Fundamentals of Databases Lab - CL2011

Design Project



**AIRLINE MANAGEMENT SYSTEM**

Submitted by:

|  |  |
| --- | --- |
| Hassaan Mumtaz | 18L1292 |
| Abdur Raheem | 18L1269 |

Submitted to:

Sir Waheed Faraz

06/03/2022

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Lahore

## Introduction

1. **Problem Analysis**

## Design Requirements

1. **Feasibility Analysis**

## Possible Solutions

1. **Preliminary Design**

## Design Description

## Software Simulation

## Experimental Results

1. **Performance Analysis**
2. **Future Scope**

## Social and Cultural Implications

1. **Conclusion References**

Airline Management System is a software based service that provides users with the ability to interact with a system that keeps a record of the Flights, Passengers, Pilots, Service Staff, Flight Controllers, available Airports, available Airplanes, and even the list of Owners.

The overall project is an implementation of a real-time Airline Management System that allows end users to book flights to and from a specific destination. The front-end UI provides end-users with the ability to choose either a One-Way or a Round trip from a specific destination our system provides. Furthermore, we provide the ability for different employees based in an airport with the ability to connect and integrate on premises management to our database system. This allows airport employees with the ability to schedule flights, create workflows, schedule maintenance and even create timetables for on-plane staff. The necessity of this entire project arises from the ability to integrate vital transportation facilities with an excessive ease of access.

With the advent of the 21st century and interaction between countries and tourism exponentially increasing interaction between countries as a result flights to and from countries are exponentially increasing hence providing efficient data integrated solutions with a one-stop shop capable of not only managing Flights but also encapsulating the key interactions inside an Airport can facilitate not only end-users but also Airport staff.

The current solutions airports provide are primarily based on two-completely different system i.e. one to manage passengers and another to manage airport staff. This results in redundancy in data that our solution tends to avoid, by creating a networked model capable of storing normalized data. For example, there can be redundancy when both “Sam” is categorized as a Passenger and a Pilot even though he was a Passenger for the Flight from London to Lahore on 21/04/22, and a Pilot for 32 other flights. This results in a mixture of data that can be easily solved by normalizing our table in a single Point of Interest (POI).

The main objective of our system also tends to provide the ability to manage in house workers by allotting Pilots to fly certain Flights ,Flight Controllers to work at a certain Airport, and Service Workers to maintain certain Airplanes. It also allows us the capability to add new persons to the database, that can be further categorized in Airport Workers or Passengers. Besides this, like generic Flight Management System our System also holds the capability to book Airplanes from one destination to another.

The website will provide interface to the user through which they can get the end result. The system will also allow the users make their own accounts so that they can register themselves and also can save either operate the system as an Airport worker or look into available flights as a Passenger.

The design requirements of the given problem statement were as follows:

* To design an Airport Management System with the basic functionalities:
  1. Insert new People into the system
     + Insert new Pilots
     + Insert new Service Workers
     + Insert new Passengers
     + Insert new Flight Controllers
  2. Show new Flights
     + Show Passengers on board certain flights
     + Show Pilots on board certain flights
  3. Allocate Personnel to Airplanes for Maintenance
     + Add Service Workers to work on a Plane
  4. Add new Airplanes to the System
  5. Manage Flight schedules with regards to people.
  6. Manage Flights with respect to Airplanes
* The database designed should be actively linked to the webpage.
* Certain people should have access to change data in the system.

In terms of Time Management we allocated certain scheduled tasks to certain people. That is in the initial stages of Database Design most of the work was done by Hassaan but in other cases when we moved onto implementation of our services on the web most of these tasks were done by Abdur Raheem.

|  |  |
| --- | --- |
| **Task** | **Task Division** |
| Researching various frame-works for both front-end and back-end implementation | Hassaan Mumtaz 45%  Abdur Raheem 65% |
| Database Design | Hassaan Mumtaz 100% |
| Implementation of back-end server on SQL Server | Hassaan Mumtaz 100% |
| Implementation of front-end on React.js | Abdur Raheem 100% |
| Implementation of back-end server on Node.js | Hassaan Mumtaz 30%  Abdur Raheem 70% |
| Documentation | Hassaan Mumtaz 70%  Abdur Raheem 30% |

[There were many possible stacks that both our front and back-end services could be implemented on:

For example for back-end technologies:

1. **PHP:**

PHP is a popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic, PHP powers everything from the database to the most popular websites in the world.

1. **Django:**

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Django takes care of much of the hassle of web development, so developers can focus on writing appa without needing to reinvent the wheel.

1. **ASP.net:**

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

With regards to front-end technologies:

**HTML/CSS:**

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, CSS the (visual and aural) layout, for a variety of devices. Along with graphics and scripting, HTML and CSS are the basis of building Web pages and Web Applications.

However, the framework we used was :

For back-end implementation we used **Node.js** which is an open-source server that is primarily used for event-driven server, due to its single-threaded nature. It's used for traditional web sites and back-end API services, but was designed with real-time, push-based architectures in mind. This allows us to configure more data by using it’s “real-time” pushing in nature. For our front-end design we used **React.js** which is a free and open-source front-end JavaScript library for building user interfaces based on UI components. React can be used as a base in the development of single-page, mobile, or server-rendered applications.

For integrating our back-end server to our database we used **Express.js** which allowed seamless integration to facilitate using all the tabular structure in **SQL Server** with our front-end.

To create a necessary design for our database first we had to create a list of essential necessities using the highlighted design requirements:

There were a number of assumptions we had to take into account before implementing our EER Diagram:

1. Only a Corporation can be an Owner.
2. A number of Owners can own an Airplane.
3. An Airplane can have a number of flights (One to Many relation).
4. Many Passengers can reserve many seats on a Flight(Many to Many).
5. There are only 4 types of people Passenger,Pilot,Flight Controller, and Service Staff.
6. One Airplane can be served by N number of Service staff.

Furthermore, we also had to keep in mind the basic crux of how all these entities will interact with each other and drawing relationships(*M-N, M-1,1-M)* resulted in a more comprehensive database design.

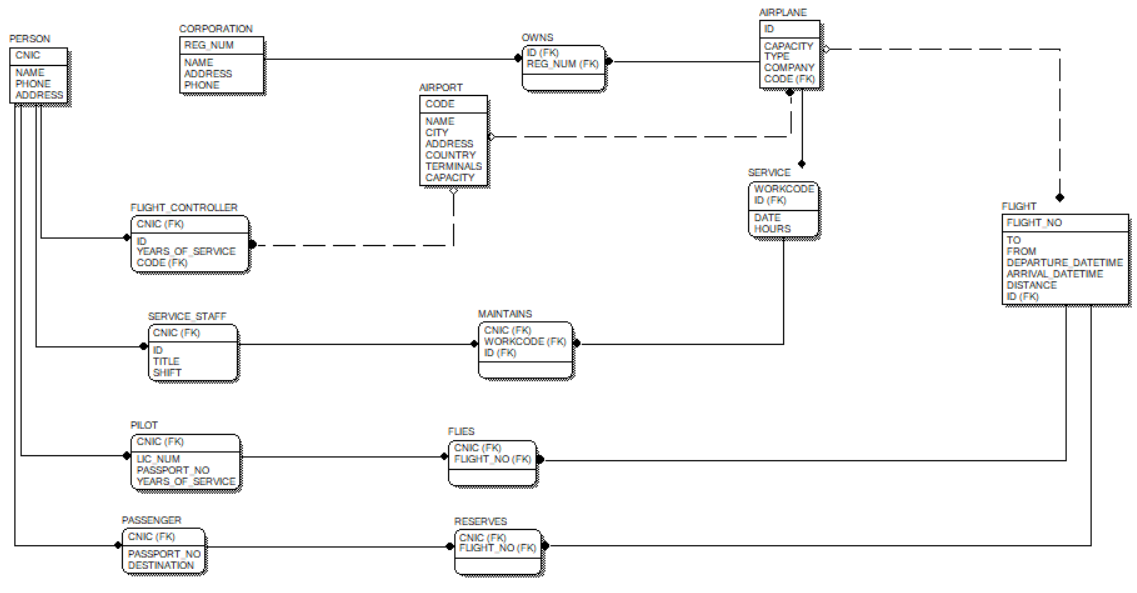


Fig.1. Database Design on ERWIN

We then converted this database design to our SQL Scripts to implement as a back-end server on SQL Server.

