Dharmsinh Desai University, Nadiad
Faculty of Technology

Computer Engineering Department
B.Tech - Semester - VI

Subject :- Web Services Development

Project Title : Invoice Generator

Guided By:- Prof. Prashant M. Jadav / Prof. Ankit P. Vaishnav

NILKANTH SATISHBHAI PATEL (CE099) [20CEUOS051]

Dharmsinh Desai University, Nadiad
Department of Computer Engineering
Faculty of Technology

CERTIFICATE

This is to certify that the project work carried out
in the subject of
Web Services Development
Is the bonafide work of

NILKANTH SATISHBHAI PATEL (CE099) [20CEUOS051]

Of B.Tech. Semester-VI Computer Engineering during the academic Year **2022-2023**

Prof. Ankit P. Vaishnav
Associate Professor
Computer Engg. Department
Faculty of Technology
D.D.U., Nadiad

Dr. C. K. Bhensdadia
Professor and HoD,
Computer Engg. Department
Faculty of Technology
D.D.U.,Nadiad

➢ Abstracts :-

The main objective of the project is to perform CRUD operations on users of invoice management users. Which are cashier and admin. Admin can add other admin and cashier. Simple Invoice generator also added to the project.

➢ Introduction :-

Invoice generation is the process which is done by cashier. This is about manage Items and Manage user's information. That user may be an admin or a cashier. Many applications have their own unique features, storage, organization, and sharing capabilities. In this app admin can manage user's and cashiers can generate invoice.

➢ **Technology Used :-**

- Html
- Css
- .NET Framework
- EF Core
- Dot net core mvc
- Mssql

➢ **Tools Used :-**

- Visual Studio 2019
- MsSql Management Studio
- Google Chrome

System Requirement Specification (SRS) :-

Users Of the Application :-

1.) Admin

Functional Requirements:-

R1: Manage Items

R2: Manage Users

R3: Generate Invoice

R.1 :- Manage Items

R.1.1 :- Insert Item

Description : - Only admin is able to add item in system. Cashier or any other user don't have the authority to add, delete or update item.

Input:- Id, Item name, Item quantity, Item price.

Output:- Data will be successfully stored in database.

R.1.2 :- Update Item

Description : - Only admin is able to update item in system. Cashier or any other user don’t have the authority to add, delete or update item. But here input Id has to be same as original.

Input:- Id, Item name, Item quantity, Item price.

Output:- Data will be successfully updated in database.

R.1.3 :- Delete Item

Description : - Only admin is able to remove item in system. Cashier or any other user don’t have the authority to add, delete or update item.

Input:- Item Id.

Output:- Data will be successfully removed from database.

R.2 :- Manage Users

R.2.1 :- Insert Cashier

Description : - Only admin is able to add cashier or new admin in system.

Input:- Id, name, role.

Output:- User will be successfully stored in database.

R.2.2 :- Update User Info

Description : - Only admin is able to update user's details in system.

Input:- Id, name, role.

Output:- Data will be successfully updated in database.

R.2.3 :- Delete User

Description : - Only admin is able to remove specific user in system. It could be an admin or cashier.

Input:- User Id.

Output:- Data will be successfully removed from database.

R.3 :- Generate Invoice

Description : - This job is done by cashier. Items will be fetched from database and then cashier can manage invoice of user.

Input:- Name, Phone No, Item, Quantity, Price.

Output:- Total amount of bill will be returned.

➢ Models/Contracts :-

1. For Items:

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    IEnumerable<Item> GetItems();
    [OperationContract]
    void InsertItem(Item iobj);
    [OperationContract]
    void UpdateItem(Item iobj);
    [OperationContract]
    void DeleteItem(int id);
}
```

```
[DataContract]
public class Item
{
    [DataMember]
    [Key]
    [Required]
    public int iid { get; set; }
    [DataMember]
    [Required]
    public string iName { get; set; }
    [DataMember]
    [Required]
    public int iQuantity { get; set; }
    [DataMember]
    [Required]
    public string iPrice { get; set; }

