San José State University

College of Engineering/Computer Engineering Department

PROJECT REPORT

DISTRIBUTED AIRLINE RESERVATION SYSTEM USING HYBRID SERVER

GROUP-11

HARSHA SANJEEVA
ROOP KUMAR SRIRAMULU
RUTHRA VADIVEL MURUGESAN

Under the guidance of

Dr. Rod Fatoohi

INDIVIDUAL CONTRIBUTION

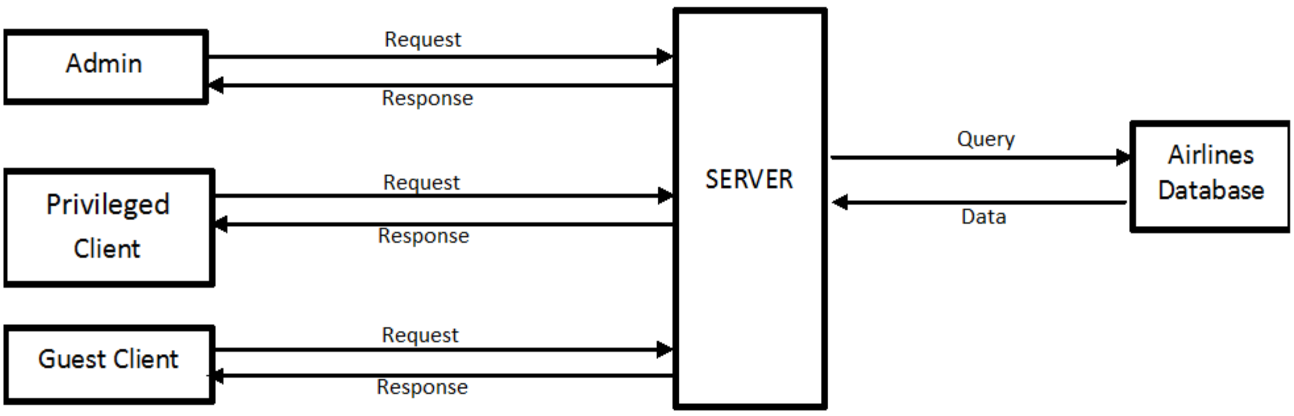
MEMBER	CONTRIBUTION
HARSHA SANJEEVA (011478870)	• Design finalizations. • Requirements specifications. • Code program. • Features: API (BSD Sockets), Multithreading. • Report documentation. • Preparation of Presentation slides.
ROOP KUMAR SRIRAMULU (011431212)	• Design finalizations. • Requirements specifications. • Code program. • Features: Multiprocessing, Secure Hash Algorithm, Password Masking. • Report documentation. • Preparation of Presentation slides.
RUTHRA VADIVEL MURUGESAN (011460332)	• Design finalizations. • Requirements specifications. • Code program. • Features: Zombies Handling, Database, IPv6. • Report documentation. • Preparation of Presentation slides.

Table of Contents

1. INTRODUCTION

The project "DISTRIBUTED AIRLINE RESERVATION SYSTEM" aims at creating a customer portal for booking airlines tickets. There are three major components involved in the project – Server, Client and Database. The reservation system is coded in the C programming language with BSD sockets used for the establishment of connection between the server and the client. In this system, the participants or the clients can book tickets for their domestic destination within this country by the interaction with the server, which uses the database to help the client in the successful booking of the tickets.

2. DESIGN



3. DESIGN DESCRIPTION

The server would accept the request from the client, process it, then send a request to the **airline database** based on the information provided by the client. The **database** would fetch all the data related to information provided by the server, which in turn sends it to the client from which it received the request. The client (customer) would choose the flight based on their choice of the available flights for that date and destination. After that successful ticket booking, would be done displaying the amount that must be paid.

4. CHARACTERISTICS OF THE DESIGN COMPONENTS

Server:

- Concurrent connection-oriented Multiprocessing and Multithreading server.
- Has connection to Guest client, Privileged client, Admin and database.
- Handles the entire airline reservation system with the help of database.
- Checks whether the user is an Admin, Privileged Customer or Guest and takes control over individual client.
- Validation of the Admin or Privileged Client using valid credentials (Username and Password).

- Gets the input from the Client and searches the database based on the key value given by the Client.
- Send the data fetched from the database to the Client.
- Upon receiving the intended flight confirmed from the Client display the fare and book the tickets successfully.
- Validates the personal details mentioned by the clients for booking tickets are in correct format.
- Error handling for various scenarios.

Admin:

- VIEW: View the flight details from the Airline database.
- INSERT: Add new flight details to the Airline database.
- DELETE: Remove flight details from the Airline database.
- UPDATE: Modify flight details in the Airline database.

Privileged Client:

- Member of the system.
- Login with valid credentials.
- Search flights for intended place of travel and date of travel.
- View the multiple flights available for the intended place of travel and date of travel and view the normal and discounted fares for the respective flights.
- Will have access to the discounts in the flight rates.
- Can request for one-way or return trip.
- Book the ticket successfully.

Guest Client:

- Client can book the ticket without having the login credentials.
- Search flights for intended place of travel and date of travel.
- View the multiple flights available for the intended place of travel and date of travel
- and view the normal fares for the respective flights.
- The client will not be given discount details of the flight.
- Can request for one-way or return trip.
- Book the ticket successfully.

Airline database:

- It contains all the information such as Flight number, Flight name, Destination, Place of Arrival, Departure date, return date, number of seats to be booked and the fares of the flights.
- Active communication with server.
- It will be contacted only by the server and can be updated only by the admin.

5. DETAILED DESCRIPTION ABOUT THE COMPONENTS

Server:

The server designed is a concurrent connection-oriented multiprocessing and multithreading server, which will handle the privileged and guest clients request by creating a new child process and the admin's request by creating a thread. The server is interfaced with the database and it will fetch the data from the database for the ticket reservation using the inputs given by the client.

Server will be listening at a port number waiting for the new connection from the client. When the client establishes a successful connection with the server, the server would prompt the user to choose whether the user is a admin or a privileged user or guest. Based on the choice of the user the server would behave accordingly.

In the case when the user is an admin, the server would create a new thread and validates the admin, using the admin credentials (Userid and Password). When the server confirms that the user is an admin, the server provides the options for the admin to view, insert, update or delete data in the database. The admin will have access to the database only through the server. The server must get the values from the client for the fields from which the data must be viewed, updated, deleted or inserted based on the options provided by the server. The server will have all the query to be executed to fetch values from the database. The server will have the query based on the primary key of the table.

For viewing the data in the database, the server has the following query,

SELECT * FROM air_lines;

For inserting the data in the database, the server has the following query,

INSERT INTO air_lines

Values (*"12 different values for all the 12 fields in the table"*);

The 12 different field values to be inserted will be obtained from the admin as input.

For updating the data in the database, the server has the following query,

UPDATE air_lines

SET Seats= *"value from admin"*

WHERE Flight_No = *"value from admin"*;

In this query, the primary key will Flight Number as it is the only unique value in the table.

Also, the admin will have access only to update the seat numbers using this query. It is designed in such a manner only for the sake of simplicity.

For deleting the data in the database, the server has the following query,

DELETE FROM air_lines

WHERE Flight_No = *"value from admin"*;

In this query, the primary key will Flight Number as it is the only unique value in the table.

Once the task has been completed, the server would ask whether the admin needs to perform any one more and based on the reply from the admin, the flow would continue.

The server is integrated with the database using MYSQL and C integrating commands, and it is the only point of access to the database.

When the user is a privileged customer, the server would create a new child process and validate the user, whether the user is a member of the system, by validating the user credentials (Username and Password). After successful login, the server would redirect the client to code that is designed for the interaction. In this interaction, the server would gain all the relevant details such as the type of trip the user prefers, following that the details of the departure and destination places, and the dates of travel. Having obtained these details from the client the server would send queries to the database to fetch the data and will display the flight details to the client. The server would then prompt for the booking details from the client if the client is willing to book tickets. Also, the server calculates the fare based on the discounted rate in the database as the client is a member of the reservation system.

Now, if the user is a guest client, the would perform the same tasks as it will do it for the privileged member by creating a new child process except for the validation of the login credentials. As the user is a guest, the user will not be member of the system and so the server will not have to validate any login credentials. The guest user will not be able to see the discounted rates. So, the server calculates the fare based on the normal fare in the database and not on the discounted fare.

The server also validates whether the personal details mentioned by the clients are in correct format. The validation formats are:

- Age must be an integer,
- Number of seats must be an integer,
- Phone number must be only 10 characters.

If the data entered is not in the correct format, then the server would send error messages to the client until it gets the data in the correct format.

Successful booking of ticket by the server

For this to happen, the server must have fetched all the data pertaining to the client's requirements from the database. These data must be sent to the client.

The client must choose which itinerary is best for him and must tell the server to book the ticket. (Proceed to next step).

The server will then ask for the following details to book the ticket.

The following things need to be collected from the client (privilege and guest) to book a ticket after the data is being fetched from the client.

- Name
- Age
- Flight number (From the list to be selected)
- No. of seats needed
- E-mail address.

Payment calculation

For Guest:

- Amount= No. of seats * Cost of one ticket.

For member:

- Amount= No. of seats * Discount rate.

After obtaining all the inputs, the server will book the ticket and will display "Ticket booked successfully, Enjoy your Trip !!!!"

Privileged User:

Privileged user is a member of the reservation system. In this case, the user would be prompted to enter the username and password credentials. After the validation of the credentials, the user would be redirected to select the type of trip, which is One way or Round-Trip travel. For example, if the user selects one-way travel then, server would ask for some details pertaining to the one-way travel such as only the departure date and not the return date. After that the user would be asked to input the place of travel and the destination. Then server would prompt to enter the departure date and return date (if applicable). Privileged user is given a special provision, where special discount would be offered.

Guest User:

The guest user is a client who is not a member of the reservation system, so the user will not have any login credentials and will directly search for the flight details. The procedure of booking the ticket remains the same for the guest as it is for the privileged client. The guest-user will be able to view all the details of the database except for the discount rate and so will not have the provision to avail it.

Admin:

Admin is one of the clients, who has access to the database. When the server recognizes the client as an Admin, the server prompts the admin to login to the system using login credentials. The any of the following. admin can access the database only through the server. After the validation of the credentials, the admin will be redirected to access the database specific code, the admin will be given options by the server to perform

- View
- Insert
- Update
- Delete

Out of these options the admin can choose one option / or more options one by one. So that the admin will have the control to do the changes one at a time.

Database:

The database, which stores all the details related to the ticket reservation, is created using MYSQL. There are 3 different tables created for storing different data in the table. The database is integrated with the server using MYSQL and C integrating commands. The database created is named "**airlines**", which has the following tables

1. "**air_lines**", contains all the details necessary for booking the ticket. The database created consists of 12 fields. The database can be accessed only by the server.
2. "**pass**", includes the password of the admin and the privileged clients.
3. "**people**", contains the details of the people, who have booked the flight.

6. <u>FEATURES IMPLEMENTED</u>

<u>Network Features:</u>

1. BSD Sockets:

The API used for the connecting the server and client is BSD sockets.

2. Concurrent Multithreading Server:

The hybrid server designed has concurrent connection-oriented multithreading as one of its feature. The server will handle each admin client using a separate thread. The child thread will be created by the server when a new connection is accepted by the server by using the thread creation function.

3. Concurrent Multiprocessing Server:

The hybrid server designed has concurrent connection-oriented Multiprocessing as one of its feature. The server will handle each member client using a separate child. The child process will be created by the server when a new connection is accepted by the server by calling **fork()** function.

4. **Handling Zombies using Signal handler:**

The code has the functionality to handle zombies. As the entry of the defunct child would have access to the records inside the descriptor table, which is a potential threat, the entry is removed or handled by the signal handlers implemented in the code. So as to exit the child in a right way.