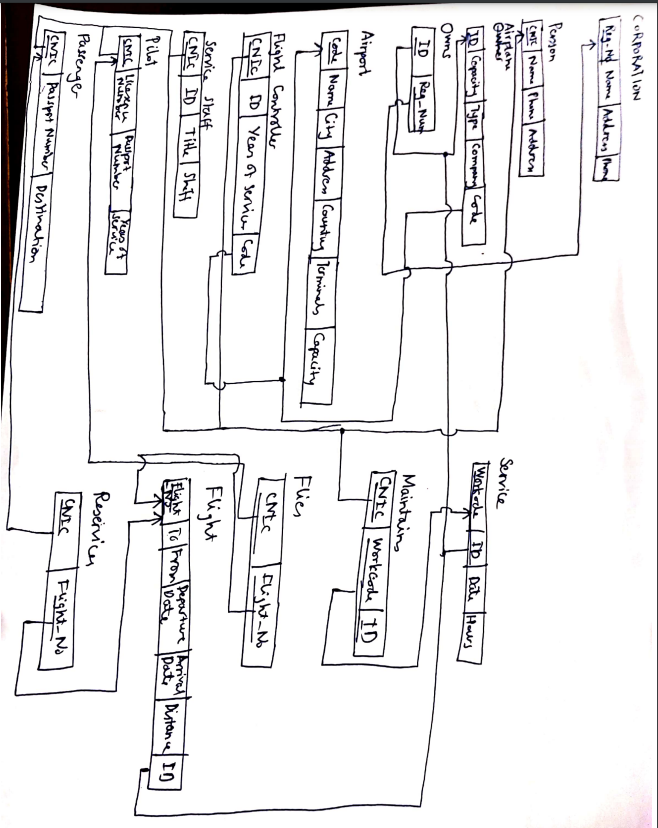


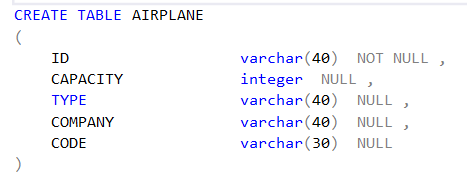
Fig.2. Relational Mapping

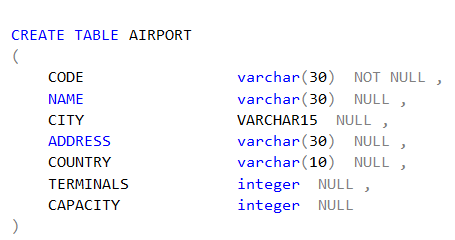
Essentially our systems covers every services that unifies and simplifies the work of either an Airline Booking System or an Airport Staff Management System by doing the following:

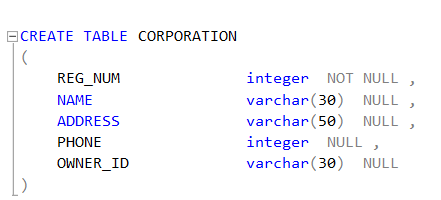
1. Creating a database with help of EER to Relational Mapping. This was done by using the “Forward Engineering” method built in ERWIN.
2. Using the provided DDL commands to *CREATE* all the necessary tables.
3. Creating DML based *ALTER* commandsto update necessary data by using CASCADING updating or deletion.
4. Creating DML based *INSERT* commands to run our necessary data into the tables. We then use all the *Primary Key* and *Foreign Key*  constraints to make our data less redundant. Furthermore, the division of data is done so as to not need extensive *Normalization.*
5. We move on to implementing*STORED PROCEDURES* for individual task execution.
   1. *getFlightDetails*: This allows us to input a certain destination, current location, date of arrival and departure. This allows us to cross examine either all available flights to a location, all flights from a location, all flights regardless of anything, all flights from a specific location to a specific destination, and all flights a specific location to a specific destination on a specific day.
   2. *addNewPassengers:* This allows us to input a new person into the *PERSON* and *PASSENGER* tables.
   3. *BookingFlight:* This allows us to book a specific flight on a specifc flight number by using the name of the person that is booking this flight. If we book it properly then our procedure extracts the CNIC of the person who’s name for example is “Rashid” and then inputs the values of Flight Number and CNIC into the *RESERVES* table.
   4. *CurrentPassenger:* To get a list of passengers from a specific flight we join both *RESERVES* and *PASSENGERS* to get the name of the people on that flight.
   5. *addNewFlight:* This allows us to create a new flight using all the data provided, and using it to insert into *AIRPLANE* and *FLIGHT.*
   6. *getFlightController:* To get a list of available flight controllers in the database.
   7. *getCorporationName:* To get the name of the Corporation that owns the Airplane with that ID.
   8. *addNewCorporation:* This adds a new Owner
   9. *addNewAirport:* This adds a new Airport
   10. *addNewAirplane:* This adds a new airplane using data from an already available airport.
   11. *getServiceStaff:* Gets a list of service staff working on a specific plane
   12. *getPilot:* Gets Pilot data to fly a specific flight.
   13. *addNewPilot:* Adds a new pilot to the pool of people.
   14. *addServiceSchedule:* Add a new service for a specific plane and assigns people in that city.
6. We then implemented our front-end design using React.js that allowed us to configure our services using basic HTML/CSS formatting to create necessary boxes for data visuals and data insertion i.e. if data is imported showing data in tabular format.
7. Next we used Express.js that is a library in the framework of Node.js to configure connections to our SQL Server. This allowed us to use SQL Scripting embedded in our Node.js to CRUD(Create,Read,Update and Delete) anything.

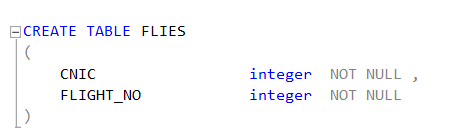
Our Design was implemented on two main fronts. Back-end SQL Server and Front-End React app.

