**Project Overview — Detailed with Technologies**

**Solution Structure**

/IoTAnomalyDetection

│

├/DeviceSimulator # Simulated sensor client (Console App)

├/MLModel # ML.NET model training + inference (Class Library)

├/BackendAPI # ASP.NET Core API

├/Dashboard # Blazor Server

├/Shared # Shared Class Library

**Technologies Used**

| **Area** | **Technology** | **Purpose** |
| --- | --- | --- |
| UI Framework | **Blazor** (likely WebAssembly or Server) | Component-based UI in C# |
| Real-Time Messaging | **SignalR (ASP.NET Core)** | Pushes live sensor updates to dashboard |
| Shared Contracts | **Shared Class Library (.NET)** | SensorData model shared between projects |
| Communication | **SignalR Client Library** (Microsoft.AspNetCore.SignalR.Client) | Real-time client |
| Backend Hosting | ASP.NET Core Web App | Hosts SignalR Hub |
| Styling | Basic CSS / Bootstrap | Layout and UI styling |

**Key Files / Components**

* SensorDashboard.razor: Blazor page that displays sensor data
* SensorData.cs: Shared data contract for signal messages
* SensorHub.cs: SignalR Hub broadcasting messages
* Program.cs: Configures SignalR hub and services
* SensorDataMap: Stores and updates sensor info live

**Features Currently Implemented**

Real-time updates of sensor values  
UI updates on new data using SignalR  
Visual indicator for anomaly detection (IsAnomaly)  
Sorted view of sensors by ID

**Future Enhancements (Optional)**

| **Feature** | **Description** |
| --- | --- |
| Charting | Add dynamic charts using ChartJs.Blazor.Fork or switch to **Syncfusion**, **LiveCharts2**, or **Plotly.NET** for easier integration |
| Sensor Categories | Group sensors by type/location |
| History View | Display time-series or past data with filtering |
| Anomaly Analytics | Add chart threshold zones or ML-based detection |
| Authentication | Secure SignalR endpoints using **JWT** or **Azure AD** |
| Deployment | Host in **Azure App Service**, **Docker**, or **AWS** |
| Notification System | Toasts/email alerts on anomalies |
| Data Storage | Save historical data in **SQLite**, **PostgreSQL**, or **InfluxDB** |
| Dashboard UX | Use **MudBlazor** or **Radzen** for rich UI widgets |
| Offline Support | Enable caching using IndexedDB for Blazor WASM |
| Mobile View | Responsive design for tablets or mobile use |

**Recommendation Moving Forward**

Since the core live-update system works:

* Keep improving the UI (non-chart-based enhancements)
* Consider adding **unit tests** for data processing logic
* Modularize SignalR handling into a service
* Revisit charting later — maybe use a higher-level Blazor chart library