

CSCI 765: Project On Inventory Management System of Online Grocery Store

Sharmin Hossain- Student ID: 1337949

09 May 2022

Abstract

An online grocery system is either a brick-and-mortar supermarket or grocery store that allows online ordering, or a standalone e-commerce service that includes grocery items. There is usually a delivery charge for this service. The online ordering is made through e-commerce websites or mobile apps.

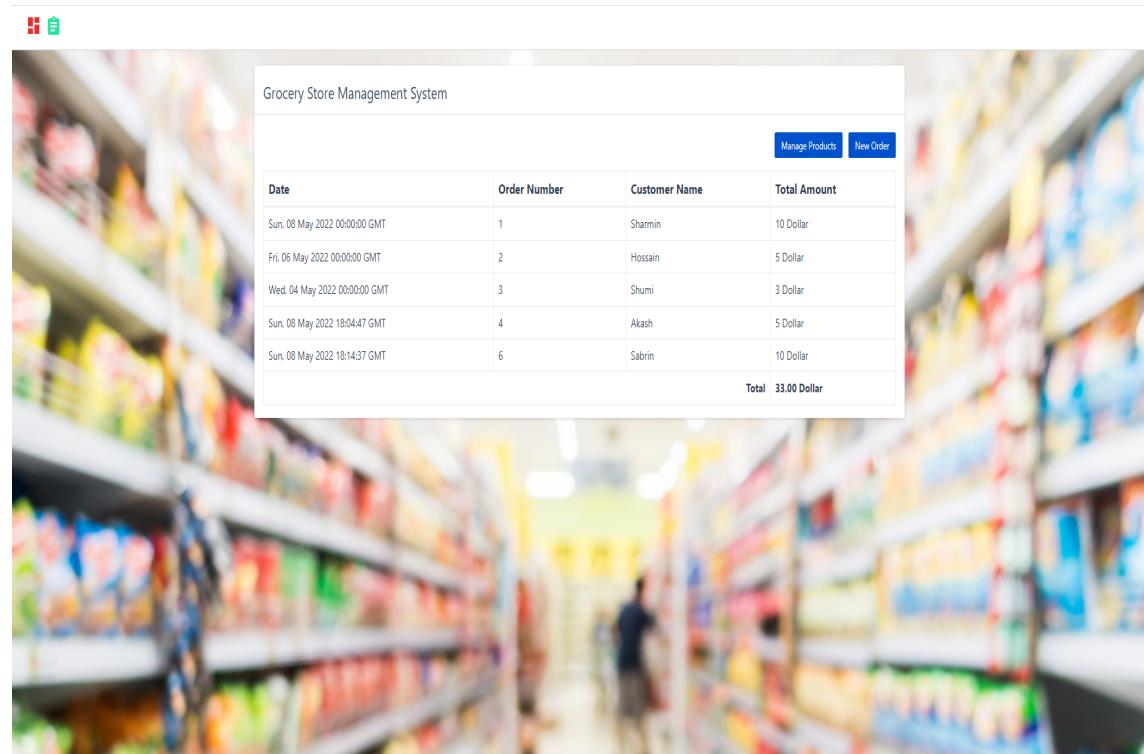
The COVID-19 pandemic greatly accelerated the growth of online grocers, and in the first few months of the pandemic's online grocery shopping increased by 300%. In addition, first-time online grocery shoppers accounted for 41% of online grocery shoppers. The epidemic of COVID-19 has hastened the uptake of online grocery shopping. Pre-COVID-19 food shopping activity accounted for 9% of the market, but 63 percent of consumers worldwide purchased more groceries online after the outbreak than they did before they were socially isolated.

Introduction

Initially my plan was to create a python application about a Inventory management system of a online grocery store which can connect to a MySQL database system (created from scratch) and display operation related pages on the web. E-commerce is getting popular in my country (Bangladesh) but the existing applications are not much developed for the southasian needs and demands yet. That motivates me to choose this idea as my project.

The existing applications are doing great in their respective fields but to work with people like from my country, the applications need to be more user friendly and less complicated. As i have worked with e-commerce system previously, i have a clear idea what higher management demands from our system.

Here is an overview of what my dashboard page, I have only included the basic features here so it can be operationally more useful.



