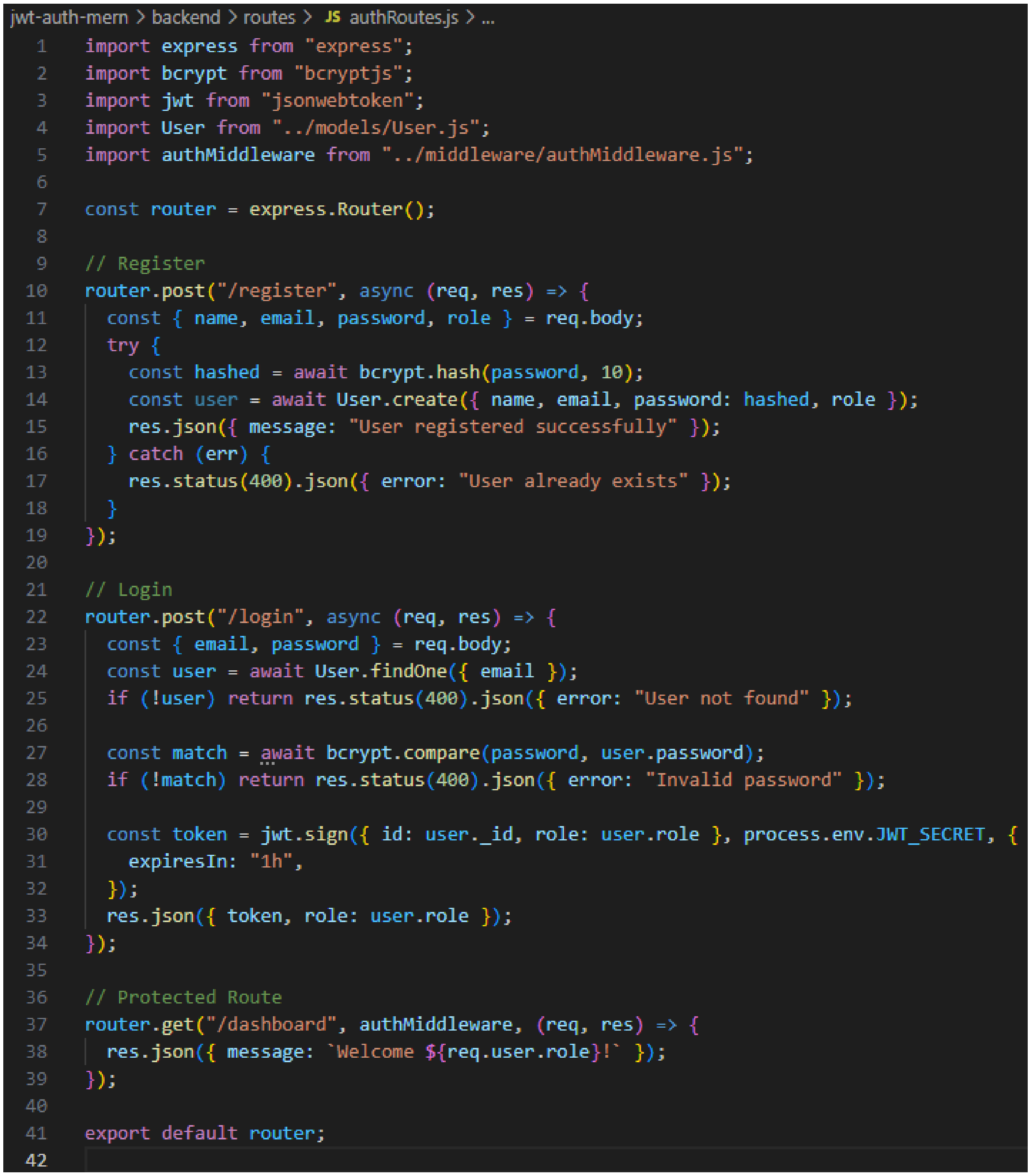
Experiment 6

Aim - Implement **authentication and user roles** with JWT.

