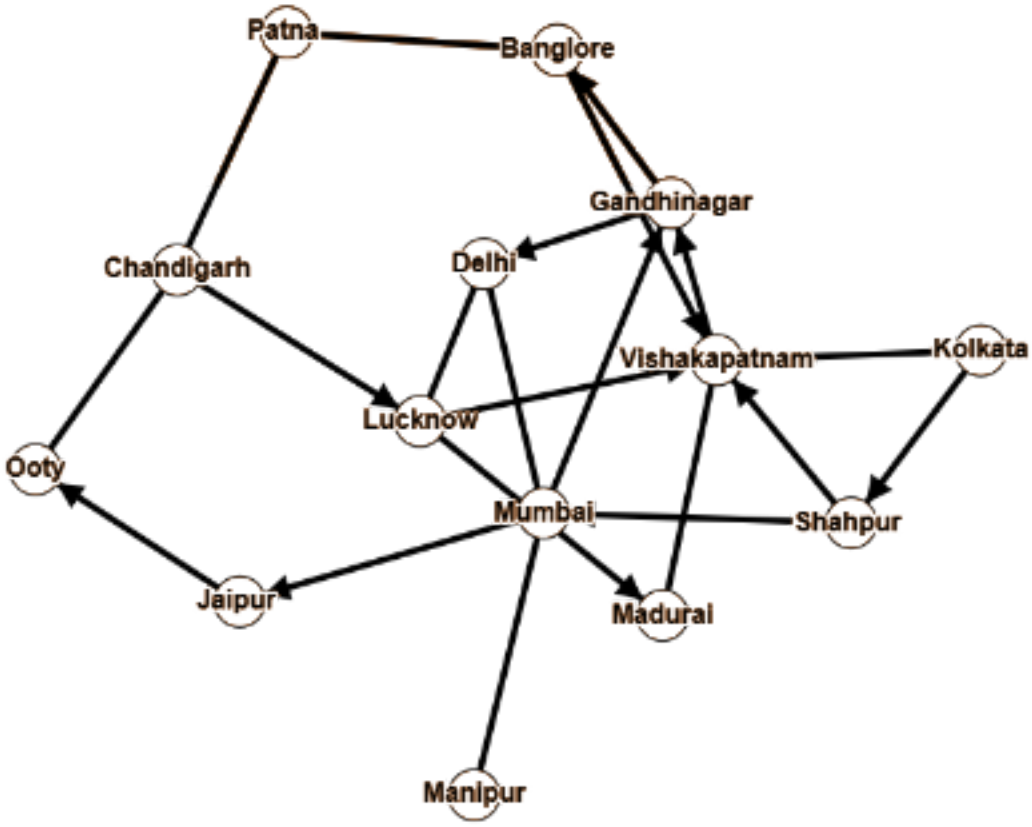
DATA STRUCTURES PROJECT REPORT

Flight Reservation System



~ The flight route graph that we have implemented

191071067 - Vaishnavi Shah
191070066 - Siddharth Shah
191070021 - Dhruvin Gandhi
191070044 - Bhavya Mehta

PROBLEM STATEMENT

Implement flight reservation system using path finder algorithm. It should have functionalities to give the path for all the flights as well as the cheapest flight, the quickest flight. It should also allow one to book and cancel the ticket.

MOTIVATION

Almost all the flight portals have various sorting and filter options based on the pricing and various other factors. But for the best customer experience, there is a need for the most suitable flight to be shown based on his/her preference. The user should be able to see the fastest flight available if there are time constraints as well as the cheapest one along with all the other flights available for him/her. Hence we have included all these features in our project.

METHODOLOGY

The cities to and from which the flights are routing, have been plotted out as a graph which is then defined using an adjacency matrix. The graph that has been implemented is the one that is shown at the beginning of this report.

There are two types of adjacency matrix, one in which the weights resemble the cost and the other resembling the duration of the flight.

Main functions used:

dijkstra() : this algorithm has been used to find the shortest path between source and destination. This path may resemble the cheapest flight or the flight with the least duration.

printAllPaths() : this function is used to print all the paths between the source and destination.

searchFlight() : This function is to take source and destination from the user and then accordingly show all the flights available as well as the filtered flights.

book() : It will book the ticket according to the flight number that will be entered by the user.

