SCHOOL OF COMPUTING AND COMPUTER ENGINEERING

CSC511: DBMS DESIGN

PROFESSOR: DR. BO LI

PROJECT REPORT – ELECTRONIC VENDOR APPLICATION

SUBMISSION DATE: 04-05-2023

GROUP MEMBERS:

1) Sumanth Kumar Gogineni
2) Rajya Lakshmi Moparthy
3) Sai Teja Reddy Vanga
4) Sohail Khan
5) Prashanth Kumar Machani

TABLE OF CONTENTS

1. Abstract

The idea starts or arises when the users/people wanted to make life better by sitting at home and purchasing items or food (Amazon, Walmart, COSTCO), booking tickets (Flight tickets, movie tickets), bookings (hotel, restaurant), online entertainment (NETFLIX), online payments (Google Pay, Pay Pal, etc.), reading and signing documents online (Adobe), etc. All these are the common things that we do in our day-to-day life. But to make life easier and simpler there comes the concept of E-commerce applications.

Coming to the examples of E-commerce applications we have:

1. AMAZON
2. NETFLIX etc.

In this project, we have selected and implemented the E-commerce application. This describes an e-commerce application that enables users to search a database for a wide variety of products but requires them to log in first. To create an account, new users may be required to provide personal information that cannot be changed after account creation. Once logged in, users can browse products and add them to their cart, where they can review and remove items prior to completing a purchase. Furthermore, the application allow users to place order and admin to request manufacturer to restock the products in the stores if the inventory is low in the stores.

The application prioritizes user privacy and security by requiring authentication and accumulating only the required personal data. The cart dashboard feature gives users more control over their purchasing experience, while the overall design of the application strives to be intuitive and simple to navigate. The transaction procedure is designed to be streamlined and uncomplicated, allowing the user to easily input payment and shipping information and receive an order confirmation.

This e-commerce application's primary aim is to give users a purchasing experience that is both easy and safe while also putting a strong focus on the simplicity of use and control that users have over the application.

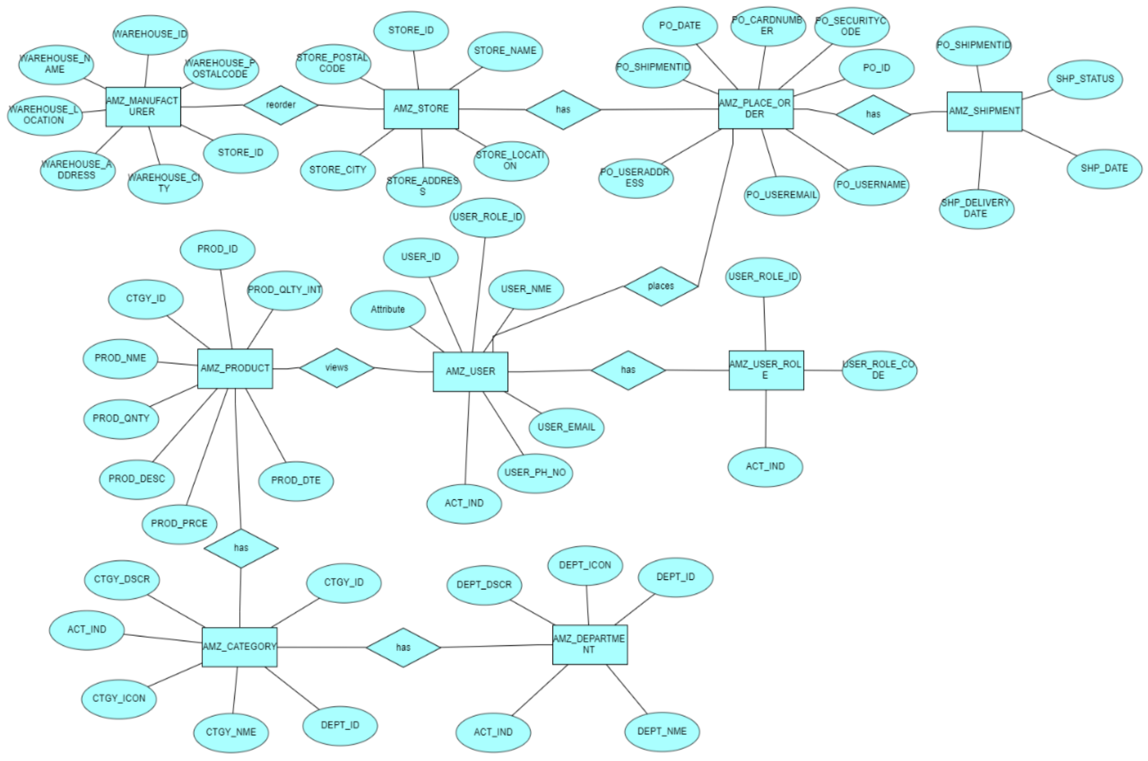
2. Problem Summary

Problem Summary: The purpose of the application could be to provide a platform for vendors to sell their products online, manage their inventory, track sales, and process payments.

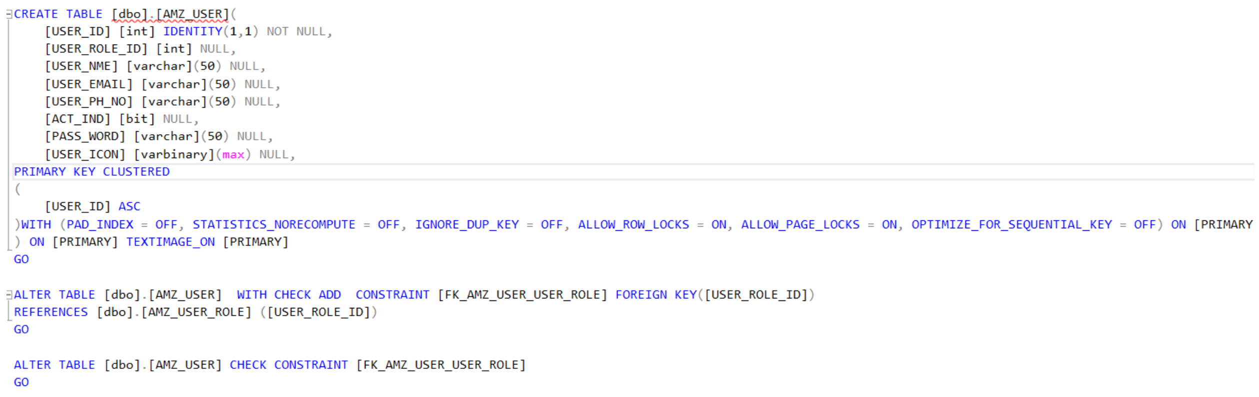
- The project's goal was to create an e-commerce shopping application that would let users browse and buy things.
- The application was created with a sign-up and sign-in page to enable safe system access and a search feature to make it simpler for users to find products.
- Customers could inspect and remove goods from their carts before making a purchase using the cart dashboard and the items dashboard, which showed all of the products that were available.
- Allow users to place orders and also allow admit reorder stock if the stock available in the store is limited. API are created to allow users to place orders and admin to request manufacturer to restock the products in the stores.
- The back end of the program was created using C#, the front-end was created using JavaScript, and the database was created using SQL Server.
- During the project, the team learned about database architecture, entity framework, object-oriented programming, SOLID principles, and the value of creating ER and relational diagrams before beginning to construct database tables and views.
- Overall, the project provided a great exposure to the core programming concepts involved in developing a real-time e-commerce application.

3. Database Design

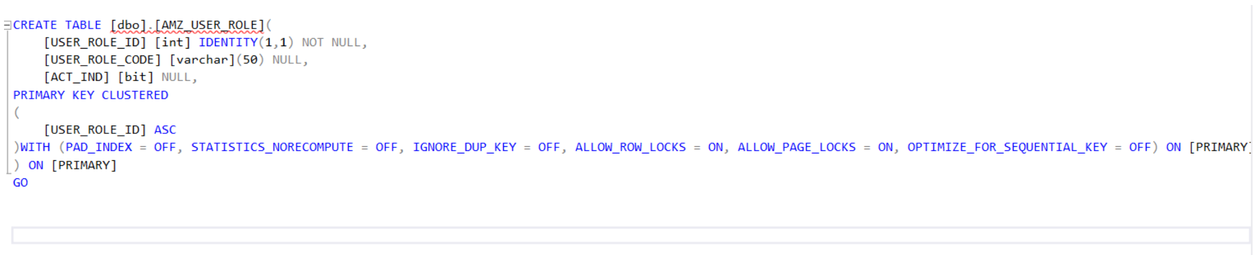
Entity- Relationship Diagram:



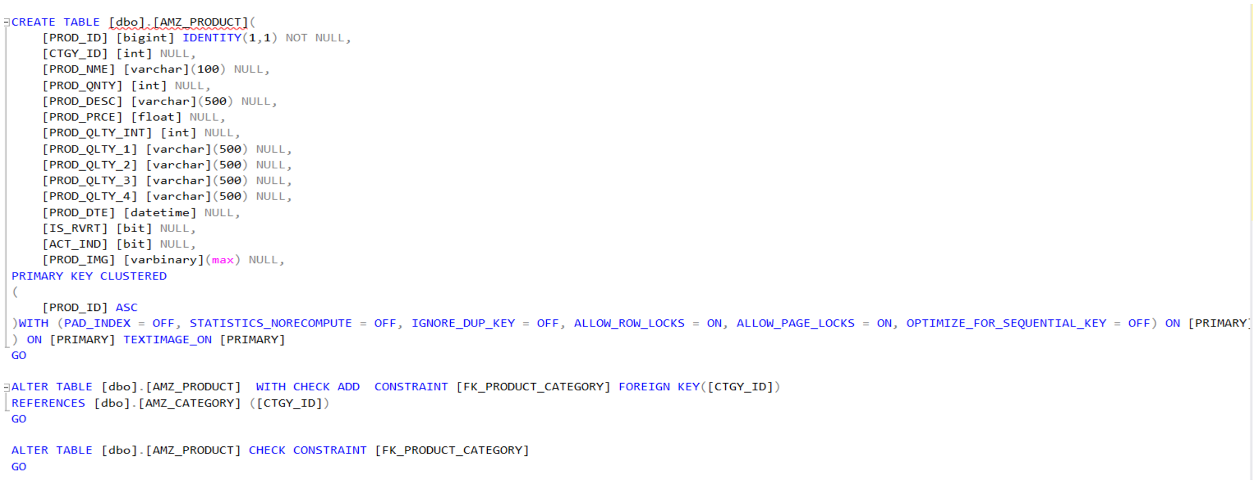
4. Relational Schema:



