# **Journal: Sadguru’s Amrit Tulya - Tea Shop**

I had read the assignment on Tuesday, June 30, 2020 in the evening after completing my work.

Initially, I was thinking on what are the requirements that need to be included in the project, apart from the overall requirements given in the question. Then, I was thinking about the technologies on which should I implement this application. So, I spent a day or two on gathering those requirements and thinking about the tech stack that I should use for this application.

So, I have developed this application using **Angular, .Net Core, SQL Server and Ef Core (ORM)**.

After that, I started working on the assignment after finishing the work in the office. For this assignment, I mostly worked on weekends extensively due to company off.

Below, I am listing out the tasks which I did for implementation of this web application. I have divided these tasks as per the components required. But I have mostly worked on minimum 2-3 tasks simultaneously, as I was thinking of completing them as FEATURES.

## Tasks:

### Backend Functionality

Backend functionality for this app includes adding items to the inventory, then viewing the details of the item selected/added and deleting specific item.

There were bonus points for uploading image along with the item details. I thought of implementing that too because I had never that before and thought it would be a great learning experience for me.

In API, I created 2 controllers, one for the CRUD operations, and other for Uploading the image to the server. In this I have implemented **Dependency Injection (DI)**, thanks to the .Net Core, in built container. With the help of **Ef Core**, it was super easy to create DB Connections and performing database related operations. Apart from that, I kept all methods async to achieve efficiency and it was part of the requirements. I have added **migrations** for the database that I used for storing the data. Also, I have used **Swagger for API documentation**. Because of it, it was simple to test the API on its own without using any external software. Lastly, to make our API more convenient to use for other components, I have added **CORS** for smooth connections with the API.

### Front-end Functionality

In front end, there were mainly 4 functionalities, i.e., to add an item, show list of items present in the Inventory, delete item from the list and view the details of the selected item.

To implement these, I have used Angular Framework, Bootstrap themes, etc. I made use of routing to route for the details page. I have created a component each for addition, list and details of the item. For upload, I had to use one more component. Apart from all of these, I have added another component for page not found, in case there is some error occurred.

### Front-end presentational aspects

For styling the front-end, I made use of Font-Awesome Icons, and Bootstrap themes. They came very handy for adding some styles to the UI. Also, I have used images for background, to add items and when you don’t add image, it will display no image found in the list. Adding to that, changed the website icon to Teacup, as it symbolizes instantly what kind of website user is going to visit.

### Backend- Frontend Integration

Backend-Frontend Integration gave me pain because, even after adding the CORS, it was not running properly. It took me a day to resolve the issue and look closely to each aspect of front end and its connection with backend. I have also tried connecting to API using HttpInterceptor, but that did not help. Later, I did some research on the topic and found the solution.

### Unit Test Coverage

Simultaneously, I have started adding unit test cases for the each of methods using **xUnit** and **FluentAssertions**. Even though, I had never introduced to writing unit test cases before, I was able to get hold of it through some online tutorials and implementing them along the way. Because of these test cases, I was able to test as well as verify accuracy of the code.