🚀 Guide to Building "Kosova Travel Guide"

Since you’re using **Next.js, Tailwind CSS, Express.js, MongoDB, Vercel, and Auth0**, here’s a step-by-step roadmap to get started, along with a **draft version of your project structure**.

**🛠 1. Project Setup (Frontend & Backend)**

**1.1 Initialize the Next.js Project**

First, create your Next.js app with Tailwind CSS:

bash

CopyEdit

npx create-next-app@latest kosova-travel-guide

cd kosova-travel-guide

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

* Configure tailwind.config.js:

js

CopyEdit

module.exports = {

content: ["./pages/\*\*/\*.{js,ts,jsx,tsx}", "./components/\*\*/\*.{js,ts,jsx,tsx}"],

theme: { extend: {} },

plugins: [],

};

* Add Tailwind to globals.css:

css

CopyEdit

@tailwind base;

@tailwind components;

@tailwind utilities;

**1.2 Setup Express.js Backend**

In the root folder, create a backend folder and initialize an Express.js app:

bash

CopyEdit

mkdir backend && cd backend

npm init -y

npm install express mongoose cors dotenv

* Create server.js:

js

CopyEdit

require("dotenv").config();

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

const app = express();

app.use(express.json());

app.use(cors());

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log("MongoDB connected"))

.catch(err => console.log(err));

app.get("/", (req, res) => res.send("API is running..."));

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

**🌍 2. Building Core Features**

**2.1 Authentication (Auth0)**

Install Auth0 dependencies in Next.js:

bash

CopyEdit

npm install @auth0/nextjs-auth0

* Create an **Auth0 account**, get the **Client ID, Client Secret, and Domain**, and configure a .env.local file:

ini

CopyEdit

AUTH0\_SECRET=yourSecret

AUTH0\_BASE\_URL=http://localhost:3000

AUTH0\_ISSUER\_BASE\_URL=https://your-tenant.auth0.com

AUTH0\_CLIENT\_ID=yourClientId

AUTH0\_CLIENT\_SECRET=yourClientSecret

* Inside pages/api/auth/[...auth0].js:

js

CopyEdit

import { handleAuth } from '@auth0/nextjs-auth0';

export default handleAuth();

* Add login/logout buttons in the Navbar:

js

CopyEdit

import { useUser } from '@auth0/nextjs-auth0';

export default function Navbar() {

const { user } = useUser();

return (

<nav className="p-4 bg-gray-900 text-white flex justify-between">

<h1>Kosova Travel Guide</h1>

{user ? (

<a href="/api/auth/logout">Logout</a>

) : (

<a href="/api/auth/login">Login</a>

)}

</nav>

);

}

**2.2 Interactive Map with Destinations**

Use **Leaflet.js** for map integration:

bash

CopyEdit

npm install react-leaflet leaflet

* Create components/Map.js:

js

CopyEdit

import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";

export default function Map() {

return (

<MapContainer center={[42.6629, 21.1655]} zoom={8} className="h-96 w-full">

<TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />

<Marker position={[42.6629, 21.1655]}>

<Popup>Welcome to Kosovo!</Popup>

</Marker>

</MapContainer>

);

}

**2.3 AI Chatbot (Kosovo Travel Assistant)**

For a simple AI chatbot, you can use OpenAI’s API:

bash

CopyEdit

npm install openai

* Set up the API in pages/api/chat.js:

js

CopyEdit

import { Configuration, OpenAIApi } from "openai";

export default async function handler(req, res) {

const openai = new OpenAIApi(new Configuration({ apiKey: process.env.OPENAI\_KEY }));

const response = await openai.createChatCompletion({

model: "gpt-3.5-turbo",

messages: [{ role: "system", content: "You are a Kosovo travel assistant." }, ...req.body.messages],

});

res.json({ message: response.data.choices[0].message.content });

}

* Add an input form in components/ChatBot.js:

js

CopyEdit

import { useState } from "react";

export default function ChatBot() {

const [messages, setMessages] = useState([]);

const [input, setInput] = useState("");

const sendMessage = async () => {

const res = await fetch("/api/chat", {

method: "POST",

body: JSON.stringify({ messages: [...messages, { role: "user", content: input }] }),

headers: { "Content-Type": "application/json" },

});

const data = await res.json();

setMessages([...messages, { role: "user", content: input }, { role: "assistant", content: data.message }]);

setInput("");

};

return (

<div>

<div>{messages.map((msg, idx) => <p key={idx}><b>{msg.role}:</b> {msg.content}</p>)}</div>

<input value={input} onChange={(e) => setInput(e.target.value)} />

<button onClick={sendMessage}>Send</button>

</div>

);

}

**2.4 Currency Converter**

Use an API like **exchangeratesapi.io** to fetch currency rates dynamically.

**🎯 3. Deployment**

**3.1 Deploy Backend to Railway/Vercel**

* Install Railway CLI and deploy:

bash

CopyEdit

railway up

* For Vercel:

bash

CopyEdit

vercel

**3.2 Deploy Next.js Frontend**

Push your Next.js app to GitHub and deploy via Vercel:

bash

CopyEdit

vercel --prod

**🎮 4. Extra: Checkpoint Progress System**

Use a **progress bar** (circular) from react-circular-progressbar:

bash

CopyEdit

npm install react-circular-progressbar

* In components/Progress.js:

js

CopyEdit

import { CircularProgressbar } from 'react-circular-progressbar';

import 'react-circular-progressbar/dist/styles.css';

export default function Progress({ progress }) {

return <CircularProgressbar value={progress} text={`${progress}%`} />;

}

**✅ Next Steps**

* Build the **database models** for users, destinations, and plans.
* Implement a **saved itinerary feature**.
* Add **more UI enhancements**.

Top of Form

Bottom of Form