MERN

Full-Stack MERN Authentication System

MongoDB + Express + React + Node.js

(Registration, Login, and Dashboard) from scratch, It includes setting up both backend (Node.js, Express, MongoDB) and frontend (React), using JWT, borypt, and routing

Full-Stack MERN Authentication System Setup Guide (Login/Register + Dashboard)

STEP-BY-STEP SETUP

1. BACKEND SETUP

1.1 Create backend directory

```
mkdir backend
otj backend
nonn inr...e
```

1.2. install backend dependencies

```
nom install express mongpass cors
dotenv bo apits jsoriwottaken
```

1.3. Create backend files/folders

```
mkdr. controllers models routes
touch server.js
```

1.4. Create.env file in backend/

```
PORT=5000
MONCO, URL=mogodb://1270;10,1.217/17
JWT_SEERET your_WL_secret_here
```

Realace your_wT_secret_hers with-a strong randon j.shing.

1.5. server.js setup

1.6. User:Js setup

```
const mongoase require((exprese);
const mongoase  require(mongoase);
const core require[:cone);
const dotenv  require((dotenv);

doterv..co:fig!);

const app + express|
app use(core));
app use(express (soml);

Const authRoutes require((outouInRoutes);
app usery.pp.(auth : /oux Routes);

mongpase
connect process.env.MONGO_URI,
  useNewtmUL;Parser: freg;
  useVFlleoTopology : \Ta0.,UsedUplogy.|
  puist)c : (|enth a.coresk "MorigOBB
  connection error : : em|

Const PORT, procesvm PORT I 5000.

app.listen (PORT);

(2xio.comfogciki"Server running in 8%
rt (PORT.)
```

ERN

Full-Stack MERN Authentication System

MongoDB + Express + React + Node.js

(Registration, Login, and Dashboard) from **scratch**. It includes setting up the backend (Node.js, Express, MongoDB) and frontend (React), using JWT, bcrypt, and routing.

📄 Full-Stack MERN Authentication System Setup Guide (Login/Register + Dashboard)

📁 Project Folder Structure

```
mern-auth-app/
├── backend/
│   ├── controllers/
│   ├── models/
│   ├── routes/
│   ├── .env
│   └── server.js
└── frontend/
    ├── public/
    ├── src/
    │   ├── components/
    │   ├── pages/
    │   ├── App.js
    │   └── index.js
    └── package.json
```

🔧 STEP-BY-STEP SETUP

🖥️ 1. BACKEND SETUP

1.1. Create backend directory

mkdir backend

cd backend

npm init -y

1.2. Install backend dependencies

npm install express mongoose cors dotenv bcryptjs jsonwebtoken

1.3. Create backend files/folders

mkdir controllers models routes

touch server.js

1.4. Create .env file in backend/

PORT=5000

MONGO_URI=mongodb://127.0.0.1:27017/mern-auth

JWT_SECRET=your_jwt_secret_here

Replace your_jwt_secret_here with a strong random string.

1.5. server.js setup

```
// backend/server.js

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

const dotenv = require("dotenv");


dotenv.config();
```

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

const authRoutes = require("./routes/authRoutes");
app.use("/api/auth", authRoutes);

mongoose
  .connect(process.env.MONGO_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => console.log("MongoDB connected"))
  .catch((err) => console.error("MongoDB connection error:", err));

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

1.6. User Model

```
// backend/models/User.js
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  username: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
```

```
});

module.exports = mongoose.model("User", userSchema);
```

1.7. Auth Routes

```
// backend/routes/authRoutes.js
const express = require("express");
const router = express.Router();
const User = require("../models/User");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

// Register
router.post("/register", async (req, res) => {
 const { username, email, password } = req.body;
 try {
  const existingUser = await User.findOne({ email });
  if (existingUser)
   return res.status(400).json({ message: "User already exists" });

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

  const newUser = new User({ username, email, password: hashedPassword });
  await newUser.save();

  res.status(201).json({ message: "User registered successfully" });
 } catch (err) {
```

```
    res.status(500).json({ message: "Server error" });
  }
});

// Login
router.post("/login", async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user)
      return res.status(400).json({ message: "Invalid credentials" });

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

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1d",
    });

    res.status(200).json({
      token,
      user: { username: user.username, email: user.email },
    });
  } catch (err) {
    res.status(500).json({ message: "Server error" });
  }
```

```
});

module.exports = router;
```

1.8. Start backend

```
node server.js
```

🍃 MongoDB Setup

- Download and install **MongoDB Community Edition** and **MongoDB Compass**
- Start MongoDB service.
- Open Compass → create or view database mern-auth
- You will see your users table and hashed passwords.

🌐 2. FRONTEND SETUP

2.1. Create React frontend

```
npx create-react-app frontend

cd frontend

npm install axios react-router-dom
```

2.2. Add Routing in App.js

```
// frontend/src/App.js

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

import Register from "./pages/Register";

import Login from "./pages/Login";
```

```
import Dashboard from "./pages/Dashboard";

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

export default App;
```

2.3. Register Page

```
// frontend/src/pages/Register.js
import { useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
```

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

  const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value });

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post("http://localhost:5000/api/auth/register", form);
      alert("Registration successful");
      navigate("/login");
    } catch (err) {
      alert(err.response.data.message || "Registration failed");
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input name="username" placeholder="Username" onChange={handleChange}
required />
      <input name="email" type="email" placeholder="Email" onChange={handleChange}
required />
      <input name="password" type="password" placeholder="Password"
onChange={handleChange} required />
      <button type="submit">Register</button>
    </form>
  );
```

```
}

export default Register;
```

2.4. Login Page

```
// frontend/src/pages/Login.js
import { useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";

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

 const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value });

 const handleSubmit = async (e) => {
  e.preventDefault();
  try {
   const res = await axios.post("http://localhost:5000/api/auth/login", form);
   localStorage.setItem("token", res.data.token);
   alert("Login successful");
   navigate("/dashboard");
  } catch (err) {
   alert(err.response.data.message || "Login failed");
  }
 };
```

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

export default Login;
```

2.5. Dashboard Page

```
// frontend/src/pages/Dashboard.js
function Dashboard() {
  return (
    <div>
      <h2>Welcome to the Dashboard</h2>
      <p>You are logged in!</p>
    </div>
  );
}

export default Dashboard;
```

✅ How to Run the Full App

In two terminals:

1. **Backend**

cd backend

node server.js

2. **Frontend**

cd frontend

npm start

Open http://localhost:3000 to test frontend

🧠 Summary Tips

- Use .env to store secrets (MONGO_URI, JWT_SECRET)
- Hash passwords before saving to MongoDB using bcryptjs
- Use jsonwebtoken to create secure login tokens
- Secure protected routes by checking token in future (optional enhancement)
- React frontend communicates with backend via axios