The presence or absence of zombies can be verified by giving the following command

ps -aux | grep "*servername*"

"Servername" is the name of the server.

5. **Database:**

The database implemented is created using MYSQL. The server interacts with the database to help the client in booking tickets.

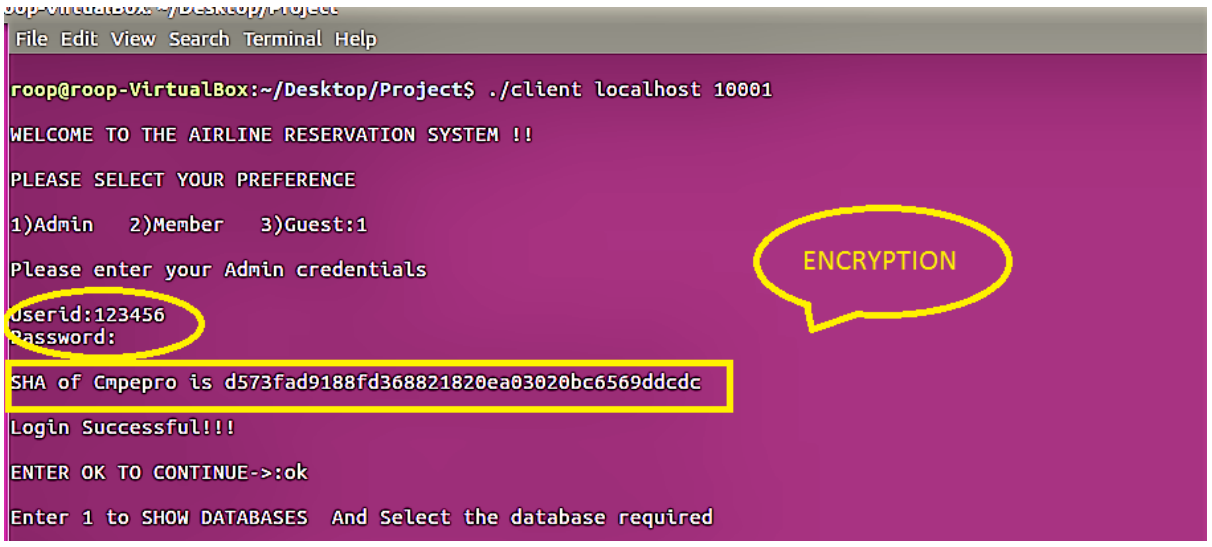
Database: airlines
Table: air_lines
Table: pass
Table: people.

6. **IPv6:**

The system designed is an IPv6 compatible system, wherein the server and the client are connected to each other using the loopback address (::1 or 0:0:0:0:0:0:0:1).

Security features:

The password that is being entered by the clients (admin and privileged) are masked and encrypted for security purpose and stored in the database. This encryption is done using the Secure Hash Algorithm (SHA).



```
roop@roop-VirtualBox:~/Desktop/Project$ ./client localhost 10001
WELCOME TO THE AIRLINE RESERVATION SYSTEM !!
PLEASE SELECT YOUR PREFERENCE
1)Admin   2)Member   3)Guest:1
Please enter your Admin credentials
Userid:123456
Password:
SHA of Cmpepro is d573fad9188fd368821820ea03020bc6569ddcdc
Login Successful!!!
ENTER OK TO CONTINUE->:ok
Enter 1 to SHOW DATABASES  And Select the database required
```

7. **Secure Hash Algorithm 1 (SHA-1)**

For encrypting the passwords while transmitting a cryptographic hash function named Secure Hash Algorithm is used. It was developed by National Security Agency. SHA-1 function is included in the "openssl" and "crypto" C library files. When a string to the SHA function is passed, it gives a 160-bit output. This output will be in the form of a 40-digit hash value called as message digest.

Hash Values conversion of the password data present in the pass table.

SHA1("Cmpepro")

gives hexadecimal: d573fad9188fd368821820ea03020bc6569ddcdc

SHA1("Network")

gives hexadecimal: 53ebc572b4a44802ba114729f07bdaaf5409a9d7

The hash of the zero-length string is:

SHA1("")

gives hexadecimal: da39a3ee5e6b4b0d3255bfef95601890afd80709

On the server side, this 40-digit hash value is matched with the stored hash data (Created during member registration) in the database and the password is validated.

8. **Password Masking**

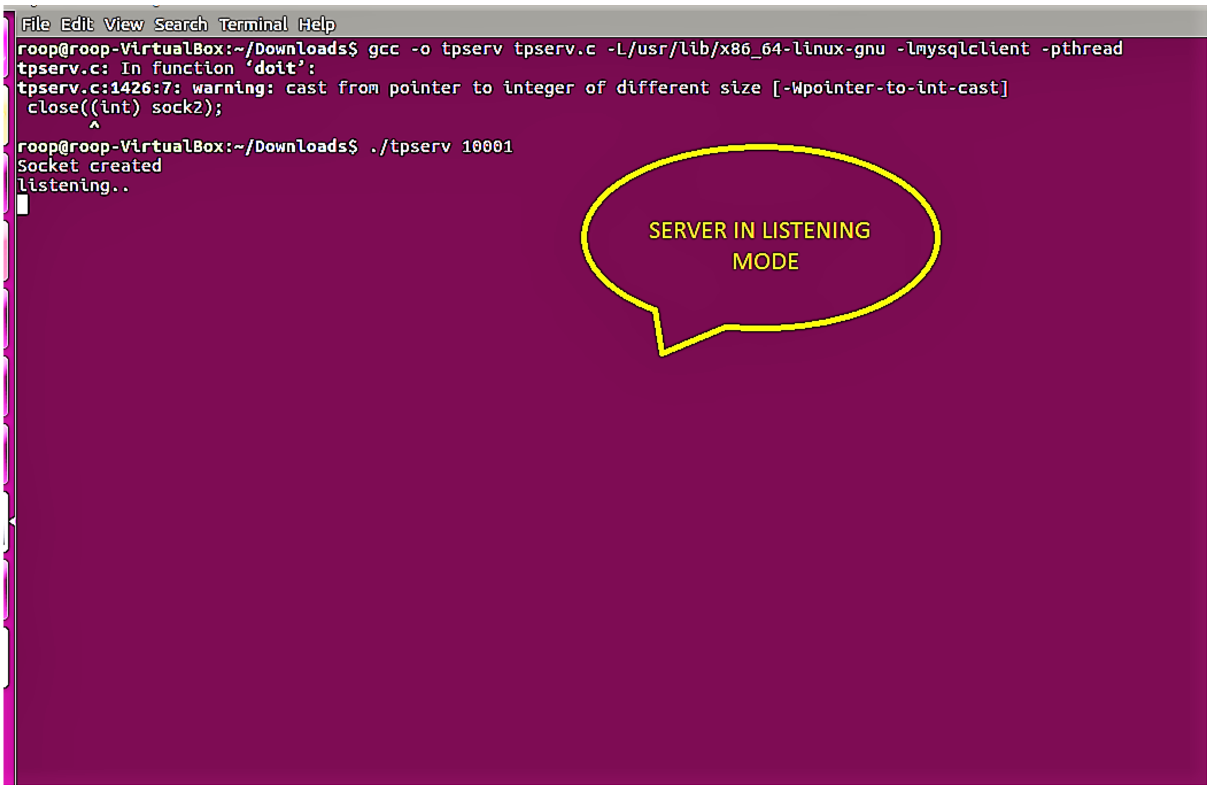
The password entered by the users on their screens are masked or made invisible using the password masking feature. This is done using the Linux bash functions written in the bash script. This bash script is run by the C program. There are basically only two functions involved "read" and "echo". An option of read, "s" is used to mask the string entered on-screen and an echo function is used to directly echo it into a text file. This text file, now temporarily contains the 40-digit hash function which is file transferred to the server.

7. STEPS INVOLVED IN THE INTERACTION BETWEEN SERVER AND CLIENT

Server:

The server is compiled and placed in the listening stage to accept the connections.

```
gcc -o server server.c
./server 10001
```



```
roop@roop-VirtualBox:~/Downloads$ gcc -o tpserv tpserv.c -L/usr/lib/x86_64-linux-gnu -lmysqlclient -pthread
tpserv.c: In function ‘doit’:
tpserv.c:1426:7: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
 close((int) sock2);
       ^
roop@roop-VirtualBox:~/Downloads$ ./tpserv 10001
Socket created
listening..
```

Clients:
Admin

The Admin client is connected to the server and enters the details asked by the server.

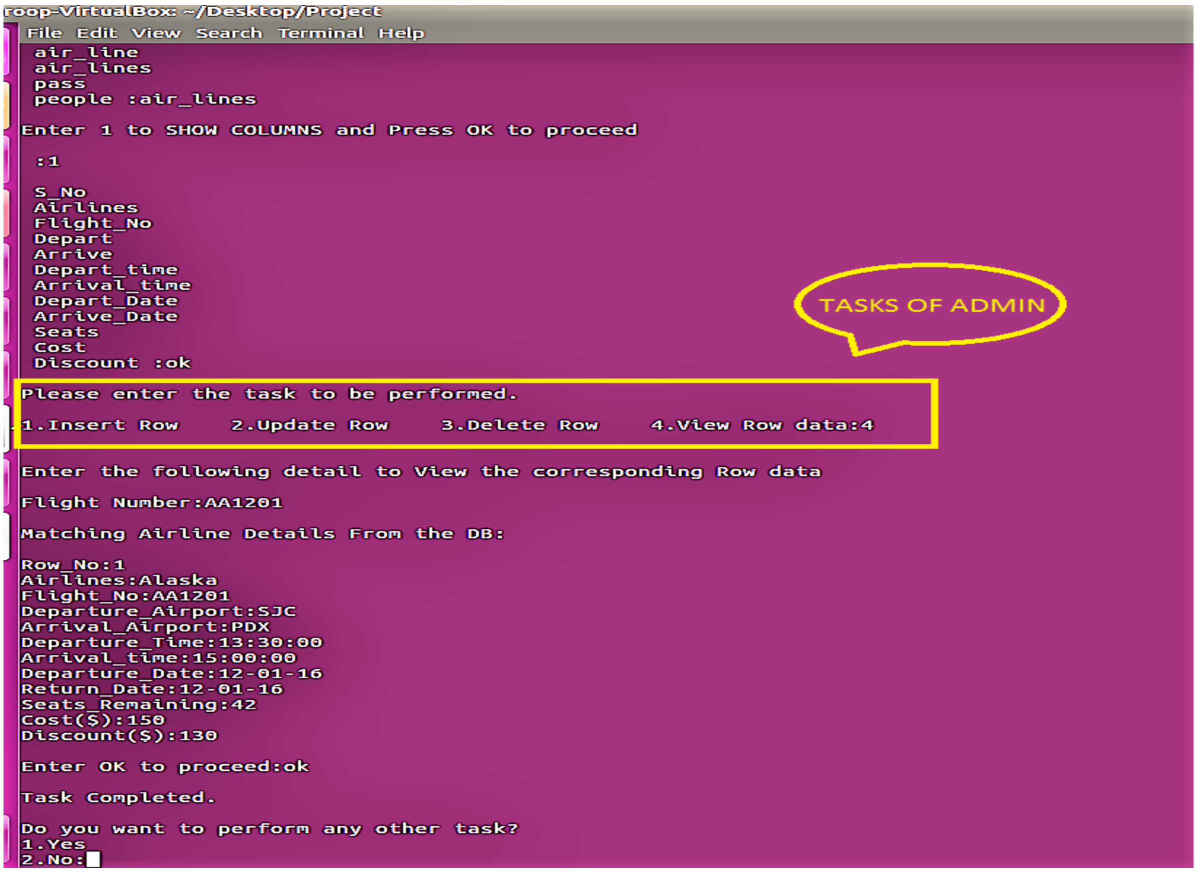
gcc -o admin admin.c
./admin ::1 10001

After getting connected to the server, the server will ask the client
“What type of client are you?
1.Admin
2.Privileged
3.Guest

