size: 24px;
  line-height: 1;
  letter-spacing: normal;
  text-transform: none;
  display: inline-block;
  white-space: nowrap;
  word-wrap: normal;
  direction: ltr;
  -webkit-font-feature-settings: 'liga';
  -webkit-font-smoothing: antialiased;
}
```

```
import { Link, useLocation } from "wouter";

const Header = () => {
  const [location] = useLocation();

  return (
    <header className="bg-white shadow-sm">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex justify-between items-center py-4">
          <div className="flex items-center space-x-2">
            <span className="material-icons text-primary text-3xl">headphones</span>
            <Link href="/">
              <a className="text-xl font-bold text-gray-900">Juan-a Listen</a>
            </Link>
          </div>

          <div className="flex items-center space-x-4">
            <div className="hidden md:flex items-center space-x-4">
              <Link href="/">
                <a className={`font-medium ${location === "/" ? "text-primary" : "text-gray-600
hover:text-primary"}`}>
                  Home
                </a>
              </Link>
              <Link href="/lessons/1">
                <a className={`font-medium ${location.startsWith("/lessons") ? "text-primary"
: "text-gray-600 hover:text-primary"}`}>
```

```
                My Lessons
              </a>
            </Link>
            <a href="#" className="text-gray-600 hover:text-primary font-medium">K-12
Activities</a>
            <a href="#" className="text-gray-600 hover:text-primary font-medium">Student
Help</a>
          </div>
          <button className="p-2 rounded-full hover:bg-gray-100">
            <span className="material-icons">notifications</span>
          </button>
          <div className="relative">
            <button className="w-8 h-8 rounded-full bg-gray-300 flex items-center
justify-center text-sm font-medium">
              JS
            </button>
          </div>
        </div>
      </div>
    </div>
  </header>
 );
};

export default Header;

client/src/components/HeroSection.tsx
import { useState } from "react";
import { Link } from "wouter";
import { Button } from "@/components/ui/button";
const HeroSection = () => {
 return (
  <div className="bg-gradient-to-r from-primary to-secondary rounded-xl
overflow-hidden shadow-lg mb-8">
   <div className="md:flex">
    <div className="p-8 md:w-1/2">
     <h1 className="text-3xl font-bold text-white mb-4">Juan-a Listen: Learn with Fun
Audio</h1>
     <p className="text-white/90 mb-6">Enhance your English and Filipino listening
skills with interactive activities aligned to K-12 curriculum for Grades 7-8.</p>
     <div className="flex space-x-4">
      <Link href="/lessons/1">
       <Button className="bg-white text-primary px-4 py-2 rounded-lg font-medium
flex items-center">
```

```
            <span className="material-icons mr-2">play_circle</span>
            Start My Journey
          </Button>
        </Link>
        <Link href="/">
          <Button variant="outline" className="bg-white/20 text-white px-4 py-2
rounded-lg font-medium flex items-center border-white/30 hover:bg-white/30">
            <span className="material-icons mr-2">playlist_add</span>
            Explore Activities
          </Button>
        </Link>
      </div>
    </div>
    <div className="md:w-1/2 relative h-64 md:h-auto overflow-hidden">
      <img

src="https://images.unsplash.com/photo-1522202176988-66273c2fd55f?ixlib=rb-4.0.3&ixi
d=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=80
0&h=600"
        alt="Students using audio learning technology"
        className="w-full h-full object-cover"
      />
    </div>
   </div>
  </div>
 );
};
export default HeroSection;
client/src/components/AudioPlayer.tsx
import { useState, useEffect, useRef } from "react";
import { AudioPlayer as AudioPlayerLib, formatTime } from "@/lib/audio";
import { Button } from "@/components/ui/button";
import { Slider } from "@/components/ui/slider";
type AudioPlayerProps = {
 audioTitle: string;
 audioUrl: string;
 onPlaybackChange?: (isPlaying: boolean) => void;
};
const AudioPlayer = ({ audioTitle, audioUrl, onPlaybackChange }: AudioPlayerProps) => {
 const [isPlaying, setIsPlaying] = useState(false);
 const [progress, setProgress] = useState(0);
 const [volume, setVolume] = useState(0.7);
 const [duration, setDuration] = useState(0);
 const [currentTime, setCurrentTime] = useState(0);
const [showVolumeControl, setShowVolumeControl] = useState(false);

