Assessment Task:

Note: As a Backend Developer when you are developing the APIs, so APIs should not be openly available for use. So for this, I have secured my APIs by Implementing JWT Token Approach.

**Execute this Script**: databasescript.sql (Provided in Repo)

**Things not done: Write unit testing (Not Done).**

**Task: 1:**

Create customers. System shall not allow to add new customer with already existing customer email.

Status: Implemented

**API Name**: /api/Auth/registerCustomer – API to create/register Customer.

Input: **(Both inputs are Required)**

**{**

**"username": "string",**

**"password": "string"**

**}**

**API Name**: /api/Auth/Customerlogin – API for Customer Login and Get the JWT Token.

Input: **(Both inputs are Required)**

**{**

**"username": "string",**

**"password": "string"**

**}**

**API Name**: /api/Auth/refresh-token – API to Refresh the Token before the token expires, the responsibility to refresh the token by calling this api is of frontend developer. (**This API will Require the JWT Token as Authorization**)

In Header

authorization: bearer eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjEzNWUxYWY3LTg4NWYtNGQyMC05ZjQ4LWI2NmE3YjhiZjFmNyIsImh0dHA6Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL25hbWUiOiJzdHJpbmciLCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJDdXN0b21lciIsImV4cCI6MTY2NDIxNjMyNH0.Kph35mKhkL\_X7rJIVzc0fYHm5OkagfiBdXlrq5HGeOgW-T2jeaWjQPgpIZXck1hYJiugtIVckCJ3icn8SgzPrg

Input: **(Both inputs are Required)**

**{**

**"username": "string",**

**"password": "string"**

**}**

**Task 2:**

Create orders. Order shall have order information and items.

Status: Implemented

**API Name**: /api/Orders/AddOrders – API to Create Orders.

In Header

authorization: bearer eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjEzNWUxYWY3LTg4NWYtNGQyMC05ZjQ4LWI2NmE3YjhiZjFmNyIsImh0dHA6Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL25hbWUiOiJzdHJpbmciLCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJDdXN0b21lciIsImV4cCI6MTY2NDIxNjMyNH0.Kph35mKhkL\_X7rJIVzc0fYHm5OkagfiBdXlrq5HGeOgW-T2jeaWjQPgpIZXck1hYJiugtIVckCJ3icn8SgzPrg

Input: **(Both inputs are Required) and Input will contain the List of Items.**

[

{

"itemName": "Rice",

"qty": 2

},

{

"itemName": "Noodles",

"qty": 2

}

]

**Task 3:**

Retrieve list of orders based on specific page index and page size.

In Header

authorization: bearer eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjEzNWUxYWY3LTg4NWYtNGQyMC05ZjQ4LWI2NmE3YjhiZjFmNyIsImh0dHA6Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL25hbWUiOiJzdHJpbmciLCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJDdXN0b21lciIsImV4cCI6MTY2NDIxNjMyNH0.Kph35mKhkL\_X7rJIVzc0fYHm5OkagfiBdXlrq5HGeOgW-T2jeaWjQPgpIZXck1hYJiugtIVckCJ3icn8SgzPrg

**API Name**: /api/Orders/GetOrdersofLoggedInUser?pageSize=10&index=0

Output: will provide the list of orders of particular logged in user (from header jwt authorization) according to pagesize and index the records will be returned.

**}**