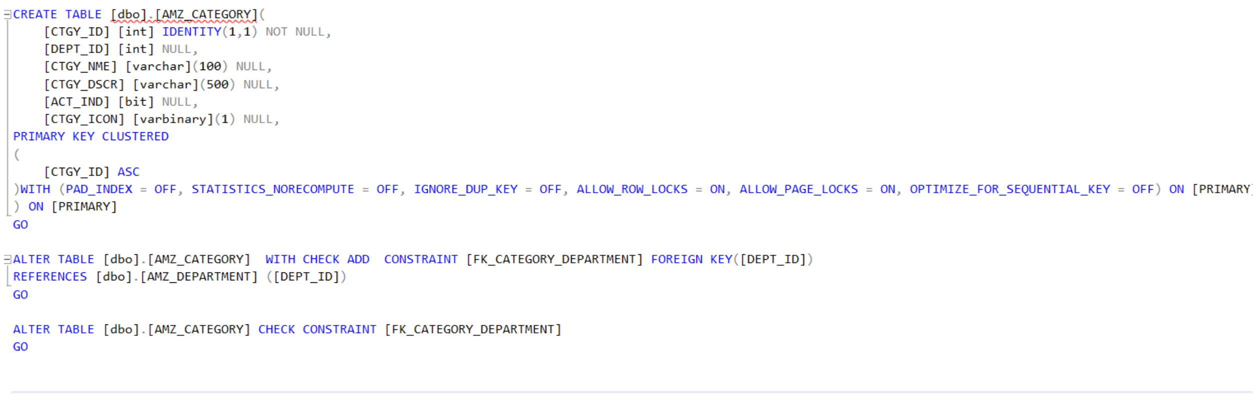
```
CREATE TABLE [dbo].[AMZ_USER](
    [USER_ID] [int] IDENTITY(1,1) NOT NULL,
    [USER_ROLE_ID] [int] NULL,
    [USER_NME] [varchar](50) NULL,
    [USER_EMAIL] [varchar](50) NULL,
    [USER_PH_NO] [varchar](50) NULL,
    [ACT_IND] [bit] NULL,
    [PASS_WORD] [varchar](50) NULL,
    [USER_ICON] [varbinary](max) NULL,
PRIMARY KEY CLUSTERED
(
    [USER_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[AMZ_USER]  WITH CHECK ADD  CONSTRAINT [FK_AMZ_USER_USER_ROLE] FOREIGN KEY([USER_ROLE_ID])
REFERENCES [dbo].[AMZ_USER_ROLE] ([USER_ROLE_ID])
GO

ALTER TABLE [dbo].[AMZ_USER] CHECK CONSTRAINT [FK_AMZ_USER_USER_ROLE]
GO
```



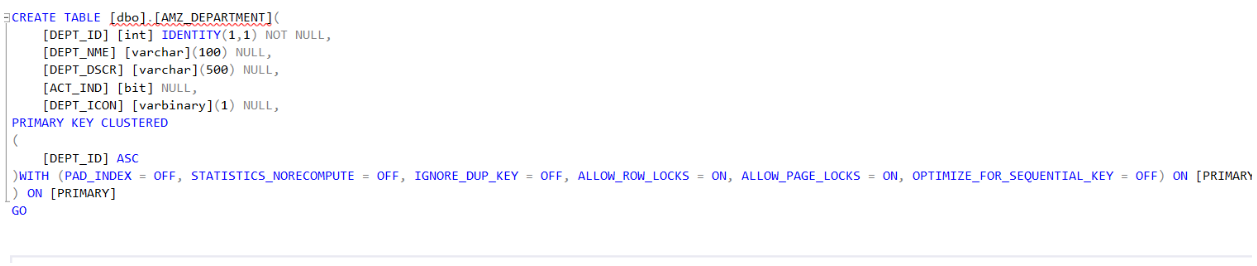
```sql
CREATE TABLE [dbo].[AMZ_USER_ROLE](
	[USER_ROLE_ID] [int] IDENTITY(1,1) NOT NULL,
	[USER_ROLE_CODE] [varchar](50) NULL,
	[ACT_IND] [bit] NULL,
PRIMARY KEY CLUSTERED
(
	[USER_ROLE_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```



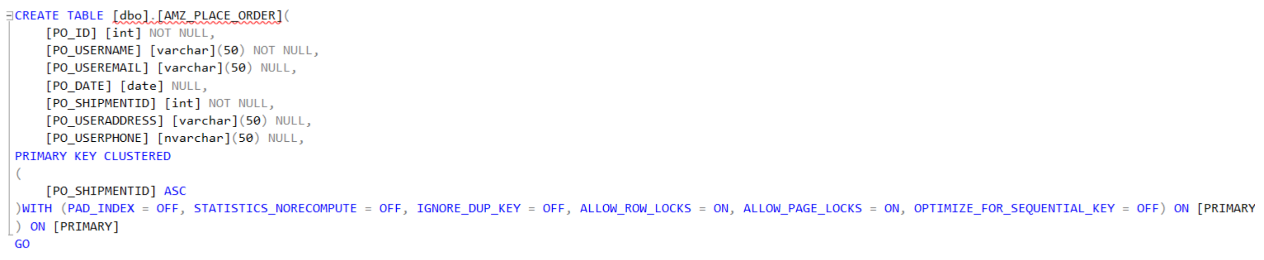
```sql
CREATE TABLE [dbo].[AMZ_PRODUCT](
	[PROD_ID] [bigint] IDENTITY(1,1) NOT NULL,
	[CTGY_ID] [int] NULL,
	[PROD_NME] [varchar](100) NULL,
	[PROD_QNTY] [int] NULL,
	[PROD_DESC] [varchar](500) NULL,
	[PROD_PRCE] [float] NULL,
	[PROD_QLTY_INT] [int] NULL,
	[PROD_QLTY_1] [varchar](500) NULL,
	[PROD_QLTY_2] [varchar](500) NULL,
	[PROD_QLTY_3] [varchar](500) NULL,
	[PROD_QLTY_4] [varchar](500) NULL,
	[PROD_DTE] [datetime] NULL,
	[IS_RVRT] [bit] NULL,
	[ACT_IND] [bit] NULL,
	[PROD_IMG] [varbinary](max) NULL,
PRIMARY KEY CLUSTERED
(
	[PROD_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[AMZ_PRODUCT]  WITH CHECK ADD  CONSTRAINT [FK_PRODUCT_CATEGORY] FOREIGN KEY([CTGY_ID])
REFERENCES [dbo].[AMZ_CATEGORY] ([CTGY_ID])
GO

ALTER TABLE [dbo].[AMZ_PRODUCT] CHECK CONSTRAINT [FK_PRODUCT_CATEGORY]
GO
```



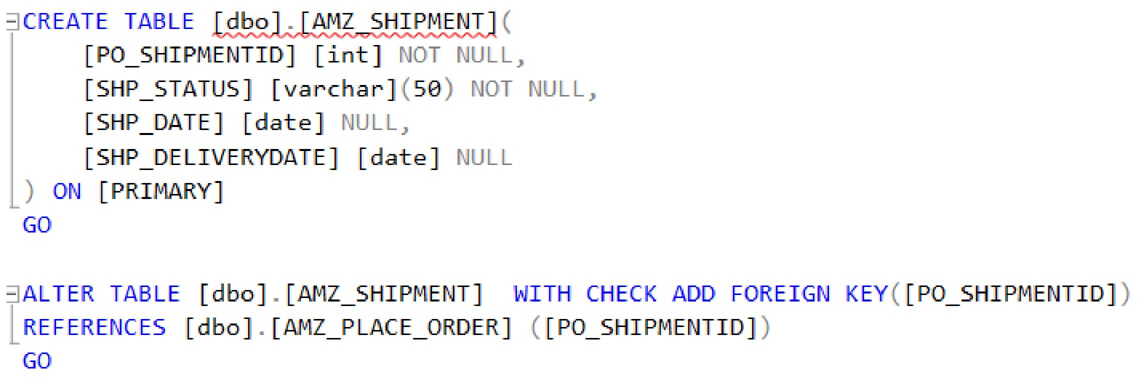
```sql
CREATE TABLE [dbo].[AMZ_CATEGORY](
	[CTGY_ID] [int] IDENTITY(1,1) NOT NULL,
	[DEPT_ID] [int] NULL,
	[CTGY_NME] [varchar](100) NULL,
	[CTGY_DSCR] [varchar](500) NULL,
	[ACT_IND] [bit] NULL,
	[CTGY_ICON] [varbinary](1) NULL,
PRIMARY KEY CLUSTERED
(
	[CTGY_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AMZ_CATEGORY]  WITH CHECK ADD  CONSTRAINT [FK_CATEGORY_DEPARTMENT] FOREIGN KEY([DEPT_ID])
REFERENCES [dbo].[AMZ_DEPARTMENT] ([DEPT_ID])
GO

