Create a full “Customize” tab in the frontend, for the coordinator where I can configure parameters for three devices: HLK-LD2450 radar, TSL2561 light sensor, and SK6812B LED strips, and group them in a coordinator card
All parameters must connect correctly to backend API routes, MQTT topics, WebSocket streams, and database storage.

Program the firmware to act accordingly

UI Requirements

- **Add a top-level tab named “Customize” and a card for the coodirnator.**
- **Inside it, add three collapsible sections: “Radar (LD2450)”, “Light (TSL2561)”, “LED Strip (SK6812B)”.**
- **Implement the following controls:**

HLK-LD2450 - make the live monitor update accordingly

- **Detection range: min distance, max distance.**
- **Field of view: left angle, right angle.**
- **Sensitivity slider.**
- **Mode selector: Single Target, Multi Target, Presence.**
- **Static-presence toggle.**
- **Reporting rate (Hz) and fps of the frontend.**

TSL2561

- **Integration time (13.7 / 101 / 402 ms).**
- **Gain (1× / 16×).**
- **Auto-range toggle.**
- **Lux threshold low/high.**
- **Interrupt toggle.**

SK6812B

- **Global brightness.**
- **Color controls (RGB, and should be resetable).**
- **White channel intensity (W only).**
- **Effect selector: Static, Breathing, Rainbow, Chase.**
- **Effect speed.**

Backend / MQTT / WebSocket integration
For each parameter:

- **Create REST endpoints for saving/loading config from the database.**
- **Create MQTT topics for dispatching parameter updates to devices (radar, light, LEDs).**
- **Ensure WebSockets support live sync, so changes in the UI immediately update the frontend if a device reports new values.**
- **Implement a single canonical config model in the database (MongoDB) that stores all customizable settings.**
- **Connect the MQTT publish/subscribe logic so that device → backend → frontend → device all stay consistent.**
- **Ensure the frontend always reflects real sensor state by merging WS and REST data.**

Code output requirements

- **Generate all TypeScript interfaces, JSON schemas, components, backend route handlers, MQTT topic patterns, and WebSocket integration points.**
- **Use clean, production-ready code structure with clear naming conventions.**
- **Implement error handling, default values, and fail-safe behavior if a device is offline or MQTT fails.**

Lastly i want you to remove the calibration tab, since it wornt be necessary.