Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;

// Base class representing an item
class Item {
public:
    int id;
    string category;
    string name;
    virtual int getPrice() const = 0;  // Virtual function to get the price
    virtual void display() const = 0;  // Virtual function to display item details
};

// Derived class for clothing items
class Clothing : public Item {
public:
    int price;

    Clothing(int id, string name, int price) : price(price) {
        this->id = id;
        this->category = "Clothing";
        this->name = name;
    }

    int getPrice() const override {
        return price;
    }

    void display() const override {
        cout << "Id: " << id << "\t";
        cout << "Category: " << category << "\t";
        cout << "Item Name: " << name << "\t";
        cout << "Price: " << price << " NPR\n";
        cout << "-----------------------------------------------------------------\n";
    }
};
```

```
// Derived class for electronics items
class Electronics : public Item {
public:
    int price;

    Electronics(int id, string name, int price) : price(price) {
        this->id = id;
        this->category = "Electronics";
        this->name = name;
    }

    int getPrice() const override {
        // Add 13% VAT
        return price + static_cast<int>(0.13 * price);
    }

    void display() const override {
        cout << "Id: " << id << "\t";
        cout << "Category: " << category << "\t";
        cout << "Item Name: " << name << "\t";
        cout << "Price (including 13% VAT): " << getPrice() << " NPR\n";
        cout << "------------------------------------------------------------------------\n";
    }
};

// Derived class for groceries items
class Groceries : public Item {
public:
    int price;

    Groceries(int id, string name, int price) : price(price) {
        this->id = id;
        this->category = "Groceries";
        this->name = name;
    }

    int getPrice() const override {
        return price;
    }

    void display() const override {
        cout << "Id: " << id << "\t";
        cout << "Category: " << category << "\t";
        cout << "Item Name: " << name << "\t";
        cout << "Price: " << price << " NPR\n";
```

```
        cout << "-----------------------------------------------------------------------\n";
    }
};

void addItem(vector<Item*>& items, Item* newItem);
void addItem(vector<Item*>& items, vector<Item*>& selectedItems, Item* newItem);
void displayItems(const vector<Item*>& selectedItems, string cName, string cAddress);
void saveCustomerInfo(string cName, string cAddress, const vector<Item*>&
selectedItems);
int calculateTotal(const vector<Item*>& selectedItems);
int applyDiscount(int total);

vector<Item*> getItemsByCategory(const vector<Item*>& items, const string&
category);

int main() {
    cout << "\t\t\t**********************" << endl;
    cout << "\t\t\tWelcome to Nepal Mart" << endl;
    cout << "\t\t\tShopping gets better" << endl;
    cout << "\t\t\t**********************" << endl;
    string Name;
    string Address;

    cout << "Enter customer's name: ";
    cin >> Name;
    cout << "Enter customer's address: ";
    cin >> Address;
    cout << "\n";

    vector<Item*> items;

    // Predefined clothing items
    addItem(items, new Clothing(1, "T-Shirt", 2000));
    addItem(items, new Clothing(2, "Jeans", 3200));
    addItem(items, new Clothing(3, "Jacket", 4230));
    addItem(items, new Clothing(4, "Socks pair", 800));

    // Predefined electronics items
    addItem(items, new Electronics(1, "Smartphone", 50000));
    addItem(items, new Electronics(2, "Laptop", 150000));
    addItem(items, new Electronics(3, "Fridge", 80000));
    addItem(items, new Electronics(4, "Washing Machine", 90000));
    addItem(items, new Electronics(5, "Bulb", 2000));

    // Predefined groceries items
    addItem(items, new Groceries(1, "Bread", 50));
```

```
    addItem(items, new Groceries(2, "Milk", 120));
    addItem(items, new Groceries(3, "Rice", 3200));
    addItem(items, new Groceries(4, "Oil", 300));
    addItem(items, new Groceries(5, "Flour", 250));

    vector<Item*> selectedItems;

    int choice;

    do {
        // Display menu for category selection
        cout << "\nSelect a category:\n";
        cout << "1. Clothing\n";
        cout << "2. Electronics\n";
        cout << "3. Groceries\n";
        cout << "0. Done (Finish selecting items)\n";

        cin >> choice;

        if (choice > 0 && choice <= 3) {
            vector<Item*> currentCategoryItems = getItemsByCategory(items, (choice == 1)
? "Clothing" : (choice == 2) ? "Electronics" : "Groceries");

