package.json

```
{
 "name": "user-auth",
 "version": "1.0.0",
 "main": "server.js",
 "type": "module",
 "scripts": {
  "start": "node server.js"
 },
 "dependencies": {
  "bcryptjs": "^2.4.3",
  "dotenv": "^16.0.3",
  "express": "^4.18.2",
  "jsonwebtoken": "^9.0.0",
  "mongoose": "^7.0.0"
 }
}
```

⚙️ Step 2: .env

```
PORT=5000
MONGO_URI=mongodb://localhost:27017/userAuthDB
JWT_SECRET=mysecretkey
```

🧩 Step 3: config/db.js

```
import mongoose from "mongoose";


const connectDB = async () => {
```

```
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log("✅ MongoDB Connected");
  } catch (err) {
    console.error("❌ Database connection failed:", err.message);
    process.exit(1);
  }
};

export default connectDB;
```

👤 Step 4: models/User.js

```
import mongoose from "mongoose";

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

export default mongoose.model("User", userSchema);
```

🔐 Step 5: routes/auth.js

```
import express from "express";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";
import User from "../models/User.js";
```

```
const router = express.Router();


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


    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = new User({ name, email, password: hashedPassword });
    await newUser.save();


    res.status(201).json({ msg: "User registered successfully" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});


// 🔓 Login
router.post("/login", async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user) return res.status(400).json({ msg: "User not found" });
```

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

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "1h" });
    res.json({ token });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

export default router;
```

🧰 Step 6: middleware/authMiddleware.js

```
import jwt from "jsonwebtoken";

const auth = (req, res, next) => {
  const token = req.header("Authorization");
  if (!token) return res.status(401).json({ msg: "No token, authorization denied" });

  try {
    const decoded = jwt.verify(token.split(" ")[1], process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    res.status(400).json({ msg: "Invalid token" });
  }
```

```
};


export default auth;
```

🚀 Step 7: server.js

```
import express from "express";

import dotenv from "dotenv";

import connectDB from "./config/db.js";

import authRoutes from "./routes/auth.js";


dotenv.config();

const app = express();


connectDB();


app.use(express.json());

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


app.get("/", (req, res) => {

  res.send("User Authentication API Running 🚀");

});


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

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