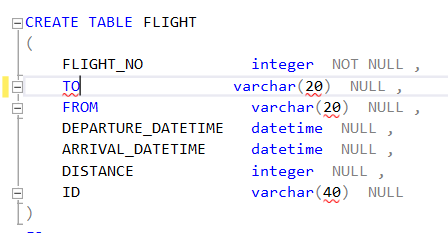
For implementation we had to create tables as stated:

**

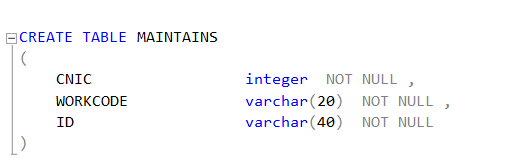
**

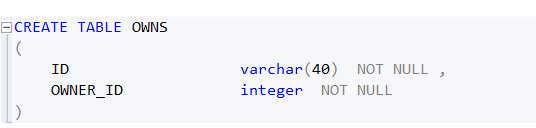
**

**

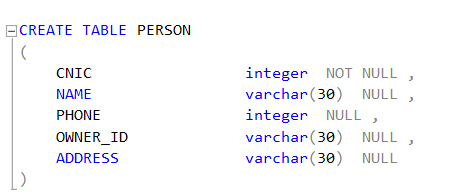
**

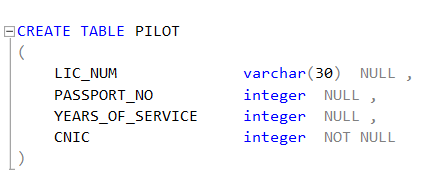
**

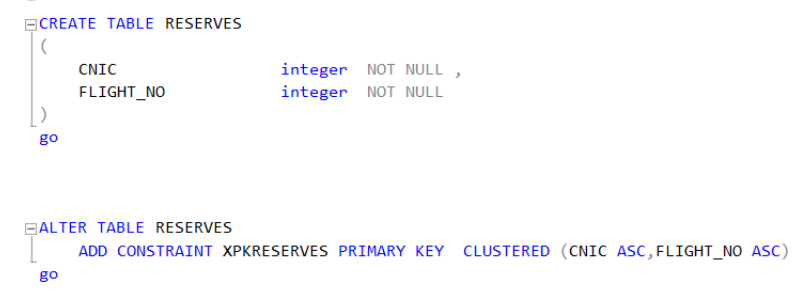
**

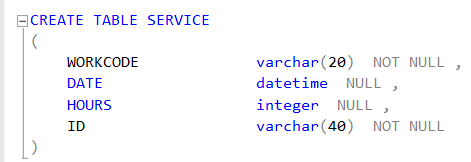
**

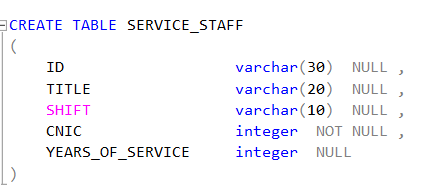
**

**

**



**

**

List of all the tables that have been created.

Now we intend to take into account all the data that has been inserted:

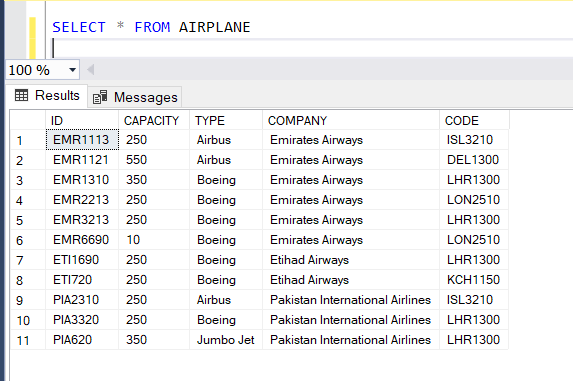


Fig.3. Airplane Data



Fig.4. Airport Data

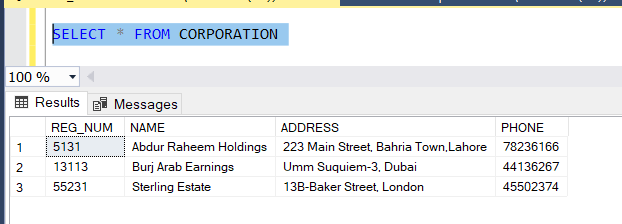


Fig.5. Corporation Data

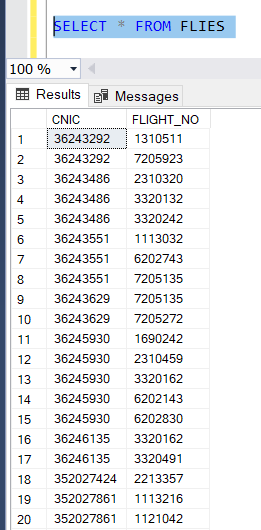


Fig.6. Flies Data(Contains Pilot and Airplane credentials)

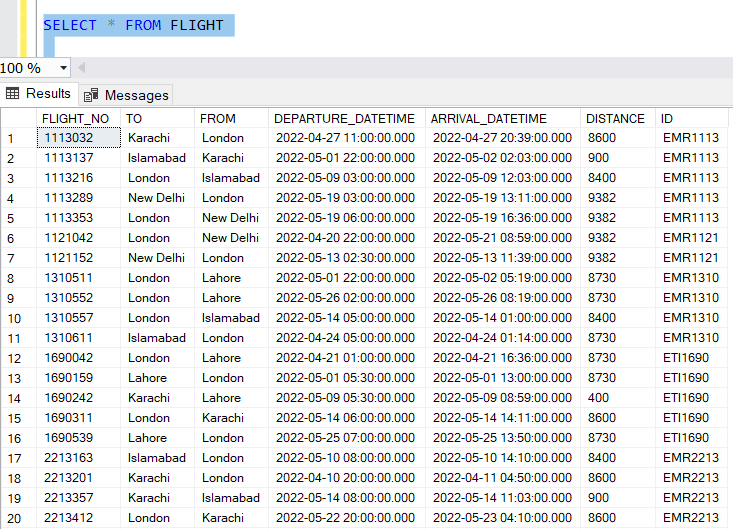


Fig.7. List of Flights

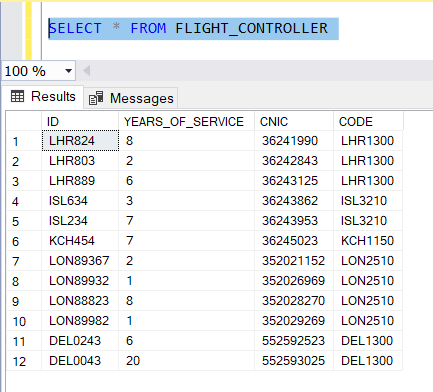


Fig.8. Flight Controllers Data

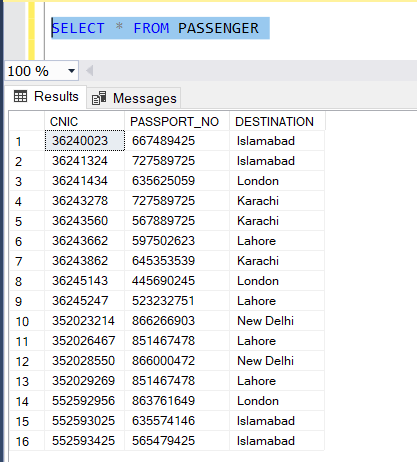
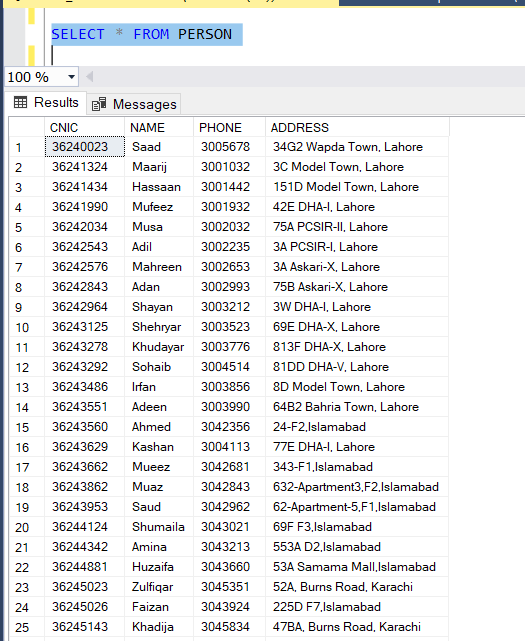