            // Display items in the selected category
            cout << "\nItems in the selected category:\n";
            for (const auto& item : currentCategoryItems) {
                item->display();
            }

            // Let the customer choose specific items
            int itemChoice;
            do {
                cout << "Enter the ID of the item to add (0 to finish): ";
                cin >> itemChoice;

                if (itemChoice != 0) {
                    // Find the selected item by ID
                    for (const auto& item : currentCategoryItems) {
                        if (item->id == itemChoice) {
                            addItem(items, selectedItems, item);
                            break;
                        }
                    }
                }
            } while (itemChoice != 0);
        } else if (choice != 0) {
```

```
            cout << "Invalid choice. Please enter a valid option.\n";
        }
    } while (choice != 0);

    // Display the final bill with the total price of selected items only
    displayItems(selectedItems, Name, Address);
    saveCustomerInfo(Name, Address, selectedItems);

    // Cleanup: Delete dynamically allocated items
    for (const auto& item : items) {
        delete item;
    }

    return 0;
}

void addItem(vector<Item*>& items, Item* newItem) {
    items.push_back(newItem);
}

void addItem(vector<Item*>& items, vector<Item*>& selectedItems, Item* newItem) {
    items.push_back(newItem);
    selectedItems.push_back(newItem);
}

void displayItems(const vector<Item*>& selectedItems, string cName, string cAddress) {
    int total = 0;
    bool electronicsSelected = false;  // Variable to track whether Electronics items are
selected

    cout << "\n\n\n";
    cout << "\t  Nepal Mart \n";
    cout << "\t--------------- \n";
    cout << "\n";
    cout << "Name: " << cName << "\t Address: " << cAddress << "\n";
    cout << "\n";

    for (const auto& item : selectedItems) {
        item->display();
        total += item->getPrice();

        // Check if Electronics items are present in the selected items
        if (dynamic_cast<Electronics*>(item) != nullptr) {
            electronicsSelected = true;
        }
    }
```

```
    if (electronicsSelected) {
        // Display VAT only if Electronics items are selected
        int discountedTotal = applyDiscount(total);
        cout << "\tTotal (After 15% Discount and including 13% VAT for Electronics): " <<
discountedTotal << " NPR\n";
    } else {
        // Display total without VAT for other categories
        int discountedTotal = applyDiscount(total);
        cout << "\tTotal (After 15% Discount): " << discountedTotal << " NPR\n";
    }

    cout << "\n\n";
    cout << "\t Thanks for visiting \n";
    cout << "\n\n";
}

void saveCustomerInfo(string cName, string cAddress, const vector<Item*>&
selectedItems) {
    ofstream file("customer_info.txt", ios::app);
    if (file.is_open()) {
        file << "Name: " << cName << "\t Address: " << cAddress << "\t Total Items: " <<
selectedItems.size() << "\n";
        for (const auto& item : selectedItems) {
            file << "Id: " << item->id << "\t Category: " << item->category << "\t Item Name: "
<< item->name << "\t Price: " << item->getPrice() << " NPR\n";
        }
        file << "Total: " << calculateTotal(selectedItems) << " NPR\n\n";
        file.close();
    } else {
        cout << "Unable to open file for saving customer information." << endl;
    }
}

int calculateTotal(const vector<Item*>& selectedItems) {
    int total = 0;
    for (const auto& item : selectedItems) {
        total += item->getPrice();
    }
    return total;
}

int applyDiscount(int total) {
    // Apply 15% discount
    return static_cast<int>(total * 0.85);
}
```