}
```

2. For Users:

```
[ServiceContract]
 public interface ICashierService1
 {
     [OperationContract]
     IEnumerable<Cashier> GetCashiers();
     [OperationContract]
     void InsertCashier(Cashier cobj);
     [OperationContract]
     void UpdateCashier(Cashier cobj);
     [OperationContract]
     void DeleteCashier(int id);
}


  [DataContract]
  public class Cashier
  {
      [DataMember]
      [Key]
      [Required]
      public int cid { get; set; }
      [DataMember]
      [Required]
      public string cName { get; set; }
      [DataMember]
      [Required]
      public string cRole { get; set; }

}
```

➢ **Implementation :-**

- **Function to insert item:**

  It will help to add new item in the system.

```
public void InsertItem(Item iobj)
```

- **Function to update item:**

  It will help to update existing item in the system.

```
public void UpdateItem(Item iobj)
```

- **Function to delete item:**

  It will help to delete existing item in the system.

```
public void DeleteItem(int id)
```

- **Function to get items:**

  It is used to see the list of all existing items in the system.

```
public IEnumerable<Item> GetItems()
```

- **Function to insert user:**

  It will help to add new user in the system.

```
public void InsertCashier(Cashier cobj)
```

- **Function to update user:**

  It will help to update existing user details in the system.

```
public void UpdateCashier(Cashier cobj)
```

- **Function to delete user:**

  It will help to delete existing user from the system.

```
public void DeleteCashier(int id)
```

- **Function to get users:**

  It is used to see the list of all existing users in the system.

