# Sunward Portal Management System

**DOCUMENT INFORMATION**

| | |
|---|---|
| Prepared by | Appcentric Technologies |
| Client | Sunward |
| Project Name | Sunward Portal Management System |
| Prepared by | Arun |
| Module Name | Dynamic Dashboard (DevExpress BI Dashboard — Blazor) |
| Pages | 8 |

## Disclaimer and Confidentiality Statement

This document is a technical reference prepared by **Appcentric Technologies** for the **Sunward Portal Management System** project. It provides detailed information on the system architecture, technical approach, implementation methodology, and related technical specifications.

This document is **confidential** and is intended solely for the internal use of the authorized technical team of **Sunward**.

No part of this document may be copied, modified, distributed, or disclosed to any third party without prior written approval from **Appcentric Technologies**.

This is a technical reference for the Dynamic Dashboard module implementation. It provides architecture, UI mapping, entity/data model, business rules, integration details, and deployment guidance for developers. This document is confidential and intended for the Sunward technical team only.

# Table of Contents

## 1. Project Overview

The Dynamic Dashboard module lets authorized users create, edit, save, share and view BI dashboards at runtime using the DevExpress Dashboard component for Blazor (Viewer + Designer modes). It supports reusable dashboard templates, per-user dashboards, shared dashboards, widget libraries, and data source management

This module follows the Email module's structure (topic/conversation style for the Email doc) but focused on dashboards. Use the Email doc's document style and approval flow as the template.

## 2. Objectives

- Provide a run-time dashboard viewer for end users (filters, drill-down, export). DevExpress Demos
- Provide a secure dashboard designer for authorized users to create/modify dashboards (drag & drop charts, grids, maps, filters). DevExpress Documentation
- Persist dashboard definitions, layout and per-user library in DB so dashboards are editable and versioned.
- Allow data-source configuration (SQL Server, stored procedures, views), parameterization, and credentials storage.
- Support dashboard sharing, favorites, and role-based visibility.
- Support export (PDF/Excel/Image) and scheduled snapshots if required

## 3. System Modules & Key Functions

| Module | Description | Key Users |
|---|---|---|
| **Dashboard Viewer** | Displays dashboards; supports parameter filters, interactivity (cross-filtering, drill-down), export. (User role: All users). | All Users |
| **Dashboard Designer** | editing canvas for creating widgets, data binding, layout; save/publish version. (User role: Designer/Admin). | Designer/Admin |

## 4. Functional Components

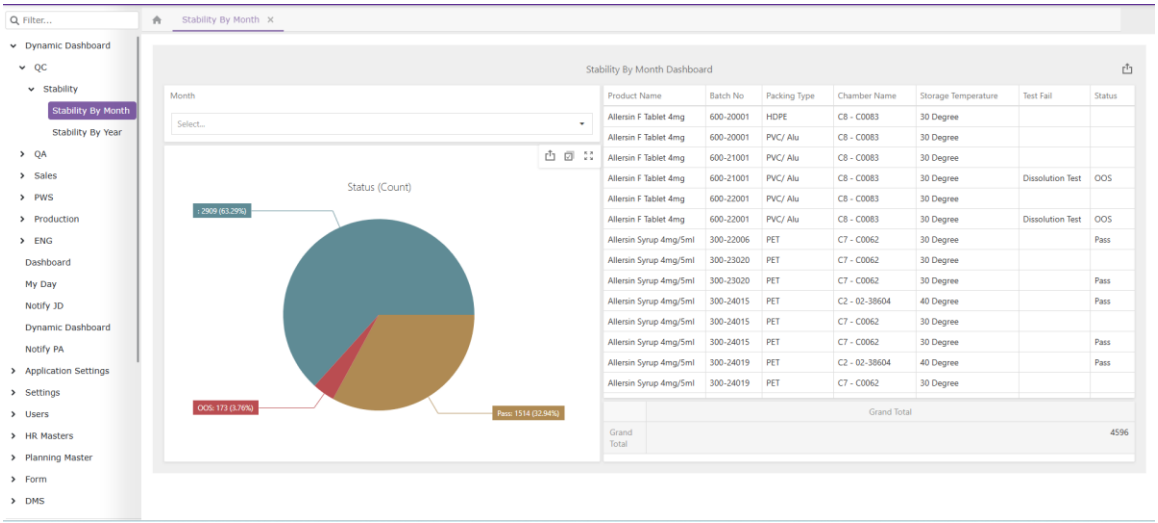
1. **Dashboard Viewer** — displays dashboards; supports parameter filters, interactivity (cross-filtering, drill-down), export. (User role: All users) DevExpress Demos
2. **Dashboard Designer** — editing canvas for creating widgets, data binding, layout; save/publish version. (User role: Designer/Admin) DevExpress Documentation
3. **Dashboard Library / Catalog** — left pane listing available dashboards (favorites, shared, my dashboards).
4. **Data Source Manager** — manage connections, queries, credentials, parameter schemas.
5. **Widget/Component Palette** — reusable visualizations (chart, pie, gauge, pivot, grid, map, KPI, filter, image).
6. **Dashboard Versioning & Audit** — store version history and audit of changes.
7. **Scheduler (Optional)** scheduled data refresh or snapshot exports.