const audioPlayerRef = useRef<AudioPlayerLib | null>(null);
useEffect(() => {
  if (!audioUrl) return;

  const player = new AudioPlayerLib({
    src: audioUrl,
    volume: volume,
    onload: () => {
      setDuration(player.duration);
    },
    onplay: () => {
      setIsPlaying(true);
      if (onPlaybackChange) onPlaybackChange(true);
    },
    onpause: () => {
      setIsPlaying(false);
      if (onPlaybackChange) onPlaybackChange(false);
    },
    onstop: () => {
      setIsPlaying(false);
      if (onPlaybackChange) onPlaybackChange(false);
    },
    onend: () => {
      setIsPlaying(false);
      if (onPlaybackChange) onPlaybackChange(false);
    }
  });

  player.onProgress(setProgress);
  player.onTime(setCurrentTime);

  audioPlayerRef.current = player;

  return () => {
    if (audioPlayerRef.current) {
      audioPlayerRef.current.unload();
    }
  };
}, [audioUrl]);
useEffect(() => {
  if (audioPlayerRef.current) {
    audioPlayerRef.current.setVolume(volume);
    }
  }, [volume]);
  const togglePlayPause = () => {
    if (!audioPlayerRef.current) return;

    if (isPlaying) {
      audioPlayerRef.current.pause();
    } else {
      audioPlayerRef.current.play();
    }
  };
  const handleSeek = (value: number[]) => {
    if (!audioPlayerRef.current) return;
    audioPlayerRef.current.seek(value[0]);
  };
  const handleVolumeChange = (value: number[]) => {
    setVolume(value[0]);
  };
  return (
    <div className="bg-white p-4 rounded-lg shadow-sm border border-gray-100">
      <div className="mb-2">
        <h3 className="text-lg font-semibold text-gray-800">{audioTitle}</h3>
      </div>

      <div className="flex items-center space-x-2 mb-2">
        <Button
          type="button"
          size="icon"
          variant="outline"
          className="rounded-full h-10 w-10 flex items-center justify-center text-primary
hover:text-primary-foreground"
          onClick={togglePlayPause}
        >
          <span className="material-icons">
            {isPlaying ? 'pause' : 'play_arrow'}
          </span>
        </Button>

        <div className="w-full flex items-center space-x-2">
          <span className="text-xs text-gray-600 w-10 text-right">
            {formatTime(currentTime)}
          </span>

          <Slider
            value={[progress]}
            max={1}
            step={0.001}
            className="w-full"
            onValueChange={handleSeek}
          />

          <span className="text-xs text-gray-600 w-10">
            {formatTime(duration)}
          </span>
        </div>

        <div className="relative">
          <Button
            type="button"
            size="icon"
            variant="ghost"
            className="h-8 w-8 rounded-full"
            onClick={() => setShowVolumeControl(!showVolumeControl)}
          >
            <span className="material-icons text-gray-600">
              {volume === 0
                ? 'volume_off'
                : volume < 0.3
                ? 'volume_mute'
                : volume < 0.7
                ? 'volume_down'
                : 'volume_up'
              }
            </span>
          </Button>

          {showVolumeControl && (
            <div className="absolute bottom-full mb-2 bg-white shadow-lg rounded-lg p-4
w-12 h-36">
              <Slider
                value={[volume]}
                max={1}
                step={0.01}
                orientation="vertical"
                className="h-full"
                onValueChange={handleVolumeChange}
              />
            </div>
          )}
        </div>
      </div>
    </div>
  );
};
export default AudioPlayer;
```

client/src/pages/comprehension/ComprehensionExercise.tsx (Fixed navigation)