Code -

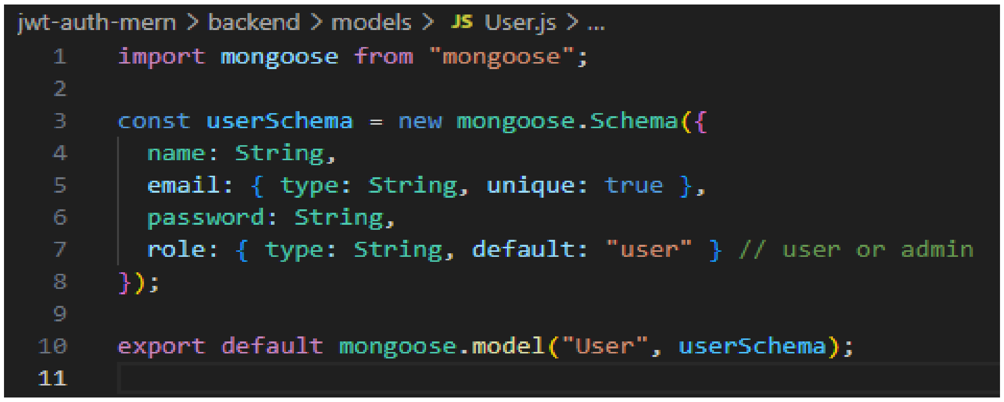
authRoutes.js -



jwt-auth-mern > backend > routes > JS authRoutes.js > ...

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

const router = express.Router();

// Register
router.post("/register", async (req, res) => {
  const { name, email, password, role } = req.body;
  try {
    const hashed = await bcrypt.hash(password, 10);
    const user = await User.create({ name, email, password: hashed, role });
    res.json({ message: "User registered successfully" });
  } catch (err) {
    res.status(400).json({ error: "User already exists" });
  }
});

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

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