The client replies
1

Then the server will ask the client to enter the login credentials. If the validation of the credentials is success, then the admin will be notified of that.
The login details are

Userid : 123456
Password: Cmpepro


roop-VirtualBox: ~/Desktop/Project
File Edit View Search Terminal Help
TASKS OF ADMIN

```
 air_line
 air_lines
 pass
 people :air_lines
Enter 1 to SHOW COLUMNS and Press OK to proceed

 :1

 S_No
 Airlines
 Flight_No
 Depart
 Arrive
 Depart_time
 Arrival_time
 Depart_Date
 Arrive_Date
 Seats
 Cost
 Discount :ok
Please enter the task to be performed.

1.Insert Row     2.Update Row     3.Delete Row     4.View Row data:4

Enter the following detail to View the corresponding Row data

Flight Number:AA1201

Matching Airline Details From the DB:

Row_No:1
Airlines:Alaska
Flight_No:AA1201
Departure_Airport:SJC
Arrival_Airport:PDX
Departure_Time:13:30:00
Arrival_time:15:00:00
Departure_Date:12-01-16
Return_Date:12-01-16
Seats_Remaining:42
Cost($):150
Discount($):130

Enter OK to proceed:ok

Task Completed.

Do you want to perform any other task?
1.Yes
2.No:
```

Privileged User:

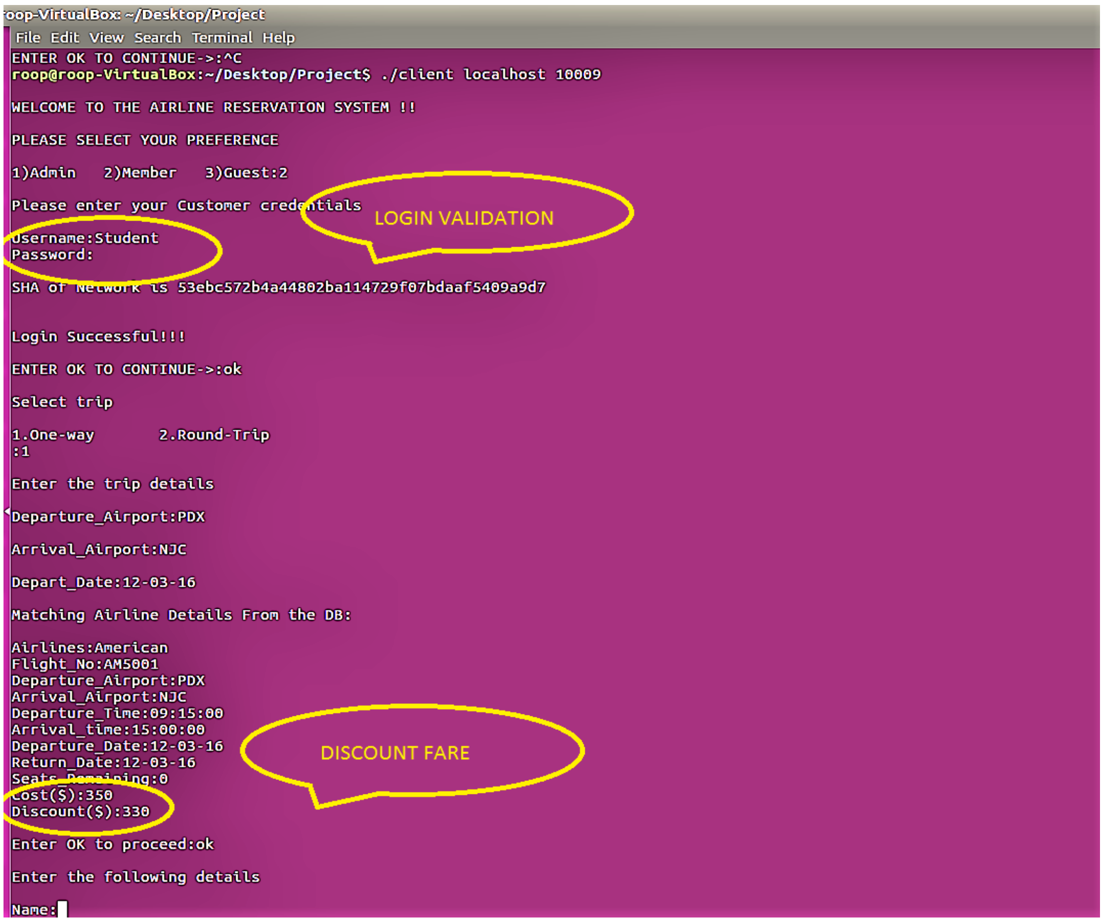
The Privileged client is connected to the server and enters the details asked by the server.

gcc -o privileged privileged.c
./privileged ::1 10001

After getting connected to the server, the server will ask the client
"What type of client are you?
1.Admin
2.Privileged
3.Guest

The client replies
2

Then the server will ask the client to enter the login credentials. If the validation of the credentials is success, then the client will be notified of that and the server will ask the client to select the type of trip and get the details based on the trip type selected.
The login credentials are
Username: Student
Password: Network


oop-VirtualBox: ~/Desktop/Project
File Edit View Search Terminal Help
LOGIN VALIDATION
DISCOUNT FARE

```
ENTER OK TO CONTINUE->:^C
roop@roop-VirtualBox:~/Desktop/Project$ ./client localhost 10009

WELCOME TO THE AIRLINE RESERVATION SYSTEM !!

PLEASE SELECT YOUR PREFERENCE

1)Admin   2)Member   3)Guest:2

Please enter your Customer credentials

Username:Student
Password:

SHA of Network is 53ebc572b4a44802ba114729f07bdaaf5409a9d7


Login Successful!!!

ENTER OK TO CONTINUE->:ok

Select trip

1.One-way       2.Round-Trip
:1

Enter the trip details

Departure_Airport:PDX

Arrival_Airport:NJC

Depart_Date:12-03-16

Matching Airline Details From the DB:

Airlines:American
Flight_No:AM5001
Departure_Airport:PDX
Arrival_Airport:NJC
Departure_Time:09:15:00
Arrival_time:15:00:00
Departure_Date:12-03-16
Return_Date:12-03-16
Seats_Remaining:0
Cost($):350
Discount($):330

Enter OK to proceed:ok

Enter the following details

Name:
```

Guest User:

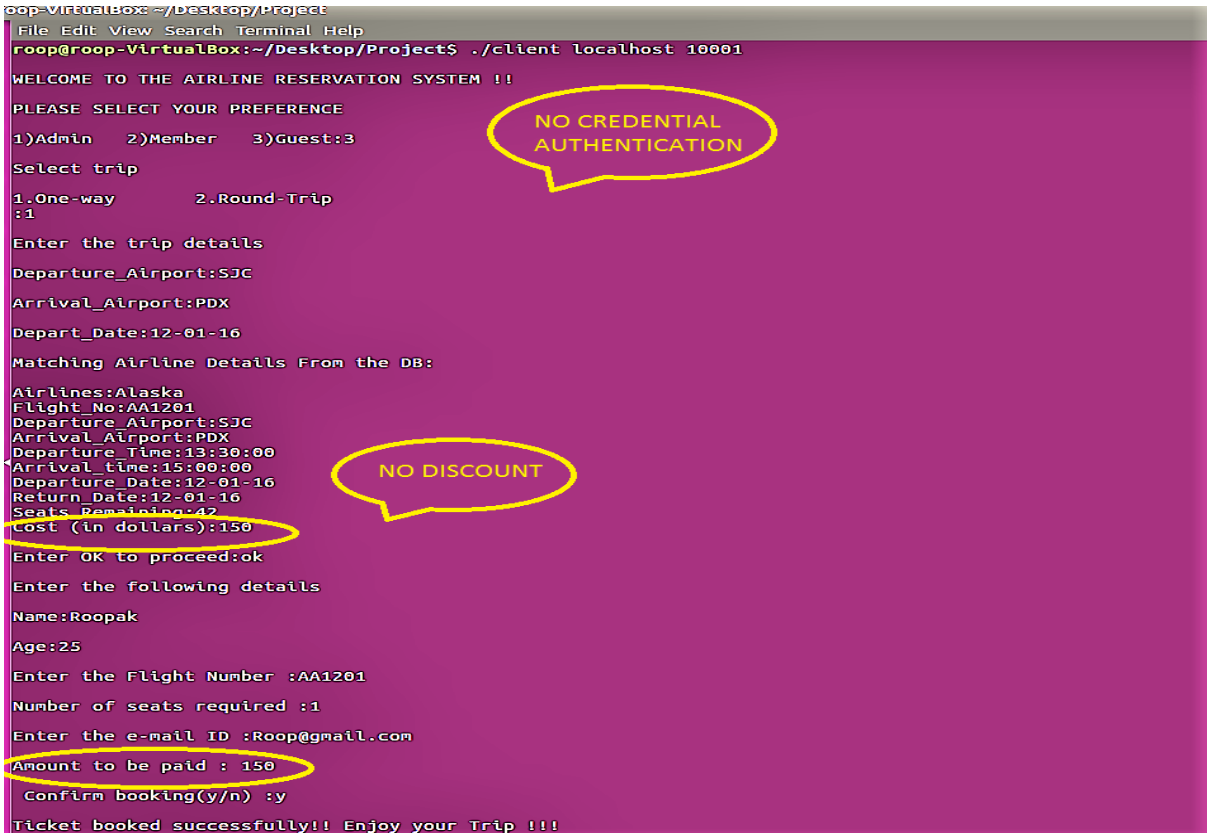
The Guest client is connected to the server and enters the details asked by the server.

gcc -o guest guest.c
./guest ::1 10001

After getting connected to the server, the server will ask the client
“What type of client are you?
1.Admin
2.Privileged
3.Guest

The client replies
3

As the client is a guest, the user will not have any login validations. The server will ask the client to select the type of trip and get the details based on the trip type selected directly.