Fig.9. Passengers Data



Fig,10 People Data(Contains both Workers and Passengers)

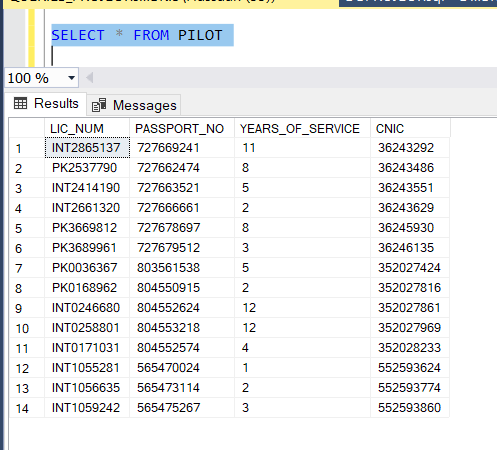


Fig.11. List of Pilots

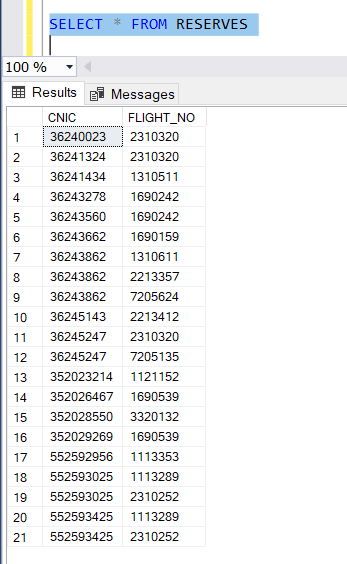


Fig.12. List of Reservations

We then move onto showing how our outputs for our stored procedures will look like:

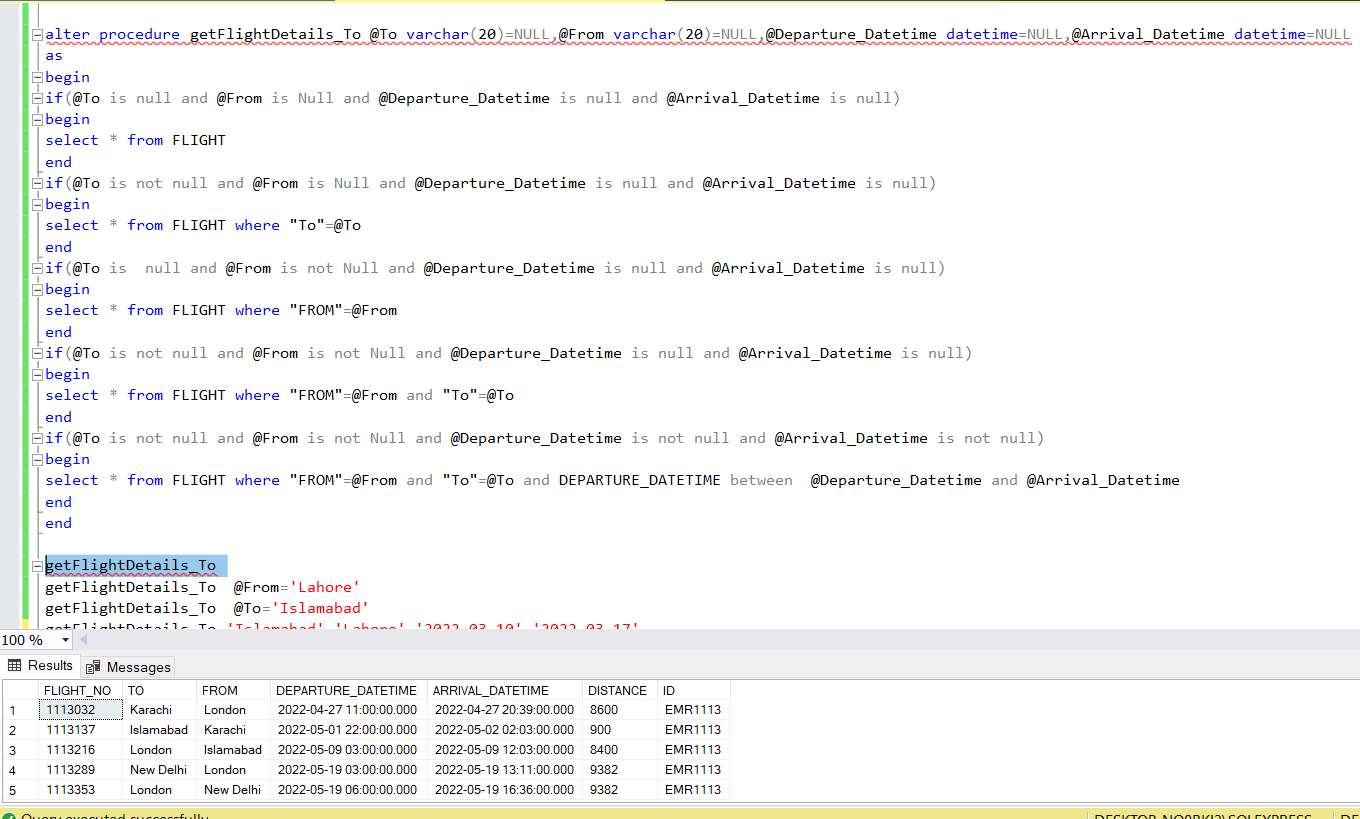


Fig.13. Gets list of all Flights

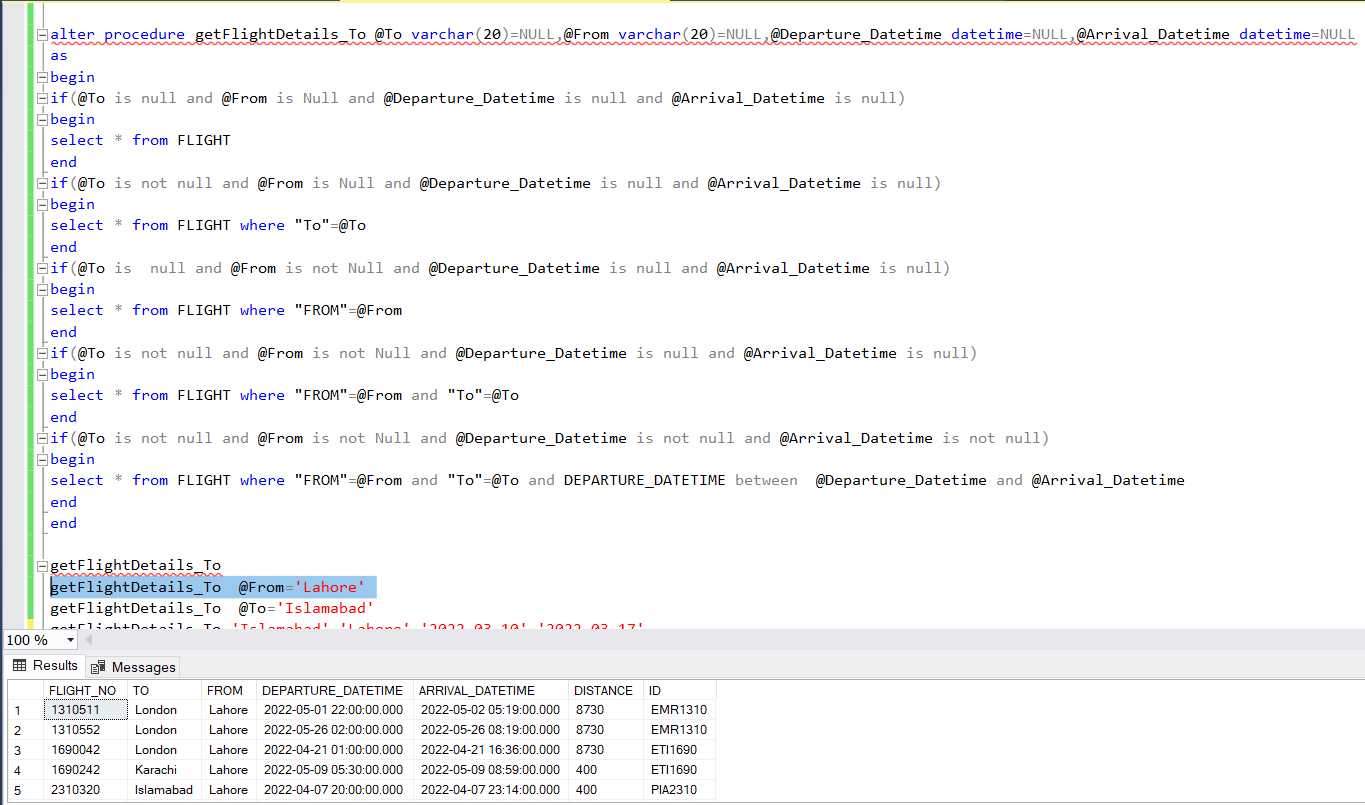


Fig.14. Gets list of Flights leaving from Lahore

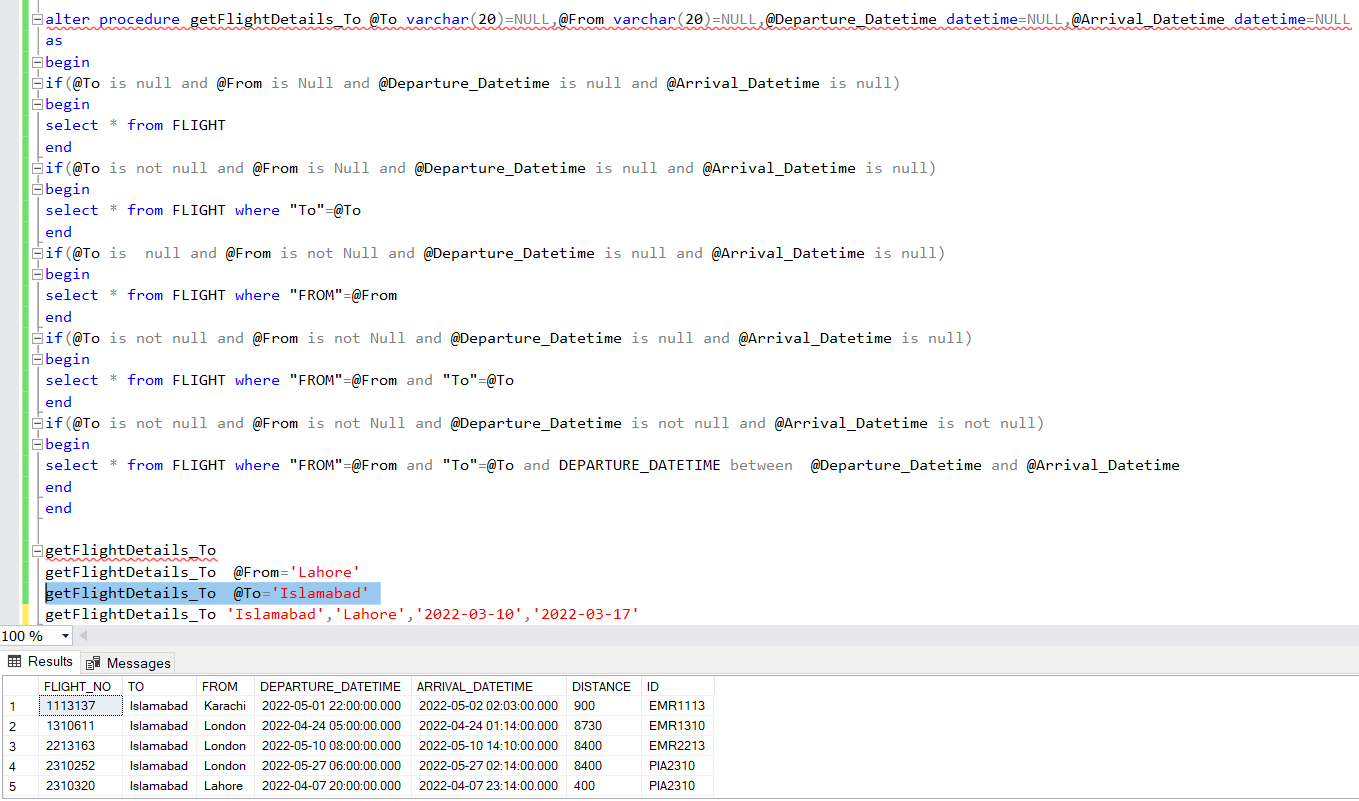


Fig.15. Gets list of flights to Islambad

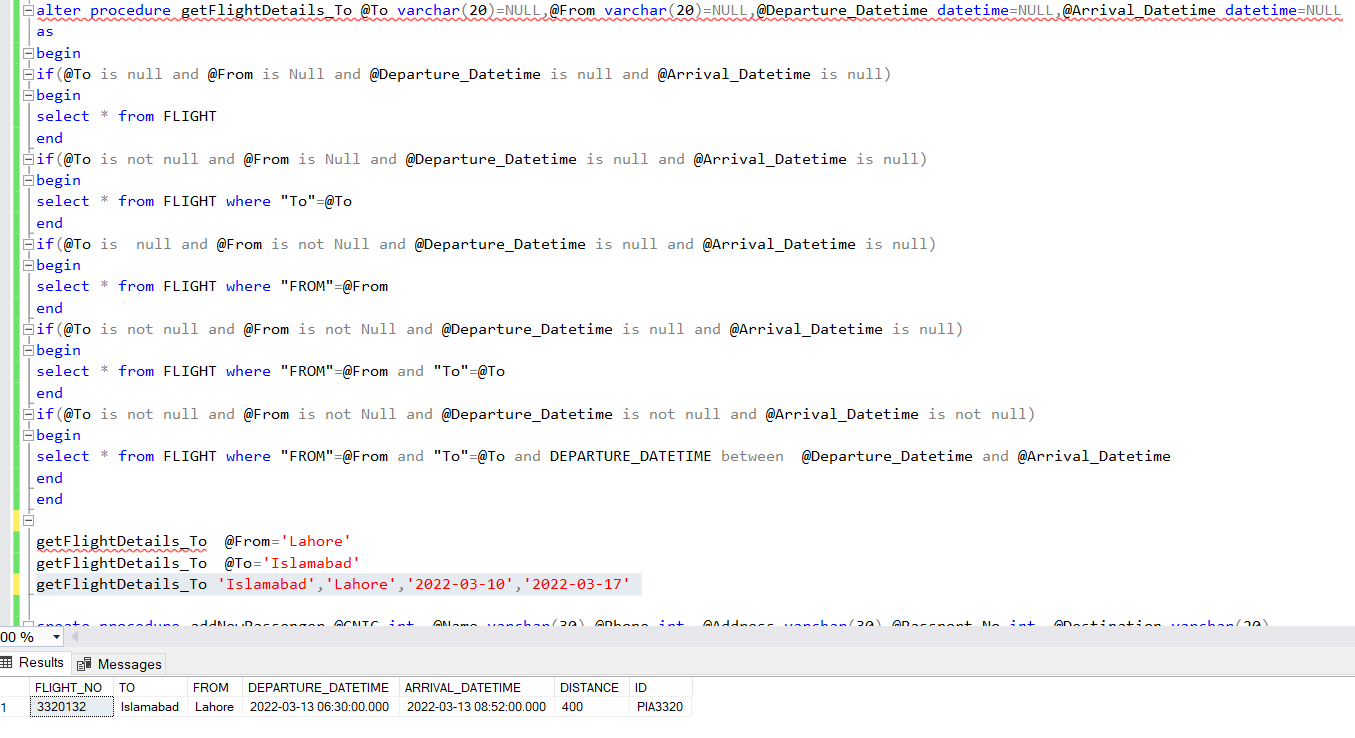


Fig.16. Get list of flights from Lahore to Islamabad between the 10th and 13th of March