PSEUDO CODE

```
 string cities[14] ← {"Mumbai", "Delhi", "Gandhinagar", "Kolkata",
                      "Shahpur", "Visakhapatnam", "Lucknow", "Madurai",
                      "Manipur", "Jaipur", "Ooty", "Chandigarh", "Patna","Banglore"};

int adjMatrixTime[14][14] ←      {{0, 2, 5, 0, 0, 0, 0, 0, 8, 3, 0, 0, 0, 0},
                                 {2, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0},
                                 {0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9},
                                 {0, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {11, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {0, 0, 7, 8, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
                                 {0, 6, 0, 0, 0, 5, 0, 7, 0, 0, 0, 0, 0, 0},
                                 {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0},
                                 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0},
                                 {0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 10, 0, 5, 0},
                                 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 4},
                                 {0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 4, 0}};

int adjMatrixCost[14][14] ←      {{0, 40, 100, 0, 0, 0, 0, 0, 45, 60, 0, 0, 0, 0},
                                 {55, 0, 0, 0, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0},
                                 {0, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 80},
                                 {0, 0, 0, 0, 15, 65, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {10, 0, 0, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {0, 0, 50, 75, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0},
                                 {0, 25, 0, 0, 0, 30, 0, 10, 0, 0, 0, 0, 0, 0},
                                 {0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 50, 0, 0, 0},
                                 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 90, 0, 0},
                                 {0, 0, 0, 0, 0, 0, 20, 0, 0, 0, 60, 0, 45, 0},
                                 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 45, 0, 50},
                                 {0, 0, 0, 0, 0, 35, 0, 0, 0, 0, 0, 0, 15, 0}};
DECLARE : visited[14], path[14]
INITIALIZE: path_index ← 0, flightNo ← 672832, no_of_paths ← 0
vector<vector<int>> pathcost(14);
```

```
struct node
{
   DECLARE: string variables s, d, name, pass, age, date
            int ticket_id;
            struct node *next;
} *booked← NULL;

int getMinDist(int distance[], int visited[]):
   INITIALIZE int min ← I, i, vertex ← 0,V ← 14.
   FOR i ← 0 and i < V, i++:
      IF visited[i] = 0 and distance[i] <= min:
         min ← distance[i]
         vertex ← i
     END IF
   END FOR
   RETURN  vertex
   EXIT

int getLoop(int visited[])
   DECLARE int V ← 14;
   FOR int i ← 0 and i < V , i++
      IF visited[i] ← 0
         RETURN 1;
      END IF
   END FOR
   RETURN 0
   EXIT

void switchcase(int i):
   IF i = 1:
      searchFlight()
   ELSE IF i = 2:
      book("a", "b", 0, 0)
   ELSE IF i = 3:
      Delete()
   ELSE IF i = 4
      display()
   ELSE IF i = 5
      RETURN
   ELSE
      PRINT Enter some Valid choice
   END IF
   EXIT
```

```
void dijkstra(int adjMatrix[14][14], int startVertex, int destination):
    INITIALIZE int V ← 14, source ← startVertex, i, prev[14] ← {0}, distance[V], visited[V]
    vector<vector<int>> paths(14)
    FOR i ← 0 and i < V, i++:
        IF adjMatrix[startVertex][i] != 0:
            distance[i] ← adjMatrix[startVertex][i]
            prev[i]←source
        ELSE
            distance[i] ← I
             visited[i] ← 0
        END IF
    END FOR
    distance[startVertex] ← 0
    visited[startVertex] ← 1
    prev[source]← -1
    WHILE  getLoop(visited) DO:
        startVertex ← getMinDist(distance, visited)
        visited[startVertex] ← 1
        FOR i = 0 and i < V, i++:
            IF i = startVertex or i = source
                CONTINUE
            IF adjMatrix[startVertex][i] != 0 and visited[i] = 0:
                IF (distance[startVertex] + adjMatrix[startVertex][i]) < distance[i]:
                    distance[i] = distance[startVertex] + adjMatrix[startVertex][i]
                    prev[i] = startVertex
                END IF
            END IF
        END FOR
    END WHILE
    FOR i ← 0 and i < V, i++:
        paths[i].push_back(i)
        IF i!=source:
            int d = prev[i];
             WHILE d != -1 DO:
                        paths[i].push_back(d)
                        d ←prev[d]
             END WHILE
       END IF
    END FOR
```

```
   i ← destination
   INITIALIZE int u ← paths[i].size() - 1, flag ← 0;
   PRINT  cities[paths[i][u]]
   FOR u ← paths[i].size() - 2 and  u >= 0, u--:
      flag ← 1;
      PRINT cities[paths[i][u]]
   END FOR
   IF !flag:
      PRINT No flights available.
   END IF
EXIT


void printAllPathsUntil(int adj[][], int u, int d):