```
public IEnumerable<Cashier> GetCashiers()
```

➢ **Testing** :-

For testing the our application, a mixed approach integration testing and regression testing is used.

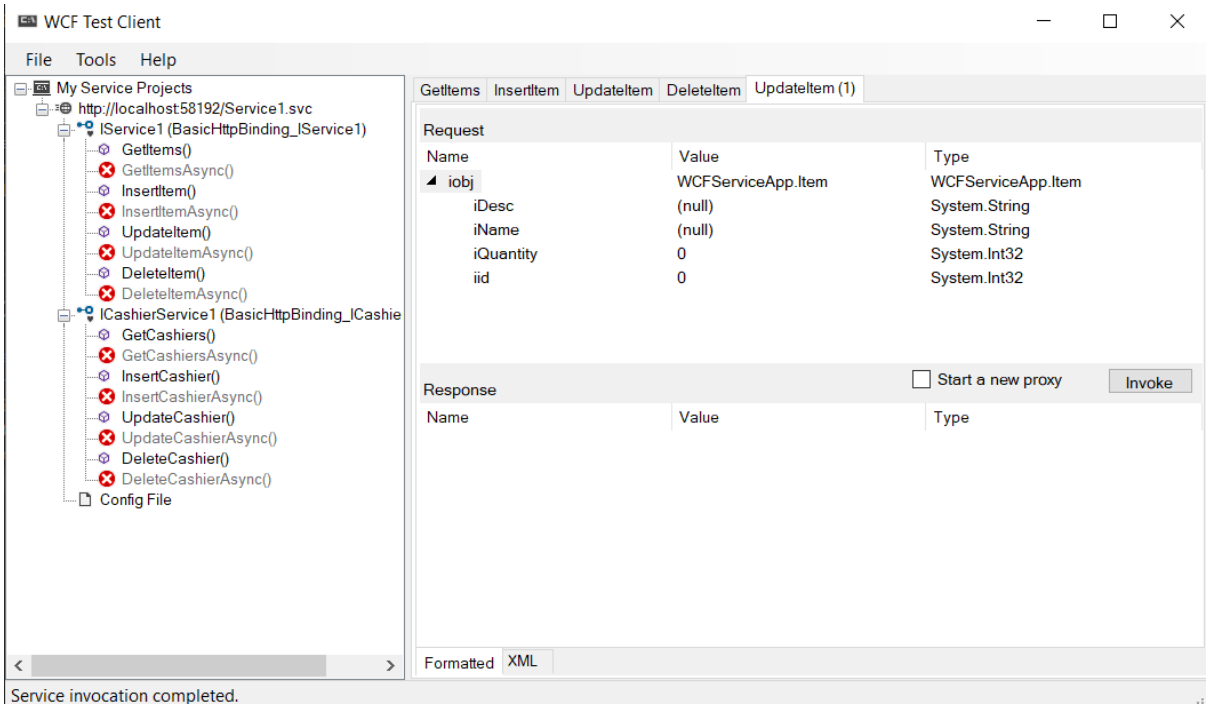
• Integration testing : each main part of application is tested after each small small function is tested first and then combine them and test that main part.

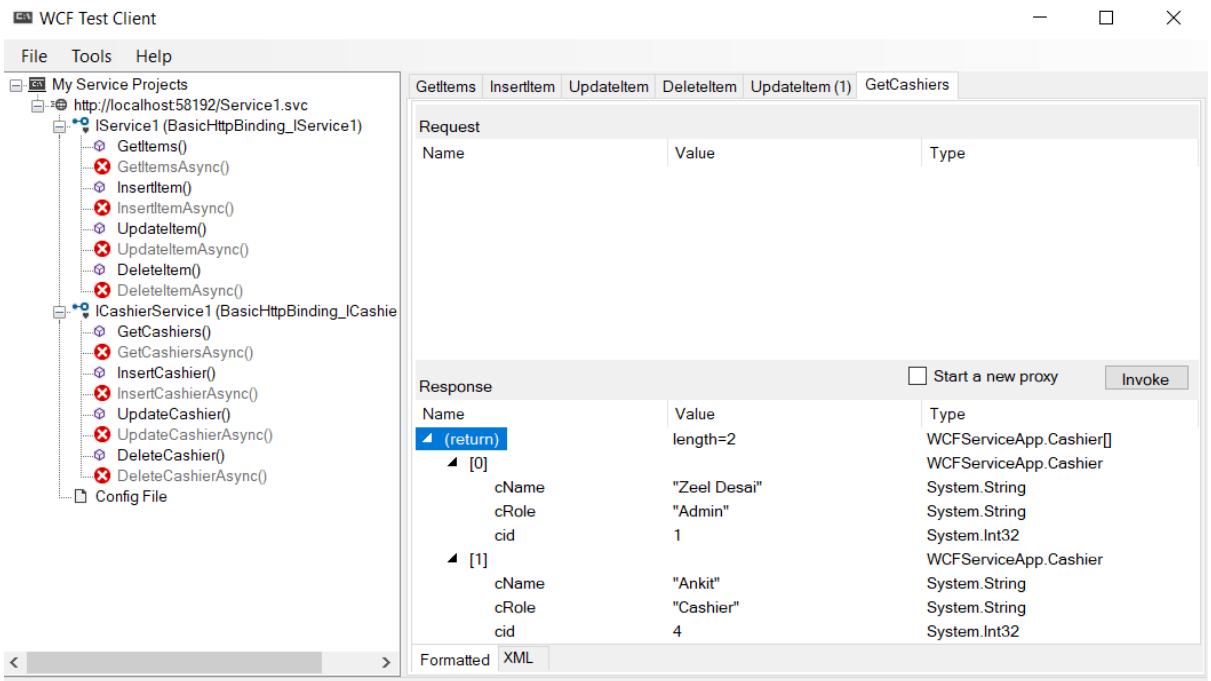
• Regression testing : After main part of application is created, added them in the whole system, and then test the whole system that make sure the whole system is work fine after adding some main part.

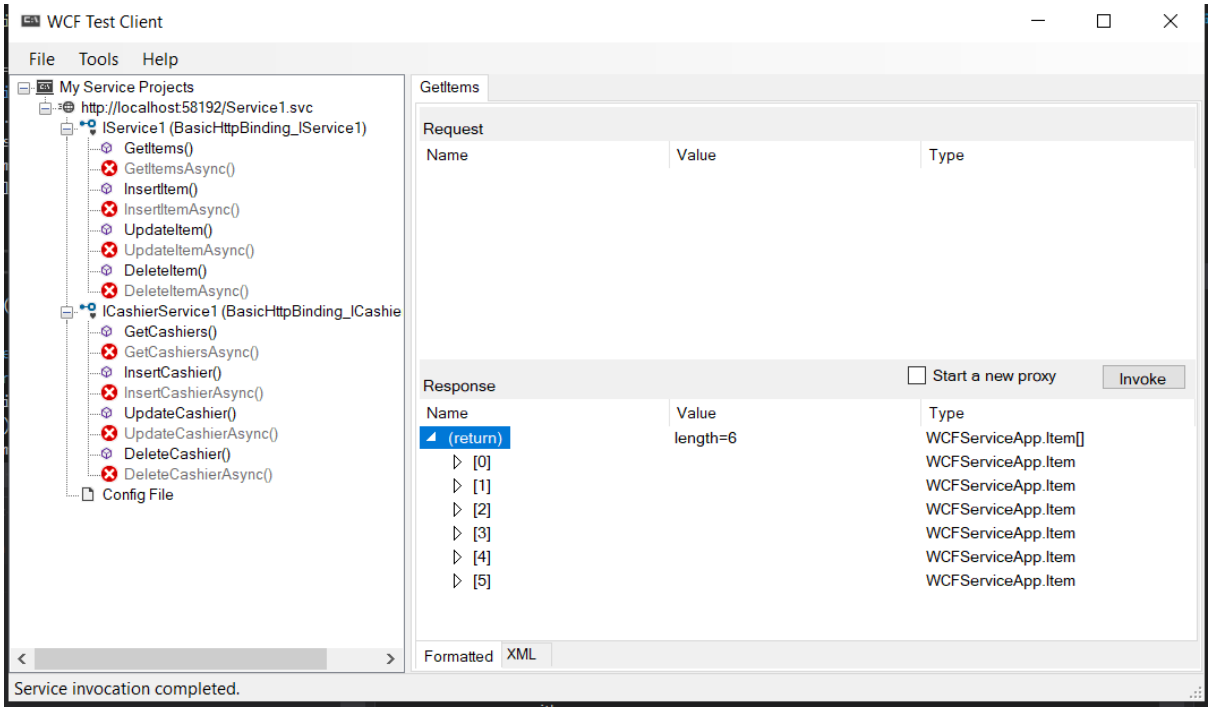
Manual Testing :- Manual testing is used to find and fix the bug in our application.

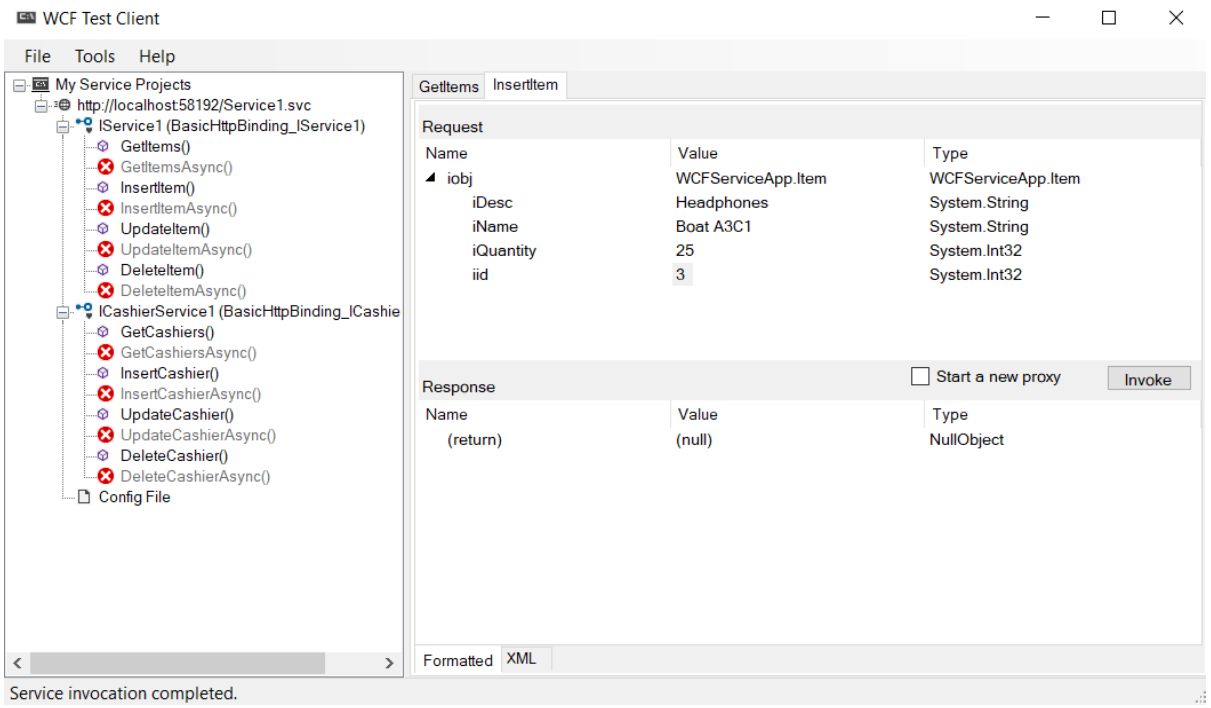
➢ Screenshots :-

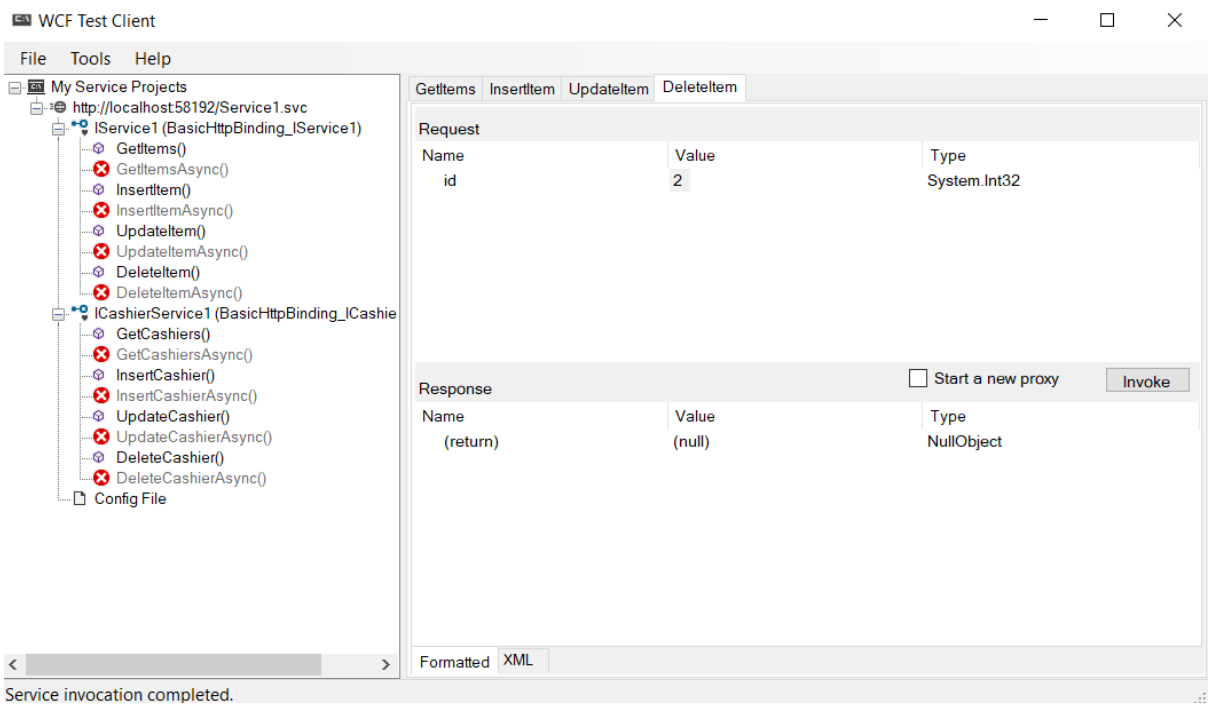
Screenshots of WebServiceApp :



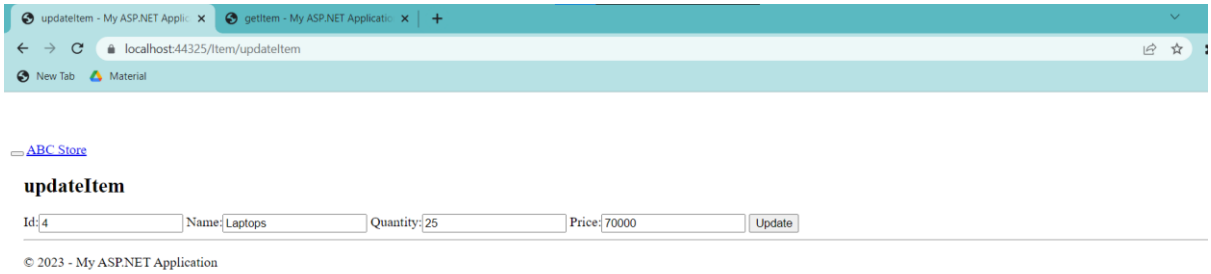

WCF Test Client
File
Tools
Help
My Service Projects
http://localhost:58192/Service1.svc
IService1 (BasicHttpBinding_IService1)
GetItems()
GetItemsAsync()
InsertItem()
InsertItemAsync()