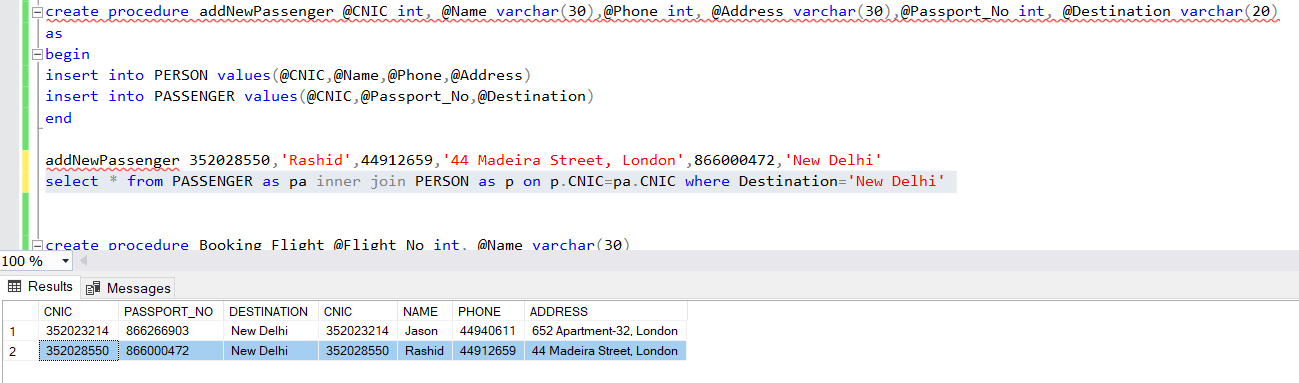


Fig.17. Rashid has been added to the database

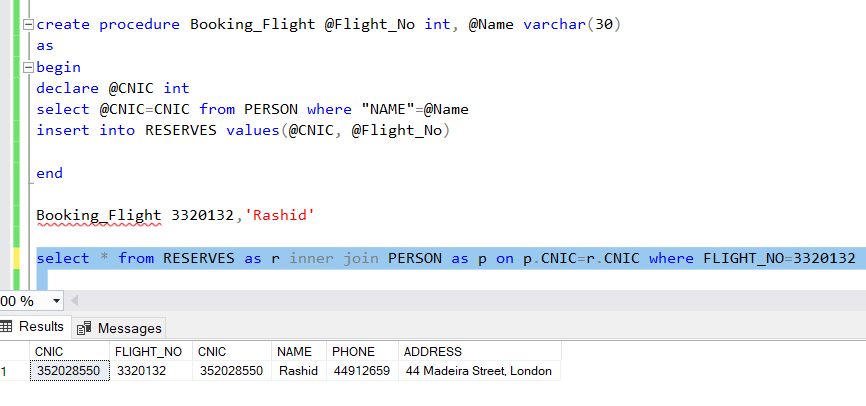


Fig.18. Rashid reserves Flight\_NO 3320132

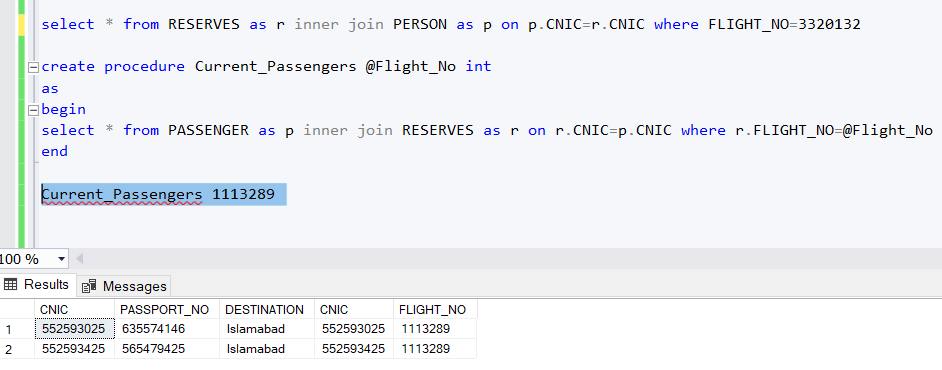


Fig.19. List of current Passengers on Flight 1113289

Similarly, we create similar procedures for all the ones stated at the start of this section, and repeat them continuously to get them into our front-end.

