Contents

[Program Start Up 2](#_Toc34920478)

[Run application 2](#_Toc34920479)

[Connect to database 2](#_Toc34920480)

[Testing 2](#_Toc34920481)

[Program Requirements 2](#_Toc34920482)

[Program Architecture 3](#_Toc34920483)

[API 3](#_Toc34920484)

[Controller 3](#_Toc34920485)

[Service and data layer 3](#_Toc34920486)

[Model 3](#_Toc34920487)

[Client 4](#_Toc34920488)

[Components 4](#_Toc34920489)

[Services: 4](#_Toc34920490)

[Models 4](#_Toc34920491)

[Database 4](#_Toc34920492)

# Program Start Up

Run application

Debug the project

Connect to database

To point the application to your database update the \*\*\*\*\*\*.txt with your connection string.

The file to create the data base is also included.

Testing

Unit test were developed to test the logic methods of the API. You can find these is the \*\*\*\*.Test project.

# Program Requirements

|  |  |
| --- | --- |
| Create price entry | A user should be able to create a PriceItem. A PriceItem has:   * price (decimal with precision two) * timestamp(date and time).   To do this the user enters three fields a price, a date, and a time. For user entry:   * Valid price input is a decimal with precision two, * Valid date input is any date. * Valid time input is an hour and either zero or thirty minutes. |
| Display table of all price entries | All PriceItems items should be displayed to the user in a table with columns for price and timestamp. |
| Display graph of all price entries | All PriceItems should be charted on a graph with the timestamp as the \*\*\*\* and the price as the \*\*\* axis. |
| Display minimum price | Display the price value for the PriceItem that has the minimum price. |
| Display maximum price | Display the price value for the PriceItem that has the maximum price. |
| Display average price | Display the average value of price for all PriceItems. |
| Display most expensive 60 minute period | Display the start timestamp and the end timestamp for the sixty minute period that collectively has the highest value for PriceItems entered than any other sixty minute period. For clarification if this period started at 12:00 it would include entries for 12:30 and for 13:00. |

# Program Architecture

The application is composed of a REST API built using .NET Core that connects to an on premise SQL Server Express database and a web client built with Angular. It uses MVC architecture.

## API

The API is composed of:

Controller

PriceController has three endpoints. Create a PriceItem, Get all PriceItems, and Get PriceItem calculations. It uses Web MVC to define endpoints. Below each of the endpoints are defined.

|  |  |
| --- | --- |
| URL | /priceitems |
| HTTP method | GET |
| Request body content | None |
| Response body content | List<PriceItem> |
| Function | Returns all PriceItems in the system |

|  |  |
| --- | --- |
| URL | /priceitems |
| HTTP method | POST |
| Request body content | PriceItem |
| Response body content | None |
| Function | Creates a new PriceItem in the system |

|  |  |
| --- | --- |
| URL | /priceitems/calculations |
| HTTP method | GET |
| Request body content | None |
| Response body content | PriceItemCalcuation |
| Function | Returns a PriceItemCalculation that contains the minimum, maximum, average price for all PriceItems and the most expensive hour. |

### Service and data layer

The data layer connects to a SQL Server DB, and the service layer provides create and read functionality for the database.

### Model

This contains the object models.

PriceItem

This item is instantiated to create PriceItems and is mapped to the database using Entity Framework. It has the fields Price, Timestamp, and Id.

PriceCalculation

This item is instantiated to create PriceCalculations. It is used to send API response information about the min, max, average, and most expensive hour.

## Client

The Client in built with Angular. Its basic architecture is:

Components: \*\*\*\*\*\* component is responsible for displaying the PriceItem information to eh user and accepting input for a new PriceItem, it uses angular data binding to do this. I It is styled with bootstrap 4.

Services: \*\*\*\*\* service is responsible for making requests and receiving responses to the API.

Models:

## Database

There is a SQL Server database. This contains one table – PriceItems. This table has three columns price, timestamp, and id.