UpdateItem()
UpdateItemAsync()
DeleteItem()
DeleteItemAsync()
ICashierService1 (BasicHttpBinding_ICashie
GetCashiers()
GetCashiersAsync()
InsertCashier()
InsertCashierAsync()
UpdateCashier()
UpdateCashierAsync()
DeleteCashier()
DeleteCashierAsync()
Config File
GetItems
InsertItem
UpdateItem
DeleteItem
UpdateItem (1)
GetCashiers
Request
Name
Value
Type
Response
Start a new proxy
Invoke
Name
Value
Type
(return)
length=2
WCFServiceApp.Cashier[]
[0]
WCFServiceApp.Cashier
cName
"Zeel Desai"
System.String
cRole
"Admin"
System.String
cid
1
System.Int32
[1]
WCFServiceApp.Cashier
cName
"Ankit"
System.String
cRole
"Cashier"
System.String
cid
4
System.Int32
Formatted
XML
Service invocation completed.

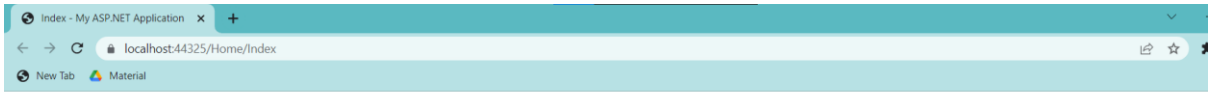

WCF Test Client
File
Tools
Help
My Service Projects
http://localhost:58192/Service1.svc
IService1 (BasicHttpBinding_IService1)
GetItems()
GetItemsAsync()
InsertItem()
InsertItemAsync()
UpdateItem()
UpdateItemAsync()
DeleteItem()
DeleteItemAsync()
ICashierService1 (BasicHttpBinding_ICashie
GetCashiers()
GetCashiersAsync()
InsertCashier()
InsertCashierAsync()