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

// Protected Route
router.get("/dashboard", authMiddleware, (req, res) => {
  res.json({ message: `Welcome ${req.user.role}!` });
});

export default router;

```

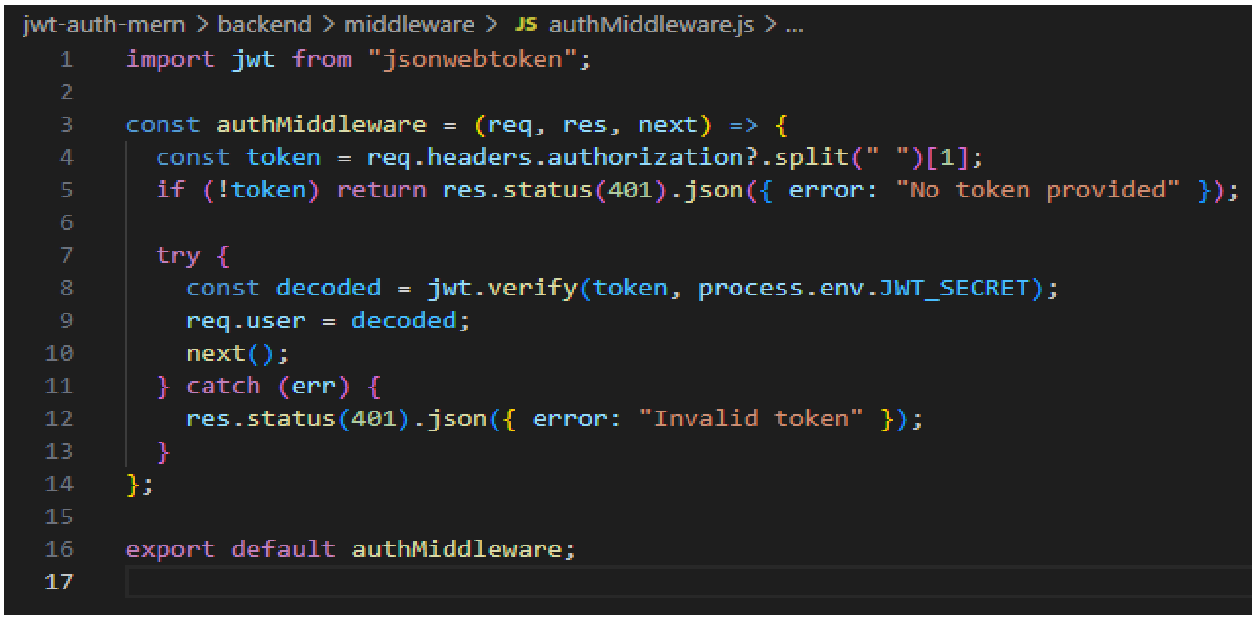
models/User.js -



jwt-auth-mern > backend > models > JS User.js > ...

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema({
  name: String,
  email: { type: String, unique: true },
  password: String,
  role: { type: String, default: "user" } // user or admin
});

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

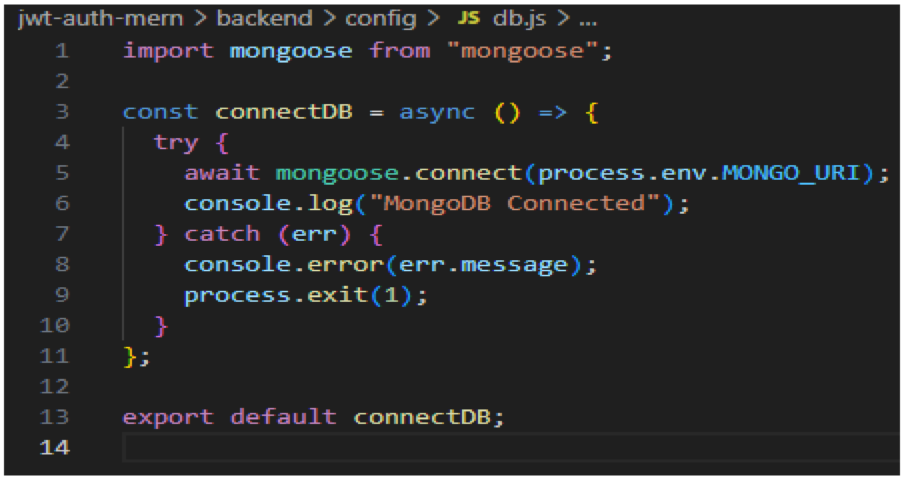
authMiddleware.js -



jwt-auth-mern > backend > middleware > JS authMiddleware.js > ...