### 4.1 UI Screens & Component Mapping

(I used your uploaded screenshots as UI style references — filenames below: they show Viewer + Designer modes and left-side library listing.)

**Screenshots available in project:**

1. Viewer mode (dashboard + data grid).
2. Designer mode (canvas + palette).
3. Designer with palette components.

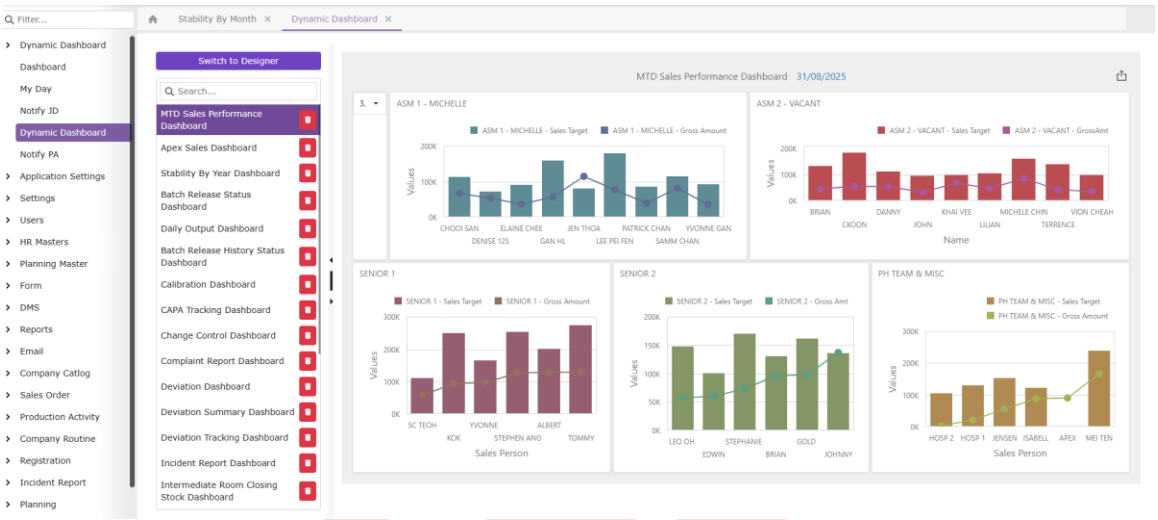


## 4.2 Main UI Pages

1. **Dashboard List (left panel)** search, favorites, create new, delete (trash icon), and "Switch to Designer/Viewer" top button (illustrated in screenshots).

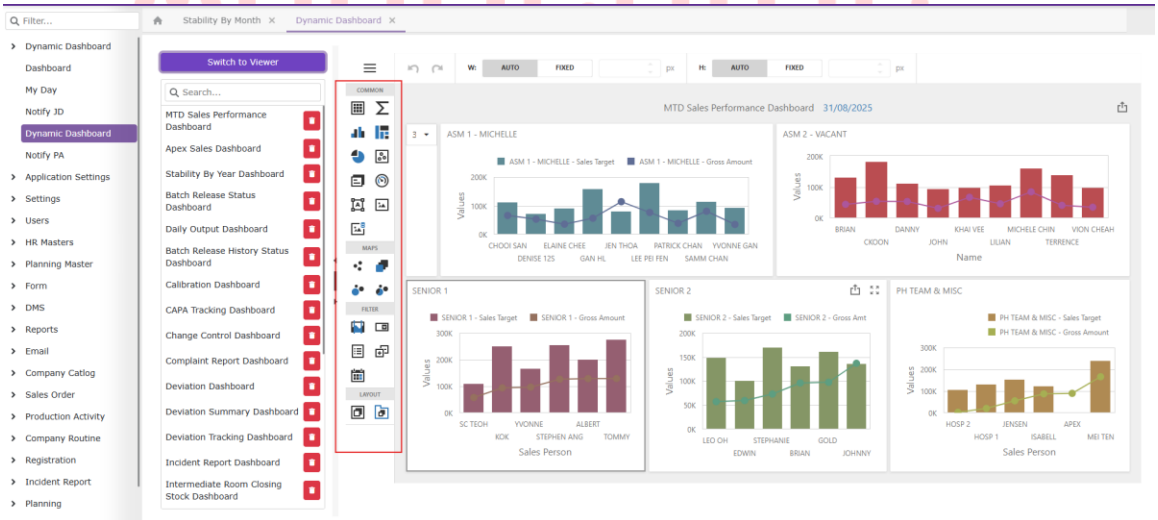
2. **Viewer Canvas (center/right)** header with dashboard title, date, export icon; canvas shows tiles/containers with charts & grids.

