1. Folder Structure

otakuoasis/ server/ models/H · Anime.ts | Comment.ts | H

-auth.ts

middleware/

src/ | |

index.tsx|

Home.tsx

L - AnimeDetail.tsx | |

app.ts | |

User.ts

routes/ |

anime.ts

auth.ts

client/ |

App.tsx|| H

pages/

Browse.tsx

components/

AuthForm.tsx |

AnimeBrowser.tsx | |

AnimeComments.tsx |

public/ |

index.html

2. Backend Setup (Node.js + Express + MongoDB)

import express from "express"; import

mongoose from "mongoose"; import cors from "cors"; import auth Routes from "./routes/auth"; import animeRoutes from "./routes/anime";

const app = express();

app.use(cors()); app.use(express.json());

app.use("/api/auth", authRoutes); app.use("/api/

anime", animeRoutes);

mongoose.connect(process.env.MONGO_URI!,

{ dbName: "otakuoasis" }).then(() => { console.log("MongoDB connected"); app.listen(4000, () => console.log("Server running on http://localhost:4000")); });

server/models/Anime.ts

```
import { Schema, model } from "mongoose";
const SourceSchema = new Schema({
type: { type: String, enum: ["subbed", "dubbed"],
required: true },
language: { type: String, required: true },
streamUrl: { type: String, required: true }, downloadUrl: { type: String, required: true },
});
const EpisodeSchema = new Schema({
episodeNumber: Number,
title: String,
sources: [SourceSchema],
lastUpdated: Date,
});
const AnimeSchema = new Schema({ title: { type: String, required: true },
description: String,
genres: [String],
releaseYear: Number,
episodes: [EpisodeSchema],
coverImage: String,
status: {type: String, enum: ["ongoing",
"finished"], default: "ongoing" },
lastUpdated: Date,
});
export default model("Anime", AnimeSchema);
```

server/models/User.ts

```
import { Schema, model } from "mongoose";
const UserSchema = new Schema({
username: { type: String, required: true, unique:
true},
email: { type: String, required: true, unique:
true},
passwordHash: { type: String, required: true },
avatar: String,
createdAt: { type: Date, default: Date.now },
favorites: [{ type: Schema.Types.ObjectId, ref:
"Anime" }],
});
```

```
export default model("User", UserSchema);
server/models/Comment.ts
import { Schema, model } from "mongoose";
const CommentSchema = new Schema({
anime: { type: Schema.Types.ObjectId, ref:
"Anime", required: true },
user: { type: Schema.Types.ObjectId, ref: "User",
required: true },
text: {type: String, required: true },
createdAt: { type: Date, default: Date.now },
});
export default model("Comment",
CommentSchema);
server/middleware/auth.ts
import { Request, Response, NextFunction } from
"express";
import jwt from "jsonwebtoken";
export function authMiddleware(req: Request,
res: Response, next: NextFunction) {
const token =
req.headers.authorization?.replace("Bearer ", ""); if (!token) return res.status(401).json({
error: "No
token" });
try {
const payload=jwt.verify(token,
process.env.JWT_SECRET!);
(req as any).user = payload;
next();
} catch {
return res.status(401).json({ error: "Invalid
token" });
}
}
server/routes/auth.ts
import { Router } from "express";
import Anime from "../models/Anime"; import Comment from "../models/Comment";
import { authMiddleware } from "../middleware/
auth";
```

```
const router = Router();
router.get("/", async (req, res) => {
const { genre } = req.query;
let filter: any = {}; if (genre) filter.genres = genre;
const animeList = await Anime.find(filter);
res.json(animeList);
});
router.get("/:id", async (req, res) => {
const anime = await
Anime.findByld(req.params.id);
if (!anime) return res.status(404).json({ error:
"Anime not found" });
res.json(anime);
});
router.post("/:id/comment", authMiddleware,
async (req, res) => {
const { text } = req.body;
const comment = await Comment.create({
anime: req.params.id,
user: req.user.userld,
text,
createdAt: new Date(),
});
res.json(comment);
});
router.get("/:id/comments", async (req, res) => {
const comments = await Comment.find({ anime: req.params.id }).populate("user",
"username
avatar");
res.json(comments);
});
export default router;
```