```
import { useState, useEffect } from "react";
import { useParams, useLocation } from "wouter";
import { useQuery } from "@tanstack/react-query";
import { useToast } from "@/hooks/use-toast";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardHeader, CardTitle, CardDescription, CardFooter } from
"@/components/ui/card";
import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group";
import { Label } from "@/components/ui/label";
import AudioPlayer from "@/components/AudioPlayer";
import { apiRequest } from "@/lib/queryClient";
type Question = {
  id: number;
  question: string;
  options: string[];
  correctOption: number;
  userAnswer?: number;
};
const ComprehensionExercise = () => {
  const { id } = useParams();
  const exerciseId = Number(id);
  const userId = 1; // Fixed user ID for demo
  const [, navigate] = useLocation();
  const { toast } = useToast();

  const [audioUrl, setAudioUrl] = useState<string>("");
  const [audioTitle, setAudioTitle] = useState<string>("");
  const [questions, setQuestions] = useState<Question[]>([]);
  const [activeQuestion, setActiveQuestion] = useState<number>(0);
  const [answers, setAnswers] = useState<Record<number, number>>({});
  const [isSubmitted, setIsSubmitted] = useState<boolean>(false);
  const [score, setScore] = useState<number>(0);
  const [isAudioPlaying, setIsAudioPlaying] = useState<boolean>(false);
```

```
// Fetch exercise data
const { data: exercise, isLoading, error } = useQuery({
  queryKey: [`/api/exercises/${exerciseId}`],
  queryFn: () => apiRequest(`/api/exercises/${exerciseId}`),
});

// Fetch audio data if exercise is loaded
const { data: audio } = useQuery({
  queryKey: [`/api/audios/${exercise?.audioId}`],
  queryFn: () => apiRequest(`/api/audios/${exercise?.audioId}`),
  enabled: !!exercise?.audioId,
});

// Initialize questions from exercise data
useEffect(() => {
  if (exercise && exercise.content) {
    if (typeof exercise.content === 'string') {
      try {
        const content = JSON.parse(exercise.content);
        if (content.questions && Array.isArray(content.questions)) {
          setQuestions(content.questions);
        }
      } catch (e) {
        console.error("Failed to parse exercise content", e);
      }
    } else if (exercise.content.questions) {
      setQuestions(exercise.content.questions);
    }
  }
}, [exercise]);

// Set audio URL when audio data is loaded
useEffect(() => {
  if (audio) {
    setAudioUrl(audio.url);
    setAudioTitle(audio.title);
  }
}, [audio]);

const handleAnswerChange = (questionId: number, optionIndex: number) => {
  setAnswers(prev => ({
    ...prev,
    [questionId]: optionIndex
  }));
```

```
};

const handleSubmit = async () => {
  // Calculate score
  let correctAnswers = 0;
  const updatedQuestions = questions.map(q => {
    const userAnswer = answers[q.id];
    const isCorrect = userAnswer === q.correctOption;
    if (isCorrect) correctAnswers++;

    return {
      ...q,
      userAnswer
    };
  });

  setQuestions(updatedQuestions);
  const finalScore = (correctAnswers / questions.length) * 100;
  setScore(finalScore);
  setIsSubmitted(true);

  try {
    // Save progress to server
    await apiRequest('/api/progress', 'POST', {
      userId,
      exerciseId,
      lessonId: exercise?.lessonId,
      score: finalScore,
      completed: true,
      lastAttemptedAt: new Date()
    });

    toast({
      title: "Exercise completed!",
      description: `You scored ${finalScore.toFixed(0)}%`,
      variant: "default",
    });
  } catch (error) {
    console.error("Failed to save progress", error);
    toast({
      title: "Error saving progress",
      description: "There was a problem saving your progress.",
      variant: "destructive",
    });
    }
  };

  const goToNextQuestion = () => {
    if (activeQuestion < questions.length - 1) {
      setActiveQuestion(prev => prev + 1);
    }
  };

  const goToPreviousQuestion = () => {
    if (activeQuestion > 0) {
      setActiveQuestion(prev => prev - 1);
    }
  };