Date	Order Number	Customer Name	Total Amount
Sun. 08 May 2022 00:00:00 GMT	1	Sharmin	10 Dollar
Fri. 06 May 2022 00:00:00 GMT	2	Hossain	5 Dollar
Wed. 04 May 2022 00:00:00 GMT	3	Shumi	3 Dollar
Sun. 08 May 2022 18:04:47 GMT	4	Akash	5 Dollar
Sun. 08 May 2022 18:14:37 GMT	6	Sabrin	10 Dollar
Total			33.00 Dollar

Figure 1: Dashboard Page

Course Relevance

Topic 1: Enhanced Entity Relationship (EER Modeling)

ER model stands for an Entity-Relationship model. It is used to define the data elements and relationship for a specified system. It develops a very simple and easy to design view of data. EER is a high-level data model that incorporates the extensions to the original ER model. The Extended Entity-Relationship (EER) model is a conceptual (or semantic) data model, capable of describing the data requirements for a new information system in a direct and easy to understand graphical notation.

I have EER modeling for my database "grocery". I did not create the model by hand. As I am using MYSQL, after creating tables, the ER model was automatically created based on my Primary & Foreign key constraints. As I am making a grocery app, its important to have multiple order, product & delivery related tables for my database. And these tables will relate to each other to make the connection for future querying. My EER model has 'one to one' relation, 'one to many' relation as well as 'many to many' relations.

Topic 2: Data Definition Language & Data Manipulation Language

Data definition language is a syntax for creating and modifying database objects such as tables, indices, and users. Data manipulation is the process of changing or altering data in order to make it more readable and organized.

I have created tables using "Create Table" option of SQL. Inserted values in the table using "Insert Into Values" option. Added multiple data types (INT, DOUBLE, DATETIME, VARCHAR) in the tables. From the tables, I have made so far, I can easily fetch some data by writing query using join keyword which belongs to the "Data Manipulation" part. I can also change column name of a table in SQL.

Topic 3: Applications and Security

The Python programming language has powerful features for database programming. Python supports various databases like SQLite, MySQL, Oracle, Sybase, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements.

For programmatically accessing database, I have used "Python". To connect with database, I have used "mysql.connector.connect" function. Imported "datetime" and "mysql.connector" packages so the relevant functions work properly. Had to use Java script and HTML to make the frontend of the application.

Implementation

Figure 1: Enhanced Entity Relationship (EER Modeling)