UpdateCashier()
UpdateCashierAsync()
DeleteCashier()
DeleteCashierAsync()
Config File
GetItems
Request
Name
Value
Type
Response
Start a new proxy
Invoke
Name
Value
Type
(return)
length=6
WCFServiceApp.Item[]
[0]
WCFServiceApp.Item
[1]
WCFServiceApp.Item
[2]
WCFServiceApp.Item
[3]
WCFServiceApp.Item
[4]
WCFServiceApp.Item
[5]
WCFServiceApp.Item
Formatted
XML
Service invocation completed.


WCF Test Client
File
Tools
Help
My Service Projects
http://localhost:58192/Service1.svc
IService1 (BasicHttpBinding_IService1)
GetItems()
GetItemsAsync()
InsertItem()
InsertItemAsync()
UpdateItem()
UpdateItemAsync()
DeleteItem()
DeleteItemAsync()
ICashierService1 (BasicHttpBinding_ICashie
GetCashiers()
GetCashiersAsync()
InsertCashier()
InsertCashierAsync()
UpdateCashier()
UpdateCashierAsync()
DeleteCashier()
DeleteCashierAsync()
Config File
GetItems
InsertItem
Request
Name
Value
Type
iobj
WCFServiceApp.Item
WCFServiceApp.Item
iDesc
Headphones
System.String
iName
Boat A3C1
System.String
iQuantity
25
System.Int32
iid
3
System.Int32
Start a new proxy
Invoke
Response
Name
Value
Type
(return)
(null)
NullObject
