**TASK-7: IMPLEMENTING MESSAGES USING SERVICE BUS**

**Goal**:

* Create Service Bus Queue.
* Flow the messages from **1 Microservice** to **Another**.

1. Clone the Frontend Project from <https://siddharthdwivedi318@dev.azure.com/siddharthdwivedi318/Experiential%20Learning/_git/Quick-Cart-FrontEnd>

2. Execute **Orders** **Table**

create table orders

(

orderid int identity,

custEmail varchar(50) references Customers(emailID),

prodID int,

prodCost int,

orderdate dateTime default getDate()

)

and Execute your own **Procedure** in the **Database** with this Format

alter proc usp\_AddOrder

(@custEmail varchar(50),

@prodId int,

@prodCost int,

@orderdate datetime

)

as

begin

    begin try

        insert into orders values (@custEmail,@prodId,@prodCost,@orderdate)

        return 1

    end try

    begin catch

        return 0

    end catch

end

Make sure in the customer’s table your **emailID** is present.

3. Create **Payment** **Microservice** Using Azure Function which would Process the Payment and then send the Payment Details to the Service Bus Queue.

Write the code yourself!

4. Create **Order** **Microservice** using Azure Function which would get triggered whenever there is a new message in the Queue. This service would insert a new row in the Order table.

**Reference Code:**

[FunctionName("OrderService")]

        public void Run([ServiceBusTrigger("paymnt-queue", Connection = "conString")]string myQueueItem, ILogger log)

        {

            log.LogInformation("Order MS Ran");

            dynamic data = JsonConvert.DeserializeObject(myQueueItem);

            string prodID = data.ProdID;

            string prodCost = data.ProdCost;

            string custEmail = "siddharthd@cloudthat.com";

            int result = 0;

            //SQL connection

            SqlConnection conObj = new SqlConnection("Data Source=quickcart-server.database.windows.net;Initial Catalog=QuickCart-DB;user id=demouser; password=Siddharth@1234");

            //command

            SqlCommand cmdObj = new SqlCommand("usp\_AddOrder", conObj);

            cmdObj.CommandType = CommandType.StoredProcedure;

            try

            {

                SqlParameter emailPrm = new SqlParameter();

                emailPrm.ParameterName = "@custEmail";

                emailPrm.Value = custEmail;

                emailPrm.Direction = ParameterDirection.Input;

                emailPrm.SqlDbType = SqlDbType.VarChar;

                SqlParameter idPrm = new SqlParameter();

                idPrm.ParameterName = "@prodId";

                idPrm.Value = prodID;

                idPrm.Direction = ParameterDirection.Input;

                idPrm.SqlDbType = SqlDbType.Int;

                SqlParameter costPrm= new SqlParameter();

                costPrm.ParameterName = "@prodCost";

                costPrm.Value = prodCost;

                costPrm.Direction = ParameterDirection.Input;

                costPrm.SqlDbType = SqlDbType.Int;

                SqlParameter orderDatePrm = new SqlParameter();

                orderDatePrm.ParameterName = "@orderdate";

                orderDatePrm.Value = DateTime.Now;

                orderDatePrm.Direction = ParameterDirection.Input;

                orderDatePrm.SqlDbType = SqlDbType.DateTime;

                cmdObj.Parameters.Add(emailPrm);

                cmdObj.Parameters.Add(idPrm);

                cmdObj.Parameters.Add(costPrm);

                cmdObj.Parameters.Add(orderDatePrm);

                SqlParameter prmReturnValue = new SqlParameter();

                prmReturnValue.Direction = ParameterDirection.ReturnValue;

                cmdObj.Parameters.Add(prmReturnValue);

                conObj.Open();

                cmdObj.ExecuteNonQuery();

                int res = Convert.ToInt32(prmReturnValue.Value);

                if (res == 1)

                    result = 1;//it means added

                else

                    result = 0;//error

            }

            catch (Exception e)

            {

                result = -1;

            }

            finally

            {

                conObj.Close();

            }

            log.LogInformation("" + result);

        }

**TASK-8: IMPLEMENTING API MANAGEMENT**

**Goal**:

* Creating API management resource
* Working with Policies.
* Working with Developer’s Portal.

1. Make sure that QuickKartwebService, login function app, Payment Microservice are hosted and ready.

2. Create **API management resource**.

3. Put the APIs behind the **gateway** and Test it from the Postman.

4. Make sure APIs can only be accessed via Gateway URL hence ensure App Service can only allow traffic from Gateway. Direct access should not be allowed.

5. Ensure that this IP is blocked to access all the APIs 223.190.253.129

6. Ensure any response coming from the App Service to API Gateway is added with an Extra header, “Sample Key”:”Sample value”

7. Create a Product called “QuickKart-Product” and add all the APIs in this Product.

8. Set up Developer’s Portal, Make sure guests can view the QuickKart-Product.

9. Let the users sign up to developer’s portal and obtain the subscription of the Product.

10. View the list of developers from Portal.