   INITIALIZE  visited[u] ←, path[path_index] ←U, path_index←path_index+1,pathcost ←0,
pathTime←0
    IF u = d:
      IF path_index<=7:
        flightNo←flightNo+1
        FOR  i←0 and i < path_index - 1, i++:
             pathcost ←pathcost+adjMatrixCost[path[i]][path[i + 1]]
             pathTime←pathTime+adjMatrixTime[path[i]][path[i + 1]]
        END FOR
        DISPLAY(flight schedule)
        FOR  i←0 and i<path_index - 1, i++:
              DISPLAY(path connecting the cities)
        END FOR
        DISPLAY(last path connecting to the city)
        no_of_paths←no_of_paths+1
      END IF
    ELSE:
      FOR  i ←0 and i < 14, i++:
         IF !visited[i] and adj[u][i]:
            printAllPathsUntil(adj, i, d)
      END FOR
   END IF
   path_index--
   visited[u]←0
```

```
void printAllPaths(int adj[14][14], int s, int d):
        FOR  i←0 and  i < 14,i++:
                visited[i] ←0
        END FOR
        printAllPathsUntil(adj, s, d)
        EXIT

void searchFlight():
   DECLARE i, j, l ← 0, source, destination
   DO
      DISPLAY " * Enter source city : "
      FOR  i←0 and  i < 14, i++:
         FOR j ← i and j < i + 3 ,i++:
            IF j <= 13:
               DISPLAY "Press " j + 1  "." cities[j]
            ELSE
               BREAK
            END IF
         END FOR
         i ← j - 1
      END FOR
      Take Input for source;
      source--
      IF source < 0 or source > 14:
         PRINT "PLEASE ENTER A VALID OPTION"
   WHILE source < 0 || source > 14 DO:
      Display  "* Enter destination city : "
    FOR  i←0 and  i < 14 ,i++:
         FOR j ← i and j < i + 3,i++:
            IF j <= 13:
               Display "Press " j + 1  "." cities[j]
            ELSE
               BREAK
            END IF
         END FOR
         i ← j - 1;
      END FOR
      Take Input for destination
      destination--
      IF destination < 0 or destination > 14:
         Display "PLEASE ENTER A VALID OPTION"
```

```
        IF destination = source
            Display  "PLEASE ENTER A VALID DESTINATION OTHER THAN THE SOURCE"
        END IF
WHILE  destination < 0 or destination >= 14 or  destination = source
END DO-WHILE
    PRINT  "       Searching all flights "
    PRINT  "~ FLIGHT TABLE ~"
    Display "FlightNo Time Cost of ticket Route"
    printAllPaths(adjMatrixCost, source, destination)
    Display "| The most time efficient route is : | "
    dijkstra(adjMatrixTime, source, destination);
    Display "| The most pocket friendly route is : | "
    dijkstra(adjMatrixCost, source, destination);
   DO
        Display "  ~ MENU ~ "
        Display " * Press 1.Search Flight  * Press 2.Book Flight  * Press 3.Exit"
        Display "Enter your choice : "
        Take Input for i
        IF i < 1 or  i > 3 :
            Display "PLEASE ENTER A VALID OPTION"
    WHILE i <= 0 or i >= 4
    END DO-WHILE
    IF i = 1:
        FOR j ← 0 and j < pathcost.size() j++:
            pathcost[j].clear();
        END FOR
        no_of_paths ← 0;
        searchFlight();
    IF i = 2:
        DECLARE fno
        book(cities[source], cities[destination], 1, fno)
    END IF


void display():
    DECLARE struct node *p
    IF booked != NULL:
        PRINT Details of the booked tickets in a Tabular form
    ELSE
        PRINT “Oops!No tickets booked till now !"
    END IF
        p ← booked
```

```
   WHILE p != NULL DO:
      PRINT Details of the booked tickets in a Tabular for
       p ← p->next;
   END WHILE
EXIT

void Delete():
  DECLARE  int id
   PRINT " * Enter ticket id to be canceled :"
   Take input for id
  INITIALIZE struct node *p ← booked, *q;
   WHILE p != NULL DO:
      q ← p;
      IF p->ticket_id = id:
         IF p = booked
            booked ← booked->next
            DELETE p
            PRINT “The Ticket was Cancelled”
            RETURN
         ELSE
            q->next ← p->next;
            PRINT “The Ticket was Cancelled”
            DELETE p
            RETURN
         END IF
      END IF
   END WHILE
   PRINT "   --> No such ticket with the given id found ! "
EXIT

void book(string s, string d, int def, int fno):
   DECLARE string variables name, pass, age, date
   DECLARE struct node *t, *p
   time_t now ← time(0);
   Take Input for Name
   Take Input for Passport ID
   Take Input for Age
   Take Input for Date
   IF def != 1:
      Take Input for Source City
      Take Input for Destination City
   END IF
   Take Input for Flight Number
```

```
   IF booked = NULL:
      t ← new node
      t->s ← s
      t->d ← d
      t->name ← name
      t->age ← age
      t->date ← date
      t->pass ← pass
      t->ticket_id ← now
      t->next ← NULL
      booked ← t
      DISPLAY Confirmed details of the ticket
   ELSE:
      p ← booked
     DECLARE  struct node *q
      WHILE p DO:
        q←p
        p ← p->next
      END WHILE
      t ← new node
      t->s ← s
      t->d ← d
      t->name ← name
      t->age ← age
      t->date ← date
      t->ticket_id ← now
      t->pass ← pass
      t->next ← NULL
      q->next←t
      DISPLAY Confirmed details of the ticket
   END IF
EXIT
```

OUTPUT

```
        Welcome To Flight Reservation System

   <><><><><><><><><><><><><><><><><><><><><><><><><><>
   Book your Flight tickets at affordable prices!
   <><><><><><><><><><><><><><><><><><><><><><><><><><>

                   ~~ MAIN MENU ~~

   ________________________________________________

         * Press 1.Search Flight
         * Press 2.Book Flight
         * Press 3.Cancel Flight
         * Press 4.View your Bookings
         * Press 5.Exit

   ________________________________________________

Enter your choice : 1
 * Enter source city :
    Press 1.Mumbai                Press 2.Delhi                 Press 3.Gandhinagar
    Press 4.Kolkata               Press 5.Shahpur               Press 6.Vishakapatnam
    Press 7.Lucknow               Press 8.Madurai               Press 9.Manipur
    Press 10.Jaipur               Press 11.Ooty                 Press 12.Chandigarh
    Press 13.Patna                Press 14.Banglore
Enter your choice : 1

* Enter destination city :
    Press 1.Mumbai                Press 2.Delhi                 Press 3.Gandhinagar
    Press 4.Kolkata               Press 5.Shahpur               Press 6.Vishakapatnam
    Press 7.Lucknow               Press 8.Madurai               Press 9.Manipur
    Press 10.Jaipur               Press 11.Ooty                 Press 12.Chandigarh
    Press 13.Patna                Press 14.Banglore
Enter your choice : 8
       Searching all flights ./././././
```

```
      Searching all flights ./.../.../
                                    ~~ FLIGHT TABLE ~~
-------------------------------------------------------------------------------------------------------------------------------------------+
FlightNo          Time          Cost of ticket             Route
-------------------------------------------------------------------------------------------------------------------------------------------+
672833            14 hrs          Rs.95            Mumbai -> Delhi -> Lucknow -> Vishakapatnam -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672834            15 hrs          Rs.70            Mumbai -> Delhi -> Lucknow -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672835            21 hrs          Rs.185            Mumbai -> Gandhinagar -> Delhi -> Lucknow -> Vishakapatnam -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672836            22 hrs          Rs.160            Mumbai -> Gandhinagar -> Delhi -> Lucknow -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672837            17 hrs          Rs.220            Mumbai -> Gandhinagar -> Banglore -> Vishakapatnam -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672838            36 hrs          Rs.270            Mumbai -> Gandhinagar -> Banglore -> Patna -> Chandigarh -> Lucknow -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672839            34 hrs          Rs.255            Mumbai -> Jaipur -> Coty -> Chandigarh -> Lucknow -> Vishakapatnam -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+
672840            35 hrs          Rs.230            Mumbai -> Jaipur -> Coty -> Chandigarh -> Lucknow -> Madurai
-------------------------------------------------------------------------------------------------------------------------------------------+

+-----------------------------------+
| The most time efficient route is : | Mumbai -> Delhi -> Lucknow -> Vishakapatnam -> Madurai
+-----------------------------------+
+-----------------------------------+
| The most pocket friendly route is : | Mumbai -> Delhi -> Lucknow -> Madurai
+-----------------------------------+
```

```
                                    ~ MENU ~
          ____________________________________________________

                * Press 1.Search Flight
                * Press 2.Book Flight
                * Press 3.Exit
          ____________________________________________________

Enter your choice : 2
 ~~ Welcome aboard Flier ! ~~
 * Enter your name: Bhavya
 * Enter passport id: 117552
 * Enter your age : 18
 * Enter date of travel: 12:12:2020
 * Enter flight No : 672834
Booking your flight ..........
   Your seat is booked for flight 672834 from Mumbai to Madurai with TicketId 1606030172 on 12:12:2020.
   Thank you !!
```

```
                ~~ MAIN MENU ~~
   ____________________________________________

        * Press 1.Search Flight
        * Press 2.Book Flight