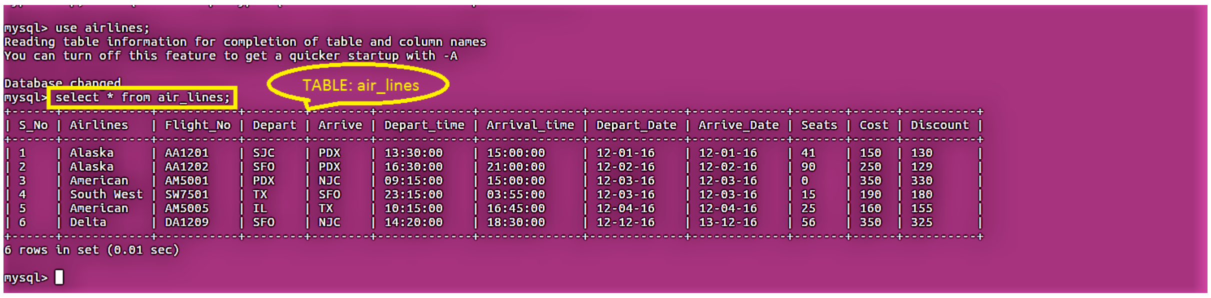


```
File  Edit  View  Search  Terminal  Help
roop@roop-VirtualBox:~/Desktop/Project$ ./client localhost 10001
WELCOME TO THE AIRLINE RESERVATION SYSTEM !!
PLEASE SELECT YOUR PREFERENCE
1)Admin   2)Member   3)Guest:3
Select trip
1.One-way        2.Round-Trip
:1
Enter the trip details
Departure_Airport:SJC
Arrival_Airport:PDX
Depart_Date:12-01-16
Matching Airline Details From the DB:
Airlines:Alaska
Flight_No:AA1201
Departure_Airport:SJC
Arrival_Airport:PDX
Departure_Time:13:30:00
Arrival_time:15:00:00
Departure_Date:12-01-16
Return_Date:12-01-16
Seats Remaining:42
Cost (in dollars):150
Enter OK to proceed:ok
Enter the following details
Name:Roopak
Age:25
Enter the Flight Number :AA1201
Number of seats required :1
Enter the e-mail ID :Roop@gmail.com
Amount to be paid : 150
 Confirm booking(y/n) :y
Ticket booked successfully!! Enjoy your Trip !!!
```

Database:

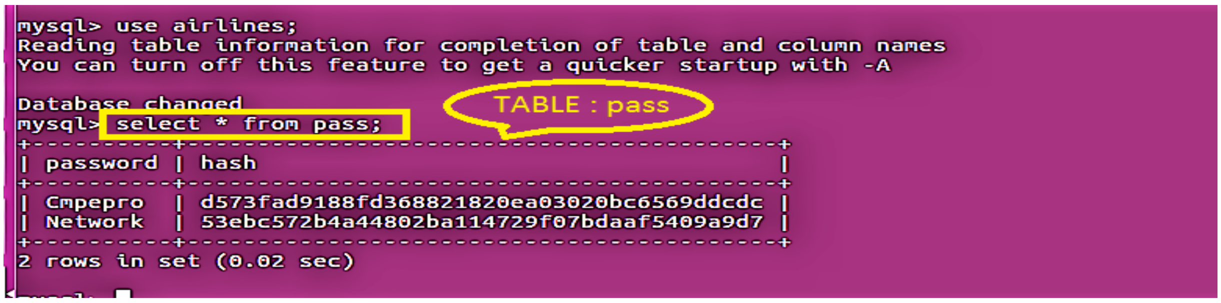
The database is created using MYSQL. The database contains following 3 different tables.

Database: airlines.
Table: air_lines



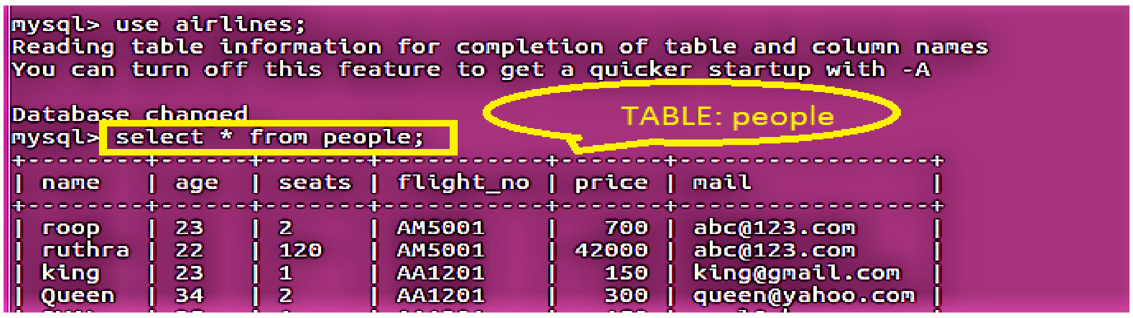
```
mysql> use airlines;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from air_lines;
+------+------------+-----------+--------+--------+-------------+--------------+-------------+-------------+-------+------+----------+
| S_No | Airlines   | Flight_No | Depart | Arrive | Depart_time | Arrival_time | Depart_Date | Arrive_Date | Seats | Cost | Discount |
+------+------------+-----------+--------+--------+-------------+--------------+-------------+-------------+-------+------+----------+
| 1    | Alaska     | AA1201    | SJC    | PDX    | 13:30:00    | 15:00:00     | 12-01-16    | 12-01-16    | 41    | 150  | 130      |
| 2    | Alaska     | AA1202    | SFO    | PDX    | 16:30:00    | 21:00:00     | 12-02-16    | 12-02-16    | 90    | 250  | 129      |
| 3    | American   | AM5001    | PDX    | NJC    | 09:15:00    | 15:00:00     | 12-03-16    | 12-03-16    | 0     | 350  | 330      |
| 4    | South West | SW7501    | TX     | SFO    | 23:15:00    | 03:55:00     | 12-03-16    | 12-03-16    | 15    | 190  | 180      |
| 5    | American   | AM5005    | IL     | TX     | 10:15:00    | 16:45:00     | 12-04-16    | 12-04-16    | 25    | 160  | 155      |
| 6    | Delta      | DA1209    | SFO    | NJC    | 14:20:00    | 18:30:00     | 12-12-16    | 13-12-16    | 56    | 350  | 325      |
+------+------------+-----------+--------+--------+-------------+--------------+-------------+-------------+-------+------+----------+
6 rows in set (0.01 sec)

mysql> 
```

Database: airlines.
Table: pass.



```
mysql> use airlines;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from pass;
+----------+------------------------------------------+
| password | hash                                     |
+----------+------------------------------------------+
| Cmpepro  | d573fad9188fd368821820ea03020bc6569ddcdc |
| Network  | 53ebc572b4a44802ba114729f07bdaaf5409a9d7 |
+----------+------------------------------------------+
2 rows in set (0.02 sec)
```

Database: airlines.
Table: people.



```
mysql> use airlines;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from people;
+--------+------+-------+-----------+-------+-----------------+
| name   | age  | seats | flight_no | price | mail            |
+--------+------+-------+-----------+-------+-----------------+
| roop   | 23   | 2     | AM5001    |   700 | abc@123.com     |
| ruthra | 22   | 120   | AM5001    | 42000 | abc@123.com     |
| king   | 23   | 1     | AA1201    |   150 | king@gmail.com  |
| Queen  | 34   | 2     | AA1201    |   300 | queen@yahoo.com |
```

8. ERROR-HANDLING

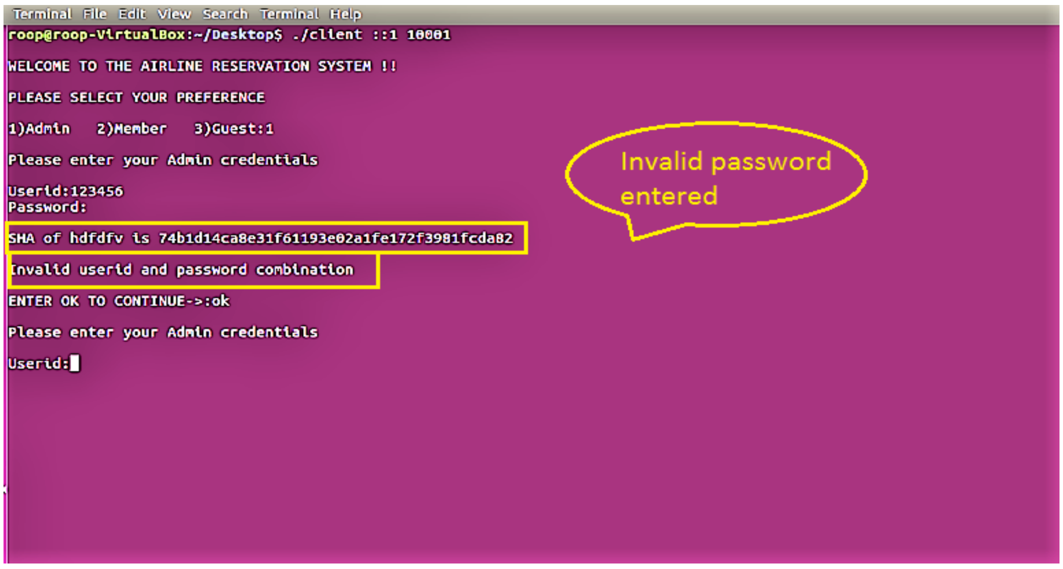
The server, in addition to all the other features, also has the provision to handle errors. Error handling is done at almost all parts of the programming. In this way, many errors that may being committed by the client will be solved by the server and makes the client to book tickets in a successful way.

One of the major error handling parts are **user credentials validation**. As the admin and privileged client has user credentials to be validated, the server also has this provision to handle the error created while entering wrong username/userid and credentials. In this way the admin and user will be given many chances until they type the correct credentials.

When the user types wrong username or password, the server will give the message

"Invalid username and password combination" and instructs the user to provide correct details.

Screenshot:

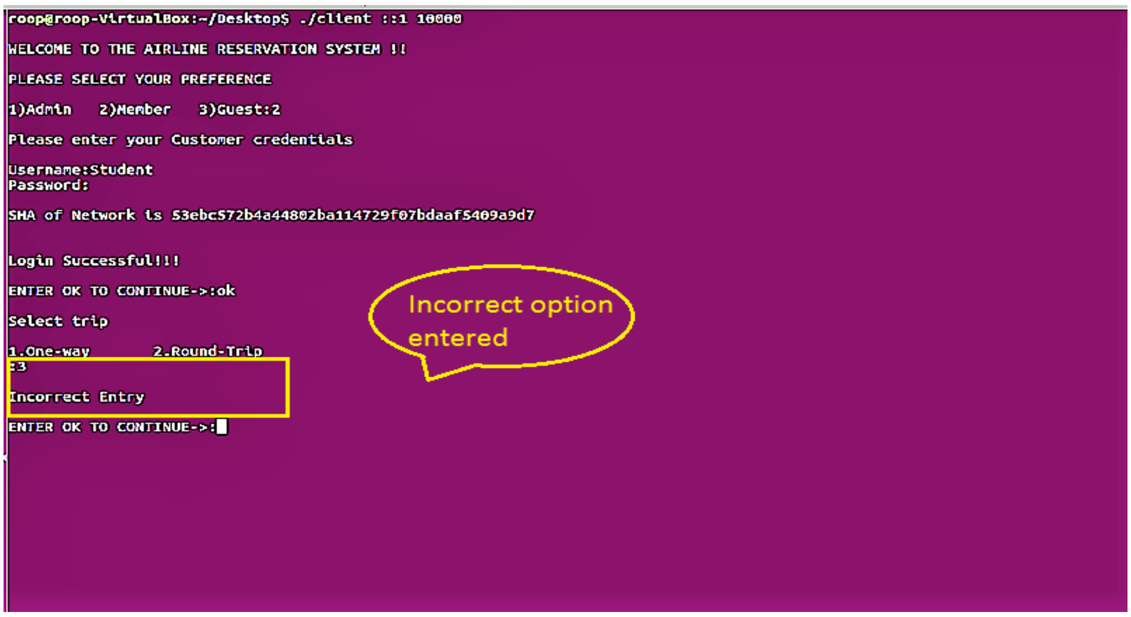


```
roop@roop-VirtualBox:~/Desktop$ ./client ::1 10001

WELCOME TO THE AIRLINE RESERVATION SYSTEM !!

PLEASE SELECT YOUR PREFERENCE

1)Admin   2)Member   3)Guest:1

Please enter your Admin credentials

Userid:123456
Password:

SHA of hdfdfv is 74b1d14ca8e31f61193e02a1fe172f3981fcda82

Invalid userid and password combination

ENTER OK TO CONTINUE->:ok

Please enter your Admin credentials

Userid:
```

