SOURCE CODE:

package.json

```
{
  “name” : “user-auth-system” ,
  “version” : “1.0.0” ,
  “description” : “Node.js user authentication system with JWT and MongoDB” ,
  “main” : “server.js” ,
  “scripts” : {
    “start” : “node server.js” ,
    “dev” : “nodemon server.js”
  },
  “dependencies” : {
    “bcryptjs” : “^2.4.3” ,
    “cookie-parser” : “^1.4.6” ,
    “cors” : “^2.8.5” ,
    “dotenv” : “^16.3.1” ,
    “express” : “^4.18.2” ,
    “jsonwebtoken” : “^9.0.2” ,
    “mongoose” : “^7.6.1”
  },
  “devDependencies” : {
```

```
    “nodemon”: “^3.0.3”
  }
}
```

. env

```
PORT=5000
MONGO_URI=mongodb://localhost:27017/userAuthDB
JWT_SECRET=mysecretkey123
```

Config/db.js

```
Const mongoose = require(“mongoose”);

Const connectDB = async () => {
  Try {
    Await mongoose.connect(process.env.MONGO_URI);
    Console.log(“✅ MongoDB Connected Successfully”);
  } catch (error) {
    Console.error(“❌ MongoDB Connection Failed:”, error.message);
    Process.exit(1);
  }
};

Module.exports = connectDB;
```

Models/User.js

```
Const mongoose = require(“mongoose”);
Const bcrypt = require(“bcryptjs”);

Const userSchema = new mongoose.Schema({
  Name: { type: String, required: true },
  Email: { type: String, required: true, unique: true },
  Password: { type: String, required: true },
});

// Encrypt password before saving
userSchema.pre(“save”, async function (next) {
  if (!this.isModified(“password”)) next();
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

userSchema.methods.matchPassword = async function (enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};

Module.exports = mongoose.model(“User”, userSchema);
```

utils/generateToken.js

```
const jwt = require(“jsonwebtoken”);

const generateToken = (id) => {
```

```
  return jwt.sign({ id }, process.env.JWT_SECRET, {
    expiresIn: “30d”,
  });
};

Module.exports = generateToken;
```

Controllers/authControllers.js

```
Const User = require(“../models/User”);
Const generateToken = require(“../utils/generateToken”);

Const registerUser = async (req, res) => {
  Try {
    Const { name, email, password } = req.body;

    Const userExists = await User.findOne({ email });
    If (userExists) return res.status(400).json({ message: “User already exists” });

    Const user = await User.create({ name, email, password });
    If (user) {
      Res.status(201).json({
        _id: user.id,
        Name: user.name,
        Email: user.email,
        Token: generateToken(user.id),
      });
```

```
    } else {
      Res.status(400).json({ message: “Invalid user data” });
    }
  } catch (error) {
    Res.status(500).json({ message: error.message });
  }
};

Const loginUser = async (req, res) => {
  Const { email, password } = req.body;

  Const user = await User.findOne({ email });

  If (user && (await user.matchPassword(password))) {
    Res.json({
      _id: user.id,
      Name: user.name,
      Email: user.email,
      Token: generateToken(user.id),
    });
  } else {
    Res.status(401).json({ message: “Invalid email or password” });
  }
}
Const getUserProfile = async (req, res) => {
  Const user = await User.findById(req.user.id).select( “-password” );
  If (user) res.json(user);
```

```
  Else res.status(404).json({ message: “User not found” });
};

Module.exports = { registerUser, loginUser, getUserProfile };
```

middleware/authMiddleware.js

```
const jwt = require(“jsonwebtoken”);
const User = require(“../models/User”);

const protect = async (req, res, next) => {
  let token;

  if (req.headers.authorization && req.headers.authorization.startsWith(“Bearer”)) {
    try {
      token = req.headers.authorization.split(“ “)[1];
      const decoded = jwt.verify(token, process.env.JWT_SECRET);
      req.user = await User.findById(decoded.id).select(“-password”);
      next();
    } catch (error) {
      Res.status(401).json({ message: “Not authorized, invalid token” });
    }
  }

  If (!token) res.status(401).json({ message: “Not authorized, no token” });
};
```

```
Module.exports = { protect };
```

routes/authRoutes.js

```
const express = require(“express”);
const { registerUser, loginUser, getUserProfile } = require(“
        ../controllers/authController”);
const { protect } = require(“../middleware/authMiddleware”);

const router = express.Router();

router.post(“/register”, registerUser);
router.post(“/login”, loginUser);
router.get(“/profile”, protect, getUserProfile);

module.exports = router;
```

server.js

```
const express = require("express");
const dotenv = require("dotenv");
const cookieParser = require("cookie-parser");
const cors = require("cors");
const connectDB = require("./config/db");

dotenv.config();
```

```
connectDB();

const app = express();

app.use(express.json());
app.use(cookieParser());
app.use(cors());

app.use("/api/auth", require("./routes/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}`));
```