The front-end will depict data in this manner:

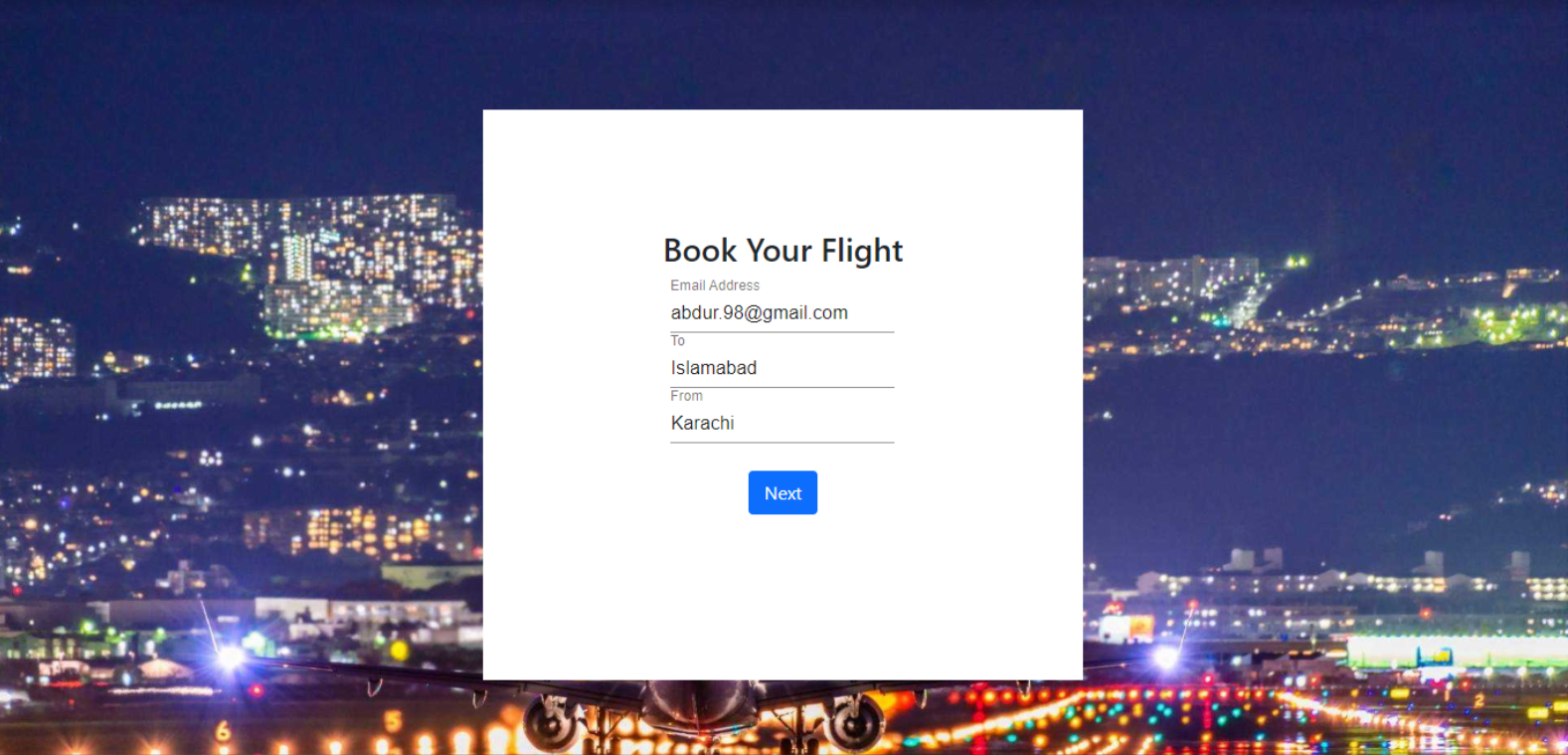


Fig.20. Front-End for Showing Available Flights

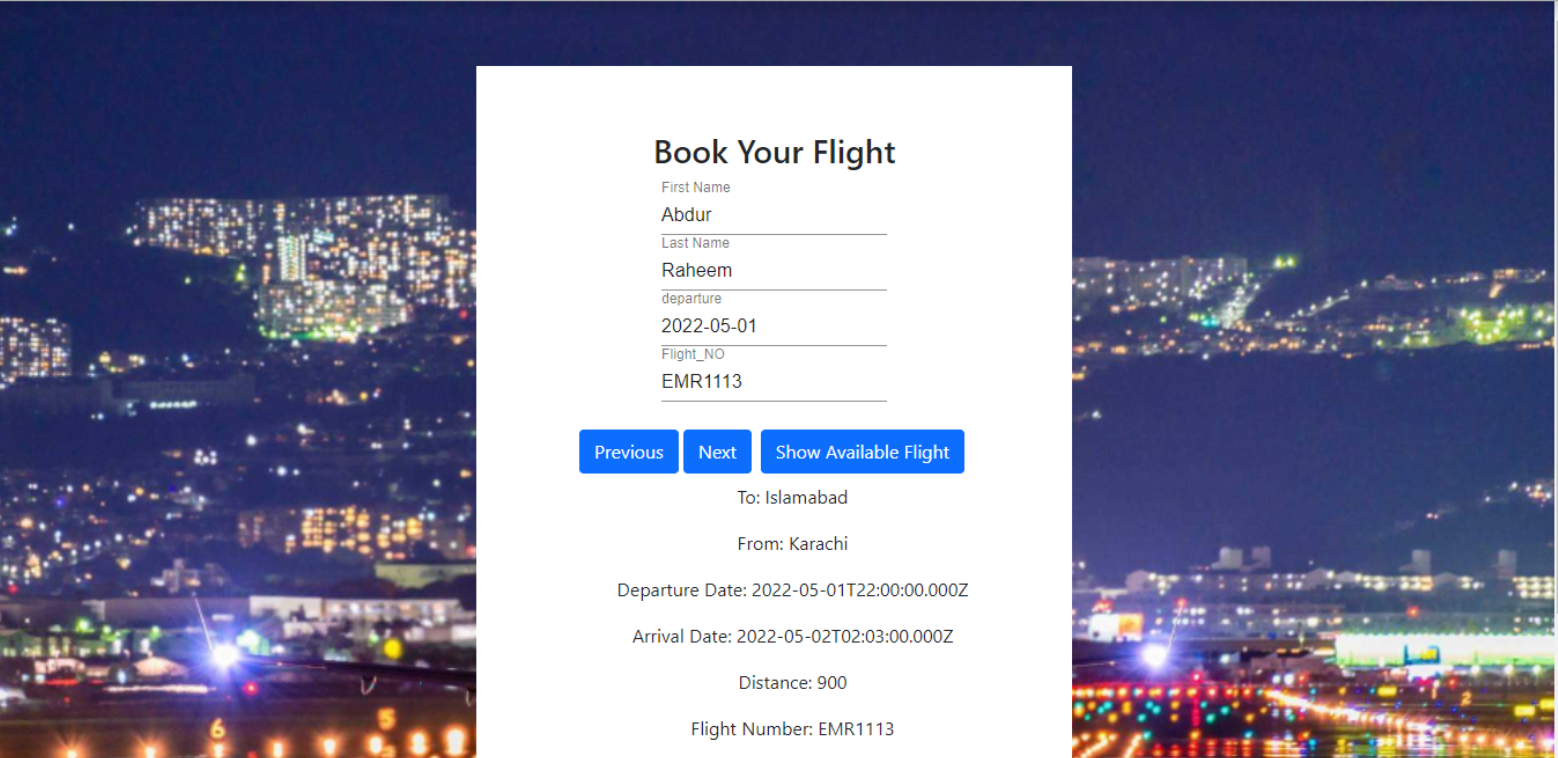


Fig.21. Showing Options for Passengers to book a flight.

