

# Ecommerce

**Roll Number: IMT2021072    Name: Ramsai Koushik**

---

## Instructions to run the code:

### Server side:

```
cc -o server main_server.c server.c
```

```
./server
```

### Client side:

```
cc -o client main_client.c client.c
```

```
./client
```

## Use Cases:

### Login\Register:

- **On the Client side, users can login as an admin or register/login as a customer.**
- To register the user has to provide **a name, email address, password**. From next time when the user wants to login, the user has to provide the **same email address and password**.

## Admin use cases:

### A. Add a Product:

- Admin can offer to sell a product through the ecommerce platform, which can later be purchased by customers.
  - Admin has to provide **product name, product price and product quantity** to add the product.
-

- 
- Admin has to provide valid details of a product. By valid I mean quantity and price should be a positive value, if the product details are not valid, the product will not be added.

#### **B. Update a Product:**

- Admin can update an existing product through the ecommerce platform.
- Admin has to provide the **productId,product price and product quantity** of the product to be updated.
- Admin can update the values of quantity and price to valid values. By valid, I mean the updated quantity or price should be a positive value,if the values are not valid the product will not be updated.

#### **C. Remove a Product:**

- Admin can remove an existing product through the ecommerce platform.
- Admin has to provide the **productId** of the product to be removed.
- Admin should provide the productId of the product to be removed, if the product exists, the product will be removed and will no longer be available for any of the customers to buy it. Else there will be no effect.

#### **D. List all Products:**

- Admin can list all the products he is offering at the moment, all the products along with their details will be displayed.
- Admin will just have to select this **choice**.

### **Customer Use Cases:**

#### **E. List all Products:**

- A Customer can list all the products that are available to add to his cart.
- A Customer should just have to select this **choice**.

#### **F. Add a product to cart:**

- A Customer can add a product to his cart, which can be later purchased.
- A Customer should provide the **productId** of the product and **quantity** he/she would like to add to the cart.

- 
- If the product exists and there is sufficient quantity to add to cart, the product will be added to cart, else the product will not be added to cart.

#### **G. Update a product in Cart:**

- A Customer can update the quantity of an existing product in the cart.
- Customer should provide the **productId** of the product and **quantity** he/she would like to update the product quantity in cart.
- If the product exists and there is sufficient quantity to add to cart, the product will be updated in the cart, else the product will not be updated in cart.

#### **H. Remove a Product from Cart:**

- A customer can remove a product from their cart.
- The Customer should provide the **productId** of the product they would like to remove from their cart
- If the product is in the cart, it will be removed. Else there will be no effect

#### **I. Display Cart Items:**

- A customer can choose to list all the products in his cart.
- The customer should just have to select this **choice**.

#### **J. Purchase Cart:**

- A customer can choose to purchase all the products in their cart.
- The customer should just have to select this **choice**.
- The customer will be shown the cost of each product in the cart, and the total bill, when the customer enters the total amount, a receipt will be generated will have the following details:
  - Receipt Number
  - Customer Name
  - List of all the products bought
  - Total amount paid to buy these products.

### **Approach of implementation:**

**As we see the structs used in the code in the below image, the approach of above functionalities is explained below:**

---

An overview of how customer functionalities were implemented:

The file customers.txt stores all the structs of customer, which includes the Cart array, the cart array is an array of Cart Items, where each cart Item stores the productId and the quantity of the product. So when a user wants to remove/update a product in cart or add a product to cart we only modify the User's "cart array". When the user wants to see all the products in the cart, the code iterates through the cart array and calls the function ProductById, which takes the product ID as an argument and returns the product (which has all the details), the code stores all these products in an array and passes to the client side. Similarly for