Formatted
XML
Service invocation completed.


WCF Test Client
File
Tools
Help
My Service Projects
http://localhost:58192/Service1.svc
IService1 (BasicHttpBinding_IService1)
GetItems()
GetItemsAsync()
InsertItem()
InsertItemAsync()
UpdateItem()
UpdateItemAsync()
DeleteItem()
DeleteItemAsync()
ICashierService1 (BasicHttpBinding_ICashie
GetCashiers()
GetCashiersAsync()
InsertCashier()
InsertCashierAsync()
UpdateCashier()
UpdateCashierAsync()
DeleteCashier()
DeleteCashierAsync()
Config File
GetItems
InsertItem
UpdateItem
DeleteItem
Request
Name
Value
Type
id
2
System.Int32
Start a new proxy
Invoke
Response
Name
Value
Type
(return)
(null)
NullObject
Formatted
XML
Service invocation completed.

Screenshots of WebApp :





ABC Store

Index

Name:

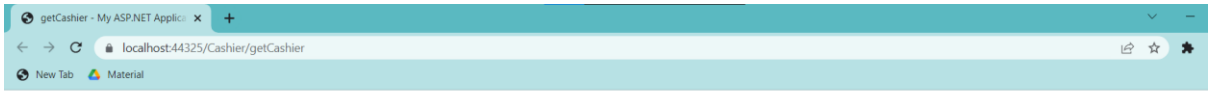
Mobile:

Item: Select Item

Quantity:

Price:

Add In List



ABC Store

getCashier

C_Id	C_Name	C_Role
1	Zeel Desai	Admin
4	Ankit	Cashier

ABC Store

insertCashier

Id: 3 Name: Vivek Role: Cashier Insert

ABC Store

deleteCashier

Id: 3 Delete

ABC Store

updateCashier

Id: 3 Name: Vivek Patel Role: Cashier Update

ABC Store

getItem

Id	Name	Quantity	Price
4	Laptops	25	60000
6	Pen Box	60	100
7	Penci Box	20	60
8	Speaker	10	2240
9	AC	15	22000
10	Speaker Pair	45	3000

Insert Update Delete

ABC Store

insertItem

Id: 1 Name: TV Quantity: 10 Price: 12000 Insert

ABC Store

deleteItem

Id: 8 Delete

- **Conclusion :-**

The functionality are implemented in system after understanding all the thing .

Functionalities that are successfully added into the application are:

User functionality:-

1. Add user
2. View all users
3. Update user
4. Delete user

(All Same functionalities are used for items)

- **Limitation and Future Extension :-**

Limitation :- This is application is not accurate as per today's invoice management. It requires lot more functionalities to be implemented. This is the simplest version invoice generator. But for learning purpose of WCF with EF core I can came up with this much for now.

Future Extension :- I will upgrade this app to more user friendly. I will work on better data management of items and users. Also authentication and authorization will be implemented soon.