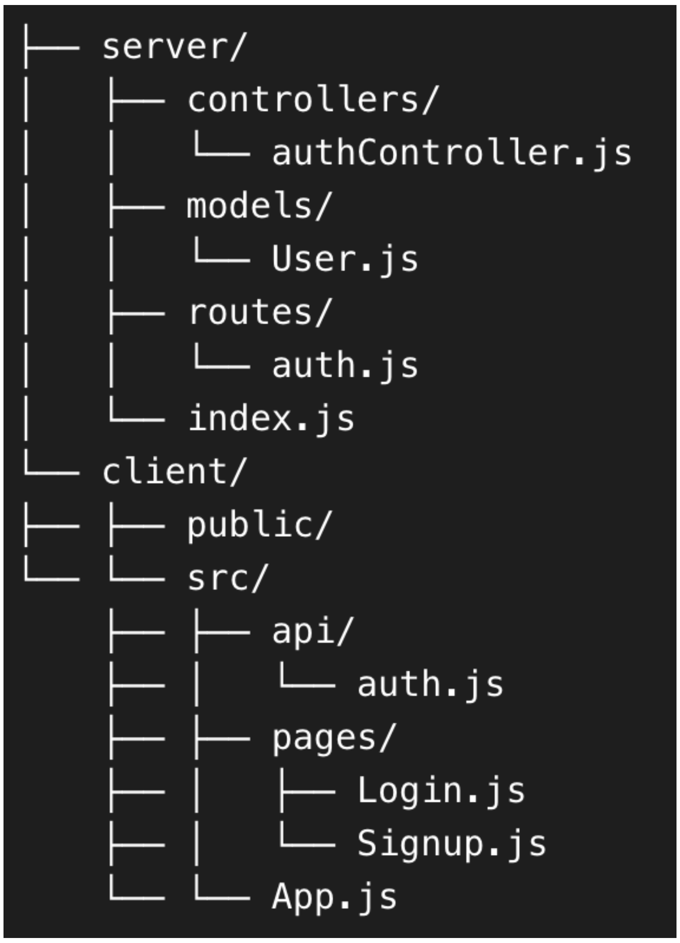
Medium Full Stack

- User authentication (signup, login)
- CRUD posts
- Commenting system
- Like/unlike posts

Firstly let us install packages:

- `npm i express jsonwebtoken dotenv cors bcryptjs`
- `npm install prisma --save-dev`
- `npm install @prisma/client`
- `npx prisma generate`

Folder structure:



```
├── server/
│   ├── controllers/
│   │   └── authController.js
│   ├── models/
│   │   └── User.js
│   ├── routes/
│   │   └── auth.js
│   └── index.js
└── client/
├── ├── public/
└── └── src/
    ├── ├── api/
    ├── │   └── auth.js
    ├── ├── pages/
    ├── │   ├── Login.js
    ├── │   └── Signup.js
    └── └── App.js
```

Let us build all of them one by one!

Starting with the `server` inside the `index.js`

```
const express = require('express');
const cors = require('cors');
const { PrismaClient } = require('./generated/prisma');
require('dotenv').config();

const app = express();
const prisma = new PrismaClient();

app.use(cors({
    origin: 'http://localhost:5173',
    credentials: true
}));

app.use(express.json());

app.use('/api/auth', require('./routes/auth'));

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

`auth.js`

```
const express = require('express');
const router = express.Router();
const { signup, login } = require('../controllers/authControllers.js');

// @route   POST /api/auth/signup
// @desc    Register new user
// @access  Public
router.post('/signup', signup);

// @route   POST /api/auth/login
// @desc    Authenticate user
// @access  Public
router.post('/login', login);
```

```
module.exports = router;
```

authControllers.js

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { PrismaClient } = require('../generated/prisma');
const prisma = new PrismaClient();

exports.signup = async (req, res) => {
  console.log(req.body);
  try {
    const { username, email, password } = req.body;
    const hashed = await bcrypt.hash(password, 10);
    const user = await prisma.user.create({ data: { username, email, passwor
    const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET, { expires
    res.cookie('token', token, {
      httpOnly: true,
      secure: process.env.NODE_ENV === 'production',
      sameSite: 'lax',
      maxAge: 24 * 60 * 60 * 1000
    });
    res.status(201).json({ user: { id: user.id, username, email } });
  } catch (err) {
    console.error(err);
    res.status(400).json({ error: 'Signup failed' });
  }
};


exports.login = async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await prisma.user.findUnique({ where: { email } });
    if (!user) return res.status(404).json({ error: 'User not found' });
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(401).json({ error: 'Invalid credentials' });
    const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET, { expires
      res.cookie('token', token, {
        httpOnly: true,
        secure: process.env.NODE_ENV === 'production',
        sameSite: 'lax',
        maxAge: 24 * 60 * 60 * 1000
      });

      res.json({ user: { id: user.id, username: user.username, email } });
    } catch (err) {
      console.error(err);
      res.status(500).json({ error: 'Login failed' });
    }
  };
```

.env

```
DATABASE_URL="postgresql://myuser:kartik@localhost:5432/mydb?schema=
JWT_SECRET="@This is some random string that cannot be guesssed tis eas
PORT=4444
```

THE BASIC backend code is done, now let us implement the frontend code:

Installation:

api/auth.js

```
import axios from 'axios';

const API_URL = 'http://localhost:4444/api/auth';

export const signup = (username, email, password) => {
  return axios.post(
    `${API_URL}/signup`,
    { username, email, password },
    { withCredentials: true }
  );
};

export const login = (email, password) ⇒ {
   return axios.post(
      `${API_URL}/login`,
      { email, password },
      { withCredentials: true }
   );
};
```

App.jsx

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Signup from "./pages/Signup";
import Login from "./pages/Login";
import Dashboard from "./pages/Dashboard";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Signup />} />
        <Route path="/signup" element={<Signup />} />
        <Route path="/login" element={<Login />} />
        <Route path="/dashboard" element={<Dashboard />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Dashboard.jsx

```
import React from "react";

const Dashboard = () ⇒ {
  return (
    <div>
      <h1>Welcome to the dashbaord</h1>
    </div>
  );
};

export default Dashboard;
```

Login.jsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { login } from "../api/auth";

export default function Login() {
  const [form, setForm] = useState({ email: "", password: "" });
  const navigate = useNavigate();

  const handleChange = ({ target: { name, value } }) ⇒
    setForm((f) ⇒ ({ ...f, [name]: value }));

  const handleSubmit = async (e) ⇒ {
    e.preventDefault();
    try {
      await login(form.email, form.password);
      navigate("/dashboard");
    } catch (err) {
      console.error("Login error:", err);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        name="email"
        type="email"
        placeholder="Email"
        value={form.email}
        onChange={handleChange}
      />
      <input
        name="password"
        type="password"
        placeholder="Password"
        value={form.password}
        onChange={handleChange}
      />
      <button type="submit">Log In</button>
    </form>
  );
}
```

signup.jsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { signup } from "../api/auth";

export default function Signup() {
  const [form, setForm] = useState({ username: "", email: "", password: "" });
  const navigate = useNavigate();

  const handleChange = ({ target: { name, value } }) ⇒
    setForm((f) ⇒ ({ ...f, [name]: value }));

  const handleSubmit = async (e) ⇒ {
    e.preventDefault();
    try {
      console.log(form)
      await signup(form.username, form.email, form.password);
      navigate("/dashboard");
    } catch (err) {
      console.error("Signup error:", err);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        name="username"
        placeholder="Username"
        value={form.username}
        onChange={handleChange}
      />
      <input
        name="email"
        type="email"
        placeholder="enter email"
        autoComplete="email"
        value={form.email}
        onChange={handleChange}
      />
      <input
        name="password"
        type="password"
        placeholder="Password"
        autoComplete="current-password"
        value={form.password}
        onChange={handleChange}
      />
      <button type="submit">Sign Up</button>
    </form>
  );
}
```

Now let us build an admin portal so that we can add blogs

let us create a model for `Post`

Of course inside the `prisma`

```
model Post {
  id        Int      @id @default(autoincrement())
  title     String
  content   String
  author    User     @relation(fields: [authorId], references: [id])
  authorId  Int
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
```

Inside `controllers` we can add:

```
|– server/
        | – controllers/
                  |– authController.js
                  |—– postController.js

|—– client/
        |– src/
               |– api/
                      |– auth.js
                      |—– posts.js
               |– pages/
                      |– Signup.jsx
                      |– Login.jsx
                      |– AdminPortal.jsx
                      |– NewPost.jsx
        └── App.jsx
        └── public/
```

`AdminPortal.jsx`

```
import { useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { deletePost, fetchPosts } from "../api/posts.js";

export default function AdminPortal() {
  const [posts, setPosts] = useState([]);
  const navigate = useNavigate();

  useEffect(() => {
    (async () => setPosts((await fetchPosts()).data))();
  }, []);

  const handleDelete = async (id) => {
    if (confirm("Delete this post?")) {
      await deletePost(id);
      setPosts(posts.filter((p) => p.id !== id));
    }
  };

  return (
    <div>
      <button onClick={() => navigate("/admin/new")}>New Post</button>
      <ul>
        {posts.map(({ id, title }) => (
          <li key={id}>
            <h3>{title}</h3>
            <button onClick={() => navigate(`/admin/edit/${id}`)}>Edit</button>
            <button onClick={() => handleDelete(id)}>Delete</button>
          </li>
        ))}
      </ul>
    </div>
  );
}
```