Purchase cart, the same process as display is followed in the server side, but the server has to update each product quantity available, because these products are being bought, this will be done by using the function

```
typedef struct person {
    char name[50];
    char emailaddress[50];
    char password[50];
} person;

typedef struct cartItem{
    int productId;
    int quantity;
} cartItem;

typedef struct customer{
    int customerId;
    person details;
    cartItem cart[50];
    int assigned;
} customer;

typedef struct product{
    int productId;
    char productName[50];
    int productPrice;
    int productQuantity;
} product;

typedef struct result{
    int res;
    char reason[50];
} result;
```

---

**“BuyProductById”, which takes product id and quantity that should be bought as parameters and updates the product quantity in the “products.txt” file, then the client takes all the products and lists them to the user to pay the money, after which all these products are written to a .txt file to create a receipt. The customer struct also stores an instance of “struct person” which only has the details of the user, these details are used for login purposes, when the user wants to login the user has to provide the email address and password which will be checked against the ones stored in the file “customers.txt”.**

**An overview of how admin functionalities were implemented:**

**The file “products.txt” stores all the structs of products, which contains the details of a product like name,price,quantity available.Initially before running the code, a large number of empty product struct’s are written, where the productId is set from 1 to 1500 and productQuantity is set to -1(This is to indicate that record is empty) before writing to the “products.txt”.**

**So when a user wants to add a product, the code checks if the data is valid, and searches for an empty record and writes the product at that location in products.txt. To update/ remove a product the code checks if there exists a product with the given productId and if it exists it will perform the operations and the data in “products.txt” is updated , else there will be no change in the file “products.txt” and the user will be informed about whether the query is processed successfully or not. When the Admin logs out, a log file is created which contains all the list of products. To create this log file, a list of all products is sent to the client side, and the client creates a log file. This log file is named on the basis of the current date and time.**

**A brief description of all functions implemented in server.c :**

- **UserMenu:**

- 
- This function is used for login/register of a customer or login of the admin. It takes the choice from the client(whether admin or customer and if customer whether login or register) and calls the "ValidateUser " function or "Register" function based on the choice. It recursively calls itself until a successful login. It returns the userId of the logged in user.
  - **Register:**
    - This function is used for registering a customer. It checks if any other user has the same email address. If so, it will not allow the user to register a duplicate account, as every user should have a unique email address. If there isn't any user with the same email address it will add the customer to the customer.txt file and will return the user id of the customer, for further queries.
    - **Locking Mechanism:** It does **Mandatory write lock** on the "customers.txt" file as it has to iterate through all the users to check if there exists a user with the same email address.
  - **ValidateUser:**
    - This function is used for the login of a customer/admin.It takes the choice(whether admin/customer) and the login details, to know whether it is an admin/customer. For a customer login It iterates through all the records in "customers.txt" and checks if there exists an account with the given details.If there exists a matching record it returns the customerId for further queries.Else it returns 0.
    - **Locking Mechanism:** It does **Mandatory read lock** on the "customers.txt" file as it has to iterate through all the users to check if there exists a user with the same email address. and password.
  - **Menu:**
    - This function is used for taking in the choice of query from the client and calling the corresponding function to perform the query.
  - **DisplayCartItems:**
-

- 
- This function is used for displaying all the cart items. It fetches the customer record from the “customers.txt” and iterates through the cart array and calls the function **ProductById** for each product in the cart, to get all details of the product. It stores all these products and sends it to the client side, the client side then prints all these products to the terminal.
  - **Locking Mechanism:** It implements **Record locking (read)** on the “customers.txt” , it **read locks** only the record of the customer who requested to display all cart items.
  - **AddProductToCart:**
    - This function is used to add a product to the cart. It calls the function **“ProductCheck”** to check if sufficient quantity of product is available, if it is available, it continues with the process, else it will not add the product and returns. It fetches the customer record from the “customers.txt” and iterates through the cart array to check if the product is already in the cart, if so it will not add the product and sends the same to the client. If the product is not in the cart array, it adds the product to the cart. After modifying the customer, it writes the customer record back in the same location it fetched it from.
    - **Locking Mechanism:** It implements **Record Locking(write)** on the “customers.txt” file, it **write locks** only the record of the customer who wants to add the product to his cart.
  - **UpdateProductInCart:**
    - This function is used to update a product in the cart. It calls the function **“ProductCheck”** to check if sufficient quantity of product is available, if it is available, It fetches the customer record from the “customers.txt” and iterates through the cart array to check if the product is present in the cart, if so it will update the product. Else it won't make any changes to the cart and send the same to the client. It has to write the customer record back in the same location it fetched it from, after updating the customer.
    - **Locking Mechanism:** It implements **Record Locking(write)** on the “customers.txt” file, it **write locks** only the record of the customer who wants to update the product in cart.
  - **RemoveProductFromCart:**
-

- 
- This function is used to remove a product from the cart. It fetches the customer record from the “customers.txt” and iterates through the cart array to check if the product is present in the cart, if so it will remove the product by setting the quantity of cartItem to -1 ,if product not in cart it doesn't make any changes and sends the result of the operation to the client. After modifying the customer, it writes the customer record back in the same location it fetched it from.
  - **Locking Mechanism:** It implements **Record Locking(write)** on the “customers.txt” file, it **write locks** only the record of the customer who wants to add the product to his cart.
  - **ProductCheck:**
    - This function is used to check if sufficient quantity of given productId is available or not. It fetches the record corresponding to the productId from the “products.txt” and checks using if condition if sufficient quantity is available.
    - **Locking Mechanism:** It implements **Record Locking(read)** on the “products.txt” file, it **read locks** only the record of the product it wants to check.
  - **ListAllProducts:**
    - This function is used to display all the products available for purchase. It has to fetch every record from the “products.txt” and send all the products available to the client.
    - **Locking Mechanism:** It implements **Mandatory locking (read)** on the “products.txt” file, as it has to go through all the products in the “products.txt” file.
  - **ProductById:**
    - This function is used to return a “product”(which has all the details) given the productId.
    - **Locking Mechanism:** It implements **Record Locking(read)** on the “products.txt” file, it **read locks** only the record of the product it needs to return.
  - **PurchaseCart:**
-



- 
- This function is used to execute a purchase of all the products in the cart, which means it returns all the products to the client and also updates the quantities of the product bought using the function **"BuyProductById"**. It fetches the customer record corresponding to the userId, and iterates through the cart, and for each cart Item, it calls the **"BuyProductById"** function to get the details of the product and also to update the quantity of the product available in the "products.txt". After all this, it sends all the products to the client. It also clears the cart, as the products are purchased by setting all the cart Item quantities as -1. It writes the customer record after all the operations to the "customers.txt" at the same location where it was fetched from. It also calls the function **"ReceiptNumber"** to generate a receipt number for the bill.
  - **Locking Mechanism:** It implements **Record Locking(write)** on the "customers.txt" file, it **write locks** only the record of the customer who wants to purchase the products in his cart.
  - **BuyProductById:**
    - This function is used to return the product details and update the quantity of the product available given the productId, and the quantity to be bought.
    - **Locking Mechanism:** It implements **Record Locking(write)** on the "products.txt" file, it **write locks** only the record of the product it wants to update.
  - **AddProduct:**
    - This function is used to add a product to the "products.txt". It iterates through the product records to check if there is an empty record, i.e. a record can be added at that location and adds the records at the first empty record location it finds. An empty record is identified by checking the product quantity, if the product quantity is -1, it is an empty record.
    - **Locking Mechanism:** It implements **Mandatory Locking (write)** on the "products.txt" file, as it has to go through all the records of the "products.txt" to find the first empty record.
  - **UpdateProduct:**