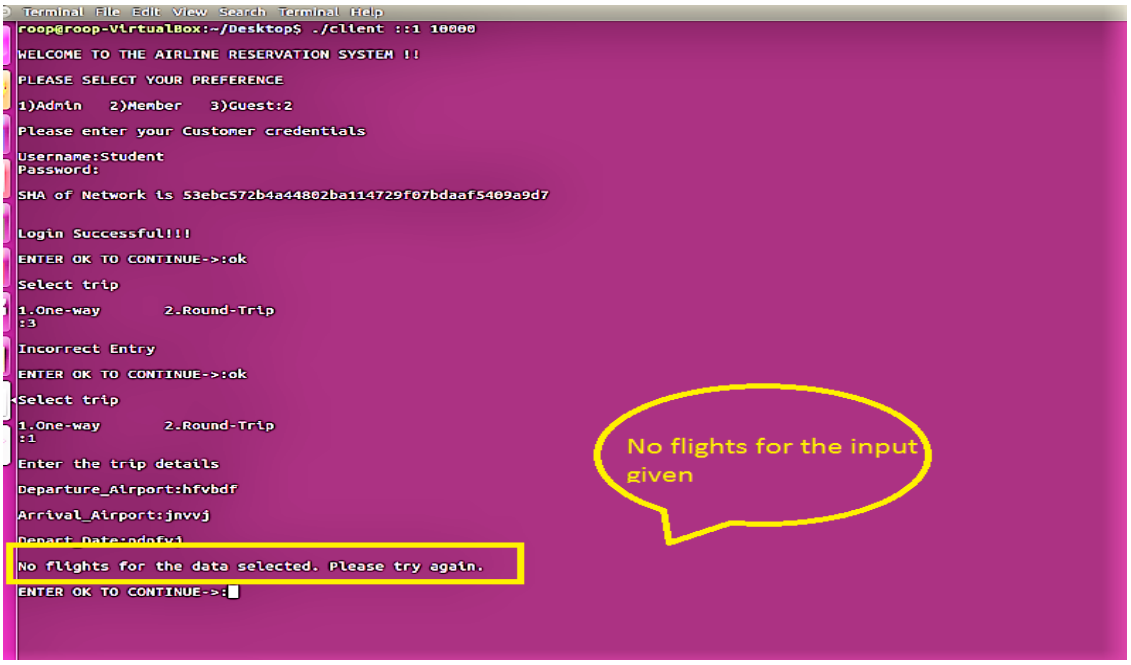
Next, important error handling part is the **validation of the privilege or guest client's personal details** entered in the booking section. In this section, when the client enters invalid data or data in invalid format the server will send error to the client to correct those errors. This may include age, gender, number of seats and phone number validation. If any error occurs in any of the details, the server will handle those errors and get the correct details from the client.

The server will also handle the error that is created, when the user asks for the **data that is not present in the database**.

<u>Screenshot:</u>



```
roop@roop-VirtualBox:~/Desktop$ ./client ::1 10000
WELCOME TO THE AIRLINE RESERVATION SYSTEM !!
PLEASE SELECT YOUR PREFERENCE
1)Admin   2)Member   3)Guest:2
Please enter your Customer credentials
Username:Student
Password:
SHA of Network is 53ebc572b4a44802ba114729f07bdaaf5409a9d7

Login Successful!!!
ENTER OK TO CONTINUE->:ok
Select trip
1.One-way        2.Round-Trip
:3
Incorrect Entry
ENTER OK TO CONTINUE->:
```



```
roop@roop-VirtualBox:~/Desktop$ ./client ::1 10000
WELCOME TO THE AIRLINE RESERVATION SYSTEM !!
PLEASE SELECT YOUR PREFERENCE
1)Admin   2)Member   3)Guest:2
Please enter your Customer credentials
Username:Student
Password:
SHA of Network is 53ebc572b4a44802ba114729f07bdaaf5409a9d7

Login Successful!!!
ENTER OK TO CONTINUE->:ok
Select trip
1.One-way        2.Round-Trip
:3
Incorrect Entry
ENTER OK TO CONTINUE->:ok
Select trip
1.One-way        2.Round-Trip
:1
Enter the trip details
Departure_Airport:hfvbdf
Arrival_Airport:jnvvj
Depart_Date:ndnfvj
No flights for the data selected. Please try again.
ENTER OK TO CONTINUE->:
```

9. LEARNINGS

On the technical front, from this project, we have learnt how to create a distributed airline reservation system using sockets in C language. This project provided us with wide vicinity towards the working of a concurrent connection-oriented multiprocessing and multithreading (Hybrid) server. This explained clearly how the server creates a new child process or thread to handle every request that it receives and how the child terminates once the process is complete. Also, we understood the relationship between server and client. This project also, helped us to gain some knowledge about MYSQL. Through this project, we understood and implemented the various concepts such as IPv6, zombies handling etc., learnt in the class practically.

In addition to the technical knowledge we gained, this team project helped us to learn how to work collaboratively in a group to provide better results. This would help us for real time project to be done in our professional life.

10. Resources Used

- BSD Sockets with C programming language.
- MYSQL.
- LINUX environment.

11. Future Enhancements

- User Interface: Provide user interface using HTML tags or C#.
- IPv4-IPv6 Compatibility: The system designed is only IPv6 compatible. Further amelioration includes the IPv4-IPv6 compatibility.
- Semaphore: When multiple admins try to update the data at the same time, there can be discrepancies created. This can be avoided using locking of accept using semaphore.
- Implementation of different API: Winsock.

12. CHALLENGES FACED

One of the major challenges we faced was the **integration of database with the C program**. Database was written in MYSQL but the socket programming was written in C programming language. So, combining MYSQL part with the C programming language quite challenging. Our main database got crashed due to the improper implementation. So, it was difficult at that time because our progress had been hampered and our database had to be reconstructed.

Next challenging part was **passing on the data from the server**, that has been retrieved from the database, **to the client.** As the data sent will be read only after receiving an acknowledgement from the client, there were some problem in the format in which the data was sent. So, figuring out some solution for this problem was time consuming.

At the initial stage of the project, the **development of the requirements** for each component **and implementation of features** were tough and made us to redesign the requirements very often.

13. CONCLUSION

The “Distributed Airline Reservation System" created will thus allow a customer to book the ticket successfully if there is a valid flight flying from that place of departure to place of arrival on intended date of travel. Thus, the project provided us with the better understanding on the BSD socket programming.

14. REFERENCES

Internetworking with TCP/IP Vol. 3, Client-Server programming and applications, Comer and Stevens.

UNIX Network Programming Vol. 1, 3/e: The Sockets Networking API, Stevens, Fenner & Rudoff.

MYSQL

http://www.mysqltutorial.org/basic-mysql-tutorial.aspx

SHA-1

Wikipedia - https://en.wikipedia.org/wiki/SHA-1

15. APPENDIX

PROGRAM CODE:

Server Code:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdarg.h>
#include <netdb.h>
#include <errno.h>
#include <stdio.h>
#include <mysql/mysql.h>
#include <signal.h>
#include <ctype.h>
#include <pthread.h>
#include <stdint.h>
#include <semaphore.h>


/* ----------------------------------Declaration of variables-------------------------------*/
sem_t mutex;
void sig_int(int);
void * doit (void *);
pthread_t tid;
socklen_t clilen;
int *thread_soc;
struct thread thread1;
char prim_key[1024];
int sock,new_sock;
int join,list;
struct sockaddr_in6 server;
struct sockaddr_in6 client;
int length;
char buffer[2096],buffer1[5000],data[1024],column[1024];
char dat[2096];
char userid[2096];
char password[2096];
char ad_in[2096]="1";
char pri_up[2096]="2";
char gu_del[2096]="3";
char serv[2096] = "localhost";
char user[2096] = "root";
char pass[2096] = "Abcd@1234";
char database[2096] = "airlines";
char table[2096]= "air_lines";
int port;
char *sockt= NULL;
char s_no[2096];
char airlines[2096];
char flight_no[2096];
char depart[2096];
char dis[1024];
char arrive[2096];
char depart_date[2096];
char arrive_date[2096];
char depart_time[2096];
char arrive_time[2096];
char seats[2096];
int seats_remain;
char cost[2096];
char discount[2096]="";
char query[2096];
char rate[2096];
MYSQL *establish;
MYSQL_RES *reply;
MYSQL_ROW rows;
int k,k1;

/*------------------------------------Thread structure-------------------------------------------*/
struct thread{

unsigned int sock_new;

}obj;

/*----------------------------------decrypt function declaration-----------------------------*/
void decrypt()
{
printf("password in dedcrypt: %s\n", password);
establish = mysql_init(NULL);
```

```
  if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
    printf("Unable to connect to the database\n");
    exit(1);
  }
sprintf(query,"SELECT * FROM pass WHERE hash='%s'",password);

mysql_query(establish,query);
reply = mysql_store_result(establish);
bzero(buffer,2096);

 while (rows = mysql_fetch_row(reply))
{
sprintf(buffer,"%s",rows[0]);
}
mysql_free_result(reply);
  mysql_close(establish);
}


/*------------------------------Zombie Handler Function--------------------------------------*/
void
sig_child(int signo)
{
pid_t pid;
int stat;
while((pid=waitpid(-1,&stat,WNOHANG))>0)
printf("child %d terminated\n",pid);
return;
}

void
sig_int(int signo)
{
exit(signo);
}
/*----------------------------------Main Function-------------------------------------------*/
int main(int argc, char *argv[])        // The argc and argv are used to get the
command line arguments.
{

if (argc<2)
```

```
{
printf("port or filename not provided\n");
}


/*-----------------------------------Socket creation-----------------------------------------*/

sock=socket(AF_INET6, SOCK_STREAM, 0);                          // creating a
socket with arguments such as domain, connection type (TCP), protocol
if(sock==-1)
{
printf("Socket is not created");                          // Error when socket not created
exit(-1);
}

printf("Socket created\n");

/*--------------------Defining the value for the server structure------------------------*/

bzero((char *)&server,sizeof(server));
server.sin6_flowinfo= 0;
server.sin6_family= AF_INET6;                          // Assigning the Family name
AF_INET
server.sin6_port=htons(atoi(argv[1]));                    // Assigning the port number for
the server
server.sin6_addr=in6addr_any;                          // IP address is not assigned to
any specific socket.
port=server.sin6_port;

/*------------------------------- Binding of socket-------------------------------------*/

length=sizeof(server);
join=bind(sock, (struct sockaddr *)&server, length);// passing arguments such as sd,
server struct & length of struct to the bind function
if (join==-1)
        {
        printf("Bind has not happened");                          // error when bind does
not happend
        exit(-1);
        }
struct dirent *ent;
```

```
/*---------------------------Placing the socket in listening mode ---------------------------/

list=listen(sock,5);                                    // passing arguments sd, and
queue to the listen function
if (list==-1)
{
printf("Listen has not happened");                      // Error when listen function fails
exit(-1);
}

printf("listening..\n");

/*-------------- Accepting a connection and creation of new socket------------------- */

signal(SIGCHLD,sig_child);

signal(SIGINT,sig_int);
for(;;)
{
new_sock=accept(sock,(struct sockaddr *)&client, &length);          // Connection
establishment and creation if new socket.
if (new_sock==-1)
{
//printf("Unable to accept the connection");                 // Error when connection
is not established.
exit(-1);
}

printf("accepted\n");

/*----------------------------------Multi-Processing-------------------------------------*/
int pid;
pid=fork();
if(pid<0)
{
        close(new_sock);
        exit(-1);
}
if(pid>0)
{
        close(new_sock);
}
if(pid==0)
{

if(write(new_sock,"\nWELCOME TO THE AIRLINE RESERVATION SYSTEM
!!\n\nPLEASE SELECT YOUR PREFERENCE\n\n1)Admin   2)Member
3)Guest",sizeof("WELCOME TO THE AIRLINE RESERVATION SYSTEM
!!\n\nPLEASE SELECT YOUR PREFERENCE\n\n1)Admin   2)Member
3)Guest\n\n"))<0)
        {
        printf("Cannot find the client\n");
        }

        if(read(new_sock,buffer,2096)<0)
        {
        printf("Client type cannot be read\n");
        }