```
import jwt from "jsonwebtoken";

const authMiddleware = (req, res, next) => {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.status(401).json({ error: "No token provided" });

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    res.status(401).json({ error: "Invalid token" });
  }
};

export default authMiddleware;
```

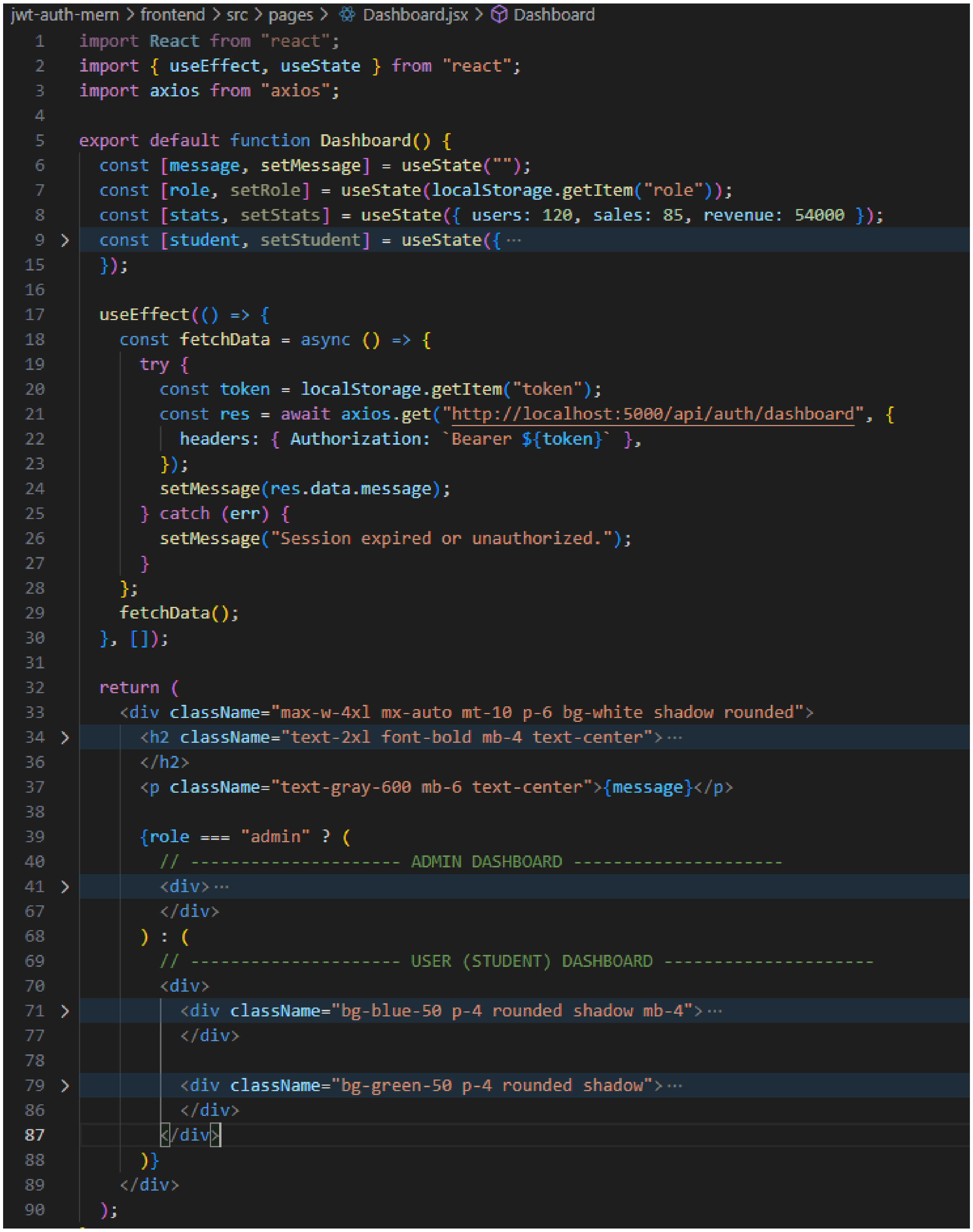
db.js -



jwt-auth-mern > backend > config > JS 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(err.message);
    process.exit(1);
  }
};

export default connectDB;
```