3. Frontend Setup (React + Tailwind)

client/src/App.tsx

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import Browse from "./pages/Browse";
import AnimeDetail from "./pages/AnimeDetail";
```

```
import AuthForm from "./components/ AuthForm";
function App() {
return (
<BrowserRouter>
<Routes>
<Route path="/" element={<Home />} />
<Route path="/browse" element={<Browse />} /> <Route path="/anime/:id" element={<AnimeDetail />} />
<Route path="/login" element={<AuthForm mode="login" />} />
<Route path="/signup" element={<AuthForm mode="signup" />} />
</Routes>
</BrowserRouter>
);
}
export default App;
client/src/pages/Home.tsx export default function Home() {
return (
<main className="min-h-screen
bg-gradient-to-br from-indigo-900 via-purple-900 to-pink-900 text-white">
<header className="flex items-center justify-between px-8 py-6">
<h1 className="text-4xl font-extrabold tracking-tight">OtakuOasis</h1> <nav
className="space-x-4"> <a href="/browse"
className="hover:underline">Browse</a>
<a href="/login"
className="hover:underline">Login</a>
</nav>
</header>
<section className="mt-12 text-center"> <h2 className="text-3xl font-bold mb-4">Stream and Download All Anime Dubbed and Subbed!</h2> <p
className="mb-8">Your ultimate anime paradise awaits.</p> <a href="/browse"
className="px-6 py-3 bg-pink-600 text-white rounded-lg text-lg font-bold hover:bg-pink-700 transition">Browse
Anime</a> </section>
</main>
);
} client/src/pages/Browse.tsx import { useEffect, useState } from "react";
import AnimeBrowser from "../components/
AnimeBrowser";
```

```
export default function Browse() { const [animeList, setAnimeList] = useState([]);
useEffect(() => {
fetch("/api/anime") .then(res => res.json()) .then(setAnimeList);
}, []);
return (
<div className="max-w-7xl mx-auto p-8">
<h2 className="text-2xl font-bold mb-6">Browse All Anime</h2> <AnimeBrowser animeList={animeList} />
</div> );
} client/src/components/Anime Browser.tsx import { useState } from "react";
const genres = ["Action", "Romance", "Comedy", "Adventure", "Horror", "Slice of Life", "Fantasy"]; const versions = ["subbed", "dubbed"];
export default function AnimeBrowser({ animeList }) { const [selectedGenre, setSelectedGenre] =
useState(""); const [selectedVersion, setSelectedVersion] =
useState("subbed");
const filteredAnime = animeList.filter((anime) => (selectedGenre?
anime.genres.includes(selectedGenre): true)
);
return (
<section> <div className="flex space-x-4 mb-4"> <select onChange={e : => setSelectedGenre(e.target.value)}
value={selectedGenre}>
<option value="">All Genres</option> {genres.map(g => <option key={g} value={g}>{g}
</option>)}
</select>
<div>
{versions.map(v => (
<button
key={v}
onClick={() = => setSelectedVersion(v)} className={px-4 py-2 mx-1 rounded ${
selectedVersion ===v? "bg-indigo-600
text-white": "bg-gray-200 text-black" }}
>
{v.charAt(0).toUpperCase() + v.slice(1)}
</button> ))}
</div>
</div>
```