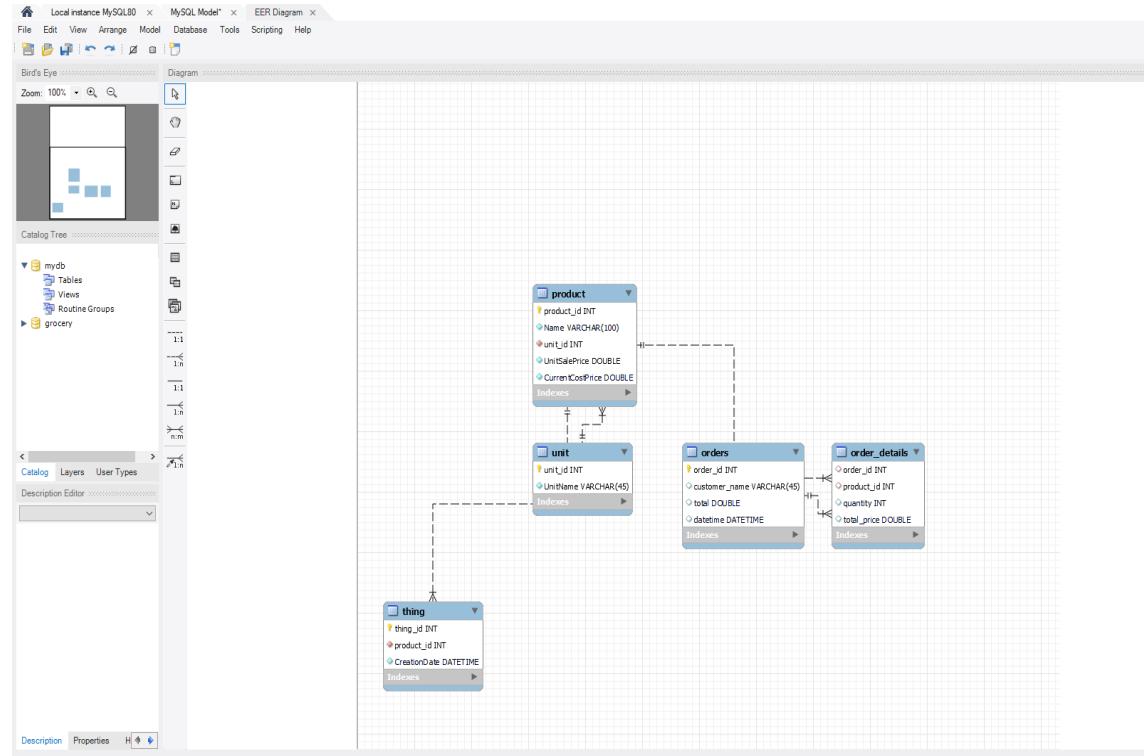


Figure 2: EER Diagram

Figure 2: Data Definition Language & Data Manipulation Language

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema 'grocery' containing tables like 'order', 'product', and 'unit'.
- Query Editor:** Displays the DDL code for creating the 'product' table:

```

1 CREATE TABLE `product` (
2     `product_id` int NOT NULL AUTO_INCREMENT,
3     `Name` varchar(100) NOT NULL,
4     `unit_id` int NOT NULL,
5     `UnitSalePrice` double NOT NULL,
6     `CurrentCostPrice` double NOT NULL,
7     PRIMARY KEY (`product_id`),
8     KEY `fk_unitid_idx` (`product_id`,`unit_id`),
9     KEY `fk_unitid_idx1` (`unit_id`),
10    CONSTRAINT `fk_unitid` FOREIGN KEY (`unit_id`) REFERENCES `unit` (`unit_id`) ON UPDATE RESTRICT
11 ) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```
- Information Schema:** Shows the 'product' table definition with columns: product_id, Name, unit_id, UnitSalePrice, CurrentCostPrice.
- Action Output:** Lists the history of actions performed on the table:
 - 2 10:43:52 SELECT * FROM grocery.product LIMIT 0, 1000 (4 rows returned)
 - 3 10:47:57 SELECT * FROM grocery.product LIMIT 0, 1000 (13 rows returned)
 - 4 10:48:14 Select count(*) from product LIMIT 0, 1000 (1 row(s) returned)
 - 5 11:45:35 Apply changes to product (No changes detected)
 - 6 11:45:40 Apply changes to product (No changes detected)
 - 7 11:45:48 SELECT * FROM grocery.product LIMIT 0, 1000 (13 rows returned)

Figure 3: Create Table

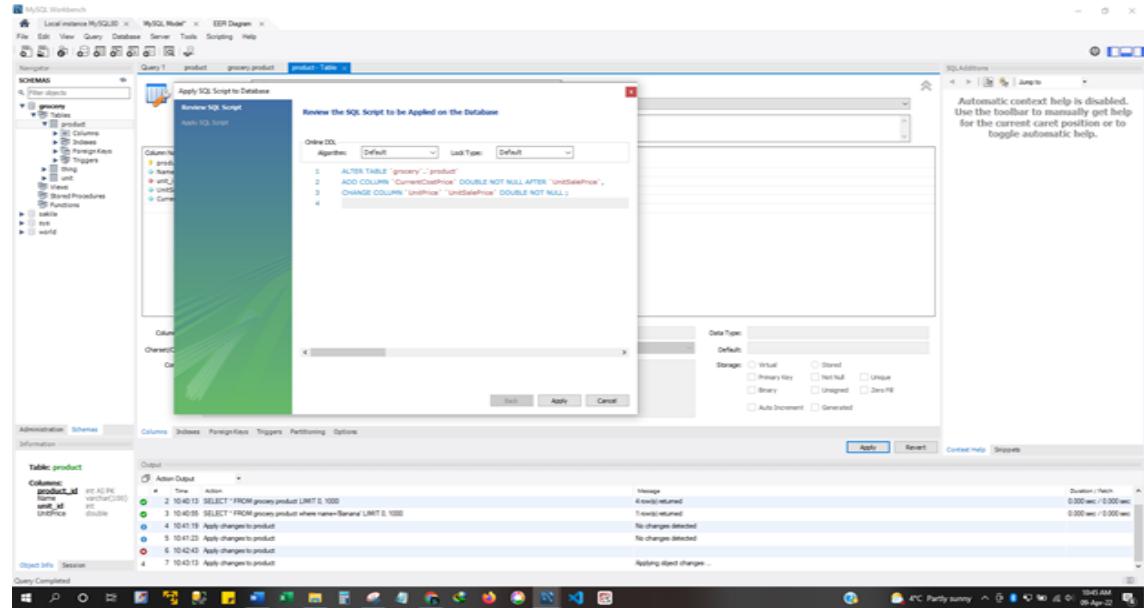


Figure 4: Alter Table

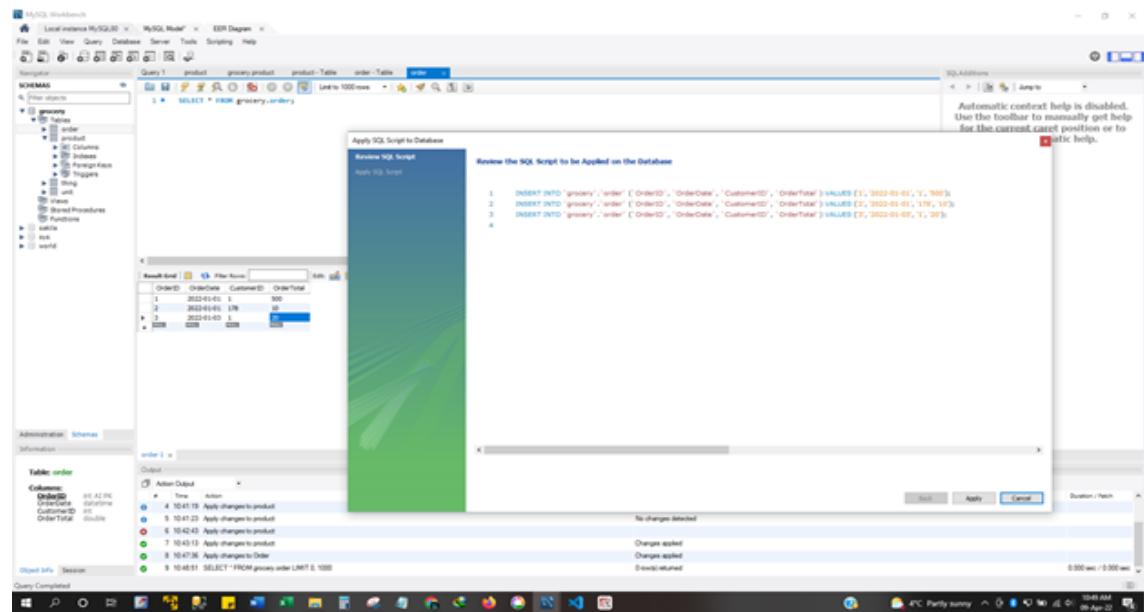
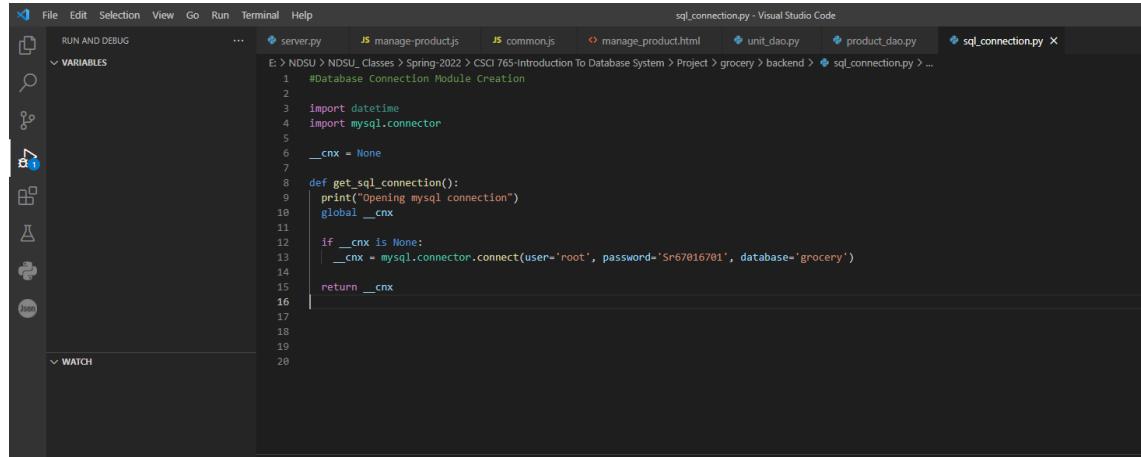


Figure 5: Insert values into tables

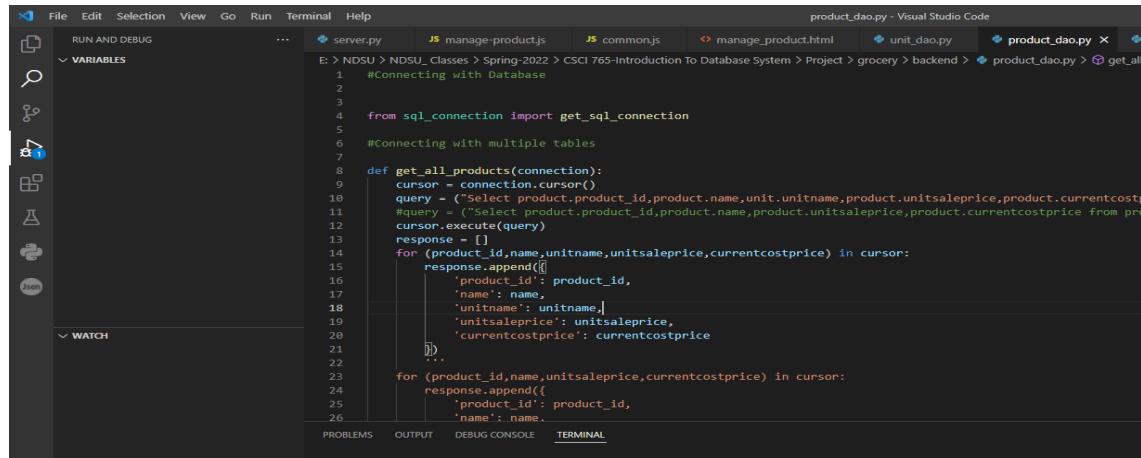
Figure 3: Applications and Security



A screenshot of Visual Studio Code showing the `sql_connection.py` file. The code is a Python script for creating a database connection module. It imports `datetime` and `mysql.connector`, defines a global variable `_cnx`, and a function `get_sql_connection` that connects to a MySQL database using the provided credentials.