  if (isLoading) {
    return (
      <div className="max-w-4xl mx-auto p-4 sm:p-6">
        <Card>
          <CardContent className="p-6">
            <div className="animate-pulse flex flex-col space-y-4">
              <div className="h-6 bg-gray-200 rounded w-3/4"></div>
              <div className="h-32 bg-gray-200 rounded"></div>
              <div className="space-y-2">
                <div className="h-4 bg-gray-200 rounded w-5/6"></div>
                <div className="h-4 bg-gray-200 rounded"></div>
                <div className="h-4 bg-gray-200 rounded w-5/6"></div>
              </div>
            </div>
          </CardContent>
        </Card>
      </div>
    );
  }

  if (error || !exercise) {
    return (
      <div className="max-w-4xl mx-auto p-4 sm:p-6 text-center">
        <Card className="p-6">
          <CardContent className="pt-6 flex flex-col items-center">
            <span className="material-icons text-destructive text-5xl
mb-4">error_outline</span>
            <h2 className="text-2xl font-bold text-gray-800 mt-4">Exercise not found</h2>
          <p className="text-gray-600 mt-2 mb-6">We couldn't load the exercise data.
Please try again later.</p>
          <Button onClick={() => navigate("/")}>
            Return to Dashboard
          </Button>
        </CardContent>
      </Card>
    </div>
  );
 }

 return (
   <div className="max-w-4xl mx-auto p-4 sm:p-6">
     <div className="flex justify-between items-center mb-6">
       <Button
         variant="ghost"
         className="mr-4"
         onClick={() => navigate(`/lessons/${exercise.lessonId}`)}
       >
         <span className="material-icons mr-1">arrow_back</span>
         Back to Lesson
       </Button>

       <div className="text-right">
         <h1 className="text-xl font-bold text-gray-900">Comprehension Exercise</h1>
         <p className="text-sm text-gray-600">Listen and answer questions</p>
       </div>
     </div>

     <Card className="mb-6">
       <CardHeader>
         <CardTitle>{exercise.title}</CardTitle>
         <CardDescription>{exercise.instructions}</CardDescription>
       </CardHeader>
       <CardContent>
         <div className="mb-6">
           {audioUrl && (
             <AudioPlayer
               audioTitle={audioTitle}
               audioUrl={audioUrl}
               onPlaybackChange={setIsAudioPlaying}
             />
           )}
         </div>
```

```
{questions.length > 0 && (
  <div className="mt-6">
    <div className="flex justify-between mb-4">
      <h3 className="text-lg font-semibold">
        Question {activeQuestion + 1} of {questions.length}
      </h3>

      {isSubmitted && (
        <div className="text-right">
          <span className="text-lg font-bold">
            Score: {score.toFixed(0)}%
          </span>
        </div>
      )}
    </div>

    <div className="mb-6">
      <p className="text-gray-800 mb-4">{questions[activeQuestion].question}</p>

      <RadioGroup
        value={answers[questions[activeQuestion].id]?.toString()}
        onValueChange={(value) =>
          handleAnswerChange(questions[activeQuestion].id, parseInt(value))
        }
        disabled={isSubmitted}
      >
        {questions[activeQuestion].options.map((option, idx) => (
          <div key={idx} className={`flex items-center p-3 rounded-lg border mb-2 ${
            isSubmitted
              ? idx === questions[activeQuestion].correctOption
                ? "bg-green-50 border-green-200"
                : idx === questions[activeQuestion].userAnswer
                  ? "bg-red-50 border-red-200"
                  : "border-gray-200"
              : "border-gray-200 hover:bg-gray-50"
          }`}>
            <RadioGroupItem
              value={idx.toString()}
              id={`option-${idx}`}
              className="mr-2"
            />
            <Label htmlFor={`option-${idx}`} className="w-full cursor-pointer">
              {option}
```

```
                </Label>

                {isSubmitted && idx === questions[activeQuestion].correctOption && (
                  <span className="material-icons text-green-500
ml-2">check_circle</span>
                )}

                {isSubmitted &&
                 idx === questions[activeQuestion].userAnswer &&
                 idx !== questions[activeQuestion].correctOption && (
                  <span className="material-icons text-red-500 ml-2">cancel</span>
                )}
              </div>
            ))}
          </RadioGroup>
        </div>