- 
- This function is used to update a product, if the product with given productId exists. Before updating it checks if the data provided is valid, i.e , product quantity and product price are positive.
  - **Locking Mechanism:** It implements **Record Locking(write)** on the “products.txt” file, it **write locks** only the record of the product it wants to update.
  - **RemoveProduct:**
    - This function is used to remove a product, if the product with given productId exists.It fetches the product with given productId and updates the productQuantity to -1, indicating it no longer is available and can be considered as an empty record. It writes this product at the same location it is fetched from.
    - **Locking Mechanism:** It implements **Record Locking(write)** on the “products.txt” file, it **write locks** only the record of the product it wants to remove.
  - **ReceiptNumber:**
    - This function is used to read the receipt number from the file and update it with receipt number + 1 and write back to the file. This receipt number is also sent to the client for creating the receipt for purchase products .
    - **Locking Mechanism:** It implements **Record Locking(write)**, it only locks the first 4 bytes of the file “receiptNumber.txt” as it will only read an integer and write an integer.

**Functions on the client side are used for sending the choice of client, and receiving the data using “socket programming concepts”.**

**A brief description of some unique functions implemented in client.c :**

- **PurchaseCart:**
  - This function is used to execute the purchase cart operation.It reads all the products from the client and displays the same to the user. It also creates a file with the name “receiptNumber.txt” , where receipt number is sent from

---

the client, writes all these products to this file(receipt), along with the total bill amount and customer details.

- Concepts used: creating a file using "**creat**" system call, and writing data to it.

---

The following are the Os concepts used in the project:

- **Socket Programming**
- **File Locking (Record Locking and Mandatory locking)**
- **System Calls:**
  - write(to send the data to client from server and vice versa)
  - read(for client to receive the data from server and vice versa)
  - creat(for creating receipts,log files when admin logs out)
  - open
  - fcntl (for file locking)
  - Socket related system calls - socket , bind, listen, accept,connect

Screenshots of the execution of code

Registering a customer:

```
UserMenu:
1.Admin
2.Customer
3.Exit
Please provide your choice
2
1.Register
2.Login
1
name: Vijay
emailAddress: Saradhi
password: Yanu
Registration Sucessfull

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
|
```

---

## Screenshots of the execution of code

- Admin Login:

```
UserMenu:
1.Admin
2.Customer
3.Exit
Please provide your choice
1
emailaddress: admin
password: pass
Login Succesful

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
```

Add a product ,remove a product and list all products:

```
Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
1
Please provide the details of the product,i.e,name,price,quantity with a space in between them
IceCream 120 7
Product added sucessfully, product is assigned with productId:5

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
4

List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:2 , Product Name: stand , Product Price:18 , Product Quantity:14
ProductId:3 , Product Name: Bat , Product Price:7860 , Product Quantity:1
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:7
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:5
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
2
Please provide the productId of the product you would like to remove
2
Product removed sucessfully

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
4

List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:3 , Product Name: Bat , Product Price:7860 , Product Quantity:1
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:7
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:5
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100
```

---

**Update a product:(the below screenshot shows list of products before update  
update query and list of products after update)**

```
Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
4

List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:3 , Product Name: Bat , Product Price:7860 , Product Quantity:1
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:7
ProductId:6 , Product Name: specs , Product Price:1008 , Product Quantity:5
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
3
Please provide the productId along with the price and quantity you would like to update to
,i.e,id,price,quantity with a space in between them
3 100000 9
Product updated sucessfully

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
4

List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:3 , Product Name: Bat , Product Price:100000 , Product Quantity:9
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:7
ProductId:6 , Product Name: specs , Product Price:1008 , Product Quantity:5
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100
```

---

## Login of Customer, add product to cart, display cart:

```
pramsai@pramsai-IdeaPad-Slim-7-14ITL05:~/0s4th/0s-Ecommerce-Project-main$ cc -o client main_client.c client.c
pramsai@pramsai-IdeaPad-Slim-7-14ITL05:~/0s4th/0s-Ecommerce-Project-main$ ./client
Hello from server

UserMenu:
1.Admin
2.Customer
3.Exit
Please provide your choice
2
1.Register
2.Login
2
emailaddress: ram
password: may15
Login Succesful

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
1

List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:3 , Product Name: Bat , Product Price:100000 , Product Quantity:9
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:7
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:5
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
3
Please provide the productId and quantity to add the product to cart
1 45
Product added to cart sucessfully

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
2

List of cart items:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:45
```