ALTER TABLE [dbo].[AMZ_CATEGORY] CHECK CONSTRAINT [FK_CATEGORY_DEPARTMENT]
GO
```



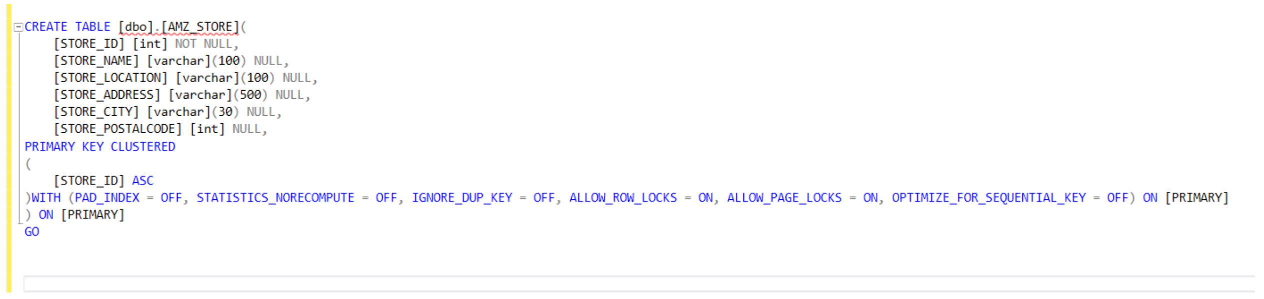
```sql
CREATE TABLE [dbo].[AMZ_DEPARTMENT](
	[DEPT_ID] [int] IDENTITY(1,1) NOT NULL,
	[DEPT_NME] [varchar](100) NULL,
	[DEPT_DSCR] [varchar](500) NULL,
	[ACT_IND] [bit] NULL,
	[DEPT_ICON] [varbinary](1) NULL,
PRIMARY KEY CLUSTERED
(
	[DEPT_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY
) ON [PRIMARY]
GO
```



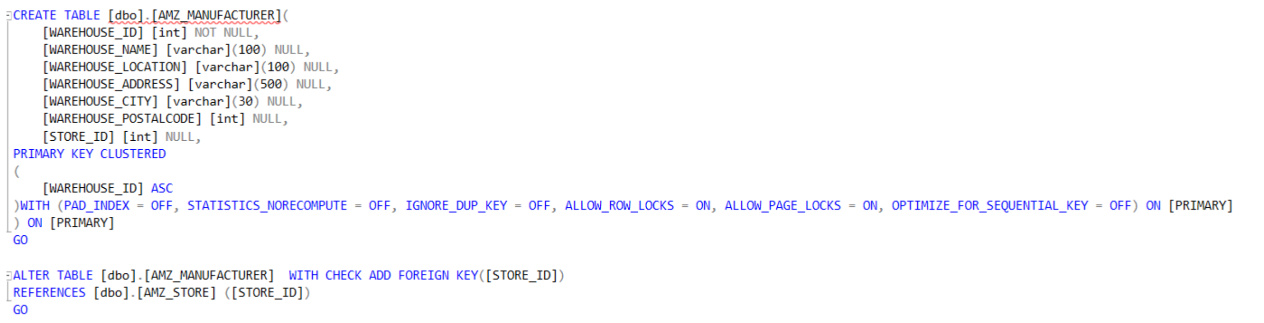
```sql
CREATE TABLE [dbo].[AMZ_PLACE_ORDER](
    [PO_ID] [int] NOT NULL,
    [PO_USERNAME] [varchar](50) NOT NULL,
    [PO_USEREMAIL] [varchar](50) NULL,
    [PO_DATE] [date] NULL,
    [PO_SHIPMENTID] [int] NOT NULL,
    [PO_USERADDRESS] [varchar](50) NULL,
    [PO_USERPHONE] [nvarchar](50) NULL,
PRIMARY KEY CLUSTERED
(
    [PO_SHIPMENTID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY
) ON [PRIMARY]
GO
```



```sql
CREATE TABLE [dbo].[AMZ_SHIPMENT](
    [PO_SHIPMENTID] [int] NOT NULL,
    [SHP_STATUS] [varchar](50) NOT NULL,
    [SHP_DATE] [date] NULL,
    [SHP_DELIVERYDATE] [date] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AMZ_SHIPMENT]  WITH CHECK ADD FOREIGN KEY([PO_SHIPMENTID])
REFERENCES [dbo].[AMZ_PLACE_ORDER] ([PO_SHIPMENTID])
GO
```



```sql
CREATE TABLE [dbo].[AMZ_STORE](
    [STORE_ID] [int] NOT NULL,
    [STORE_NAME] [varchar](100) NULL,
    [STORE_LOCATION] [varchar](100) NULL,
    [STORE_ADDRESS] [varchar](500) NULL,
    [STORE_CITY] [varchar](30) NULL,
    [STORE_POSTALCODE] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [STORE_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```



```sql
CREATE TABLE [dbo].[AMZ_MANUFACTURER](
    [WAREHOUSE_ID] [int] NOT NULL,
    [WAREHOUSE_NAME] [varchar](100) NULL,
    [WAREHOUSE_LOCATION] [varchar](100) NULL,
    [WAREHOUSE_ADDRESS] [varchar](500) NULL,
    [WAREHOUSE_CITY] [varchar](30) NULL,
    [WAREHOUSE_POSTALCODE] [int] NULL,
    [STORE_ID] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [WAREHOUSE_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AMZ_MANUFACTURER]  WITH CHECK ADD FOREIGN KEY([STORE_ID])
REFERENCES [dbo].[AMZ_STORE] ([STORE_ID])
GO
```

5. Implementation Details

Our team's e-commerce application allows any user or customer to browse the enormous variety of products in our database. All users must sign into the application in order to browse the items and enjoy a hassle-free shopping experience. If the user does not already have an account with us, they may be required to sign up for the application by providing personal information. After creating the account, the consumer will no longer be able to alter their personal information. After creating the account, the user will be able to see the huge variety of products offered in our shop and place an order. As part of this project, we designed a cart dashboard, which will allow the consumer to review the products and delete them from the cart before making the order. Furthermore, the consumer should be able to make an order for the things they are interested in. Finally, allow admit reorder stock if the stock available in the store is limited. API are created to allow users to place orders and admin to request manufacturer to restock the products in the stores.

To build any software application we need a good understanding of back-end and front-end programming languages such as Java, C#, python, JavaScript, or TypeScript. As part of this project, we have used C# programming language to develop Web API and JavaScript to develop the user-friendly interface. We have used SQL server database to store the data. The API developed handles the clients request and communicates with database tables and views to feed the data back and populate on the front-end screens. This project aids in the comprehension of the development of an interactive web page and the technology utilized to achieve it. The project has taught us how to utilize back-end and front-end technologies to create a website, steps to connect to a database to retrieve data, and how to modify data and web pages to offer the user a shopping cart application.

6. Concurrency of interfaces

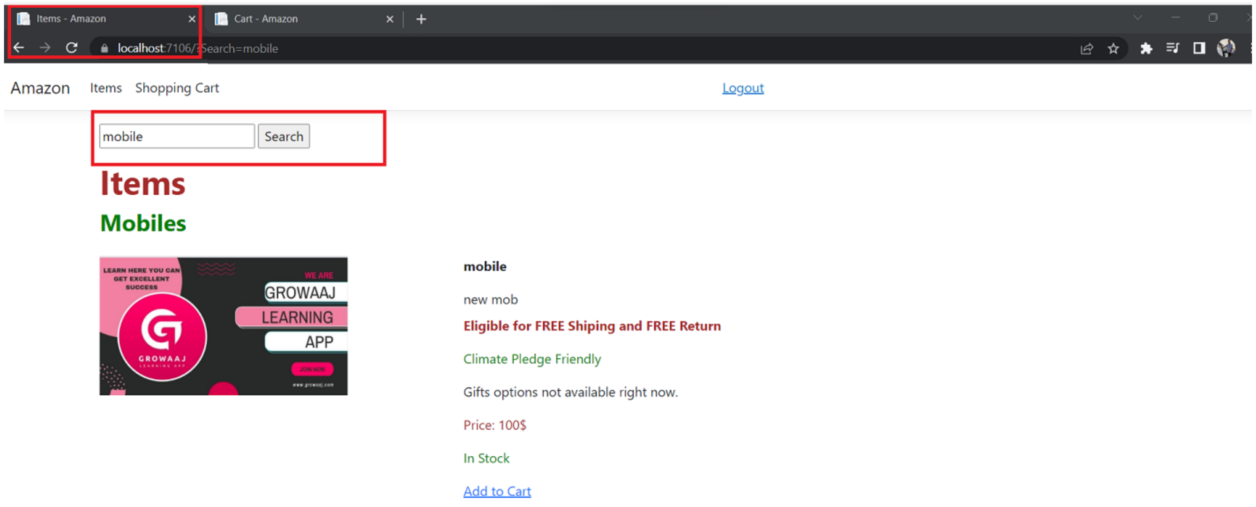
Concurrent operation of interfaces in computer science refers to a system's capacity to handle numerous input/output activities from various interfaces at the same time.

In our project, we have handled concurrent operations of Database by the following:

The user can Sign up and Sign in into the application and can perform different operations and also user can perform placing an order to the Store at the same time using API.

To test the functionality the application is opened in two tabs, in one tab the user is searching for the product, in the other tab another customer adds products to the cart. The applications and table handle both customers request and provide service without interruption. Below are the screenshots provided for the same scenario.

Tab 1:



Items

Mobiles

mobile

new mob

Eligible for FREE Shiping and FREE Return

Climate Pledge Friendly

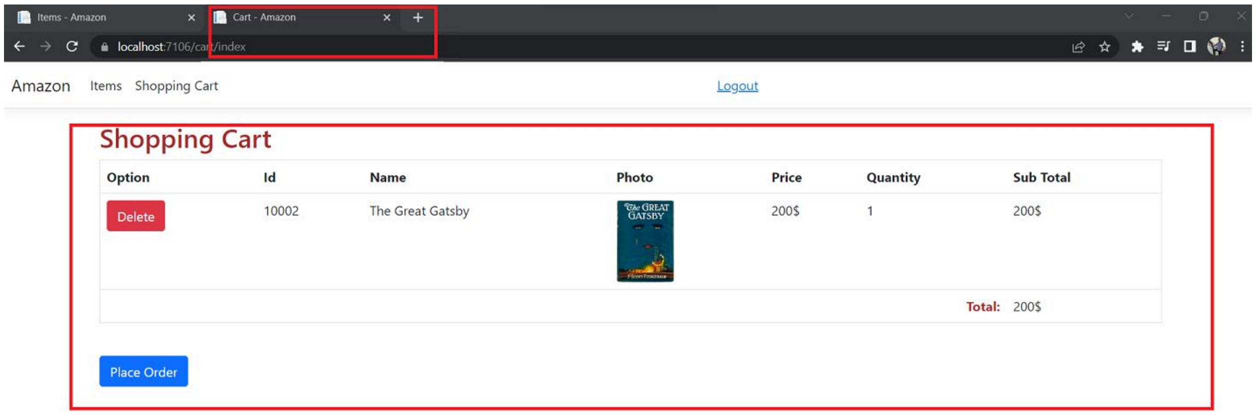
Gifts options not available right now.

