DANIYA TAZEEN(Daniya (Security + Crisis Safety Net)

backend/daniya-auth → JWT/OAuth login system.

backend/daniya-security → Encryption + emergency contact API.)

**Day 1 Log – Daniya (Security + Crisis Safety Net)**

**🎯 Goal**

Set up **MongoDB Atlas** connection in a test backend (security-experiments).

**✅ Steps Completed**

**1. Installed Node.js & npm**

* Installed Node.js v22.19.0 (confirmed with):

node -v

npm -v

Output:

v22.19.0

10.9.3

**2. Created GitHub Repo**

* Repo: **SIH-2025-MINDMATE-TEAM** (created by Qurrath).
* Cloned repo locally:

git clone https://github.com/qurrath2005/SIH-2025-MINDMATE-TEAM.git

cd SIH-2025-MINDMATE-TEAM

**3. Created Branch for Security Work**

git checkout -b security-auth

**4. Created security-experiments Folder**

mkdir security-experiments

cd security-experiments

**5. Initialized Node Project & Installed Dependencies**

npm init -y

npm install express mongoose dotenv

**6. Created .env File**

Inside security-experiments/.env:

MONGO\_URI=mongodb+srv://mindmate\_user:MindMate%402025@mindmatecluster.il9ivvo.mongodb.net/mindmate?retryWrites=true&w=majority&appName=MindMateCluster

**7. Created db.js**

const mongoose = require("mongoose");

async function connectDB() {

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;

**8. Created server.js**

require("dotenv").config();

const express = require("express");

const connectDB = require("./db");

const app = express();

// connect to MongoDB

connectDB();

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

res.send("MindMate Security Test ✅");

});

const PORT = 5000;

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

**9. Ran the Server**

node server.js

Output:

🚀 Server running on port 5000

✅ MongoDB Connected Successfully

Tested at <http://localhost:5000> → got response:

MindMate Security Test ✅

**🔟 Pushed Code to GitHub**

git add .

git commit -m "Added MongoDB Atlas connection test for security module"

git push origin security-auth

Branch created: **security-auth**  
Opened Pull Request → waiting for merge.

**✅ Day 1 Completed**

* MongoDB Atlas cluster setup ✔️
* Connection tested via Express backend ✔️
* Code pushed to GitHub (security-auth branch) ✔️

**📅 Day 2 Log – Daniya (Security + Crisis Safety Net)**

**🎯 Goal**

Build the **Authentication System** with:

* User model
* Signup + Login APIs
* JWT authentication

**✅ Step 1 — Checkout Branch**

cd C:\Users\Daniya\SIH-2025-MINDMATE-TEAM

git checkout security-auth

cd security-experiments

**✅ Step 2 — Create Folder Structure**

mkdir models

mkdir routes

**✅ Step 3 — Install Dependencies**

npm install bcryptjs jsonwebtoken

Output:

added 14 packages, and audited 100 packages in 2s

16 packages are looking for funding

found 0 vulnerabilities

**✅ Step 4 — Create User Model**

📂 models/User.js

const mongoose = require("mongoose");

const UserSchema = new mongoose.Schema({

name: { type: String, required: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

emergencyContacts: [{ type: String }]

}, { timestamps: true });

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

**✅ Step 5 — Create Auth Routes**

📂 routes/auth.js

const express = require("express");

const bcrypt = require("bcryptjs");

const jwt = require("jsonwebtoken");

const User = require("../models/User");

const router = express.Router();

// Signup

router.post("/signup", async (req, res) => {

try {

const { name, email, password } = req.body;

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

if (user) return res.status(400).json({ msg: "User already exists" });

const salt = await bcrypt.genSalt(10);

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

user = new User({ name, email, password: hashedPassword });

await user.save();

res.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: "Invalid credentials" });

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 });

}

});

module.exports = router;

**✅ Step 6 — Update server.js**

📂 server.js

require("dotenv").config();

const express = require("express");

const connectDB = require("./db");

const authRoutes = require("./routes/auth");

const app = express();

// Middleware

app.use(express.json());

// Connect DB

connectDB();

// Routes

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

const PORT = 5000;

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

**✅ Step 7 — Update .env**

📂 .env

MONGO\_URI=mongodb+srv://mindmate\_user:MindMate%402025@mindmatecluster.il9ivvo.mongodb.net/mindmate?retryWrites=true&w=majority&appName=MindMateCluster

JWT\_SECRET=supersecretkey

**✅ Step 8 — Test in Postman**

**🔹 Signup Request**

**POST** → http://localhost:5000/api/auth/signup

{

"name": "Daniya",

"email": "daniya@example.com",

"password": "123456"

}

✅ Response:

{ "msg": "User registered successfully" }

MongoDB Atlas → users collection entry created with hashed password.

**🔹 Login Request**

**POST** → http://localhost:5000/api/auth/login

{

"email": "daniya@example.com",

"password": "123456"

}

✅ Response:

{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..." }

Token decoded at jwt.io → contained id, iat, and exp.

**✅ Step 9 — Push to GitHub**

git add .

git commit -m "Day 2: Added User model, auth routes, JWT login/signup"

git push origin security-auth

Output:

Everything up-to-date

**🎯 Outcome**

* User model created in MongoDB Atlas
* Signup + Login APIs working with JWT authentication
* Tested successfully in Postman
* Day 2 code pushed to GitHub (branch: security-auth)