MY databse name admin course and my collection name is admin with id ,email , passsword now give me signup , login ,logout code in express js with **JWT (token-based)** and also guide me how to test in thunder client in Vs Code.

Below is **JWT (token-based) authentication** using **Express + MongoDB**, with:-

- □ Signup
- □ Login (returns TOKEN)
- □ Protected route
- □ Logout (client-side token removal)
- □ Thunder Client testing guide

1□ Install Required Packages

```
npm install express mongoose bcrypt jsonwebtoken
```

2□ Folder Structure

```
project/
├── server.js
├── models/
│      └── Admin.js
├── middleware/
│      └── auth.js
```

3□ Admin Model (`models/Admin.js`)

```
const mongoose = require("mongoose");

const adminSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  }
});
```

```
module.exports = mongoose.model("admin", adminSchema);
```

---

4 JWT Middleware (`middleware/auth.js`)

```
const jwt = require("jsonwebtoken");

const authMiddleware = (req, res, next) => {
  const token = req.header("Authorization");

  if (!token) {
    return res.status(401).json({ message: "No token, access denied" });
  }

  try {
    const decoded = jwt.verify(token, "jwtsecretkey");
    req.adminId = decoded.id;
    next();
  } catch (err) {
    res.status(401).json({ message: "Invalid token" });
  }
};

module.exports = authMiddleware;
```

---

5 Express Server (`server.js`)

```
const express = require("express");
const mongoose = require("mongoose");
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");

const Admin = require("./models/Admin");
const authMiddleware = require("./middleware/auth");

const app = express();
app.use(express.json());

// MongoDB Connection
mongoose
  .connect("mongodb://127.0.0.1:27017/admin course")
  .then(() => console.log("MongoDB Connected"))
  .catch(err => console.log(err));

/* ================= SIGNUP ================= */
```

```
app.post("/signup", async (req, res) => {
  const { email, password } = req.body;

  const existingAdmin = await Admin.findOne({ email });
  if (existingAdmin) {
    return res.status(400).json({ message: "Email already exists" });
  }

  const hashedPassword = await bcrypt.hash(password, 10);

  const newAdmin = new Admin({
    email,
    password: hashedPassword
  });

  await newAdmin.save();

  res.status(201).json({ message: "Signup successful" });
});

/* ================= LOGIN ================= */
app.post("/login", async (req, res) => {
  const { email, password } = req.body;

  const admin = await Admin.findOne({ email });
  if (!admin) {
    return res.status(400).json({ message: "Invalid email or password" });
  }

  const isMatch = await bcrypt.compare(password, admin.password);
  if (!isMatch) {
    return res.status(400).json({ message: "Invalid email or password" });
  }

  const token = jwt.sign(
    { id: admin._id },
    "jwtsecretkey",
    { expiresIn: "1h" }
  );

  res.json({
    message: "Login successful",
    token
  });
});

/* ================= PROTECTED ROUTE ================= */
app.get("/dashboard", authMiddleware, (req, res) => {
  res.json({
    message: "Welcome to Admin Dashboard",
    adminId: req.adminId
  });
});

/* ================= LOGOUT ================= */
// JWT logout = delete token on frontend
app.post("/logout", (req, res) => {
```

```
  res.json({ message: "Logout successful (delete token on client)" });
});

app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```

---

6 Test with Thunder Client (VS Code)

Signup

- **Method:** POST
- **URL:** `http://localhost:3000/signup`
- **Body → JSON**

```
{
  "email": "admin@gmail.com",
  "password": "123456"
}
```

Login (GET TOKEN)

- **Method:** POST
- **URL:** `http://localhost:3000/login`
- **Body → JSON**

```
{
  "email": "admin@gmail.com",
  "password": "123456"
}
```

Response:

```
{
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Copy this token

□ Access Protected Route

- **Method:** GET
- **URL:** `http://localhost:3000/dashboard`
- **Headers:**

```
Authorization: <PASTE_TOKEN_HERE>
```

□ Response:

```
{
  "message": "Welcome to Admin Dashboard",
  "adminId": "65a..."
}
```