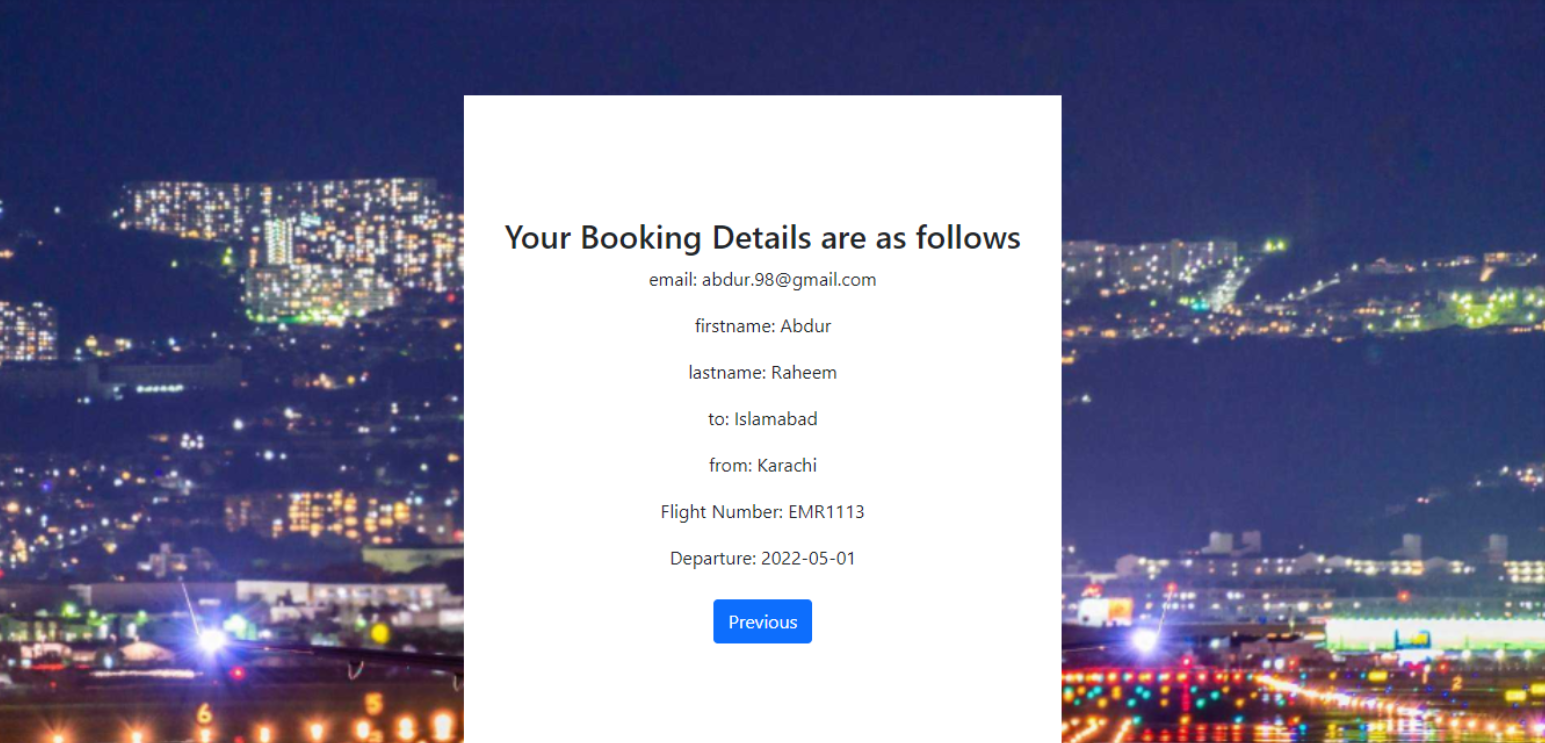


Fig.22. Successfully booked.

We already have implemented only a segment of the portion that was intended to be implemented on basis of overall design of EER Diagram. However, the system could only be categorized in terms of depicting data with respect to “Passengers” not in regards to “Administrators” i.e. in our design we could only CRUD on Passengers, Read essential flight details. This was seen as high performance in contrast to our previously designed over-encompassing design.

Everything is being automated and moved from the filing system to online databases on the cloud.

These cloud platforms include Azure, MongoDB, Firebase, and Oracle.

On-premises database management system (DBMS) revenues continue to decrease as cloud DBMS grows in popularity as the standard. As DBMS deployments are built increasingly on cloud-native or cloud-first architecture, it is now accepted that the future of DBMS is in the cloud.

This project can be implemented for a bigger database and be used for other real-life entities

RDBMS allow for essential saving od space and time as compared to conventional filing systems. It is used for storing essential data in an organized manner that can scale day to day activities and transactions such as production exponentially. However, SQL databases are also being replaced by NoSQL databases that provide

an even better space and time complexitiy.

The Airport Management System we implemented provided the basic functionalities available in any other major Management System that covers every service that provides an ease of connectivity in airport staff and passengers by controlling the features of airport management, details of workers, to scheduling administration jobs to depicting flights for a passenger to board. As a result we could easily configure this solution and create a necessary design for others to use.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SN | Attribute | Complex Activities | Project Mapping | Evaluator’s Comments |
| 1 | **Range of resources** | Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies). | The database developed would need a DBMS along-with a variety of diverse database objects (views, stored procedures, triggers etc). Students can use any DBMS to write the SQL. Also, choose a suitable front end technology compatible with the DBMS to design a fully functional website. |  |
| 2 | **Level of interaction** | Require resolution of significant problems arising from interactions between wide-ranging or conflicting technical, engineering or other issues. | The website developed would be a 3-tier architecture requiring a database at the back end connected with the application layer at the front end (choose any open end or back end). The client tier would make use of browser level technologies. The 3 layers of software would need to interact properly with each other for the system to work. |  |
| 3 | **Innovation** | Involve creative use of engineering principles and research-based knowledge in novel ways. | The website designed would test the creativity of the students in terms of novelty of design and ease of use. |  |
| 4 | **Consequences to society and the environment** | Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation. | If the mini-world for which the website is developed is chosen wisely and the front end is user friendly, can be used by the students later on to start an online business of their own (facebook etc started like that). |  |
| 5 | **Familiarity** | Can extend beyond previous experiences by applying principles-based approaches | Students have to make sure that their solution is resistant to hacking, not being covered in the course, and they try to solve the problem based on the previous experiences and principle based approach. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| SN | Attributes | Description | Evaluator’s Comments |
| - | Preamble | Students would analyze the system requirements of a domain of their own choice, design an electronic database for it and then implement a user friendly functional website underpinned by the database. |  |
| 1 | Range of conflicting requirements | N/A |  |
| 2 | Depth of analysis required | From an engineer’s perspective students were to analyze the system requirements and design an EER diagram (high level conceptual data model) for the database that would underpin their website. |  |
| 3 | Depth of knowledge required | Using their knowledge of EER data model to relational model conversion students were to create the relational model and then apply their knowledge of SQL to create this database using a DBMS. Students were to explore additional features of the DBMS in order to make their solution as robust as possible. |  |
| 4 | Familiarity of issues | Students were to make their website that is resistant to popular hacking techniques (e.g. SQL Injection) as possible. |  |
| 5 | Extent of applicable codes | Students were to evaluate their database tables and state the normal form in which each table lies, that corresponds to the standard codes of practice use in the industry. |  |
| 6 | Extend of stakeholder involved and level of conflicting requirements | The client/teacher was the stake holder and students have to get their document signed from them. |  |
| 7 | Consequences | The website could be used later on by the students to start their own online business. |  |
| 8 | Interdependence | The website so developed would be a 3-tier architecture involving interaction between the database, application and client layer. The 3 layers of software were to connect with and interact with each other without issues. |  |