Price: 100$

In Stock

Add to Cart

Tab 2:



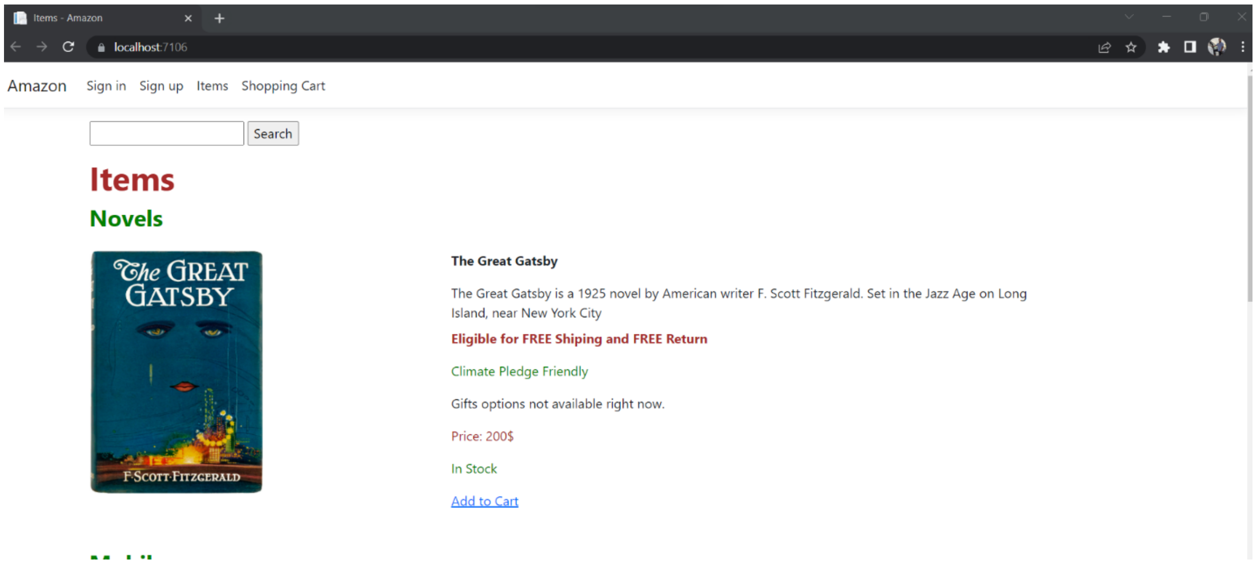
Shopping Cart

Option	Id	Name	Photo	Price	Quantity	Sub Total
Delete	10002	The Great Gatsby		200$	1	200$
					Total:	200$

As shown in above screenshots, our application and tables are capable of handle multiple users and database operation at the same. Moreover, flexible to provide services without interruption.

7. Results and Analysis

Home Page: Below screenshot is the home page. Customers can be able to view the products available in the database.


Items - Amazon

localhost:7106

Amazon Sign in Sign up Items Shopping Cart

Search

Items

Novels

The Great Gatsby

The Great Gatsby is a 1925 novel by American writer F. Scott Fitzgerald. Set in the Jazz Age on Long Island, near New York City

Eligible for FREE Shiping and FREE Return

Climate Pledge Friendly

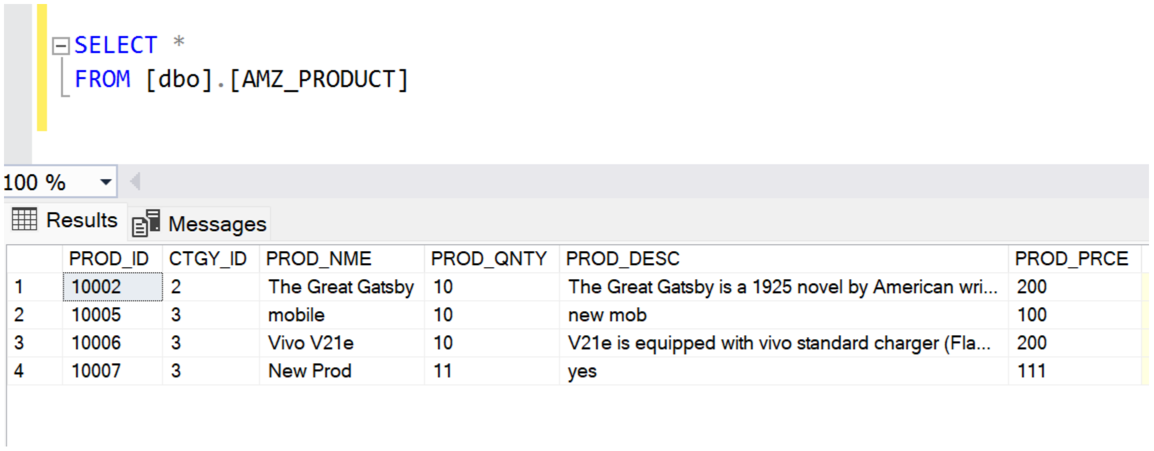
Gifts options not available right now.

Price: 200$

In Stock

Add to Cart

Different products information is in the Product Table in our database.

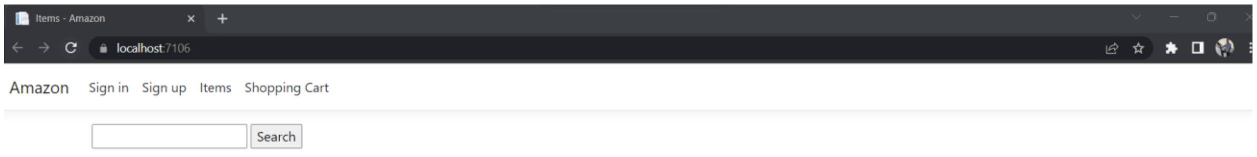

```
SELECT *
FROM [dbo].[AMZ_PRODUCT]
```

100 %

Results Messages

	PROD_ID	CTGY_ID	PROD_NME	PROD_QNTY	PROD_DESC	PROD_PRCE
1	10002	2	The Great Gatsby	10	The Great Gatsby is a 1925 novel by American wri...	200
2	10005	3	mobile	10	new mob	100
3	10006	3	Vivo V21e	10	V21e is equipped with vivo standard charger (Fla...	200
4	10007	3	New Prod	11	yes	111

On top left corner, there is sign up, sign in, items and shopping cart dashboards and Search option to filter or search for a particular the products.

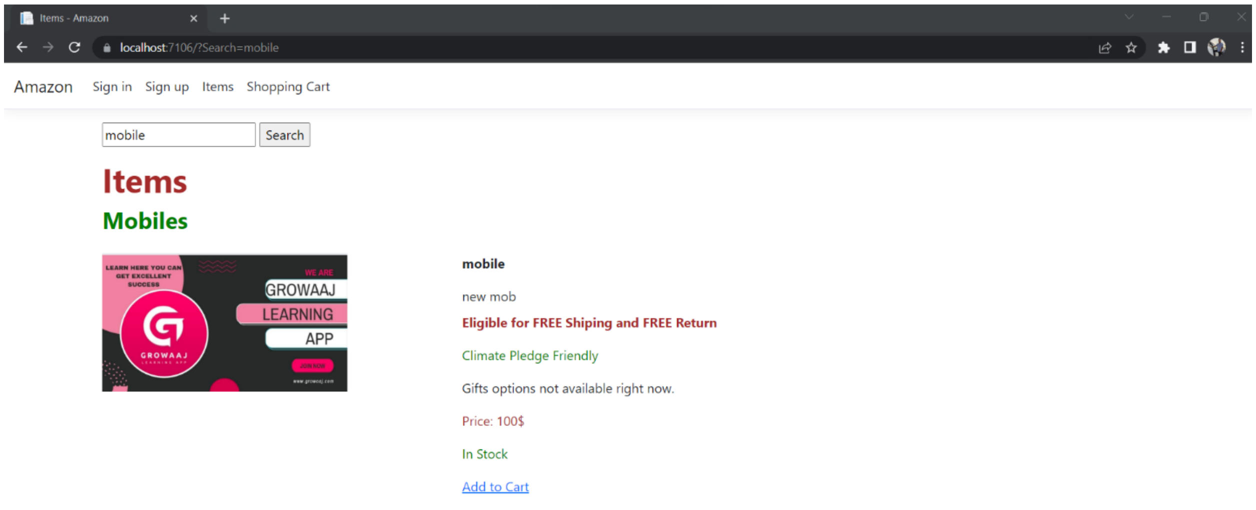

Items - Amazon

localhost:7106

Amazon Sign in Sign up Items Shopping Cart

Search

Search functionality option to filter or search for a particular product and give hustle-free experience to the user. In the screenshot below we have searched for mobile related products.



Items

Mobiles

mobile

new mob

Eligible for FREE Shiping and FREE Return

Climate Pledge Friendly

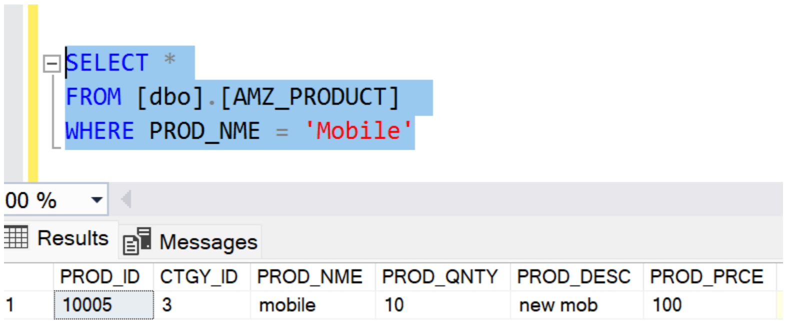
Gifts options not available right now.