        * Press 3.Cancel Flight
        * Press 4.View your Bookings
        * Press 5.Exit
   ____________________________________________

Enter your choice : 2
 ~~ Welcome aboard Flier ! ~~
 * Enter your name: Sid
 * Enter passport id: 887532
 * Enter your age : 19
 * Enter date of travel: 21:01:2021
 * Enter source city : Manipur
 * Enter destination city: Ooty
 * Enter flight No : 556432
Booking your flight ..........
   Your seat is booked for flight 556432 from Manipur to Ooty with TicketID 1606030196 on 21:01:2021.
   Thank you !!


                ~~ MAIN MENU ~~
   ____________________________________________

        * Press 1.Search Flight
        * Press 2.Book Flight
        * Press 3.Cancel Flight
        * Press 4.View your Bookings
        * Press 5.Exit
   ____________________________________________

Enter your choice : 4
                      ~~ BOOKED TICKETS ~~
-------------------------------------------------------------------------------------------------------+
 Date            TicketID           Name              Route
-------------------------------------------------------------------------------------------------------+
12:12:2020     1606030172           Bhavya         Mumbai -> Madurai
-------------------------------------------------------------------------------------------------------+
21:01:2021     1606030196             Sid         Manipur -> Ooty
-------------------------------------------------------------------------------------------------------+
```

```
                ~~ MAIN MENU ~~
  ____________________________________________

        * Press 1.Search Flight
        * Press 2.Book Flight
        * Press 3.Cancel Flight
        * Press 4.View your Bookings
        * Press 5.Exit
  ____________________________________________

Enter your choice : 3
 ~~ Welcome aboard Flier ! ~~
 * Enter ticket id to be canceled :1606030172
   The given ticket was canceled !
    *Thank You

                ~~ MAIN MENU ~~
  ____________________________________________

        * Press 1.Search Flight
        * Press 2.Book Flight
        * Press 3.Cancel Flight
        * Press 4.View your Bookings
        * Press 5.Exit
  ____________________________________________

Enter your choice : 4
                      ~~ BOOKED TICKETS ~~
-------------------------------------------------------------------------------------------------------+
 Date            TicketID           Name               Route
-------------------------------------------------------------------------------------------------------+
21:01:2021     1606030196              Sid        Manipur -> Ooty
-------------------------------------------------------------------------------------------------------+


                ~~ MAIN MENU ~~
  ____________________________________________

        * Press 1.Search Flight
        * Press 2.Book Flight
        * Press 3.Cancel Flight
        * Press 4.View your Bookings
        * Press 5.Exit
  ____________________________________________

Enter your choice : 5
                              Exiting ..........
                              Thank You, Have a Nice day ! :D
```

DISCUSSION

Flight reservation is one of the most common real-time applications which uses graphs. The perplexing complexity of this application made the task more and more interesting. Our aim was to create an application which will do easy routing of flights between cities and book tickets for whichever flight route the user decides to take, after viewing all the available options as well as the filtered options. We have tried to make this system as efficient as possible. And used multiple graph based algorithms to compute the paths. Dijkstra to find the cheapest and the quickest flight. Depth first search to compute all the paths. Linked List for storing the data of the tickets that have been used.

The EM of the Flight schedule application involves modelling the data structure and execution steps for algorithms.The data of the application are vertices and edges.

This project not only helped us revise our data structures and algorithms but also allowed us to experience a small implementation of real time application. And provided us with an opportunity to not only construct such an application under severe time constraints but also learn the process and understanding that goes behind such work.