```
vector<Item*> getItemsByCategory(const vector<Item*>& items, const string& category)
{
    vector<Item*> result;
    for (const auto& item : items) {
        if (item->category == category) {
            result.push_back(item);
        }
    }
    return result;
}
```

Algorithm:

The algorithm for this project is explained below:

1. Introduction and Initialization:
   - Display a welcome message.
   - Add user's name and address.
   - Initialize item vectors.

2. Item Management:
   - Define base class (Item) with price and display functions.
   - Create derived classes (Clothing, Electronics, Groceries) with specific implementations.

3. User Interaction:
   - Display category menu and let user choose items.
   - Allow users to select items until they finish, (enter 0) to exit the selection.

4. Display Final Bill:
   - Show selected items with details, calculate total, and apply a 15% discount.
   - If electronics items are present, add 13% VAT.

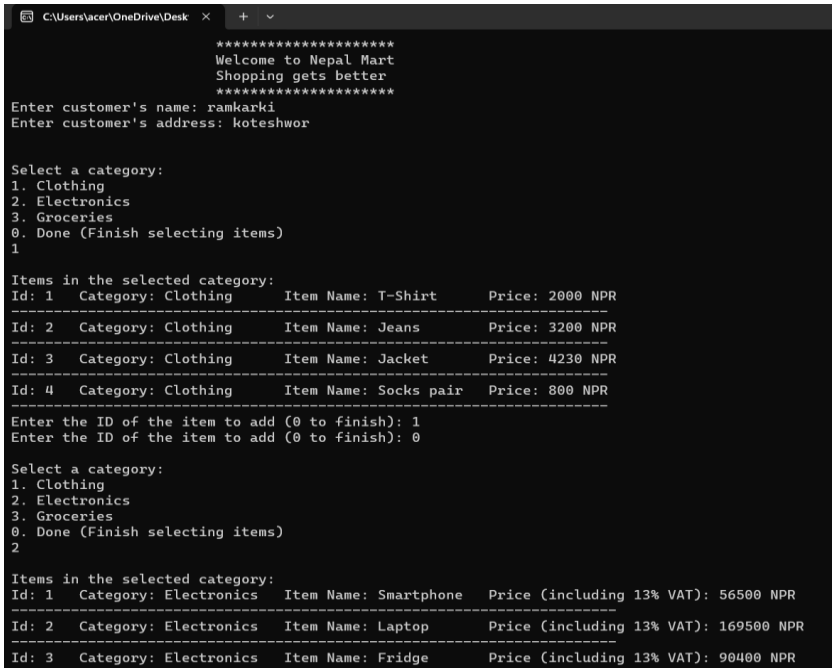
5. Save Customer Information:
   - Open "customer_info.txt" and save user and item details.

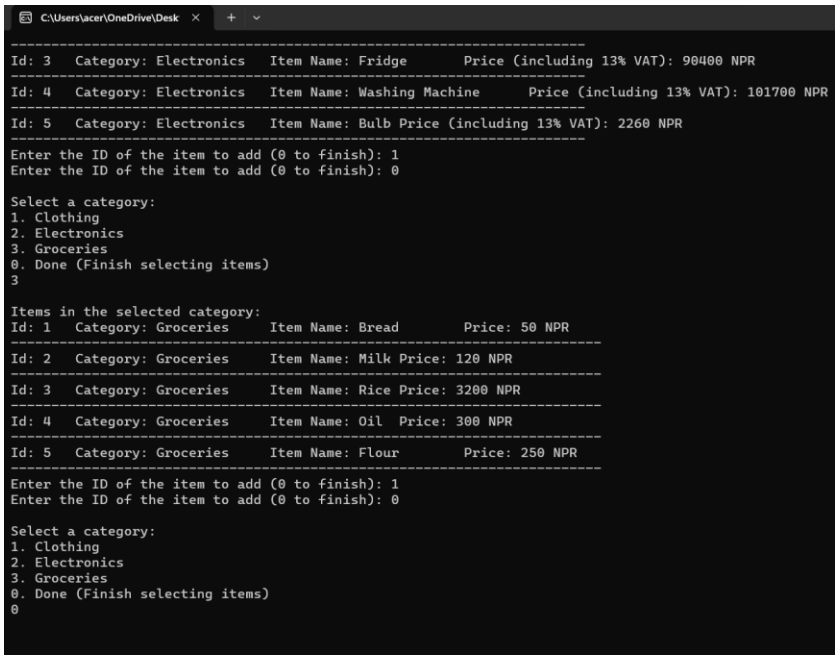
6. End of Program:
   - Display a thank-you message.

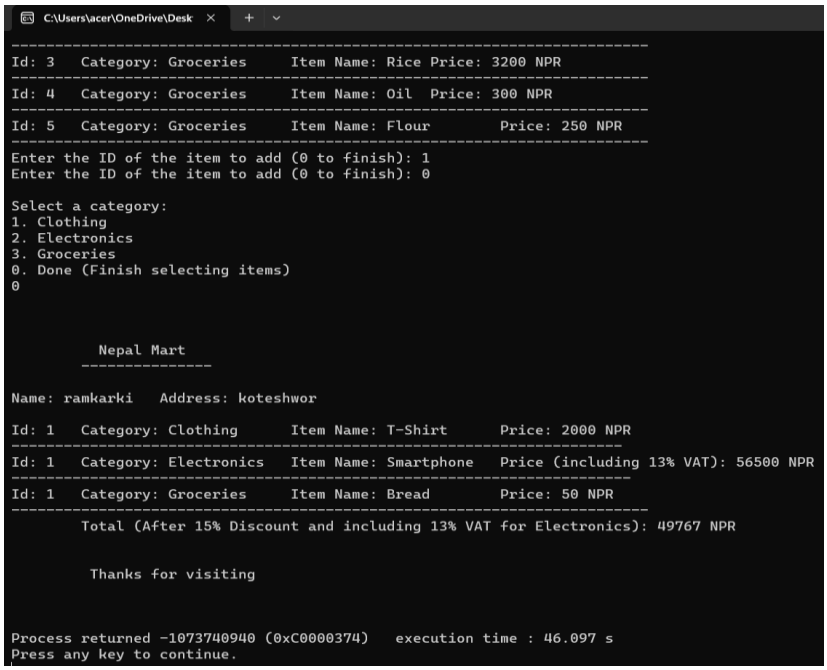
Code Testing:

Here, the output of the code and information of customers saved in file (customer_info.txt) are presented below:

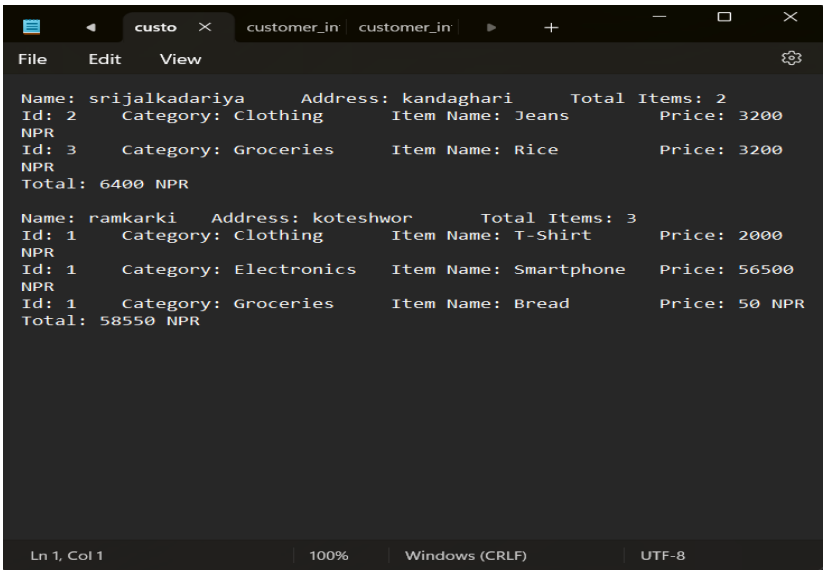
Outputs:







(Customer_info.txt) file:



Conclusion:

After the completion of this project, I was able to learn the basic principles of C++. I was able to develop this object-oriented project which allows the costumers to choose items from different categories. Also, this project allows the user to save their information in a text file for future reference. Overall, the project makes a good balance between simplicity and functionality making it accessible and efficient for the users.
All the requirements that are mentioned in the question like user interface, polymorphism, inheritance, encapsulation, objects and classes, file handling and billing interface are fulfilled.

THANK YOU