}

strcpy(dat,buffer);
if ((strcasecmp(pri_up,dat)==0)||(strcasecmp(gu_del,dat)==0))
{

int pid;
pid=fork();
if(pid<0)
{
        close(new_sock);
        exit(-1);
}
if(pid>0)
{
        close(new_sock);
}
if(pid==0)
{
        close(sock);
/*----------------------------------------Customer Part-------------------------------------*/
int m,m1;
m=strcasecmp(dat,pri_up);

int check2,check3;
if(m==0)
{
do{
```

```
m1=0;

            if(write(new_sock,"\nPlease enter your Customer
credentials\n\nUsername",sizeof("\nPlease enter your Customer
credentials\n\nUsername"))<0)
            {
            printf("Cannot ask for the credential username\n");
            }
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read the username\n");
            }

            strcpy(userid,buffer);
printf("userid: %s\n",userid);
check2=strcasecmp(userid,"Student");
            bzero(buffer,2096);
            if(write(new_sock,"Password",sizeof("Password"))<0)
            {
            printf("Cannot ask for the credential password \n");
            }


            bzero(buffer,2096);
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read the password\n");
            }
bzero(password,2096);
            sscanf(buffer,"%s",password);
printf("paswrd: %s\n",password);

decrypt();                                          //Calling Decrypt Function

            check3=strcmp(buffer,"Network");

            if((check2==0)&&(check3==0))
            break;

            if((check2!=0)||(check3!=0))
        {
            if(write(new_sock,"\nInvalid userid and password combination\n\nENTER
OK TO CONTINUE->",sizeof("\nInvalid userid and password
combination\n\nENTER OK TO CONTINUE->"))<0)
```

```
        {
        printf("Cannot write the invalid combination \n");
        }
        bzero(buffer,2096);
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read the reply\n");
        }
m1 = 1;

        }
}while(m1);

        if((check2==0)&&(check3==0))
        {
        if(write(new_sock,"\nLogin Successful!!!\n\nENTER OK TO CONTINUE-
>",sizeof("\nLogin Successful!!!\n\nENTER OK TO CONTINUE->"))<0)
        {
        printf("Cannot write \n");
        }
        }
        bzero(buffer,2096);
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read the reply\n");
        }


}

int check4,check5;
int x1;
check4=strcasecmp(dat,gu_del);
check5=strcasecmp(dat,pri_up);

int check6;
int check7;

if((check4==0)||(check5==0))      // checking if the client is member or guest so that
trip type can be selected.
        {
do{
x1=0;
```

```
        if(write(new_sock,"\nSelect trip\n\n1.One-way\t2.Round-
Trip\n",sizeof("\nSelect trip\n\n1.One-way\t2.Round-Trip\n"))<0)
                {
                printf("Cannot select the trip \n");
                }

        if(read(new_sock,buffer1,2096)<0)
                {
                printf("Cannot read the trip\n");
                }
check6=strcasecmp(buffer1,ad_in);
check7=strcasecmp(buffer1,pri_up);

if(check6!=0 && check7!=0)
{
if(write(new_sock,"\nIncorrect Entry\n\nENTER OK TO CONTINUE-
>",sizeof("Incorrect Entry\n\nENTER OK TO CONTINUE->"))<0)
        {
        printf("Cannot write\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read\n");
        }
x1=1;
}

}while(x1);
}

int z1;
int a1=1,b1=1;
check6=strcasecmp(buffer1,ad_in);
check7=strcasecmp(buffer1,pri_up);

do{
z1=0;
if((check6==0)||(check7==0))

{


bzero(buffer,2096);
```

```
		if(write(new_sock,"\nEnter the trip
	details\n\nDeparture_Airport",sizeof("\nEnter the One-way
	details\n\nDeparture_Airport"))<0)
		{
		printf("Cannot write \n");
		}
		bzero(buffer,2096);
		if(read(new_sock,depart,2096)<0)
		{
		printf("Cannot read \n");
		}
		bzero(buffer,2096);
		if(write(new_sock,"\nArrival_Airport",sizeof("\nArrival_Airport"))<0)
		{
		printf("Cannot write Arrival_Airport \n");
		}
		bzero(buffer,2096);
		if(read(new_sock,arrive,2096)<0)
		{
		printf("Cannot read Arrival_Airport\n");
		}
		bzero(buffer,2096);
		if(write(new_sock,"\nDepart_Date",sizeof("\nDepart_Date"))<0)
		{
		printf("Cannot write Depart_Date \n");
		}
		bzero(buffer,2096);
		if(read(new_sock,depart_date,2096)<0)
		{
		printf("Cannot read Depart_Date\n");
		}

	}

	if (check7==0)

	{

		bzero(buffer,2096);
		if(write(new_sock,"\nReturn_Date",sizeof("\nReturn_Date"))<0)
		{
		printf("Cannot write Return_Date \n");
		}
		bzero(buffer,2096);
```

```
            if(read(new_sock,arrive_date,2096)<0)
            {
            printf("Cannot read Return_Date\n");
            }
}
/*-----------------------------sql query------------------------------------------------------*/
if((check6==0)||(check7==0))
{
//database="airlines";

establish = mysql_init(NULL);

  /* Connect to database */

  if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
    printf("Unable to connect to the database\n");
    exit(1);
}

  /*---------------------------------- send SQL query --------------------------------------*/

if (check6==0)
{

sprintf(query,"SELECT * FROM air_lines WHERE Depart='%s' and Arrive='%s' and
Depart_Date='%s'",depart,arrive,depart_date);
}

else if (check7==0)

{

sprintf(query,"SELECT * FROM air_lines WHERE Depart='%s' and Arrive='%s' and
Depart_Date='%s' and Arrive_Date='%s' ",depart,arrive,depart_date,arrive_date);
}



 mysql_query(establish, query);


  reply = mysql_store_result(establish);
  /*----------------------------output table name------------------------------------------ */

  while (rows = mysql_fetch_row(reply))

{
        bzero(buffer,2096);
if(check5==0)
{
a1=0;

        sprintf(buffer,"\nMatching Airline Details From the
DB:\n\nAirlines:%s\nFlight_No:%s\nDeparture_Airport:%s\nArrival_Airport:%s\nDe
parture_Time:%s\nArrival_time:%s\nDeparture_Date:%s\nReturn_Date:%s\nSeats_R
emaining:%s\nCost($):%s\nDiscount($):%s\n\nEnter OK to
proceed",rows[1],rows[2],rows[3],rows[4],rows[5],rows[6],rows[7],rows[8],rows[9],r
ows[10],rows[11]);
}
if(check4==0)
{
b1=0;

        sprintf(buffer,"\nMatching Airline Details From the
DB:\n\nAirlines:%s\nFlight_No:%s\nDeparture_Airport:%s\nArrival_Airport:%s\nDe
parture_Time:%s\nArrival_time:%s\nDeparture_Date:%s\nReturn_Date:%s\nSeats_R
emaining:%s\nCost (in dollars):%s\n\nEnter OK to
proceed",rows[1],rows[2],rows[3],rows[4],rows[5],rows[6],rows[7],rows[8],rows[9],r
ows[10]);
}
strcpy(rate, rows[10]);
strcpy(prim_key,rows[2]);
strcpy(dis, rows[11]);
seats_remain=atoi(rows[9]);
        if(write(new_sock,buffer,sizeof(buffer))<0)
        {
        printf("Error sending the database \n");
        }
   bzero(buffer,2096);
        if(read(new_sock,buffer,sizeof(buffer))<0)
        {
        printf("Error reading the database \n");
        }
}

  /* close connection */
  mysql_free_result(reply);
  mysql_close(establish);

}

if(a1!=0)
{
if(b1!=0)
{

if(write(new_sock,"\nNo flights for the data selected. Please try again.\n\nENTER
OK TO CONTINUE->",sizeof("No flights for the data selected. Please try
again.\n\nENTER OK TO CONTINUE->"))<0)
        {
        printf("Cannot write\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read\n");
        }
z1=1;
}
}
}while(z1);

/*-----------------------------------Booking of ticket--------------------------------------*/
int g, g1;
char n_name[2096], n_age[2096], n_seats[2096], n_email[2096];
if((check6==0)||(check7==0))

{
        bzero(buffer,2096);
        if(write(new_sock,"\nEnter the following details\n\nName",sizeof("\nEnter the
following details\n\nName"))<0)
        {
        printf("Cannot enter the name \n");
        }
        bzero(buffer,2096);
        if(read(new_sock,n_name,2096)<0)
        {
        printf("Cannot read name\n");
        }
        printf("name: %s\n",n_name);
        bzero(buffer,2096);
        if(write(new_sock,"\nAge",sizeof("\nAge"))<0)
        {
        printf("Cannot write Age \n");
        }
        bzero(buffer,2096);
        if(read(new_sock,n_age,2096)<0)
        {
        printf("Cannot read Age\n");
        }
printf("age: %s\n",n_age);
int age;
age=atoi(n_age);
if(isdigit(age)!=0)
{
        bzero(buffer,2096);
        if(write(new_sock,"\nType age in terms of number\n",sizeof("\nType age in
terms of number\n"))<0)
        {
        printf("Cannot write Age \n");
        }
exit(-1);
}

do
{
g1=0;
        bzero(buffer,2096);
        if(write(new_sock,"\nEnter the Flight Number ",sizeof("Enter the Flight
Number "))<0)
        {
        printf("Cannot write Flight Number\n");
        }
        bzero(flight_no,2096);
        if(read(new_sock,flight_no,2096)<0)
        {
        printf("Cannot read Flight Number\n");
        }
printf("flight: %s\n",flight_no);
```

```
if(strcmp(flight_no,prim_key)!=0)
{
if(write(new_sock,"\nIncorrect Flight number\n\nENTER OK TO CONTINUE-
>",sizeof("Incorrect Flight number\n\nENTER OK TO CONTINUE->"))<0)
        {
        printf("Cannot write\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read\n");
        }
g1 = 1;
}
} while(g1);

        bzero(buffer,2096);
do
{
g1=0;
        if(write(new_sock,"\nNumber of seats required ",sizeof("\nNumber of seats
required "))<0)
        {
        printf("Cannot write No. of seats required\n");
        }

        if(read(new_sock,n_seats,2096)<0)
        {
        printf("Cannot read No. of seats required\n");
        }
printf("seats: %s\n",n_seats);
        if((atoi(n_seats))>seats_remain)
        {
        if(write(new_sock,"\nNumber of seats entered is greater than the seats
available\n\nENTER OK TO CONTINUE->",sizeof("\nNo. of seats entered is greater
than the seats available\n\nENTER OK TO CONTINUE->"))<0)
        {
        printf("Cannot write No. of seats required\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read\n");
        }
g1=1;
        }
```

```
}while(g1);

        bzero(buffer,2096);
        if(write(new_sock,"\nEnter the e-mail ID ",sizeof("\nEnter the e-mail ID
"))<0)
        {
        printf("Cannot write Phone Number\n");
        }
        bzero(buffer,2096);
        if(read(new_sock,n_email,2096)<0)
        {
        printf("Cannot read Phone Number\n");
        }

printf("email: %s\n",n_email);

bzero(buffer,2096);
int total;
int r;
printf("ok here\n");
        establish = mysql_init(NULL);
  if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)
{
    printf("Unable to connect to the database");
    exit(1);
}

bzero(query,1024);
sprintf(query,"SELECT Discount FROM air_lines WHERE
Flight_No='%s'",flight_no);
mysql_query(establish, query);
reply = mysql_store_result(establish);
rows = mysql_fetch_row(reply);

printf("rate: %s\n",rate);

if(check5==0)
{
strcpy(discount,dis);
}
if(check4==0)
{
bzero(discount,2096);
}
```

