Dashboard Module Design Document

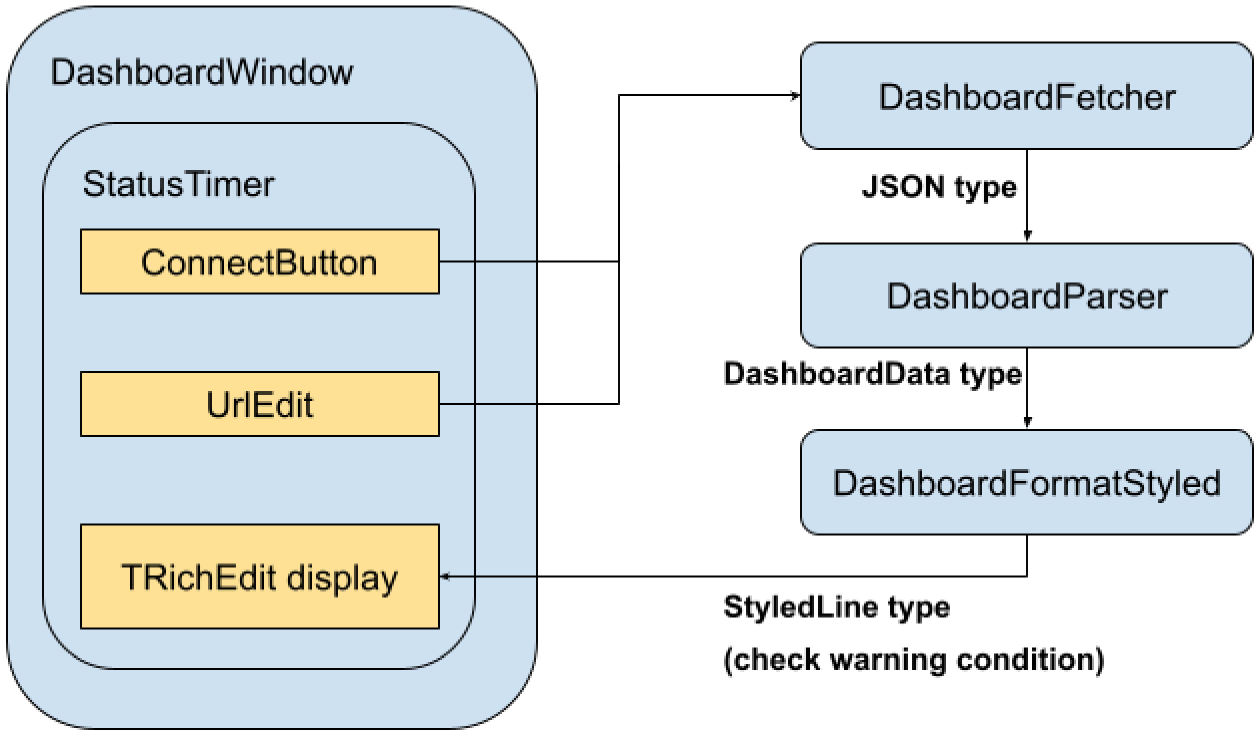
1️⃣ System Overview

The Dashboard module fetches status information provided by Raspberry Pi in real-time and displays it to the user through a GUI.

Data flow:

```
Raspberry Pi Status (HTTP/JSON) →
DashboardFetcher (JSON type) →
DashboardParser (DashboardData type)→
DashboardFormatStyled →
DashboardWindow (TRichEdit display)
```

2️⃣ Architecture Diagram



3️⃣ Key Components

3.1 DashboardData (Data Model)

File: `DashboardData.h`

Field Name	Type	Description
cpu	double	CPU usage (%)
temperature	double	CPU temperature (°C)
diskUsedPercent	double	Disk usage (%)
diskFree	__int64	Disk free space (bytes)
memoryUsedPercent	double	Memory usage (%)
memoryFree	__int64	Memory free space (bytes)

3.2 DashboardFetcher (HTTP Fetcher)

File: `DashboardFetcher.h / .cpp`

Responsibility:

- Perform HTTP GET request to a specified URL
- Return response body (JSON) as a string

Interface:

```
String FetchJson(const String& url);
```

Error Handling: Throws `Exception("HTTP GET failed: ...")` on error.

3.3 DashboardParser (JSON Parser)

File: `DashboardParser.h / .cpp`

Responsibility:

- Convert JSON string → `DashboardData` structure

Interface:

```
DashboardData ParseFromJson(const String& json);
```

Error Handling: Throws `Exception("Invalid JSON")` if JSON is invalid.

3.4 DashboardFormatStyled (Formatter)

File: `DashboardFormatStyled.h / .cpp`

Responsibility:

- Convert `DashboardData` into a vector of human-readable `StyledLine`
    - Each item includes label, value, level
    - level → `"normal"`, `"warning"`, `"critical"`

Interface:

```
std::vector<StyledLine> FormatWithStyle(const DashboardData& data);
```

StyledLine structure:

Field Name	Type	Description
label	String	Display label (e.g. `"CPU usage"`)
value	String	Display value (e.g. `"42.5 %"`)

level	String	Status level (`"normal"` / `"warning"` / `"critical"`)

Level Criteria:

Item	normal	warning	critical
CPU usage	< 60	< 80	>= 80
Temperature	< 60	< 70	>= 70
Disk/Memory	always `"normal"`		

3.5 DashboardWindow (UI Layer)

File: `DashboardWindow.h / .cpp`

Responsibility:

- Provide UI (TRichEdit-based status display)
- Start/stop connection via ConnectButton
- Periodic status updates via Timer
- Asynchronously fetch and update UI (`FetchAndDisplayAsync`)

Key Interfaces:

Method Name	Description
FetchAndDisplayAsync	Asynchronously fetch and display status in TRichEdit
ConnectButtonClick	Toggle Connect/Disconnect
StatusTimerTimer	Periodic status update

Internal State Variables:

Variable Name	Description
isConnected	Connection status

Color Mapping:
`GetColorForLevel(level)` method converts level → color.

4️⃣ Overall Flow

1️⃣ User clicks ConnectButton → `isConnected = true` → Timer activated
2️⃣ Timer triggers → `FetchAndDisplayAsync()` executes
3️⃣ `FetchAndDisplayAsync`:

- DashboardFetcher → `FetchJson(URL)`
- DashboardParser → `ParseFromJson(JSON)`
- DashboardFormatStyled → `FormatWithStyle(DashboardData)`
- UI update: `StyledLine` displayed in `TRichEdit` (with color)

4️⃣ On Disconnect → Timer deactivated and UI cleared.
