|  | **Cognizant Academy**  **truYum**  **FSE Web API on Azure Specification Document**  **Version 1.0** |
| --- | --- |
| |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | --- | --- | --- | --- | | **Name** | Seshadri M R | Seshadri M R | Seshadri M R | | **Role** | Learning Solution Designer | Learning Solution Designer | Learning Solution Designer | | **Signature** |  |  |  | | **Date** |  |  |  | |
|  |

**Table of Contents**

[**1.0**](#_1fob9te) **Introduction 3**

[**1.1**](#_3znysh7) **Purpose of this document 3**

[**1.2**](#_2et92p0) **Project Overview 3**

[**1.3**](#_tyjcwt) **Scope 3**

[**1.4**](#_3dy6vkm) **Hardware and Software Requirement 3**

[**2.0**](#_1t3h5sf) **Reference learning and requirement 3**

[**3.0**](#_4d34og8) **Menu item listing and adding to cart 4**

[**3.1**](#_2s8eyo1) **MenuItemListing WebApi 4**

[**3.2**](#_17dp8vu) **OrderItem WebApi 5**

[**4.0**](#_3rdcrjn) **Cloud deployment and consumption in Console application 5**

[**5.0**](#_26in1rg) **Change Log 6**

# Introduction

## Purpose of this document

The purpose of this document is to define the service implementation for truYum project, deployment on Azure and its consumption in a Console application.

## Project Overview

Yum is a leading restaurant chain that operates 50+ restaurants across the country. Due to successful business model and solid understanding about consumer needs, the Yum executive management feels that they can increase their business by going online and delivery food to the door steps of the customer. They are planning to open an online portal called truYum through which they want to reach out a larger customer base.

## Scope

Develop truYum application web services using Web API on .Net core and deploy that to Azure

## Hardware and Software Requirement

1. Hardware Requirement:
   1. Developer Desktop PC with 8GB RAM
2. Software Requirement
   1. Visual studio 2017
   2. Chrome Browser
   3. Postman tool

# Reference learning and requirement

Please go thru all of these k-point videos for simple WebApi Azure deployment. This has detailed explanation of every component to be developed on Azure and WebApi communication using Kubernetes.

[AzureWithCICD-1](https://cognizant.kpoint.com/app/video/gcc-19532393-d4e0-4fd9-8a0c-80ecbdb349d3)

[AzureWithCICD-2](https://cognizant.kpoint.com/app/video/gcc-6633a958-ab72-4c69-b926-fe832e4b56a1)

[AzureWithCICD-3](https://cognizant.kpoint.com/app/video/gcc-553eb186-c1cf-448e-96fc-a96fe37b2e6a)

[AzureWithCICD-4](https://cognizant.kpoint.com/app/video/gcc-fad7d4af-d651-4501-99c6-2785190670c2)



Attached is the truYum case study requirement for WebApi. This case study will be developed based on that.

This case study will **NOT** have database connectivity.

# Menu item listing and adding to cart

The requirement is to create 2 Web Apis **MenuItemListing** and **OrderItem**.

* MenuItemListing is to get the hard coded menu items
* OrderItem is to view menu items and choose the items to add to cart

## MenuItemListing WebApi

Create a WebApi MenuItemListing with a controller MenuItemController. Create New class ‘MenuItem’ as business entity

1. Create the class for MenuItem business entity with the appropriate fields as listed below.

| Class Name: MenuItem | |  |  |
| --- | --- | --- | --- |
| Field Name | Data annotation for field | | Field datatype |
| Id | Key | | Int |
| Name | Required | | string |
| freeDelivery | Display(Name=”Free Delivery”) | | Boolean |
| Price | Required | | Double |
| dateOfLaunch | [DataType(DataType.Date)]  [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)] | | Datetime |
| Active |  | | Boolean |

1. In the GET action method of the MenuItemController, create a list of MenuItem with hard coded data.
2. Create GET action method with Id as input parameter to fetch the menuitem name based on that Id

## OrderItem WebApi

Create a WebApi OrderItem with a controller OrderController. Create New class ‘Cart’ as business entity.

1. Create the class for Cart business entity with the appropriate fields as listed below.

| Class Name: Cart | |  |  |
| --- | --- | --- | --- |
| Field Name | Data annotation for field | | Field datatype |
| Id | Key | | Int |
| userId |  | | int |
| menuItemId |  | | int |
| menuItemName |  | | string |

1. In the POST action method of the OrderController, take the menuItemId as input parameter, hardcode the Id and userId as 1. Create an instance of Cart class and set the properties.
2. Hit the GET action method of the MenuItemListing WebApi that takes menuItemId as input parameter. On getting the menuItem name from the WebApi, set that name in the Cart instance. The POST action method should return the Cart instance.

# Cloud deployment and consumption in Console application

Follow the steps listed below

* Go thru the learning kpoint videos as listed in Section 2.0. Deploy MenuItemListing and OrderItem WebApi on Azure
  + Dockerize the application
  + Create Azure Kubernetes Cluster to deploy both the Web Api to convert to Microservice
* Use ‘Authorize’ keyword in the controllers of both the WebApi
* Create a Console application to show the Menu items and Order menu item
* Use the JWT generation method to create a token, in the console application. This should be used while invoking the WebApi on cloud. The token expiration time is set at 10 minutes. It can be changed as per the need.
* Invoke the MenuItemListing WebApi on Kubernetes to display the menu items
* Choose the menuitem from the list and hit the OrderItem WebApi to order, view the cart item.
  + If an incorrect menu item id is chosen, throw a validation message “Incorrect menu item id chosen. Please choose the appropriate Id”

# JWT generation code

private string GenerateJSONWebToken()

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

expires: DateTime.Now.AddMinutes(10),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

# Change Log

|  | **Changes Made** | | | |
| --- | --- | --- | --- | --- |
| V1.0.0 | Initial baseline created on <dd-Mon-yy> by <Name of Author> | | | |
| Vx.y.z | <Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed> | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |