**SVKM's NMIMS**
**Mukesh Patel School of Technology Management & Engineering**
Program: B.Tech, Sem II
A.Y. 2021 - 22
**Course: Data Structures and Algorithms**

**Project Report**

| Name of the Project: | Groceria | |
|---|---|---|
| | | |
| Details of Project Members | | |
| Batch | Roll No. | Name |
| I3 | I060 | Mohammad Adil Shaikh |
| I3 | I064 | Mohammed Az Syed |
| I3 | I070 | Chetan Yadav |
| Date of Submission: 23/04/22 | | |

**Note:**

1. Create a readme file if you have multiple files
2. All files must be properly named (I004_DSAProject)
3. All functions and variable should have proper names
4. The code must be properly commented.
5. Submit all relevant files of your work
6. **Plagiarism is highly discouraged**

**Rubrics for the Project evaluation:**

• **Program Correctness**: Program should work correctly on all inputs.
• **Code Elegance**: There are many ways to write the same functionality into your code, and some of them are needlessly slow or complicated. For example, if you are repeating the same code, it should be inside creating a new method/function or for loop.
• **Readability**: Variables and functions should have meaningful names. Code should be organized into functions/methods where appropriate. There should be an appropriate amount of white space so that the code is readable, and indentation should be consistent.
• **Documentation**: Your code and functions/methods should be appropriately commented. However, not every line should be commented because that makes your code overly busy. Think carefully about where comments are needed.
. • **Viva** – Student should be able to explain the logic of the project and answer relevant questions

A.1 **Aim of the Project/ Problem Statement:** To optimize the grocery delivery system and provide an immersive user experience

A.2 **Application /Usefulness of the Problem statement chosen:** Useful for any online retail purposes

A.3 **Description of Project:**

We have made a linked list-based database for user management, inventory management and for the final cart and used graphs to find the minimum distance between source and destination. We have used a menu-driven approach to make it interactive and user friendly.

We start with a welcome page which asks the user whether they are a new or existing customer, depending on their choice, they are further directed to either creating a new account or options to proceed for account info/account modification/ordering groceries respectively. The choice that they make takes them to different functions accordingly and we further make choices and obtain our desired output.

Code and its explanation:

```
UserDatabase u1;
Categories fruits, vegetables, dairy, meats, snacks, bakery;
Cart c1;
```

Implemented three linked lists using classes, we have used classes instead of struct as this helps in decreasing the total number of global variables used in the program making the code more efficient.

```
class UserDatabase{//linked list for user database using classes
    public:
    string userId;
    string pass;
    string name;
    int orders;
    int counter = 0;
    UserDatabase* next;
    UserDatabase* head = NULL;
```

```
class Categories{//Linked list for inventory management using classes
    public:
    int serialNo;
    string name;
    int price;
    Categories* next;
    Categories* head = NULL;
```

```
class Cart{ //Linked list for the final bill using classes
        public:
        string name;
        int cost;
        int number;
        int id;
        Cart*next;
        Cart* head =NULL;
```

The data is entered in the list using insertEnd() /insertBeg() functions of the linked list and we have used deletePos() function in the cart to ensure the user can delete items with ease

```
bool checkUser(string User, string Pass){//Boolean function to check if user exists
    counter = 0;
    UserDatabase *current= head;
    while(current != NULL ){
        if(current ->userId == User && current->pass == Pass){
            return true;
        }
        else{
            counter++;
            current = current->next;
        }
    }
    return false;
}
```

The above function checks if the user exists in the existing user database, it traverses the list until the user id and password match an existing user id and password in the existing database.

```
void modifyUser(string user_Id, string Name, int option){//Function to modify user details
    UserDatabase *current = head;
    for(int i = 0; i < counter - 1; i++)
        current = current ->next;
    if(option == 1)
        current ->userId = user_Id;
    else if(option == 2)
        current ->name = Name;

}
```

This function assists the user in modifying their username as well as their name in the existing user database.

```
void displayList(){// Function to display the list of categories
    Categories *current = head;
    while(current != NULL){
        cout<<current ->serialNo<<") "<<left<<setw(30)<<current ->name<<"Rs. "<<current ->price<<"\n";
        current = current->next;
    }
}
```

The above function displays the menu of the items in the respective categories. It uses the setw() function to set width of the strings to ensure the alignment of the output is in an orderly manner and proper columns.

```
void line(){// Function to print a normal line for aesthetic in terminal
    cout<<"\n";
    for(int i=0;i<93;i++)
        cout<<"-";

    cout<<"\n";
}

void sline(){ // Function to print an asterisk line for aesthetic in terminal
    cout<<"\n";
    for(int i=0;i<93;i++)
    cout<<"*";
    cout<<"\n";
}
```

The above functions assist in printing dash line (-------) & asterisk line (*****) in the terminal for aesthetics. This helps in printing the lines in multiple places without having to write the ‘cout’ statements again and again

```
void displayCart(){ // Function to display the cart with final amount
    if(head == NULL){
      cout<<"Empty Cart"<<endl;
    }
    else{
      int i = 1;
      int cost=0;
      Cart *current =  head ;
      cout<<left<<setw(10)<<"SerialNo"<<left<<setw(35)<<"Name"<<left<<setw(25)<<"Price"<<left<<setw(10)<<"Number of Products"<<endl;
      cout<<endl;
      while(current!=NULL){
        cout<<left<<setw(10)<<i<<left<<setw(35)<<current->name<<left<<setw(25)<<current->cost<<left<<setw(10)<<current->number<<"\n\n";
        cost=cost+(current->cost * current->number);
        current=current->next;
        i++;
      }
      line();
      cout<<"Final Cost:\t"<<cost;
      line();
    }
}
```

The ‘displayCart()’ function helps in displaying the cart and it also calculates the final price of all the selected items and their total quantity. We also use the setw() function to ensure the cart is evenly printed.

```
int minDist(int distance[], bool visit[], int size){//Function to find minimum distance between nodes
    int minimum=INT_MAX,index;
    for(int k=0;k<size;k++){
        if(visit[k]==false && distance[k]<=minimum){
            minimum=distance[k];
            index=k;
        }
    }

    return index;
}
```

The above function helps in finding the minimum distance between the nodes, it basically returns the node which is yet to be visited and also is at the minimum distance from the source node.

```
void dijkstraAlgo(int graph[10][10],int src, int destination, int size){ //Dijkstra algorithm
    int distance[size];
    bool visit[size];

    for(int k = 0; k<size; k++){
        distance[k] = INT_MAX;
        visit[k] = false;
    }

    distance[src] = 0;
    for(int c = 0; c < size; c++){
        int m=minDist(distance, visit, size);
        visit[m]=true;
        for(int k = 0; k<size; k++){
            if(!visit[k] && graph[m][k] && distance[m]!=INT_MAX && distance[m]+graph[m][k]<distance[k])
                distance[k]=distance[m]+graph[m][k];
        }
    }

    cout<<"Estimated delivery Time from GroceriaDeli:- "<<distance[destination]<<" min";
```

Above is the function for Dijkstra's algorithm. When called the function calculates the fastest delivery time to the delivery locations (already provided) based on the selection.

```
u1.insert("abc@gmail.com", "123", "ABC",8);
u1.insert("xyz@gmail.com", "456", "XYZ",4);
```

```
fruits.insert(1, "Apple", 150);
fruits.insert(2, "Banana", 60);
```

Here is an example of how we create the local, pre-existing database for user as well as inventory.

We have incorporated menu driven approach (Switch case) to evaluate a user's choice and to further allow them to make more choices as they progressively interact with the program until their exit.

A.5 **Contribution of each project Members:**

| Roll No. | Name: | Contribution |
|---|---|---|
| I060 | Mohammad Adil Shaikh | Graph, login and error rectification |
| I064 | Mohammed Az Syed | Menu-driven program, prototyping and stack application |
| I070 | Chetan Yadav | Database module and beta testing |

A.6 **Learning from the Project:**

Learned how to implement useful data structures like a linked list and graphs. The usefulness of Dijkstra's algorithm. Implemented a real-life application of data structures.

A.7 **Challenges you faced while doing the Project**

The various issues we faced during the project were the Buggy IDEs, lengthy code, aesthetics in the terminal, scheduling difficulties.

A.9 **Conclusion:**

Understood the theoretical concepts of data structure and algorithms more clearly due to practical implementation