---

**Update Product in Cart: (the below screenshot contains the list of cart items before update and query update and list of products after update )**

```
Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
2

List of cart items:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:45
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:7
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:4

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
4
Please provide the productId and quantity to update the product in cart
5 3
Product with id: 5 update in cart sucessfully

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
2

List of cart items:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:45
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:3
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:4
```

---

**Remove Product from Cart:** (the below screenshot contains the list of cart items before removing and query remove from cart and list of products after removing)

```
Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
2

List of cart items:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:45
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:3
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:4

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
5
Please provide the productId to remove the product from cart
1
Product with id: 1 removed from cart sucessfully

Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
2

List of cart items:
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:3
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:4
```

---

## PurchaseCart:

```
Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
6
1. Product Name: IceCream, Price: 120, Quantity: 3, Cost: 360
2. Product Name: spect, Price: 1008, Quantity: 4, Cost: 4032
Total Bill- 4392
Please enter the payment
4392
Receipt generated! receipt number- 1

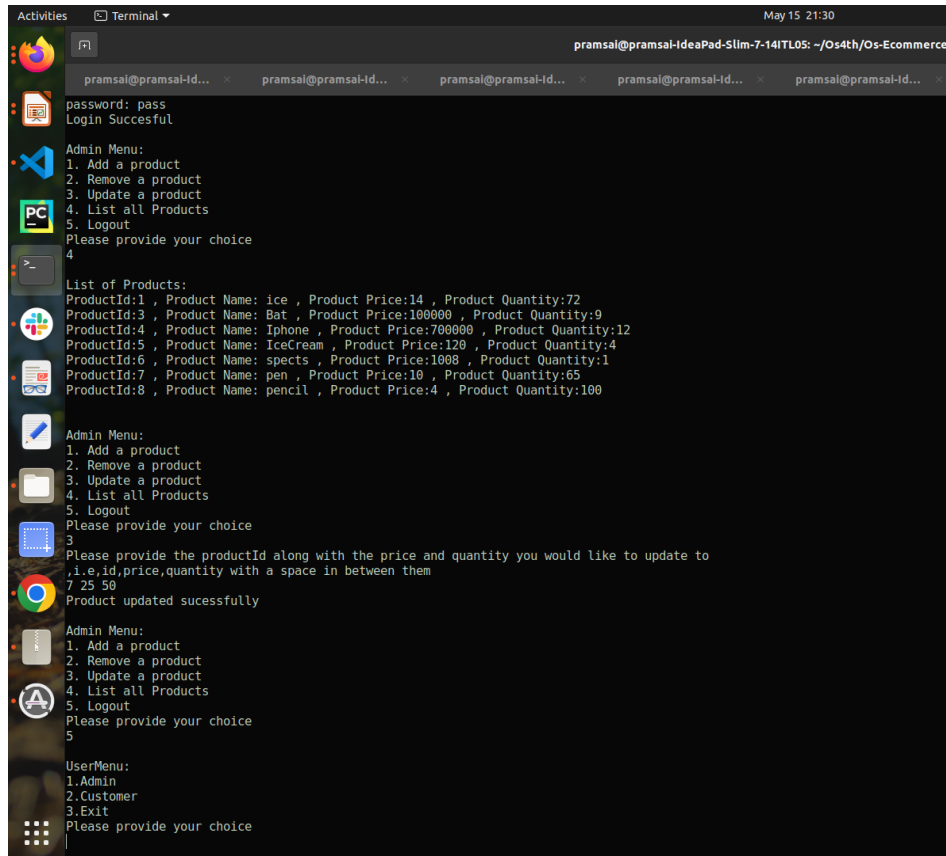
Customer Menu:
1. List all the products
2. Display cart items
3. Add product to cart
4. Update a product in cart
5. Remove a product from cart
6. Purchase Cart
7. Logout
Please provide your choice
1

List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:3 , Product Name: Bat , Product Price:100000 , Product Quantity:9
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:4
ProductId:6 , Product Name: spect, Product Price:1008 , Product Quantity:1
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100
```