3. **Designer Sidebar (center)** widget palette (icons), layout/size controls, save/publish, undo/redo, width/height controls.




## 4.3 Component mapping to DevExpress control

Charts, grids, pivot — DevExpress dashboard items. Use the DevExpress Dashboard component as viewer/designer in Blazor.



## 5. Data Model

Suggested entities to store dashboards, components, and data-source metadata. (Relational model; adapt to existing Sunward DB conventions.

1. UserID, DashboardID, Filters, LayoutOverrides
2. DevExpress dashboard definitions are typically serialized — store in Serialized Definition (XML or JSON) so Designer/Viewer can load/save them. DevExpress docs explain the dashboard storage model and how to load/save definitions server-side

## 6. Business Logic

### 6.1 Create New Dashboard (Designer)

- User clicks **Switch to Designer** (left UI). System checks DashboardPermissions for Designer role.
- Designer opens blank canvas or clones an existing dashboard version. Designer uses palette widgets (charts, grids, etc.) to add items.
- For each widget, designer binds to a DashboardDataSource or creates an ad-hoc query.
- Save → creates new records in DashboardVersions and updates Dashboards. Store serialized definition.

### 6.2 View Dashboard (Viewer)

- Viewer selects dashboard from library; application loads `Dashboards.CurrentSessionID` → get `SerializedDefinition`.
- Dashboard loads into DevExpress DashboardViewer component; parameters are resolved either from URL query or user settings.
- Interactions (filters/explore) happen client-side; some data retrieval occurs via server, using configured `DashboardDataSource`.

## 7. Integration with DevExpress Dashboard (Blazor)

**Key integration points**

- Use DevExpress Dashboard component for Blazor: DashboardViewer in viewer pages and DashboardDesigner for the editor. Official examples & API are here. DevExpress Documentation+1
- Load and save dashboard definitions from DB: the DevExpress component supports loading dashboard layouts via serialized XML/JSON; implement server endpoints to fetch/save definitions.

- Data source binding: either use server-side data sources (recommended) or configure dashboard to query via REST endpoints that return JSON tables.

Example (conceptual) — loading a dashboard in Blazor:

```
// server-side: Get serialized dashboard definition

var definition = await _dashboardRepo.GetSerializedDefinitionAsync(dashboardId);

// return definition to client/viewer which uses DevExpress DashboardViewer to load it

// Blazor page (pseudo)

<DashboardViewer Dashboard="@MyDashboard" />

@code {

    Dashboard MyDashboard { get; set; }

    protected override async Task OnInitializedAsync() {

        var xml = await Http.GetStringAsync($"api/dashboard/{dashboardId}/definition");

        MyDashboard = Dashboard.FromXml(xml); // pseudo-method: use DevExpress API

    }

}
```

## 8. Summary

The Dynamic Dashboard module provides a fully interactive business intelligence platform within the Sunward Portal, enabling users to **view, create, customize, and manage dashboards** in real time. Built using **DevExpress BI Dashboard for Blazor**, the module offers powerful analytical tools, dynamic visualization components, and flexible data connectivity options.

Through the **Dashboard Viewer**, users can analyze KPIs, trends, and performance metrics using charts, grids, pivot tables, and visual filters. The **Dashboard Designer** allows authorized users to build dashboards using drag-and-drop components, bind data sources, configure layout properties, and publish dashboards for departmental or organization-wide use.

The system includes a **dashboard library**, version control, shared dashboards, user-specific favorites, and robust permission management to ensure secure and streamlined access. The

integration with SQL-based data sources enables real-time data visualization while maintaining high performance and optimized load times.

This module centralizes analytical reporting, eliminates manual reporting processes, and empowers teams to make faster and more informed decisions. With its flexible architecture and strong DevExpress integration, the Dynamic Dashboard module is scalable, maintainable, and ready for future enhancements such as scheduled reports and automated snapshot exporting.