```
<div className="grid grid-cols-2 md:grid-cols-4 gap-6"> {filteredAnime.map(anime => (
<div key={anime.title}
className="bg-gradient-to-br from-purple-800 to-indigo-700 rounded-xl p-4">
<img src={anime.coverImage} alt={anime.title} className="w-full h-40 object-cover mb-2
rounded" />
<h3 className="font-bold text-lg">{anime.title}
</h3>
<p>{anime.genres.join(", ")}</p>
<a href={/anime/${anime._id}} className="mt-2
px-3 py-1 bg-pink-600 text-white rounded block text-center">
View Details
</a>
</div>
))}
</div>
</section>
);
} client/src/pages/AnimeDetail.tsx
import { useEffect, useState } from "react";
import
{ useParams } from "react-router-dom"; import AnimeComments from "../components/
AnimeComments";
export default function Anime Detail() { const {id} = useParams(); const [anime, setAnime]
= useState(null); const [version, setVersion] = useState("subbed");
useEffect(() => { fetch(/api/anime/${id}) .then(res => res.json()) .then(setAnime);
}, [id]);
if (!anime) return <div>Loading...</div>;
return (
<main className="max-w-3xl mx-auto py-8"> <img src={anime.coverImage}
alt={anime.title}
className="w-full rounded-xl mb-6" />
<h1 className="text-3xl
font-bold">{anime.title}</h1>
<p className="mb-3">{anime.description}</p>
<div className="mb-4"> <span className="mr-2">Genres:</span>
{anime.genres.map(g => (
<span key={g} className="px-2 py-1
bg-indigo-700 text-white rounded mr-2">{g}</
```

```
span> ))}
</div>
<div className="mb-4"> <button
className={px-4 py-2 mr-2 rounded ${version
=== "subbed" ? "bg-pink-600 text-white":
"bg-gray-300 text-black"}}
onClick={() => setVersion("subbed")}
Subbed
</button>
<button
className={px-4 py-2 rounded ${version
"dubbed"? "bg-pink-600 text-white":
"bg-gray-300 text-black"}} onClick={() => setVersion("dubbed")}
>
Dubbed
</button>
</div>
<div>
<h2 className="text-xl font-bold mb-2">Episodes</h2>
<ul>
{anime.episodes.map(ep =>
<li key={ep.episodeNumber} <span className="font-bold">Episode
className="mb-2">
{ep.episodeNumber}:</span> {ep.title}
{ep.sources.filter(src => src.type
version).map(src =>
<div key={src.streamUrl} className="flex gap-2
mt-2">
<a
href={src.streamUrl}
target="_blank"
rel="noopener"
className="px-3 py-1 bg-indigo-500 text-white
>
rounded hover:bg-indigo-600" Watch {version.charAt(0).toUpperCase() + version.slice(1)}
</a>
<a
href={src.downloadUrl}
```

```
target="_blank"
rel="noopener"
className="px-3 py-1 bg-pink-500 text-white
rounded hover:bg-pink-600" download
>
Download {version.charAt(0).toUpperCase() +
version.slice(1)}
</a>
</div>
)}
</li>
}}
</ul>
</div>
<AnimeComments animeId={anime._id} user={null /* pass user object here if logged in
*/} />
</main>
);
} client/src/components/AuthForm.tsx
import { useState } from "react";
export default function AuthForm({ mode :
=
"login", onAuth }) { const [username, setUsername] = useState("""");
const [email, setEmail] = useState(""""); const [password, setPassword] = useState("");
const [error, setError] = useState("""");
async function handleSubmit(e) { e.preventDefault(); const payload = { username,
password, ...(mode
===
"signup" && { email }) };
const res = await fetch(/api/auth/${mode}, { method: "POST",
headers: { "Content-Type": "application/json"},
body: JSON.stringify(payload),
});
const data await res.json();
if (res.ok) onAuth && onAuth(data);
else setError(data.error || "Authentication
failed");
}
```

```
return (
<form className="bg-gradient-to-tr from-indigo-900 to-pink-900 p-8 rounded-xl shadow-neon" onSubmit={handleSubmit}>
<h2 className="text-2xl font-bold
mb-6">{mode
===
= "signup" ? "Sign Up" : "Login"}
to OtakuOasis</h2>
<input
className="block w-full mb-3 px-4 py-2 rounded
border-none"
placeholder="Username"
value={username}
onChange={e => setUsername(e.target.value)}
required
/>
{mode
<input
===
"signup" && (
className="block w-full mb-3 px-4 py-2 rounded border-none"
type="email"
placeholder="Email"
value={email}
onChange={e => setEmail(e.target.value)}
required
/>
})}
<input
className="block w-full mb-3 px-4 py-2 rounded
border-none"
type="password"
placeholder="Password"
value={password}
onChange={e => setPassword(e.target.value)}
required
1>
From my TECNO
```