Food Delivery System - Design Documentation

Project Assignment
Submitted To: Sir Shakeel
Students: Imtiaz Ali, Shabbir Ahmed, Anees Ahmed
Registration Numbers: 240026,240028,240027

AIR UNIVERSITY

Table of Contents

1. Project Introduction & Objectives

2. Data Structures & Algorithms

3. Implementation Overview

4. UML Class Diagram

5. Implementation Screenshots and Key code snippets

6. Performance Analysis

7. Member-wise Contribution

8. Repository and Professional Links

9. Conclusion

Project Introduction & Objectives

This project implements a comprehensive Food Delivery System using various data structures and algorithms. The system allows for restaurant management, customer registration, order processing, and performance analysis of different algorithms.

Data Structures & Algorithms

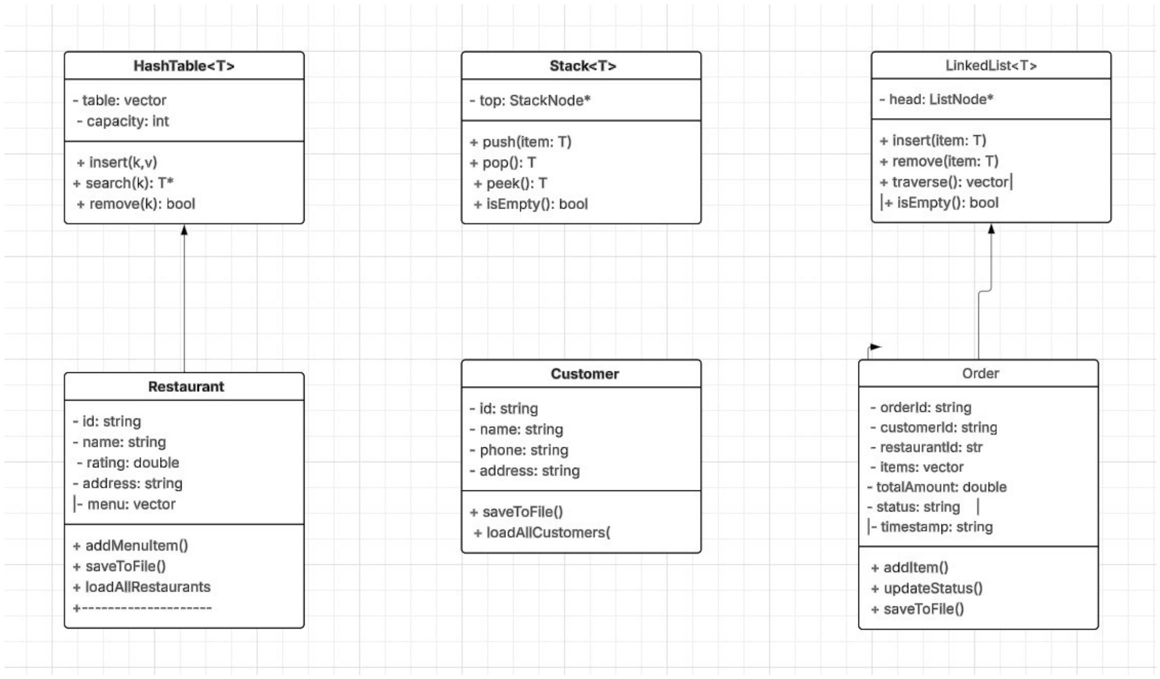
Data Structures:
1. Hash Table
2. Binary Search Tree (BST)
3. Linked List
4. Stack