Price: 100$

In Stock

Add to Cart

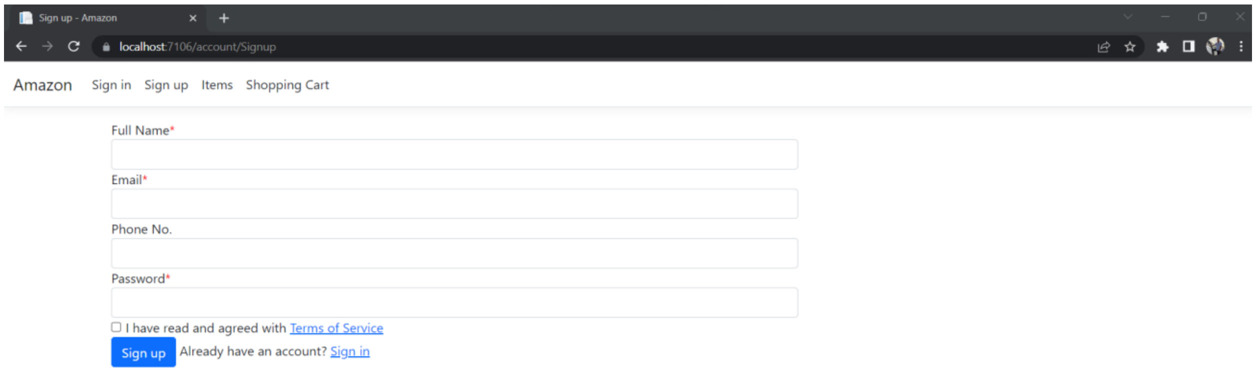
SQL Query for filtering in the database using WHERE condition.



```
SELECT *
FROM [dbo].[AMZ_PRODUCT]
WHERE PROD_NME = 'Mobile'
```

	PROD_ID	CTGY_ID	PROD_NME	PROD_QNTY	PROD_DESC	PROD_PRCE
1	10005	3	mobile	10	new mob	100

Sign up Page: A new customer can create an account to place an order.



Full Name*

Email*

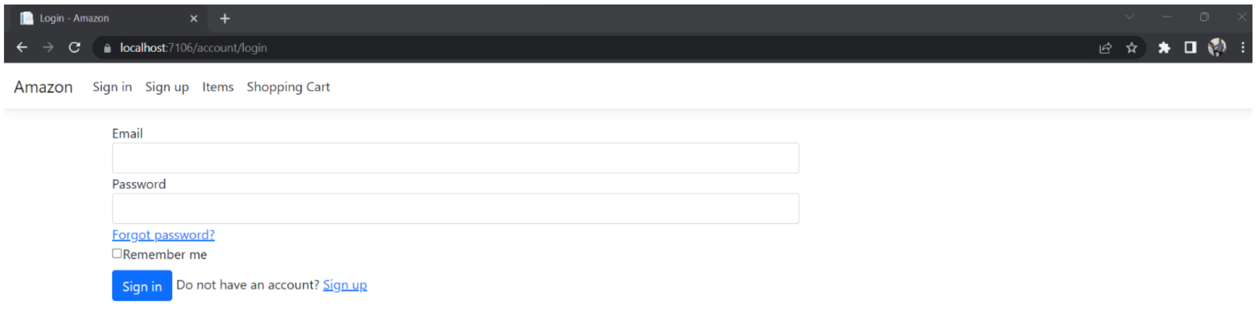
Phone No.

Password*

☐ I have read and agreed with Terms of Service

Sign up Already have an account? Sign in

Sign in page: Exiting customers should be able to login into the application after by providing their login credentials.



Login - Amazon

localhost:7106/account/login

Amazon Sign in Sign up Items Shopping Cart

Email

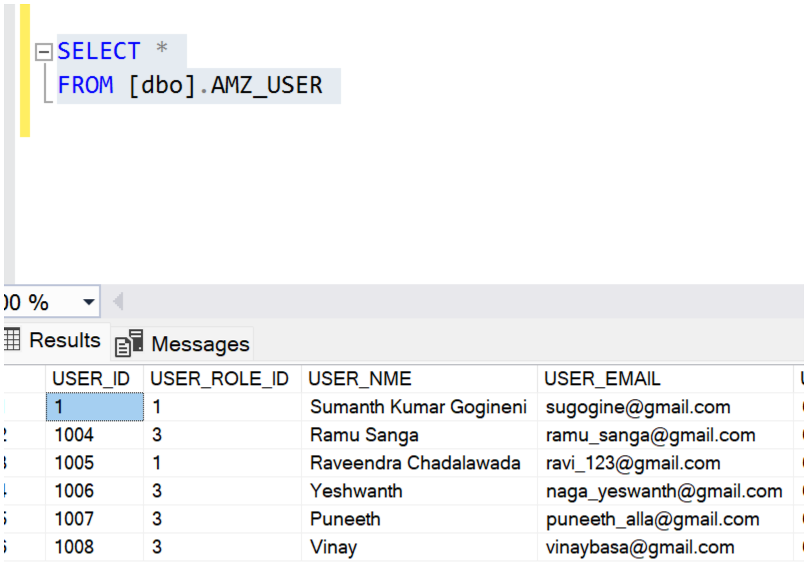
Password

Forgot password?

Remember me

Sign in Do not have an account? Sign up

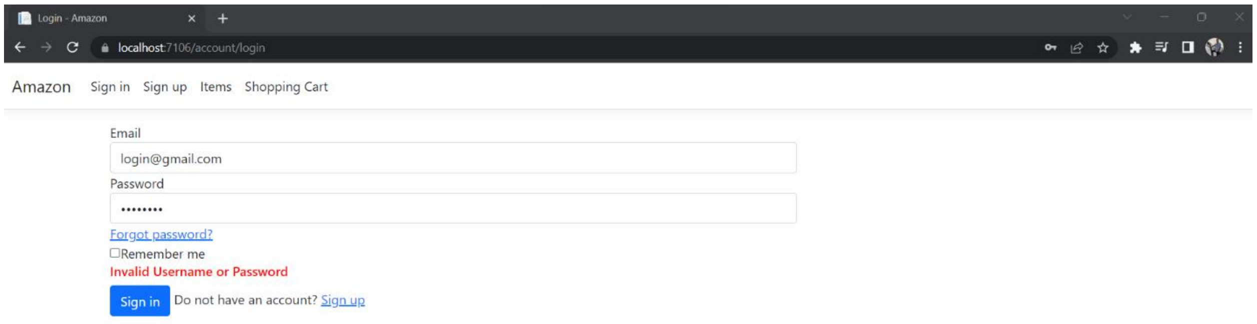
We have a lot of users who signed up for our project and we have written SQL query to their details.



```
SELECT *
FROM [dbo].AMZ_USER
```

	USER_ID	USER_ROLE_ID	USER_NME	USER_EMAIL
	1	1	Sumanth Kumar Gogineni	sugogine@gmail.com
2	1004	3	Ramu Sanga	ramu_sanga@gmail.com
3	1005	1	Raveendra Chadalawada	ravi_123@gmail.com
4	1006	3	Yeshwanth	naga_yeswanth@gmail.com
5	1007	3	Puneeth	puneeth_alla@gmail.com
6	1008	3	Vinay	vinaybasa@gmail.com

Validation: If a new user login into the application without creating an account. The system will not allow users to login.



Login - Amazon

localhost:7106/account/login

Amazon Sign in Sign up Items Shopping Cart

Email

login@gmail.com

Password

••••••••

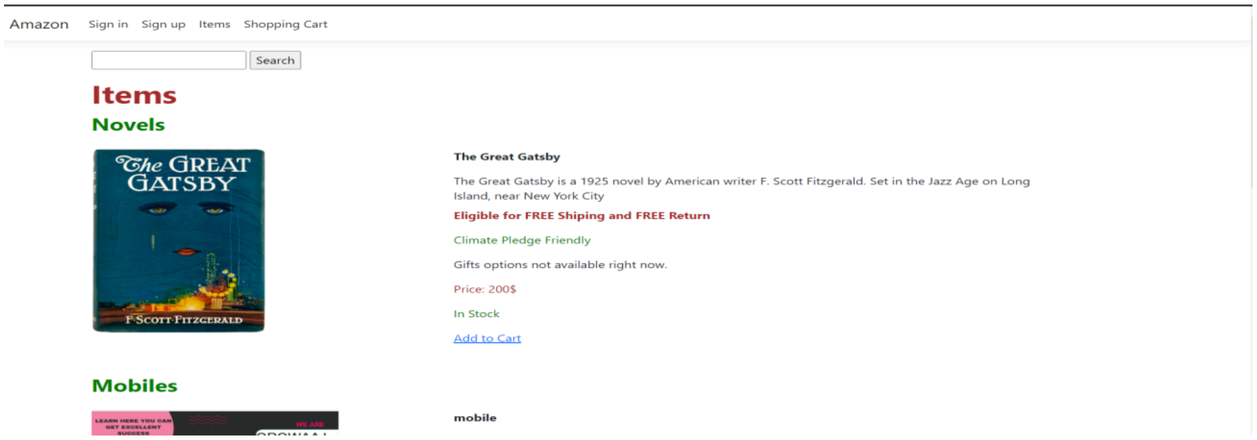
Forgot password?

Remember me

Invalid Username or Password

Sign in Do not have an account? Sign up

Items Dashboard: Items dashboard display all the items available in the system.



Amazon Sign in Sign up Items Shopping Cart

Search

Items

Novels

The Great Gatsby

The Great Gatsby is a 1925 novel by American writer F. Scott Fitzgerald. Set in the Jazz Age on Long Island, near New York City

Eligible for FREE Shiping and FREE Return

Climate Pledge Friendly

Gifts options not available right now.