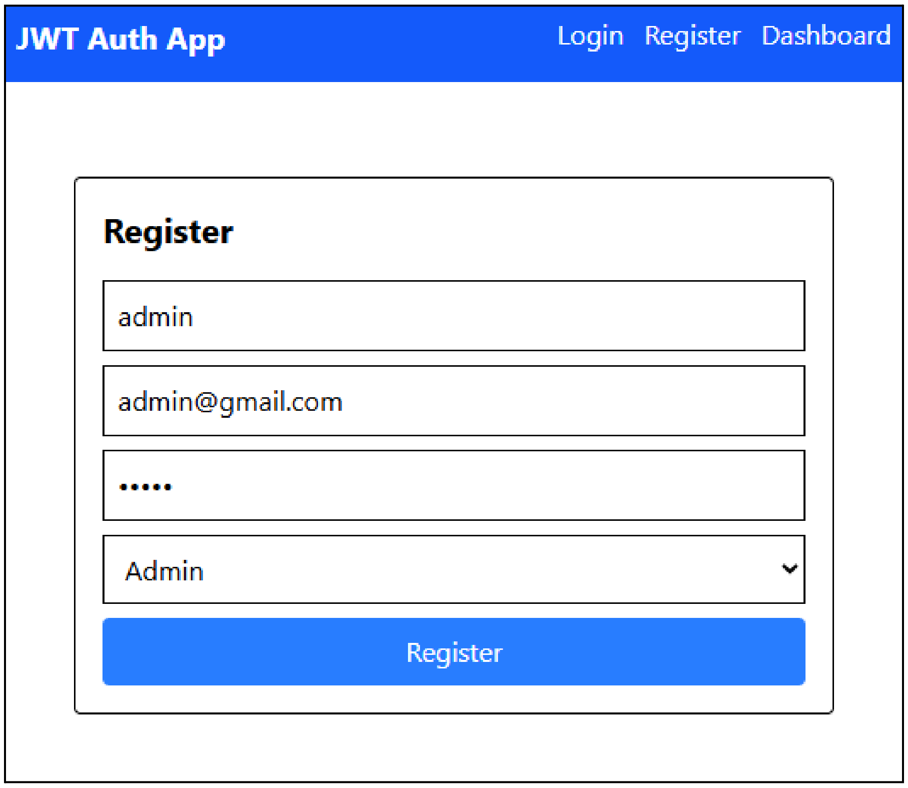
Dashboard.jsx -



jwt-auth-mern > frontend > src > pages > Dashboard.jsx > Dashboard

```
import React from "react";
import { useEffect, useState } from "react";
import axios from "axios";

export default function Dashboard() {
  const [message, setMessage] = useState("");
  const [role, setRole] = useState(localStorage.getItem("role"));
  const [stats, setStats] = useState({ users: 120, sales: 85, revenue: 54000 });
  const [student, setStudent] = useState({ ···
  });

  useEffect(() => {
    const fetchData = async () => {
      try {
        const token = localStorage.getItem("token");
        const res = await axios.get("http://localhost:5000/api/auth/dashboard", {
          headers: { Authorization: `Bearer ${token}` },
        });
        setMessage(res.data.message);
      } catch (err) {
        setMessage("Session expired or unauthorized.");
      }
    };
    fetchData();
  }, []);

  return (
    <div className="max-w-4xl mx-auto mt-10 p-6 bg-white shadow rounded">
      <h2 className="text-2xl font-bold mb-4 text-center">···
      </h2>
      <p className="text-gray-600 mb-6 text-center">{message}</p>

      {role === "admin" ? (
        // -------------------- ADMIN DASHBOARD --------------------
        <div>···
        </div>
      ) : (
        // -------------------- USER (STUDENT) DASHBOARD --------------------
        <div>
          <div className="bg-blue-50 p-4 rounded shadow mb-4">···
          </div>

          <div className="bg-green-50 p-4 rounded shadow">···
          </div>
        </div>
      )}
    </div>
  );
```

Output -


JWT Auth App

Register

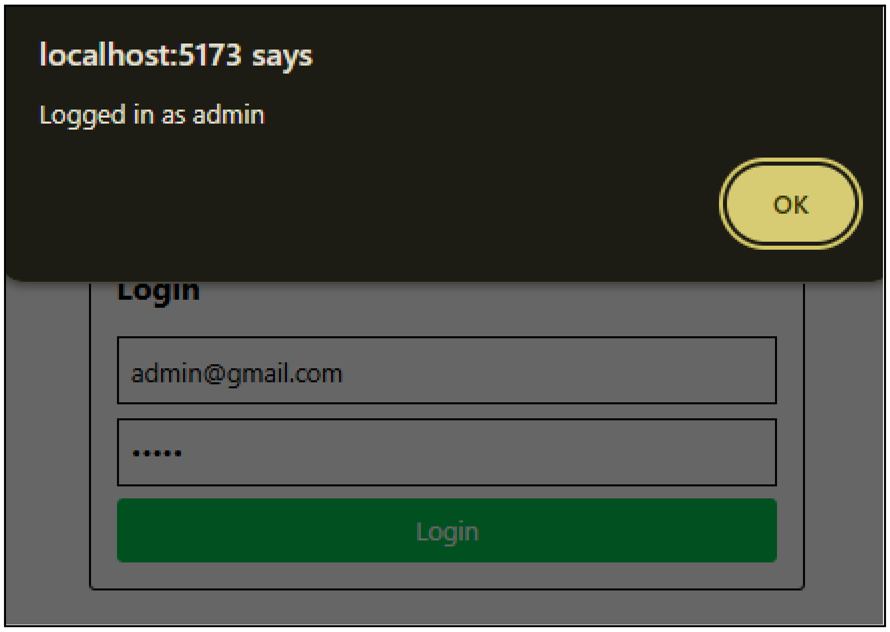
admin

admin@gmail.com

•••••

Admin

Fig 1.1 - Register for roles(admin)