## The receipt generated :

```
1.txt  X  C main_server.c  C main_client.c
Os-Ecommerce-Project-main > 1.txt
1  Receipt Number- 1
2  Customer Email: ram
3  Customer Name: Ram
4  1. Product Name: IceCream, Price: 120, Quantity: 3, Cost: 360
5  2. Product Name: spect, Price: 1008, Quantity: 4, Cost: 4032
6  Total Bill- 4392
7  |
```

The log file created when admin logs out:



The terminal window shows the following sequence of events:

```
pramsal@pramsal-IdeaPad-Slim-7-141TL0S: ~/Os4th/Os-Ecommerce-P
password: pass
Login Successful

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
4

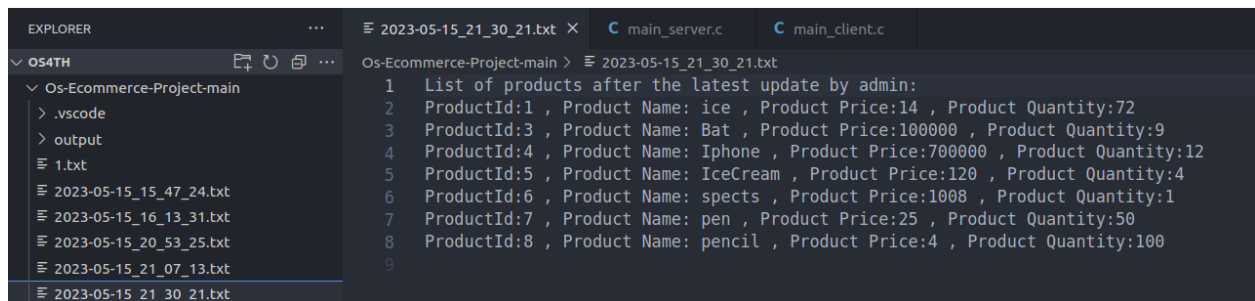
List of Products:
ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
ProductId:3 , Product Name: Bat , Product Price:100000 , Product Quantity:9
ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:4
ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:1
ProductId:7 , Product Name: pen , Product Price:10 , Product Quantity:65
ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
3
Please provide the productId along with the price and quantity you would like to update to
,i.e,id,price,quantity with a space in between them
7 25 50
Product updated sucessfully

Admin Menu:
1. Add a product
2. Remove a product
3. Update a product
4. List all Products
5. Logout
Please provide your choice
5

UserMenu:
1.Admin
2.Customer
3.Exit
Please provide your choice
```

The log file created when admin logs out, we can also see that the name of the log file is based on the current date and time:



The file explorer shows a list of log files in the 'output' directory. The files are named based on the date and time of the log entry:

- 1.txt
- 2023-05-15\_15\_47\_24.txt
- 2023-05-15\_16\_13\_31.txt
- 2023-05-15\_20\_53\_25.txt
- 2023-05-15\_21\_07\_13.txt
- 2023-05-15\_21\_30\_21.txt

The selected file, 2023-05-15\_21\_30\_21.txt, contains the following content:

```
1 List of products after the latest update by admin:
2 ProductId:1 , Product Name: ice , Product Price:14 , Product Quantity:72
3 ProductId:3 , Product Name: Bat , Product Price:100000 , Product Quantity:9
4 ProductId:4 , Product Name: Iphone , Product Price:700000 , Product Quantity:12
5 ProductId:5 , Product Name: IceCream , Product Price:120 , Product Quantity:4
6 ProductId:6 , Product Name: spectrs , Product Price:1008 , Product Quantity:1
7 ProductId:7 , Product Name: pen , Product Price:25 , Product Quantity:50
8 ProductId:8 , Product Name: pencil , Product Price:4 , Product Quantity:100
9
```