        <div className="flex justify-between mt-6">
          <Button
            variant="outline"
            disabled={activeQuestion === 0}
            onClick={goToPreviousQuestion}
          >
            <span className="material-icons mr-1">navigate_before</span>
            Previous
          </Button>

          {activeQuestion < questions.length - 1 ? (
            <Button
              disabled={answers[questions[activeQuestion].id] === undefined}
              onClick={goToNextQuestion}
            >
              Next
              <span className="material-icons ml-1">navigate_next</span>
            </Button>
          ) : (
            !isSubmitted && (
              <Button
                disabled={!Object.keys(answers).length ||
                          Object.keys(answers).length < questions.length}
                onClick={handleSubmit}
              >
                Submit Answers
              </Button>
              )
            )}
          </div>
        </div>
      )}
    </CardContent>
    <CardFooter className="flex justify-between border-t pt-6">
      <Button
        variant="outline"
        onClick={() => navigate(`/lessons/${exercise.lessonId}`)}
      >
        Cancel
      </Button>

      {isSubmitted && (
        <Button onClick={() => navigate(`/lessons/${exercise.lessonId}`)}>
          Complete & Return to Lesson
        </Button>
      )}
    </CardFooter>
  </Card>
 </div>
 );
};
export default ComprehensionExercise;
```

server/storage.ts (Demo Data)

```
// Filipino English lesson for Grade 7-8
const englishLesson: InsertLesson = {
  title: "Filipino Traditions and Communication",
  description: "Learn about Filipino cultural expressions and effective communication
skills",
  category: "English Language Arts",
  duration: 40,
  imageUrl: "https://images.unsplash.com/photo-1516321497487-e288fb19713f",
  audioCount: 5,
  rating: 4.9,
  ratingCount: 248
};
const lesson1 = this.createLesson(englishLesson);
// Create audio for this lesson
const audio1: InsertAudio = {
  lessonId: lesson1.id,
  title: "Filipino Greetings and Expressions",
  duration: 240, // 4:00 in seconds
  url: "/audio/filipino-greetings.mp3"
};
const createdAudio1 = this.createAudio(audio1);
server/routes.ts (Main API routes)
export async function registerRoutes(app: Express): Promise<Server> {
  const apiRouter = express.Router();

  // Users endpoints
  apiRouter.get("/users/:id", async (req: Request, res: Response) => {
    const id = Number(req.params.id);
    if (isNaN(id)) {
      return res.status(400).json({ message: "Invalid user ID" });
    }

    const user = await storage.getUser(id);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }

    res.json(user);
  });

  apiRouter.post("/users", async (req: Request, res: Response) => {
    try {
      const userData = insertUserSchema.parse(req.body);
      const newUser = await storage.createUser(userData);
      res.status(201).json(newUser);
    } catch (error) {
      res.status(400).json({ message: "Invalid user data", error });
    }
  });

  // Lessons endpoints
  apiRouter.get("/lessons", async (_req: Request, res: Response) => {
    const lessons = await storage.getLessons();
    res.json(lessons);
  });

  apiRouter.get("/lessons/:id", async (req: Request, res: Response) => {
    const id = Number(req.params.id);
    if (isNaN(id)) {
      return res.status(400).json({ message: "Invalid lesson ID" });
    }

    const lesson = await storage.getLesson(id);
    if (!lesson) {
      return res.status(404).json({ message: "Lesson not found" });
    }

    res.json(lesson);
  });

  apiRouter.get("/lessons/category/:category", async (req: Request, res: Response) => {
    const { category } = req.params;
    const lessons = await storage.getLessonsByCategory(category);
    res.json(lessons);
  });

  apiRouter.post("/lessons", async (req: Request, res: Response) => {
    try {
      const lessonData = insertLessonSchema.parse(req.body);
      const newLesson = await storage.createLesson(lessonData);
      res.status(201).json(newLesson);
    } catch (error) {
      res.status(400).json({ message: "Invalid lesson data", error });
    }
  });