localhost:5173 says

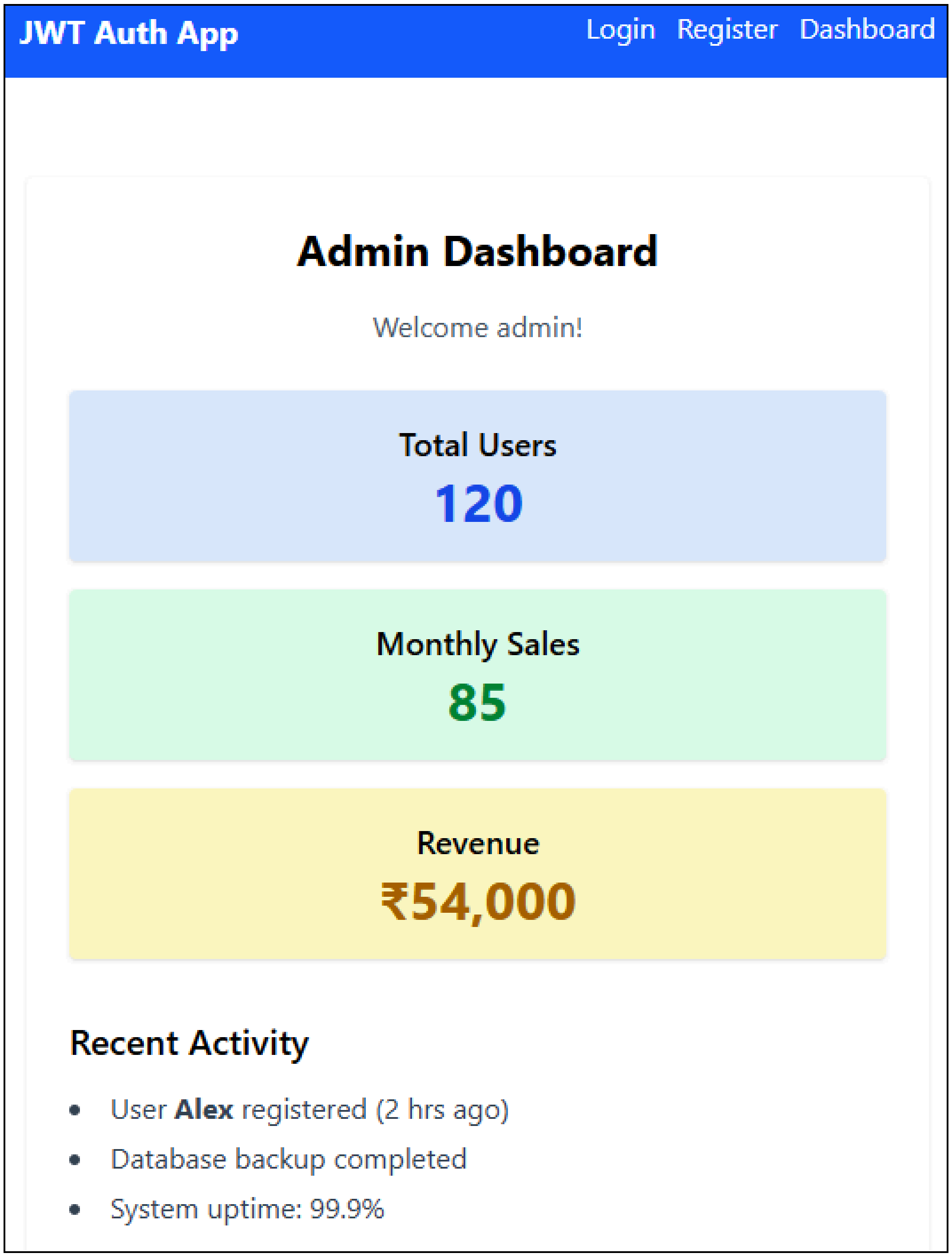
Logged in as admin

Login

admin@gmail.com

•••••

Fig 1.2 - Login for roles(admin)



Admin Dashboard

Welcome admin!