Price: 200$

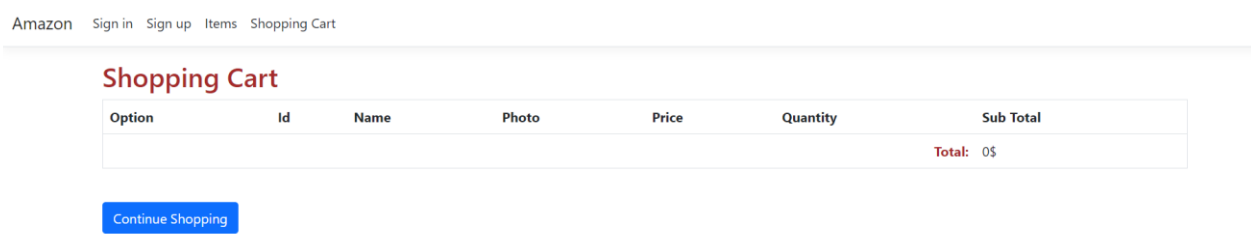
In Stock

Add to Cart

Mobiles

mobile

Cart Dashboard: Cart dashboard allows the consumer to review the products and delete them from the cart before making the order.



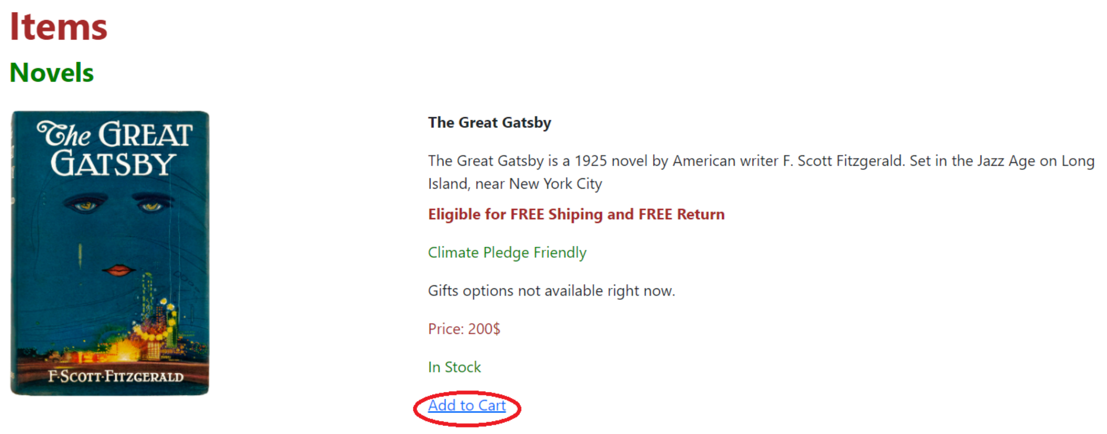
Amazon Sign in Sign up Items Shopping Cart

Shopping Cart

Option	Id	Name	Photo	Price	Quantity	Sub Total
					Total:	0$

Continue Shopping

Add product to cart: Option to add products to the cart.



Items

Novels

The Great Gatsby

The Great Gatsby is a 1925 novel by American writer F. Scott Fitzgerald. Set in the Jazz Age on Long Island, near New York City

Eligible for FREE Shiping and FREE Return

Climate Pledge Friendly

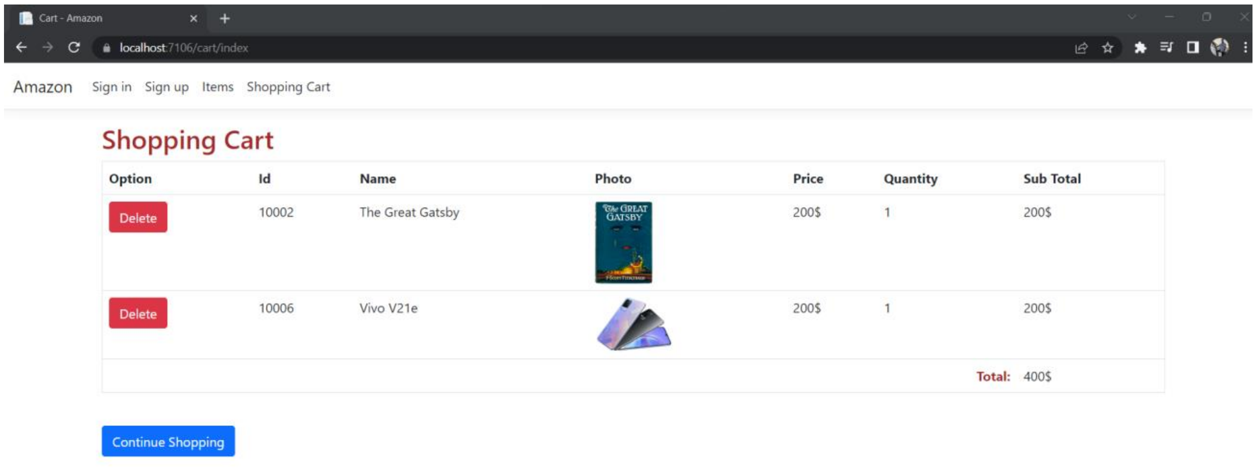
Gifts options not available right now.

Price: 200$

In Stock

Add to Cart

Cart Page: Below screenshot shows products that are added to the cart by the customer.

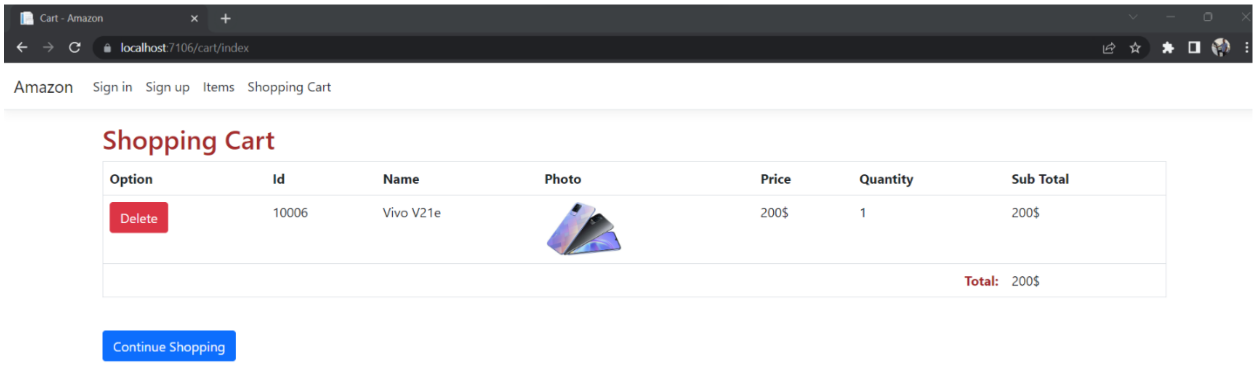

Cart - Amazon
localhost:7106/cart/index
Amazon Sign in Sign up Items Shopping Cart

Shopping Cart

Option	Id	Name	Photo	Price	Quantity	Sub Total
Delete	10002	The Great Gatsby		200$	1	200$
Delete	10006	Vivo V21e		200$	1	200$
					Total:	400$

Continue Shopping

Delete option available in cart page to delete the products before pacing the order

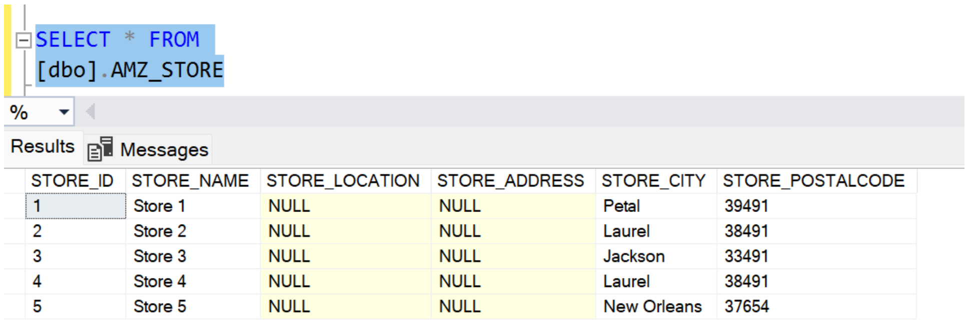

Cart - Amazon
localhost:7106/cart/index
Amazon Sign in Sign up Items Shopping Cart

Shopping Cart

Option	Id	Name	Photo	Price	Quantity	Sub Total
Delete	10006	Vivo V21e		200$	1	200$
					Total:	200$

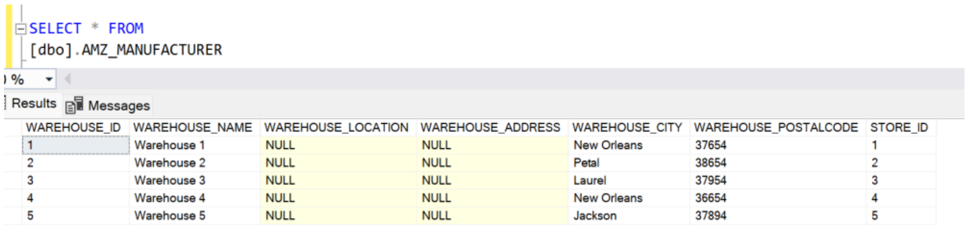
Continue Shopping

We have two different tables in our schema which is helpful for reordering the stock when stock is finished in certain stores.


```
SELECT * FROM
[dbo].AMZ_STORE
```