```
r = atoi(rate)-atoi(discount);

printf("r: %d\n",r);

total=atoi(n_seats)*r;
seats_remain = seats_remain-atoi(n_seats);
sprintf(seats, "%d", seats_remain);
sprintf(buffer, "\nAmount to be paid : %d\n\n Confirm booking(y/n) ",total);
        if(write(new_sock,buffer,sizeof(buffer))<0)
        {
        printf("Cannot write Booking successful\n");
        }
        bzero(buffer,2096);
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read the reply\n");
        }
        if(strcmp(buffer,"y")==0)
        {
        bzero(buffer,2096);
establish = mysql_init(NULL);

  if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
    printf("Unable to connect to the database\n");
    exit(1);
}

sprintf(query,"Insert into people values('%s','%s','%s','%s','%d','%s')",n_name, n_age,
n_seats, flight_no, total, n_email);

        mysql_query(establish,query);

sprintf(query,"update air_lines set Seats='%s' where Flight_No='%s';",seats,
flight_no);
mysql_query(establish,query);

mysql_free_result(reply);
  mysql_close(establish);
        if(write(new_sock,"\nTicket booked successfully!! Enjoy your Trip
!!!\n",sizeof("Ticket booked successfully!! Enjoy your Trip !!!\n"))<0)
        {
        printf("Cannot write Booking successful\n");
		}
		bzero(buffer,2096);
		if(read(new_sock,buffer,2096)<0)
		{
		printf("Cannot read the reply\n");
		}
		}
		else
		{
		bzero(buffer,2096);
		if(write(new_sock,"\nPlease Try again later\n\nENTER OK TO CONTINUE-
> \n",sizeof("\nPlease Try again later\n\nENTER OK TO CONTINUE-> \n"))<0)
		{
		printf("Cannot write\n");
		}
		bzero(buffer,2096);
		if(read(new_sock,buffer,2096)<0)
		{
		printf("Cannot read the reply\n");
		}
		}
if (strcasecmp(buffer,"ok")==0)

exit(1);

}
}

close(new_sock);
}


if (strcasecmp(dat,ad_in)==0)
{
obj.sock_new = new_sock;
uintptr_t *sock1 = malloc(sizeof(uintptr_t));
*sock1 = new_sock;

//-------------------------------------Mutex FUnction----------------------------------
sem_init(&mutex,0,1);

/*-------------------------------------Multi-Threading----------------------------------*/

pthread_create(&tid, NULL, doit,sock1);
```

```
printf("Calling doit function\n");


}
}
printf("No file entered");
return 0;

}
/*--------------------------------Doit Function--------------------------------------------- */
void* doit (void *sock2)
{


int new_sock1 = *((int *) sock2);
struct thread thread1;
char prim_key[1024];
int join,list;
struct sockaddr_in6 server;
struct sockaddr_in6 client;
int length;
char buffer[2096],buffer1[5000],data[1024],column[1024];
char dat[2096];
char userid[2096];
char password[2096];
char ad_in[2096]="1";
char pri_up[2096]="2";
char gu_del[2096]="3";
char serv[2096] = "localhost";
char user[2096] = "root";
char pass[2096] = "Abcd@1234";
char database[2096] = "airlines";
char table[2096]= "air_lines";
int port;
char *sockt= NULL;
char s_no[2096];
char airlines[2096];
char flight_no[2096];
char depart[2096];
char arrive[2096];
char depart_date[2096];
char arrive_date[2096];
char depart_time[2096];
char arrive_time[2096];
char seats[2096];
int seats_remain;
char cost[2096];
char discount[2096]="";
char query[2096];
char rate[2096];
MYSQL *establish;
MYSQL_RES *reply;
MYSQL_ROW rows;

int k,k1;
/*-------------------------decrypt-------------------------*/

void decrypt()
{

establish = mysql_init(NULL);

  if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
    printf("Unable to connect to the database\n");
    exit(1);
  }
sprintf(query,"SELECT * FROM pass WHERE hash='%s'",password);

mysql_query(establish,query);
reply = mysql_store_result(establish);
bzero(buffer,2096);

 while (rows = mysql_fetch_row(reply))
{
sprintf(buffer,"%s",rows[0]);
}
mysql_free_result(reply);
  mysql_close(establish);
}



/*--------------------------------show columns---------------------------------*/
void show_columns()
{
char a[1024];
```

```
     if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
      printf("Unable to connect to the database\n");
      exit(1);
   }

sprintf(query,"show columns from %s;",table);
printf("query= %s\n",query);
mysql_query(establish,query);

reply = mysql_store_result(establish);

bzero(buffer,2096);
bzero(a,1024);

   /*-------------------------------output table name ---------------------------------------*/

   for(int i=0;rows = mysql_fetch_row(reply); i++)
{

sprintf(a,"\n %s ",rows[0]);
strcat(a,"");
strcat(buffer,a);

}

return;
}

/*----------------------------show tables--------------------------------*/
void show_tables()
{
char a[1024];

     if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
      printf("Unable to connect to the database\n");
      exit(1);
   }
```

```
sprintf(query,"show tables;");
mysql_query(establish,query);


reply = mysql_store_result(establish);

bzero(buffer,2096);

  /*---------------------------- output table name----------------------------------------- */

  for(int i=0;rows = mysql_fetch_row(reply); i++)
{

sprintf(a,"\n %s ",rows[0]);
strcat(a,"");
strcat(buffer,a);

}

return;
}

/*-----------------------update table----------------------------*/


int check,check1;
do{
k1=0;



        if(write(new_sock,"\nPlease enter your Admin
credentials\n\nUserid",sizeof("\nPlease enter your Admin
credentials\nUserid\n\n"))<0)
        {
        printf("Cannot ask for the credential userid\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read the username\n");
        }

        strcpy(userid,buffer);
check=strcmp(userid,"123456");
printf("userid: %s\n",userid);
```

```
            bzero(buffer,2096);
bzero(password,2096);
            if(write(new_sock,"Password",sizeof("Password"))<0)
            {
            printf("Cannot ask for the credential password \n");
            }



            bzero(buffer,2096);
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read the password\n");
            }

sscanf(buffer,"%s",password);

printf("paswrd: %s\n",password);
decrypt();
            check1=strcmp(buffer,"Cmpepro");

            if((check==0)&&(check1==0))
            break;

            if((check!=0)||(check1!=0))
        {
            if(write(new_sock,"Invalid userid and password combination\n\nENTER OK
TO CONTINUE->",sizeof("Invalid userid and password combination\n\nENTER OK
TO CONTINUE->"))<0)
            {
            printf("Cannot write the invalid combination \n");
            }
            bzero(buffer,2096);
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read the reply\n");
            }
k1 = 1;

            }
}while(k1);

            if((check==0)&&(check1==0))
            {
```

```
            if(write(new_sock,"Login Successful!!!\n\nENTER OK TO CONTINUE-
>",sizeof("Login Successful!!!\n\nENTER OK TO CONTINUE->"))<0)
            {
            printf("Cannot write \n");
            }
            }
            bzero(buffer,2096);
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read\n");
            }

int p,p1;
sem_wait(&mutex);
            if(write(new_sock,"\nEnter 1 to SHOW DATABASES  And Select the
database required\n\n ",sizeof("\nEnter 1 to SHOW DATABASES And Select the
database required\n\n "))<0)
            {
            printf("Cannot ask for the task.\n");
            }
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read the task.\n");
            }

/*-------------------------------------Show Databases------------------------------*/

establish = mysql_init(NULL);

  /*--------------------------------- Connect to database-------------------------- */


  if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
    printf("Unable to connect to the database\n");
    exit(1);
  }
sprintf(query,"show databases;");
mysql_query(establish,query);
 reply = mysql_store_result(establish);
bzero(buffer,2096);

  /* output table name */
 char a[1024];
   for(int i=0;rows = mysql_fetch_row(reply); i++)
{
strcpy(a,rows[0]);
strcat(a,"\n");
strcat(buffer,a);
}

            if(write(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error sending the database \n");
            }
    bzero(buffer,2096);
            if(read(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error reading the database \n");
            }
printf("buffer database: %s\n",buffer);
strcpy(database,buffer);
/*---------------------------------------------Show tables---------------------------------------------*/

            if(write(new_sock,"\nEnter 1 to SHOW TABLES  And Select the table
required\n\n ",sizeof("\nEnter 1 to SHOW TABLES  And Select the table required\n\n
"))<0)
            {
            printf("Cannot ask for the task.\n");
            }
            if(read(new_sock,buffer,2096)<0)
            {
            printf("Cannot read the task.\n");
            }

show_tables();


            if(write(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error sending the database \n");
            }
    bzero(buffer,2096);
            if(read(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error reading the database \n");
            }
```

```
printf("buffer: %s\n",buffer);
sscanf(buffer,"%s",table);

bzero(buffer,2096);
        if(write(new_sock,"\nEnter 1 to SHOW COLUMNS and Press OK to
proceed\n\n ",sizeof("\nEnter 1 to SHOW COLUMNS and Press OK to proceed\n\n
"))<0)
        {
        printf("Cannot ask for the task.\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read the task.\n");
        }

show_columns();

        if(write(new_sock,buffer,sizeof(buffer))<0)
        {
        printf("Error sending the database \n");
        }
   bzero(buffer,2096);
        if(read(new_sock,buffer,sizeof(buffer))<0)
        {
        printf("Error reading the database \n");
        }

int g;
/*-------------------------------------Tasks--------------------------------------------------*/
do
{
g=0;
 if(write(new_sock,"\nPlease enter the task to be performed.\n\n1.Insert
Row\t2.Update Row\t3.Delete Row\t4.View Row data",sizeof("\nPlease enter the task
to be performed.\n\n1.Insert Row\t2.Update Row\t3.Delete Row\t4.View Row
data"))<0)
        {
        printf("Cannot ask for the task.\n");
        }
        if(read(new_sock,buffer,2096)<0)
        {
        printf("Cannot read the task.\n");
        }
```

```
  /* send SQL query */
int n1;
n1=strcasecmp(buffer,ad_in);

if(n1==0)
{

        if(write(new_sock,"\nEnter the following data to insert a new
row\n\nRow_no",sizeof("\nEnter the following data to insert a new
row\n\nRow_No"))<0)
        {
        printf("Cannot send\n");
        }
        bzero(buffer,2096);
        if(read(new_sock,s_no,2096)<0)
        {
        printf("Cannot read the reply\n");
        }

        if(write(new_sock,"Airlines",sizeof("Airlines"))<0)
        {
        printf("Cannot send airlines");
        }
        fflush(stdout);
        bzero(buffer,2096);
        if(read(new_sock,airlines,2096)<0)
        {
        printf("Cannot read the reply\n");
        }

        if(write(new_sock,"Flight number",sizeof("Flight number"))<0)
        {
        printf("Cannot send flight number");
        }
        fflush(stdout);
        bzero(buffer,2096);
        if(read(new_sock,flight_no,2096)<0)
        {
        printf("Cannot read the reply\n");
        }

        if(write(new_sock,"Depart",sizeof("Depart"))<0)
        {
        printf("Cannot send Depart");
```

```
}
fflush(stdout);
bzero(buffer,2096);
if(read(new_sock,depart,2096)<0)
{
printf("Cannot read the reply\n");
}

if(write(new_sock,"Arrive",sizeof("Arrive"))<0)
{
printf("Cannot send Arrive");
}
fflush(stdout);
bzero(buffer,2096);
if(read(new_sock,arrive,2096)<0)
{
printf("Cannot read the reply\n");
}

if(write(new_sock,"Depart Date",sizeof("Depart Date"))<0)
{
printf("Cannot send Depart Date");
}
fflush(stdout);
bzero(buffer,2096);
if(read(new_sock,depart_date,2096)<0)
{
printf("Cannot read the reply\n");
}

if(write(new_sock,"Arrival Date",sizeof("Arrival Date"))<0)
{
printf("Cannot send Arrival Date");
}
fflush(stdout);
bzero(buffer,2096);
if(read(new_sock,arrive_date,2096)<0)
{
printf("Cannot read the reply\n");
}

if(write(new_sock,"Depart Time",sizeof("Depart Time"))<0)
{
printf("Cannot send Depart Time");
```