Total Users
120

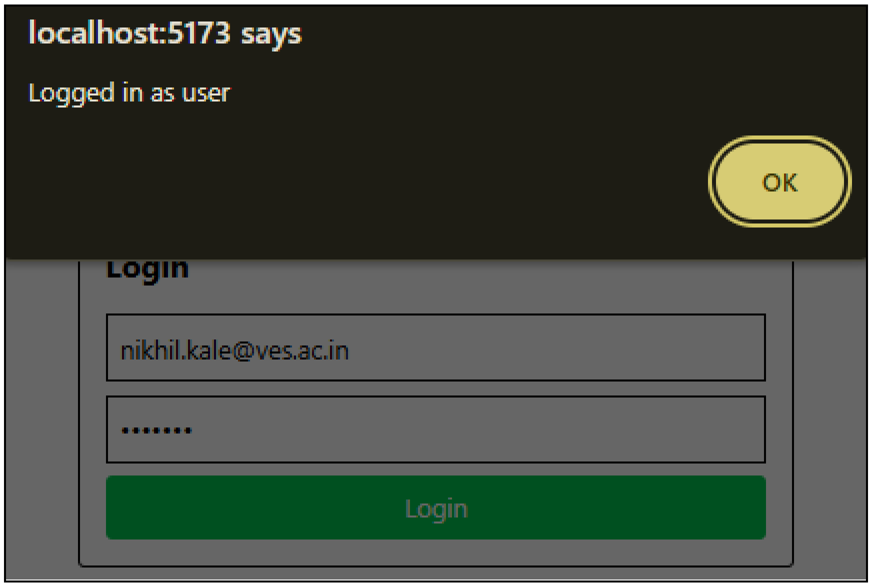
Monthly Sales
85

Revenue
₹54,000

Recent Activity

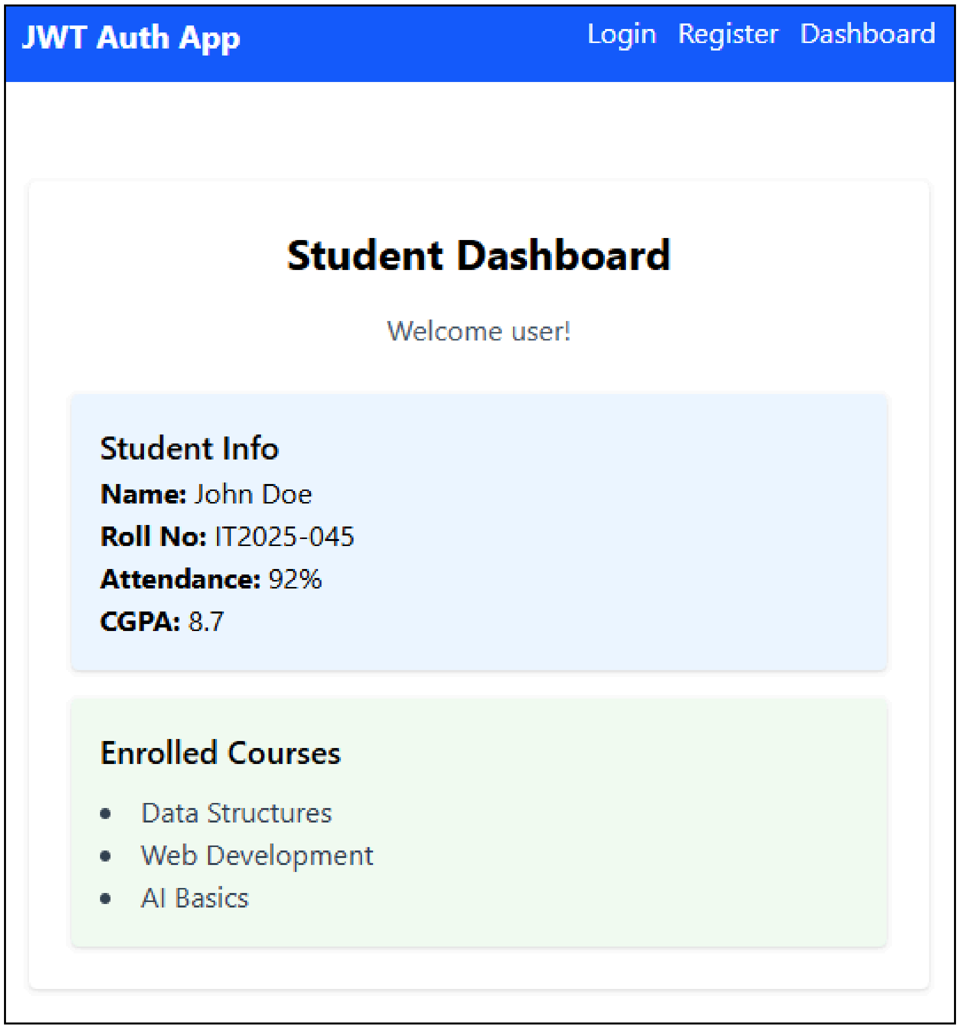
- User **Alex** registered (2 hrs ago)
- Database backup completed
- System uptime: 99.9%