Results Messages

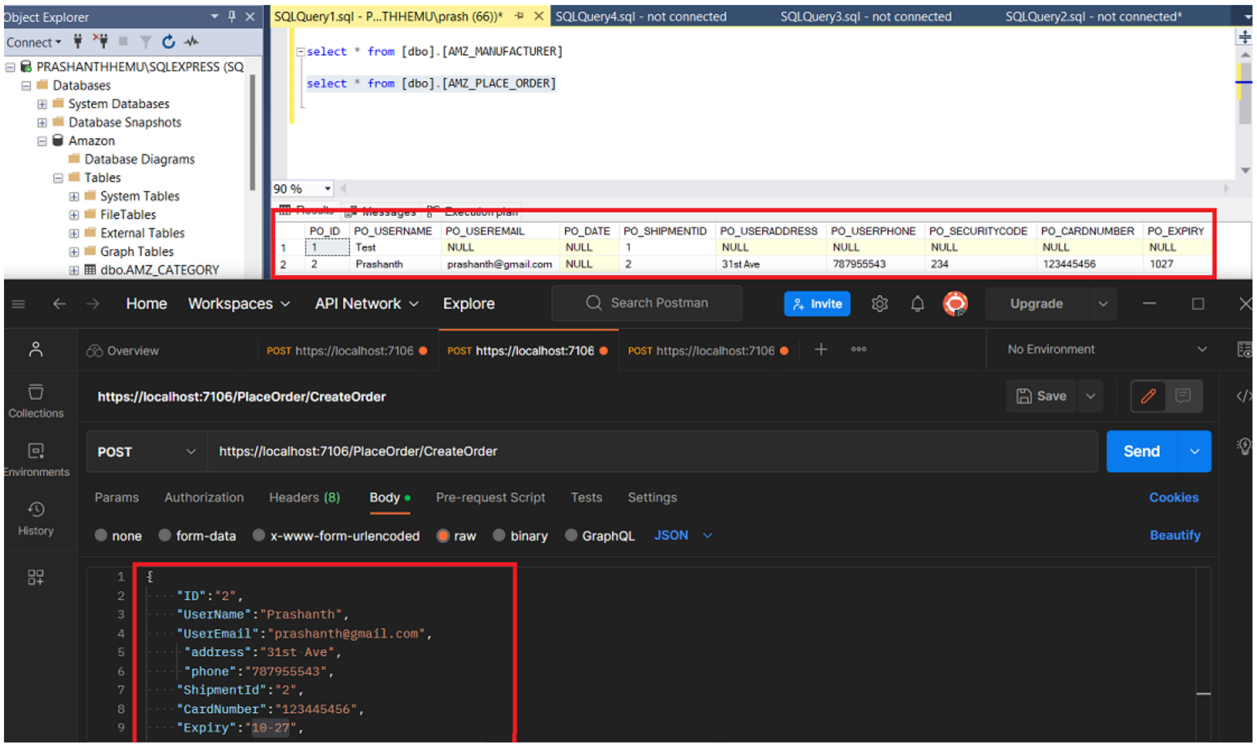
	STORE_ID	STORE_NAME	STORE_LOCATION	STORE_ADDRESS	STORE_CITY	STORE_POSTALCODE
1	1	Store 1	NULL	NULL	Petal	39491
2	2	Store 2	NULL	NULL	Laurel	38491
3	3	Store 3	NULL	NULL	Jackson	33491
4	4	Store 4	NULL	NULL	Laurel	38491
5	5	Store 5	NULL	NULL	New Orleans	37654


```sql
SELECT * FROM
[dbo].AMZ_MANUFACTURER
```

	WAREHOUSE_ID	WAREHOUSE_NAME	WAREHOUSE_LOCATION	WAREHOUSE_ADDRESS	WAREHOUSE_CITY	WAREHOUSE_POSTALCODE	STORE_ID
	1	Warehouse 1	NULL	NULL	New Orleans	37654	1
	2	Warehouse 2	NULL	NULL	Petal	38654	2
	3	Warehouse 3	NULL	NULL	Laurel	37954	3
	4	Warehouse 4	NULL	NULL	New Orleans	36654	4
	5	Warehouse 5	NULL	NULL	Jackson	37894	5

Place Order:

- We have created an API that accepts order and store card details details of the user for future purposes.
- We have tested our API through POSTMAN and below screenshot shows the results of the API functionality and database results.
- API: https://localhost:7106/PlaceOrder/CreateOrder

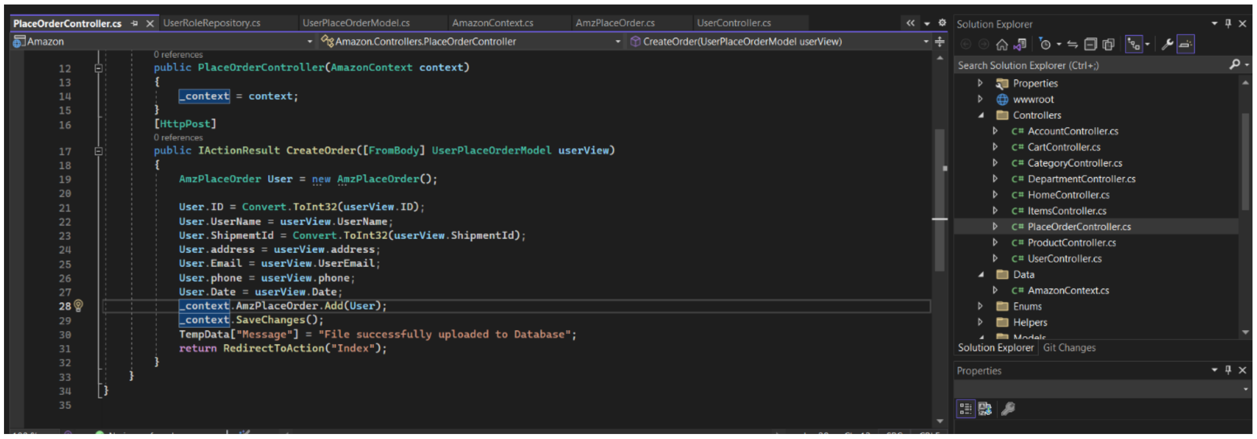

```sql
select * from [dbo].[AMZ_MANUFACTURER]

select * from [dbo].[AMZ_PLACE_ORDER]
```

	PO_ID	PO_USERNAME	PO_USEREMAIL	PO_DATE	PO_SHIPMENTID	PO_USERADDRESS	PO_USERPHONE	PO_SECURITYCODE	PO_CARDNUMBER	PO_EXPIRY
1	1	Test	NULL	NULL	1	NULL	NULL	NULL	NULL	NULL
2	2	Prashanth	prashanth@gmail.com	NULL	2	31st Ave	787955543	234	123445456	1027

POST https://localhost:7106/PlaceOrder/CreateOrder

```json
{
    "ID":"2",
    "UserName":"Prashanth",
    "UserEmail":"prashanth@gmail.com",
     "address":"31st Ave",
     "phone":"787955543",
    "ShipmentId":"2",
    "CardNumber":"123445456",
    "Expiry":"10-27",
```

API:

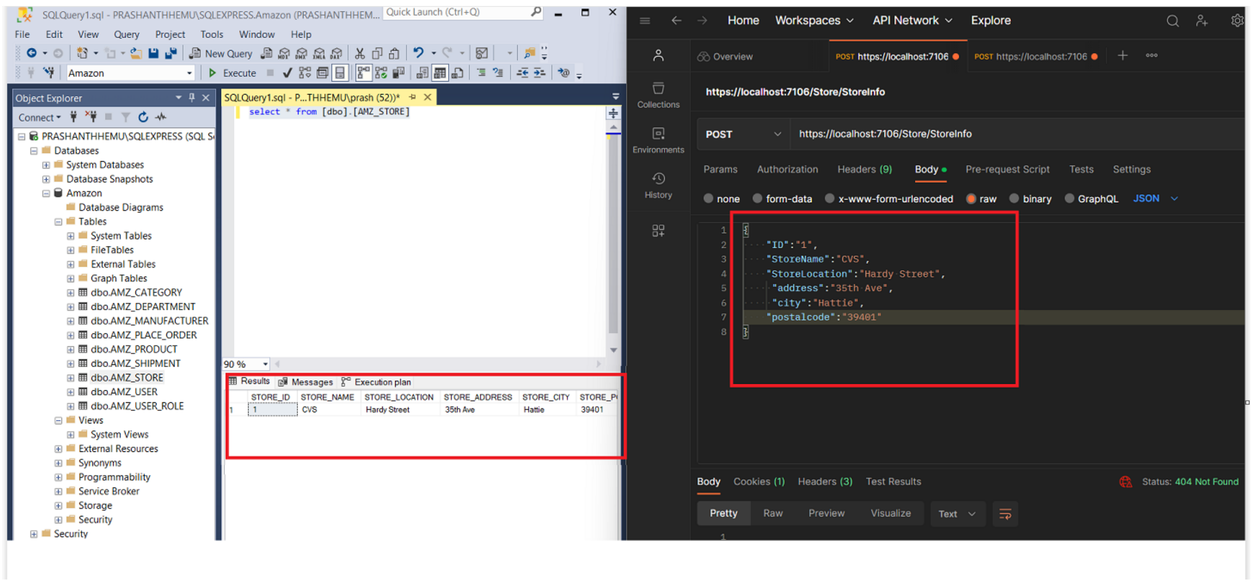


```
public PlaceOrderController(AmazonContext context)
{
    _context = context;
}
[HttpPost]
public IActionResult CreateOrder([FromBody] UserPlaceOrderModel userView)
{
    AmzPlaceOrder User = new AmzPlaceOrder();

    User.ID = Convert.ToInt32(userView.ID);
    User.UserName = userView.UserName;
    User.ShipmemtId = Convert.ToInt32(userView.ShipmentId);
    User.address = userView.address;
    User.Email = userView.UserEmail;
    User.phone = userView.phone;
    User.Date = userView.Date;
    _context.AmzPlaceOrder.Add(User);
    _context.SaveChanges();
    TempData["Message"] = "File successfully uploaded to Database";
    return RedirectToAction("Index");
}
}
}
```

Restock:

- To maintain inventory accuracy in both store and warehouse, when inventory is low in store, admin can order the required products. The manufacturer accepts the order and holds the store id and sends the shipment when the products are ready using store id.
- We have tested our API'S through POSTMAN and below screenshot shows the results of the API functionality and database results.

