**Set-Up**

1. Import the “Dishes” collection by creating a Database “Restaurant” in the local MongoDb cluster. “Dishes.Json” file is available in “Data Back up” folder.

[Open MongoDb Compass tool -> Create a Database:”Restaurant” ->Add Collection: Dishes -> Import Data: Browse->Dishes.Json]

2. Run the Solution in the IDE (Ex. Visual studio)

3. Use Fiddler/Postman/any Web Browser to test the "Get" method.

4. URL: https://localhost:44369/api/v1/Dishes/Get

5. Result: The list of dishes that is already available on the MongoDb server.

**About the Project**

**Functional Description:**

The Web-API solution serves single purpose of a “Get”. This gets all the available dishes that are saved in the MongoDB server. There are no CRUD operations except the above read operation to reduce further complexity of adding validations and exception handling.

**Technical details:**

The solution takes the “Clean Architecture” way of breaking down the layers of work. There are 3 independent layers which can be tested individually by injecting the dependency. API layer for external calls; AppCore layer which mainly consists processing logic using the models; and finally infrastructure layer which is only for Data access queries. Dependency injection is used wherever needed and NInject tool is used for this purpose. As there is no “Post” call involved, Validation filters and Exception filters is not used here. Unit tests or integration tests have not been added as it is only for “Get” method without any logical processing in the AppCore layer.

**Major problems:**

1. MongoDB driver causing problems while trying to access the DB had taken some time (around 1 or 2 hrs) as there were no helpful resources on this topic.[Error: “could not load file or assembly 'mongodb.driver, version=2.11.5.0, culture=neutral, publickeytoken=null' or one of its dependencies. the system cannot find the file specified”] Upon on some suggestions on few websites, I tried degrading Bson nugget package and again updating all the Mongodb dependencies it started working.
2. Dependency Injection issues due to improper configuration of Ninject package took 1 hour of the time. Along with this, other miscellaneous issues in trying to access other layers using DI. But these issues are not new; it was encountered previously in a different project.

**Timeline: Building the App:**

1. Going through basic MongoDB tutorial to learn about different ways of inserting data into MongoDb-Collections took some time. After learning about inserting the data, a Database was created and a collection was created inside it. An object was inserted into the collection which created the schema on the fly. This method was preferred as this was a good feature which had to be taken advantage of. In total, this step took me around 1 hour( 20mins extra may be)
2. Creating a Web-API with a regular N-Tier architecture would have taken not more than 1 hour to get this done. But to create a testable, scalable Onion layered architecture took another hour due to the some issues in between.
3. As described above the Infrastructure layer connecting to MongoDb took some time due to driver issues but eventually it started getting the data after 2 hours. After being able to connect to MongoDb from C#, there were two ways to proceed: Either fetch the results as list of Bson documents (which can be mapped later in AppCore layer) or fetch result as parsed/typed entity. The second method was chosen as there was no calculation/processing involved here.
4. AppCore layer had nothing much as there was not much logic involved in between. (like filtering or mapping or conditional processing etc). This layer took few minutes finish.
5. API had issues with dependency injection as described above but it started working in an hour.
6. In total, API was running with proper response. The total time taken to get this task done was around 5-6 hrs.

**Benefits & Disadvantages of using MongoDB (in this project):**

**Benefits:**

Json Based objects : The best combination for RestAPI models.

Creating collections on the fly to be used for later: This greatly reduces any runtime errors and any extra coding overhead to create tables and relations just for basic things.

Instead of using an ORM (ex. Entity framework which is heavy), collection automatically maps into local entity which is really great.

**Disadvantages:**

No extensive support for Joins ($lookup is an alternative but it’s not for the exact purpose).

SQL has a rigid structure with normalizing rules to be followed and hence, clarity of thought prevails for ORM users.

**Digital Menu Advantages:**

1. Modern World with restaurants moving to “Digital Signages” or “Digital Menuboards”.
2. Uses cases: Once an application is created, this app can be used for a wide variety of uses.
   1. Customer can use Mobile App to access the menu, place the order before reaching restaurant
   2. Customer comes to the restaurant and accesses the digital menu API through Kiosks/touchscreens at the restaurants (McD, KFC, Burger king are using this already)
   3. The waiter can share the QR code for the website to access the digital menu which customer can access on their phones and can translate or learn more about the ingredients.
   4. The waiters can use the digital menu in their “Digital Signage” to order the dish from customer table. The orders automatically queue up at the chef’s table or manager’s PC.
   5. The digital menu API can be used by online food delivery apps to get the information required to show the customers.
   6. The problem of 2 or more waiters (or customers) ordering the same last available dish is solved because of the real time update of availability on the digital menu.
3. A customer is more likely to visit a restaurant if the menu is publicly accessible and then customer can comfortably decide and plan on the budget and visit. A restaurant with rich colours and lights can sometimes drive away the customers who might assume the restaurant to be over-expensive restaurant.
4. A Chinese cuisine restaurant may have pizzas available and an Italian restaurant may have some noodles available, but in both these cases the digital menu availability decides on the additional customer inflow.