  // Audio endpoints
  apiRouter.get("/lessons/:lessonId/audios", async (req: Request, res: Response) => {
    const lessonId = Number(req.params.lessonId);
    if (isNaN(lessonId)) {
      return res.status(400).json({ message: "Invalid lesson ID" });
    }

    const audios = await storage.getAudios(lessonId);
    res.json(audios);
  });

  // ... More endpoints for audios, exercises, progress, integrations

  app.use("/api", apiRouter);

  // Start the server
  const PORT = process.env.PORT || 5000;
  return app.listen(PORT, () => {
    log(`serving on port ${PORT}`);
  });
}
```

shared/schema.ts

```
// User schema
export const users = pgTable("users", {
  id: serial("id").primaryKey(),
  displayName: text("display_name"),
  username: text("username").notNull().unique(),
  password: text("password").notNull(),
  avatarInitials: text("avatar_initials")
});
export const insertUserSchema = createInsertSchema(users).pick({
  username: true,
  password: true,
  displayName: true,
  avatarInitials: true
});
// Lesson schema
export const lessons = pgTable("lessons", {
  id: serial("id").primaryKey(),
  title: text("title").notNull(),
  duration: integer("duration").notNull(),
  description: text("description"),
  category: text("category").notNull(),
  imageUrl: text("image_url"),
  audioCount: integer("audio_count").notNull(),
  rating: numeric("rating"),
  ratingCount: integer("rating_count")
});
// Audio schema
export const audios = pgTable("audios", {
  id: serial("id").primaryKey(),
  title: text("title").notNull(),
  duration: integer("duration").notNull(), // in seconds
  lessonId: integer("lesson_id").notNull().references(() => lessons.id),
  url: text("url").notNull()
});
// Exercise schema
export const exercises = pgTable("exercises", {
  id: serial("id").primaryKey(),
  title: text("title").notNull(),
  content: jsonb("content").notNull(),
  type: text("type").notNull(), // sequencing, spot-error, listening-game, repeat-after,
comprehension
  audioId: integer("audio_id").notNull().references(() => audios.id),
  instructions: text("instructions")
});
// User Progress schema
export const userProgress = pgTable("user_progress", {
  id: serial("id").primaryKey(),
  score: numeric("score"),
  lessonId: integer("lesson_id").notNull().references(() => lessons.id),
  userId: integer("user_id").notNull().references(() => users.id),
  exerciseId: integer("exercise_id").notNull().references(() => exercises.id),
  completed: boolean("completed").notNull().default(false),
  lastAttemptedAt: timestamp("last_attempted_at")
});
// Integration schema
export const integrations = pgTable("integrations", {
  id: serial("id").primaryKey(),
  type: text("type").notNull(),
  name: text("name").notNull(),
  description: text("description"),
  imageUrl: text("image_url"),
  isConnected: boolean("is_connected")
});
// Exercise content schemas
export const sequencingExerciseContentSchema = z.object({
  steps: z.array(z.object({
    id: z.number(),
    text: z.string(),
    isCorrect: z.boolean().optional()
  })),
  solution: z.array(z.number())
});
export const spotErrorExerciseContentSchema = z.object({
  transcript: z.string(),
  errors: z.array(z.object({
    id: z.number(),
    text: z.string(),
    start: z.number(),
    end: z.number(),
    correction: z.string()
  }))
});
// Type definitions
```

```
export type User = typeof users.$inferSelect;
export type InsertUser = z.infer<typeof insertUserSchema>;
export type Lesson = typeof lessons.$inferSelect;
export type InsertLesson = z.infer<typeof insertLessonSchema>;
export type Audio = typeof audios.$inferSelect;
export type InsertAudio = z.infer<typeof insertAudioSchema>;
export type Exercise = typeof exercises.$inferSelect;
export type InsertExercise = z.infer<typeof insertExerciseSchema>;
export type UserProgress = typeof userProgress.$inferSelect;
export type InsertUserProgress = z.infer<typeof insertUserProgressSchema>;
export type Integration = typeof integrations.$inferSelect;
export type InsertIntegration = z.infer<typeof insertIntegrationSchema>;
```