To

Here's the **recommended folder structure** for your **MERN + Flask IoT & AI Analytics Platform** to keep everything organized and scalable.

📂 Folder Structure Overview

```
📂 Project-Root/
├── 📂 frontend/       # React.js (Frontend)
├── 📂 backend/        # Node.js + Express (Backend)
├── 📂 flask-api/      # Flask AI Processing Server
├── 📂 mqtt-broker/    # MQTT Setup (if running custom broker)
└── 📜 README.md       # Project Documentation
```

📂 Frontend (React) Folder Structure

📍 **Path:** `frontend/`

```
📂 frontend/
├── 📂 public/              # Static assets
├── 📂 src/
│   ├── 📂 components/      # Reusable UI components
│   │   ├── Header.jsx
│   │   ├── Sidebar.jsx
│   │   ├── Chart.jsx
│   │   └── DeviceList.jsx
│   ├── 📂 pages/           # Pages (IoT Dashboard, AI Analytics, etc.)
│   │   ├── Home.jsx
│   │   ├── IoTDashboard.jsx
│   │   ├── AIAnalytics.jsx
│   │   └── Login.jsx
│   ├── 📂 context/         # Global state management (React Context/Redux)
│   │   ├── AuthContext.js
│   │   └── DashboardContext.js
│   ├── 📂 services/        # API calls (Axios)
│   │   ├── api.js
│   │   ├── authService.js
│   │   └── deviceService.js
│   ├── 📂 hooks/           # Custom hooks (if needed)
│   ├── 📂 styles/          # Global styles (CSS/SCSS/Tailwind)
│   ├── App.js              # Main App Component
│   └── index.js            # Entry Point
├── .env                    # Environment Variables
└── package.json            # React Dependencies
```

```
├── vite.config.js / webpack   # Build Config
```

✅ **Why?**

- `components/` → UI elements like charts, tables, widgets.
- `pages/` → Separate routes for IoT Dashboard, AI Analytics, Admin Panel.
- `services/` → API handling for better organization.
- `context/` → Global state management for auth, dashboard state.

📂 Backend (Node.js + Express) Folder Structure

📍 **Path:** `backend/`

```
📂 backend/
├── 📂 config/          # Configuration files
│   ├── db.js           # MongoDB Connection
│   ├── mqtt.js         # MQTT Setup
├── 📂 models/          # Database Models (MongoDB Mongoose)
│   ├── User.js
│   ├── Device.js
│   ├── SensorData.js
│   ├── AIResult.js
├── 📂 routes/          # Express Routes (APIs)
│   ├── authRoutes.js
│   ├── deviceRoutes.js
│   ├── aiRoutes.js
│   ├── subscriptionRoutes.js
├── 📂 controllers/     # Business logic
│   ├── authController.js
│   ├── deviceController.js
│   ├── aiController.js
│   ├── paymentController.js
├── 📂 middleware/      # Middleware (JWT, authentication)
│   ├── authMiddleware.js
├── 📂 utils/           # Utility functions (helper functions)
│   ├── logger.js
│   ├── errorHandler.js
├── server.js           # Main Express Server File
├── package.json        # Dependencies
├── .env                # Environment Variables
```

✅ **Why?**

- `models/` → Database schema for **Users, IoT Devices, AI Results**.

- `routes/` → **API endpoints** for authentication, device control, AI analytics.
- `controllers/` → Logic handling (separating business logic from routes).
- `middleware/` → JWT authentication & request validation.
- `utils/` → Logging, error handling, helper functions.

📂 Flask AI Processing (Flask) Folder Structure

📍 **Path:** `flask-api/`

```
📂 flask-api/
├── 📂 models/          # AI models & scripts
│   ├── ai_model.py
├── 📂 routes/          # Flask Routes (APIs)
│   ├── ai_routes.py
├── 📂 utils/           # Helper functions
│   ├── preprocess.py
├── app.py              # Main Flask App
├── requirements.txt    # Python Dependencies
├── .env                # Environment Variables
```

✅ **Why?**

- `models/` → Client's **AI scripts** (e.g., machine learning models).
- `routes/` → Flask API endpoints.
- `utils/` → Preprocessing, helper functions.

📂 MQTT Broker (Optional)

📍 **Path:** `mqtt-broker/`

```
📂 mqtt-broker/
├── broker.js           # MQTT Server Setup (Mosquitto/Node.js)
├── package.json        # Dependencies
├── .env                # Configuration
```

✅ **Why?**

- Only needed if **custom MQTT broker** is required for **real-time IoT**

communication.

How Everything Connects

[React Frontend] → Calls → [Node.js Backend] → Calls → [Flask AI API]
| | |
UI for Users Stores Data in MongoDB Runs AI Script
| | |
Charts, Graphs Communicates with MQTT Returns Predictions

Deployment Notes

📌 **Frontend (React)**

- Deploy on **Vercel/Netlify**.

📌 **Backend (Node.js)**

- Host on **Hostinger VPS/AWS EC2**.
- Use **PM2** for process management.

📌 **Flask AI API**

- Deploy on **Hostinger VPS** (with Gunicorn & Nginx).

📌 **MongoDB**

- Use **MongoDB Atlas** or self-host on **VPS**.

Next Steps

✅ **Set Up Git Repositories** (Separate repos for frontend, backend, Flask).
✅ **Deploy Backend & AI Processing** first, then connect frontend.

This structure makes your app **scalable, modular, and easy to maintain**.
Let me know if you need any tweaks! 🚀

Cc	
Bcc	
Subject	