□ Without token:

```
{
  "message": "No token, access denied"
}
```

□ Logout

- **Method:** POST
- **URL:** `http://localhost:3000/logout`

□ Response:

```
{
  "message": "Logout successful (delete token on client)"
}
```

□ JWT logout means:

- Remove token from **localStorage / cookies / Thunder Client header**

□ JWT Flow Explained (Simple)

1. Login → Server gives **TOKEN**
2. Client stores token
3. Client sends token in **Authorization header**
4. Middleware verifies token
5. Access granted □

Below is a **simple React.js frontend** that works **directly with your JWT-based Express API** (signup, login, protected dashboard, logout).

I'll keep it **clean + beginner-friendly**.

---

1 Create React App

```
npx create-react-app admin-auth
cd admin-auth
npm start
```

---

2 Project Structure (Important Files)

```
src/
├── App.js
├── Login.js
├── Signup.js
├── Dashboard.js
└── api.js
```

---

3 API Helper (`src/api.js`)

This handles API calls + token.

```
const API_URL = "http://localhost:3000";

export const signup = async (data) => {
  const res = await fetch(`${API_URL}/signup`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(data)
  });
```

```
  return res.json();
};

export const login = async (data) => {
  const res = await fetch(`${API_URL}/login`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(data)
  });
  return res.json();
};

export const getDashboard = async () => {
  const token = localStorage.getItem("token");

  const res = await fetch(`${API_URL}/dashboard`, {
    headers: {
      Authorization: token
    }
  });

  return res.json();
};
```

---

4 Signup Page (`src/Signup.js`)

```
import { useState } from "react";
import { signup } from "./api";

function Signup() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [msg, setMsg] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    const res = await signup({ email, password });
    setMsg(res.message);
  };

  return (
    <div>
      <h2>Signup</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="email"
          placeholder="Email"
          onChange={(e) => setEmail(e.target.value)}
          required
        />
        <br /><br />
        <input
          type="password"
          placeholder="Password"
```

```
          onChange={(e) => setPassword(e.target.value)}
          required
        />
        <br /><br />
        <button>Signup</button>
      </form>
      <p>{msg}</p>
    </div>
  );
}

export default Signup;
```

5️⃣ Login Page (`src/Login.js`):-

```
import { useState } from "react";
import { login } from "./api";

function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [msg, setMsg] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    const res = await login({ email, password });

    if (res.token) {
      localStorage.setItem("token", res.token);
      window.location.href = "/dashboard";
    } else {
      setMsg(res.message);
    }
  };

  return (
    <div>
      <h2>Login</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="email"
          placeholder="Email"
          onChange={(e) => setEmail(e.target.value)}
          required
        />
        <br /><br />
        <input
          type="password"
          placeholder="Password"
          onChange={(e) => setPassword(e.target.value)}
          required
        />
        <br /><br />
        <button>Login</button>
```

```
      </form>
      <p>{msg}</p>
    </div>
  );
}

export default Login;
```

6️⃣ Dashboard (Protected Page) (`src/Dashboard.js`):-

```
import { useEffect, useState } from "react";
import { getDashboard } from "./api";

function Dashboard() {
  const [data, setData] = useState("");

  useEffect(() => {
    getDashboard().then(res => {
      if (res.message === "Invalid token" || res.message === "No token,
access denied") {
        window.location.href = "/login";
      } else {
        setData(res.message);
      }
    });
  }, []);

  const logout = () => {
    localStorage.removeItem("token");
    window.location.href = "/login";
  };

  return (
    <div>
      <h2>{data}</h2>
      <button onClick={logout}>Logout</button>
    </div>
  );
}

export default Dashboard;
```

7️⃣ Routing (`src/App.js`)

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

```
import Login from "./Login";
import Signup from "./Signup";
import Dashboard from "./Dashboard";

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

export default App;
```

8️⃣ Install Router

```
npm install react-router-dom
```

9️⃣ How Full Flow Works

1. **Signup** → MongoDB stores admin
2. **Login** → JWT stored in `localStorage`
3. **Dashboard** → Token sent in header
4. **Logout** → Token removed