```
}
fflush(stdin);
bzero(buffer,2096);
if(read(new_sock,depart_time,2096)<0)
{
printf("Cannot read the reply\n");
}

if(write(new_sock,"Arrival Time",sizeof("Arrival Time"))<0)
{
printf("Cannot send Arrival Time");
}
fflush(stdin);
bzero(buffer,2096);
if(read(new_sock,arrive_time,2096)<0)
{
printf("Cannot read the reply\n");
}

if(write(new_sock,"Seats",sizeof("Seats"))<0)
{
printf("Cannot send Seats");
}
fflush(stdin);
bzero(buffer,2096);
if(read(new_sock,seats,2096)<0)
{
printf("Cannot read the reply\n");
}
if(write(new_sock,"Cost per ticket",sizeof("Cost per ticket"))<0)
{
printf("Cannot send Cost");
}
fflush(stdin);
bzero(buffer,2096);
if(read(new_sock,cost,2096)<0)
{
printf("Cannot read the reply\n");
}
if(write(new_sock,"Discount",sizeof("Discount"))<0)
{
printf("Cannot send Discount Rate");
}
fflush(stdin);
```

```
        bzero(buffer,2096);
        if(read(new_sock,discount,2096)<0)
        {
        printf("Cannot read the reply\n");
        }

sscanf(buffer, "%s %s %s %s %s %s %s %s %s %s %s
%s",s_no,airlines,flight_no,depart,arrive,depart_date,arrive_date,depart_time,arrive_ti
me,seats,cost,discount);
printf("%s\n",depart);
printf("%s\n",arrive);

printf("%s\n",depart_time);
printf("%s\n",depart_date);

printf("%s\n",seats);
printf("%s\n",cost);

sem_wait(&mutex);
        sprintf(query,"Insert into air_lines
values('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')",s_no,airlines,flight_n
o,depart,arrive,depart_time,arrive_time,depart_date,arrive_date,seats,cost,discount);

        if(mysql_query(establish,query)==0)
        {
        bzero(buffer,2096);
        if(write(new_sock,"\nInserted New Row\n\nPress OK to
continue",sizeof("\nInserted New Row\n\nPress OK to continue"))<0)
        {
        printf("Error deleting the data in the database \n");
        }
        }
        else
        {
        bzero(buffer,2096);
        if(write(new_sock,"\nDatabase is not inserted with new values\n\nPress OK to
continue",sizeof("\nDatabase is not inserted with new values\n\nPress OK to
continue"))<0)
        {
        printf("Error inserting the data in the database \n");
        }
        }

        if(read(new_sock,buffer,sizeof(buffer))<0)
        {
        printf("Error reading the database \n");
        }
sem_post(&mutex);
}
int n2;
n2=strcasecmp(buffer,pri_up);
if(n2==0)
{
sem_wait(&mutex);
        if(write(new_sock,"\nEnter the following details to update the row
data\n\nFlight_No",sizeof("\nEnter the following details to update the row
data\n\nFlight_No"))<0)
        {
        printf("Cannot ask for the task.\n");
        }
fflush(stdout);
        bzero(flight_no,2096);
        if(read(new_sock,flight_no,2096)<0)
        {
        printf("Cannot read the task.\n");
        }
printf("flight: %s\n", flight_no);
        if(write(new_sock,"\nColumn Name",sizeof("\nColumn Name"))<0)
        {
        printf("Cannot ask for the task.\n");
        }
fflush(stdout);
        bzero(column,2096);
        if(read(new_sock,column,2096)<0)
        {
        printf("Cannot read the task.\n");
        }
printf("col: %s\n", column);
        if(write(new_sock,"\nEnter the data to be updated",sizeof("\nEnter the data to
be updated"))<0)
        {
        printf("Cannot ask for the task.\n");
        }
fflush(stdout);
        bzero(data,2096);
        if(read(new_sock,data,2096)<0)
        {
        printf("Cannot read the task.\n");
```

```
            }


    if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

{
     printf("Unable to connect to the database\n");
     exit(1);
   }

sprintf(query,"update %s set seats='%s' where Flight_No='%s';",table,data,flight_no);
mysql_query(establish,query);
sem_post(&mutex);
}

int n3;
n3=strcasecmp(buffer,gu_del);
if(n3==0)
{
sem_wait(&mutex);

if(write(new_sock,"\n\nEnter the following detail to perform DELETE\n\nFlight
Number",sizeof("\nEnter the following detail to perform DELETE\n\nFlight
Number"))<0)
            {
            printf("Cannot enter the update field\n  \n");
            }

            bzero(buffer,2096);
            if(read(new_sock,flight_no,2096)<0)
            {
            printf("Cannot read the reply\n");
            }

            sprintf(query,"Delete from air_lines where Flight_No='%s'",flight_no);
            if(mysql_query(establish,query)==0)
            {
            bzero(buffer,2096);
            if(write(new_sock,"\nData is deleted from the database",sizeof("\nData is
deleted from the database"))<0)
            {
            printf("Error deleting the data in the database \n");
            }
            }
            else
            {
            bzero(buffer,2096);
            if(write(new_sock,"\nData is not deleted from the database",sizeof("\nData is
not deleted from the database"))<0)
            {
            printf("Error updating the database \n");
            }
            }

            if(read(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error reading the database \n");
            }
sem_post(&mutex);
}

int n4;
n4=strcasecmp(buffer,"4");
if(n4==0)
{
sem_wait(&mutex);

bzero(buffer,2096);
if(write(new_sock,"\n\nEnter the following detail to View the corresponding Row
data\n\nFlight Number",sizeof("\nEnter the following detail to View the
corresponding Row data\n\nFlight Number"))<0)
            {
            printf("Cannot enter the update field\n  \n");
            }

            bzero(buffer,2096);
            if(read(new_sock,flight_no,2096)<0)
            {
            printf("Cannot read the reply\n");
            }


     if((mysql_real_connect(establish, serv, user, pass, database, port, sockt, 0))<0)

            {
      printf("Unable to connect to the database\n");
      exit(1);
             }
            sprintf(query,"SELECT * FROM %s WHERE 
flight_no='%s'",table,flight_no);
             mysql_query(establish, query);

              reply = mysql_store_result(establish);

   /* ------------------------------output table name-----------------------------------------*/

   while (rows = mysql_fetch_row(reply))
{

            sprintf(buffer,"\nMatching Airline Details From the 
DB:\n\nRow_No:%s\nAirlines:%s\nFlight_No:%s\nDeparture_Airport:%s\nArrival_
Airport:%s\nDeparture_Time:%s\nArrival_time:%s\nDeparture_Date:%s\nReturn_Da
te:%s\nSeats_Remaining:%s\nCost($):%s\nDiscount($):%s\n\nEnter OK to 
proceed",rows[0],rows[1],rows[2],rows[3],rows[4],rows[5],rows[6],rows[7],rows[8],r
ows[9],rows[10],rows[11]);

}

if(write(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error sending the database \n");
            }
    bzero(buffer,2096);
            if(read(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error reading the database \n");
            }

}



            bzero(buffer,2096);
            if(write(new_sock,"\nTask Completed.\n\nDo you want to perform any other 
task?\n1.Yes\n2.No",sizeof("\nTask Completed.\n\nDo you want to perform any other 
task?\n1.Yes\n2.No"))<0)
            {
            printf("Error in writing \n");
            }


            if(read(new_sock,buffer,sizeof(buffer))<0)
            {
            printf("Error in reading \n");
            }

            if(strcmp(buffer,ad_in)==0)
            {
            g=1;
            }
}while(g);
sem_post(&mutex);

 /*---------------------------- close connection --------------------------------------------*/
   mysql_free_result(reply);
   mysql_close(establish);

close((int) sock2);

}
```

Client Code

```
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdarg.h>
#include <netdb.h>
#include <errno.h>
#include <stdio.h>
#include <math.h>
#include <mysql/mysql.h>
#include <openssl/sha.h>
#include <gcrypt.h>
#define LINE_BUFSIZE 2096
#define SHA_DIGEST_LENGTH 20

int data_read,data_write;
int main(int argc, char *argv[])
{

/*-----------------declare variables-----------------------*/

int sockfd, portnum;
int i;
char buffer[2096];
char command[1024],command2[1024],dummy[255];
char password[1024];
unsigned char temp[SHA_DIGEST_LENGTH];
   char buf[SHA_DIGEST_LENGTH*2];
char line[LINE_BUFSIZE];
   FILE *pipe;
FILE *iterative;
FILE *iterativec;
struct sockaddr_in6 serv_addr;
struct hostent *server;
if (argc<3)
        {
        printf("port or hostname not provided\n");
        }
```

```
/*-------------------create socket----------------------------*/

sockfd = socket(AF_INET6, SOCK_STREAM, 0);
if (sockfd<0)
        {
        printf("cant open a socket.\n");
        exit(1);
        }

/*-----------set the address in the sockaddr_in--------------*/

bzero((char*)&serv_addr, sizeof(serv_addr));
portnum = atoi(argv[2]);
server = gethostbyname2 (argv[1],AF_INET6);
if (portnum==0)
        {
        printf("no such host.\n");
        exit(1);
        }

serv_addr.sin6_flowinfo= 0;
serv_addr.sin6_family= AF_INET6;

bcopy((char*)server->h_addr, (char*)&serv_addr.sin6_addr.s6_addr, server-
>h_length);
serv_addr.sin6_port = htons(portnum);

/*-------------------connect-------------------------------*/

int len= sizeof(serv_addr);
if(connect(sockfd,(struct sockaddr *) &serv_addr,len)<0)
        {
        printf("can't connect to socket.\n");
        exit(1);
        }

/*------------------Read and Write-----------------------*/

for(;;)
{
while(1)
{

        bzero(buffer, 2096);
```

```
            if(read(sockfd,buffer,2096)<0)          // Receive the data that is sent by the
server.
            {
            printf("Read unsuccessfull\n");          // Error message when the read cannot
be done.
            }
printf("%s:",buffer);

if((strcmp(buffer,"Password"))==0)
{

 goto LOOP;
}

            bzero(buffer, 2096);                     // Clearing the buffer

            scanf("%s",&buffer);                     // Getting user input

                        if((strcmp(buffer,"exit"))==0)
{
 exit(1);
}
            if(send(sockfd, buffer, sizeof(buffer), 0)<0)              // Send back the reply to
the server.
            {
                        printf("Send unsuccessfull\n");                // Error message when
send cannot be done.
            }


}

LOOP:
    pipe = popen("./my_script", "r");       // Bash Script to mask the password
    if (pipe == NULL)
            {
       exit(1);

    }

    while (fgets(line, LINE_BUFSIZE, pipe) != NULL) {
        printf("%s",line);
          }
iterative = fopen("data.txt","r+");
```

```
fscanf(iterative,"%s",password);


   memset(buf, 0x0, SHA_DIGEST_LENGTH*2);
   memset(temp, 0x0, SHA_DIGEST_LENGTH);

   SHA1(password, strlen(password), temp);    //Function to hash the password

   for (i=0; i < SHA_DIGEST_LENGTH; i++) {
      sprintf((char*)&(buf[i*2]), "%02x", temp[i]);
   }
 iterativec=fopen("data.txt","w+");
fprintf(iterativec,"%s",buf);
   printf("\n\nSHA of %s is %s\n", password, buf);
bzero(buffer,2096);

/*----------------------------- Sending data back to client---------------------------------- */
strcpy(buffer,buf);
if(send(sockfd, buffer, sizeof(buffer), 0)<0)
          {
                    printf("Send unsuccessfull\n");
          }
      pclose(pipe);
printf("\n");

}                                             // while end
return 0;
}
```