UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

FACULTY OF COMPUTING

SESSION 2023/2024

SEMESTER 1

SECJ2013-04 DATA STRUCTURE AND ALGORITHM

(STRUKTUR DATA DAN ALGORITMA)

ASSIGNMENT 3 - DATA STRUCTURE AND OPERATIONS - QUEUE, STACK AND TREE

LECTURER: DR. LIZAWATI BINTI MI YUSUF

NO	NAME	MATRIC NO
1	SARAH SOFEA BINTI ANUAR	A22EC104
2	FARAH HAZIRAH NISHA BINTI ABD LATIF	A22EC0159
3	MUHAMMAD ABDUH BIN ABDUL BA'ARI	A22EC0199

OBJECTIVE

The main objective for developing the courier management system is to ease the courier and customer to insert, delete, search and display the courier information. The courier management system is designed to make the process of insertion and deletion of the courier information run smoothly and follow the flow as it follows the “First In, First Out” (FIFO) principle and get packages to their destination safely and on time. The system provides a user-friendly interface for simplicity of use, with a menu-driven system that leads users through the full process of selecting the menu provided until they choose to exit the system. The system implements the queue principle as that principle is more suitable to be implemented for the courier management system. Typically, the initial detail of a parcel will be delivered to the customer first since the parcel was sent earlier than subsequent parcels, allowing the administrator to remove the information flow by flow. Usually, the first detail will be delivered first to the customer as the parcel was sent earlier than other parcels so that the admin can delete the information flow by flow. To maintain data integrity during file activity, the system also additionally employs a data hiding concept. Data hiding ensures exclusive data access to class members exclusively and preserves object integrity by limiting modifications and disruptions, whether planned or unintended.

There are few features in the courier management system which is:

1. Insert data courier

Insert information of the parcel including name, parcel type, source, destination, status and tracking number. The new information will be inserted at the back/last of the list of parcel details.

2. Remove or delete data couriers

Remove or delete the information about the parcel. The information will be deleted from the top/front of the list in the Queue.

3. Search courier information

Can find any data that exists in the list by inserting the tracking number for the parcel. The algorithm used is a linked list implementation of queue.

4. Display data courier

Displaying all updated courier information including name, parcel type, source, destination, status and tracking number.

5. Update data courier

Update the status attributes in couriers from “Pending” to “Approved” and from “Approved” to “In transit”.

SYNOPSIS

The courier management system is used to send the parcel to the specific destination and track the current location. We designed the courier system by implementing the Queue method to easily allow the user to insert, delete,change status, search and display the information of the parcel. When the customers submit the parcel details in the system, the details will be stored in the input file (COURIER.TXT). Using a Queue is suitable for a courier management system as it follows the “First In, First Out” (FIFO) principle. The first parcel detail submitted and stored is the first to be processed by the admin and worker.

In our system, the customer has a menu to choose whether they want to add details of parcel, view the courier queue, and search courier by tracking number and exit the customer menu. The customer can insert the new details about the parcel that contains, name, type of parcel, source, destination, and the status of the parcel, and tracking number. The information will be inserted as the last/back position of information of the parcel by following the Queue principle (enqueue). The status of new detail that is inserted by the customer will be “Pending” and will be changed to the “Approved” by the Admin after the admin approves the pending couriers. Next, the Worker will update the parcel from “Approved” status to “In Transit” status. In addition, the admin also has a menu to choose whether they want to view the courier queue, approve courier, dequeue the courier (remove data) and exit from admin’s menu. While, the worker has a menu to choose whether they want to view the courier queue, mark courier as In transit, dequeue the courier (remove data) and exit from worker’s menu.The admin and worker can remove or delete the details of the parcel from the input file and the details will be deleted from the top/first parcel’s information of the input file by regarding the Queue principle (dequeue). Furthermore, we also implement the search feature for the customers to search the details of the parcel by entering the tracking number. Then, the system will display all the details of the parcel that have the same tracking number as the user entered into the system. Last but not least, the system also has a display/view menu for customers, admins and workers to display all and updated details of the parcel from the input file(COURIER.txt).

PROBLEM ANALYSIS

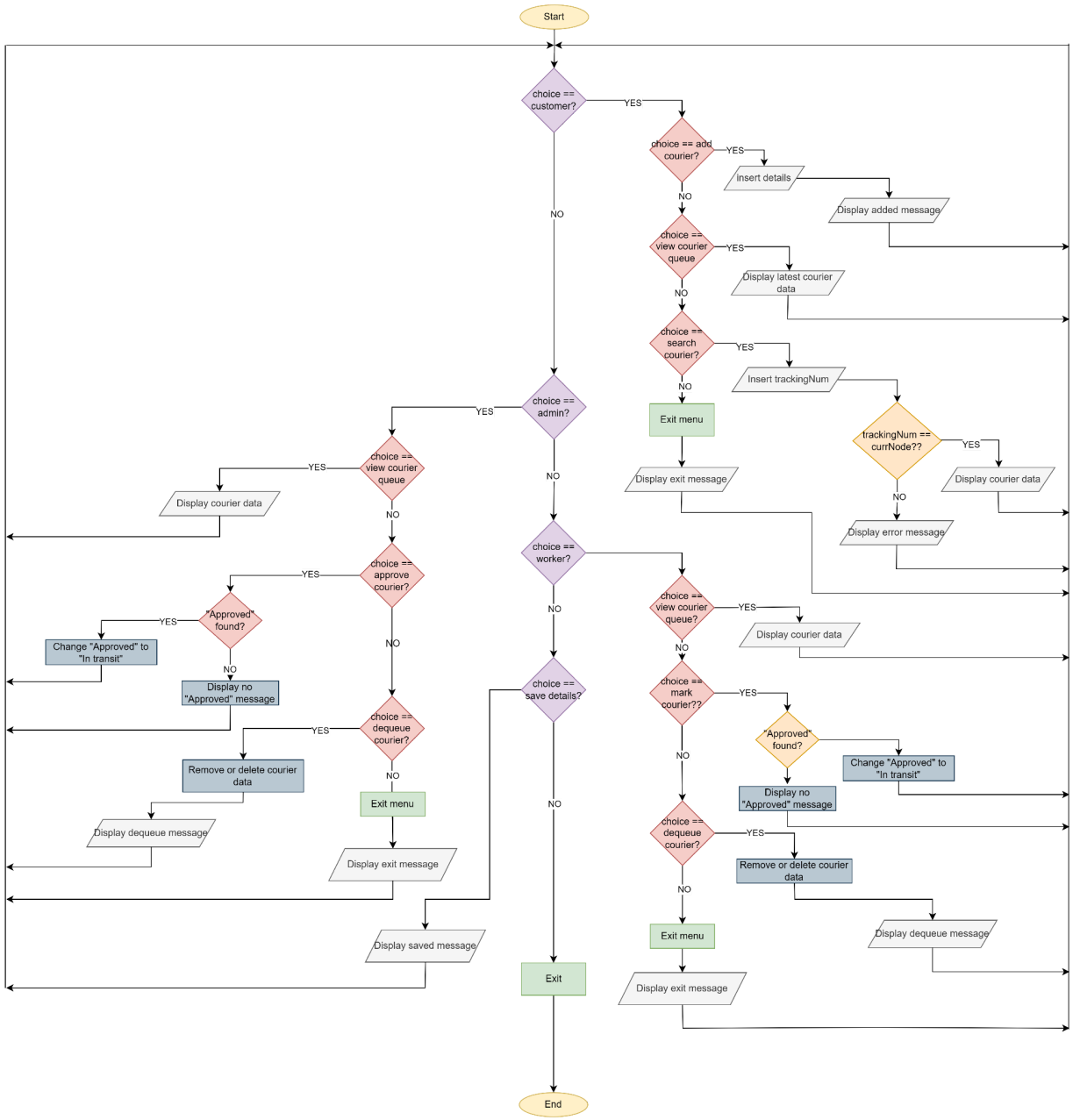
The courier management system is designed to facilitate and improve the courier services management by handling all the information that they receive from customers and synchronizing it with the updated delivery status for their parcel. To make all the process become a lot more easier than before, we have decided to use a Queue data structure that will follow the principle of "First In,First Out" (FIFO). As a result the first data to enter the system (enqueue) will also be the first data deleted (dequeue) from the system.

Other than that, to help the customer at the courier service center to find all the data for which parcel's status has been delivered or not, we have decided to implement a searching process. Our system has the ability to search information about couriers by inserting the tracking number of the parcel. All the parcel details that contain the similar tracking number like the one that we searched will be displayed. Meanwhile, the admin of the system can approve the new data either the data was inserted by input file (COURIER.TXT) or customer. Admin also have the ability to delete the first data inserted to the system each time the dequeue function is being called.

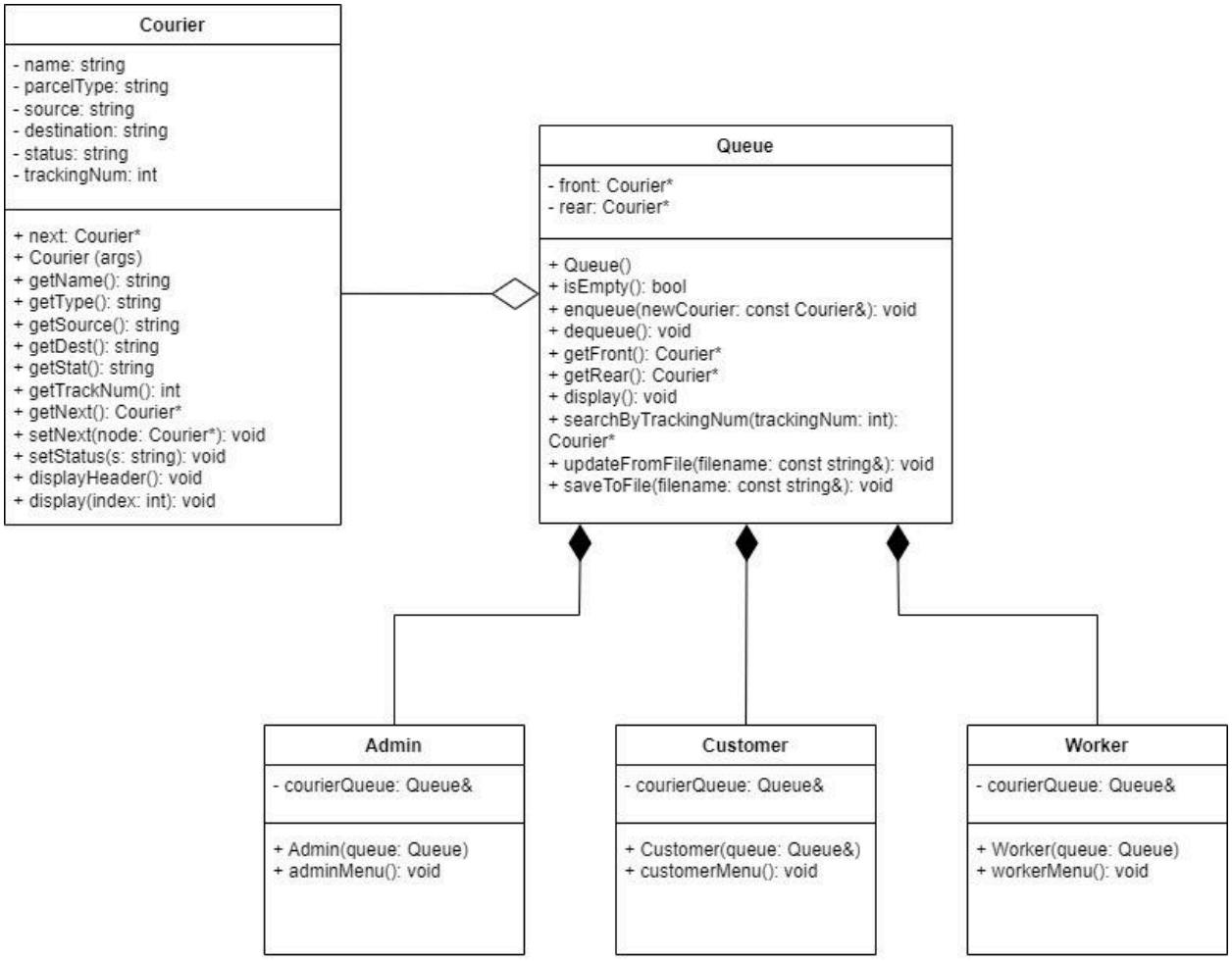
In addition, our system also offers the workers of the courier service to update the status from being "Approved" to "In Transit". From this update, customers can know that their parcels are currently being processed. The workers also have the same ability like admin which is to delete the first data inserted to the system each time the dequeue function is being called. Last but not least, our system also can save the updated data into the file after we exit the system. As a conclusion, the courier management system is a very useful system that could help all courier services management to improve their functionality and its user experience.

DESIGN

Flowchart



Class Diagram



DESIGN DESCRIPTION

ADDING A NEW COURIER

1. Get the option from the customer menu.
2. The customer menu provides an option for customers to add a new courier into the queue at the rear.

   2.1 The enqueue function in the Courier Service Systems allows customers to add new couriers to the system.

      2.1.1 Calls the ‘enqueue’ function in the ‘Queue’ class to add a new courier into the system.

      2.1.2 User inputs courier details such as name, parcel type, source, destination, tracking number.

      2.1.3 The status will be automatically set to “Pending’ by default.

      2.1.4 The courier is added to the rear of the courier que (follow queue rule).

      2.1.5 Return back to the customer menu.

UPDATING AND REMOVING FROM QUEUE

1. Get the option from the admin and worker menu.
2. From the admin menu, the courier status can be updated from “Pending” to “Approved”.
   2.1 Iterates through the courier queue to find the first courier with status “Pending”.
      2.1.1 If found, change the status from “Pending” to “Approved”.
      2.1.2 Notifies the approval to the admin.
   2.2 If no status “Pending” is found
      2.2.1 Notify the admin that there are no more pending status to approve.
3. From the worker menu, the courier status can be updated from “Approved” to “In Transit”.
   3.1 Iterates through the courier queue to find the first courier with status “Approved”.
      3.1.1 If found, change the status from “Approved” to “In transit”.
      3.1.2 Notifies the change to the worker.
   3.2 If no status “Approve” in found
      3.2.1 Notify the worker that there are no more approved status to be changed to in transit.
4. From the admin menu, the courier queue can be deleted (dequeue)
   4.1 The courier at the front is removed from the courier queue.
5. From the worker menu, the courier queue can be deleted (dequeue)
   5.1 The courier at the front is removed from the courier queue.
6. Return back to the admin/worker menu.

SEARCHING THE COURIER

1. Get the option from the customer menu.
2. From the option, courier information will be searched by tracking the tracking number entered from the customer.

   2.1 Iterates through the queue, comparing the attributes of each courier to the corresponding search key.

   2.1 If the courier found

   2.1.1 Display the courier information with the search key.

   2.2 If the courier not found

   2.2.1 Notifies the customer that there is no information for the courier with the corresponding search key.
3. Return back to the customer menu.

DISPLAYING THE COURIER

1. Get the option form the customer, admin and worker menu.
2. Display the content in the courier queue starting from front to rear.
3. Return back to the customer/admin/worker menu.

SAVE COURIER INFORMATION

1. Get the option in the main menu.
2. Save the latest courier queue to the filename “COURIER.TXT”.
3. Return back to the main menu.

SOURCE CODE

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <fstream>
#include <cstdlib>

using namespace std;

class Courier {
private:
   string name, parcelType, source, destination, status;
   int trackingNum;

public:
        Courier* next;
   Courier()
      : name(" "), parcelType(" "), source(" "), destination(" "), status(" "), trackingNum(0),
next(NULL) {}

   Courier(string name, string parcelType, string source, string destination, string status, int
trackingNum)
      : name(name), parcelType(parcelType), source(source), destination(destination),
status(status), trackingNum(trackingNum), next(NULL) {}

   string getName() const { return name; }
   string getParcelType() const { return parcelType; }
   string getSource() const { return source; }
   string getDestination() const { return destination; }
   string getStatus() const { return status; }
```

```
    int getTrackingNum() const { return trackingNum; }

    Courier* getNext() const { return next; }
    void setNext(Courier* node) { next = node; }
    void setStatus(string s) {status = s;}

    void displayHeader() const {
        cout << left << setw(5) << "No." << setw(20) << "Tracking Number" << setw(20) <<
"Name"
             << setw(20) << "Parcel Type" << setw(25) << "Source"  // Increased width to 25
             << setw(25) << "Destination" << setw(20) << "Status" << endl;
        cout << setfill('-') << setw(135) << " " << setfill(' ') << endl;
    }

    void display(int index) const {
        cout << "[" << setw(1) << index << "]   "
             << setw(20) << trackingNum << setw(20) << name
             << setw(20) << parcelType << setw(25) << source  // Increased width to 25
             << setw(25) << destination << setw(20) << status << endl;
    }
};

class Queue {
private:
    Courier* front;
    Courier* rear;

public:
    Queue() : front(NULL), rear(NULL) {}

    bool isEmpty() const {
```

```
    return front == NULL;
}

void enqueue(const Courier& newCourier) {
    Courier* node = new Courier(
        newCourier.getName(),
        newCourier.getParcelType(),
        newCourier.getSource(),
        newCourier.getDestination(),
        newCourier.getStatus(),
        newCourier.getTrackingNum()
    );

    if (isEmpty()) {
        front = rear = node;
    } else {
        rear->setNext(node);
        rear = node;
    }
}

void dequeue() {
    if (isEmpty()) {
        cout << "\nQueue is empty." << endl;
        return;
    }

    Courier* delNode = front;
    front = delNode->getNext();

    delete delNode;
```

```
}

Courier* getFront() const {
   return front;
}

Courier* getRear() const {
      return rear;
      }

void display() {
   int counter = 1;
   Courier* currNode;

   if (isEmpty()) {
      cout << "\nQueue is empty." << endl;
   } else {
      currNode = front;
      currNode->displayHeader();

      while (currNode) {
         currNode->display(counter);
         currNode = currNode->getNext();
         counter++;
         cout << endl;
      }
   }
}

Courier* searchByTrackingNum(int trackingNum) const {
   Courier* currNode = front;
```

```
    while (currNode) {
        if (currNode->getTrackingNum() == trackingNum) {
            return currNode; // Return the found Courier
        }
        currNode = currNode->getNext();
    }

    return NULL; // Return NULL if not found
}

        void updateFromFile(const string& filename) {
    ifstream inputFile(filename);

                if (!inputFile.is_open()) {
    cout << "Error opening file: " << filename << endl;
    return;
}

// Clear existing queue
while (!isEmpty()) {
    dequeue();
}

string line;
while (getline(inputFile, line)) {
    stringstream ss(line);
    string name, parcelType, source, destination, status;
    int trackingNum;

    if (getline(ss >> std::ws, name, ',') &&
```

```
        getline(ss >> std::ws, parcelType, ',') &&
        getline(ss >> std::ws, source, ',') &&
        getline(ss >> std::ws, destination, ',') &&
        getline(ss >> std::ws, status, ',') &&
        (ss >> trackingNum)) {

        Courier newCourier(name, parcelType, source, destination, status, trackingNum);
        enqueue(newCourier);
        }
}
        inputFile.close();
        }

void saveToFile(const string& filename) const {
        ofstream file(filename);

        if (!file.is_open()) {
            cout << "Error opening file: " << filename << endl;
            return;
        }

        Courier* currNode = front;

        while (currNode) {
            // Write courier data to the file
            file << currNode->getName() << ","
                 << currNode->getParcelType() << ","
                 << currNode->getSource() << ","
                 << currNode->getDestination() << ","
                 << currNode->getStatus() << ","
                 << currNode->getTrackingNum() << endl;
```

```
            currNode = currNode->getNext();
        }

        file.close();
    }
};

class Customer {
private:
    Queue &courierQueue;

public:
    Customer(Queue &queue) : courierQueue(queue) {}
    void customerMenu() {
        int choice;

        system("CLS");
        do {
            cout << "===== Customer Menu =====" << endl;
            cout << "1. Add Courier" << endl;
            cout << "2. View Courier Queue" << endl;
            cout << "3. Search Courier by Tracking Number" << endl;
            cout << "4. Exit" << endl;
            cout << "Enter your choice: ";
            cin >> choice;

            switch (choice) {
                case 1: {
                    string name, parcelType, source, destination, status;
                    int trackingNum;
```

```
            cout << "\nEnter Courier details:" << endl;
            cout << "Name: ";
            cin.ignore(); // Clear buffer
            getline(cin, name);
            cout << "Parcel Type: ";
            getline(cin, parcelType);
            cout << "Source: ";
            getline(cin, source);
            cout << "Destination: ";
            getline(cin, destination);
            status = "Pending"; // Set status to "pending" initially
            cout << "Tracking Number: ";
            cin >> trackingNum;

                                    cout << "\nCourier added successfully with status
'pending'." << endl;
                                    cout << endl;
            Courier newCourier(name, parcelType, source, destination, status, trackingNum);
            courierQueue.enqueue(newCourier);

            break;
        }
        case 2: {
            // View Courier Queue
            cout << endl;
            courierQueue.display();
            break;
        }
        case 3: {
            // Search Courier by Tracking Number
```

```
            int trackingNum;
            cout << "\nEnter Tracking Number to search: ";
            cin >> trackingNum;

            Courier* foundCourier = courierQueue.searchByTrackingNum(trackingNum);

            if (foundCourier) {
                cout << "\nCourier found:" << endl;
                cout << endl;
                foundCourier->displayHeader();
                foundCourier->display(1);
                cout << endl;
            } else {
                cout << "\nCourier with Tracking Number " << trackingNum << " not found." <<
endl;
                cout << endl;
            }
            break;
        }
        case 4:
            cout << "\nExiting customer menu. Have a nice day!" << endl;
            cout << endl;
            break;
        default:
            cout << "Invalid choice. Please try again." << endl;
      }
    } while (choice != 4);
  }
};

class Admin {
```

```
private:
    Queue &courierQueue;

public:
    Admin(Queue &queue) : courierQueue(queue) {}

    void adminMenu() {
        int choice;

        system("CLS");
        do {
            cout << "===== Admin Menu =====" << endl;
            cout << "1. View Courier Queue" << endl;
            cout << "2. Approve Courier" << endl;
            cout << "3. Dequeue Courier" << endl;  // Added option to dequeue
            cout << "4. Exit" << endl;
            cout << "Enter your choice: ";
            cin >> choice;

            switch (choice) {
                case 1: {
                    // View Courier Queue
                    int counter = 1;
                    Courier* currentCourier = courierQueue.getFront();

                    if (!currentCourier) {
                        cout << "\nQueue is empty." << endl;
                        cout << endl;
                    } else {
                        currentCourier->displayHeader();
```

```
            while (currentCourier) {
                currentCourier->display(counter);
                currentCourier = currentCourier->getNext();
                counter++;
            }
            cout << endl;
        }
        break;
    }
    case 2: {
        // Approve Courier
        bool foundPendingCourier = false;
        Courier* currentCourier = courierQueue.getFront();

        while (currentCourier) {
            if (currentCourier->getStatus() == "Pending") {
                foundPendingCourier = true;
                currentCourier->setStatus("Approved");
                cout << "\nCourier with Tracking Number " <<
currentCourier->getTrackingNum() << " approved." << endl;
            }

            currentCourier = currentCourier->getNext(); // Move to the next courier
        }

        if (!foundPendingCourier) {
            cout << "\nNo more pending couriers to approve." << endl;
        }

        cout << endl;
        break;
```

```
            }
            case 3: {
                // Dequeue Courier
                if (!courierQueue.isEmpty()) {
                    cout << "\nDequeued a courier." << endl;
                    courierQueue.dequeue();
                } else {
                    cout << "\nQueue is empty. No courier to dequeue." << endl;
                }
                cout << endl;
                break;
            }
            case 4:
                cout << "\nExiting admin menu. Have a nice day!" << endl;
                cout << endl;
                break;
            default:
                cout << "\nInvalid choice. Please try again." << endl;
        }
    } while (choice != 4);
  }
};

class Worker {
private:
  Queue &courierQueue;

public:
  Worker(Queue &queue) : courierQueue(queue) {}

  void workerMenu() {
```

```
int choice;

system("CLS");
do {
    cout << "===== Worker Menu =====" << endl;
    cout << "1. View Courier Queue" << endl;
    cout << "2. Mark Courier as In transit" << endl;
    cout << "3. Dequeue Courier" << endl;  // Added option to dequeue
    cout << "4. Exit" << endl;
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1: {
            // View Courier Queue
            int counter = 1;
            Courier* currentCourier = courierQueue.getFront();

            if (!currentCourier) {
                cout << "Queue is empty." << endl;
            } else {
                currentCourier->displayHeader();

                while (currentCourier) {
                    currentCourier->display(counter);
                    currentCourier = currentCourier->getNext();
                    counter++;
                }
                cout << endl;
            }
            break;
```

```
            }
            case 2: {
                // Mark Courier as Delivered
                Courier* currentCourier = courierQueue.getFront();

                while (currentCourier) {
                    if (currentCourier->getStatus() == "Approved") {
                        currentCourier->setStatus("In transit");
                        cout << "\nCourier with Tracking Number " <<
currentCourier->getTrackingNum() << " marked as in transit." << endl;
                    }

                    currentCourier = currentCourier->getNext();
                }

                cout << endl;
                break;
            }
            case 3: {
                // Dequeue Courier
                if (!courierQueue.isEmpty()) {
                    cout << "\nDequeued a courier." << endl;
                    courierQueue.dequeue();
                } else {
                    cout << "\nQueue is empty. No courier to dequeue." << endl;
                }
                cout << endl;
                break;
            }
            case 4:
                cout << "\nExiting worker menu. Have a nice day!" << endl;
```

```
                cout << endl;
                break;
            default:
                cout << "\nInvalid choice. Please try again." << endl;
        }
    } while (choice != 4);
  }
};

int main() {
        Queue courierQueue;
        courierQueue.updateFromFile("COURIER.txt");
  Customer customer(courierQueue);
  Admin admin(courierQueue);
  Worker worker(courierQueue);

  int choice;
  system("CLS");
  do {
    cout << "===== Main Menu =====" << endl;
    cout << "1. Customer Menu" << endl;
    cout << "2. Admin Menu" << endl;
    cout << "3. Worker Menu" << endl;
    cout << "4. Save Courier Data to File" << endl;
    cout << "5. Exit" << endl;
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
      case 1:
        customer.customerMenu();
```

```
            break;
        case 2:
            admin.adminMenu();
            break;
        case 3:
            worker.workerMenu();
            break;
        case 4: {
            // Save courier data to a file
            string filename = "COURIER.TXT";
            courierQueue.saveToFile(filename);
            cout << "\nCourier data saved to file (COURIER.TXT)." << endl;
            cout << endl;
            break;
        }
        case 5:
            cout << "\nExiting main menu. Goodbye!" << endl;
            break;
        default:
            cout << "\nInvalid choice. Please try again." << endl;
      }
   } while (choice != 5);

   return 0;
}
```

USER GUIDELINE

Task	Step
Customer menu	
1. Adding a new courier	1. Access customer menu. 2. Choose to add a new courier option. 3. Enter details: name, parcel type, source, destination, tracking number. 4. Status is automatically set to “Pending” by default. 5. Courier added to the queue (rear). 6. Return to the customer menu.
2. Searching the courier	1. Access customer menu. 2. Choose to search for courier by tracking number option. 3. Enter tracking number. 4. If found, courier information will be displayed. 4.1 If not found, no information will be displayed. 5. Return to the customer menu.
3. Display the courier	1. Access customer menu. 2. Choose the view courier queue option. 3. Content from the queue will be displayed from front to rear. 4. Return to the customer menu.
Admin menu	
1. Updating the courier data	1. Access admin menu. 2. Choose an approved courier option. 3. If the courier status is “Pending” 3.1 It will be updated to “Approved”. 3.2 Admin will be notified which courier has been updated. 4. If no more courier with status “Pending” 4.1 notify admin that there are no more pending couriers. 5. Return to the admin menu.
2. Remove courier from queue	1. Access admin menu. 2. Choose the dequeue courier option. 3. Courier at the front will be dequeue(deleted) in the list. 4. Return to the admin menu.

3. Display the courier	1. Access admin menu. 2. Choose the view courier queue option. 3. Content from the queue will be displayed from front to rear. 4. Return to the admin menu.
Worker menu	
1. Updating the courier data	1. Access worker menu. 2. Choose mark courier as In transit option. 3. If the courier status is “Approved” 3.1 It will be updated to “In transit”. 3.2 Workers will be notified which courier has been updated. 4. If no more courier with status “Approved” 4.1 notify workers that there are no more approved couriers. 5. Return to the worker menu.
2. Remove courier from queue	1. Access worker menu. 2. Choose the dequeue courier option. 3. Courier at front will be dequeue (deleted) in the list. 4. Return to the worker menu
3. Display the courier	1. Access worker menu. 2. Choose the view courier queue option. 3. Content from the queue will be displayed from front to rear. 4. Return to the worker menu.
Main menu	
1. Save courier information	1. Access the main menu. 2. Choose the save the courier data to file option. 3. Latest courier queue information will be saved to the file name “COURIER.TXT”. 4. Return to the main menu.