**Technical solution summary**

**A screenshot of a computer

AI-generated content may be incorrect.**

Kindly please note the solution structure is very much similar to the implementation that I have applied in MOD project – that uses Telerik Kendo Widgets (Grid) to render data in static and dynamic data dashboards.

In this SOA approach the solution is broken into the following components.

**SalesDataDashboard (MVC Web Application)**

Two implementations has been included in the solution. First one is to render the data in Telerik Kendo Grid (this wasn’t working in my local machine due to licence issue). I have also implemented rendering the data using Jquery Data tables to test the end-to-end solution

A screen shot of a computer program

AI-generated content may be incorrect.

A white background with many small squares

AI-generated content may be incorrect.

**SalesDataAPI (ASP.NET Web API)**

API consumed by website to fetch sales data. Website is included in the allowed CORS origin list in the web.config.

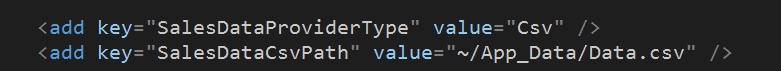


Unity has been used for DI – Services and Providers are injected at runtime.

**Service layer** implementation takes into account data handling from CSV data source and from the other potential data sources in future by ISalesDataProvider abstraction.

A computer screen shot of a computer code

AI-generated content may be incorrect.



A screen shot of a computer program

AI-generated content may be incorrect.

Applied **HttpUtility.HtmlEncode** in the **CsvSalesDataProvider** implementationwhilst parsing the data before returning to the UI to prevent script injection vulnerabilities.

Service methods use async I/O whilst reading the large CSV files.

**Unit testing** has been included that uses MOQ for service class implementation.

**Further improvements that can be made to the current solution**

* We can consider caching the data in memory e.g. using memorycache to avoid reading and parsing the file on every API request. Cache can expire time bound or when the

underlying file has changed (using FileChangeMonitor)

* The current implementation returns the statically typed viewmodel **SaleDataRowModel**. Instead, we can return dynamic view model to render in widgets like Telerik Kendo Grid, I have built this framework in the MOD systems that provides the dynamic reporting capabilities for the end users – users can define the report template with the required columns/attributes and generate these reports on the fly.