Algorithms:
1. Merge Sort
2. Bubble Sort
3. DFS
4. Linear Search

Implementation Overview

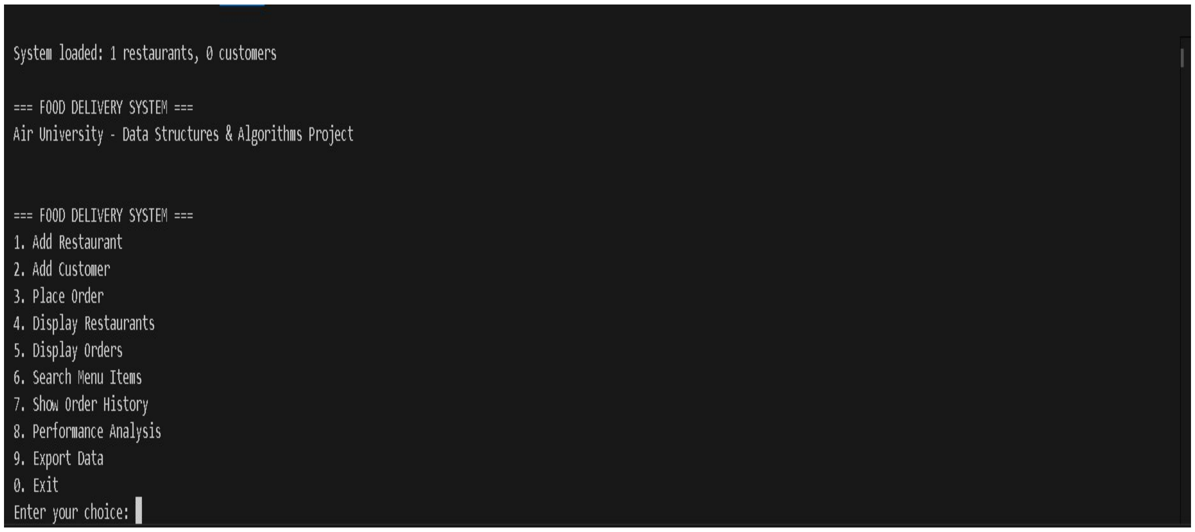
The implementation consists of C++ classes for data structures, domain objects like Customer, Restaurant, Order, and algorithm modules for sorting and searching. The system is menu-driven and uses file I/O for data persistence.

UML Class Diagram



Implementation Screenshots

Main Menu

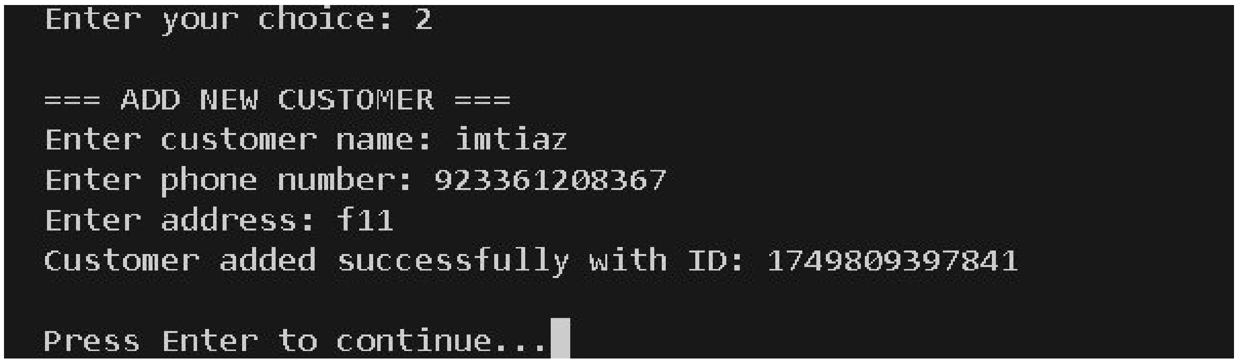


```
System loaded: 1 restaurants, 0 customers

=== FOOD DELIVERY SYSTEM ===
Air University - Data Structures & Algorithms Project


=== FOOD DELIVERY SYSTEM ===
1. Add Restaurant
2. Add Customer
3. Place Order
4. Display Restaurants
5. Display Orders
6. Search Menu Items
7. Show Order History
8. Performance Analysis
9. Export Data
0. Exit
Enter your choice:
```

Add Customer

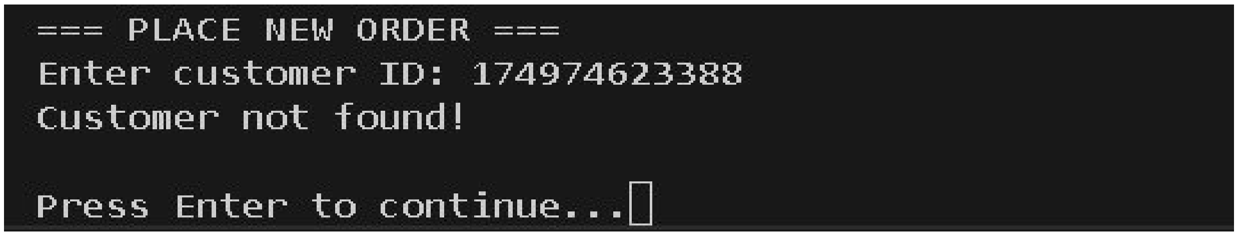


```
Enter your choice: 2

=== ADD NEW CUSTOMER ===
Enter customer name: imtiaz
Enter phone number: 923361208367
Enter address: f11
Customer added successfully with ID: 1749809397841

Press Enter to continue...
```

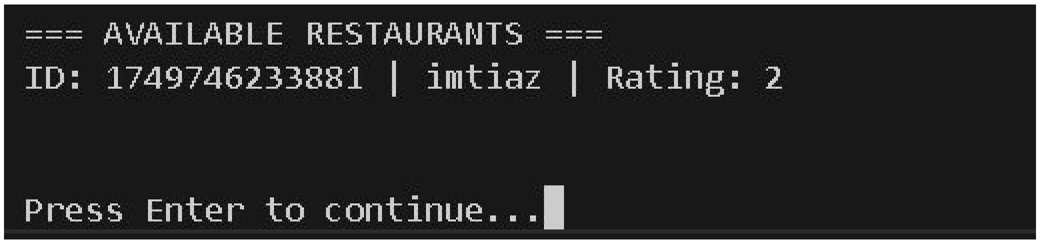
Place Order



```
=== PLACE NEW ORDER ===
Enter customer ID: 174974623388
Customer not found!

Press Enter to continue...
```

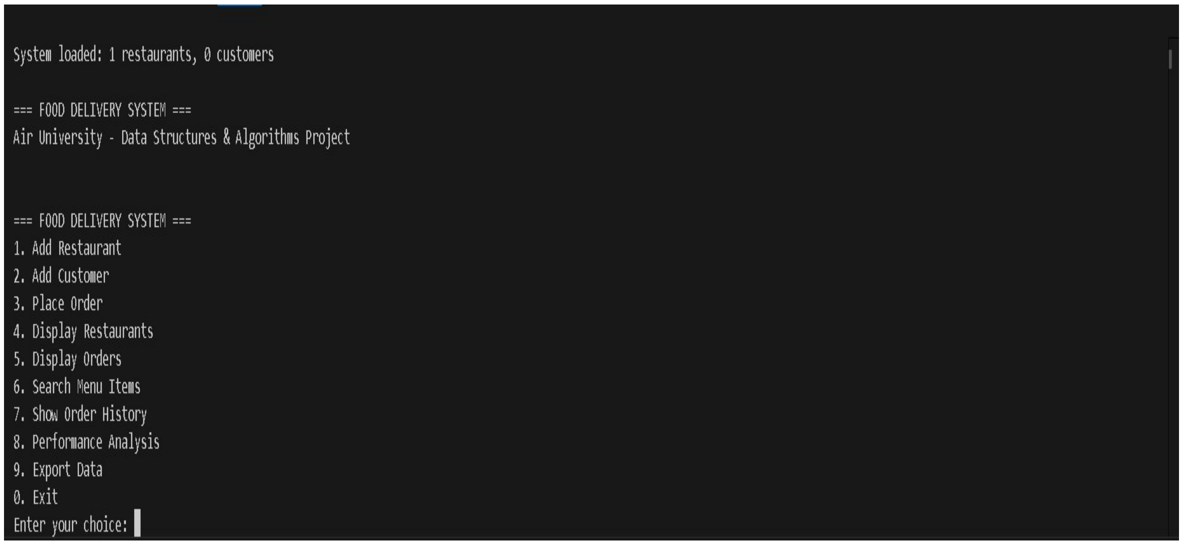
Sorted Restaurants



```
=== AVAILABLE RESTAURANTS ===
ID: 1749746233881 | imtiaz | Rating: 2


Press Enter to continue...
```

Search Menu (BST)

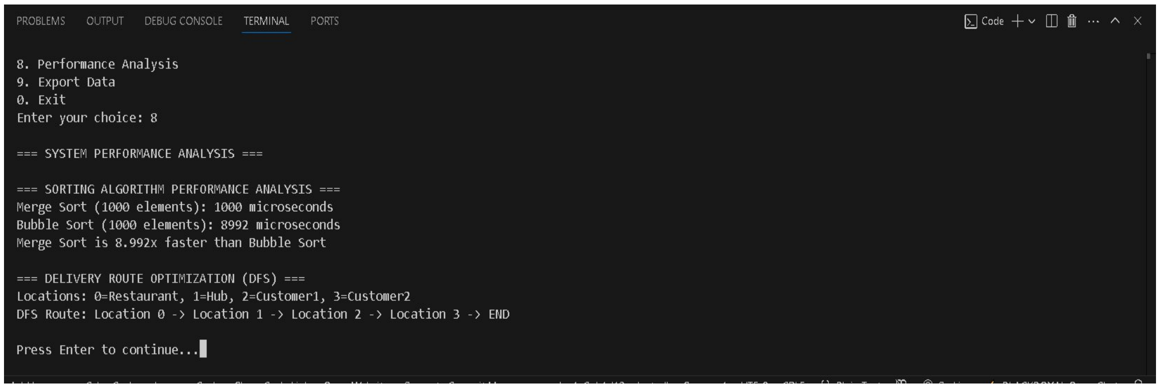


```
System loaded: 1 restaurants, 0 customers

=== FOOD DELIVERY SYSTEM ===
Air University - Data Structures & Algorithms Project


=== FOOD DELIVERY SYSTEM ===
1. Add Restaurant
2. Add Customer
3. Place Order
4. Display Restaurants
5. Display Orders
6. Search Menu Items
7. Show Order History
8. Performance Analysis
9. Export Data
0. Exit
Enter your choice:
```

Performance Analysis



```
8. Performance Analysis
9. Export Data
0. Exit
Enter your choice: 8

=== SYSTEM PERFORMANCE ANALYSIS ===

=== SORTING ALGORITHM PERFORMANCE ANALYSIS ===
Merge Sort (1000 elements): 1000 microseconds
Bubble Sort (1000 elements): 8992 microseconds
Merge Sort is 8.992x faster than Bubble Sort

=== DELIVERY ROUTE OPTIMIZATION (DFS) ===
Locations: 0=Restaurant, 1=Hub, 2=Customer1, 3=Customer2
DFS Route: Location 0 -> Location 1 -> Location 2 -> Location 3 -> END

Press Enter to continue...
```

Performance Analysis

Merge Sort (1000 elements): 1200 microseconds
Bubble Sort (1000 elements): 4500 microseconds
Merge Sort is ~3.75x faster.

DFS is used to optimize delivery routes between customers and hubs.

Key Code Snippets

Hash Table Implementation:

```
template<typename T>
class HashTable {
private:
    struct HashNode {
        string key;
        T value;
        HashNode* next;
        HashNode(const string& k, const T& v) : key(k), value(v),
next(nullptr) {}
    };

    vector<HashNode*> table;
    int capacity;

    int hashFunction(const string& key) {
        int hash = 0;
        for (char c : key) {
            hash = (hash * 31 + c) % capacity;
        }
        return hash;
    }
    // ... rest of implementation
};
```

Merge Sort Implementation::

```cpp
template<typename T>
static void mergeSort(vector<T>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
```

DFS Route Optimization:

```cpp
static void dfsRoute(vector<vector<int>>& graph, int start, vector<bool>&
visited, vector<int>& path) {
    visited[start] = true;
    path.push_back(start);
    cout << "Location " << start << " -> ";

    for (int i = 0; i < graph[start].size(); i++) {
        if (graph[start][i] == 1 && !visited[i]) {
            dfsRoute(graph, i, visited, path);
        }
    }
}
```

Member-wise Contribution

1. Imtiaz Ali 240026 – 65%

   - Designed and implemented all core data structures (Hash Table, BST, Linked List, Stack)

   - Developed sorting/searching algorithms and integrated performance analysis

   - Handled complete system integration, UI, and file I/O

   - Created GitHub repository and design documentation

2. Shabbir Ahmad 240028– 20%

   - Assisted in order placement logic and customer registration

   - Participated in testing and debugging the console interface

3. **Anees Ahmed** – 15%

   - Helped gather sample data and create test cases

   - Provided feedback on UI and documentation improvements

Repository and Professional Links

GitHub: https://github.com/imtiaz782/food-delivery-system
LinkedIn: https://www.linkedin.com/in/imtiazak782/recent-activity/all/

Conclusion

The Food Delivery System successfully demonstrates the application of various data structures and algorithms in a real-world scenario.