```
#!/usr/bin/env python
# Database Connection Module Creation
import datetime
import mysql.connector
_global __cnx = None
def get_sql_connection():
    print("Opening mysql connection")
    global __cnx
    if __cnx is None:
        __cnx = mysql.connector.connect(user='root', password='5re7016701', database='grocery')
    return __cnx
```

Figure 6: SQL Connection



A screenshot of Visual Studio Code showing the `product_dao.py` file. The code is a Python script for connecting to a database and retrieving all products. It imports `get_sql_connection` from `sql_connection` and defines a function `get_all_products` that executes a query to select product details like name, unitname, unitsaleprice, and currentcostprice.

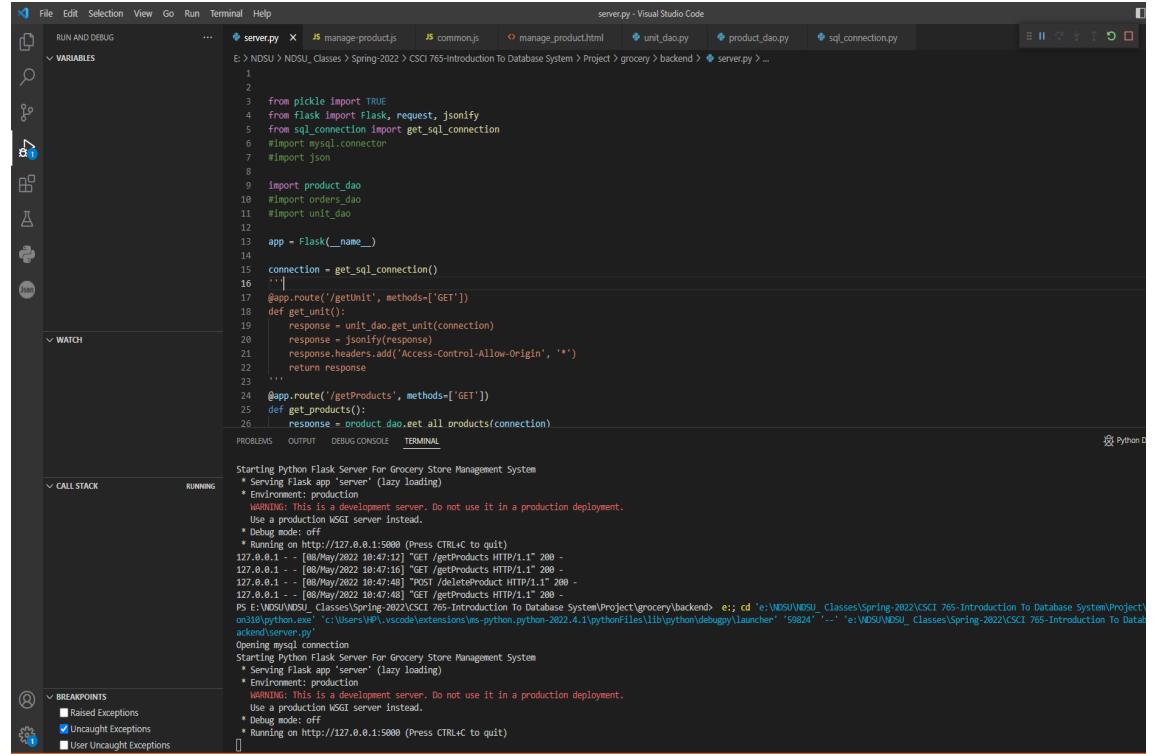
```
#Connecting with Database
from sql_connection import get_sql_connection
#Connecting with multiple tables
def get_all_products(connection):
    cursor = connection.cursor()
    query = ("Select product.product_id,product.name,unit.unitname,product.unitsaleprice,product.currentcostprice from product inner join unit on product.unit_id=unit.unit_id")
    cursor.execute(query)
    response = []
    for (product_id,name,unitname,unitsaleprice,currentcostprice) in cursor:
        response.append({
            'product_id': product_id,
            'name': name,
            'unitname': unitname,
            'unitsaleprice': unitsaleprice,
            'currentcostprice': currentcostprice
        })
    for (product_id,name,unitsaleprice,currentcostprice) in cursor:
        response.append({
            'product_id': product_id,
            'name': name,
```

Figure 7: SQL Connection

Result

Here are the final two pages (Product Manage & Order Pages) of my project. I have also included the server page code screenshot to show how its working.

In this project, i have included features like adding products, deleting products, adding new orders, removing products from orders, creating a dashboard based on inputs.



The screenshot shows the Visual Studio Code interface with the 'server.py' file open in the editor. The terminal tab at the bottom displays the output of the Python Flask application. The application is serving requests for products and units. The terminal output includes:

```
Starting Python Flask Server For Grocery Store Management System
* Serving Flask app 'server' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
127.0.0.1 - [08/May/2022 10:47:12] "GET /getUnit HTTP/1.1" 200 -
127.0.0.1 - [08/May/2022 10:47:16] "GET /getProducts HTTP/1.1" 200 -
127.0.0.1 - [08/May/2022 10:47:48] "POST /deleteProduct HTTP/1.1" 200 -
127.0.0.1 - [08/May/2022 10:47:48] "GET /getProducts HTTP/1.1" 200 -
PS E:\NDSU\NDSU Classes\Spring-2022\CSCI 765-Introduction To Database System\Project\grocery\backend> e; cd "e:\NDSU\NDSU Classes\Spring-2022\CSCI 765-Introduction To Database System\Project\on31\python.exe" < 'C:\Users\V\P\vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debug\launcher' '59824' '--' 'e:\NDSU\NDSU Classes\Spring-2022\CSCI 765-Introduction To Database System\Project\grocery\backend\server.py'
```

Figure 8: Running the local server via Python

Name	Unit	UnitSalePrice	CurrentCostPrice	Action
potatoes	Each	0.1	0.005	<button>Delete</button>
brinjals	Each	0.6	0.2	<button>Delete</button>
egg	Each	0.12	0.1	<button>Delete</button>
malta	Each	0.3	0.1	<button>Delete</button>
carrot	Each	0.6	0.4	<button>Delete</button>
tomato	Each	0.7	0.6	<button>Delete</button>
Banana	Kg	0.3	0.1	<button>Delete</button>
Orange	Kg	0.7	0.5	<button>Delete</button>
watermelon	Kg	0.4	0.2	<button>Delete</button>
pineapple	Kg	0.5	0.3	<button>Delete</button>
white chestnut	Kg	0.7	0.5	<button>Delete</button>
cucumber	Kg	0.8	0.3	<button>Delete</button>
pear	Kg	0.3	0.1	<button>Delete</button>

Figure 9: Manage Product Page

New Order

Product	Price	Quantity	Total	Add More
carrot	0.6	1	0.60	<button>Remove</button>
		Total	0.60	<button>Save</button>

Figure 10: New Order Page

Conclusions

I wanted to make a inventory management system for online grocery store. Here i have created two basic pages (Product & Order Page) for this project. Future work can be done by implementing other pages and by creating a user friendly operational environment for non-tech people and also for management. Codes used for this project are attached with a different file.

References

1. <https://www.sciencedirect.com/science/article/pii/S1567422318300085>
2. <https://ieeexplore.ieee.org/abstract/document/9489108>