Store table database and API results:



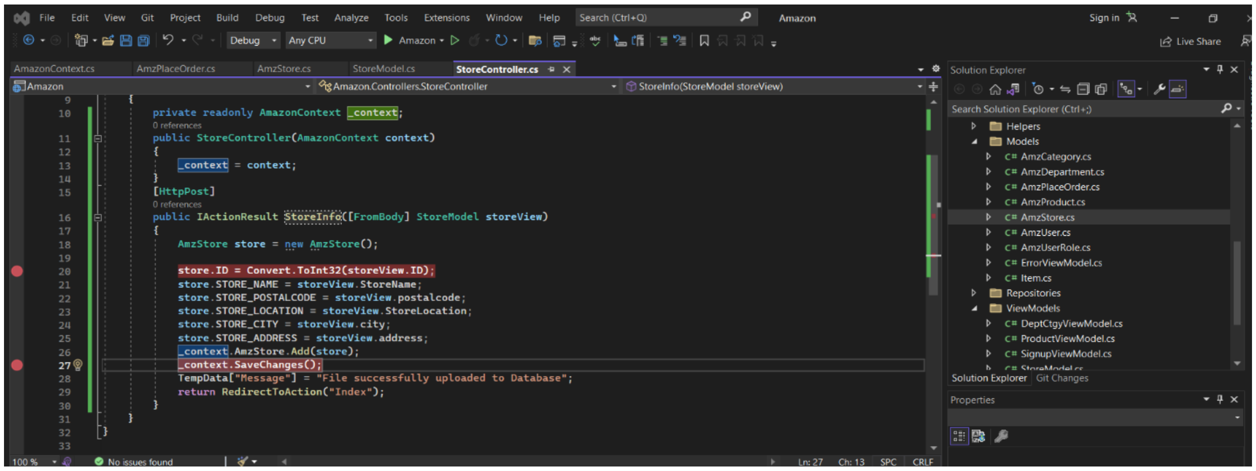
```
select * from [dbo].[AMZ_STORE]
```

	STORE_ID	STORE_NAME	STORE_LOCATION	STORE_ADDRESS	STORE_CITY	STORE_P
1	1	CVS	Hardy Street	35th Ave	Hattie	39401

POST https://localhost:7106/Store/StoreInfo

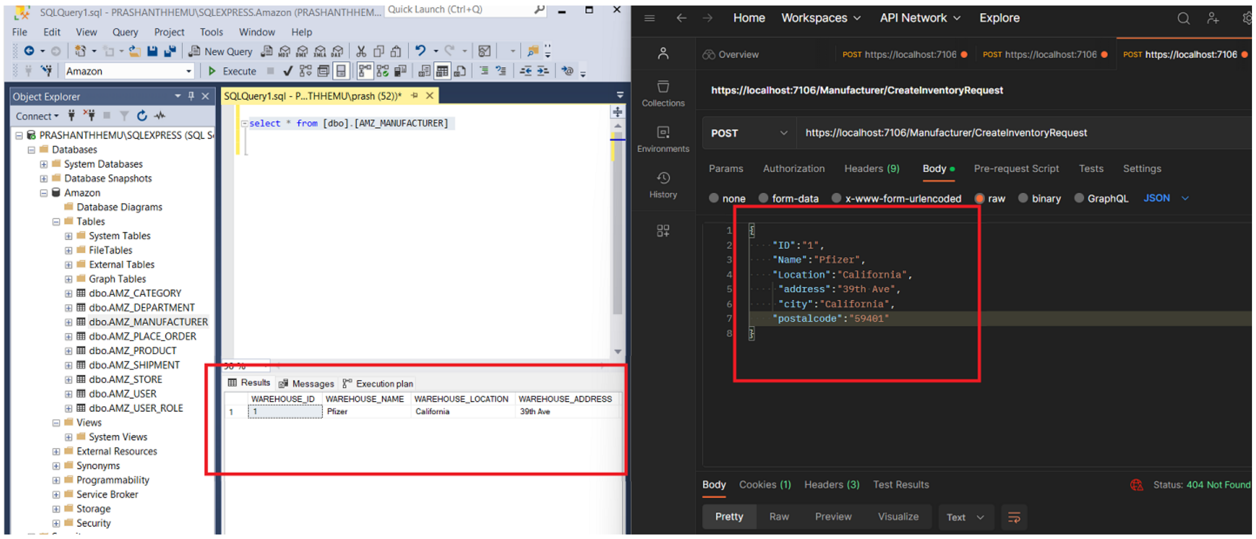
```
{
    "ID":"1",
    "StoreName":"CVS",
    "StoreLocation":"Hardy Street",
     "address":"35th Ave",
     "city":"Hattie",
    "postalcode":"39401"
}
```

Status: 404 Not Found



```
        {
            private readonly AmazonContext _context;
            0 references
            public StoreController(AmazonContext context)
            {
                _context = context;
            }
            [HttpPost]
            0 references
            public IActionResult StoreInfo([FromBody] StoreModel storeView)
            {
                AmzStore store = new AmzStore();

                store.ID = Convert.ToInt32(storeView.ID);
                store.STORE_NAME = storeView.StoreName;
                store.STORE_POSTALCODE = storeView.postalcode;
                store.STORE_LOCATION = storeView.StoreLocation;
                store.STORE_CITY = storeView.city;
                store.STORE_ADDRESS = storeView.address;
                _context.AmzStore.Add(store);
                _context.SaveChanges();
                TempData["Message"] = "File successfully uploaded to Database";
                return RedirectToAction("Index");
            }
        }
    }
```

Manufacturer tables and API results:

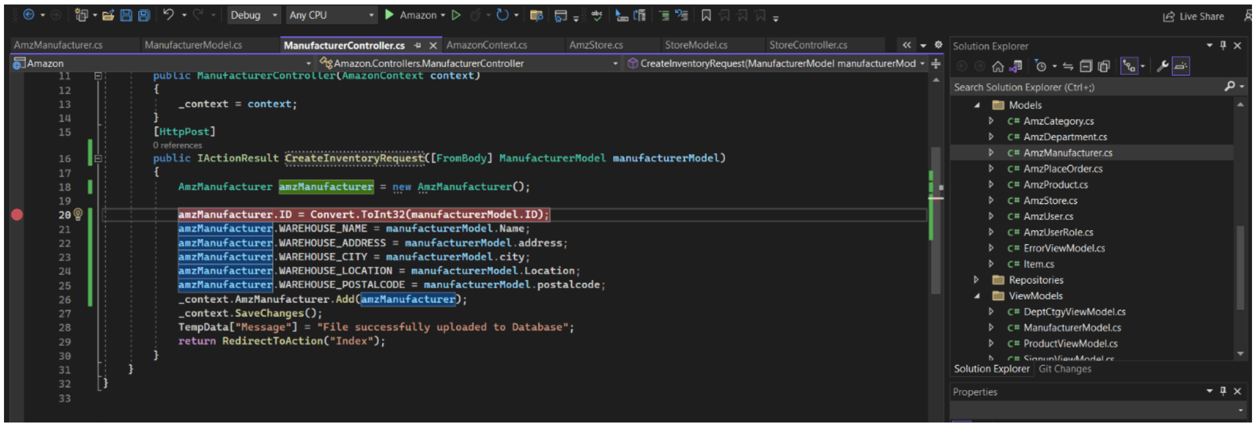


```
select * from [dbo].[AMZ_MANUFACTURER]
```

	WAREHOUSE_ID	WAREHOUSE_NAME	WAREHOUSE_LOCATION	WAREHOUSE_ADDRESS
1	1	Pfizer	California	39th Ave

```
POST https://localhost:7106/Manufacturer/CreateInventoryRequest
{
    "ID":"1",
    "Name":"Pfizer",
    "Location":"California",
     "address":"39th Ave",
     "city":"California",
    "postalcode":"59401"
}
```

Status: 404 Not Found



```
            public ManufacturerController(AmazonContext context)
            {
                _context = context;
            }
            [HttpPost]
            0 references
            public IActionResult CreateInventoryRequest([FromBody] ManufacturerModel manufacturerModel)
            {
                AmzManufacturer amzManufacturer = new AmzManufacturer();

                amzManufacturer.ID = Convert.ToInt32(manufacturerModel.ID);
                amzManufacturer.WAREHOUSE_NAME = manufacturerModel.Name;
                amzManufacturer.WAREHOUSE_ADDRESS = manufacturerModel.address;
                amzManufacturer.WAREHOUSE_CITY = manufacturerModel.city;
                amzManufacturer.WAREHOUSE_LOCATION = manufacturerModel.Location;
                amzManufacturer.WAREHOUSE_POSTALCODE = manufacturerModel.postalcode;
                _context.AmzManufacturer.Add(amzManufacturer);
                _context.SaveChanges();
                TempData["Message"] = "File successfully uploaded to Database";
                return RedirectToAction("Index");
            }
        }
    }
```

8. Conclusion

Developing e-commerce web application from the scratch to the product helped us to understand the process of developing the real-time application, integrating the various components such as back-end communication with database layer, front-end communication with business layer and finally, populating the data on the user interface pages. As part of the development of this project, we got great exposure to the core programming language concepts such as OOPS, and SOLID principles. While designing the database, we learned about the concept called entity framework to establish connect and communicate with database. We also learned about the importance of designing ER diagram and Relational diagram before starting the development of database tables and view.

9. Contribution

S. No	Student	Contribution
1	Sumanth	Created UI for the application and included UI results, implementation details in the document.
2	Prashanth	Developed API for the application and included API results, implementation details in the document.
3	Rajya Lakshmi	Design Database for the application, included ER diagram.
4	Saiteja	Performed unit and functionality testing of the application and included abstract.
5	Sohail	Worked on problem statement and relation schema.

10. Reference

- https://www.w3schools.com/sq
- https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0&tabs=visual-studio
- https://www.w3schools.com/html/html_scripts.asp