# Department of Computing

**CS-343: Web Technologies**

**Class:** BSCS-13E

# Open Ended Lab

**Date: 12.04.25**

# Instructor: Dr. Qaiser Riaz

# Open-Ended Lab: Real-Time IoT Monitoring System using MERN Stack

## Objective

Build a real-time dashboard to monitor IoT device data using the MERN stack. Students will:  
- Connect MongoDB Atlas with a Node.js/Express server using Mongoose.  
- Use middleware and error handling in Express.  
- Create RESTful APIs to send and retrieve device data.  
- Develop a React frontend with useEffect, useState, useContext to display real-time data.  
- Create a separate admin interface for uploading simulated device data.

## Schedule Breakdown

0:00 – 0:15 | Kickoff & Setup  
0:15 – 1:00 | Backend Development  
1:00 – 1:45 | Frontend Development (User Dashboard)  
1:45 – 2:15 | Admin Console Development  
2:15 – 2:45 | Testing, Debugging & Bonus Features  
2:45 – 3:00 | Demo & Wrap-up

## Lab Overview

You will build:  
- Backend API (Express + MongoDB Atlas)  
- IoT Data Dashboard (React)  
- Admin Console (React) for Data Upload

## Tools & Libraries

- MongoDB Atlas  
- Express + Mongoose  
- React (Hooks, Context API)  
- Effect (for API calls)  
- Cors, dotenv, body-parser (Express middlewares)

## Backend Instructions

1. Project Setup  
2. MongoDB Atlas Setup  
3. Express API

## 

## Frontend Instructions

1. Dashboard (User Side):  
- Create context: DeviceDataContext  
- Use useEffect to poll /api/data/latest every 5 seconds  
- Show temperature/humidity in components using cards/charts  
  
Admin Console:  
- Create a form with inputs for deviceId, temperature, humidity  
- On form submission, send POST to /api/data

## Learning Goals

Mongoose: Define models, schemas, connect to MongoDB  
Express: Build RESTful APIs with error handling  
React: Handle state/context and lifecycle hooks  
Architecture: Design modular frontend/backend separation  
Deployment (Optional): Consider Vercel/Render deployment for frontend/backend

## Key Tasks

1. Build and test API endpoints with Postman or cURL  
2. Create a DeviceContext in React using useEffect and useState  
3. Form Component in Admin Console with effect to backend

## Completion Milestones

- [ ] Backend server connected to MongoDB Atlas  
- [ ] RESTful API working for upload and retrieval  
- [ ] React dashboard displaying live IoT data  
- [ ] Admin console with working form  
- [ ] Error handling and UI validation

## Bonus Challenges (Time-Permitting)

- Add charts using Chart.js or Recharts  
- Filter by deviceId  
- Add timestamps and live status indicators  
- Persist sessions in Admin Console using localStorage

Project Shots:

IOT Dashboard

A screenshot of a device dashboard

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Admin dashboard

A screenshot of a computer

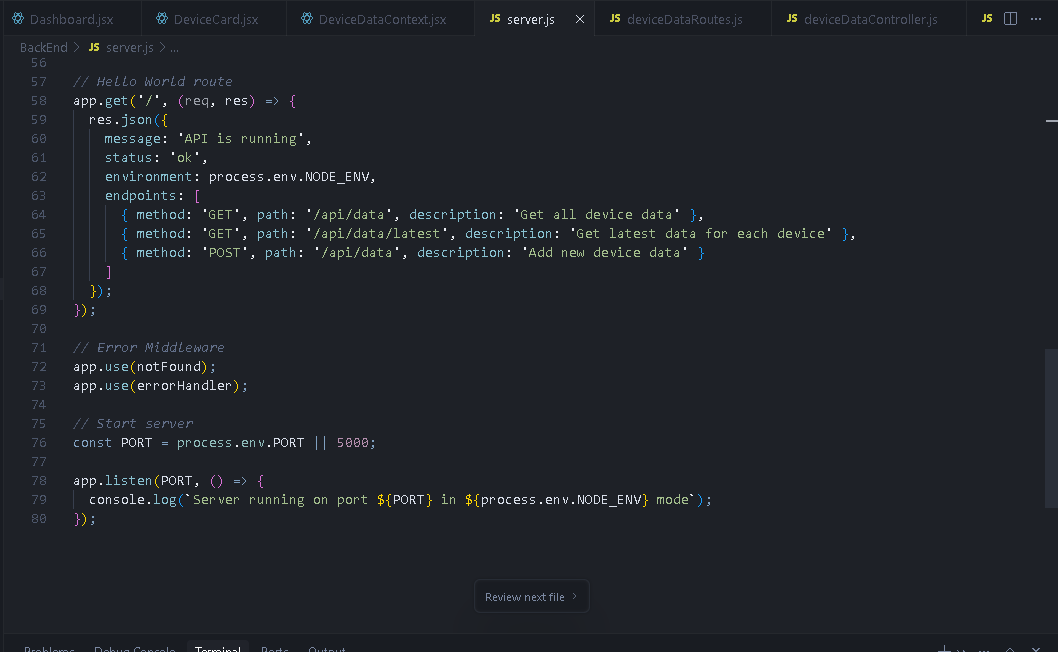
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Code Snippets

Server:



FrontEnd:

A screenshot of a computer

AI-generated content may be incorrect.

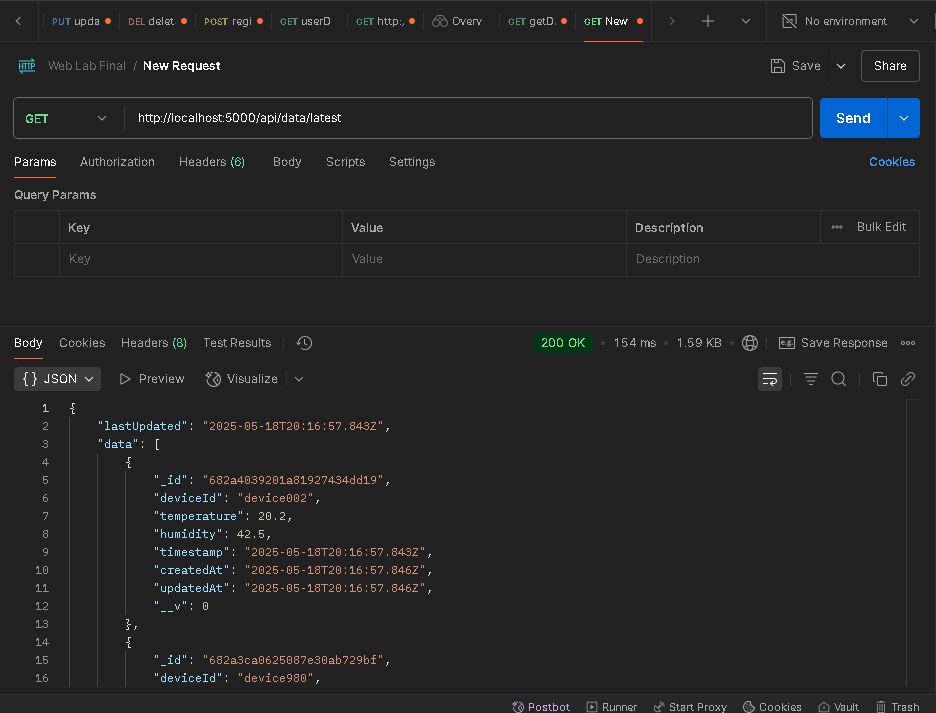
PostMan:

Get Old Data:

A screenshot of a computer

AI-generated content may be incorrect.

Get THE LATEST Data:



Add Latest Data:

**method: 'POST', path: '/api/data', description: 'Add new device data'**

A screenshot of a computer

AI-generated content may be incorrect.

Github Repo Link: