DEPARTMENT OF COMPUTER SCIENCE & IT

Ride Hailing System

Online Travel

Muqaddas Ali CT 01

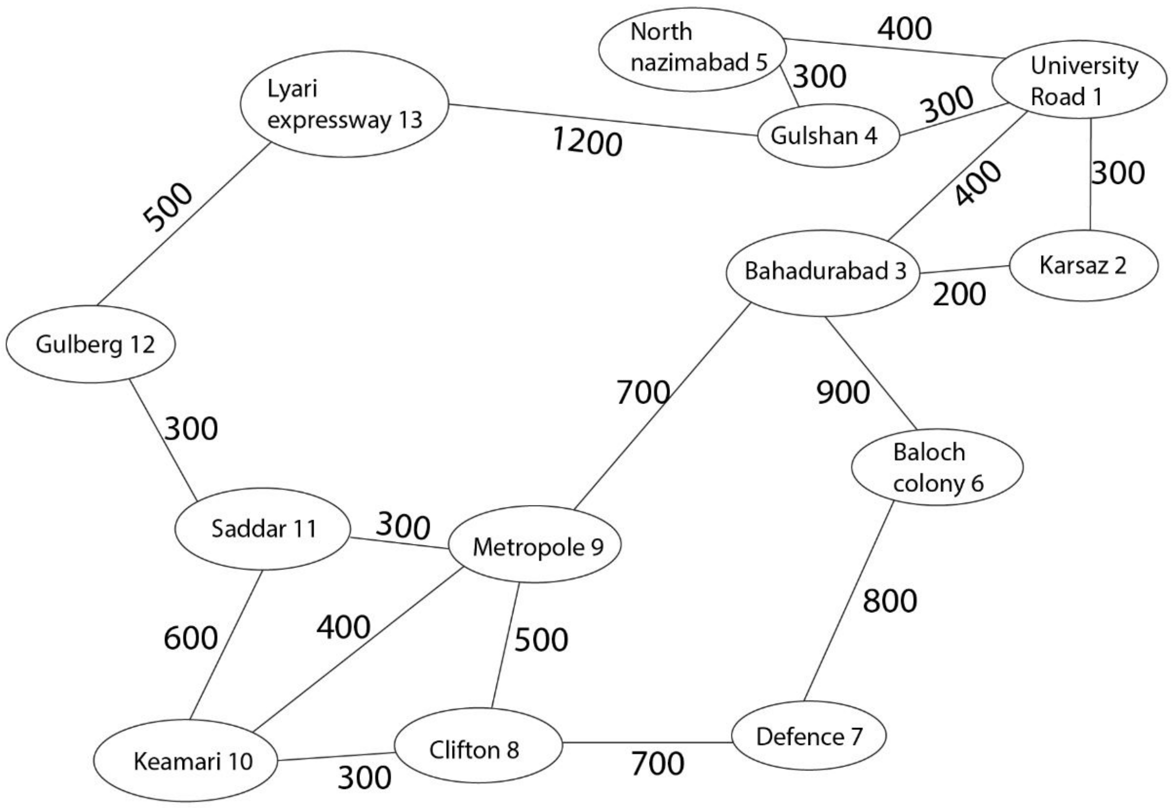
Fatima Nadeem CT 11

Aisha Shahzad CT 15

Maryam Irfan CT 18

INTRODUCTION

Our project is about booking a ride, Firstly the Main menu will be presented on the screen which includes Rider, Driver and Admin, to register yourself as a driver or a rider you need to provide your NIC along with your phone number and email address, Limitation has been added that the phone number must be of 11 digits and the CNIC number must be of 13 digits. After registering as a rider you need to enter your pickup and drop off location and your estimated fare will be calculated accordingly through the graph then the rider will be asked to enter the number of passengers, The car will be assigned to the rider from the linked list according to the capacity that the rider has entered. The ride are then stored in a linked list and if the user registers as a driver so the name is stored in the file while the user will be asked to enter the phone number, CNIC and email address. After reading the file driver will be assigned randomly to the rider, rider can also provide the feedback after availing the service. The admin can view the records of Driver and Rider while the admin can also read the feedback given by the rider.



ALGORITHM

```
FUNCTION file(text file name){
        Opens file
        IF file doesn't exist
                Outputs that it doesn't exist and exits code
        ELSE WHILE the end of file isn't reached
                Outputs each line
        Closes file }
CLASS Admin
        PRIVATE:  username = Fatima_NW , password =Hughesb@fma
        PUBLIC :   FUNCTION login(){
                        Admin enters name
                        Admin enters Password
                                IF username and password NOT correct
                                        Output wrong info and recursive call login()
                                ELSE Output logged in
                FUNCTION view feedback(){
                        Reads from Riderfeedback text file}
                FUNCTION view_driver_records() {
                        Reads from DriverDetails text file}
                FUNCTION view_rider_records() {
                        Reads from Registeredriders text file}
                FUNCTION admin_menu(){
                        Outputs 4 options
                        User Inputs option
                        SWITCH(option){
                                CASE 1   Function call view_driver_records()
                                         Function call admin_menu()
                                CASE 2   Function call view_rider_records()
                                         Function call admin_menu()
                                CASE 3   Function call view_feedback()
                                         Function call admin_menu()
                                CASE 4   BREAK
                              DEFAULT   Function call admin_menu()
                BOOLEAN isChar( c){
                         return string from a to z OR A to Z }
                BOOLEAN isName( name){
                         FOR i=0 to name size
                                IF name NOT isChar()
                                        return false;
                         Return true}
                BOOLEAN is_valid( email){
                          IF first element NOT isChar()
                                Return 0}
                FOR i=0 to email length{
```

Elite Ride-Ride hailing system

```
                        IF i is equal to @
                                @ is equal to i
                        ELSE IF i is equal to .
                .               is equal to i}
                IF . OR @ is equal to -1
                        Return 0
                IF @ is greater than .
                        Return 0
                Return  when . is NOT greater then email-1}}
CLASS Driver
        PRIVATE name,phone_number,email,NIC,age
        PUBLIC
                BOOLEAN driver Register(){
                        Driver enters name
                Driver enters phone_number
                         WHILE(length of phone number != 11 digits){
                                Ask the driver to enter the phone  number again}
                                 Driver enters email
                          WHILE(email NOT is_valid){
                                Ask the rider to enter the email again}
                                  Driver enter age
                        IF(age id greater than 18){
                                Driver enters NIC
                         WHILE(length of NIC != 13 digits){
                                Ask the rider to enter the NIC age}
                                 Append all the rider details to File "Registeredriders.txt"}
                                Return true
                        ELSE{  Print "Not old enough to register"}
                        Return false
CLASS Edge
        PUBLIC to , weight
        Initialize to equals to to and weight equals to weight
CLASS Graph
        PUBLIC V , adj , vertex_names ;
        Initialize V to V and resize adj to V
        FUNCTION addEdge( from, to, weight)
                 adj[from].push_back(Edge(to, weight));
        FUNCTION addVertexName( vertex, name) {
                vertex_names[vertex] equals to name;
        FUNCTION dijkstra( g, start)
                Djikstra Algorithm
CLASS Bill
        PUBLIC
        Bill(int m,int n){
                Object of Graph g(1200);
                Add 13 vertices calling g.addVertexName(place number.place name)
                Add 36 Edges    calling g.addEdge(from place,to place, fare amount)
```

Elite Ride-Ride hailing system

```
                    Cash equals to function dijkstra(g, m)
                    Output Cash
CLASS Car
          PUBLIC model,colour,capacity of passengers,next pointer

CLASS CarList{
          PRIVATE head
          PUBLIC
                    Initialize head to point to NULL
                    FUNCTION addCar(m , c , cp)
                              IF head is equal to NULL
                                        head equals to new Car();
                                        Head model,color,capacity of passengers equal to m,c cp
                                        Head next equals to NULL;
                              ELSE
                                        temp model,color,capacity of passengers equal to m,c cp
                                        temp next equals to head
                    BOOLEAN assignCar(n)
                              Temp is equals to head
                              WHILE temp NOT NULL
                                        IF capacity greater than n
                                                  Print temp color and model
                                                  Return true
                              Return false
CLASS RideLinkedList{
          CONSTRUCTOR RideLinkedList(){
                    Initialize head to NULL
          FUNCTION Add_ride (integer data){
                    IF (head == NULL)
                              Create a new node
                              Insert the new node at the head
                    ELSE
                              Create a new node and initialize it to head
                              Traverse till the end of the list
                              Insert at the the end
          FUNCTION Display_Ride(){
                     IF (head==NULL)
                               Print “No Ride ID” meaning that the list is empty
                     ELSE
                               Traverse the list and print the date of every node

          FUNCTION  delete_ride(integer d)
               IF(list is NOT empty)
                              IF(the data at head is equal to d)
                                        Delete the data at head
                              ELSE{
```

```
                              WHILE(we haven't reached the end of the list AND the data at the
     current node is
                                  not equal to d){
                                   Make the previous node equal to the current node
                                   Traverse current
                              IF(we have not reached the end of the list){
                                        Link the previous node to the next node of current
                                        Delete current
                              ELSE
                                        Print "Ride ID not found"
        ELSE          //If list is empty
                      Print "Ride ID not found
CLASS Rider
PRIVATE: String name, phone_number, email, NIC;  int age, ID;
PUBLIC:
    CarList cl;                    //Objects of classes CarList and RideLinkedList
    RideLinkedList list;
     FUNCTION Bool Register()
         Rider enters name
         Rider enters phone_number
         WHILE(length of phone number != 11 digits)
              Ask the rider to enter the phone  number again}
         Rider enters email
         WHILE(email is not valid)
              Ask the rider to enter the email again
          Rider enter age
              IF(age id greater than 18)
                  Rider enters NIC
                 WHILE(length of NIC != 13 digits)
                     Ask the rider to enter the NIC again
                 Print "You are registered"
                 Append all the rider details (name, phone_number, email, NIC, age) to
                 File "Registeredriders.txt"
                 RETURN TRUE
              ELSE
                 Print "Not old enough to register"
                 RETURN FALSE

     FUNCTION rider_register()
              Bool res=Register(); //If Register function returns TRUE call the IF statement
                          // If Register function returns FALSE call the ELSE statement
              IF(res){
                  Call get_ride FUNCTION
                  Call rider_menu FUNCTION

     FUNCTION give_feedback()
          int rating            string review
```

Elite Ride-Ride hailing system

```
            Ask rider to enter name
            Ask rider to enter rating out of 5
            IF(rating is less than 1 and greater than 5)
                Recall the give_feedback FUNCTION and ask user to try again
            ELSE
                Ask rider to enter a review
                Append these details (name, rating and review) in file "Rider_Feedback.txt"

    FUNCTION get_driver(){
            Open "Availabledrivers.txt" file which has stored the driver names
            IF (file doesn't exist)
                RETURN "Huma"        //A random name
            WHILE(file opens){
                Randomly pick any driver's name and return it
    FUNCTION get_ride()
            int random, n, pickup, destination
            Generate a random number and store it in variable random.
            Call the object of class CarList i.e cl and add different types of cars to a linked list
where each
            node has 3 datas: car model, car colour and number of passengers.
            Ask the rider to enter the number of passengers.
            IF the user has entered the number of passengers that is less than the vehicle's
maximum
            capacity give the user a list of locations to enter pickup and destination location
from.
            For pickup location call FUNCTION pick_up.
            For destination call FUNCTION des_loc.
            Print the randomly generated number (random) as Ride ID.
            Add this ID to the object of RideLinkedList class i.e list such that the ID get  stored in
a linked list
            using list.Add_ride(random) FUNCTION.
            Call the object of class Bill m1 so that the entered pickup and dropoff location.
            Goes to the Bill function as parameter.
    FUNCTION pick_up()
            Ask the rider to enter a pickup location
            IF (pickup location is NOT valid)
                Recall the FUNCTION pick_up and ask the rider to enter the pickup
                location again
            RETURN pickup    i.e the valid location
    FUNCTION des_loc()
            Ask the rider to enter a dropoff location
            IF (dropoff location is NOT valid)
                Recall the FUNCTION des_loc and ask the rider to enter the dropoff
                location again
            RETURN d    i.e the valid location
    FUNCTION cancel_ride()
            Ask the rider to enter the Ride ID they want to cancel
```

```
                    Call the object of RideLinkedList class i.e list and delete the ID from the linked
                    List by calling list.delete_ride(ID) FUNCTION
       FUNCTION rider_menu()
                    int option
                    Outputs 4 options
                    User Inputs option
                    SWITCH(option){
                     CASE 1        FUNCTION call get_ride()
                                 FUNCTION call rider_menu()
                    CASE 2          FUNCTION call Display_ride()
                                 FUNCTION call rider_menu()

                    CASE 3          FUNCTION call cancel_ride()
                                 FUNCTION call rider_menu()

                    CASE 4          BREAK
                    DEFAULT         FUNCTION call rider_menu()
CLASS App{
PRIVATE: App_name = "Elite Ride"
PUBLIC:
        Admin a1; Driver d1; Rider r1;
        FUNCTION main_menu()
                    Print welcome message
                    int option
                    Outputs 3 options (Rider, Driver, Rider , Admin )
                    User Inputs option
                    SWITCH(option)
                     CASE 1        FUNCTION call r1.rider_register()
                    CASE 2         FUNCTION call d1.driver_register()
                    CASE 3         FUNCTION call a1.login()
                                 FUNCTION call a1.admin_menu()
                    DEFAULT        FUNCTION call main_menu()
FUNCTION main()
               App app;
               FUNCTION call app.main_menu()
               Ask the user if they want to 1. call the main menu again or 0. terminate the code
               IF(User chooses 1) Recall main()
               ELSE Terminate code
```

COMPLEXITY ANALYSIS

class Admin {

1. **void login():** Complexity->O(n)
   because if..else neither increase nor decrease the runtime/complexity.

2.void admin_menu(): Complexity->O(1).}

1. **bool is_valid(string email):** Complexity->O(n).
   because the function iterates through the entire string once, checking each character for the presence of '@' and '.'.

class Driver {

1.void driver_register(): Complexity->O(n)

The while loop continues to execute until the length of the phone number variable is equal to 11, so the number of iterations is dependent on how many times the user inputs an incorrect phone number }

class Edge {

1. **Edge(int to, int weight):** Complexity->O(1).}

class Graph {

1.Graph(int V): Complexity->O(v)

where V is the number of vertices in the graph. It also resizes the adjacency list to have V elements. The resize method of the std::vector class typically takes O(V) time to resize the vector.

2.void addEdge(int from, int to, int weight): Complexity->O(1).

3.void addVertexName(int vertex, string name): Complexity->O(1).}

// Function to implement Dijkstra's algorithm

1. **vector<int> dijkstra(Graph g, int start):** Complexity-> O(E log V).
   where E is the number of edges and V is the number of vertices in the graph. This function is an implementation of Dijkstra's shortest path algorithm and takes two inputs, a graph object "g" and an integer "start" representing the starting vertex.

class Bill {

1. **Bill(int m,int n):** Complexity-> O(V^2 + E).

where V is the number of vertices and E is the number of edges in the graph. This is because the Dijkstra's algorithm used in the code.

}

class Car {

1. **void addCar(string m, string c, int cp):** complexity-> O(1).
2. **bool assignCar(int n):** complexity-> O(n)
   because the function iterates through the linked list by starting from the head and going through each element until the end of the list. }

class RideLinkedList {

1. **void Add_ride():**complexity-> O(n).
2. **void Display_ride():**complexity-> O(n).
3. **void delete_ride(int d):** complexity-> O(n). }

class Rider {

1. **bool Register():** complexity-> O(1).
2. **void rider_register():** complexity-> O(1).
3. **void give_feedback():** complexity-> O(1).
4. **string get_driver():** complexity-> O(n).
5. **void get_ride():** complexity-> O(n).
6. **int pick_up():** complexity-> O(n).
7. **int des_loc():** complexity-> O(1).
8. **void cancel_ride():** complexity-> O(n).
9. **void rider_menu():** complexity-> O(1). }

class App {

1. **void main_menu():** complexity-> O(1). }

int main(){

complexity-> O(1). }

TOTAL COMPLEXITY OF THE CODE: **O(n+V^2+E log V)**

O(n) indicating an operation with an input size of n

O(V^2 + E) indicating an operation with an input size of $V^2 + E$

O(E log V) indicating an operation with an input size of E log V

Therefore, when added together, the total time complexity of this code is

$O(n + V^2 + E + E \log V)$

OTHER APPLICATIONS

The concept of data structure linked list and filing is used in this project which can be utilized by a number of other booking systems because it includes the concept of dijkstra algorithm, inserting deleting and queue.

Some similar systems that can use this same concept are :

- Car Rental System
- Ride Sharing System
- Carpooling System
- Courier System etc

CONCLUSION AND LIMITATIONS

1)When either a rider/driver file doesn't exist to read for the admin the code eventually terminates

2)At some inputs it isn't checked if user entered an integer or a string

3)Drivers don't register with a vehicle,but vehicles are assigned according to rider's capacity of passengers

4)There's no payment class or method.

5)The admin can view only drivers and riders record and feedbacks

6)At some inputs adding space leads to code working incorrectly

7)All locations of karachi isnt covered

8)Rider/Driver can enter same information again Ride hailing system of Elite Ride is a Rider based system so it is more specific to the rider interface then driver and admin.It is assumed the rider can book multiple rides at run time for multiple passengers and cancel any of them