Fig 1.3 - Admin Dashboard


localhost:5173 says

Logged in as user

Fig 2.1 - Login for role(student)



Student Dashboard

Welcome user!

Student Info

Name: John Doe

Roll No: IT2025-045

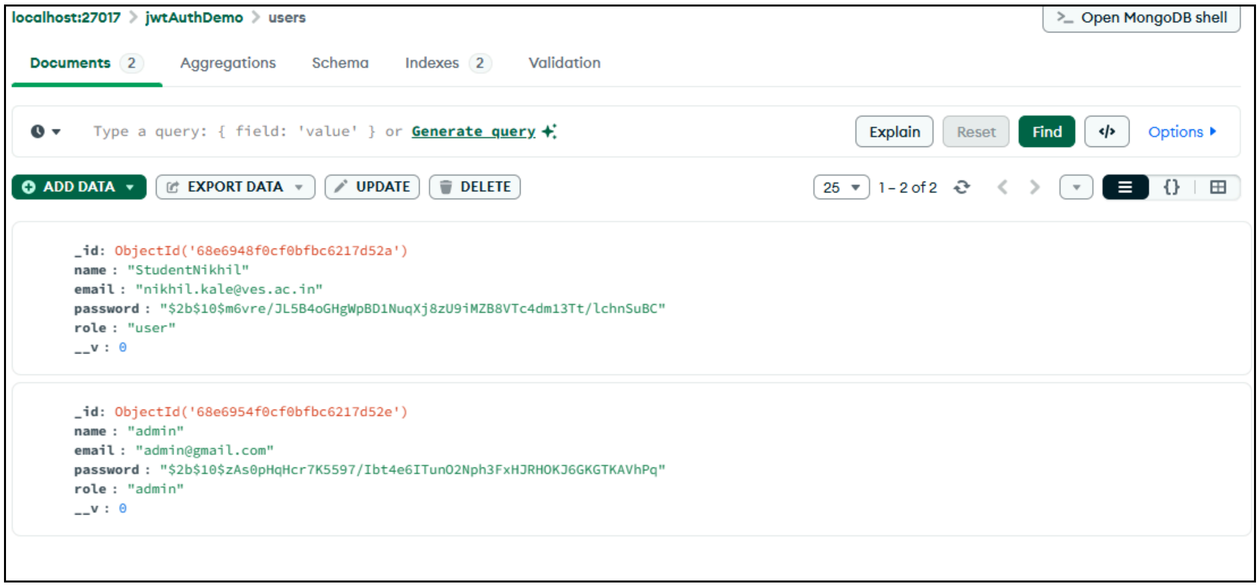
Attendance: 92%

CGPA: 8.7

Enrolled Courses

- Data Structures
- Web Development
- AI Basics

Fig 2.2 - Student Dashboard



```
_id: ObjectId('68e6948f0cf0bfbc6217d52a')
name : "StudentNikhil"
email : "nikhil.kale@ves.ac.in"
password : "$2b$10$m6vre/JL5B4oGHgWpBD1NuqXj8zU9iMZB8VTc4dm13Tt/lchnSuBC"
role : "user"
__v : 0
```

```
_id: ObjectId('68e6954f0cf0bfbc6217d52e')
name : "admin"
email : "admin@gmail.com"
password : "$2b$10$zAs0pHqHcr7K5597/Ibt4e6ITunO2Nph3FxHJRHOKJ6GKGTKAVhPq"
role : "admin"
__v : 0
```

Fig 3 - MongoDB interface